Universidad de las Ciencias Informáticas

Facultad 3

Registro de Importadores y Exportadores para el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba

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

Autoras:

Kety Raquel Torres Ponce

Leydis Medel Tirador

Tutores:

Ing. Reinier Silverio Figueroa

Ing. Julio Cesar Prieto Alvarez

Junio, 2014

“Se puede adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida.”

DECLARACIÓN DE AUTORÍA

Se declara que Kety Raquel Torres Ponce y Leydis Medel Tirador son las únicas autoras del presente trabajo y se autoriza a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autoras:

_____ _____

Kety Raquel Torres Ponce Leydis Medel Tirador

Tutores:

_____ _____

Ing. Reinier Silverio Figueroa Ing. Julio Cesar Prieto Alvarez

Primeramente agradecer a mis padres y abuelos por darme todo el amor del mundo, el apoyo y la educación que hoy poseo.

A mis hermanos por hacerme saber que siempre están ahí para mí y apoyarme también siempre que lo necesitaba.

A mis sobris por ser las cositas que más quiero en esta vida y las únicas capaces de alegrarme en todo momento, especialmente siempre que tengo un mal día, jejeej.

A mi pareja por todo el apoyo y amor brindado y por estar junto a mí en las buenas y en las malas, gracias también por todo.

A mis suegros, por toda la confianza, amor, cariño y dejarme ser parte de su familia, y a mis cuñis y sus familiares.

A toda mi familia de manera general por estar siempre ahí atentos y preocupados.

A todos mis compañeros por pasar todos estos años maravillosos, por estar todos ahí siempre que lo necesitaba, a Dayana Donatien pese a que hoy no se encuentra con nosotros, a Talía, Rancel, Katerin, Ernesto, Etna, Pupitín, Rey, Aliesky, Miri, Francia, Rosana, a todos….

A mi compañera de tesis, por todo el trabajo y esfuerzo realizado para lograr terminar con éxito nuestra tesis.

A mis tutores Reinier Silverio Figueroa, más que tutor, mi gran amigo durante estos cinco años y a Julio Cesar Prieto Alvarez mi otro tutor y además jefe de proyecto.

A todos mis profesores en estos cinco años, en especial la profe Danaysa, gracias por toda la ayuda ofrecida, los consejos y conocimientos transmitidos.

A todos los profes del proyecto SIRECC, y demás compañeros, por apoyarme y ayudarme cuando lo necesitaba; Orlando, Ana Cecilia, Francisco, Alain, Juan David, Anchelino, Pastor, Leo, Suse, Nai, Allens, Carlitín, a Yoe, que aunque no es del proyecto, es como si lo fuera, jeeje…

A los compañeros del proyecto Portales, en especial Yordanis y a los de Calidad gracias también por su gran ayuda.

A todos los especialistas de la Cámara de Comercio, en especial Susana, por su atención, apoyo y dedicación.

A todos los que de una forma u otra han contribuido en mi carrera, gracias por todo.

Kety

Agradezco a mis padres Gladys y Angel por la manera en que me educaron, por la comprensión y el apoyo que me brindaron durante toda mi vida. Por ser un ejemplo a seguir, por darme tanto amor y confiar en mí incluso en momentos que ni yo sabía que podía confiar, por perdonar mis faltas, por llorar mis derrotas y disfrutar mis victorias, por todo eso y más, les agradezco con todo mi corazón y que sepan que estoy muy orgullosa de tenerlos y espero que se sientan orgullosos de la hija que han logrado.

A mi hermano Angelito por apoyarme y comprenderme durante toda nuestra existencia.

A mis tías Bárbara e Isabel por preocuparse tanto por mí y ayudarme siempre que las necesitaba.

A mi abuela Dora por ser la mejor abuela del mundo y tener un corazón tan grande que no cabe en el pecho.

A mis queridísimos primos Roselí, Raidel, Pedrito y en especial a Kusy por preocuparse tanto por mí y enseñarme tanto de la vida.

A mi madrina Xiomara que la voy a recordar y querer durante toda mi vida por apoyarme y ayudarme tanto.

A Yuyi por ser además de mi novio mi amigo, comprenderme durante 5 años y no dejarme flaquear en momentos en que ya no podía más.

A mi tutor Reinier por tener tanta paciencia y dedicación.

A mi compañera de tesis por el trabajo realizado.

A las personas del proyecto SIRECC Julio, Danaysa, Anita, Orlando, Francisco, Juan David, Alain, Anchel, Allen, Leandro, Carlos, Ernesto y Susel por ayudarme cada vez que lo pedía.

A mi hermano del alma Yoendry que hemos compartido tantos momentos buenos y malos durante la carrera, por guiarme, comprenderme, ser un ejemplo para mí y enseñarme tantas cosas que es imposible mencionar.

A Naylín y Etna por ser mis amigas en todo momento y hacerme saber que puedo contar con ellas para lo que sea.

A mis compañeros de aula y a todos los que de una manera u otra contribuyeron con mi carrera, muchas gracias.

Leydis

DEDICATORIA

"Dedico este trabajo a toda mi familia, especialmente a mis padres, hermanos y sobrinas, además a mi pareja y amistades".

Kety

"Dedico este trabajo a mis padres por ser las personas más importantes de mi vida."

Leydis

RESUMEN

Para informatizar la sociedad cubana, como parte de la Batalla de Ideas, surge la Universidad de las Ciencias Informáticas (UCI). Esta cuenta con un Centro de Gobierno Electrónico (CEGEL) ubicado en la Facultad 3, al que se le asignó la tarea de informatizar los procesos que en la Cámara de Comercio de la República de Cuba se realizan: registros de empresas cubanas, sucursales extranjeras, agencias de viajes, trámites migratorios e importadores/exportadores. Para dar cumplimiento a este objetivo se creó el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC), aplicación de escritorio estructurada en cinco módulos para la gestión de los procesos mencionados anteriormente. Actualmente en la Cámara de Comercio de la República de Cuba el proceso de registro de empresas importadoras/exportadoras se realiza de forma manual, se trabaja con grandes volúmenes de información en forma de archivos físicos, lo que trae consigo difícil acceso a la documentación y demora en el proceso de registro. Para dar solución a estos problemas se decide desarrollar un módulo, no sin antes detallar la selección del modelo de desarrollo a utilizar, y a partir de este, las herramientas y tecnologías, así como los patrones de diseño y estilos arquitectónicos. El resultado introducido de la investigación, contribuye a mejorar la disponibilidad de la información en el proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba.

PALABRAS CLAVES

Disponibilidad, exportadores, importadores, SIRECC.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En los últimos años han tenido lugar vertiginosos avances en el ámbito de las Tecnologías de la Información y las Comunicaciones (TIC). Independientemente de estos avances, sigue existiendo preocupación por incrementar el uso y el aprovechamiento de las tecnologías y con ello, conformar nuevos paradigmas donde la sociedad sea la principal beneficiada. En general, se ha podido observar cómo existe un auge en cuanto al uso de las TIC, posibilitando contar con diversas aplicaciones de estas tecnologías y facilitando a las personas la posibilidad de tener un mayor acceso a la información de cualquier parte del mundo.

El desarrollo de las TIC ha incorporado nuevas formas de interacción en la sociedad a través del gobierno electrónico, el cual prácticamente en todos los países cuenta con programas nacionales o líneas de acción gubernamentales. Estos programas buscan incentivar el acceso generalizado entre todos los individuos a este tipo de tecnologías. "*Se entiende al gobierno electrónico como el uso de las TIC en los órganos de la administración para mejorar la información y los servicios ofrecidos a los ciudadanos, orientar la eficacia y eficiencia de la gestión pública e incrementar sustantivamente la transparencia del sector público y la participación de los ciudadanos"*. (Montenegro, 2013)

Las cámaras de comercio son ejemplos de entidades del gobierno que se han sumado a la utilización de las TIC. La Cámara de Comercio de la República de Cuba es una entidad que promueve las ofertas exportables de productos y servicios, así como las oportunidades de negocios e inversión de la empresa cubana y la sustitución de importaciones, en beneficio de la economía nacional. Presta servicios a sus asociados, a las sucursales de compañías extranjeras acreditadas en el país, así como a empresarios extranjeros interesados en negociar con Cuba. De manera general constituye una herramienta para la reinserción de la economía cubana en el mundo de las relaciones económicas internacionales, pues potencia e intercambia información valiosa entorno a las posibilidades de negocios a escala mundial. (Cámara de Comercio, 2009)

Como parte de la Batalla de Ideas; estrategia llevada a cabo con el fin de mejorar la calidad de vida de las personas, ampliar las oportunidades educacionales para el pueblo cubano y aumentar el acceso a la cultura, para informatizar la sociedad cubana, surge la Universidad de las Ciencias Informáticas (UCI), que presenta como misión:

- ✓ *"formar profesionales comprometidos con su Patria y altamente calificados en la rama de las Ciencias Informáticas. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática"*. (UCI, 2012)

La UCI cuenta con el Centro de Gobierno Electrónico (CEGEL) ubicado en la Facultad 3, el cual se dedica fundamentalmente al desarrollo de soluciones informáticas para la gestión de los procesos legales, registrales, notariales, de gestión de gobierno y servicios jurídicos a la ciudadanía y generan documentos electrónicos que por necesidad deben poseer valor probatorio ante la ley. (CEGEL, 2013)

A CEGEL se le asignó la tarea de informatizar los procesos que en la Cámara de Comercio de la República de Cuba se realizan, para dar cumplimiento a este objetivo se crea el proyecto SIRECC, el cual contará con los módulos: Afiliación de Empresas Cubanas, Sucursales Extranjeras, Agencias de Viajes, Trámites Migratorios, Reportes y Registro de Importadores y Exportadores, este último constituye el objeto de la presente investigación.

En la entrevista realizada a la Lic. Susana Enríquez Domínguez, Jurista Internacional de la Cámara de Comercio (*Anexo 1. Entrevista realizada a especialista de la Cámara de Comercio de la República de Cuba)*, se constató que actualmente el proceso de registro de empresas importadoras/exportadoras se realiza de forma manual, lo que conlleva a pérdida de datos y deterioro de los documentos que llevan almacenados gran cantidad de tiempo. Se trabaja con grandes volúmenes de datos en forma de archivos físicos por lo que la información que se gestiona enfrenta problemas de difícil acceso, y se dificulta la consulta de empresas específicas, convirtiéndose en un proceso engorroso y lento.

Por otra parte, los funcionarios demoran de una a dos horas en la realización de un trámite para el registro de una empresa importadora/exportadora y en períodos de ratificación sólo veinte empresas son procesadas diariamente, lo cual afecta la cantidad de personas a atender y la satisfacción de los clientes.. A partir de la situación problémica descrita anteriormente se identifica el siguiente **problema de la investigación**:

¿Cómo gestionar la información en el proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba de manera que contribuya al aumento de la disponibilidad de la información?

Se presenta como **objeto de estudio**: proceso de desarrollo de soluciones informáticas para el gobierno electrónico.

Para dar solución al problema planteado se identifica como **objetivo general**: desarrollar el Módulo Registro de Importadores y Exportadores para SIRECC que contribuya al aumento de la disponibilidad de la información en el proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba.

A partir del objetivo trazado se determina como **campo de acción**: informatización del proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba.

Para facilitar el cumplimiento del objetivo general tributan los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación estableciendo las fundamentaciones necesarias que sirvan como base para el presente trabajo.
- ✓ Definir el modelo de desarrollo, así como las diferentes herramientas y lenguajes a emplear en la solución.
- ✓ Desarrollar el Módulo Registro de Importadores y Exportadores para el proyecto SIRECC.
- ✓ Realizar la valoración de la solución propuesta para evaluar el cumplimiento de los objetivos planteados en la investigación.

Como **tareas de la investigación** se definen:

- ✓ Estudio y análisis de los procesos de registro de empresas importadoras/exportadoras de la Cámara de Comercio de la República de Cuba.
- ✓ Análisis del estado del arte de los sistemas existentes que informatizan procesos de registro de empresas importadoras/exportadoras.
- ✓ Definición de la metodología, herramientas, lenguajes y patrones a utilizar para el desarrollo de la solución.

- ✓ Diseño del Módulo Registro de Importadores y Exportadores.
- ✓ Implementación del Módulo Registro de Importadores y Exportadores.
- ✓ Aplicación de pruebas de software al Módulo Registro de Importadores y Exportadores.
- ✓ Valoración de las variables de la investigación.

Se precisa como **idea a defender**: con el desarrollo del Módulo Registro de Importadores y Exportadores para SIRECC se contribuirá al aumento de la disponibilidad de la información en el proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba.

Operacionalización de las variables de investigación:

		Dimensiones	Indicadores	Forma de valorar
Variable Dependiente	Disponibilidad de la Información	Personal autorizado	Acceso a la Información	Entrevista y Mecanismos de seguridad brindados por el marco de trabajo Kairos (niveles de acceso, autenticación)
		Tiempo requerido	Tiempo de respuesta de las consultas realizadas (TR)	Entrevista y TR<3s (segundos), a partir del RnF 7
Variable Independiente	Módulo Registro de Importadores y Exportadores	Alcance	Porciento de requisitos funcionales implementados (PR)	PR = CRI/TR CRI: cantidad de requisitos funcionales implementados TR: total de requisitos funcionales
		Calidad	Calidad interna	Pruebas internas
			Calidad externa	Pruebas de liberación

Tabla 1. Operacionalización de las variables de investigación

Como **métodos científicos de la investigación** se emplearon los que a continuación se describen, propuestos en (Hernández León, y otros, 2011):

Métodos teóricos:

- ✓ Método analítico-sintético: con el uso de este método se realizó un profundo análisis de los documentos, las publicaciones y bibliografías consultadas para el desarrollo de la investigación, posibilitaron hacer una síntesis de estas principales fuentes y llegar a conclusiones parciales sobre el tema en estudio. Además se analizó información relacionada con las cámaras de comercio, haciendo énfasis en el proceso de registro de empresas importadoras/exportadoras.

- ✓ Modelación: Este método es uno de los más importantes en el campo de la construcción de software y se utiliza en el diseño de modelos de datos, diagrama de clases, de componentes y otros. Permitió realizar una reproducción simplificada de la realidad, abstracciones, con el objetivo de explicarla y comprender mejor los procesos de negocio de la presente investigación.

Métodos empíricos:

- ✓ Entrevista: este método se utilizó para recopilar información referente al proceso de registro de empresas importadoras/exportadoras en la Cámara de Comercio de la República de Cuba. Con las entrevistas realizadas se presentan evidencias, argumentos y conocimientos más bastos y profundos sobre el proceso. Se llegaron a conclusiones parciales, permitiendo argumentar aún más la situación problémica, el uso de las variables de la investigación y en un final una valoración de estas.

Estructura del documento:

El presente trabajo consta de tres capítulos, estructurados de la siguiente manera:

- ✓ **CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA**: en el capítulo se abordan los aspectos teóricos relacionados con la investigación, tendencias actuales, modelo de desarrollo, tecnologías y herramientas a emplear en la construcción de la solución, lenguajes utilizados, así como la teoría referente a los patrones de diseño y estilos arquitectónicos.

- ✓ **CAPÍTULO 2. PROPUESTA DE SOLUCIÓN**: se enfatiza en la parte técnica de la solución, se presentan los diferentes artefactos que se obtienen del diseño realizado. También se detalla la implementación desarrollada a partir del diagrama de clases, el modelo de datos, diagrama de componentes y en general toda una descripción del diseño e implementación. Se realizan diferentes pruebas de software al producto para así contribuir a la máxima calidad posible. Además se valida el diseño realizado al módulo, aplicando diferentes métricas.

- ✓ **CAPÍTULO 3. VALORACIÓN DE LA SOLUCIÓN**: se realiza la valoración correspondiente a las variables del problema de la investigación, atendiendo a dimensiones e indicadores. También se explican y describen las pruebas de liberación por parte de CEGEL y CALISOFT, y las pruebas de aceptación aplicadas al producto, puntualizando finalmente los resultados obtenidos en cada una de las pruebas aplicadas.

FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se abordan los aspectos teóricos relacionados con la investigación, las herramientas, lenguajes y tecnologías que se emplean en la construcción de la solución y aspectos esenciales que sirvan de soporte a la misma, teniendo en cuenta: lenguajes utilizados, herramienta CASE, sistema gestor de base de datos, marco de trabajo, Entorno de Desarrollo Integrado y servidor de aplicaciones. Se realiza una descripción de los principales procesos que presenta el Registro de Importadores y Exportadores. Se define además, el modelo de desarrollo a utilizar basado en el Programa de Mejora propuesto por la UCI, se describe un conjunto de patrones de diseño y estilos arquitectónicos, así como las pruebas de software que se realizaron una vez concluida la implementación.

1.2. Descripción de los procesos de negocio

La Cámara de Comercio de la República de Cuba actualmente realiza diferentes actividades y gestiones enmarcadas en el contexto comercial de las empresas importadoras/exportadoras inscritas a ella. Los procesos identificados que constituyen objeto de informatización son: inscripción, modificación de facultades y cancelación de facultades de una empresa importadora/exportadora. Estos procesos fueron descritos, según lo definido en el *Manual de la Dirección Jurídica*, ubicado en el *Expediente del Proyecto SIRECC*, para más información dirigirse a (Dirección Jurídica, 2010).

Para la inscripción de las empresas importadoras/exportadoras, estas deben estar previamente autorizadas por una resolución emitida por el Ministerio del Comercio Exterior (MINCEX). El encargado del registro revisa la documentación que presenta la entidad directamente ante la Cámara de Comercio y verifica que cumpla con las formalidades establecidas, registra los datos de la empresa y crea su expediente correspondiente. Seguidamente se emite la certificación correspondiente y se envía una notificación a la empresa informando de la disponibilidad de los documentos.

Para el proceso de modificación de facultades, el MINCEX emite una resolución modificando las facultades que fueron concedidas a la empresa. El encargado del registro actualiza la información de la empresa. Por último se notifica a la entidad interesada la disponibilidad de los documentos. En el caso de cancelación de las facultades que le han sido conferidas a una empresa se registra lo así dispuesto en la resolución enviada por el MINCEX.

Con el propósito de informatizar estos procesos en el siguiente epígrafe se estudian sistemas existentes en la actualidad que informatizan procesos relacionados con el registro de empresas importadoras/exportadoras, haciendo énfasis en sus características y funcionalidades.

1.3. Sistemas estudiados.

Para lograr el cumplimiento de los objetivos propuestos y encontrar una solución factible al problema en cuestión, se realizó un estudio centrando la búsqueda en sistemas que realizan procesos relacionados con el registro de empresas importadoras/exportadores. Los sistemas analizados son Pedimento Empresarial, ONE GOAL atmosphere y SENTAI, los cuales son descritos, teniendo en cuenta sus principales características.

1.3.1. Pedimento Empresarial

Es el primer sistema de cómputo en México hecho por especialistas con más de 18 años de experiencia en el desarrollo de software para comercio exterior. Es una aplicación web. Está diseñado para ayudar a las empresas importadoras/exportadoras a crear su propia base de datos con toda la información de sus operaciones aduaneras provenientes de los pedimentos que realizan los agentes aduanales a nombre de las empresas. Como consecuencia de la gran aceptación que ha tenido Pedimento Empresarial en el mercado de las empresas importadoras/exportadoras, el sistema se ha convertido en poco tiempo en una herramienta indispensable para el control de las operaciones de comercio exterior y eficientizar sus áreas contables y fiscales, además ha evitado a muchas agencias aduanales la generación de informes para sus clientes, dándoles un valor agregado en su servicio y sin costo adicional para la Agencia Aduanal. Las empresas importadoras/exportadoras que contratan Pedimento Empresarial pueden crear sus propios reportes sin importar qué agente aduanal despachó la mercancía o en qué aduana se realizó el despacho. El usuario del sistema puede buscar pedimentos por cualquier dato contenido en los mismos, además de obtener fácilmente reportes y estadísticas de acuerdo a sus necesidades. (IMAR Sistemas, 2013)

1.3.2. ONE GOAL atmosphere

A partir del estudio realizado en (ABPRO, 2014), ONE GOAL atmosphere es una aplicación web. Asocia la factura electrónica para exportación a un pedimento de salida para cuidar que las declaraciones en puntos totales cuadren con los ingresos totales. Permite el registro de eventos logísticos nacionales e internacionales para el control exacto de la ubicación de la mercancía. Además posee una serie de beneficios como: maneja almacenes fiscales y en tránsito para el control de los costos fiscales de los productos, integra los gastos totales de importación y los de la agencia aduanal al costo de los productos de forma sencilla y transparente, facilita la conciliación de los anticipos a agencias aduanales y agentes de carga sin importar si las facturas son de diferentes proveedores, comunicación directa a la Ventanilla Única de Comercio Exterior de México (VUCEM) para simplificar procesos.

También emite facturas con información detallada del pedimento en ventas de primera mano, comparte información con socios de negocios o usuarios remotos, identifica y presenta todos los costos indirectos por producto, optimiza tiempos operativos y reduce costos administrativos, deduce impuestos desde que la mercancía está en tránsito, integra un costeo preciso de las importaciones incluyendo impuestos, gastos aduanales y logística nacional e internacional, visualiza la ubicación de las mercancías en tránsito y evita pagos duplicados y conciliaciones erróneas de gastos aduaneros por falta de seguimiento adecuado.

1.3.3. SENTAI

En (Pardo Barrios, y otros, 2009) "*se plantea que la Corporación de Importadores y Exportadores de Cuba (CIMEX) controla sus actividades empresariales con un sistema (SENTAI) automatizado e integrado, orientado a la Gestión Empresarial de Comercio Mayorista. SENTAI es implementado sobre el sistema de base de datos "PROGRESS" y soportado en el sistema operativo "UNIX". Las bases de datos pueden residir en servidores centrales, territoriales o zonales, en correspondencia con las características de las empresas, compañías y otras entidades usuarias del sistema. Por sus características fundamentales, es muy flexible y adaptable a las necesidades específicas de los usuarios que lo explotan*".

Atendiendo a otras peculiaridades, SENTAI es aplicable fundamentalmente a empresas, compañías y entidades en general, que desarrollan actividades comerciales de compra, distribución y ventas mayoristas; y actividades de comercio minorista o detallista, pero puede ser configurable también para

actividades comerciales de gastronomía, actividades de prestación de servicios, de reparación de equipos técnicos en general; por órdenes de trabajo, actividades de transporte de carga automotor, actividades de naturaleza productiva y otras.

1.3.4. Valoración de los sistemas

Para satisfacer las necesidades de la Cámara de Comercio de la República de Cuba, específicamente en el Módulo Registro de Importadores y Exportadores es necesario crear un sistema de escritorio que permita a la entidad registrar y controlar toda la información de las empresas vinculadas a ella y que tengan la facultad de importar, exportar o ambas. Debido a todas estas exigencias el producto final debe permitir registrar los datos de las empresas y otorgarles facultades, modificarlas y cancelarlas, para ello ver *epígrafe 1.2. Descripción de los procesos de negocio*. Además debe permitir realizar búsquedas por diferentes parámetros donde se muestre un listado con las empresas registradas que posean esas características facilitando el acceso a la información, listar registros de empresas inscritas con anterioridad y exportar documentos.

Tomando como referencia el estudio realizado a estas aplicaciones se concluye que no responden a las necesidades de la Cámara de Comercio, ya que permiten a una entidad crear su propia base de datos con toda la información de sus actividades empresariales, es decir permiten a aquellas empresas importadoras/exportadoras llevar un registro específicamente de sus operaciones, como es el caso de Pedimento Empresarial. En cuanto a ONE GOAL atmosphere, es un aplicación web al igual que Pedimento Empresarial, y permite el registro de eventos logísticos nacionales e internacionales para el control exacto de la ubicación de la mercancía, procesos que no son aspectos a tener en cuenta en la informatización de los procesos de negocio de la Cámara de Comercio. Lo mismo sucede con SENTAI, es un sistema aplicable sólo a entidades que desarrollan actividades comerciales de compra, distribución y ventas mayoristas y actividades de comercio minorista o detallista, de gastronomía, actividades de prestación de servicios y de reparación de equipos técnicos en general. En el siguiente epígrafe se define en esencia el modelo de desarrollo de software usado en la investigación, así como las etapas llevadas a cabo.

1.4. Ingeniería de software

Atendiendo a lo planteado en (Sommerville, 2005), la ingeniería del software es una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después que se utiliza. En esta definición, existen dos frases claves:

- ✓ "*Disciplina de la ingeniería: los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde sean convenientes, pero las utilizan de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlo. Los ingenieros también saben que deben trabajar con restricciones financieras y organizacionales, por lo que buscan soluciones tomando en cuenta estas restricciones*."

- ✓ "*Todos los aspectos de producción de software: la ingeniería del software no solo comprende los procesos técnicos del desarrollo de software, sino también con actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software*."

Con el objetivo de organizar el trabajo y guiar al equipo de trabajo durante el proceso de desarrollo y ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto, surgen los modelos de desarrollo de software. En el siguiente subepígrafe se explica la selección del modelo empleado en la presente investigación.

1.4.1. Modelo de desarrollo de software

Para el desarrollo de la investigación se utilizó el Programa de Mejora propuesto por la UCI y a partir de lo plasmado en (CEGEL, 2013), este programa define un modelo para representar los procesos asociados al desarrollo de software y la prestación de servicios de las TIC. "*Presenta un modelo de referencia para el proceso de desarrollo de software en la UCI y permite elaborar una estrategia para la gestión de conocimiento en el marco de un programa de mejora continua*".

El Programa de Mejora define para el ciclo de vida básico de los proyectos las fases: Estudio Preliminar, Modelado del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas Internas, Pruebas de Liberación, Despliegue y Soporte (*Anexo 2. Fases del Programa de Mejora propuesto por la UCI*), dentro

de cada una se genera un conjunto de artefactos necesarios para la construcción de un sistema de software con la calidad requerida. Para el desarrollo de la aplicación se tomaron como referencia las fases Análisis y Diseño, Implementación, Pruebas Internas, Pruebas de Liberación y parte del Despliegue.

Este modelo es orientado a componentes debido a que se desarrolla teniendo en cuenta la estructura de agrupar por componentes, es adaptable al cambio y a las particularidades específicas de cada proyecto; siendo un modelo iterativo incremental que además está centrado en la arquitectura. Todas estas características demuestran que el Programa de Mejora explota los rasgos fundamentales y buenas prácticas de Proceso Unificado de Rational (RUP por sus siglas en inglés).

Como parte de este programa se estructura un expediente de proyecto (*Expediente del Proyecto SIRECC*), abarcando con este un grupo de documentos y artefactos generados en cada una de las fases del modelo, los cuales son necesarios como parte de las buenas prácticas del desarrollo de software. Cada una de estas fases o etapas del modelo de desarrollo son explicadas en el siguiente subepígrafe.

1.4.2. Análisis y Diseño

En el *Proyecto Técnico de SIRECC*, basado en el Programa de Mejora propuesto por la UCI, se detalla que durante la fase de Análisis y Diseño es modelado el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos o necesidades definidas con el cliente. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la implementación. Los modelos desarrollados en esta etapa son más formales y específicos de una implementación. Durante esta fase son desarrollados, diagramas de clases, modelo de datos, entre otros. (Prieto Alvarez, 2013)

1.4.3. Implementación

Con las salidas obtenidas durante la fase anterior se comienza la Implementación, donde a partir de los resultados obtenidos con anterioridad se implementa el sistema en términos de componentes, ficheros de código fuente, scripts, ejecutables y similares. Además se definen los estándares de codificación usados, la aplicación de los estilos arquitectónicos y patrones de diseño, así como el manejo de la seguridad en el módulo, entre otros aspectos relevantes en la fase.

1.4.4. Pruebas Internas y Pruebas de Liberación

Al concluirse cada una de las fases en el desarrollo de un software debe realizarse un conjunto de validaciones y pruebas, que permitan determinar si el resultado es el esperado. "*Una prueba es una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan y registran, y se realiza una evaluación de algún aspecto*". (Pressman, 2006)

Para el Módulo Registro de Importadores y Exportadores se realizan las comprobaciones establecidas por el Programa de Mejora: Pruebas Internas y Pruebas de Liberación. Las Pruebas Internas, "*se realizan por los propios desarrolladores, se verifica el resultado de la implementación probando cada construcción, según sea necesario, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas*". Las Pruebas de Liberación "*son pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación*", en este caso son aplicadas por el Centro Nacional de Calidad de Software (CALISOFT) y el grupo de calidad de CEGEL. (Prieto Alvarez, 2013)

Dentro de las Pruebas Internas se realizan **revisiones** a la especificación de requisitos en su correspondiente fase de Requisitos. Para la validación del diseño se aplicaron **métricas** orientadas a clases: conjunto de métricas de CK, dentro de ellas Falta de Cohesión en los métodos (FCM), Acoplamiento entre Clases de Objetos (AECO) y Árbol de Profundidad de la Herencia (APH). También se realizaron pruebas de caja blanca a la codificación y caja negra a las funcionalidades, confeccionando los Casos de Pruebas (CP) correspondientes para cada prueba.

A partir de lo expuesto en (Pressman, 2006), se aplicaron **pruebas de caja blanca**: estas pruebas aseguran que la operación interna del programa se ajusta a las especificaciones y que todos los componentes internos se han probado adecuadamente. En resumen se centran en encontrar errores viendo el código interno. Para realizar estas pruebas se aplica la **prueba de la ruta básica** que consiste en una técnica de prueba de caja blanca. "*Permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución"*. Los casos de pruebas derivados para ejercitar el conjunto

básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba.

En cuanto a las **pruebas de caja negra**: "*se llevan a cabo sin tener conocimiento de la estructura/funcionamiento interno del sistema, de ahí su nombre. Quien realiza la prueba sólo conoce las entradas apropiadas que deberá recibir la aplicación, así como las correspondientes salidas, sin llegar a saber cómo es que se realiza este proceso*". (Pressman, 2006)

Estas pruebas funcionales se realizaron también por el grupo de calidad de CEGEL y por CALISOFT para la realización de las Pruebas de Liberación. En la actualidad CALISOFT realiza las pruebas clasificadas en: Basadas en Riesgos, de Confiabilidad, Eficiencia, Estáticas, Funcionalidad, Mantenibilidad (en desarrollo), Portabilidad y Usabilidad. (CALISOFT, 2010). De las anteriormente mencionadas se aplicaron al módulo las Pruebas de Funcionalidad y Pruebas Estáticas.

Con el objetivo de detectar fallas en la implantación del sistema con la participación de los analistas y el cliente, y como parte de la fase de Despliegue se aplicaron **pruebas de aceptación**, en un ambiente de trabajo productivo y verificando las funcionalidades del sistema. Durante todas las fases del modelo de desarrollo se utilizaron herramientas y tecnologías como apoyo al desarrollo de la propuesta de solución. Seguidamente se muestra el estudio y selección de estas herramientas.

1.5. Herramientas

Para apoyar y facilitar el desarrollo de la solución se utiliza una serie de herramientas. En el presente epígrafe se describen y explican cada una de estas herramientas, teniendo en cuenta sus características y versión. Dentro de estas herramientas se pueden mencionar por ejemplo: Visual Paradim, Netbeans, Kairos, GlassFish, PostgreSQL.

1.5.1. Herramientas CASE

"*La Ingeniería del Software Asistida por Computadoras (CASE) comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas*." (Sommerville, 2005)

Una de estas herramientas es Visual Paradigm 8.0, en (Visual Paradigm, 2013) se puede constatar que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. La misma facilita una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite diseñar diagramas de clases, generar código desde diagramas y generar documentación. Además posee diversas características: uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, capacidades de ingeniería directa (versión profesional) e inversa, modelo y código que permanece sincronizado en todo el ciclo de desarrollo, disponibilidad de múltiples versiones, para cada necesidad, disponibilidad de integrarse en los principales entornos de desarrollo y presenta licencia gratuita y comercial.

1.5.2. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario o Graphic User Interface (GUI). En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Además posee las siguientes características: multiplataforma, soporte para diversos lenguajes de programación, integración con sistemas de control de versiones, reconocimiento de sintaxis, extensiones y componentes para el IDE, depurador, importar y exportar proyectos, múltiples idiomas y manual de usuarios y ayuda. (Sánchez, 2013)

Para el desarrollo del presente trabajo se utilizó el IDE NetBeans en su versión 7.2. Es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio utilizando Java, JavaFX, PHP, JavaScript y Ajax, Ruby y RubyonRails, Groovy y Grails y C / C + +. (Sánchez, 2013)

1.5.3. Servidor de aplicaciones

Otra de las herramientas necesarias para el desarrollo de la solución es el servidor de aplicaciones, que maneja generalmente la mayor parte de las funciones pertenecientes a la lógica de negocio y al acceso a datos de una aplicación. Un servidor de aplicaciones permite facilitar alta disponibilidad, escalabilidad y mantenimiento de un sistema. Oracle Glassfish Open Source Server: es un servidor de aplicaciones de software, posee licencia de código abierto e implementa las tecnologías en la plataforma Edición Empresarial de Java o Java Enterprise Edition (JEE). Glassfish posee una excelente interfaz para facilitar las configuraciones, es un servidor rápido, presenta una alta escalabilidad y consume poca memoria. Posee una consola de administración web que posibilita al administrador manejar el servidor de manera remota. Posee una implementación robusta para el control de concurrencias, tolerancias a fallos, seguridad y alta disponibilidad. (Manchado, 2010)

1.5.4. Marco de trabajo

A partir de lo estudiado en el *Proyecto Técnico* de *SIRECC* ubicado en el *Expediente del Proyecto SIRECC*, para este proyecto fue establecido el Marco de Trabajo Kairos, desarrollado anteriormente para el Sistema de Gestión de Antecedentes Penales de la República Bolivariana de Venezuela. Para acelerar el proceso de desarrollo y facilitar el trabajo de los programadores se trabaja sobre este marco de trabajo que está diseñado para la creación de aplicaciones con arquitectura cliente-servidor utilizando la plataforma Ediciones Empresariales de Java o Java Enterprise Edition (JEE).

JEE cuenta con un conjunto de especificaciones que facilitan el desarrollo y despliegue de aplicaciones empresariales multi-capa. También ofrece un conjunto de especificaciones y técnicas que proporcionan soluciones completas, seguras, estables y escalables para el desarrollo, despliegue y gestión de aplicaciones de múltiples niveles de funcionalidad basadas en servidores. Se reduce el costo y complejidad de desarrollo, lo cual resulta en servicios que se pueden desplegar y extender fácilmente. JEE cuenta con los Enterprise Java Beans (EJB) que son componentes del lado del servidor que encapsulan la lógica de negocio de una aplicación y apuntan a crear un desarrollo rápido y simple para aplicaciones distribuidas, transaccionales, seguras y portables. (García, 2011)

Para hacer el mapeo objeto-relacional de Kairos se usó Java Persistence API (Aplication Programming Interface o Interfaz de Programación de Aplicaciones) o API para la Persistencia en Java (JPA), que

proporciona un modelo de persistencia basado en POJO's (Plain Old Java Object) para mapear bases de datos relacionales en Java. JPA fue desarrollado en conjunto con la especificación 3.0 de EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones de escritorio; incluso fuera de la plataforma JEE. Existen varias implementaciones de JPA como: EclipseLink, TopLink e Hibernate. Se utilizó EclipseLink debido a que es un marco de trabajo estable y muy recomendable para soluciones desarrolladas en Java y desplegadas en un servidor de aplicaciones Glassfish. (García, 2011)

Otra de las herramientas usadas por el marco de trabajo es Swing que consiste en un conjunto de clases de Java que simplifican la construcción de aplicaciones de escritorio. Permite tener un sistema de ventanas y componentes gráficos, independiente del sistema operativo y la librería de dibujo que se tengan disponibles en la máquina cliente. Swing es un extenso conjunto de componentes que van desde los más simples como etiquetas, hasta los más complejos como tablas, árboles, y documentos de texto con estilo. La característica más notable de los componentes Swing es que están escritos al 100% en Java. El marco de trabajo Kairos además de todas las características anteriormente mencionadas, posee una serie de mecanismos de seguridad, acreditándolo como un marco de trabajo potente a utilizar (García, 2011). Estos mecanismos pueden ser consultados en el *epígrafe 2.11. Seguridad en la aplicación*.

1.5.5. Sistema gestor de base de datos

Según (PostgreSQL Group, 1996), un Sistema Gestor de Base de Datos (SGBD) o DataBase Management System (DBMS) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Básicamente es un software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos garantizando además la seguridad de los mismos.

En la investigación el gestor de base de datos que se utilizó fue PostgreSQL en su versión 9.3. PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Es el sistema de

gestión de bases de datos de código abierto más potente del mercado. También se destaca por una amplia lista de prestaciones: la API de acceso al SGBD se encuentra disponible en C, C++, Java, Perl, PHP, Python y TCL, entre otros; cuenta con una amplia gama de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario, su administración se basa en usuarios y privilegios, es altamente confiable en cuanto a estabilidad se refiere, y tiene en cuenta el control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario. (PostgreSQL Group, 1996)

Para el desarrollo de la presente investigación fue necesario el uso de un lenguaje de programación, un lenguaje de modelado y un lenguaje de consulta orientado a objetos, seguidamente se brinda una caracterización de los seleccionados para la implementación de la solución.

1.6. Lenguajes

El término lenguaje "*puede ser entendido como un recurso que hace posible la comunicación*". (IEEE, 1999) En el mundo de la informática existen varios tipos de lenguajes que no son usados precisamente para establecer la comunicación entre los hombres. A continuación se mencionan y describen los principales lenguajes empleados en la investigación.

1.6.1. Lenguaje de modelado

"El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios." (Jacobson, y otros, 2000)

1.6.2. Lenguaje de programación

La programación es, en el vocabulario propio de la informática, el procedimiento de escritura del código fuente de un software. De esta manera, puede decirse que la programación le indica al programa informático qué acción tiene que llevar a cabo y cuál es el modo de concretarla. Con estas nociones en claro, podemos afirmar que un lenguaje de programación es aquella estructura que, con una base

sintáctica y semántica, imparte distintas instrucciones a un programa de computadora. (IEEE, 1999) El lenguaje de programación más apropiado para la implementación de la solución es Java por integrarse con las tecnologías y herramientas que se emplean en el desarrollo de la aplicación y por ser dominado por el equipo de desarrollo.

Java: es uno de los principales lenguajes de desarrollo de software en la actualidad. Entre las principales características que han llevado a tan rápida expansión, se encuentra que es independiente del sistema operativo y la plataforma, estandarizado, sencillo, distribuido, robusto, seguro. (Fernández, 2005)

- ✓ Independencia de la plataforma: Java es la tecnología ideal para desarrollar aplicaciones independientes del sistema operativo (Windows, Linux, SunOS, etc.), así como en un amplio espectro de dispositivos: ordenadores personales, entre otros.

- ✓ Estandarización: a diferencia de C++ y otros lenguajes, Java es un lenguaje completamente estandarizado, de modo que el código es independiente del entorno de desarrollo elegido.

- ✓ Sencillez: Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

- ✓ Distribuido: Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

- ✓ Seguridad: Java es uno de los primeros lenguajes en considerar la seguridad como parte de su diseño.

1.6.3. Lenguaje de consulta

En (Keith, y otros, 2009) "*el Lenguaje de Consulta para la Persistencia en Java o Java Persistence Query Language (JPQL) es un lenguaje de consulta independiente de plataforma orientada a objetos definidos como parte de la especificación Java Persistence API (JPA) y Application Programming Interface (API)*".

JPQL se utiliza para hacer consultas en entidades almacenadas en bases de datos relacionales. Está fuertemente inspirado en SQL, y sus consultas se asemejan a la sintaxis de las consultas SQL, solo que operan contra objetos-entidad en lugar de hacerlo directamente con las tablas de la base de datos. Seguidamente se definen un conjunto de patrones de diseño y estilos arquitectónicos que se proponen para la construcción de la solución, haciendo énfasis en las principales funciones y problemas que resuelven cada uno. (Meneses, 2012)

1.7. Estilos arquitectónicos

Según la IEEE-1471, la arquitectura de software se entiende como "*la organización fundamental de un sistema encarnada en sus componentes, las relaciones de los componentes con cada uno de los otros y con el entorno, y los principios que orientan su diseño y evolución*". A partir de lo expresado en (Reynoso, y otros, 2004) se plantean los siguientes estilos arquitectónicos a utilizar y sus características.

Uno de los estilos arquitectónicos es cliente-servidor que se basa en la existencia de "*un componente servidor, que ofrece ciertos servicios, escucha que algún otro componente requiera uno; un componente cliente solicita ese servicio al servidor a través de un conector. El servidor ejecuta el requerimiento (o lo rechaza) y devuelve una respuesta."*

El estilo n-capas constituye una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. La ventaja principal de este estilo arquitectónico es que en caso de que sobrevenga algún cambio, solamente se modifica el nivel requerido evitando cambios relevantes en los niveles adyacentes.

Arquitectura basada en componentes: describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado. La arquitectura basada en componentes tiene las siguientes características: es un estilo de diseño para aplicaciones compuestas por componentes individuales, pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas, define una aproximación de

diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

De manera general, el estilo arquitectónico cliente-servidor es aplicable a la presente investigación ya que es necesario para cumplir con las exigencias del cliente debido a que se procesan varias peticiones concurrentemente, n-capas para organizar de manera jerárquica la estructura del sistema y basado en componentes con el objetivo de facilitar el mantenimiento y la reutilización, además acelerar el proceso de desarrollo del módulo. Luego de establecidos los estilos arquitectónicos a utilizar para el desarrollo del módulo se pasó a la definición de los patrones de diseño.

1.8. Patrones de diseño

En (Larman, 2000) se expresa que un patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Expresado lo anterior con palabras más simples, el patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Atendiendo al diseño y a la arquitectura, se mencionan y explican cada uno de los patrones seleccionados, implementados además por el marco de trabajo Kairos.

Larman expresa que los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). Existen dos tipos de patrones para el diseño los Patrones Generales de Software para la Asignación de Responsabilidades o General Responsibility Assignment Software Patterns (GRASP) y el Grupo de los Cuatro o Gang of Four (GoF). Los patrones GRASP son:

- ✓ Controlador: se encarga de coordinar la ejecución de las funciones de las demás clases. Es un objeto de interfaz no destinada al usuario que se encarga de asignar la responsabilidad de manejar un evento del sistema a una determinada clase.

- ✓ Experto: es un patrón que se usa más que cualquier otro al asignar responsabilidades. Su objetivo es asignar una responsabilidad al experto en información, es decir; la clase que cuenta con la información necesaria para cumplir la responsabilidad.

- ✓ Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

- ✓ Bajo acoplamiento: estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como experto o alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

- ✓ Alta cohesión: es un principio que se debe tener presente en todas las decisiones de diseño, es la meta principal que debe buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Se da una alta cohesión funcional cuando los elementos de un componente "*colaboran para producir algún comportamiento bien definido*". Se trata de asignar una responsabilidad de modo que la cohesión siga siendo alta.

Los patrones GoF usados en la investigación son:

- ✓ Fachada: Se le da el nombre de fachada a la clase definida que ofrece una interfaz común unificada con un conjunto heterogéneo de interfaces, como la de un subsistema, y asigna la responsabilidad de colaborar con el subsistema. Las interfaces heterogéneas pueden ser un conjunto de funciones, un esquema, un grupo de otras clases o un subsistema (local o remoto).

- ✓ Instancia única o singlenton: garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

- ✓ Factoría: se define un objeto "factoría" para crear objetos de otras clases, cuando existen consideraciones especiales, como una lógica de creación compleja o el deseo de separar las responsabilidades de la creación para mejorar la cohesión.

Además de estos patrones se usará la inyección de dependencias, proporcionada por el contenedor de EJB:

- ✓ Inyección de dependencias: se especializa en inyectar objetos en una clase, en lugar de ser la propia clase quien cree el objeto.

Una vez establecidos los patrones de diseño a utilizar para el desarrollo del módulo, se dio paso a la definición de la teoría necesaria para llevar a cabo la valoración de las variables del problema en cuestión; evaluando con los resultados obtenidos la propuesta de solución.

1.9. Valoración de la solución

En el marco teórico de la investigación se define en el problema a resolver como variables: disponibilidad de la información (variable dependiente) y Módulo Registro de Importadores y Exportadores (variable independiente). En cuanto a la valoración del Módulo Registro de Importadores y Exportadores se realiza un estudio de su alcance atendiendo a porciento de los requisitos funcionales implementados y su calidad tanto interna como externa. En cuanto a la disponibilidad de la información, se analizan las dimensiones personal autorizado, considerando el acceso a la Información y tiempo requerido teniendo en cuenta el tiempo de respuesta de las consultas realizadas.

El tiempo de respuesta obtenido a la hora de consultar información se valora con respecto al tiempo especificado en el RnF 7, el cual plantea que: "*el tiempo de respuesta brindado por el sistema será menor 3 segundos*". Esta valoración tiene lugar con la aplicación implantada en el entorno real, o sea en la Cámara de Comercio, y con ayuda de las funcionalidades que proporciona la herramienta Netbeans. También otro aspecto a valorar son los mecanismos que proporciona el marco de trabajo Kairos, los cuales contribuyen de manera positiva al indicador acceso a la información.

Para llevar a cabo la valoración final también se aplicó una entrevista a dos especialistas de la Cámara de Comercio, específicamente del departamento Dirección Jurídica, por ser los usuarios finales de la aplicación y constituyen las personas encargadas de trabajar directamente con el proceso registro de

empresas importadoras/exportadoras. Luego de aplicada la entrevista se analizaron los resultados y se arrojaron conclusiones.

1.10. Conclusiones parciales

Con la realización de la fundamentación teórica de la investigación se pudo llegar a las siguientes conclusiones:

- ✓ Se realizó un análisis de los principales sistemas de registro de información de empresas importadoras/exportadoras obteniendo como resultado que ninguno de los sistemas estudiados cumple con las exigencias del Módulo Registro de Importadores y Exportadores para SIRECC.

- ✓ Se analizaron herramientas y lenguajes donde se decidió para la realización del presente trabajo utilizar el modelo de desarrollo Programa de Mejora propuesto por la UCI y el marco de trabajo Kairos, como lenguaje de modelado UML y como herramienta CASE Visual Paradigm 8.0, como servidor de bases de datos PostgreSQL en su versión 9.3, el IDE NetBeans 7.2 y servidor de aplicaciones Glassfish 3.1.2.

- ✓ Se definen los patrones de diseño GRASP y GoF, así como los estilos arquitectónicos que son cliente-servidor, n-capas y basado en componentes, así como las diferentes pruebas de software a aplicar una vez concluido el producto final, como pruebas internas y de liberación.

CAPÍTULO 2 PROPUESTA DE SOLUCIÓN

2.1. Introducción

En el presente capítulo se describen temas relacionados con el diseño e implementación del módulo. Se obtiene una serie de artefactos de diseño, como el diagrama de clases y el modelo de datos. Con la obtención de estos artefactos se facilita el trabajo a los programadores, ganando en comprensión y rapidez, logrando así un mejor desarrollo de la solución. El capítulo está estructurado a partir de las fases definidas en el Programa de Mejora propuesto por la UCI (*Anexo 2. Fases del Programa de mejora propuesto por la UCI)*, adecuado al proyecto SIRECC y específicamente a la investigación, cuyo alcance limita en la fase de Despliegue, donde se realizarán las pruebas de aceptación, como una primera etapa de esta fase.

2.2. Descripción de la solución

La UCI, en específico CEGEL y la Cámara de Comercio de la República de Cuba, están interesados en el desarrollo de un módulo que permite a grandes rasgos, registrar la información de las empresas importadoras/exportadoras, además de importar y exportar los documentos correspondientes a su registro. El desarrollo del módulo permite el almacenamiento y validación de los datos obtenidos durante la inscripción de empresas importadoras/exportadoras, su manejo centralizado durante las modificaciones y cancelaciones de las empresas, además visualizar los cambios que han tenido las empresas en cuanto a facultades (importar, exportar o ambas). Para ello existen tres procesos fundamentales a informatizar como bien se explicaron en el capítulo anterior: inscripción, modificación de facultad y cancelación de facultad, para ello ver e*pígrafe 1.2. Descripción de los procesos de negocio*.

La propuesta de solución del trabajo consiste en el diseño, implementación y prueba del Módulo Registro de Importadores y Exportadores, solución que está provista de las funcionalidades necesarias para que exista un correcto funcionamiento a la hora de registrar una empresa y de modificar o cancelar sus facultades. El módulo se debe integrar con otros módulos, que se encuentran en fase de desarrollo.

2.3. Antecedentes de la investigación

A continuación se realiza una descripción de cada una de las fases de Estudio Preliminar, Modelado del Negocio y Requisitos, cada uno de los documentos y artefactos mencionados se encuentran ubicados en el *Expediente del Proyecto SIRECC.* En una primera fase del proceso de informatización realizada al registro de empresas importadoras/exportadoras, se efectuó un **Estudio Preliminar**, donde se llevó a cabo la planeación del proyecto, un estudio del cliente, del alcance de la solución, que permitió realizar estimaciones de tiempo, esfuerzo y costo; además se definió la visión del proyecto. Se realizó un estudio de la documentación del cliente y de los procesos que realizan descritos en *Manual de procedimiento de cada uno de los registros,* para lograr un entendimiento del negocio, identificar las áreas a informatizar y obtener una planificación preliminar de la duración del proyecto. Se generó una serie de artefactos, como: *Acta de Inicio* y *de Aceptación del Proyecto*, *Minutas de Reunión* para cada uno de los entregables, además el *Acta de Entrega en cumplimiento del hito de desarrollo "Levantamiento de Requisitos v1.0"* pactado con el cliente en la *Ficha de Proyecto*, donde se hizo entrega de los *Documentos de Especificación de Requisitos* de los diferentes módulos del sistema, *Acta de Entrega en cumplimiento de acuerdos tomados con el cliente en minuta de reunión el 30/11/2012*, haciendo entrega de los documentos: *Proyecto Técnico, Ficha del Proyecto* y el *Cronograma de Ejecución de SIRECC,* todos firmados por el cliente.

Luego de concluida la fase de Estudio Preliminar se pasó al **Modelado del Negocio**, donde se comprendió cómo funciona el negocio que se desea informatizar. En esta fase se modelaron los diferentes procesos con que cuenta el sistema a desarrollar. Se generaron diferentes entregables como *Reglas de Negocio*, *Modelo de Procesos de Negocio, Glosario de Términos* y *Minutas de reunión* que fueron necesarias realizar con el cliente.

Posteriormente se pasó a la fase de **Requisitos**, donde se realizó el levantamiento de requisitos, obteniendo 21 requisitos funcionales y 33 no funcionales y se validó con el cliente la propuesta a partir de prototipos. En esta fase se generó una serie de artefactos como *Plan de Pruebas*, donde se establece la planificación de las actividades que son realizadas por el equipo del proyecto durante el desarrollo del software para verificar la calidad del producto que se está desarrollando; *Plan de Desarrollo de Software,* donde se define el plan para la ejecución del proyecto, partiendo de su visión, metas e involucrados; *Especificación de Requisitos de Software*, donde se detalla todo lo referente a los requisitos funcionales;

atendiendo a número de requisito, nombre, descripción, complejidad y prioridad para el cliente, también contiene un prototipo de interfaz y cada uno de los campos que posee con su tipo de dato y las reglas o restricciones.

Otros de los artefactos generados son: *Registro de Revisiones para el Compromiso al Plan, Registro Problemas, Desviaciones y Acciones, Registro Planes y Registro de Monitoreo, Proyecto Técnico,* donde se describen los requisitos, restricciones y riesgos de alto nivel del proyecto que deben ser del conocimiento de los clientes, la alta gerencia y el equipo de desarrollo; la *Matriz de Disponibilidad de Proveedores*, el *Registro de Revisiones de Inconsistencias*, además de las *Minutas de reunión* realizadas hasta esa fecha. Seguidamente se presentan los requisitos funcionales y algunos de los no funcionales pertenecientes al Módulo Registro de Importadores y Exportadores, para ver todos los requisitos no funcionales, dirigirse al documento *Especificación de Requisitos No funcionales de Software* ubicado en el *Expediente de proyecto*:

2.3.1. Requisitos funcionales:

RF01: Inscribir empresa importadora/exportadora.

RF02: Modificar datos de la empresa importadora/exportadora.

RF03: Visualizar datos de la empresa importadora/exportadora.

RF04: Cancelar inscripción.

RF05: Buscar empresa importadora/exportadora.

RF06: Listar empresas importadoras/exportadoras.

RF07: Listar registros.

RF08: Eliminar registro.

RF09: Crear expediente.

RF10: Modificar datos del expediente.

RF11: Cancelar expediente.

RF12: Visualizar datos de expediente.

RF13: Importar documento.

RF14: Modificar documento.

RF15: Eliminar documento.

RF16: Adicionar documento generado al expediente.

RF17: Listar documentos.

RF18: Visualizar documento.

RF19: Exportar documento.

RF20: Visualizar empresas por diferentes parámetros.

RF21: Visualizar empresas por actividad económica.

2.3.2. Requisitos no funcionales:

Usabilidad

- ✓ Finalidad:

RnF3. Este sistema estará enfocado a la gestión de información relacionada con los procesos registrales que se llevan a cabo en la Cámara de Comercio.

- ✓ Ambiente:

RnF4. El sistema deberá presentar una interfaz de usuario fácil de entender y usar.

Confiabilidad

RnF8. El tiempo máximo de inactividad es de 5 min, y una vez que el sistema quede inactivo el usuario deberá autenticarse nuevamente y el sistema mantendrá el estado que tenía antes de pasar a la inactividad.

Eficiencia

RnF12. El sistema debe garantizar el acceso concurrente para todos los usuarios del sistema durante la jornada laboral establecida.

Soporte

RnF13. El uso del sistema requerirá un tiempo de preparación previa por los usuarios finales para su correcta explotación.

Restricciones de diseño

RnF15. Para el montaje del sistema se requerirá del sistema gestor de bases de datos PostgreSQL 9.1 y del servidor de aplicaciones Glassfish 3.1.2.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RnF17. Integrar ayuda al sistema. El sistema debe contar con una ayuda integrada que permita al usuario orientarse en cada una de sus interfaces. La ayuda debe brindar una explicación detallada del contenido de la interfaz.

Interfaz

RnF19. El sistema presentará una interfaz legible, simple de usar e interactiva.

Seguridad

RnF25. El sistema podrá ser utilizado solamente por usuarios autenticados en el mismo.

Requisitos de software

RnF30. Instalar en las estaciones de trabajo el software necesario para el correcto funcionamiento del sistema como: visor de documentos PDF, máquina virtual de java versión 1.6 actualización 26.

Como conclusión de las fases de Estudio Preliminar, Modelado de Negocio y Requisitos, se obtuvo una serie de artefactos, constituyendo las salidas de las fases anteriormente mencionadas y se convirtieron en entradas a la fase de Análisis y Diseño, la cual se detalla en el siguiente epígrafe.

2.4. Análisis y Diseño

En esta fase los requisitos descritos durante la etapa anterior fueron refinados y estructurados para conseguir una comprensión más precisa de los mismos y una descripción que sea fácil de mantener y ayude a estructurar el sistema (incluyendo su arquitectura), a partir de esta actualización se obtuvo una nueva versión de la *Especificación de Requisitos*. Como uno de los artefactos generados se encuentra el *Modelo del Diseño* donde se creó el diagrama de paquetes, el diagrama de clases, necesarios para la implementación exitosa del módulo. También otro de los artefactos generados fueron las *Minutas de Reunión* realizadas hasta la fecha y los *Documentos de Arquitectura de Software,* como *Arquitectura Vista de Seguridad* y *Arquitectura Vista de Presentación*, los cuales pueden ser consultados en el (*Expediente del Proyecto SIRECC*).

2.4.1. Diagrama de paquetes

El diagrama de paquetes muestra las agrupaciones lógicas en las que se encuentra dividido el módulo y las dependencias entre ellas. Suministra una descomposición de la jerarquía lógica del módulo. Un fragmento del diagrama se puede visualizar en la *Figura 1. Diagrama de paquetes Registro de Importadores y Exportadores*. No se especifican detalles por mantener la confidencialidad de la información del proyecto. Los paquetes representados son: el paquete presentación que contiene los formularios y acciones, los gestores de negocio del cliente, gestores de negocio del servidor y gestores de acceso a datos, cada uno de ellos relacionados entre sí.

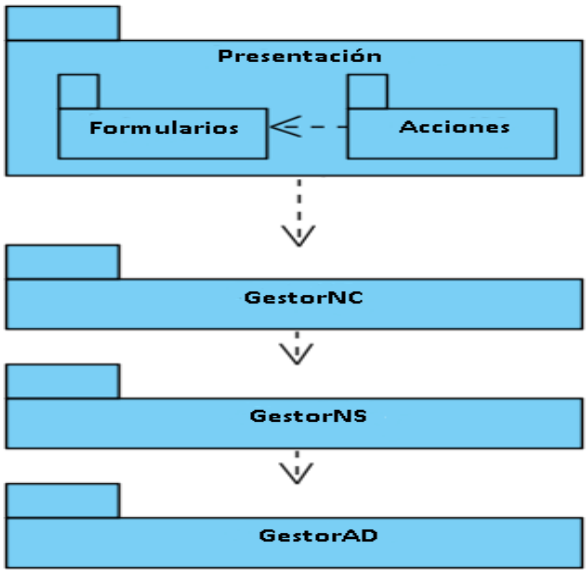

Figura 1. Diagrama de paquetes Registro de Importadores y Exportadores

Además del diagrama de paquetes también fue necesario el diseño del diagrama de clases con cada una de sus relaciones.

2.4.2. Diagrama de clases del diseño

El diagrama de clases del diseño describe las principales clases que lo conforman, sus asociaciones y métodos, en este caso no se especifican atributos ni métodos por mantener la confidencialidad de la información del proyecto. En la *Figura 1. Fragmento diagrama de clases del Módulo Registro de Importadores y Exportadores,* se puede observar el diseño de las clases por las que está conformado el diagrama. En este caso está acotado al requisito *Inscribir empresas importadoras/exportadora*. En primer lugar las acciones (*AccionRegistrarDatosImportadorExportador*) validan y recogen los datos insertados por el usuario en el formulario (*pnlRegistrarDatosImportadorExportador*), estos datos son enviados al gestor de negocio del cliente (*GestorNCImportadorExportador*) el cual gestiona la información del negocio en el cliente de las empresas importadoras/exportadoras, registradas en el sistema y le pide a la interfaz fachada de gestores remota (*FachadaGestoresRemotos*) comunicación con el servidor. El gestor de negocio del servidor (*GestorNSImportadorExportador*) envía los datos a través de las entidades de dominio (*EDImportadorExportador*) al gestor de acceso a datos (*GestorADImportadorExportador*) para almacenarlos, no sin antes convertir estas entidades de dominio en entidades persistentes (*DImportadorExportador*) con la factoría (*FactoriaEDImportadorExportador*).

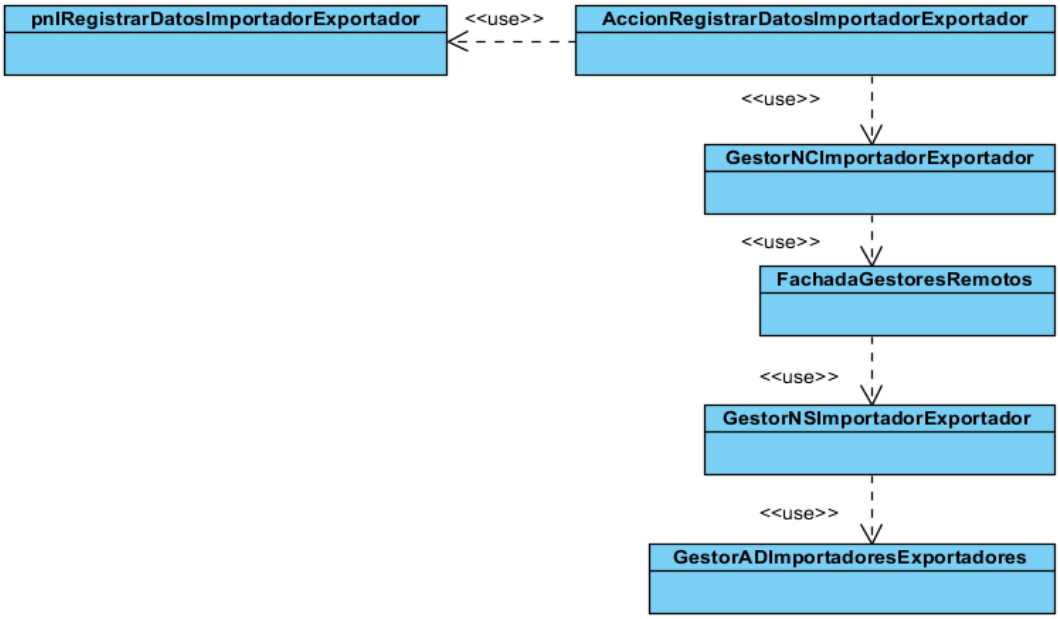

Figura 2. Fragmento diagrama de clases del Módulo Registro de Importadores y Exportadores

Además del diagrama de clases del diseño fue necesario conocer y obtener la estructura de la base de datos, para ello se diseña el modelo de datos.

2.4.3. Modelo de datos

Con el diseño del modelo de datos se define la estructura de la base de datos, está compuesto por entidades, que manejan la información a persistir; atributos, que son las características básicas de las entidades, y sus relaciones, que son las que enlazan a las entidades entre sí. Seguidamente en la *Figura 3. Fragmento modelo de datos del Módulo Registro de Importadores y Exportadores* se muestra un fragmento del modelo de datos de la presente investigación, para visualizar el diagrama completo dirigirse al *Anexo 3. Modelo de datos de Registro de Importadores y Exportadores,* en este caso no se especifican detalles por mantener la confidencialidad de la información del proyecto. Una de las entidades más significativas es **DImportadorExportador**, esta contiene los atributos específicos de las empresas importadoras/exportadoras que se inscriben, como su identificador, la fecha de la inscripción, estatus jurídico; que se refiere a si la empresa es: capital cien porciento extranjero, contrato de asociación económica, estatal, mixta o sociedad mercantil cubana; su facultad que puede ser importar, exportar o ambas; y su concepto que puede ser modificación, cancelación o ratificación.

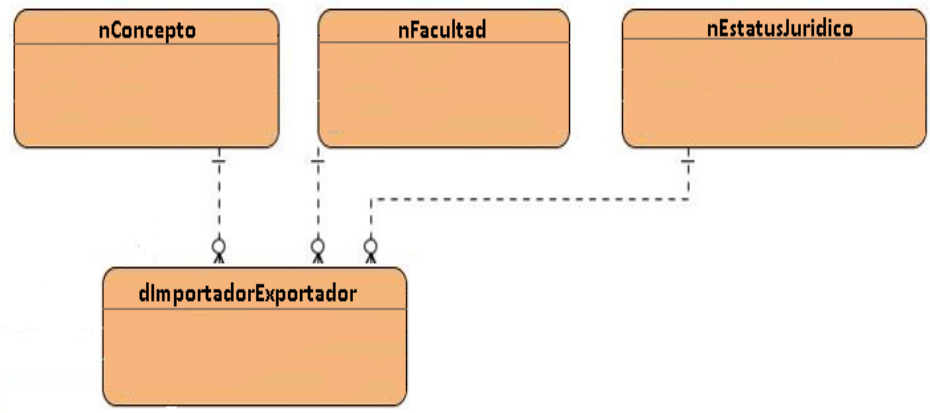

Figura 3. Fragmento modelo de datos del Módulo Registro de Importadores y Exportadores

A partir de los resultados obtenidos en la fase de Análisis y Diseño, se desarrolla la siguiente fase de Implementación del Módulo Registro de Importadores y Exportadores, donde esos artefactos generados constituyen las entradas de esta etapa, facilitando así la implementación.

2.5. Implementación

En el epígrafe se muestra el modelo de implementación con otros de los diagramas necesarios para implementar la solución, como el diagrama de componentes y de despliegue. Se evidencia la utilización de los diferentes patrones de diseño y estilos arquitectónicos en la solución. Además se dejan plasmados los estándares de codificación utilizados y se describe cómo se maneja la seguridad en la aplicación.

2.5.1. Diagrama de componentes

En la *Figura 4. Diagrama de componentes Registrar Datos Importador y Exportador* se representa cómo el módulo está dividido en componentes y se muestra la organización y las dependencias entre estos componentes. Este diagrama se estructura en paquetes, que son divisiones físicas del sistema. Los paquetes están organizados en una jerarquía de capas donde cada capa tiene una interfaz bien definida. En este caso para lograr una mejor visualización y entendimiento del diagrama, se acotó a Registrar datos de importador/exportador.

En cada uno de los paquetes se puede observar los componentes que lo conforman, por ejemplo en el paquete *Presentación* podemos observar el componente que representa a los formularios, específicamente *pnlRegistrarDatosImportadorExportador*, y las acciones, como

AccionRegistrarDatosImportadorExportador, en el paquete *Negocio-Cliente* podemos encontrar la *FachadaGestoresRemoto* y el *GestorNCImportadorExportador* representando a los gestores de negocio del cliente, en el paquete *Negocio-Servidor* encontramos los gestores de negocio del servidor, por ejemplo el *GestorNSImportadorExportador*, en el paquete *Acceso a Datos* se representan los gestores de acceso a datos, las factorías y el conjunto de entidades de dominio, en este caso podemos visualizar el *GestorADImportadorExportador*, la *FactoriaEDImportadorExportador* y la *EDImportadorExportador*.

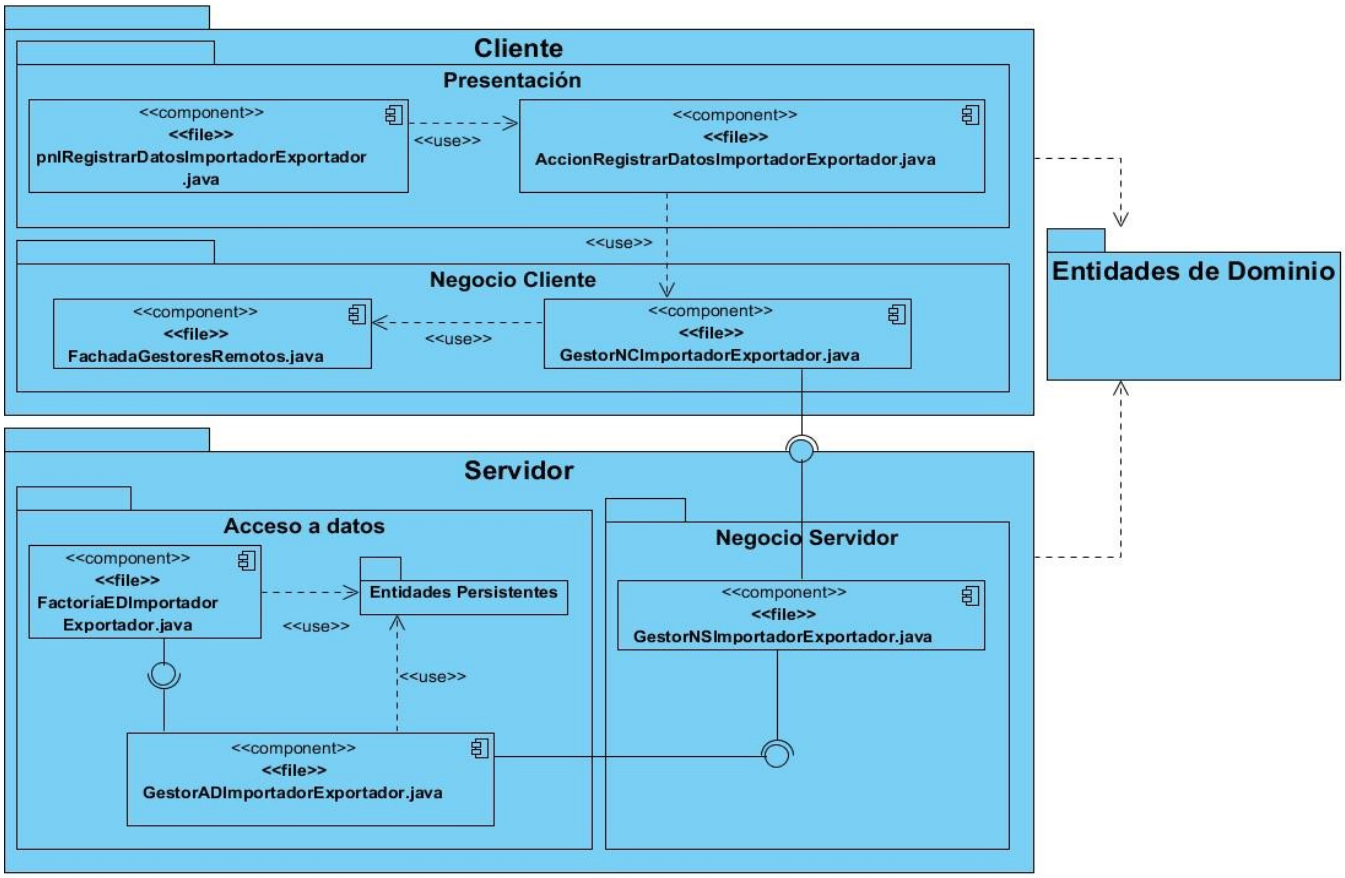

Figura 4. Diagrama de componentes del Módulo Registro de Importadores y Exportadores

Luego de diseñado el diagrama de componentes con cada una de sus especificidades se diseña el diagrama de despliegue.

2.5.2. Diagrama de despliegue

El diagrama de despliegue elaborado permite describir la distribución física del sistema en términos de componentes de software entre los recursos o nodos físicos, como una computadora personal (PC) cliente. Permitió capturar la configuración de los elementos de procesamiento (por ejemplo: servidor de aplicaciones) y las conexiones entre estos en el sistema. Además describe la configuración del sistema para su ejecución en un ambiente del entorno real.

El diagrama de la *Figura 5. Diagrama de despliegue del Módulo Registro de Importadores y Exportadores* muestra cómo está definido el despliegue de la aplicación. La cual se encontrará distribuida en dos nodos físicos; la PC-Cliente y un servidor central, ambos tienen en común la utilización de la Máquina Virtual de Java pues la aplicación ha sido desarrollada sobre esa plataforma. Además en el servidor central se encuentra el servidor de aplicaciones Glassfish, el cual administra las conexiones al gestor de base de datos PostgreSQL según la demanda de peticiones realizadas a Sirecc-cliente, atendidas por el componente Sirecc-ea. También se tiene un nodo con los dispositivos impresora y escáner, para la impresión y digitalización de los documentos.

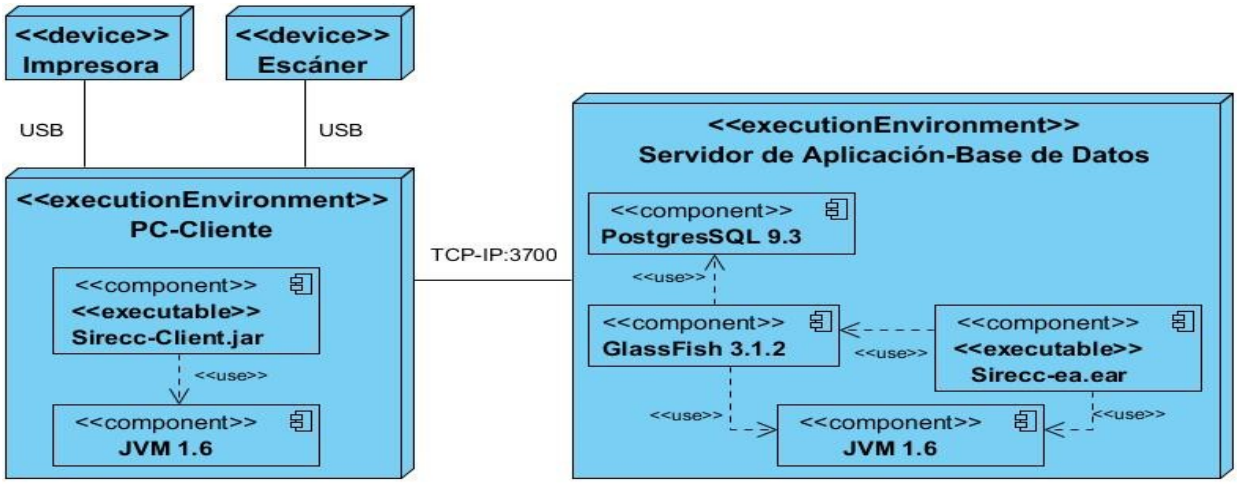

Figura 5. Diagrama de despliegue del Módulo Registro de Importadores y Exportadores

Durante el diseño de los diagramas necesarios para la implementación, fue necesario evidenciar el uso de los patrones de diseño y de los estilos arquitectónicos.

2.5.3. Estilos arquitectónicos

A partir de lo expuesto en el documento *Proyecto Técnico de SIRECC* ubicado en el *Expediente del Proyecto SIRECC, e*l marco de trabajo Kairos provee a las aplicaciones que lo utilizan funcionalidades y componentes que agilizan y simplifican su desarrollo. Kairos para proporcionar un nivel de mantenibilidad adecuado a la solución está enfocado en un estilo cliente-servidor. Este estilo permite contar con una solución distribuida donde se puede establecer una comunicación entre aplicaciones clientes y servidoras, que permiten dar solución a los requisitos de los usuarios, abstrayéndolos de la distribución física de la solución. El objetivo fundamental de la aplicación cliente es la interacción entre la solución y los usuarios finales. Las aplicaciones clientes no implementan reglas de negocio, solamente solicitan información que necesiten a la aplicación servidor. Por su parte la aplicación servidor contiene la implementación de todas las reglas de negocio de la solución, además de la gestión de la información que se maneja, siendo su objetivo fundamental; dar respuesta al cliente ante una petición de información.

A pesar de que el estilo arquitectónico cliente-servidor proporciona un alto nivel organizativo dentro de la solución, no es suficiente, ya que el diseño de las aplicaciones cliente y servidor continúa siendo complejo. Para proporcionar una mejor organización se utiliza además el estilo n-capas, el cual permite establecer una organización jerárquica entre cada una de las capas, las cuales agrupan funcionalidades similares, garantizando que cada capa proporcione sus servicios a la capa inmediata superior y que esta a su vez se sirva de las prestaciones que le brinda la capa inmediata inferior.

Además el diseño arquitectónico se orienta a la reutilización de componentes desarrollados previamente sin tener en cuenta la composición de los mismos, con el fin de reducir el ciclo de desarrollo y proporcionar un diseño más escalable y refinado. La arquitectura basada en componentes permitirá descomponer el diseño de la solución en componentes funcionales, dentro de estos podemos encontrar los formularios, los gestores, las factorías y las acciones.

La arquitectura específicamente está diseñada con el uso del patrón cuatro capas de manera horizontal, que contiene los componentes que responden a los requerimientos funcionales del sistema y dos capas verticales, que contienen las clases que afectan a las demás capas de la aplicación.

La aplicación está estructurada por las siguientes capas:

Cliente:

- ✓ Presentación: capa que contiene los componentes de interfaz de usuario. Permite mostrar y validar la información requerida por el usuario, obteniendo esta información a través de solicitudes a la capa inmediata superior (negocio del cliente).
- ✓ Negocio del cliente: capa que gestiona la lógica de negocio del cliente. Esta es una capa cuya responsabilidad fundamental es llevar a cabo procesos de validación y gestionar la información entre la aplicación cliente y el servidor.

Servidor:

- ✓ Negocio del servidor: capa que gestiona los componentes que implementan la lógica de negocio del sistema.
- ✓ Acceso a Datos: capa que gestiona la información entre la aplicación servidor y la base de datos.

Capas verticales:

Capa que contiene las clases como son las entidades de dominio que afectan todas las capas horizontales y las entidades persistentes que afecta la aplicación servidor en su conjunto.

En la *Figura 6. Arquitectura del Módulo Registro de Importadores y Exportadores*, se pueden observar con mayor claridad estos estilos utilizados.

Figura 6. Arquitectura del Módulo Registro de Importadores y Exportadores

2.5.4. Patrones de diseño

Como bien se explica en el capítulo anterior el módulo se desarrolla con la utilización del marco de trabajo Kairos y este a su vez implementa una serie de patrones de diseño, los cuales son empleados en la solución.

Patrones GRASP:

Patrón experto: este patrón se manifiesta en las clases Entidades de Dominio, ya que estas contienen toda la información correspondiente a las tablas de la BD. Estas clases poseen la información necesaria para cumplir con sus responsabilidades. Por ejemplo, se muestra la clase EDImportadorExportador la cual contiene atributos propios en los que es experta.

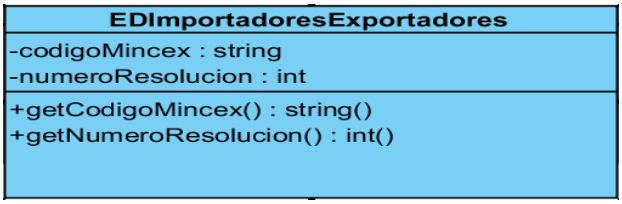

Figura 7. Aplicación del patrón experto

Patrón factoría (Gof) y creador: estos patrones se evidencian en las clases Factorías, ya que asignan a determinadas clases la responsabilidad de crear una instancia de otra clase. En este caso se muestra la *FactoriaEDImportadoresExportadores*, encargada de convertir la entidad de dominio

EDImportadoresExportadores en entidad persistente *DImportadorExportador*. En este caso se define un objeto "factoría" para crear objetos de otras clases.

Figura 8. Aplicación del patrón creador y factoría

Patrón alta cohesión y bajo acoplamiento: la utilización del patrón alta cohesión se encarga de la asignación de responsabilidades propiciando sólo las dependencias necesarias entre las clases y la utilización del patrón bajo acoplamiento proporciona las dependencias mínimas entre las diferentes clases, en la *Figura 9. Aplicación del patrón alta cohesión y bajo acoplamiento* se puede constatar cómo la acción se comunica directamente y sólo con el gestor de negocio del cliente, este con el gestor de negocio del servidor y este a su vez con el gestor de acceso a datos, como lo refleja el estilo n-capas utilizado, para ello ver e*pígrafe 2.5.3 Estilos arquitectónicos utilizados, Figura 6. Arquitectura del Módulo*. De esta manera a la hora de realizar modificaciones necesarias, no implica grandes cambios a todo el negocio. También manteniendo lo más bajo posible el acoplamiento se logra que las distintas "piezas" del software funcionen sin depender demasiado unas de otras y posibilita una mayor reutilización de cada una de las clases.

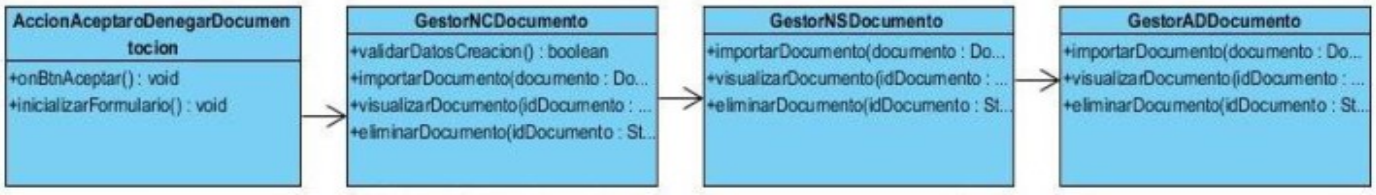

Figura 9. Aplicación del patrón alta cohesión y bajo acoplamiento

Patrón controlador: se evidencia en las Acciones, ya que estas clases se encargan de coordinar la ejecución de las funciones de las demás clases. Todas las peticiones realizadas por el usuario son manejadas por estas acciones, en este caso podemos visualizar la *AccionRegistrosEmpresasImpExp*, con una representación de sus atributos y métodos. Esta acción específicamente se encarga de cargar el formulario donde permite insertar los nuevos valores de la empresa y así ser registrados.

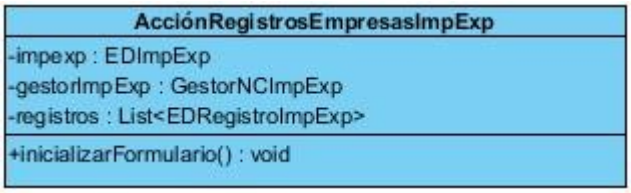

Figura 10. Aplicación del patrón controlador

Patrones GOF:

Patrón fachada: evidenciado en la clase *FachadaGestoresRemoto*, la utilización de este patrón posibilita un punto de acceso común entre varias clases, en este caso permite comunicar al cliente con el servidor. En la *Figura 11. Aplicación del patrón fachada* se muestra la clase representativa de este patrón con sus principales funcionalidades.

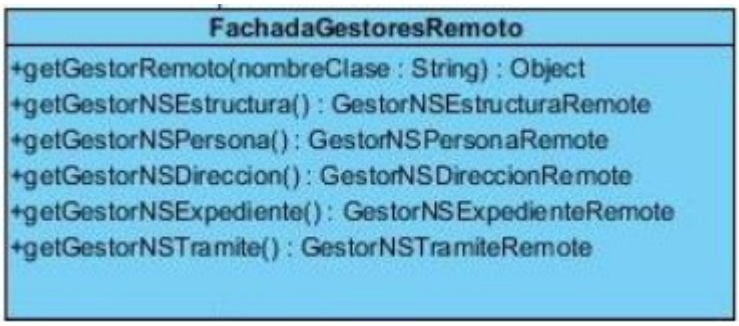

Figura 11. Aplicación del patrón fachada

Patrón instancia única o singlenton: patrón diseñado para la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón es utilizado en la clase GestorMarcoTrabajo que permite abstraerse de la creación de instancias, seguidamente se muestra.

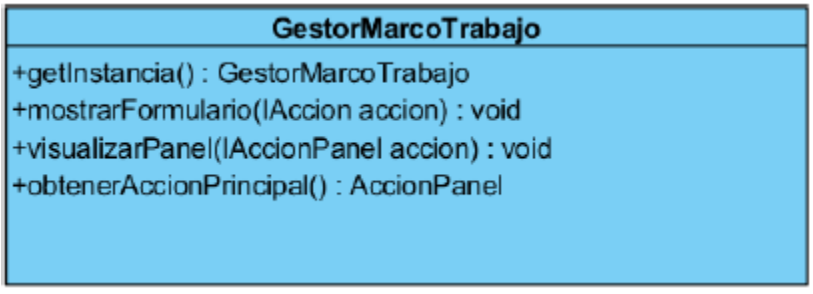

Figura 12. Aplicación del patrón singlenton

Otro de los patrones utilizados es **inyección de dependencias**: este patrón posibilitó inyectar objetos en una clase, en lugar de ser la propia clase quien cree el objeto. En este caso se muestra un fragmento de la clase GestorADImportadorExportador, donde se puede visualizar una inyección para instanciar un objeto del sessionbean (son EJB's) GestorADSaclapLocal, utilizado en la ejecución de distintas funcionalidades.

```
@EJB
private GestorADSaclapLocal gestorADSaclap;
```

Figura 13. Aplicación del patrón inyección de dependencias

2.5.5. Estándar de codificación

Con el objetivo de lograr uniformidad en la implementación de la solución y entender código nuevo más rápidamente y más a fondo, se utiliza el Estándar de Codificación para Java definido por la Dirección Técnica de la Producción, Basado en Convenciones de Código para el lenguaje de programación JAVA™ por Scott Hommel Sun Microsystems Inc. A grandes rasgos se describen algunas de estas normas y para examinarlas con un mayor nivel de detalle pueden ser consultadas en (de Producción, 2011):

Nomenclatura General:

- ✓ Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- ✓ En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: "fechaCreacion" y no solamente "fecha".
- ✓ El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase, que rige que para un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

Nombres de ficheros: los nombres de fichero describen el uso o propósito de la clase y utilizan el estilo de escritura UpperCamelCase ejemplo: GestorADEmpresaCubana (Gestor de Acceso a Datos de la Empresa Cubana).

Organización de los ficheros: Los ficheros tienen el siguiente orden:

- ✓ Comentarios de comienzo.
- ✓ Sentencias package e import.
- ✓ Declaraciones de clases e interfaces.

Longitud de la línea: Evitar las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas terminales y herramientas.

2.5.6. Implementación del Módulo Registro de Importadores y Exportadores

A continuación se realiza una descripción de manera general de la clase AccionRegistrarDatosImportadorExportador, el resto puede ser consultado en el *Anexo 4. Descripción de las clases del Módulo Registro de Importadores y Exportadores* para lograr así una mayor comprensión de la aplicación:

Nombre de la Acción	Descripción
AccionRegistrarDatosImportadorExportador	Esta acción específicamente se encarga de cargar el formulario donde permite insertar los nuevos valores de la empresa y así ser registrados.
También se encuentran las acciones AccionBuscarListarEmpresasImportadorasExportadoras, AccionRegistrarDatosImportadorExportador, AccionVisualizarDatosImportadorExportador, RegistrosEmpresaImportadoraExportadora, AccionCancelarInscripcionImportadoresExportadores y AccionInsertarDatosFuncionarioImportadorExportador, entre otras.	

Tabla 2. Descripción de las clases acciones del Módulo Registro de Importadores y Exportadores

El marco de trabajo Kairos brinda una estructura que es la utilizada en el desarrollo del proyecto SIRECC, la cual se explica de manera general, teniendo en cuenta específicamente el Módulo Registro de Importadores y Exportadores.

En la *Figura 14. Estructura del Proyecto SIRECC y del Módulo Registro de Importadores y Exportadores*, se puede observar que el proyecto *SIRECC* está compuesto fundamentalmente por el paquete *sirecc*, que contiene las carpetas *negocio* y *presentación*. En la carpeta *negocio* podemos encontrar las clases relacionadas con los gestores de negocio del cliente como *GestorNCImportadorExportador* y la fachada de gestores remotos (*FachadaGestoresRemotos)*, en la de *presentación* cada una de las acciones definidas en la aplicación como por ejemplo la *AccionModificarDatosImportadorExportador* y todo lo referente a los formularios, ejemplo: *FrmCancelarInscripcionImportadorExportador*. *SIRECC* también está compuesto por el paquete *sirecc-ejb*, que contiene las entidades persistentes como *Dimportadorexportador*, las factorías como *FactoriaEDImportadoresExportadores* y los gestores de acceso a datos como *GestorADImportadoresExportadores* y el paquete *sirecc-ejb-interfaces* donde se encuentran las interfaces de los gestores de negocio del servidor, como *GestorNSImportadorExportadorRemote*. El proyecto también incluye el paquete *sirecc-entidades* que contiene las entidades de dominios definidas, por ejemplo: *EDImportadoresExportadores*.

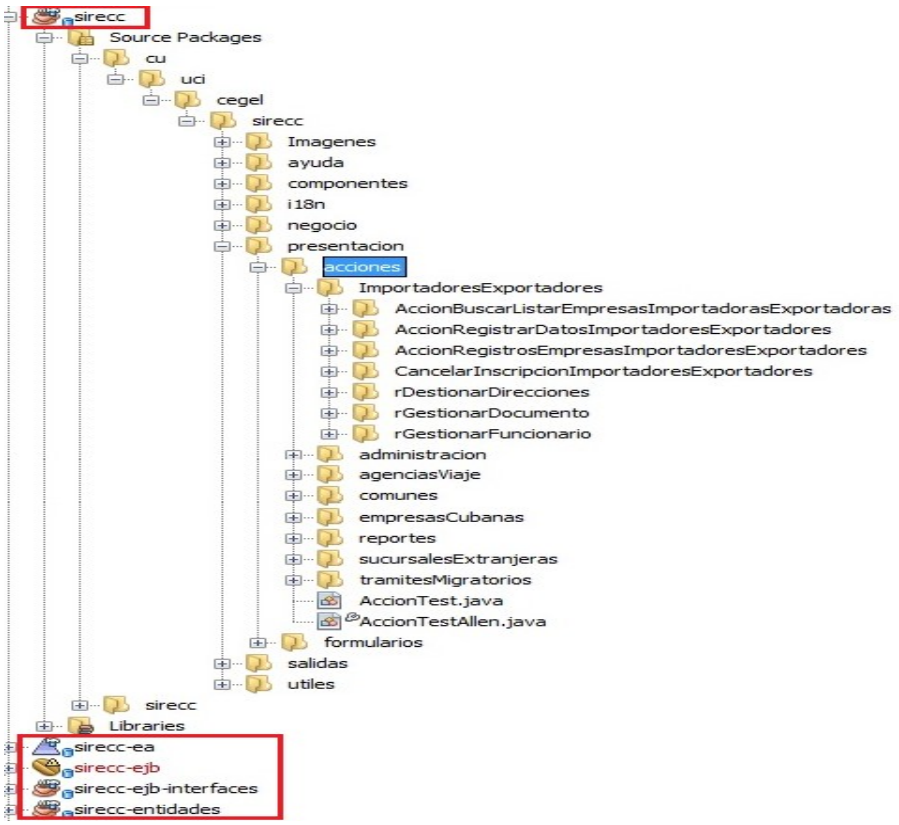

Figura 14. Estructura del proyecto SIRECC y del Módulo Registro de Importadores y Exportadores

2.5.7. Seguridad de la solución

Las aplicaciones de software manejan gran cantidad de información, que constituye el bien más preciado de cualquier entidad, como pueden ser datos de los clientes, de cuentas bancarias o del funcionamiento de la propia institución. Por este motivo se hace necesario que esta sea accedida, creada y modificada por las personas que cuenten con los permisos para ello. En caso contrario la organización correría el riesgo de que utilicen dicha información maliciosamente para que sea manipulada, ocasionando lecturas erradas o incompletas. Por esta razón uno de los elementos más importantes dentro de las soluciones de software empresariales es la seguridad, la cual requiere el empleo de una considerable cantidad de tiempo y esfuerzo.

La seguridad en el Módulo Registro de Importadores y Exportadores cumple con las especificaciones planteadas en el documento *Arquitectura Vista de Seguridad* para el proyecto SIRECC, que es definido por el Programa de Mejora establecido en la UCI. Este documento puede ser consultado en el *Expediente del Proyecto SIRECC*. El cual presenta como objetivo fundamental, chequear e implementar todos los aspectos relacionados con el acceso a la aplicación, la modificación, lectura o eliminación de la información, así como establecer los niveles de seguridad de SIRECC. Este documento plantea que la seguridad se establece por niveles, en cada uno de estos niveles se especifica una serie de aspectos que se lleva a cabo en la aplicación para así contribuir con la seguridad, como el uso de roles.

Para complementar las medidas de seguridad en la aplicación se definieron los siguientes requisitos no funcionales en el rango de RnF-25 hasta RnF-29, pueden ser consultados en el *Expediente del Proyecto SIRECC*.

- ✓ RnF25. El sistema podrá ser utilizado solamente por usuarios autenticados en el mismo.
- ✓ RnF26. El sistema brindará la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado.
- ✓ RnF27. El sistema mostrará las funcionalidades de acuerdo a quien n esté autenticado en el mismo.
- ✓ RnF28. El sistema debe asegurar el almacenamiento de las credenciales de los usuarios utilizando algoritmos criptográficos que oculten la identidad verdadera de los usuarios.

- ✓ RnF29. El sistema debe permitir almacenar todas las acciones de los usuarios sobre el sistema como constancia de las acciones realizadas.

El marco de trabajo Kairos presenta un conjunto de mecanismos de seguridad que permiten el aseguramiento de la información que maneja la aplicación desarrollada. Seguidamente se describen cada uno de ellos, a partir de lo definido en (Matos Rodríguez , y otros, 2012)

Mecanismo de autenticación y autorización:

Mecanismo que permite autenticar a los usuarios contra el servidor de aplicaciones Glassfish y controla el acceso a las funcionalidades publicadas en este servidor. Este presenta un dominio de seguridad, el cual actúa como intermediario entre el servidor de aplicaciones y la Base de Datos, donde se encuentra la información de identificación de las cuentas de usuarios, las funcionalidades requeridas para llevar a cabo la autenticación desde una aplicación cliente y define cómo se debe realizar la autorización en el servidor de aplicaciones.

- ✓ **Los dominios de seguridad:** permiten autenticar a los usuarios contra una Base de Datos. El servidor Glassfish permite la configuración del dominio de seguridad de manera sencilla a través de su consola de administración web.

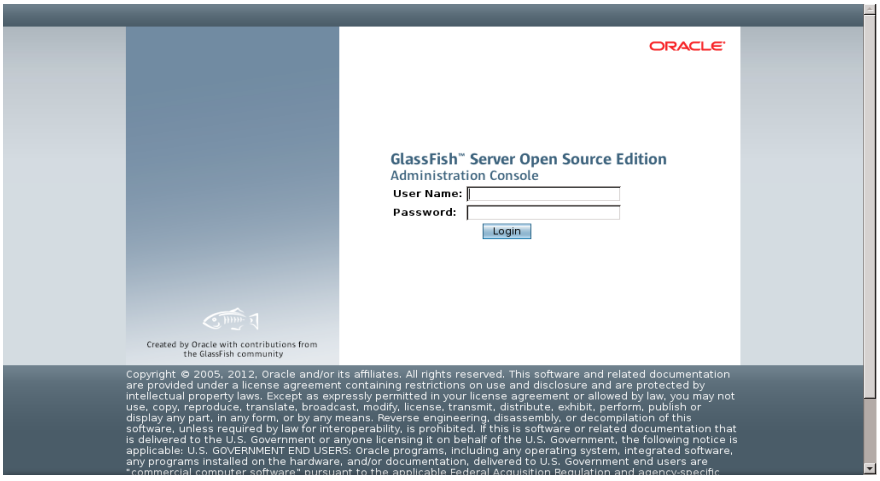

Figura 15. Consola de administración del Glassfish

- ✓ **Autenticación desde el cliente:** El servidor Glassfish contiene la lógica encargada de publicar en el mismo la identidad del cliente para su posterior acceso. Esta se encarga de iniciar una sesión y finalizarla.

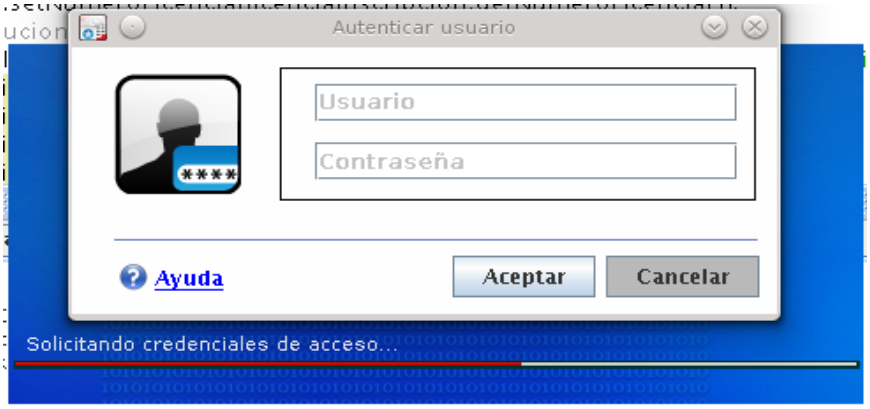

Figura 16. Solicitud de credenciales en SIRECC

- ✓ **Autorización en el servidor:** Verifica que quien intente invocar una funcionalidad determinada en el servidor cuenta con los permisos necesarios para hacerlo. Se basa en el uso de roles (privilegio que posee un cliente para invocar cierta funcionalidad o acceder a cierto recurso en un ambiente asegurado). Para llevar a cabo este proceso la plataforma JEE soportada por el servidor Glassfish provee la anotación @RolesAllowed, la cual admite como argumento un conjunto de roles en forma de cadenas de texto que representan los roles con que debe contar un usuario para acceder al recurso anotado, y puede ser utilizada a nivel de método o a nivel de clase.

```
@RolesAllowed(RolBase.USUARIO_BASICO)
public class GestorADImportadoresExportadores
```

Figura 17. Autorización en el servidor

Validación de la fortaleza de la contraseña:

Para verificar la fortaleza de una contraseña el marco de trabajo Kairos contiene un mecanismo que permite la configuración y el control de la fortaleza de las contraseñas de las cuentas de usuario. El nivel de fortaleza de las contraseñas viene dado por la medición de parámetros como la cantidad total de

caracteres, la cantidad de caracteres especiales, la cantidad total de dígitos y la cantidad de letras en mayúscula.

Encriptación de contraseña:

El marco de trabajo realiza una encriptación de contraseñas utilizando las funciones de resumen md5 y sha-1, las cuales son sólo de ida o irreversibles, garantizando la confidencialidad de las contraseñas. Gracias a esto no es posible revertir la encriptación para obtener la contraseña a partir de su valor encriptado, posibilitando que las contraseñas sean almacenadas y transmitidas por la red o cualquier otro canal inseguro con un nivel menor de riesgos.

Bloqueo de sesión por inactividad:

Kairos presenta un mecanismo que permita bloquear la aplicación luego de un tiempo definido sin la actividad de un usuario**.** Este detecta los eventos de teclado y ratón que se presentan en la aplicación, de esta forma se controla el tiempo de inactividad, el cual consiste en el tiempo en que no ocurren dichos eventos. Cuando se bloquea la aplicación se oculta la interfaz principal y se muestra un cuadro de diálogo que permite la autenticación de un usuario para desbloquear la aplicación.

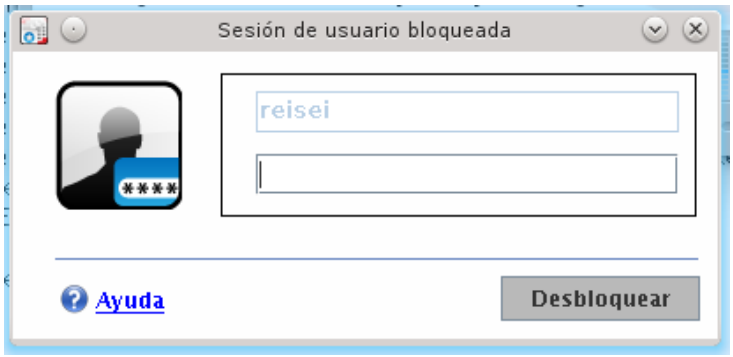

Figura 18. Sesión de usuario bloqueada

2.6. Pruebas internas

En este epígrafe se explican cada una de las **pruebas internas** llevadas a cabo y los resultados obtenidos. Se validó la especificación de requisitos aplicando una serie de revisiones y una lista de verificación. En cuanto al diseño, se emplean métricas orientadas a clases: colección de métricas de CK y por último se le realizan pruebas a la codificación, como las pruebas de caja blanca y de caja negra.

2.6.1. Validación de la especificación de requisitos

Para obtener la información referente a la validación de requisitos se efectuó una entrevista a la analista principal del proyecto SIRECC, la cual se puede visualizar en el *Anexo 5. Entrevista a analista principal del proyecto SIRECC*. En la fase de Requisitos se aplicó la **Revisión Técnica Formal (RTF)**, también llamada **Inspección**, teniendo en cuenta lo definido por (Pressman, 2006). El beneficio obvio de la RTF es el descubrimiento temprano de los errores, de modo que no se propaguen al paso siguiente en el desarrollo del software.

Se efectuó una **Revisión** a la especificación de requisitos del Módulo Registro de Importadores y Exportadores, luego de realizada la revisión se redactó un *Registro de Inconsistencias.* Los responsables fueron el jefe de proyecto y el administrador de la calidad y los participantes: administrador de la calidad, analista del módulo y revisores, esta información fue adquirida del *Plan de Pruebas*. El *Plan de Pruebas* se encarga de la planificación de las actividades que serán realizadas por el equipo del proyecto durante el desarrollo del software para verificar la calidad del producto que se está realizando. El fin de esta última revisión fue el *Acta de Liberación-Evaluación de Productos de Software*, en este caso el artefacto liberado fue *Especificación de Requisitos del Módulo Registro de Importadores y Exportadores*. Todos los artefactos generados se encuentran en el *Expediente del Proyecto SIRECC.*

Además de esta técnica aplicada, se examinó cada requisito frente a una serie de preguntas en forma de lista de verificación. Se puede consultar en el documento ***Criterios para validar requisitos del cliente****,* que además constituye una técnica para validar los requisitos y un artefacto definido por el Programa de Mejora para estas validaciones. En este documento se redactaron y aplicaron preguntas como; ¿El requisito (REQ) no es ambiguo?, ¿El REQ está completo?, ¿El REQ puede ser implementado?, ¿El REQ puede ser probado?, entre otras, asegurando la calidad de los mismos. Dejando bien claro el resultado obtenido, en este caso todos los requisitos fueron aprobados satisfactoriamente, permitiendo el entendimiento y compromiso con el cliente.

Prototipos de interfaz, es otra de las técnicas utilizadas donde se hicieron simulaciones del posible producto. Cada uno de los prototipos le permitió al cliente tener una idea de las interfaces gráficas del módulo. Estos se realizaron de forma no funcional con la herramienta Visual Paradigm con el objetivo de lograr mayor aprobación por parte del cliente.

2.6.2. Validación del diseño de la solución

El diseño fue otro de los aspectos validados en la investigación. En la actualidad existen muchas agrupaciones de métricas, en este trabajo se estudiaron y emplearon diversas de las definidas dentro de las Métricas Orientadas a Clases: Colección de Métricas de CK, propuestas en (Pressman, 2006), y precisamente teniendo a Pressman como referencia es que se definen, explican y aplican cada una de las métricas. Las cuales abarcan atributos que permiten medir la calidad del diseño propuesto, estos atributos son cohesión, acoplamiento y herencia; atributos que por sus características evidenciadas en la solución, fueron tenidos en cuenta, para así determinar el grado de cada uno de ellos en este caso en el diseño. La métrica basada en la herencia se concentra en la manera en que las operaciones se reutilizan en la jerarquía de clases. En cuanto a la cohesión, un software debe diseñarse de manera que todas las operaciones trabajen en combinación para alcanzar un solo propósito, un buen diseño. El grado de cohesión de una clase se determina al examinar el grado en que "*el conjunto de propiedades que posee es parte del dominio del problema o el diseño*". Por último el acoplamiento está representado por el número de colaboraciones entre clases o el de mensajes pasados entre objetos. Seguidamente se describen y explican cada una de las métricas aplicadas al diseño del módulo.

Métricas Orientadas a Clases: Colección de Métricas de CK

La clase es la unidad fundamental de un sistema orientado a objetos. (Pressman, 2006). Chidamber y Kemerer propusieron uno de los conjuntos de métricas de software orientado a objetos al que se hace referencia con más frecuencia. Seguidamente se aplican las empleadas en la validación del diseño:

- ✓ **Falta de Cohesión en los Métodos (FCM)**

FCM es el número de métodos que acceden a uno o más de los mismos atributos. Un valor bajo de FCM indica una alta cohesión y una clase bien diseñada. Es probable que si el sistema tiene buena subdivisión de clase, implica simplicidad y alta reusabilidad. Una clase cohesiva tenderá a proporcionar un alto grado de encapsulamiento. Un valor más alto de FCM indica la disminución del encapsulamiento y el aumento de la complejidad, lo que aumenta la probabilidad de errores. FCM varía en el intervalo [0,2]. Si FCM > = 1 indica una clase muy problemática. Lo deseable es mantener alta la cohesión, es decir conservar baja la FCM. (Pressman, 2010) La métrica se le aplica a una de las clases más significativas del negocio, el

GestorADImportadorExportador, posteriormente a cada uno de los atributos de la clase se le fija una identificación:

GestorADImportadoresExportadores	
Atributos	**Identificación**
factoriaEDDatosEstructura	a
factoriaEDImpExp	b
factoriaEDEstructura	c
factoriaEDFuncionario	d
gestorADDireccion	e
gestorADDocumento	f
gestorADFuncionario	g
gestorADSaclap	h
manager	i

Tabla 3. Identificación de los atributos de la clase GestorADImportadoresExportadores

Luego de determinadas las identificaciones de cada una de las propiedades, a los métodos se le establecerá los atributos a los cuales él accede:

GestorADImportadoresExportadores	
Métodos	**Atributos**
inscribirImportadorExportador	h,c,a,b,g,e,f,i
modificarImportadorExportador	h,g,e,f,i
buscarEmpresaImportadoraExportadora	h,d,b,c,e,f,i
buscarEmpresasImportadorasExportadoras	b,c,e,i
buscarEmpresaPorActividadEconomica	h,c,i
cantImportadoresExportadoresExisten	i
listadoActualPartidas	i
listadoTodasPartidas	b,i
obtenerRegistros	i
eliminarRegistros	i
listarRegistros	i
comprobarCambiosActividades	-
actividadesIguales	i
obtenerFuncionariosActivos	i
devolverDocumento	i
cancelarInscripcionImportadorExportador	i

Tabla 4. Métodos de la clase y atributos a los que acceden

Después de tener los atributos que son accedidos por los métodos, hay que calcular la cantidad de métodos que acceden a uno o más de los mismos atributos:

GestorADImportadoresExportadores		
Atributos	**Identificación**	**mA**
factoriaEDDatosEstructura	a	1
factoriaEDImpExp	b	4
factoriaEDEstructura	c	4
factoriaEDFuncionario	d	1
gestorADDireccion	e	4
gestorADDocumento	f	3
gestorADFuncionario	g	2
gestorADSaclap	h	4
manager	i	15
	∑(mA)	38

Tabla 5. Número de métodos que acceden a un mismo atributo de la clase

m: Número de métodos en la clase. m=16

a: Número de atributos en la clase. a=9

mA: Número de métodos que acceden a un atributo.

$$1 - \sum (mA) \; / \; (m \times a)$$

=1-38/(16*9)

=1-38/144=1-0.26=0.74

Como la FCM es baja, o sea FCM no es > = 1, indica que existe una cohesión alta, por tanto el GestorADImportadoresExportadores es una clase bien diseñada. Este resultado puede ser visualizado en la *Figura19. Resultado de la aplicación de la métrica FCM.*

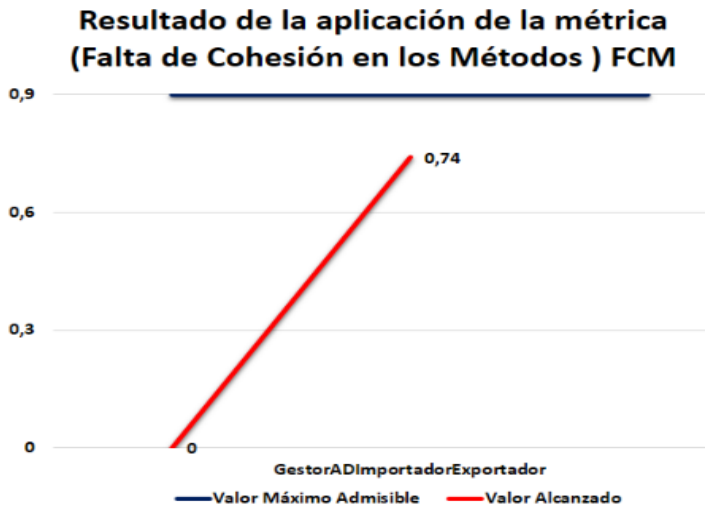

Figura 19. Resultado de la aplicación de la métrica FCM

Además de la métrica Falta de Cohesión en los Métodos, se aplicó:

- ✓ **Acoplamiento entre Clases de Objeto (AECO)**

Las tarjetas índice CRC (Clase-Responsabilidad-Colaboración) son modelos para representar clases. Están divididas en tres secciones; a lo largo de la cabecera de la tarjeta usted escribe el nombre de la clase, las responsabilidades de la clase y los colaboradores, que son las clases que se requieran para que una clase reciba la información necesaria para completar sus funcionalidades. El objetivo que persiguen es desarrollar una representación organizada de las clases. (Pressman, 2010)

El AECO es el número de colaboraciones enlistadas, para una clase en su tarjeta de índice CRC. A medida que AECO aumenta, es probable que disminuya la facilidad de reutilización de una clase. Valores elevados de AECO también complican las modificaciones y la prueba que asegura que esas modificaciones se han hecho. En general para cada clase debe mantenerse el valor más bajo que sea razonable. Es deseable que el valor de AECO se encuentre por debajo de 14 para poder asegurar la mantenibilidad y reusabilidad del diseño propuesto. Las múltiples colaboraciones con una misma clase son tratadas como un único acceso. (A. Sahraoui, y otros) Los métodos que no tienen colaboraciones no se tuvieron en cuenta.

GestorADImportadoresExportadores	
Responsabilidades	**Colaboradores**
inscribirImportadorExportador	FactoriaEDEstructuraLocal
	FactoriaEDDatosEstructuraLocal
	FactoriaEDImportadoresExportadoresLocal
	GestorADDocumentoLocal
	GestorADFuncionarioLocal
	GestorADDireccionLocal
	GestorADSaclapLocal
modificarImportadorExportador	GestorADDireccionLocal
	GestorADSaclapLocal
	GestorADDocumentoLocal
	GestorADFuncionarioLocal
buscarEmpresaImportadoraExportadora	FactoriaEDEstructuraLocal
	FactoriaEDImportadoresExportadoresLocal
	FactoriaEDFuncionarioLocal
	GestorADDireccionLocal
	GestorADDocumentoLocal
	GestorADSaclapLocal
	FactoriaEDEstructuraLocal
buscarEmpresasImportadorasExportadoras	FactoriaEDEstructuraLocal
	FactoriaEDImportadoresExportadoresLocal
	GestorADDireccionLocal
buscarEmpresaPorActividadEconomica	FactoriaEDEstructuraLocal
	GestorADSaclapLocal
listadoTodasPartidas	FactoriaEDImportadoresExportadoresLocal
Total	8

Tabla 6. Tarjeta Índice CRC para la clase GestorADImportadoresExportadores

Se obtuvo un valor de AECO igual a 8, lo que permite afirmar que el diseño propuesto en esta investigación es válido según la métrica aplicada, además como el AECO se encuentra por debajo de 14 se puede asegurar la mantenibilidad y reusabilidad del diseño propuesto. El resultado obtenido puede ser visualizado en la *Figura 20. Resultado de la aplicación de la métrica AECO.*

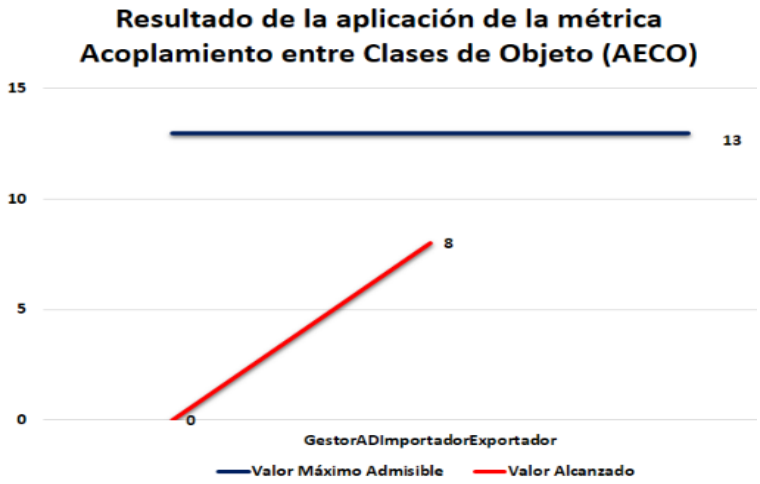

Figura 20. Resultado de la aplicación de la métrica AECO

- ✓ **Árbol de Profundidad de la Herencia (APH)**

Esta métrica es la longitud máxima desde el nodo hasta la raíz del árbol, o sea mide el máximo nivel de la jerarquía de herencia. A medida que crece la APH, es probable que las clases del nivel inferior hereden muchos métodos. Esto se presta a posibles dificultades cuando se trata de predecir el comportamiento de una clase. Una jerarquía de clase profunda (su APH es mayor) también se presta a una mayor complejidad de diseño. Por el lado positivo, valores grandes de APH, indican que se puede reutilizar muchos métodos. (Pressman, 2006)

En (Vázquez Escudero, y otros, 2002) se sugiere un umbral de 6 niveles como indicador de un abuso en la herencia. Sistemas construidos a partir de marcos de trabajos suelen presentar unos niveles de herencia altos, ya que las clases se construyen a partir de una jerarquía existente. En lenguajes como Java o Smalltalk, las clases siempre heredan de la clase Object, lo que añade uno a APH. En la *Figura 21. Árbol de Profundidad de la Herencia*, se puede observar cada uno de los niveles que conforman el árbol, siendo un total de 5 niveles, incluyendo la clase Object, a partir de lo planteado en la métrica. Se considera un abuso de la utilización a partir de 6 niveles, por tanto en este caso las clases no son complejas, atendiendo a su diseño y potencial de reutilización, es menos propensa a errores, y las clases del nivel inferior heredan los métodos necesarios, en general, el indicador de herencia es bueno para este árbol.

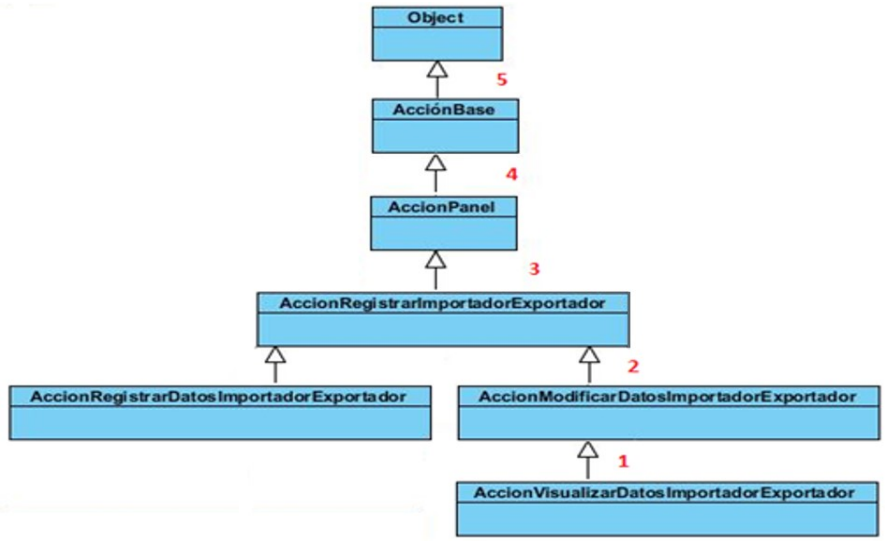

Figura 21. Árbol de Profundidad de la Herencia

2.6.3. Validación del código fuente

Una vez generado el código fuente es necesario probar el software para descubrir y corregir la mayor cantidad posible de errores antes de entregarlo al cliente. Para ello según lo definido en el *Capítulo 1: Fundamentación Teórica*, se aplican internamente, pruebas de caja blanca y de caja negra.

Pruebas de caja blanca

Al emplear esta prueba, según (Pressman, 2006), los ingenieros del software podrán derivar casos de prueba que:

- ✓ Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
- ✓ Ejerciten los lados verdaderos y falsos de todas las decisiones lógicas.
- ✓ Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- ✓ Ejerciten estructuras de datos internos para asegurar su validez.

☑ **Prueba de la ruta básica**

Se selecciona para aplicarle la técnica, la funcionalidad *comprobarCambiosActividades*(). Primeramente se numera cada una de las sentencias de código del método en la *Figura 22. Código fuente numerado, de la funcionalidad comprobarCambiosActividades()*.

```
public boolean comprobarCambiosActividades(EDImportadoresExportadores edImpExp, List<Integer> listaId) throws Exception {
    boolean cambiosActividades = true;  //1
    for (int i = 0; i < edImpExp.getActividades().size(); i++) { //2
        if (!listaId.contains((Integer) edImpExp.getActividades().get(i).getId())) { //3
            cambiosActividades = false; //4
            break; //5
        }
    }
    return cambiosActividades; //6
}
```

Figura 22. Código fuente numerado de la funcionalidad

Luego de numeradas cada una de las sentencias, se construye el grafo de flujo correspondiente, el cual describe un flujo de control de la funcionalidad, para ello ver *Figura 23. Grafo de Flujo asociado a la funcionalidad comprobarCambiosActividades()*.

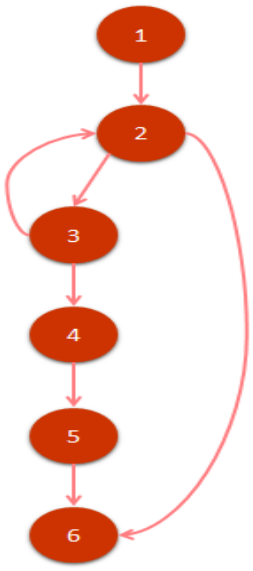

Figura 23. Grafo de flujo asociado a la funcionalidad

Inmediatamente de haber construido el grafo se realiza el cálculo de la complejidad ciclomática. La complejidad ciclomática es una métrica de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. Cuando se emplea en el contexto del método de Prueba de la Ruta Básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado al menos una vez. (Pressman, 2006) El cálculo de esta complejidad se realiza mediante las tres fórmulas descritas en la *Tabla 7. Cálculo complejidad ciclomática*, deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

PASO 1	**PASO 2**	**PASO 3**
V (G) = (E – N) + 2 Complejidad Ciclomática: **V (G)** Gráfica de Flujo: **G** Cantidad total de aristas: **E** Cantidad total de nodos: **N** **V (G) =** (7 – 6) + 2 **V (G) =** 3	**V (G) = P + 1** Cantidad total de nodos predicados (nodos de los cuales parten dos o más aristas): **P** **V (G)** = 2 +1 **V (G)** = 3	**V (G) = R** Cantidad total de regiones, para cada fórmula “V (G)” representa el valor del cálculo: **R** **V (G)** = 3

Tabla 7. Cálculo complejidad ciclomática

A partir del cálculo realizado se obtuvo el mismo resultado en cada una de las fórmulas aplicadas, en este caso se obtiene como complejidad ciclomática 3, lo que un límite superior de 3 posibles caminos independientes por donde el flujo puede circular y por implicación un límite superior del número de pruebas que deben diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones del programa, 3 de igual manera. Los caminos básicos por los que puede recorrer el flujo son:

Camino básico #1: 1-2-3-4-5-6

Camino básico #2: 1-2-3-2-3-4-5-6

Camino básico #3: 1-2-3-2-6

Seguidamente de haber determinado los caminos básicos se procedió a ejecutar los Casos de Pruebas (CP) para cada uno de estos. Para definir los CP se consideraron las entradas, descripción, condiciones de ejecución y resultados esperados. A continuación se muestra un ejemplo de uno de los CP, los demás pueden ser consultados en el *Anexo 6. CP aplicados en camino básico* realizados.

Camino básico #1: 1-2-3-4-5-6	
Entradas	Las variables "listaId", "edImExp.getActividades().get(i).getId()" y "edImExp.getActividades()"
Condiciones de ejecución	La longitud de "edImExp.getActividades()" tiene que ser al menos 1 y el elemento "edImExp.getActividades().get(i).getId()" no esté contenido en la lista "listaId".
Resultados esperados	Se espera que el método devuelva "false". El resultado fue correcto.

Tabla 8. Caso de Prueba: Camino básico #1

Se puede verificar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado anteriormente, por tanto se concluye que los resultados son satisfactorios.

Pruebas de caja negra

Para la ejecución de este tipo de pruebas se realiza un conjunto de CP para determinar si los requisitos estaban parcial o completamente satisfactorios. En total se realizaron 21 CP, probándose cada una de las funcionalidades y evaluándose sus resultados. Estos CP pueden ser consultados en (*Expediente del Proyecto SIRECC)*. En la *Tabla 9. Fragmento del CP04_Cancelar Inscripción*, se puede visualizar un fragmento del CP del requisito cancelar inscripción, con cada uno de los escenarios definidos, las entradas y el flujo central.

Escenario 1	Descripción	Fecha de cancelación	Estado	Número de resolución	Fecha de la resolución	Observaciones	Respuesta del sistema	Flujo central
EC 1.1Escenario Válido	Permite cancelar la inscripción de la empresa importadora y exportadora	V 10/07/2013	V Pasiva	V 2255	V 20/01/2012	V No existen Observaciones.	Se cancela el registro de la empresa .	Se debe seleccionar en el menú la opción IMPORTADORES EXPORTADORES. Posteriormente en el submenú que aparece se debe seleccionar la opción Cancelar empresa importadora exportadora. Luego el sistema muestra la interfaz Buscar Empresa importadora exportadora en la cual se debe buscar la empresa que se desea cancelar. Posteriormente se selecciona en los resultados de la búsqueda, la empresa que se desea cancelar y se presiona el botón Cancelar Empresa.
		V 10/07/2013	V Pasiva	V 2255	V 20/01/2012	N/A		
EC 1.2 Escenario Inválido	No permite cancelar la inscripción de una empresa importadora exportadora satisfactoriamente debido a que existen datos incorrectos.	I 10/07/20	V Pasiva	V 2255	V 20/01/2012	V No existen Observaciones.	El sistema muestra un mensaje indicando que hay campos que han sido introducidos incorrectamente .	
		V 10/07/2013	I 234	V 2255	V 20/01/2012	V No existen Observaciones.		
		V 10/07/2013	V Pasiva	I FALSO	V 20/01/2012	V No existen Observaciones.		
		V 10/07/2013	V Pasiva	V 2255	I 20/01/2013	V No existen Observaciones.		
		V 10/07/2013	V Pasiva	V 2255	V 20/01/2012	I 567		
		I	I	I	I	I		

Figura 24. Fragmento del CP04_Cancelar Inscripción

Para verificar que el sistema cumple con el mínimo de calidad necesaria se realizan pruebas internas antes de entregar el producto al grupo de calidad de CEGEL y a CALISOFT. En una primera iteración se detectaron un total de 38 no conformidades (NC), de ellas se resolvieron 29 y 9 quedaron pendientes para una próxima iteración, donde finalmente se resolvieron todas. Los resultados pueden ser visualizados en la *Figura 24. Resultados de pruebas funcionales realizadas internamente*.

Luego de aplicadas todas estas validaciones y pruebas, el producto fue entregado al equipo de calidad de CEGEL y a CALISOFT, donde le aplican también pruebas de caja negra o funcionales como parte de las pruebas de liberación. Los resultados de estas pruebas se pueden constatar en el *Capítulo 3. Validación de la Solución Propuesta*.

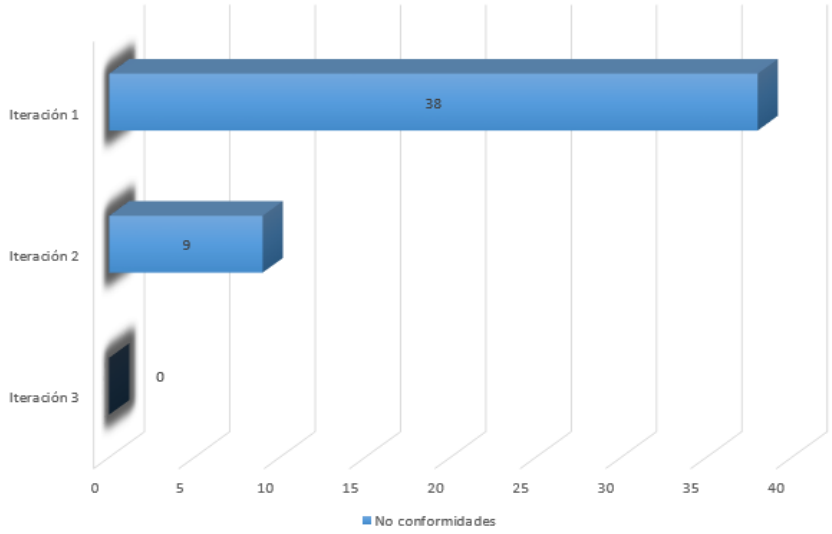

Figura 25. Resultados de pruebas internas (caja negra)

2.7. Conclusiones parciales

Luego de terminado el presente capítulo se puede concluir especificando que:

- ✓ Se detallaron los requisitos del Módulo Registro de Empresas Importadoras y Exportadoras, tanto funcionales como no funcionales, acorde a cada una de las necesidades definidas por la Cámara de Comercio.

- ✓ Se resumieron las técnicas para la validación de requisitos aplicadas en la fase de Requisitos, demostrando que la especificación es correcta, sin ambigüedades y están acordes a las necesidades del cliente.

- ✓ La utilización de métricas para la validación del diseño aplicadas en la fase de Análisis y Diseño, demostró que el diseño es adecuado, de manera general se diseñó correctamente la aplicación.

- ✓ Las pruebas realizadas a la codificación permitieron que el producto fuera entregado a los especialistas de calidad del centro de la facultad y a los de CALISOFT, con la menor cantidad posible de defectos.

- ✓ El diseño del modelo de datos permitió conocer las relaciones existentes entre las diferentes tablas de la base de datos.

- ✓ El diagrama de clases del diseño permitió conocer la estructura de cada una de las clases de la solución, con sus relaciones y funcionalidades.

- ✓ Los diagramas de despliegue y de componentes facilitaron la fase de implementación con cada una de sus especificidades.

- ✓ La utilización de los estándares de codificación lograron uniformidad en la implementación de la solución y mejoraron la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo.

- ✓ La seguridad del sistema se estableció a partir de los diferentes mecanismos propuestos por el marco de trabajo Kairos, demostrando la fortaleza del módulo.

CAPÍTULO 3

VALORACIÓN DE LA SOLUCIÓN

3.1. Introducción

En el presente capítulo se refleja todo lo referente a las pruebas externas realizadas a la solución por el grupo de calidad de CEGEL y los especialistas de CALISOFT, mostrando los resultados obtenidos en cada caso. Ambos centros realizaron pruebas funcionales o caja negra, como parte de las pruebas de liberación y para ello utilizaron los CP elaborados por el equipo de desarrollo. También se realiza una valoración de las variables de la investigación definidas en el marco teórico y se explica la aplicación de una entrevista a especialistas de la Cámara de Comercio para en un final comprobar que se resuelve la problemática planteada a partir de la solución propuesta.

3.2. Pruebas de liberación

3.2.1. Pruebas aplicadas por grupo de calidad de CEGEL

El grupo de especialistas de CEGEL a partir de los casos de pruebas elaborados, aplicaron pruebas de caja negra o de funcionalidad, explicadas en el capítulo anterior. Como resultado de estas pruebas se detectaron un total de 104 NC. De estas, 24 no procedieron y el resto (80) fueron resueltas, de las cuales 22 eran de CP y 58 de Aplicación (App).

En la primera iteración fueron detectadas 71 NC representando el 68% del total, de ellas 19 no procedieron y el resto (52) fueron resueltas, de las cuales 15 eran de CP y 37 de App. En la segunda iteración fueron encontradas 21 NC representando un 20% del total, de ellas 4 no procedieron y el resto (17) fueron resueltas, de las cuales 3 eran de CP y 14 de App. En la tercera y última iteración se detectaron 12 NC representando un 12% del total, de ellas 1 no procedió y el resto (11) fueron resueltas, de las cuales, 4 eran de CP y 7 de App, quedando resueltos el 100% de los errores encontrados. Para visualizar los resultados ver *Figura 25. Resultados de aplicar pruebas por parte del grupo de calidad de CEGEL*.

Figura 26. Resultados de pruebas por parte de CEGEL

Una vez concluidas estas pruebas se emitió un acta de liberación por parte del centro, validando que el sistema cumple con el mínimo de calidad requerida para ser usado por el cliente, esta acta fue entregada al jefe de proyecto.

3.2.2. Pruebas aplicadas por CALISOFT

El Centro Nacional de Calidad de Software (CALISOFT) brinda el servicio de evaluación de productos enfocado en la ejecución de pruebas que evalúen las características planteadas en la Norma ISO 9126 para asegurar que el software posee la calidad necesaria para ser utilizado. Este centro cuenta con un competente grupo de coordinadores que llevan a cabo el proceso de evaluación y la selección de las pruebas que el cliente desea que se le realicen a su aplicación. El objetivo de esta evaluación del producto es contribuir a que funcione como el cliente espera, que sea seguro y usable, lo cual se traduce en la mejor calidad.

Al módulo se le aplicaron las pruebas funcionales o caja negra, por parte de CALISOFT, los resultados se encuentran registrados en el gestor de proyectos: Redmine. Estas pruebas se enfocaron en los requisitos funcionales de la aplicación y se obtuvo como resultado un total de 13 NC durante 3 iteraciones completas. En la primera iteración de prueba se encontraron 8 NC entre ellas 4 de funcionalidad, 4 de formato, 1 de validación y 1 de error de interfaz. En la segunda iteración se detectaron 3 NC y las tres de funcionalidad y en la tercera iteración un total de 2 de las cuales 1 fue de funcionalidad y la otra de validación, todas fueron corregidas para un 100% de NC resueltas. Para visualizar los resultados ver *Figura 26. Resultados de aplicar pruebas por parte de CALISOFT*.

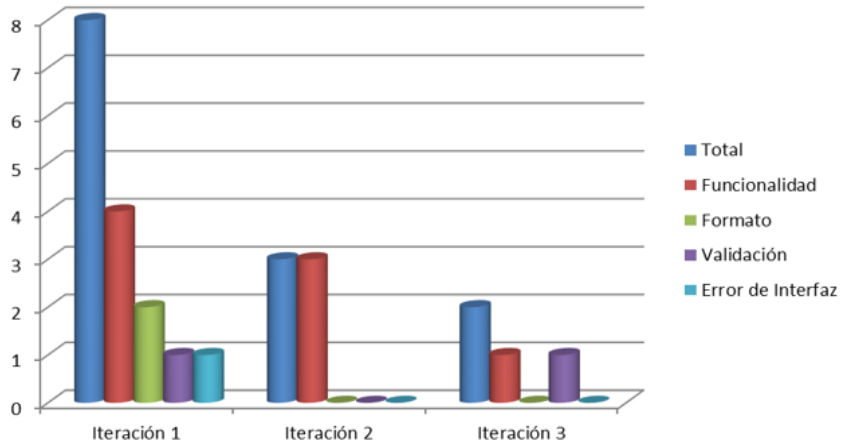

Figura 27. Resultados de pruebas por parte de CALISOFT

Luego de verificar que todas las NC fueron resueltas, el centro emitió un *Acta de Liberación* del producto, validando que cumple con el mínimo de calidad necesaria para ser usado por el cliente, la cual fue entregada al jefe de proyecto.

Las pruebas de aceptación se realizaron el 20/05/2014, los participantes fueron los desarrolladores y el cliente, en este caso especialistas de la Cámara de Comercio. En estas se comparó el producto final con las necesidades del cliente, analizando si el módulo satisface los requisitos aprobados al comienzo del desarrollo, el cliente estuvo satisfecho al 100% con el producto muestra de esto es el *Acta de Aceptación* firmada por la entidad y entregada al jefe de proyecto.

3.3. Valoración de las variables de la investigación

En el marco teórico de la investigación se definieron como variables:

- ✓ **Variable independiente**: Módulo Registro de importadores y exportadores.
- ✓ **Variable dependiente**: disponibilidad de la información.

Para la **disponibilidad de la información**, (Belt Ibérica, S.A., 2013) ofrece recomendaciones en la gestión de la seguridad de la información y enfatiza en los principios básicos de la seguridad de los sistemas de información. En este sentido define que la disponibilidad: "*Asegura que el acceso a los datos o a los recursos de información por personal autorizado se produce correctamente y en tiempo. Es decir,*

la disponibilidad garantiza que los sistemas funcionan cuando se les necesita". Enmarcado en los procesos que se realizan en la Cámara de Comercio, se entiende por disponibilidad que: la información correcta sea accedida por el personal autorizado en el tiempo requerido.

A partir de esta interpretación se establecieron como dimensiones **personal autorizado** y **tiempo requerido**, cada una con sus indicadores:

Personal autorizado:

- ✓ Acceso a la información

Tiempo requerido:

- ✓ Tiempo de respuesta de las consultas realizadas

Para la valoración del acceso a la información se tienen en cuenta los mecanismos de seguridad que nos proporciona el marco de trabajo Kairos, para más detalle consultar el *Capítulo 2. Propuesta de Solución, epígrafe 2.5.7 Seguridad en la aplicación*. Además se realizó una comparación del tiempo de respuesta de las consultas realizadas y el tiempo especificado en el RnF 7, el cual plantea que: "*el tiempo de respuesta brindado por el sistema será menor de 3 segundos*". Con ayuda de la herramienta Netbeans se obtuvo un tiempo menor de un segundo para el tiempo de respuesta obtenido a la hora de consultar información, como muestran los resultados se cumplió satisfactoriamente el RnF 7. De esta manera se puede concluir que la propuesta contribuye al aumento de la disponibilidad de la información.

En el caso de la variable **Módulo Registro de importadores y exportadores** se establecieron como dimensiones **alcance** y **calidad**, cada una con sus respectivos indicadores.

Alcance:

El alcance de software no es más que la especificación de los requisitos del software estableciendo las metas y objetivos del mismo, describiéndolos en el contexto del sistema basado en computadora. (Pressman R., 2001)

En base a esta definición, dada por Roger Pressman, el alcance de la solución se estimó a partir del indicador: porciento de requisitos funcionales implementados. Para más información sobre los requisitos funcionales ver *Capítulo 2: Propuesta de Solución, epígrafe 2.2. Requisitos Funcionales*.

El Porciento de requisitos funcionales implementados (PR) se determina a partir de:

- ✓ Cantidad de requisitos funcionales implementados (CRI=21)
- ✓ Total de requisitos funcionales (TR=21)

PR=CRI/TR*100

PR=21/21*100

PR=100 %

A partir de los resultados obtenidos se afirma que fueron implementados el 100% de los requisitos funcionales, asegurando que la solución satisface todas las funcionalidades requeridas por el cliente. Se concluyó que el sistema tiene un alcance total sobre los requisitos propuestos.

Calidad:

Para medir la calidad se definió el indicador **calidad interna**, que se determinó a partir de las pruebas internas que aparecen descritas en el *Capítulo 2: Propuesta de Solución, epígrafe 2.6. Pruebas internas* y el indicador **calidad externa** asegurado a partir de las pruebas de liberación aplicadas que aparecen en el *Capítulo 3: Valoración de la Solución, epígrafe 3.2. Pruebas de liberación*.

A partir de todas las valoraciones realizadas se aplicó una entrevista a dos especialistas de la Cámara, específicamente del departamento Dirección Jurídica, por ser los usuarios finales de la aplicación. Donde se obtuvieron resultados satisfactorios ya que el 100% calificaron los indicadores con la máxima categoría de alta, lo que indica que se cumplen en su totalidad luego de obtenido el módulo, estos resultados pueden ser visualizados en la *Figura 27. Resultados encuesta realizada*. Para más información dirigirse al *Anexo 8. Entrevista para validación de las variables de la investigación.*

Figura 28. Resultados entrevista realizada

3.4. Conclusiones parciales

En el presente capítulo se puede concluir diciendo que:

- ✓ Se aplicaron pruebas de liberación por parte del grupo de calidad de CEGEL, obteniendo el 100% de NC resueltas, y a partir de estos resultados se emitió un acta de liberación por parte del centro.
- ✓ Especialistas de CALISOFT aplicaron pruebas de liberación, obteniendo el 100% de NC resueltas, y a partir de estos resultados se emitió un acta de liberación por parte de CALISOFT.
- ✓ Se realizaron pruebas de aceptación por parte de los desarrolladores y el cliente, donde estuvo satisfecho al 100% con el producto obtenido, emitiéndose un acta de aceptación.
- ✓ Se realizó una valoración de las variables de la investigación, donde se pudo concluir que con el desarrollo del Módulo Registro de importadores y exportadores para el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba se contribuyó al aumento de la disponibilidad de la información y se comprobó en un final que la solución propuesta resuelve la problemática planteada.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se puede concluir que:

- ✓ Se analizaron herramientas y lenguajes donde se decidió para la realización del presente trabajo utilizar el modelo de desarrollo Programa de Mejora propuesto por la UCI y el marco de trabajo Kairos, como lenguaje de modelado UML 2.0 y como herramienta CASE Visual Paradigm 8.0, como servidor de bases de datos PostgreSQL en su versión 9.3, el entorno de desarrollo integrado NetBeans 7.2 y servidor de aplicaciones Glassfish 3.1.2.

- ✓ Se implementó satisfactoriamente el Módulo Registro de Importadores y Exportadores para el proyecto SIRECC, abarcando las etapas establecidas por el Programa de Mejora y acorde al diseño realizado y a las necesidades definidas con el cliente.

- ✓ Se validó el diseño realizado con la utilización de las métricas FCM, AECO y APH, aplicadas en la fase de Análisis y Diseño, las cuales demostraron que el diseño es adecuado. A partir de los resultados obtenidos en la aplicación de APH se puede afirmar que las clases no son complejas, atendiendo a su diseño y potencial de reutilización. En cuanto a AECO se puede asegurar la mantenibilidad y reusabilidad del diseño propuesto y FCM asegura una alta cohesión.

- ✓ Las pruebas internas y de liberación realizadas tanto por el equipo de desarrollo internamente, como por el grupo de calidad de CEGEL y CALISOFT, arrojaron resultados satisfactorios, el 100 % de las NC encontradas fueron corregidas, asegurando que el producto cuenta con la calidad necesaria para ser entregado al cliente.

- ✓ Se realizó una valoración de las variables de la investigación, donde se puede concluir que con el desarrollo del Módulo Registro de importadores y exportadores para SIRECC se contribuye al aumento de la disponibilidad de la información y se comprobó en un final que la solución propuesta resuelve la problemática planteada.

RECOMENDACIONES

A partir de los resultados de la investigación efectuada y de la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar el posterior seguimiento de la propuesta, se exponen seguidamente algunas recomendaciones:

- ✓ Continuar con el mejoramiento del módulo a partir de nuevos cambios que puedan surgir de acuerdo a nuevas necesidades de los clientes.
- ✓ Reutilizar componentes del módulo en aplicaciones registrales.
- ✓ Realizar las pruebas pilotos, como parte de la puesta en marcha del software en el entorno real del cliente.

BIBLIOGRAFÍA

A. Sahraoui, Houari , Godin, Robert y Miceli, Thierry . *Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?* Universidad de Montreal, Canadá : s.n.

ABPRO. 2014. ABPRO. *ABPRO.* [En línea] 2014. [Citado el: 5 de diciembre de 2013.] http://www.abpro.com.

Belmonte Hernández, Oscar. 2005. *Introducción al lenguaje de programación Java.* 2005.

Belt Ibérica, S.A. 2013. Belt Ibérica, soluciones de seguridad global. *Belt Ibérica, soluciones de seguridad global.* [En línea] Belt Ibérica, S.A., 2013. [Citado el: 22 de mayo de 2014.] http://www.belt.es.

CALISOFT. 2010. Centro Nacional de Calidad de Software. *Centro Nacional de Calidad de Software.* [En línea] 2010. [Citado el: 13 de 5 de 2014.] https://calisoft.uci.cu.

Camacho, Erika, Cardeso, Fabio y Núñez, Gabriel. 2004. *Arquitectura de Software. Guía de Estudio.* 2004.

Cámara de Comercio. 2009. El Portal del Empresario en Cuba. Cámara de Comercio de la República de Cuba. *El Portal del Empresario en Cuba. Cámara de Comercio de la República de Cuba.* [En línea] 2009. [Citado el: 28 de 1 de 2014.] http://www.camaracuba.cu/.

CEGEL. 2013. Suite de Gestión de Proyectos (GESPRO 13.05). Laboratorio de Investigaciones en Gestión de Proyectos, UCI. *Suite de Gestión de Proyectos (GESPRO 13.05). Laboratorio de Investigaciones en Gestión de Proyectos, UCI.* [En línea] 2013. [Citado el: 28 de 1 de 2014.] http://gespro.cegel.prod.uci.cu/.

de Producción, Dirección Técnica. 2011. *Estándar de codificación para Java.* La Habana, Cuba : s.n., 2011.

Dirección Jurídica. 2010. *Manual de la Dirección Jurídica.* La Habana : s.n., 2010.

Hernández León, Rolando ALfredo y Coello González, Sayda. 2011. *El Proceso de Investigación Científica.* Ciudad de La Habana : Editorial Universitaria, 2011.

IEEE. 1999. IEEE Advancing Technology for Humanity. *IEEE Advancing Technology for Humanity.* [En línea] 1999. [Citado el: 6 de 2 de 2014.] http://www.ieee.org.

IMAR Sistemas. 2013. IMAR SISTEMAS. *IMAR SISTEMAS.* [En línea] 2013. [Citado el: 4 de diciembre de 2013.] http://www.imarsistemas.com.

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

Keith, Mike y Schincariol, Merrick . 2009. *Pro JPA 2. Mastering the Java Persistence API.* United States of America : Editorial Board, 2009.

Larman, Craig. 2000. *UML y Patrones. Segunda Edición. Introducción al análisis y diseño orientado a objetos.* Ciudad de México : Prentice Hall, 2000.

Matos Rodríguez , Lenis y Sarmiento Sastrique, Daniel . 2012. *Propuesta de un mecanismo de seguridad para el marco de trabajo Kairos.* La Habana : s.n., 2012.

Meneses, Rafael Gustavo. 2012. *Persistencia Java Persistence API – JPA 2.0.* Bogotá : s.n., 2012.

Montenegro, Andrés. 2013. EAP+Blog. Reflexiones sobre el gobierno digital: ventajas y retos para el estado costarricense. *EAP+Blog. Reflexiones sobre el gobierno digital:*

ventajas y retos para el estado costarricense. [En línea] 2013. [Citado el: 28 de 1 de 2014.] http://eapucr.blogspot.com/2013/05/reflexiones-sobre-el-gobierno-digital.html.

Oracle Corporation. 2013. Oracle. *Oracle.* [En línea] Oracle, 2013. [Citado el: 5 de diciembre de 2013.] http://www.oracle.com/technetwork/java.

—. 2013. Oracle Corporation. NetBeans IDE. The Smarter and Faster Way to Code. *Oracle Corporation. NetBeans IDE. The Smarter and Faster Way to Code.* [En línea] 2013. [Citado el: 29 de 1 de 2014.] https://netbeans.org/..

ORACLE. 2013. ORACLE. GlassFish - World's first Java EE 7 Application Server. *ORACLE. GlassFish - World's first Java EE 7 Application Server.* [En línea] 2013. [Citado el: 4 de 2 de 2014.] https://glassfish.java.net.

Pardo Barrios, Yadira y Pérez González, Yudier. 2009. *Análisis del Módulo Conexión con Cajas de SENTAI.* Ciudad de La Habana : s.n., 2009.

PostgreSQL Group. 1996. The PostgreSQL Global Development Group. PostgreSQL. The world´s most advanced open source database. *The PostgreSQL Global Development Group. PostgreSQL. The world´s most advanced open source database.* [En línea] 1996. [Citado el: 5 de 2 de 2014.] http://www.postgresql.org/.

Pressman, Roger S. 2005. *Ingeniría de Software. Un Enfoque Práctico. Sexta Edición.* s.l. : McGrawHill, 2005.

Prieto Alvarez, Julio Cesar. 2013. *Proyecto Técnico SIRECC.* La Habana : s.n., 2013.

Reynoso, Carlos y Kiccillof, Nicolás . 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires : s.n., 2004.

Rodriguez Cruz, Yunier y Galán Domínguez, Esther. 2008. *La inteligencia organizacional: necesario enfoque de gestión de información y del conocimiento.* La Habana : s.n., 2008.

Sánchez, Alan. 2013. Alan Sánchez. *Alan Sánchez.* [En línea] 2013. http://alanss18.wordpress.com/2013/01/25/entorno-de-desarrollo-integradoide/.

Serra Manchado, David. 2010. *Estudio del servidor de aplicaciones Glassfish y de las aplicaciones J2EE.* 2010.

Sommerville, Ian. 2005. *Ingeniería del Software. Séptima Edición.* Madrid, España : Pearson Educación, 2005.

Suárez García, Carlos Alejandro. 2013. *Diseño e Implementación de un módulo de integración para la Solución Tecnológica.* 2013.

UCI. 2012. UCI, Portal de la Universidad de las Ciencias Informáticas. . *UCI, Portal de la Universidad de las Ciencias Informáticas. .* [En línea] 2012. [Citado el: 28 de 1 de 2014.] www.uci.cu.

Vázquez Escudero, Pedro Jesús, Moreno García, María N. y García Peñalvo, Francisco J. 2002. *Informe Técnico. MÉTRICAS ORIENTADAS A OBJETOS.* Salamanca : Departamento de Informática y Automática. Universidad de Salamanca, 2002.

Visual Paradigm. 2013. Visual Paradigm. *Visual Paradigm.* [En línea] 2013. [Citado el: 28 de 1 de 2014.] http://www.visual-paradigm.com/.