Universidad de las Ciencias Informáticas

Facultad 3

Título: Módulo Agencia de Viajes 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

Autor(es): Nailin Pérez Martínez
Carlos Jiménez León

Tutor(es): Ing. Orlando Miranda Gómez
Ing. Ana Cecilia Labrador Valdés

La Habana, Cuba
Junio, 2014

Bueno es ir a la lucha con determinación, abrazar la vida y vivir con pasión. Perder con clase y vencer con osadía, porque el mundo pertenece a quien se atreve y la vida es mucho más para ser insignificante.

Charles Chaplin

Declaración de Autoría

Se declara que Nailin Pérez Martínez y Carlos Jiménez León son los únicos autores del presente trabajo y se autoriza al Departamento CEGEL de la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

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

_____ _____

Nailin Pérez Martínez Carlos Jiménez León

_____ _____

Ing. Ana Cecilia Labrador Valdés Ing. Orlando Miranda Gómez

AGRADECIMIENTOS

A mi mami por creer en mí, por apoyarme siempre en mis decisiones, por ser más que todo mi amiga, padre y maestra. Por todos los sacrificios que has hecho por mí, por estar siempre ahí cuando te necesito y sobre todo por el amor incondicional que siempre me has brindado.

A mi hermano por ser mi ejemplo a seguir, por darme ánimos para decidirme a coger la carrera que me gustaba, por darme siempre el calor de hermano, padre, amigo, cómplice. Muchas gracias por tu cariño, apoyo y confianza.

A mi sobrinito Renier y primita Dreila por brindarme tanto amor y cariño cuando me ven, por esperarme en cada pase, por hacerme sentir una persona especial cuando están conmigo. A mi abuelita linda por su amor, por hacerme reír y por su confianza. A mis primas Yanet y Yaqui por apoyarme, quererme y siempre estar atentas de mis problemas. A mí cuñada Yili por cuidar tan bien de los dos hombres que más quiero, por hacer tuyos mis problemas y por el apoyo que siempre me has dado. A mis tíos y primas de La Habana por estar siempre pendientes de mí, por su preocupación y por su cariño. A toda mi familia que siempre me brindó amor y apoyo.

A Robe, sus padres y familia por estos años de amor, confianza, por apoyarme y sobre todo por hacerme sentir parte de su familia.

A mis amigos Yoendrys y Leidis, por ser más que amigos mi familia de la UCI, por estar conmigo en cada momento de alegría, estrés, llanto, fiesta, en fin por ser tan especiales. A Katy, Leandro y Susel por todo el cariño y confianza que me brindaron.

A mi tutor y amigo Orlando, por ser mi paño de lágrimas, por aguantar mi estrés y malacrianzas, por sus horas de dedicación y empeño, por siempre estar cuando lo necesito. A mi tutora Anita por su experiencia brindada, las horas dedicadas de trabajo, por la ayuda y consejos.

A mi compañero de tesis Carlos por haberme pedido realizar este sueño junto a él, sin ti no hubiera sido lo mismo.

A mis amigos de Ciego principalmente Darita, The Fantastic Four, Yordy y Delkita; a los que conocí aquí, a los que ya son ingenieros, gracias a todos por el apoyo y cariño. A mi grupo, compañeros del proyecto SIRECC y del laboratorio 315, por estos años que hemos compartido.

A todas las personas que hicieron posible realizar este sueño.

Nay

AGRADECIMIENTOS

Mi agradecimiento más profundo a mis padres que siempre me han apoyado a lo largo de la vida, por creer en mí, por su apoyo incondicional, por todo su sacrificio y su infinito amor, gracias por su dedicación. A mi mamá por ser lo más bello y especial que tengo en la vida, porque me apoyó en todo momento con el único objetivo de convertirme en un profesional, le agradezco a la vida y a dios por tener una madre como tú, muchas gracias por todo tu amor. A mi papá por haberme dado de su inteligencia y ese ejemplo de padre excepcional, siempre he sabido que has luchado para darme lo mejor que has podido, por eso de esta manera hoy te lo agradezco desde el fondo de mi corazón, te quiero mucho. A mi hermano por ser muy importante para mí, por ser un modelo a seguir en la vida. Gracias por todas las veces que me has defendido y lo sigues haciendo, eres mi pilar más fuerte en aquellos momentos en los que necesito un aliento, un apoyo, perdón por las veces que me he puesto bravo contigo, no lo he hecho por malicia, sino simplemente por inocencia de mi corta edad. Te quiero hermano, y podrá pasar el tiempo y quedar guardadas las palabras pero nunca cambiara el amor que siento por ti y mi eterno agradecimiento por estar ahí. Gracias por tu amor y dedicación. A mi madrina por estar siempre presente en los momentos malos y buenos de mi vida, espero que la vida te llene siempre de bendiciones ya que eres una persona muy especial para mí. Lo mejor que le pude ocurrir a una persona es poder contar con alguien como tú. Gracias por ser quien eres, por transmitirme esa paz. Gracias por llegar a mi vida y por ser esa luz que tanto necesitaba. La vida no me alcanzará para poder agradecer todo lo que has hecho por mí. Gracias por todo tu apoyo, te quiero mucho. A mi familia en general gracias por todo su apoyo, en especial a mi abuela, mi tía Olgui, mi tía Milagro, mi prima Yily , mis primitos tatica y tatico, mi cuñada Yusi, a todos ustedes gracias por todo. También quiero agradecer a todas las personas con las cuales he tenido la dicha de compartir estos maravillosos cinco años que nunca olvidaré, a mis amigos que quiero como si fuéramos hermanos, Angel, Ernesto a ustedes gracias por estar ahí cuando yo lo necesitaba, otros amigos como Alieski, mi compañero de cuarto Anchel, Leandro, Yoendry, Leydis, Susel, Kety mi amigos de Camagüey el flaco, Reymer, los del proyecto SIRECC Reinier, Alain, Francisco, Abel, Julio, a mis tutores Anita y Orlando muchas gracias por toda su ayuda, a mi querida compañera de tesis Naylin por todo su apoyo y comprensión, en fin a todos los que me ayudaron de una u otra forma a realizar este trabajo, muchas gracias.

Charly

DEDICATORIA

A mi padre de crianza Rolando, que a pesar de no estar aquí conmigo se que se siente orgulloso de su niña por hacer sus sueños realidad.

A mi madre por ser lo más grande del mundo para mí.

A mi hermano por ser una de las personas más especiales de mi vida.

Nailin

El resultado de este trabajo, va dedicado a mis padres, por educarme, apoyarme, quererme y por haber sabido hacer de mí una mejor persona, porque siempre han confiado en mí y me han apoyado en todo momento, que son y serán siempre mi mayor bendición.

Carlos

RESUMEN

La Cámara de Comercio es una institución que permite la reinserción de la economía cubana en el mundo de las relaciones económicas internacionales, además de desarrollar el comercio interno entre las empresas nacionales. Actualmente la entidad no cuenta con un sistema informático que apoye el proceso de registro de las agencias de viajes, el cual se hace engorroso a la hora de actualizar la información, provoca demora en el tiempo de respuesta y no contiene un alto nivel de seguridad. Debido a esto se decidió el desarrollo de la aplicación de escritorio Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC), que permite aumentar la seguridad y disminuir el tiempo en la gestión de los procesos, además de mantener un mejor seguimiento y actualización de la información en la institución. Para el desarrollo de la aplicación se hizo una selección de las herramientas, tecnologías y modelo de desarrollo a utilizar. Se muestran los estándares empleados en las fases de diseño e implementación permitiendo una mejor comprensión del código y ayudar a la obtención de los artefactos necesarios para dar solución al problema planteado. Finalmente se realizaron validaciones y pruebas al módulo a través de métricas de software.

PALABRAS CLAVES

Actualizar información, Cámara de Comercio, Seguridad, Tiempo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

INTRODUCCIÓN

La creación de la Cámara de Comercio en Cuba transitó por diversas etapas desde la época colonial hasta la actualidad. Esta entidad se fundó el 10 de mayo de 1876, siendo una institución que solo representaba a los comerciantes habaneros. Con el paso del tiempo y de los diversos gobiernos, fue cambiando sus características y los intereses que representaba. El nombre oficial se renovó varias veces debido a peticiones de la asociación de comerciantes, obteniendo en 1877 el de Junta General de Comercio. Posteriormente, ese mismo año, volvió a cambiar de nombre a Cámara Oficial de Comercio, Industria y Navegación, y en 1927 tomó el nombre de Cámara de Comercio de la República de Cuba (Cámara de Comercio de la República de Cuba, 2013).

En 1963 la cámara se disolvió y en su lugar se creó otra bajo el mismo nombre, pero con estructura y fines diferentes de acuerdo con la ley No. 1091 (Ley 1091, 2013) del 1ro de febrero de ese mismo año, la cual cuenta con 12 artículos. En su contenido se muestran las funciones de la Cámara de Comercio, además de una descripción de los miembros activos y honorarios. Se mencionan y explican los órganos de gobierno y administración, los ingresos propios de la Cámara y se expresa en el último artículo que no se debe usar el nombre de Cámara de Comercio en ninguna institución privada existente, como tampoco se podrá hacer uso de los objetivos y funciones de esta ley en esa institución.

Las entidades de la Cámara se encuentran ubicadas en varias partes del mundo, ordenadas por leyes donde se tiene en cuenta la intención por las que fueron creadas y las disposiciones del comercio de cada país. Según las relaciones económicas existentes en la Cámara de Comercio de la República de Cuba, se identifica la necesidad de emplear las tecnologías de la información y las comunicaciones, requiriéndose de nuevas formas de servicios y sobre todo que las personas que intervienen en la administración de la Cámara encuentren la forma de lograr mercados provechosos (Cámara de Comercio de la República de Cuba, 2013).

Dado el compromiso social de la institución cubana de responder al crecimiento de la sociedad, con un amplio intercambio donde el país obtenga sus mejores resultados socio-económicos, se observa como misión principal promover el desarrollo de empresas cubanas asociadas a la Cámara en beneficio de la economía nacional (Cámara de Comercio de la República de Cuba, 2013).

Introducción

La Universidad de las Ciencias Informáticas (UCI) surgió como parte de los programas de la Batalla de Ideas, a partir de ella se iniciaron nuevos programas destinados a elevar el nivel cultural de la población y su calidad de vida. El objetivo era crear una Ciudad Digital Avanzada, formando un capital humano especializado. Para esto se investiga y producen software y servicios informáticos para la sociedad cubana y para el mundo. De esta forma se aporta y contribuye a un mundo mejor necesario y posible, para la Sociedad de la Información inclusiva y solidaria que este requiere (UCI, 2014).

La UCI cuenta con varios centros de producción, uno de ellos es el Centro de Gobierno Electrónico (CEGEL) que radica en la Facultad 3. Este centro se propone satisfacer necesidades de clientes gubernamentales mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad, competitividad, fidelidad y eficiencia, a partir de un personal altamente calificado (CEGEL, 2014).

Uno de los proyectos vinculados a CEGEL es el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC), que tiene el objetivo de informatizar los cinco registros de la entidad: Trámites Migratorios, Importadores y Exportadores, Afiliación de Empresas Cubanas, Sucursales Extranjeras y Agencia de Viajes que es el objetivo principal de la investigación.

Entre los procesos que se realizan en el Registro de Agencia de Viajes de la Cámara de Comercio se encuentran: Inscribir agencias de viajes nacionales e internacionales, Actualizar datos de las agencias, Renovar inscripción y Cancelar inscripción. Estos en su mayoría son realizados manualmente, solo se apoyan en herramientas informáticas como Microsoft Excel y Word provocando demora en el tiempo de respuestas. Su información se almacena en formato duro, agrupando así muchos expedientes que al encontrarse dispersos pueden ser difíciles de acceder, ocasionando pérdida de estos y guardando documentos que pueden estar desactualizados y deteriorados en el peor de los casos. Concluido el proceso de inscripción se guardan los datos en hojas de cálculo creadas en Microsoft Excel, el trabajo con esta herramienta llega a ser poco eficiente a causa de que puede mostrar solamente información básica cuando a veces es necesario conocer más datos. También presenta dificultad para actualizar periódicamente la información, actividad que puede demorar mucho tiempo. Además los datos pueden ser fáciles de modificar por cualquier persona, al no tener niveles de acceso definidos, no existe seguridad para acceder a ellos. Los documentos oficiales generados, son elaborados usando Microsoft Word, como por ejemplo las licencias de inscripción de las Agencias de Viajes Nacionales, donde los

funcionarios se demoran demasiado en teclear un documento similar a los demás de su mismo tipo. Por los motivos planteados anteriormente, el registro se vuelve inseguro, lento, además de presentar dificultades en el seguimiento y actualización de la información en la institución.

Teniendo en cuenta lo antes expuesto, surge como **problema a resolver**: El manejo de los datos en el Registro de Agencia de Viajes de la Cámara de Comercio de la República de Cuba afecta el tiempo de respuesta en el proceso y la seguridad de la información.

Se propone como **Objeto de estudio**: Sistemas registrales para el gobierno electrónico.

Quedando definido como **Campo de acción**: Gestión de los procesos de las Agencias de Viajes de la Cámara de Comercio de la República de Cuba.

De esta forma el **Objetivo general** del presente trabajo es: Desarrollar un sistema informático para gestionar los procesos del Registro de las Agencias de Viajes de la Cámara de Comercio de la República de Cuba que contribuya a disminuir el tiempo de respuesta en el proceso y aumentar el nivel de seguridad de la información.

En la siguiente investigación se propone como **idea a defender**: Con el desarrollo de un sistema informático para gestionar los procesos del Registro de Agencias de Viajes se contribuirá a disminuir el tiempo de respuesta en el proceso y aumentar el nivel de seguridad de la información.

Los **Objetivos específicos** que se persiguen en la investigación son**:**

1. Elaborar el marco teórico de la investigación correspondiente al módulo Agencia de Viajes.
2. Diseñar e implementar el módulo Agencia de Viajes.
3. Verificar la solución propuesta para evaluar el cumplimiento de los objetivos planteados en la investigación.

Para garantizar el cumplimiento a los objetivos específicos se trazaron las siguientes **Tareas de la investigación:**

- Descripción de los procesos de Agencia de Viajes de la Cámara de Comercio de la República de Cuba.

- Caracterización de las metodologías, tecnologías y lenguajes más utilizados a nivel mundial para el desarrollo de aplicaciones de escritorios.
- Obtención del diseño de clases del módulo Agencia de Viajes.
- Implementación del módulo Agencia de Viajes.
- Aplicación de pruebas de calidad de software al módulo Agencia de Viajes.
- Valoración de las variables de investigación.

Operacionalización de las variables de investigación:

Variable Independiente	Variables Dependientes	Dimensiones	Indicadores	Unidad de medida
Sistema informático para gestionar los datos en el Registro de Agencias de Viajes.		Alcance	Cumplimiento de los requisitos.	IRI=CRI/CTR
		Calidad	Interna	CNC
			Externa	CTNC
	Disminuir el tiempo de respuesta en el proceso.	Tiempo	Tiempo de respuesta en el proceso.	TIAVS<TIAVA TMAVS<TMAVA TCAVS<TCAVA
	Aumentar el nivel de seguridad de la información.	Nivel de Seguridad	Confidencialidad	CRNF
			Integridad	CRNF
			Disponibilidad	CRNF

Tabla 1. Operacionalización de las variables de investigación.

Quedando definido:

IRI: Índice de requisitos implementados.

CRI: Cantidad de requisitos implementados.

CTR: Cantidad total de requisitos.

CNC: Cantidad de no conformidades detectadas por el Grupo de Calidad.

CTNC: Cantidad total de no conformidades detectadas por Calisoft.

TIAVS: Tiempo de inscripción de la Agencia de Viajes en el sistema.

TIAVA: Tiempo de inscripción de la Agencia de Viajes actualmente.

TMAVS: Tiempo de modificación de la Agencia de Viajes en el sistema.

TMAVA: Tiempo de modificación de la Agencia de Viajes actualmente.

TCAVS: Tiempo de cancelación de la Agencia de Viajes en el sistema.

TCAVA: Tiempo de cancelación de la Agencia de Viajes actualmente.

CRNF: Cantidad de requisitos no funcionales que coinciden con los indicadores.

Para el desarrollo de la investigación se utilizaron los siguientes **métodos científicos**:

Métodos Teóricos:

- Análisis Histórico-Lógico: Se utilizó para el estudio de los procesos de Registro de Agencia de Viajes constatando teóricamente cómo ha evolucionado con el tiempo. Además permitió hacer comparaciones críticas de trabajos realizados anteriormente para favorecer el desarrollo de esta investigación.

- Análisis Sintético: Fue utilizado para analizar teorías, tendencias, documentos y bibliografía relacionados con el tema para así sintetizar y extraer los elementos más importantes que se refieren al objeto de estudio y que pueden ser de utilidad para la investigación. Esto sirvió como base teórica para la bibliografía.

- Modelación: Fue usado para la realización de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la solución, facilitando así su desarrollo.

Métodos Empíricos:

- Medición: Se evidencia en la aplicación de métricas de calidad y la realización de pruebas que aseguren la calidad de los requisitos y artefactos generados en el proceso de desarrollo de software.

- Entrevista: Se realizó con el objetivo de obtener información de interés y utilidad para comprender el funcionamiento del Registro de Agencia de Viajes de la Cámara de Comercio.

La investigación está estructurada en tres capítulos, el contenido de estos se describe a continuación:

Capítulo 1. Fundamentación Teórica: En este capítulo se abordan los aspectos teóricos y técnicos de la investigación. Se realiza un estudio sobre sistemas existentes en otros países relacionados con la información registral de Agencias de Viajes para la Cámara de Comercio. Se hace un análisis detallado de las herramientas y tecnologías que facilitarán la implementación durante el proceso de desarrollo, de acuerdo con los criterios que estén establecidos en el proyecto.

Capítulo 2. Solución Propuesta: Se abordan los elementos teóricos y técnicos relacionados con la implementación de la solución. Se exponen las funcionalidades y restricciones del sistema mediante los requisitos funcionales y no funcionales. Se muestran y describen los diagramas realizados para la solución. Además se describe la arquitectura propuesta para el desarrollo del sistema y se especifican los patrones de diseño que se utilizarán.

Capítulo 3. Análisis de los resultados: En este capítulo se explican y aplican las métricas a utilizar, se muestra el resultado de la validación a través de las pruebas de caja blanca y caja negra realizadas. Se validará la solución a través del proceso de aceptación con el cliente, avalado por la carta de aceptación para los artefactos que le fueron presentados.

CAPÍTULO 1:FUNDAMENTACIÓN TEÓRICA

Introducción

En el capítulo se describen los aspectos teóricos y técnicos a utilizar durante el desarrollo del presente trabajo. Se hace énfasis en los patrones de diseño y estilos arquitectónicos, modelo de desarrollo, lenguajes de programación y modelado, así como herramientas y tecnologías a utilizar en la implementación. También se realiza un estudio de las aplicaciones que permiten registrar Agencias de Viajes a la Cámara de Comercio en países foráneos, y se describe de manera resumida el procedimiento para el registro.

1.1. Sistemas de Gestión de Agencias de Viajes

En la actualidad existen varias aplicaciones que permiten el registro de agencias, es por esto que se realizó un estudio de sistemas existentes en países foráneos con el fin de entender sus funcionalidades.

CCL es un sistema de la Cámara de Comercio de Lima basado en una plataforma de servicios integrados, estos permiten a las empresas hacer negocios a nivel local e internacional, apoyadas con herramientas de primer nivel y a bajo costo. El proyecto y los servicios presentados están basados en cuatro pilares: apoyo para aprovechar el crecimiento del volumen de negocios, exposición comercial, comercio internacional e internacionalización y mejora del entorno empresarial o para los negocios (León, 2011).

Cámara Madrid es un portal de la Cámara de Comercio del municipio español Bilbao que permite obtener información de corte empresarial relacionados con el comercio internacional. Las acciones desarrolladas por la Cámara en el ámbito del comercio exterior tuvieron como objetivo incentivar y formar nuevo tejido exportador en la Comunidad de Madrid, mejorar la competitividad internacional de las empresas y consolidar su presencia en mercados exteriores. La aplicación cuenta con un portal de comercio exterior que ofrece a las empresas con vocación internacional la oportunidad de hacer negocios en la red y acceder a otros mercados. A través de este las empresas pueden contactar con compañías de todo el mundo (Cámara de Comercio de Madrid, 2012).

OFIVIAJE es un software para agencias de viajes que brinda la posibilidad de optimizar procesos comerciales y administrativos, es usado por muchas agencias en España y Latinoamérica. Está centrado en los sectores de Comercio y Turismo/Hotelería. Trabaja bajo la filosofía de dato único, no teniendo que volver a introducir un dato que ya haya sido introducido con anterioridad. Contiene un Expediente que puede ser de Servicios Individuales o de Grupos, donde se tiene todos los datos de los servicios, fechas, vendedor, cliente, saldo actual, importe de venta y cobros, descuentos, pasajeros, notas y reservas. Las reservas son realizadas a hoteles, pasajes –avión, barco, tren, bus–, paquetes, restaurantes, seguros, alquiles de coches y cualquier otro de tipo de reserva (OfiViaje, 2014).

TRAVELIO es la plataforma software desarrollada por Hiberus para la gestión de empresas en el sector del turismo. Dispone de módulos específicamente adaptados para la gestión de mayoristas de viajes, receptivos, agencias online, centrales de reservas, cadenas hoteleras y empresas turísticas y de servicios. Es una aplicación Web accesible desde cualquier lugar con conexión a Internet. Está sustenta de cuatro pilares: gestión de contenidos, gestión de productos, gestión de clientes, motor de e-business. Permite la gestión de un extenso abanico de tipologías de producto tales como alojamientos, transferencias, seguros, excursiones, servicios, vuelo regular y cruceros (Travelio, 2012).

GESVI es una herramienta para la gestión integral de la agencia de viajes. Permite disponer de un sistema de gestión comercial y administrativa centralizada, adaptada a las necesidades de una agencia de viajes a través de la conexión a la red desde cualquier parte del mundo. Trabaja sobre el sistema de bases de datos Oracle. Los servidores de datos y de aplicaciones se encargan de almacenar datos y de procesarlos, estando conectados entre ellos y a los usuarios mediante redes que utilizan estándares de Internet (Gesvi, 2014).

Para satisfacer las necesidades del Registro de Agencia de Viajes de la Cámara de Comercio de la República de Cuba se necesita emplear características de sistemas de escritorio que registren agencias vinculadas a Cámaras de Comercio, controlando y almacenando la información de otras entidades o empresas vinculadas. Basado en el estudio de estas aplicaciones, se tiene en cuenta que las necesidades de la Cámara de Comercio de la República de Cuba no son respondidas por estos sistemas, debido a las características propias de la Cámara y que los sistemas existentes realizan gran parte de sus servicios a través de portales web, de manera informativa fundamentalmente, además de necesitar del uso de Internet lo cual es una limitación de la entidad cubana. Los sistemas OFIVIAJE,

TRAVELIO y GESVI son aplicables solo a entidades que desarrollen actividades de reservas a hoteles, paquetes, excursiones, pasajes en tren, avión, barco y cualquier otro tipo de reserva.

1.2. Descripción del Proceso Registral de Agencia de Viajes

Para realizar una explicación detallada de cada proceso, es necesario dividirlo en tres áreas fundamentales: Agencias de Viajes Nacionales, Sucursales de Agencias de Viajes Extranjeras y Contratos de Representación. En estas áreas se realizan prácticamente las mismas operaciones, las cuales involucran a las partes interesadas que serían las Agencias de Viajes, la Dirección Jurídica de la Cámara de Comercio y al Ministerio del Turismo (MINTUR). Las operaciones de registro de las áreas se realizan mediante el encargado del registro de la Dirección Jurídica de la Cámara de Comercio cuando las partes interesadas así lo solicitan o a solitud del MINTUR. Estos procesos fueron descritos mediante el Manual de Dirección Jurídica situado en el Expediente del Proyecto SIRECC.

1.2.1. Registro de Agencia de Viajes

El Registro de Agencia de Viajes abarca todo el flujo de actividades para almacenar toda la información referente a las Agencias de Viajes Nacionales, Sucursales de Agencias de Viajes Extranjeras y Contratos de Representación de Agencias de Viajes Extranjeras a través de la Agencia de Viajes Nacional.

Las Agencias de Viajes Nacionales y Sucursales de Agencias de Viajes Extranjeras realizan prácticamente el mismo proceso de registro, donde deben contar con una aprobación del MINTUR que autoriza su constitución, además de los documentos que deben presentar, luego se creará el expediente asociado a la agencia donde se registrarán todas las informaciones relacionadas con esta. La diferencia entre estas dos áreas es que para el registro de las Sucursales de Agencias de Viajes Extranjeras se requieren documentos que acrediten la actividad de esta en su país de origen, relaciones con empresas cubanas y extranjeras así como otros requisitos legales y económicos.

En los Contratos de Representación, se inscribe teniendo como representante una Agencia de Viajes Nacional que será quien presente los documentos necesarios para esta inscripción ante la Cámara. Se revisará que se cumpla las formalidades establecidas y se acordará la inscripción, obteniendo como parte de esta operación la certificación de la inscripción.

1.2.2. Renovación de las Inscripciones

En el caso de las Agencias de Viajes Nacionales, se presenta la documentación requerida en el tiempo propuesto antes del vencimiento, se espera la aprobación por el MINTUR, y si es aprobada se registra la operación y se formula una nueva licencia, en caso de ser denegada entonces se procede a la Cancelación de la inscripción de la Agencia de Viajes Nacional la cual es descrita más adelante.

Por parte de las Sucursales de Agencias de Viajes Extranjeras, el encargado del registro comprueba en la renovación que los documentos presentados cumplan con los requisitos establecidos. En este caso la Cámara envía una comunicación al MINTUR junto al traslado de la solicitud con sus recomendaciones.

En los Contratos de Representación, la Agencia de Viajes Nacional presenta la solicitud de renovación ante la Cámara, el encargado de registro verifica la operación y emite la certificación, la cual es entregada a la Agencia de Viajes Nacional.

1.2.3. Cancelación o modificación de la inscripción

Para la Agencia de Viajes Nacional, el MINTUR es quien presentará la solicitud de cancelación ante el encargado del registro de la Dirección Jurídica; este registra la operación y se entregan los documentos constitutivos a la Agencia de Viajes si se solicita. En el caso de las Sucursales de Agencias de Viajes Extranjeras se realiza en el momento en que esta presenta la solicitud. Por último para los Contratos de Representación la solicitud de cancelación la presenta la Agencia de Viajes Nacional, o la operación se realiza como parte de las funciones del encargado del registro cuando se ha vencido el término establecido en la inscripción o renovación anterior.

1.3. Seguridad de la información

Se puede definir como seguridad informática un grupo de métodos y herramientas propuestos para garantizar la seguridad de los bienes informáticos de una institución. La información es un activo muy importante, por tanto es necesario que se mantenga segura. La seguridad de la información permite garantizar (Pfleeger, 2006):

- Confidencialidad: Solo puede acceder a la información las personas autorizadas.

- Integridad: La información puede ser modificada por las personas autorizadas de la forma autorizada.
- Disponibilidad: La información está disponible para las personas autorizadas en el momento que lo requieran.

El proceso de Registro de Agencia de Viajes se realiza actualmente careciendo de un alto nivel de seguridad debido a algunos de los problemas planteados en la problemática de la investigación, es por esto que se hace necesario el desarrollo de un sistema que asegure una mayor seguridad, basándose en los elementos antes expuestos.

1.4. Modelo de desarrollo

Para la realización del sistema SIRECC es seleccionado como modelo de desarrollo el Programa de Mejora, el cual muestra 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 (Laboratorio de Gestión de Proyectos, 2013).

Este modelo contiene un grupo de buenas prácticas que ayudan a las organizaciones a mejorar sus procesos. El programa explota las características fundamentales de RUP (*Rational Unified Process, Proceso Racional Unificado*), basado en requisitos. Una de las utilidades que presenta es que es orientado a componentes debido a que se desarrolla teniendo en cuenta la estructura de agruparlos. Es adaptable al cambio y a las características específicas de cada proyecto, siendo un modelo iterativo incremental que además está centrado en la arquitectura (Laboratorio de Gestión de Proyectos, 2013).

Como parte del Programa de Mejora se define un ciclo de vida, el cual permite una mejor realización del proyecto de desarrollo. En este ciclo se pueden realizar tantas iteraciones según las necesidades del proyecto a partir de la fase de negocio, además de tener un tiempo de soporte limitado. Para la realización de esta investigación, el ciclo de vida es realizado a partir de la etapa de Análisis y Diseño hasta el Despliegue. La etapa de soporte es opcional para todos los proyectos (Programa de Mejora, 2008-2010). A continuación se muestra las fases del ciclo de vida:

- Estudio preliminar
- Modelación del negocio
- Requisitos

- Análisis y Diseño
- Implementación
- Pruebas Internas
- Pruebas de Liberación
- Despliegue
- Soporte

Cada una de estas fases contiene un conjunto de artefactos que son necesarios para el desarrollo de un sistema de software con la calidad requerida. También contiene una lista de herramientas que son utilizadas en el proceso de desarrollo, estas serán explicadas y caracterizadas en la continuación de este capítulo.

1.5. Lenguajes y herramientas para el desarrollo de la solución.

1.5.1. Lenguaje de modelado del sistema

Para la realización de los diferentes diagramas propuestos en el modelo de desarrollo seleccionado, se utiliza el estándar para el modelado de software UML (*Unified Modeling Language, Lenguaje Unificado de Modelado*). Mediante su uso permite entender, diseñar, mantener y controlar la informar del sistema. Se puede emplear para un mejor entendimiento de los artefactos del sistema, permitiendo poder especificarlos y construirlos. Contiene elementos que posibilita una mejor organización del modelo, separándolo por paquetes, logrando particionar grandes sistemas en partes pequeñas con sus respectivas dependencias (Ivar Jacobson, 2000).

1.5.2. Herramienta CASE

Las herramientas CASE (*Computer-Aided Software Engineering, Ingeniería de Software asistida por Ordenador*) favorecen un conjunto de métodos, utilidades y técnicas que brindan ayuda a los analistas, ingenieros de software y desarrolladores, durante todo el ciclo de vida de desarrollo de un software. Desde principio de la década de 1990 los analistas empezaron a beneficiarse de las herramientas de productividad, denominadas herramientas CASE, que fue creada para mejorar el trabajo habitual mediante apoyo automático. Se utiliza esta herramienta desde el principio hasta el fin del ciclo de vida debido a que incrementa la productividad, permite comunicarse de manera más eficiente con los

usuarios a través de diagramas que se generan, principalmente diagramas de procesos que son los más utilizados por las entidades (Kendall, 2005).

1.5.2.1. Visual Paradigm

La herramienta CASE seleccionada para modelar los diagramas del modelo utilizado es Visual Paradigm (VP) en su versión 8.0. Soporta el ciclo de vida completo de desarrollo de software, además de brindar soporte multiplataforma, confidencialidad y estabilidad en el desarrollo orientado a objeto. Genera código para Java, que es el lenguaje seleccionado para el desarrollo y permite la ingeniería inversa. La herramienta es compatible con otras ediciones, contiene internacionalización, permite crear modelos que permanecen sincronizados en todo el ciclo de desarrollo y los diagramas generados son fáciles de actualizar. Permite la ingeniería inversa de base de datos desde Sistemas Gestores de Bases de Datos (SGBD) existente a diagramas de Entidad-Relación y la generación de bases de datos para la transformación de diagramas de Entidad-Relación en tablas de base de datos (Visual Paradigm, 2013).

1.5.3. Marco de trabajo

El marco de trabajo utilizado es Kairos, fue desarrollado por el Centro de Gobierno Electrónico (CEGEL) perteneciente a la UCI, facilita a las aplicaciones que utilicen funcionalidades y componentes que simplifican su desarrollo. Kairos provee un alto nivel de configuración y adaptabilidad, lo que posibilita que sea un marco de trabajo altamente reusable. Se basa en la plataforma JEE (*Java Enterprise Edition, Ediciones Empresariales de Java*) y propone la arquitectura Cliente-Servidor. Es por esto que cuenta con dos segmentos, uno que se encuentra en el lado del cliente facilitando la creación de la capa de presentación y la comunicación con los objetos remotos desplegados en el servidor, y el segundo que se encuentra en el lado del servidor donde se encuentra el acceso a datos y la lógica de negocio. Cada uno de estos segmentos está administrado por los estilos arquitectónicos N-Capas, Basado en Componente y Programación Orientada a Objetos. Se considera que este marco de trabajo contribuirá a disminuir el tiempo de desarrollo en el proyecto. Para obtener esta información, se hizo a referencia a los siguientes trabajos de diploma: (García, 2011), (Campos, 2011), (Valdés, 2011).

1.5.4. Entorno de desarrollo integrado

Un IDE (*Integrated Development Environment, Entorno de Desarrollo Integrado*) es un programa informático o de aplicaciones, compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los lenguajes IDE pueden funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Se caracterizan por ser multiplataforma, soportan varios lenguajes de programación, permite importar y exportar proyectos, además de permitir trabajar con idiomas (Oracle Corporation, 2013).

1.5.4.1. NetBeans 7.2

Para el trabajo con la plataforma seleccionada se utiliza el IDE NetBeans en su versión 7.2. Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. Es un producto libre y gratuito sin restricciones de uso. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Los módulos pueden ser desarrollados independientemente, por tanto las aplicaciones basadas en IDE NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación (Oracle Corporation, 2013).

1.5.5. Lenguaje de Programación Java

El lenguaje de programación utilizado en la solución es Java, desarrollado por Sun Microsystems a principios de los años 90. 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. Es simple, orientado a objeto, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (Paredes, 2010).

1.5.6. Lenguaje de consulta JPQL

JPQL (*Java Persistence Query Language, por sus siglas en inglés*) es un lenguaje de consulta independiente de plataforma orientado a objetos, que son definidos como parte de la especificación Java Persistence API. Este lenguaje se utiliza para hacer consulta en entidades almacenadas en base

de datos relacionales. Su sintaxis es similar a la del lenguaje SQL (*Structured Query Language, Lenguaje de Consulta Estructurado*) estándar, pero adaptadas al tratamiento de objetos. Las consultas JPQL se pueden declarar de forma estática en los metadatos o construir de forma dinámica en el código (Oracle, 2010).

1.5.7. Enterprise Java Bean

EJB (*Enterprise Java Bean, por sus siglas en inglés*) es una plataforma para construir aplicaciones de negocios portables, reutilizables, escalables y seguras usando lenguaje de programación Java. Es un componente software que se ejecuta del lado del servidor en una aplicación multicapa, específicamente cuatro capas. Contienen lógica de negocio, que opera sobre los datos de la aplicación. Ofrece estándares para aplicaciones de negocios basadas en componentes, orientadas a objetos y distribuidas (Rubinger, 2010).

1.5.8. Java Persistence API

JPA (Java Persistence API, por sus siglas en inglés) proporciona un estándar para gestionar datos relacionales en aplicaciones Java EE, de forma que se simplifique el desarrollo de la persistencia de datos. Facilita un modelo de persistencia basado en POJO´s (*Plain Old Java Object, por sus siglas en inglés*), o sea, objetos simples que no heredan ni implementan otras clases para mapear base de datos relacionales en Java. Se ha combinado ideas y conceptos de los principales frameworks de persistencia como las versiones anteriores de EJB, estos cuentan actualmente con una implementación JPA (Martín, 2011).

Java Persistence API consta de tres áreas:

- El Java Persistence API: Se trata de un conjunto de clases e interfaces, que serán empleadas por la capa de negocio para operar con los objetivos persistentes.
- JPQL: Aunque muchas de las operaciones habituales con datos pueden ser realizadas a través de los métodos del API estándar JPA, la especificación incluye un lenguaje de manipulación de objetos, conocido como JPQL, con el que podemos definir operaciones complejas de tratamiento de objetos.

- El mapeo de los metadatos objeto/relacional: Representa la información que permite especificar al motor de persistencia la manera en que se deben mapear los objetos con las tablas de la base de datos.

1.6. Sistema Gestor de Bases de Datos

Un SGBD (*Data Base Management System, Sistema Gestor de Base de Datos*) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Una de las principales herramientas que tiene es la administración de usuarios, privilegios y funciones de contraseñas de usuarios, además del establecimiento de límites de recursos de la base de datos. Tiene como características (Maribel, 2012):

- Abstracción de la información: Estos ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.
- Independencia: Consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante.
- Consistencia: De esta forma se vigilar que aquella información que aparece repetida se actualice de forma coherente, o sea, que todos los datos repetidos se actualicen de forma simultánea.
- Seguridad: Deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados.
- Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados.
- Respaldo y recuperación: Deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos.
- Control de la concurrencia: Lo más habitual es que sean muchas las personas que acceden a una base de datos; ésta debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

1.6.1. PostgreSQL 9.3

El sistema de gestor de base de datos seleccionado para la solución de la implementación fue PostgreSQL en su versión 9.3, esta selección fue debido al estudio de sus características y beneficios que ofertaba. Es un sistema multiplataforma, segura, sin restricción de uso y consume pocos recursos. Debido a su licencia liberal, puede ser utilizado, modificado y distribuido por cualquier persona libre de forma gratuita para cualquier propósito, ya sea privado, comercial o académico. PostgreSQL actualmente es la base de datos más avanzada de código abierto disponible en cualquier lugar. Una de las características más importantes de este es su completitud y fiabilidad al respecto de la manutención de la integridad de los datos, soportando claves primarias y todo tipo de comprobaciones a nivel de columna y fila (PostgreSQL, 1996).

1.7. Servidor de Aplicaciones GlassFish 3.1.2

Glassfish es un servidor de aplicaciones desarrollado por Sun Microsystems para plataforma Java EE. Tiene como beneficios ayudar a los desarrolladores a entregar aplicaciones de forma más eficiente, es ligero, flexible, consume poca memoria y es rápido. Cuando se combina con NetBeans puede mejorar significativamente el desarrollo iterativo. Esto significa que en lugar de seis pasos que consumen tiempo (editar, guardar, compilar, empaquetar, desplegar, vuelva a llenar los datos de sesión), el mismo proceso es reducido a tres pasos (editar, guardar y actualizar su navegador). Además GlassFish incluye soporte multilenguaje y es disponible en varios idiomas como son español, inglés, chino, portugués (GlassFish, 2013).

1.8. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Un patrón está compuesto por los siguientes elementos (Larman, 1999):

- Nombre del patrón: Describe el problema de diseño, junto con sus soluciones y consecuencias.
- Problema: Describe cuando aplicar el patrón, explica el problema y su contexto.
- Solución: Describe elementos que forman el diseño, relaciones y responsabilidades.
- Consecuencias: Son los resultados, ventajas e inconvenientes de aplicar el patrón.

Para realizar el diseño del subsistema de desarrollo adecuadamente se usan los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns, Patrones generales de Software para asignar responsabilidades*) y los GoF (*Gang of Four, Grupo de los cuatro*).

GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones. Dentro de sus patrones se encuentran Experto, Creador, Alta cohesión, Bajo acoplamiento y Controlador. A continuación se describen (Larman, 1999):

- Experto: Consiste en asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: Este patrón 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, brinda soporte al bajo acoplamiento.
- Bajo acoplamiento: Se basa en asignar una responsabilidad para mantener bajo acoplamiento. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño.
- Alta cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta. Como el patrón Bajo Acoplamiento, también Alta Cohesión es un principio que debemos tener presente en todas las decisiones de diseño: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.
- Controlador: Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se acepten. La clase controlador debería utilizarse con los eventos sistemáticos de un caso de uso, almacenando de esta forma la información referente al estado del caso.

Dentro de los patrones GoF se seleccionaron los siguientes (Larman, 1999):

- Fachada: Este nombre lo obtiene la clase definida que ofrece una interfaz común con un grupo heterogéneo de interfaces, entre ellas la clase Modem. Las interfaces heterogéneas pueden ser un conjunto de funciones, un esquema, un conjunto de otras clases o un subsistema.
- Instancia única: Este patrón garantiza la presencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Otro de patrón utilizado fue Inyección de dependencias, este se especializa en inyectar objetos en una clase, en lugar de ser la propia clase quien cree el objeto.

1.9. Estilos arquitectónicos

"Los estilos arquitectónicos son modelos semánticos que permiten a un diseñador, mediante el análisis de las propiedades conocidas de las partes que lo integran, comprender las propiedades generales de un sistema" (Pressman, 2006). Los estilos arquitectónicos utilizados para la realización del sistema fueron el Cliente-Servidor, N-Capas y Basado en componentes.

La arquitectura **Cliente-Servidor** es un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que lo proporcionan. El cliente es el proceso que permite al usuario formular los requisitos y pasarlos al servidor. Por otro lado el servidor es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. En otras palabras, este estilo arquitectónico es una extensión de programación modular, donde la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento (Sequeira, 2010).

La arquitectura **N-Capas** establece una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la capa inmediatamente inferior. Distribuye el trabajo equivalentemente y le da potencia y rapidez al programa. El desarrollo se puede llevar a cabo en varios niveles, en caso de cambios, solo se afecta al nivel requerido sin tener que revisar todo el código (Gaibor, 2008).

La arquitectura **basada en componentes** se caracteriza por ser un estilo de diseño para aplicaciones compuestas de componentes individuales. Se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación 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. Son reusable, los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas. También tienen como principio que son independiente, extensibles y encapsulados (Peláez, 2009).

1.10. Pruebas de software

"Las pruebas de software son un elemento crítico para la garantía de calidad de software y representa una revisión final de las especificaciones, diseño y codificación" (Pressman, 2006).Estas pruebas se

realizan una vez concluida la implementación, para verificar si el resultado del sistema es el esperado. Para esto se comprueba la lógica interna del software y se descubren los errores en la funcionalidad, comportamiento y rendimiento.

Para verificar el avance o comportamiento al módulo Agencia de Viajes del proyecto SIRECC, se realizan estudios a las pruebas internas, pruebas de liberación, pruebas de aceptación y pruebas piloto, apoyándonos primeramente de las pruebas de caja blanca y pruebas de caja negra:

- **Pruebas internas:** Estas pruebas son realizadas por sus propios desarrolladores, verificando el resultado de la implementación, probando cada construcción según sea necesario, así como las versiones finales a ser liberadas. Los artefactos necesarios para la realización de estas pruebas son los casos de pruebas (Programa de Mejora, 2008-2010).
- **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"* (Programa de Mejora, 2008-2010).
- **Prueba de aceptación:** El objetivo de las pruebas de aceptación es validar que un sistema cumpla con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento (Alejandro Santanach, 2012).
- **Prueba piloto:** Las pruebas piloto se realizan sobre aplicaciones que requieran un despliegue. Se realiza en un ambiente real y con el usuario final. Se utilizan principalmente para obtener las no conformidades y los cambios que necesita el sistema a medida que se está explotando en manos de usuarios reales (Alejandro Santanach, 2012).
- **Pruebas de caja blanca:** Estas pruebas son utilizadas para la lógica interna del programa, o sea busca errores a través del código interno. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con lo esperado o mencionado (Pressman, 2006). Para efectuar la prueba de caja blanca se realizan pruebas funcionales al sistema a través de la Prueba del camino básico.

  Prueba del camino básico: Mediante este método se obtiene una medida de la complejidad de un diseño procedimental, además de usarla para definir una serie de caminos básicos de

ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez (Pressman, 2006).

- **Pruebas de caja negra:** A través de estas pruebas se observan los requisitos del software, o sea se llevan a cabo sobre la interfaz de este. Aquí se examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del sistema (Pressman, 2006).

  Para desarrollar las pruebas de caja negra existen varias técnicas, entre ellas están:

  Métodos de pruebas basados en grafos: Exploran las relaciones entre los objetos del programa y su comportamiento.

  Partición equivalente: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

  Análisis de valores límite: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

  Prueba de la tabla ortogonal: Suministra un método sistemático y eficiente para probar sistemas con un número reducido de parámetros de entrada.

  Para el diseño de los casos de prueba considerando cada requisito se utiliza la siguiente aproximación sistemática:

  Pruebas basadas en requisitos: Mediante el uso de estas pruebas de validación el usuario desea demostrar que en el sistema sus requisitos han sido implementados correctamente. Permiten una constante aproximación al diseño de caso de pruebas, de forma que el usuario considera cada requisito y deriva un conjunto de pruebas para cada uno de ellos (Pressman, 2006)**.**

Conclusiones Parciales

Luego de realizado el estudio de la fundamentación teórica de la investigación en curso del presente capítulo se llega a las siguientes conclusiones:

- El estudio realizado de las aplicaciones similares que resuelven esta problemática en países foráneos, muestra la necesidad de crear un sistema Registral de Información para la Cámara de

Comercio de la República de Cuba, debido a que ninguno de estos sistemas cumple con las necesidades actuales de la Cámara de Comercio.

- El proceso de selección de las metodologías, tecnologías y herramientas propuestas siguiendo los criterios definidos garantizará un mejor desarrollo del sistema siguiendo el enfoque a procesos propuesto.
- La utilización de patrones de diseño y estilos arquitectónicos permitió resolver problemas concurrentes.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se muestra la propuesta del sistema, describiendo cómo debe funcionar y destacando sus características propias. Se describen los estilos arquitectónicos del sistema y los patrones de diseño a utilizar. Se enuncian algunos de los requisitos funcionales y no funcionales teniendo en cuenta su prioridad. También se muestra el modelado de diagramas de clases, componentes y de paquetes que compone el modelo de diseño del subsistema a desarrollar.

2.1 Propuesta del sistema

El módulo de Agencia de Viajes está compuesto por nueve procesos, para la propuesta de solución del sistema en el desarrollo de este capítulo, se estará abordando principalmente del proceso Registro de Agencia de Viajes Nacional. Luego del estudio del modelo de desarrollo en el capítulo anterior, se decidió realizar la investigación a partir de la fase de Análisis y Diseño hasta el Despliegue. En un trabajo defendido anteriormente (Granado, 2012) se realizaron las etapas de Estudio preliminar, Modelación de Negocio y Requisitos del módulo Registro de Agencia de Viajes. En este trabajo se identificaron un total de 41 requisitos funcionales, de ellos 12 de complejidad alta, 15 de complejidad media y 14 con una baja complejidad. A continuación se mencionan algunos de los fundamentales, para consultar los requisitos ver el *Anexo 1*.

Requisitos Funcionales

RF_1: Inscribir agencia de viajes nacional.

RF_2: Modificar agencia de viajes nacional.

RF_5: Buscar agencia de viajes nacional.

RF_10: Crear expediente.

RF_11: Modificar expediente.

RF_12: Cancelar expediente.

RF_23: Inscribir agencia de viajes extranjera.

RF_24: Modificar agencia de viajes extranjera.

RF_26: Visualizar datos de la agencia de viajes extranjera.

RF_28: Inscribir agencia de viajes extranjera representada.

RF_29: Modificar agencia de viajes extranjera representada.

RF_32: Listar agencia de viajes extranjera representada.

Requisitos no Funcionales

Se identificaron un total de 33 requisitos no funcionales donde se describen a continuación algunos de estos. Para un mejor entendimiento de los requisitos no funcionales consultar el *Anexo 2.*

Usabilidad:

- Tipo de Aplicación Informática

RnF_02: El producto constituirá una aplicación para un entorno de escritorio enfocada en las necesidades de la Cámara de Comercio de la República de Cuba.

- Finalidad

RnF_03: Este sistema estará enfocado a la gestión de información relacionada con los procesos registrales que se llevan a cabo en la Cámara de Comercio.

Confiabilidad:

RnF_09: El sistema deberá estar disponible las 24 horas de los días laborales para su funcionamiento y realización de las salvas que se deben ejecutar después de la jornada laboral (8 horas).

Eficiencia:

RnF_12: El sistema debe garantizar el acceso concurrente para todos los usuarios del sistema durante la jornada laboral establecida.

Soporte:

RnF_13: El uso del sistema requerirá un tiempo de preparación previa por los usuarios finales para su correcta explotación.

Restricciones de diseño:

RnF_15: 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:

RnF_19: El sistema presentará una interfaz legible, simple de usar e interactiva.

Requisitos de Licencia:

RnF_22: Se deben adquirir las licencias necesarias para el uso de sistemas operativos en estaciones de trabajo que requieren la digitalización de documentos, así como para software auxiliar y librerías necesarias. El software debe cumplir los siguientes requisitos de licencias que deben ser usadas. Licencia PostrgreSQL para el gestor de bases de datos.

Requisitos Legales, de Derecho de Autor y otros:

RnF_23: El SIRECC tiene que garantizar el cumplimiento de lo dispuesto en las normas jurídicas, expresadas en Ley Orgánica y Reglamento de la Cámara de Comercio de la República de Cuba, permitiendo adecuar el sistema de forma fácil, a cambios en dichas normas. Debe igualmente hacer referencia al Registro y depósito legal de la aplicación.

Estándares Aplicables:

RnF_24: Se aplicarán los estándares de java en función de facilitar el mantenimiento de la aplicación.

Seguridad:

RnF_25: El sistema podrá ser utilizado solamente por usuarios autenticados en el mismo.

RnF_26: El sistema brindará la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado.

RnF_27: El sistema mostrará las funcionalidades de acuerdo a quien esté autenticado en el mismo.

RnF_28: El sistema debe asegurar el almacenamiento de las credenciales de los usuarios utilizando algoritmos criptográficos que oculten la identidad verdadera de los usuarios.

RnF_29. El sistema debe permitir almacenar todas las acciones de los usuarios sobre el sistema como constancia de las acciones realizadas.

Requisitos de Software:

RnF_31: El servidor de aplicaciones debe tener como sistema operativo Ubuntu Server en su versión 12.0.4 y Glassfish 3.1.2. El servidor de base de datos debe tener como gestor PostgresSQL versión 9.1.

Requisitos de Hardware:

RnF_32: Proporcionar características mínimas de hardware a las estaciones de trabajo.

Las características técnicas mínimas de hardware deben ser las siguientes:

- 1 GB de RAM.
- 30 GB de disco duro.
- Adaptador de red Ethernet 100 Mbps.
- Sistema de Energía Ininterrumpida (UPS) 500 Va.

En el Análisis y Diseño se muestran los artefactos realizados en la investigación, como es el Modelo de diseño donde se visualiza el diagrama de paquetes, diagrama de clases, diagrama de despliegue y modelo de datos. Estos diagramas son presentados sin sus métodos y atributos debido a restricciones de confidencialidad con el cliente al que se le desarrolla el sistema, para visualizarlo dirigirse al Expediente de proyecto del Registro de Agencia de Viajes. También se mostrará el uso de los patrones de diseño y estilos arquitectónicos estudiados en el capítulo 1. En la fase de implementación se muestra el diagrama de componentes, además de explicar el estándar de codificación utilizado para la implementación.

2.2 Arquitectura del sistema

La solución de la investigación está basada la arquitectura Cliente-Servidor, ya que el diseño arquitectónico del marco de trabajo Kairos está enfocado a esta arquitectura. Este estilo tiene un alto nivel organizativo debido a la centralización de la gestión de la información y la separación de

responsabilidades. El cliente envía un mensaje solicitando un determinado servicio a un servidor, o sea realiza una petición y el servidor envía uno o varios mensajes de respuesta.

Para organizar mejor el diseño de estos elementos se utilizará el estilo N-Capas, específicamente se utilizarán 4 capas (Presentación, Negocio-Cliente, Negocio-Servidor, Acceso a Datos), donde se permite organizar ordenadamente cada una de estas. Garantizando que cada capa proporcione sus servicios a la capa inmediata superior y que la misma se sirva de las prestaciones que le brinda la capa inmediata inferior, como se muestra en la *Figura 1*.

En la aplicación Cliente se encuentran las capas Presentación y Negocio-Cliente, en el primer caso se manejan los formularios a través de las acciones, contiene los componentes de interfaz de usuarios. En esta capa se obtienen los datos a través de solicitudes al Negocio del Cliente. La capa Negocio-Cliente se encarga de gestionar la información del cliente y comunicarse con el servidor, para esto se apoya de los componentes Fachada y Gestores. En la aplicación Servidor las capas Negocio-Servidor y Acceso a Datos también dividen las funciones para responder a las peticionas realizadas. En la capa Negocio-Servidor se gestionan los componentes que implementan la lógica de negocio del sistema. En la capa de Acceso a Datos se maneja la información entre la aplicación servidor y la base de datos.

Otro de los estilos utilizado es basado en componentes, el mismo se enfoca en la descomposición del diseño en componentes funcionales o lógicos que tienen interfaces bien definidas. Un ejemplo de esta descomposición son las Acciones, Fachadas, Gestores, Factorías y Formularios. En la *Figura 1* se muestra visualmente la arquitectura del módulo Agencia de Viajes.

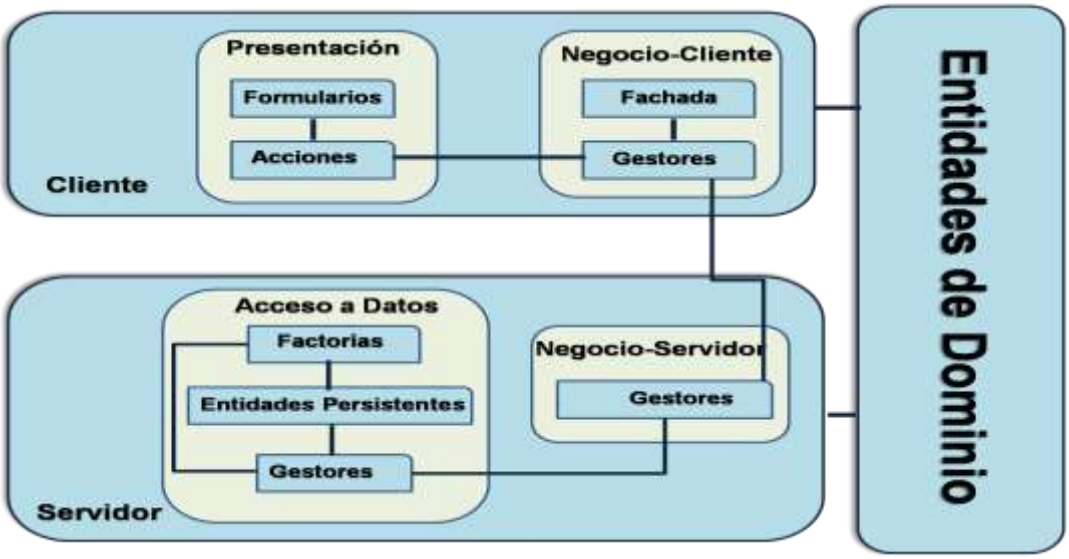


Figura 1. Arquitectura del módulo Agencia de Viajes.

2.3 Diseño del sistema

Luego del análisis realizado en la investigación del trabajo, se realiza el diseño, dejando plasmado los diagramas que cumplan con los artefactos necesarios. También se vinculan los patrones de diseño al resultado que se obtiene mediante el diagrama de clases.

2.3.1 Diagrama de paquete

El diagrama de paquete plasmado, es utilizado para reflejar la organización que existe de los paquetes y sus elementos, visualizando como el sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. A continuación se muestra el diagrama de paquetes del módulo en cuestión, representando solamente el nombre de los paquetes para una mejor visualización.

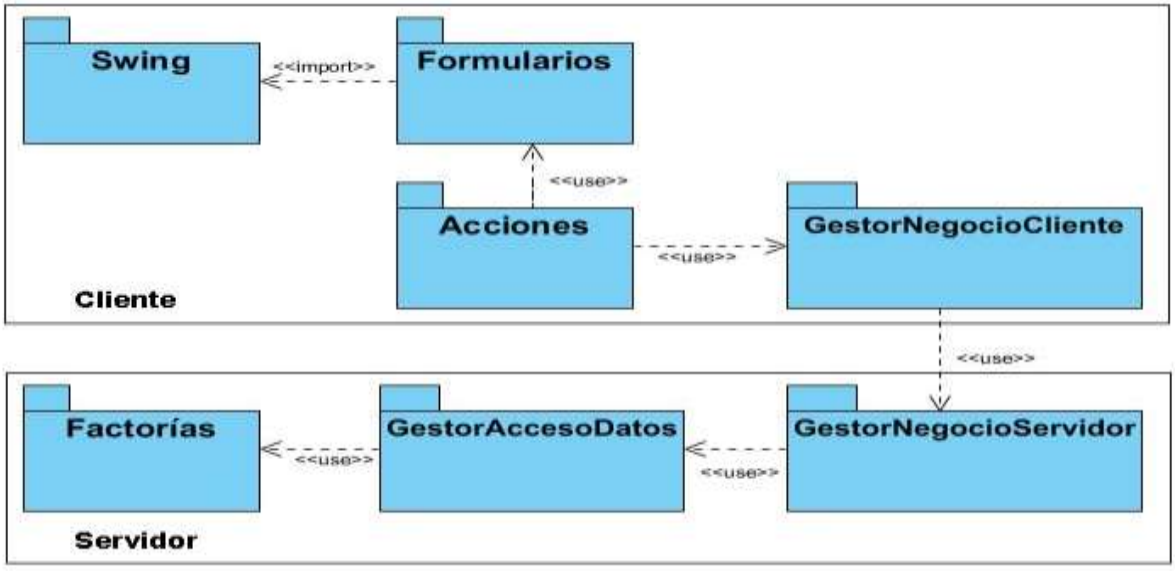


Figura 2. Diagrama de paquetes de Registro de Agencia de Viajes.

2.3.2 Diagrama de clase

Mediante el diagrama de clases del diseño se visualizan las relaciones entre las clases que involucran el sistema, representando una visión estática de este. Aquí se realiza una descripción de las categorías detallada de atributos de manera que el analista y el cliente obtengan una buena comunicación. Presentan un conjunto de clases, interfaces, colaboraciones y relaciones entre ellas. A continuación se muestra de manera resumida el diagrama de clases del Registro de Agencia de Viajes Nacional.

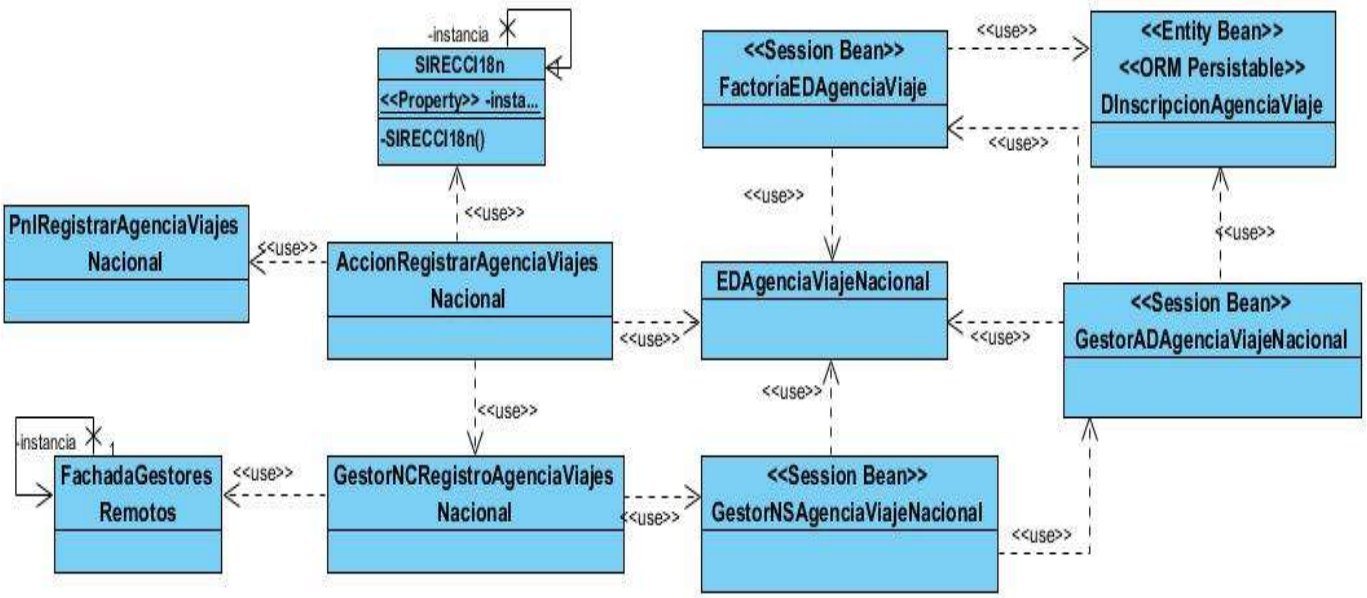


Figura 3. Diagrama de clases de Registro Agencia de Viajes Nacional.

Las clases vinculadas al diagrama son descritas a continuación:

Nombre de la Clase	Descripción
pnlRegistrarAgenciaViajeNacional	Esta clase contiene todos los elementos visuales de la interfaz, los paneles, formularios, tablas, botones, entre otros.
FachadaGestoresRemotos	Localiza y obtiene los gestores de negocio del servidor para poder realizar las funcionalidades de los gestores de negocio del cliente.
AccionRegistrarAgenciaViajeNacional	Realiza las validaciones sobre la información proporcionada por el usuario referente a la agencia de viajes nacional a inscribir y la cede al gestor de negocio del cliente correspondiente.
GestorNCRegistroAgenciaViajeNacional	Realiza las operaciones de la lógica de negocio en el cliente a partir de la información referente a las agencias de viajes proporcionada por el usuario recibida a través de la acción correspondiente.
GestorNSAgenciaViajeNacional	Realiza las operaciones de la lógica de negocio en el servidor a partir de la información referente a las agencias de viajes nacionales recibida del gestor de negocio del cliente correspondiente.

GestorADAgenciaViajeNacional	Realiza las operaciones de obtención, registro y actualización de la información referente a las agencias de viajes asociadas a la Cámara de Comercio desde la base de datos.
EDAgenciaViajeNacional	Para representar las entidades de dominio utilizadas en el módulo, las cuales contienen la información relacionada con la Agencia de Viajes.
FactoriaEDAgenciaViajes	Convierte las clases a Entidades de dominio o Entidades Persistentes.
DInscripciónAgenciaViajes	Entidad Persistente que se mapea mediante en la implementación de JPA.
SIRECC18n	Se define para cada uno de los idiomas de los nombres de los componentes visuales, es la internacionalización.

Tabla 2. Descripción de las clases de Registro de Agencia de Viajes Nacional.

2.3.3 Patrones de diseño utilizados

En el capítulo anterior se estudiaron los patrones GRASP y GoF que se utilizan en la solución, a continuación se describen cada uno de estos, haciendo referencia al diagrama de clases.

Un ejemplo de las clases donde se utiliza el patrón **Experto** es en las Entidades de Dominio, como es *EDAgenciaViajesNacional,* estas clases contienen la información necesaria para cumplir con sus responsabilidades y que nadie más posee. El patrón **Creador** se refleja en la clase *FactoriaEDAgenciaViajes*, aquí se asigna a algunas clases la responsabilidad de crear una instancia de otra clase. El patrón **Controlador** se demuestra en las Acciones, pues las peticiones realizadas por los usuarios son manejadas por estas, como por ejemplo *AccionRegistrarAgenciaViajesNacional*.

El **Bajo acoplamiento** permite mantener las relaciones mínimas entre las clases, de forma que en caso de realizar modificaciones no implique grandes cambios al resto de las clases. La **Alta cohesión** facilita las dependencias mínimas entre las clases, permitiendo de esta forma que las clases puedan ser reutilizadas.

Otro de los patrones utilizados son Fachada, Instancia única e Inyección de dependencias. En el patrón **Fachada** se localiza y obtiene los gestores de negocio del servidor para poder realizar las funcionalidades de los gestores de negocio del cliente, facilitando un punto de acceso entre varias clases, este patrón es reflejado en la clase *FachadaGestoresRemotos*.

El patrón **Instancia única** es utilizado en la clase *GestorMarcoTrabajo* que permite abstraerse de la creación de instancias.

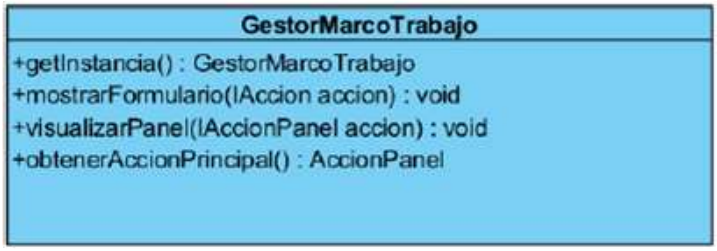


Figura 4. Aplicación del patrón Instancia única.

El patrón de **Inyección de dependencias** se especializa en inyectar objetos en una clase, en lugar de ser la propia clase quien cree el objeto.

```
@EJB
private GestorNSSistemaRemote gestorNSSistema;
```

Figura 5. Aplicación del patrón Inyección de dependencias.

2.3.4 Modelo de datos

El diseño del Modelo de datos es creado con el objetivo de comprender la estructura de la base de datos, organizando la representación abstracta de los datos de los objetos y las relaciones que existen entre ellos. A continuación se muestra el modelo de datos de Registro de Agencia de Viajes, este modelo está compuesto por 27 entidades, de ellas 10 nomencladores, uno de los nomencladores es nEstructura, a través de este se muestran las Estructuras de la Agencia de viajes representadas por identificadores, por ejemplo, idInscripciónAgenciaViajes. Para obtener la inscripción y nombramiento de las tablas de manera entendible se describen las siguientes pautas:

- El nombre de las tablas nomencladoras comenzará con la letra n por ejemplo nDireccion, mientras que las otras tablas de datos comenzará con la letra d, por ejemplo dLicencia.
- Las especializaciones se nombrarán: nombre de la tabla padre + nombre de la tabla hija.
- Para la notación de las llaves primarias se utilizará el siguiente patrón: Id + nombre de la tabla.

A continuación se muestra un segmento del modelo de datos, para ver el diagrama del modelo completo ver *Anexo 3.*

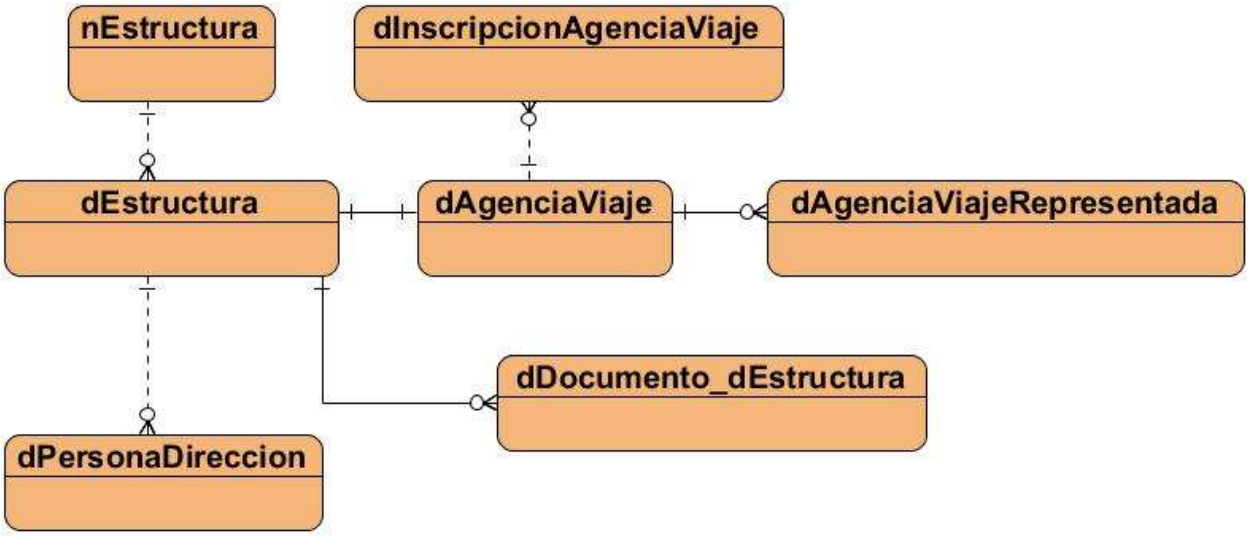


Figura 6. Modelo de Datos del Registro de Agencia de Viajes.

2.3.5 Diagrama de despliegue

Mediante el diagrama de despliegue se muestran las relaciones físicas de los distintos nodos que componen el sistema y la repartición de los componentes sobre dichos nodos. A través de este se captura la configuración de los elementos de procedimiento y las conexiones entre estos en el sistema. El modelo de despliegue consta de uno o más nodos, dispositivos y conectores entre nodos, y entre nodos y dispositivos. El diagrama de despliegue que se propone para el subsistema Agencia de Viajes se muestra en la *Figura 7*.

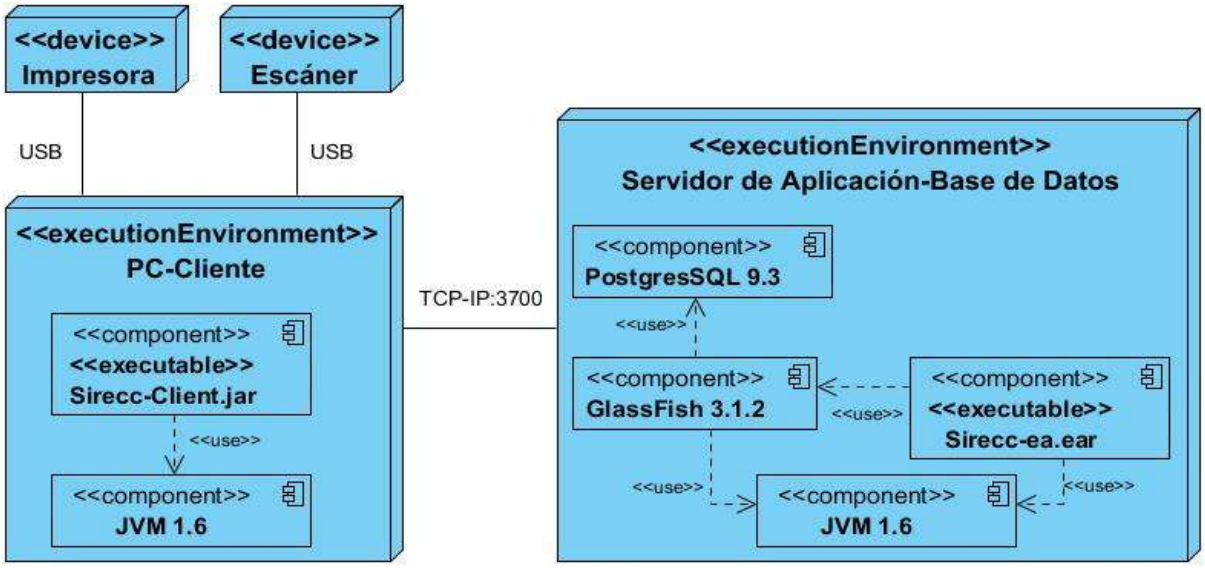


Figura 7. Diagrama de despliegue del subsistema Agencia de Viajes.

El sistema tendrá una distribución física compuesta por una PC-cliente en la cual estará instalada la máquina virtual de Java donde se ejecutará la aplicación en su lado cliente. A la PC-cliente se le conectará por USB (*Universal Serial Bus, Bus Universal en Serie*) una impresora o escáner para imprimir o escanear documentos respectivamente. Se contará con un servidor donde estará el Sistema de Gestor de Base Datos y el Servidor de Aplicaciones, en el cual se encontrará desplegada la aplicación empresarial, que es la encargada de gestionar las respuestas a las peticiones del cliente usando el protocolo TCP-IP (*Transmission Control Protocol/Internet Protocol, Protocolo de Control de Transmisión/Protocolo de Internet*).

2.4 Implementación

De manera general en la primera fase se describieron los artefactos necesarios para dar paso a la fase de Implementación. En el epígrafe se puede evidenciar los estándares de codificación utilizados para la implementación del subsistema así como el modelo de implementación, dejando plasmado el diagrama de componentes, organizado de acuerdo a la arquitectura seleccionada.

2.4.1 Modelo de implementación

En el modelo de implementación se describe fundamentalmente la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. A continuación se

mostrará el **diagrama de componentes,** representando el sistema separado en componentes, además de las dependencias y organización entre ellos.

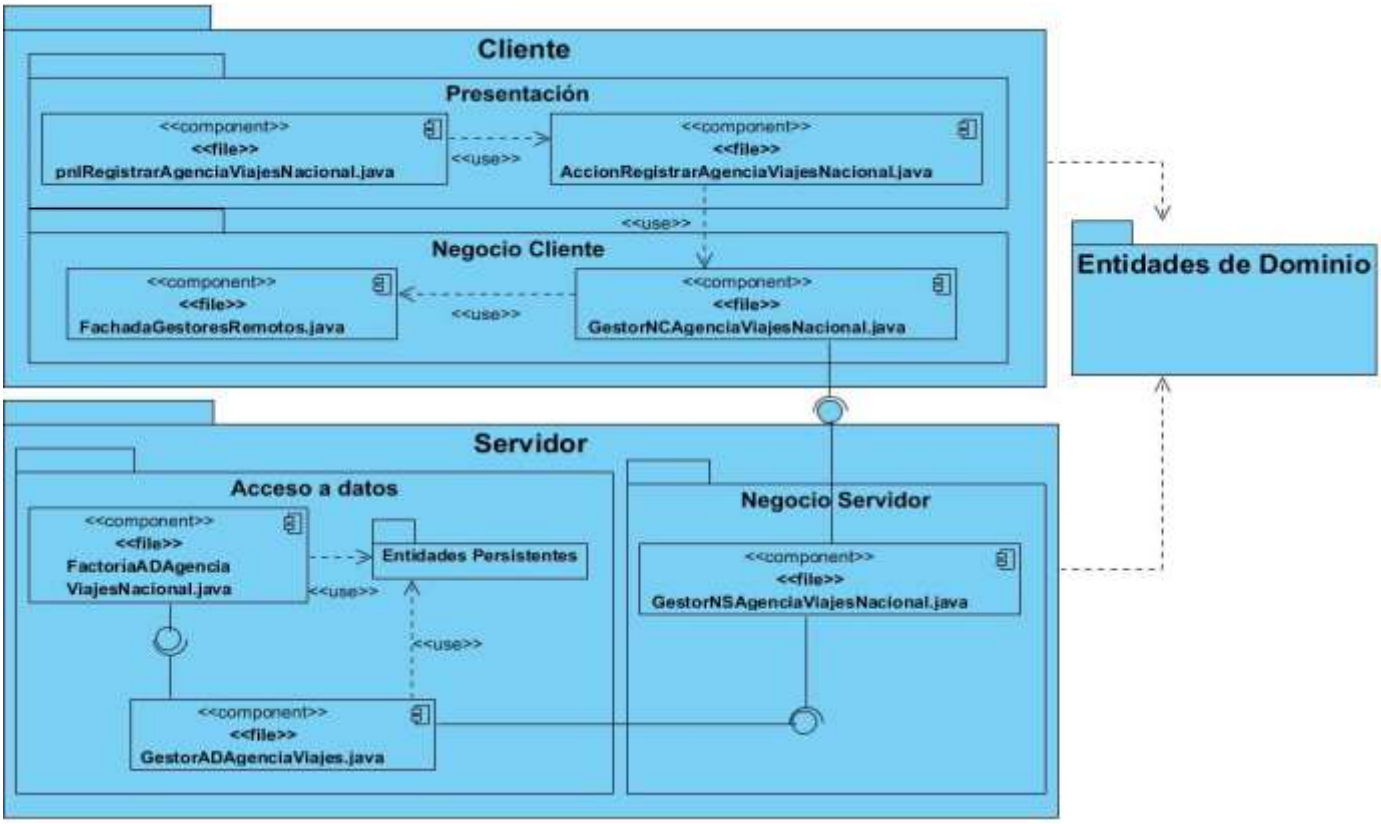


Figura 8. Diagrama de componentes del subsistema Agencia de Viajes.

2.4.2 Estándares de codificación

Los estándares de codificación se realizan con el fin de mejorar la lectura del software, permitiendo entender código nuevo con mayor rapidez y profundidad. De esta forma se logra un enfoque uniforme en la implementación de la solución, a continuación se mostrarán sus usos:

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: "fechaConstitucion" y no solamente "fecha".

- El nombre de todas las variables y métodos comenzarán con letra minúscula y si 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: *GestorADAgenciaViaje* (Gestor de Acceso a Datos de la Agencia de Viajes).

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 manejadas bien por muchas terminales y herramientas.

Métodos: Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.

Variables: Excepto las constantes que todos sus caracteres son en mayúscula, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas.

Conclusiones parciales

Con el desarrollo de este capítulo se puede concluir que:

- Se obtuvieron los resultados esenciales del Análisis y Diseño a partir del proceso Registro de Agencia de Viajes Nacional, dando paso a la fase de Implementación.
- Se describió brevemente la arquitectura propuesta y las capas que la componen lo que permitió entender la estructura del sistema para comenzar la implementación del mismo.
- Se describieron los patrones de diseño, basándonos en el análisis del diagrama de clases.
- Se definieron un conjunto de estándares de codificación, que brindaron la posibilidad de una implementación clara y libre de ambigüedades.

CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

Introducción

Teniendo en cuenta la solución propuesta en el capítulo anterior, se hace necesario comprobar que los resultados se obtuvieron con la calidad requerida. Para esto se evidencia en el desarrollo del presente capítulo el resultado de las pruebas y validaciones realizadas al subsistema, que permiten garantizar el correcto funcionamiento de la totalidad de sus requisitos.

3.1. Pruebas Internas

En este epígrafe se obtienen los resultados de las pruebas internas, realizando validaciones al diseño e implementación del software. Para esto se emplearon métricas orientadas a objetos, métricas de relaciones entre clases y se realizaron pruebas de caja blanca y caja negra.

3.1.1. Validación del diseño de la solución

Para validar el diseño de la solución se realizaron estudios a las métricas de Chidamber y Kemerer (CK) y de Lorenz y Kidd. Fueron seleccionadas luego de un estudio de las métricas propuestas por cada autor, las cuales son explicadas a continuación:

- **Métricas orientadas a clases: la colección de métricas de CK**

"La clase es la unidad fundamental de un sistema orientado a objetos. Por tanto, las medidas y métricas de una clase individual, la jerarquía de clase y las colaboraciones de clases serán invaluables para un ingeniero del software que debe valorar la calidad del diseño" (Pressman, 2006). A continuación se muestra la métrica seleccionada para la validación del diseño, esta fue seleccionada del conjunto de métricas de software propuesto por CK.

Falta de cohesión en métodos (FCM): Representa el número de métodos que acceden a uno o más de los mismos atributos. En caso de que ningún método acceda a los mismos atributos el FCM sería cero. Si la FCM es alta se pueden acoplar los métodos mediante sus atributos. Lo que se desea es conservar alta la cohesión, por tanto mantener baja la FCM (Pressman, 2006).

La métrica se le aplicará a las clases *GestorADAgenciaViajeNacional*, *GestorADAgenciaViajesExtranjera y GestorADAgenciaViajesExtranjeraRepresentada* del paquete

GestorAccesoDatos. A continuación se muestra como ejemplo la clase *GestorADAgenciaViajeNacional,* para ver la realización de la métrica a las otras clases ir al *Anexo 4.*

Primeramente se seleccionan los atributos de la clase y se le asigna una identificación como se muestra en la *Tabla 3*:

Atributos	Identificación
gestorADTrazas	a
factoriaEDFuncionario	b
factoriaEDDocumento	c
gestorADFuncionario	d
gestorADDocumento	e
gestorADDireccion	f
factoriaEDEstructura	g
factoriaEDDatosEstructura	h
factoriaEDAgenciaViajeNacional	i
manager	j

Tabla 3. Atributos identificados de la clase GestorADAgenciaViajeNacional

Después de identificar cada atributo, se le establece a los métodos de la clase los atributos correspondientes:

Métodos	Atributos
insertarAgenciaViajeNacional	a,d,e,f,g,h,i,j
buscarAgenciaViaje	f,g,i,j
modificarAgenciaViaje	a,d,e,f,j
visualizarAgenciaViaje	i,j
obtenerFuncionariosAtivos	j
cancelarAgenciaViajesNacional	a,j
devolverDocumento	j
buscarAgenciaViajesNacional	i,j
existeNombreAgenciaViajes	J

cantidadAgenciasNacionalesExisten	J
obtenerFuncionariosActivos	J
renovarInscripcionAgenciaViajesNacional	a,e,j
ListarAgenciasProximasRenovar	g,i,j
getRenovaciones	J
buscarAVNProximaRenovar	-
buscarAgenciaPorId	b,c,f,g,i,j

Tabla 4. Métodos de la clase GestorADAgenciaViajesNacional y atributos correspondientes

Una vez obtenido los atributos con acceso a cada método, se calcula la cantidad de métodos que acceden a uno o más de los mismos atributos:

Atributos	Identificador	mA
gestorADTrazas	a	4
factoriaEDFuncionario	b	1
factoriaEDDocumento	c	1
gestorADFuncionario	d	2
gestorADDocumento	e	3
gestorADDireccion	f	4
factoriaEDEstructura	g	4
factoriaEDDatosEstructura	h	1
factoriaEDAgenciaViajeNacional	i	6
manager	j	14
∑(mA)		40

Tabla 5. Total de métodos que acceden a un atributo de la clase GestorADAgenciaViajesNacional.

Finalmente, luego de obtener los datos se calcula la FCM mediante la siguiente fórmula:

$$\text{FCM} = 1 - \sum (mA)/(m \times a)$$

Donde:

m: Número de métodos en la clase. (m=15)

a: Número de atributos en la clase. (a=10)

mA: Número de métodos que acceden a un atributo. (∑ (mA)=41)

FCM =1- 40 / (15*10) = 1- 40 / 150 = 1- 0.27 = **0.73**

Para una FCM <=1 se alcanzaría un valor bajo, lo que indica que en los resultados obtenidos existe una alta cohesión, por lo cual se puede afirmar que las clases *GestorADAgenciaViajesNacional* con un valor de FCM=0.73, *GestorADAgenciaViajesExtranjera* con la FCM= 0.70 y *GestorADAgenciaViajesExtranjeraRepresentada* con FCM= 0.74 del paquete *GestorAccesoDatos* están bien diseñadas.

Árbol de Profundidad de la Herencia (APH): La profundidad del árbol de herencia en una estructura de clases, se obtiene mediante la cuenta desde el nodo raíz de la estructura, hasta el último nodo de la hoja. Este tipo de cuenta se realiza por niveles, es decir, cuantos niveles jerárquicos hay desde el nodo raíz, hasta el nodo hoja. De forma positiva se puede plantear que los valores grandes del APH indican que se podría reutilizar muchos métodos. Sin embargo, a medida que crece el APH es probable que las clases del nivel inferior hereden muchos métodos, lo cual se presta a posibles dificultades cuando se trata de predecir el comportamiento de una clase. Se propone como indicador de un abuso de herencia el umbral de 6 niveles. En el lenguaje Java, empleado para el desarrollo de la investigación, las clases siempre heredan de la clase Object, por tanto se le añade uno al APH (Chidamber, 1994), (Pressman, 2006).

En la *Figura 9,* se exponen los niveles que conforman el árbol, obteniendo un total de 5 niveles, incluyendo la clase Object. Se muestra que la clase es menos propensa a errores y que no es compleja en cuanto a diseño y reutilización, ya que no supera los 6 niveles de abuso propuesto en la bibliografía referenciada. Por tanto para este árbol de obtiene un buen indicador de herencia.

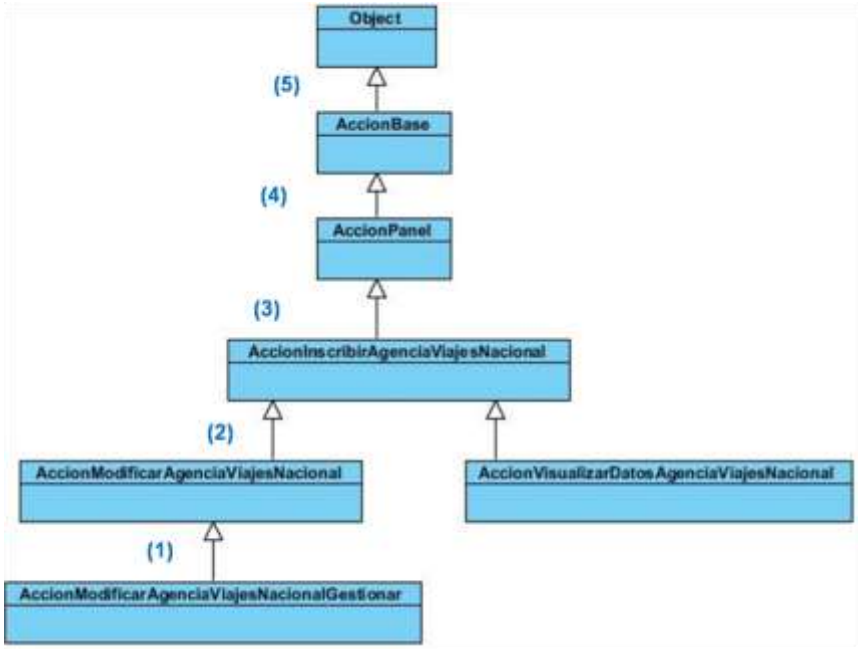


Figura 9. Árbol de profundidad de herencia.

- **Métricas orientadas a objetos propuestos por Lorenz y Kidd**

Lorenz y Kidd en su libro sobre métricas orientadas a objetos, divide las métricas basadas en clases en cuatro categorías, cada una con un diseño al nivel de componentes, como son tamaño, herencia, valores internos y valores externos (Pressman, 2006). A continuación se presenta la métrica a utilizar de las propuestas por Lorenz y Kidd.

Tamaño Operacional de Clases (TOC): Una de las métricas empleadas es el TOC, aquí se expresa el número de métodos asignados a una clase. Se determina el total de operaciones (TO) que están encapsulados a una clase, obteniendo un esquema sencillo de implementación que brinda este estudio y que simultáneamente cubre los principales atributos de calidad de software. Los atributos son:

Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.

Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Los grandes valores del TOC indican que las clases tengan una gran responsabilidad, esto implica que se reduzca la clase y se complique la implementación y prueba. De manera general, se debe dar más peso a las operaciones y atributos heredados para establecer el tamaño de la clase. Se debe realizar el cálculo de los promedios para el número de atributos y operaciones de clase. Mientras más pequeños sean los valores promedios del TOC habrá más posibilidad de que las clases puedan ser reutilizadas (Pressman, 2006).

En la presente investigación se tomó como muestra las clases representadas en el diagrama de clases del proceso Inscribir Agencia de Viajes Nacional, bajo los criterios de alto (A), medio (M) y bajo (B), siendo P el promedio de operaciones que poseen las clases de la muestra y C la cantidad de operaciones de una clase. Para un mejor entendimiento de las medidas del cálculo del dominio del TOC se puede observar la tabla representada en el *Anexo 5*.

La métrica se empleó a 8 clases, con un total de 160 operaciones, obteniendo un promedio de 20 operaciones. A continuación se mostrará el cálculo del umbral para la Responsabilidad, Complejidad de implementación y Reutilización:

Responsabilidad y Complejidad de Implementación

(TO) Baja C <= 20 Media 20< C <= 40 Alta C >40

Atributo	Clases	TO	Tamaño(TO)
Responsabilidad y Complejidad de Implementación	FactoriaEDAgenciaViajes	15	B
	AccionRegistrarAgenciaViajeNacional	39	M
	GestorNCAgenciaViajeNacional	13	B
	FachadaGestoresRemotos	25	M
	GestorNSAgenciaViajeNacional	13	B
	EDAgenciaViajeNacional	23	M
	GestorADAgenciaViajesNacional	16	B
	DInscripcionAgenciaViaje	16	B

Tabla 6. Cálculo del TOC para la Responsabilidad y Complejidad de Implementación.

Reutilización

(TO) Baja C >40 Media 20< C <= 40 Alta C <= 20

Atributo	Clases	TO	Tamaño(TO)
Reutilización	FactoriaEDAgenciaViajes	15	A
	AccionRegistrarAgenciaViajeNacional	39	M
	GestorNCAgenciaViajeNacional	13	A
	FachadaGestoresRemotos	25	M
	GestorNSAgenciaViajeNacional	13	A
	EDAgenciaViajeNacional	23	M
	GestorADAgenciaViajesNacional	16	A
	DInscripcionAgenciaViaje	16	A

Tabla 7. Cálculo del TOC para el atributo Reutilización.

A continuación se muestra los resultados de la aplicación a través de esta métrica al diseño:

Responsabilidad y Complejidad de Implementación	Cantidad de clases	Promedio
Baja	5	62,5
Media	3	37,5
Alta	0	0

Tabla 8. Resultado de la Responsabilidad y Complejidad de Implementación.

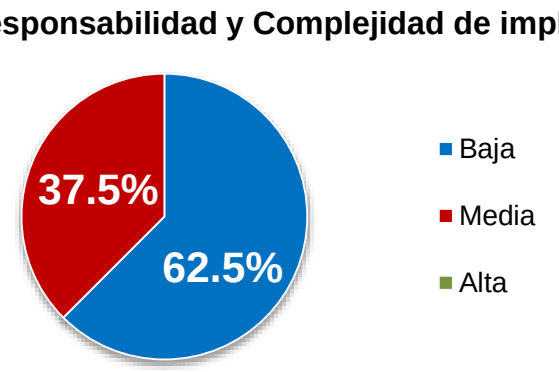


Figura 10. Representación de la evaluación de la métrica TOC para los atributos Responsabilidad y Complejidad de Implementación.

Reutilización	Cantidad de clases	Promedio
Alta	5	62,5
Media	3	37,5
Baja	0	0

Tabla 9. Resultado de la Reutilización.

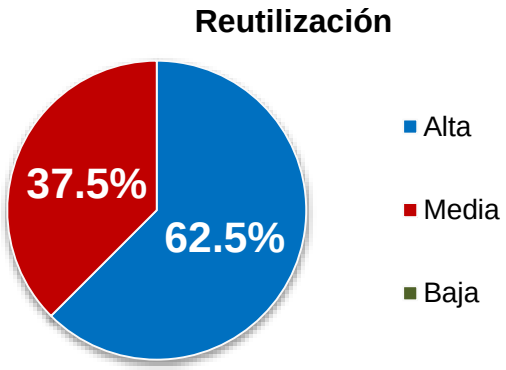


Figura 11. Representación de la evaluación de la métrica TOC para el atributo Reutilización.

Se puede concluir que el diseño del subsistema Agencia Viajes Nacional alcanzó resultados positivos durante la evaluación de la métrica. Se obtuvo una baja responsabilidad de las clases (62.5%) y complejidad de implementación (62.5%), mientras que la reutilización es alta (62.5%). Estos datos favorecen el buen uso del diseño empleado y el resultado de la implementación.

3.1.2. Validación de la implementación

La implementación se valida a través de las pruebas de caja blanca que se basa en el estudio del código a evaluar y las pruebas de caja negra que se enfoca en los requisitos funcionales del software.

- **Pruebas de caja blanca**

Las pruebas de caja blanca se utilizan con el fin de buscar errores a través del código interno. Para la validación del subsistema se realizó la prueba del camino básico a las funcionalidades de mayor complejidad, permitiendo identificar los diferentes caminos independientes de la función y probar su funcionamiento al menos una vez. A continuación se muestra el proceso realizado al método *buscarAVNProximaRenovar* de la clase *GestorADAgenciaViajesNacional*.

```
@Override
public List<EDAgenciaViajeNacional> buscarAVNProximaRenovar(int annioRenovacion, int idPais) throws Exception {

    List<EDAgenciaViajeNacional> resultados = new ArrayList<EDAgenciaViajeNacional>();        1
    if (idPais == 0 || idPais == 39) {        2
        List<EDAgenciaViajeNacional> analizar = buscarAgenciaViaje("", 0, 0, 0, 2);        3
        if (annioRenovacion != 0) {        4
            for (EDAgenciaViajeNacional agencia : analizar) {        5
                if (agencia.getFechaVencimiento() != null) {        6
                    if (agencia.getFechaVencimiento().get(Calendar.YEAR) == annioRenovacion) {        7
                        resultados.add(agencia);        8
                    }
                }
            }
        } else {        9
            return resultados=analizar;        10
        }
    }
    return resultados;        11
}
```

Figura 12. Método buscarAVNProximaRenovar de la clase GestorADAgenciaViajesNacional.

A partir de este método se crea el flujo de control lógico a través de nodos, aristas y regiones. El grafo de flujo es un grafo dirigido, definido por un par de conjuntos $G = (V, A)$, donde $V \neq \emptyset$ es un conjunto no vacío de objetos simples llamados vértices o nodos; $E \subseteq \{(\mathfrak{a},\mathfrak{b}) \in V \times V : \mathfrak{a} \neq \mathfrak{b}\}$ es un conjunto de pares ordenados de elementos de V denominados aristas o arcos, donde por definición un arco va del primer nodo ($\mathfrak{a}$) al segundo nodo ($\mathfrak{b}$) dentro del par (Gutin, 2006). A continuación se muestra el grafo:

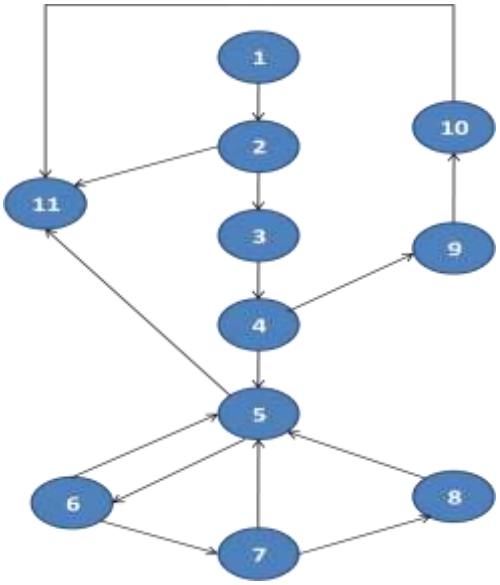


Figura 13. Grafo del flujo del código del método buscarAVNProximaRenovar.

Para obtener los caminos independientes se calcula la complejidad ciclomática, la cual se puede obtener de tres formas (Pressman, 2006):

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

  Cantidad de regiones=6

- La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A- N+2 siendo A el número de aristas del grafo de flujo y N el número de nodos del mismo.

  V (G) =15 aristas – 11 nodos + 2 = 6

- La complejidad ciclomática, V (G), de un grafo de flujo G se define como V(G)=P+1 donde P es el número de nodos predicado contenidos en el grafo de flujo G.

  V (G) = 5 nodos predicados + 1 = 6

Como complejidad ciclomática se obtiene 6 para cualquiera de las tres formas de calcularlo, por tanto la cantidad de caminos básicos que puede tomar el algoritmo en su ejecución es 6, como se muestran en la *Tabla 10*.

No. Camino	Camino básico
1	1 – 2 – 11
2	1 – 2 – 3 – 4 – 9 – 10 – 11
3	1 – 2 – 3 – 4 – 5 – 11
4	1 – 2 – 3 – 4 – 5 – 6 – 5 - 11
5	1 – 2 – 3 – 4 – 5 – 6 – 7 – 5 – 11
6	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 5 – 11

Tabla 10. Camino básico.

Luego de determinar los caminos básicos se ejecutaron para cada uno los Casos de Pruebas (CP) correspondientes. Para definir los CP se analizaron las descripciones, condiciones de ejecución, entradas, resultados esperados, resultados obtenidos y salida. Para consultar los CP ir al *Anexo 6.*

- **Pruebas de caja negra**

Las pruebas de caja negra se centran en las interfaces del software donde se especializan en la detección de errores mediante casos de prueba. Tienen como objetivo probar que el software se encuentre acorde a los requisitos obtenidos. Para una mejor realización de la prueba se llevó a cabo la técnica de partición equivalente, con la intención de encontrar de manera más rápida errores que normalmente serían detectados después de la ejecución de muchos casos de prueba o de una cantidad relativamente mayor. Los casos de prueba se realizaron apoyándonos del estudio de las pruebas basadas en requisitos, donde se obtuvo un total de 39 casos de prueba, demostrándose cada una de las funcionalidades. Los casos de pruebas se pueden examinar en el Expediente de Proyecto. Para obtener los resultados de las pruebas verificando si los requisitos se encuentran parcial o completamente satisfactorios consultar el epígrafe *3.2.1 Variable independiente (Calidad)* del presente capítulo.

3.2. Valoración de las variables

En este epígrafe se verificarán las variables dependientes y la variable independiente basándonos en las unidades de medidas y dimensiones de cada una de ellas.

3.2.1. Variable independiente

Para realizar la operacionalización a la variable independiente *Sistema informático para gestionar los datos en el Registro de Agencias de Viajes* se analizarán dos dimensiones, las cuales serán explicadas a continuación:

Alcance:

Para obtener el alcance del software se realizó un análisis al cumplimiento de los requisitos trazados, calculando el índice de requisitos implementados. Las variables propuestas fueron:

Índice de requisitos implementados (IRI)

Cantidad de requisitos implementados (CRI)

Cantidad total de requisitos (CTR)

Donde se adquieren como valores CRI=41, CTR=41, para obtener el IRI se realiza el siguiente cálculo:

IRI = CRI / CTR = 41 / 41 = 1

Una vez obtenido los resultados, se puede asegurar que la solución satisface el 100 % de las funcionalidades del sistema definidas por el cliente, cumpliendo con un índice de valor 1. Por tanto el sistema tiene un alcance total sobre los requisitos analizados.

Calidad: La calidad del software *"es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente"* (Pressman, 2006).

Para adquirir la calidad del software de manera más eficiente se obtuvieron los resultados de la calidad interna y la calidad externa, que brindan los resultados de las **Pruebas de Liberación**. La calidad interna contiene las revisiones realizadas por el Grupo de Calidad perteneciente al centro CEGEL de la facultad 3, mientras que la externa es el Centro Nacional de Calidad de Software (Calisoft). Ambos grupos se basan en realizar las pruebas en tres iteraciones, con el fin de verificar que el sistema cumple con la calidad requerida para ser entregado al cliente.

Estos grupos ofrecen el servicio de Evaluación de Productos que está enfocado en la ejecución de pruebas dinámicas para asegurar que el software cumple con la calidad para ser utilizado. Este servicio comprende la realización de varios tipos de pruebas (funcionalidad, usabilidad, comparativa de código, seguridad, entre otras), con el objetivo de verificar el cumplimiento de los requisitos bajo los cuales fue desarrollado el producto, al mismo tiempo que se evalúa hasta qué punto el sistema cumple con lo establecido con normas y estándares internacionales de calidad de software (Calisoft, 2014).

Para evaluar la calidad del sistema se realizó un análisis de los resultados obtenidos a través de la calidad interna y calidad externa, los cuales se mostrarán a continuación:

- **Calidad interna**

Grupo de Calidad: A través de este grupo se le aplicaron pruebas de funcionamiento al sistema en un marco de tres iteraciones. Finalmente se obtuvo un total de 106 no conformidades. En la primera iteración se detectaron 56, de estas 7 no proceden. En la segunda iteración se encontraron 35, de las cuales no proceden 2. Finalmente en la tercera iteración se identificaron 15 no conformidades, donde no proceden 2 de ellos. Estos errores fueron resueltos en su totalidad, para visualizar gráficamente estos resultados ver la *Figura 14*.

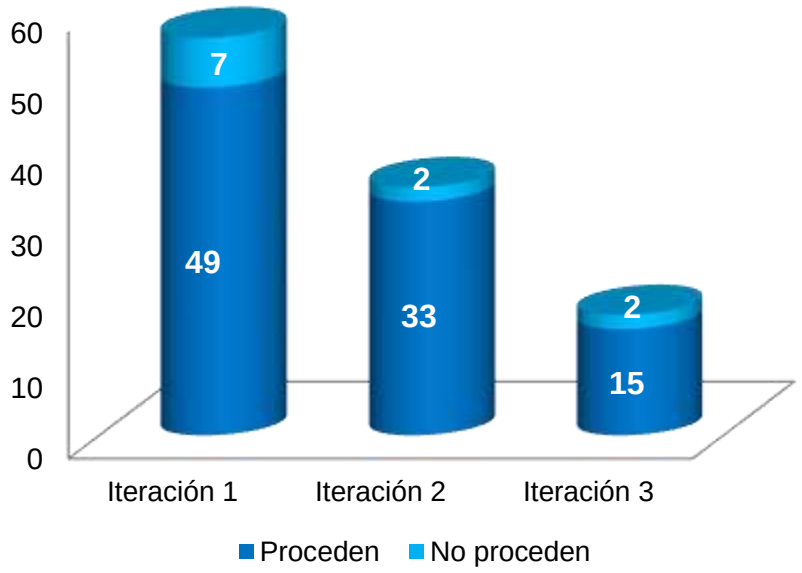


Figura 14. Resultados de no conformidades detectadas por el Grupo de Calidad.

- **Calidad externa**

Calisoft: Finalizada la revisión por parte del Grupo de Calidad, se retoman las pruebas por el grupo de coordinadores de Calisoft que llevan a cabo el proceso de evaluación, obteniendo un total de 15 no conformidades en tres iteraciones y una prueba exploratoria inicial. En la prueba explotaría inicial se obtuvieron 2 no conformidades de tipo funcionalidad. En la primera iteración se obtuvo un total de 4 no conformidades, de ellas 3 de funcionalidad y 1 de correspondencia con otro artefacto. En la segunda iteración se identificaron 4 no conformidades, todas de funcionalidad. En la tercera iteración se encontraron 5 no conformidades, 2 de funcionalidad, 1 de error de interfaz, 1 de ortografía y 1 de validación. De estas no conformidades se resolvieron un total de 100 %, en la *Figura 15* se obtienen los resultados de manera más explícita.

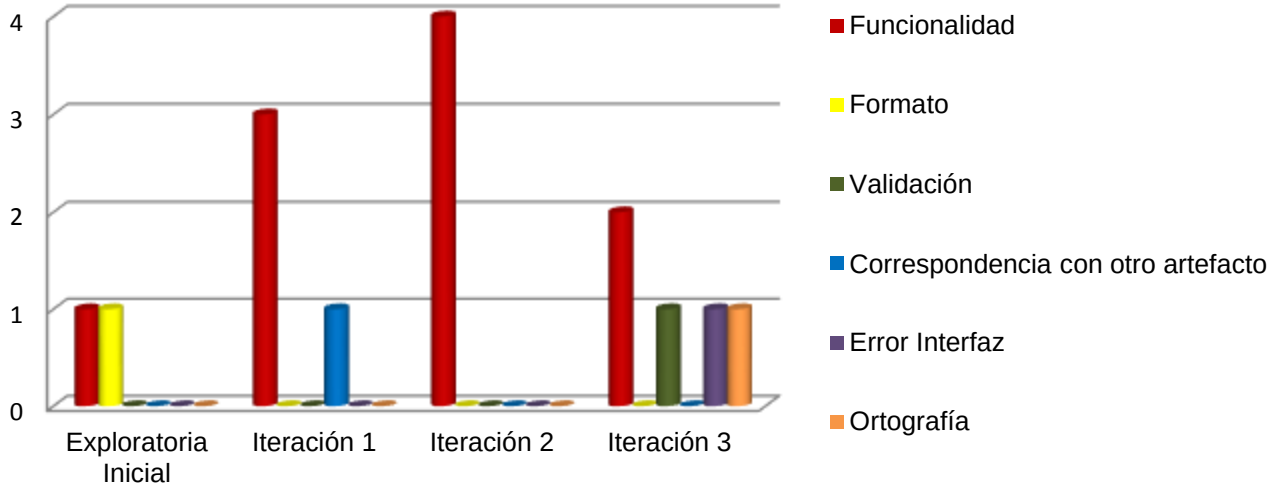


Figura 15. Resultados de no conformidades detectadas por Calisoft.

Una vez finalizadas las pruebas, se emitió un acta de liberación por ambos grupos, aprobando que el sistema cumpla con el mínimo de calidad para su utilización por el cliente.

3.2.2. Variables dependientes

A continuación se muestra los resultados obtenidos una vez verificada las variables dependientes *Disminuir el tiempo de respuesta en el proceso* y *Aumentar el nivel de seguridad de la información*.

Disminuir el tiempo de respuesta en el proceso: Esta variable se declara con el objetivo de agilizar el proceso de respuesta del Registro de Agencia de Viajes. Para medir esta variable se obtuvo la dimensión Tiempo y como indicador Tiempo de respuesta en el proceso. Para obtener una correcta medición se estableció como unidad de medida una comparación entre el tiempo de gestionar la Agencia de Viajes actualmente con el tiempo que duraría la gestión con el sistema:

TIAVS<TIAVA

TMAVS<TMAVA

TCAVS<TCAVA

Esta comparación se realiza con el fin de verificar si a través del sistema se asegura un trabajo más ágil, para esto se realizó una encuesta (Ver *Anexo 7*) a la especialista Dora Águila de la Cámara de Comercio de la República de Cuba con el objetivo de identificar los pasos y tiempo en que se lleva a cabo el proceso. De los pasos presentados se va a enfocar los resultados en el último de cada proceso. A través de la *Tabla 11* se muestra la diferencia entre el estado actual y estado en que se encontrará una vez utilizado el sistema.

Proceso	Pasos a seguir en la Cámara de Comercio	Tiempo de demora en la Cámara de Comercio	Pasos a seguir en el sistema	Tiempo de demora en el sistema
Inscripción de la Agencia de Viajes.	Conformar el expediente.	2 horas	Seleccionar el submenú Inscribir Agencia de Viajes, llenar los datos generales, adicionar los documentos asociados. Pulsar el botón Siguiente, se visualiza la Licencia de inscripción y luego pulsar el botón Terminar.	3 minutos
	Registrar en el libro (Tomo y Folio).	2 horas		
Actualización de la Agencia de Viajes.	Después de aprobada, elaborar la licencia y anotar en el libro (Tomo y Folio).	2 horas	Seleccionar el submenú Gestionar Agencia de Viajes. Buscar la agencia, seleccionar el botón Modificar Agencia de Viajes. Modificar los campos deseados, pulsar el botón Terminar.	2 minutos

Cancelación de la Agencia de Viajes.	Cancelar en el libro.	2 horas	Seleccionar el submenú Gestionar Agencia de Viajes. Buscar la agencia, seleccionar el botón Cancelar Agencia de Viajes. Llenar los campos y pulsar el botón Aceptar.	1 minuto

Tabla 11. Comparación de los procesos en la actualidad y en el sistema.

El tiempo de demora en el sistema se obtiene teniendo en cuenta el tiempo desde que se llenan los campos hasta que se da la respuesta por el sistema, ya que se desea comparar todos los pasos realizados. Sin embargo el tiempo de respuesta demora solamente 1 segundo, cumpliendo con lo establecido en el RnF_7 (ver *Anexo 2*), el cual plantea que el tiempo de respuesta brindado por el sistema será menor de 3 segundos. Por lo antes planteado se demuestra que el proceso es más eficiente, ya que en solo minutos se realizan las peticiones deseadas y en un segundo se obtiene la respuesta por el sistema. Validando de esta forma que la variable dependiente satisface la solución propuesta.

Aumentar el nivel de seguridad de la información: A través de esta variable se pretende asegurar que mediante el sistema la información referente a las Agencias de Viajes cuentan con mayor seguridad que actualmente en la Cámara de Comercio de la República de Cuba. Para esto se obtuvo la dimensión Nivel de Seguridad y como indicadores la Confidencialidad, Integridad y Disponibilidad, las cuales son representadas a través de la unidad de medida. En la unidad de medida de esta variable se muestran la cantidad de requisitos no funcionales que demuestran la seguridad para cada uno de los indicadores, estos son:

RnF_25: El sistema podrá ser utilizado solamente por usuarios autenticados en el mismo, de esta forma se asegura la confidencialidad del sistema.

RnF_26: El sistema brindará la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado, asegurando la integridad.

RnF_27: El sistema mostrará las funcionalidades de acuerdo a quien esté autenticado en el mismo, a través de esta no conformidad se demuestra que existe la disponibilidad en el sistema.

Para un mejor entendimiento teórico de los indicadores, ver epígrafe *1.3 Seguridad de la información*. Una vez obtenido los requisitos no funcionales para cada indicador, se puede señalar que cada uno cuenta con una CRNF=1, lo que demuestra que el sistema brinda confidencialidad, integridad y disponibilidad al cliente. Actualmente el personal de la Cámara de Comercio tiene acceso a toda la información, la cual es guardada en la herramienta Microsoft Excel brindando la posibilidad de que los datos sean modificados fácilmente. Mediante la utilización del sistema se mitigan estos problemas, asignándole a cada personal un usuario con roles asociados, que le permite acceder solo a la información a la cual tiene permiso, además de contener una contraseña encriptada que es guardada en la base de datos a través del algoritmo de reducción criptográfico MD5, también se tiene un registro de trazas para obtener información de las acciones que ha realizado cada usuario en el sistema. De esta forma se valida que la variable dependiente satisface la solución propuesta.

3.3. Despliegue

En la fase de Despliegue se efectúa la prueba de aceptación que fue realizada a través de los desarrolladores y especialistas de la Cámara de Comercio. Se validó que el sistema cumple con el funcionamiento esperado comparándose el producto final con las necesidades del cliente. Una vez satisfechos al 100% se firmó el acta de aceptación, la cual fue entregada al jefe de proyecto y no puede ser anexada por restricciones de confidencialidad.

Conclusiones parciales

Luego de la realización de pruebas y validaciones al subsistema se puede demostrar que:

- Se realizó la validación a las variables dependientes y variable independiente donde se mostraron resultados satisfactorios obtenidos.
- Se validó el diseño a través de métricas estudiadas, obteniendo resultados satisfactorios demostrando que los atributos de calidad alcanzaban niveles favorables.
- El código implementado se probó a partir de la utilización de pruebas de caja blanca, empleando la técnica del camino básico.
- Para detectar errores en el subsistema en cuanto a validaciones, que no fueron tomadas en cuenta durante la implementación por los desarrolladores, se aplicaron las pruebas de caja

negra con la ayuda de los diseños de casos de prueba, a los errores detectados se les dieron solución obteniendo resultados satisfactorios.

CONCLUSIONES

Con la realización del presente trabajo de diploma se desarrolló el módulo Agencia de Viajes propuesto para el sistema SIRECC, contribuyendo a la disminución del tiempo de respuesta de los procesos y al aumento de la seguridad de la información. Es por esto que se puede concluir que:

- El análisis del marco teórico de la investigación permitió obtener un dominio sobre el objeto de estudio a investigar.
- El diseño de la solución permitió la obtención de un modelo en el que se visualiza la manera en que debe ser implementado el subsistema.
- A partir de esta implementación se obtuvo una aplicación capaz de satisfacer las necesidades del cliente.
- El desarrollo de las métricas y pruebas de calidad aplicadas, tanto al sistema como a los artefactos generados, arrojaron resultados satisfactorios para un posterior uso del subsistema.

RECOMENDACIONES

Luego de haber dado cumplimiento a los objetivos de este trabajo de diploma y teniendo en cuenta las experiencias adquiridas durante el desarrollo del mismo, se recomienda:

- Continuar con el mejoramiento del subsistema a partir de nuevos cambios que puedan surgir de acuerdo a nuevas necesidades del cliente.
- Incluir nuevas funcionalidades al subsistema en consideración de lo que desee el cliente.

BIBLIOGRAFÍA

Alejandro Santanach, Juan E Vargas, Lizardo Ramírez, Diana R Prieto, Mayrín Ramos. 2012. Propuesta de un método de estimación de tiempo y esfuerzo para las pruebas de liberación, aceptación y piloto. 2012. 22.

Calisoft. 2014. Centro Nacional de Calidad del Software . Evaluación de los Productos. [En línea] 2014. [Citado el: 20 de mayo de 2014.] https://calisoft.uci.cu/index.php?option=com_k2&view=item&id=48:evaluaci%C3%B3n-de-productos.

Cámara de Comercio de la República de Cuba. 2013. Portal de la Cámara de Comercio de la República de Cuba. [En línea] 2013. [Citado el: 13 de octubre de 2013.] http://www.camaracuba.cu/index.php?option=com_content&view=article&id=44&Itemid=54.

Cámara de Comercio de Madrid. 2012. Cámara de Madrid. [En línea] 18 de octubre de 2012. [Citado el: 15 de enero de 2014.] http://www.camaramadrid.es/.

Campos, Dayana Sanz. 2011. Diseño e implementación del módulo ¨Inscripción¨ del Sistema para la Gestión de Antecedentes Penales (SIGESAP). Ciudad de la Habana : s.n., 2011.

CEGEL. 2014. Centro de Gobierno Electrónico. [En línea] 2014. [Citado el: 07 de marzo de 2014.] http://gespro.cegel.prod.uci.cu/.

Chidamber, Kemerer. 1994. A metric suite forsz object oriented design. 1994. 467-493.

Gaibor, Carmen. 2008. Arquitectura Multicapa. [En línea] 02 de febrero de 2008. [Citado el: 23 de enero de 2014.] http://es.scribd.com/doc/109269672/ARQUITECTURA-MULTICAPA.

García, Carlos Alejandro Suárez. 2011. Diseño e Implementación de un módulo de integración para la Solución Tecnológica Integral para la Modernización de la División de Antecedentes Penales. Ciudad de la Habana : s.n., 2011.

Gesvi. 2014. Herramienta para la gestión integral de la agencia de viajes. [En línea] 2014. [Citado el: 12 de marzo de 2014.] http://portal.gesvi.net/Gesvi/page/gesvi/index.

Bibliografía

GlassFish. 2013. World's first Java EE 7 Application Server. [En línea] 2013. [Citado el: 2013 de diciembre de 18.] https://glassfish.java.net/.

Granado, Yasmany García. 2012. http://repositorio_institucional.uci.cu/. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05402_12. [En línea] 2012. [Citado el: 08 de mayo de 2014.] http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05402_12.

Gutin, Jorgen Bang-Jensen y Gregory. 2006. Digraphs: Theory, Algorithms and Applications. s.l. : Cuarta Edición, 2006.

Ivar Jacobson, Grady Booch y James Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000. 464.

Kendall, Kenneth E. Kendall y Julie E. 2005. Análisis y Diseño del Sistema. México : Sexta Edition, 2005.

Laboratorio de Gestión de Proyectos. 2013. Laboratorio de Gestión de Proyectos. UCI. [En línea] 2013. [Citado el: 23 de enero de 2014.] http://gespro.cegel.prod.uci.cu/.

Larman, Craig. 1999. UML y Patrones. México : Primera Edición, 1999.

León, Miguel. 2011. Cámara de Comercio de Lima. Camarito Informa. [En línea] 18 de julio de 2011. [Citado el: 10 de diciembre de 2013.] http://www.camaralima.org.pe/bismarck/iso/Informatiso/Informatiso_julio2011.pdf.

Ley 1091. 2013. Portal de la Cámara de Comercio de la República de Cuba. [En línea] 2013. [Citado el: 15 de octubre de 2013.] http://www.camaracuba.cu/index.php?option=com_content&view=article&id=54&Itemid=64.

Maribel. 2012. Estudioteca. Telecomunicaciones. [En línea] 21 de marzo de 2012. [Citado el: 21 de enero de 2014.] http://www.estudioteca.net/universidad/telecomunicaciones/gestor-base-datos/.

Martín, Antonio J. 2011. Persistencia. [En línea] 2011. [Citado el: 17 de enero de 2014.] http://www.youblisher.com/p/153846-Persistencia-JPA/.

OfiViaje. 2014. Software Agencia Viajes Minorita. [En línea] 2014. [Citado el: 12 de marzo de 2014.] http://www.ofi.es/software/agencias-de-viaje.

Bibliografía

Oracle Corporation. 2013. NetBeans IDE. The Smarter and Faster Way to Code. [En línea] 2013. [Citado el: 11 de diciembre de 2013.] https://netbeans.org/.

Oracle. 2010. Oracle Corporation. JPQL Language Reference. [En línea] 2010. [Citado el: 20 de marzo de 2014.] http://docs.oracle.com/cd/E17904_01/apirefs.1111/e13946/ejb3_langref.html.

Paredes, Henry Terrero y Jose. 2010. Desarrollo de aplicaciones con Java. s.l. : Fundación código libre Dominicano, 2010.

Peláez, Juan. 2009. Arquitectura Basada en Componentes. [En línea] 15 de abril de 2009. [Citado el: 24 de enero de 2014.] http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-componentes/.

Pfleeger, Charles P. 2006. Segurity in computing. 2006.

PostgreSQL. 1996. The PostgreSQL Global Development Group. The world´s most advanced open source database. [En línea] 1996. [Citado el: 15 de diciembre de 2013.] http://www.postgresql.org/.

Pressman, Roger S. 2006. Ingeniería del Software. s.l. : Sexta Edición, 2006.

Programa de Mejora. 2008-2010. Ciclo de vida de proyectos pilotos del programa de mejora. La Habana : s.n., 2008-2010.

Rubinger, Bill Burke y Andrew Lee. 2010. Enterprise Java Beans 3.1. s.l. : Edition 6th, 2010.

Sequeira, Msc. Tania. 2010. Arquitectura Cliente/Servidor. [En línea] 16 de noviembre de 2010. [Citado el: 23 de enero de 2014.] http://www.slideshare.net/NoeGonzalezMendoza/arquitectura-cliente-servidor.

Travelio. 2012. Software de gestión para minoristas y agencias de viaje online. [En línea] 2012. [Citado el: 12 de marzo de 2014.] http://www.hiberus.com/travelio-software-de-gestion-para-minoristas-y-agencias-de-viaje-online.

UCI. 2014. Universidad de las Ciencias Informáticas. [En línea] 2014. [Citado el: 07 de marzo de 2014.] http://www.uci.cu/.

Valdés, Sandra González. 2011. Diseño e Implementación de los módulos de Preparación de Documentos, Digitalización de Documentos y Asociación de Metadatos del Centro de Digitalización para la División de Antecedentes Penales. Ciudad de la Habana : s.n., 2011.

Bibliografía

Visual Paradigm. 2013. Visual Paradigm. [En línea] 16 de diciembre de 2013. [Citado el: 29 de febrero de 2014.] http://www.visual-paradigm.com/.

ANEXOS

Anexo1. Documento Especificación de Requisitos de Software de Agencia de Viajes.

Consultar el documento de nombre —0113_Especificación de Requisitos de Software Agencias de Viaje, en el Expediente de proyecto del proceso de Registro de Agencia de Viaje.

Anexo2. Documento Especificación de Requisitos no funcionales de Software de Agencia de Viajes.

Consultar el documento de nombre —0113_Especificación de Requisitos No funcionales de Software en el Expediente de proyecto del proceso de Registro de Agencia de Viaje.

Anexo3. Modelo de Datos del módulo Agencia de Viajes.

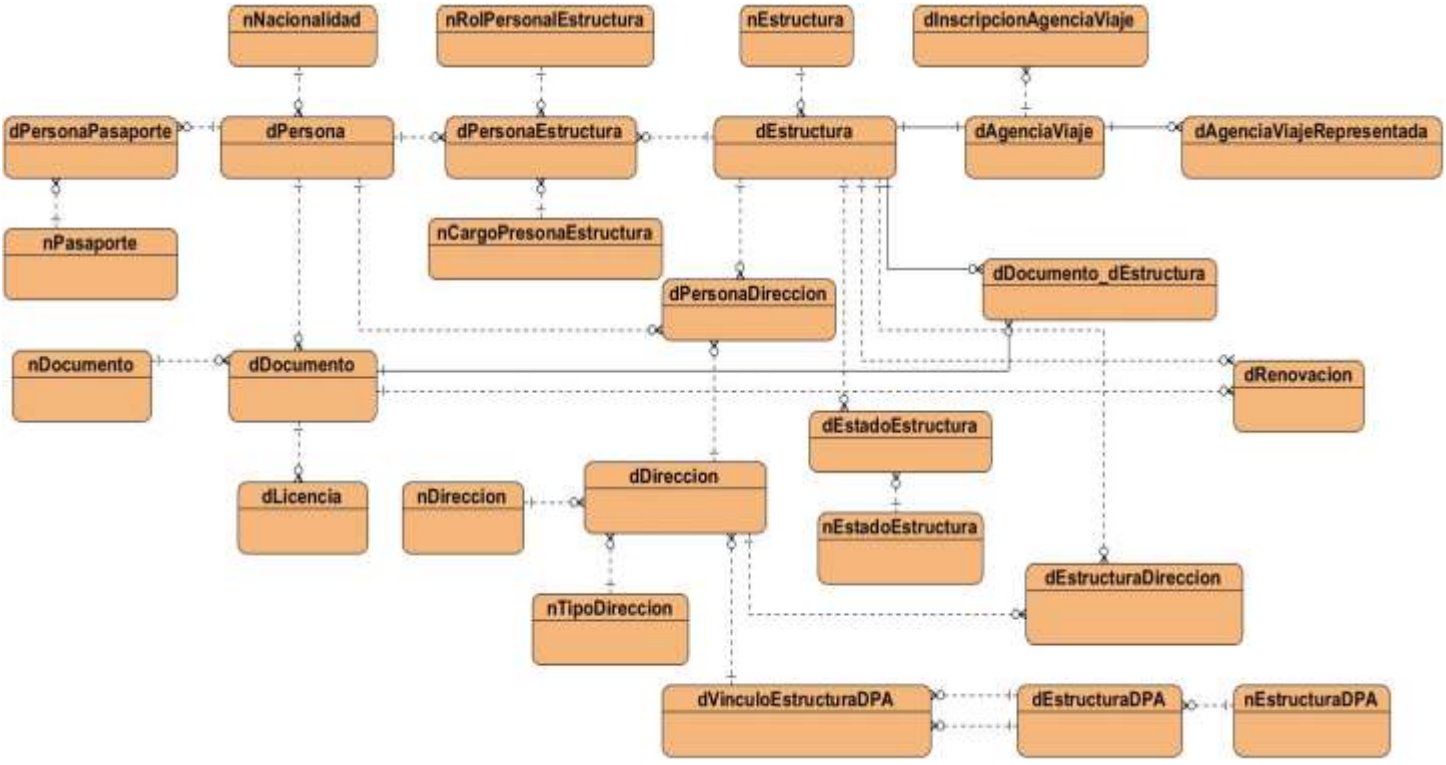


Figura 16. Modelo de datos del módulo Agencia de Viajes.

Anexo 4. Documento métrica Falta de Cohesión de Métodos.

Consultar el documento de nombre — Falta de Cohesión de Métodos, en la carpeta Documentos del proceso de Registro de Agencia de Viaje.

Anexo 5. Rango de valores para medir el cálculo del dominio del TOC.

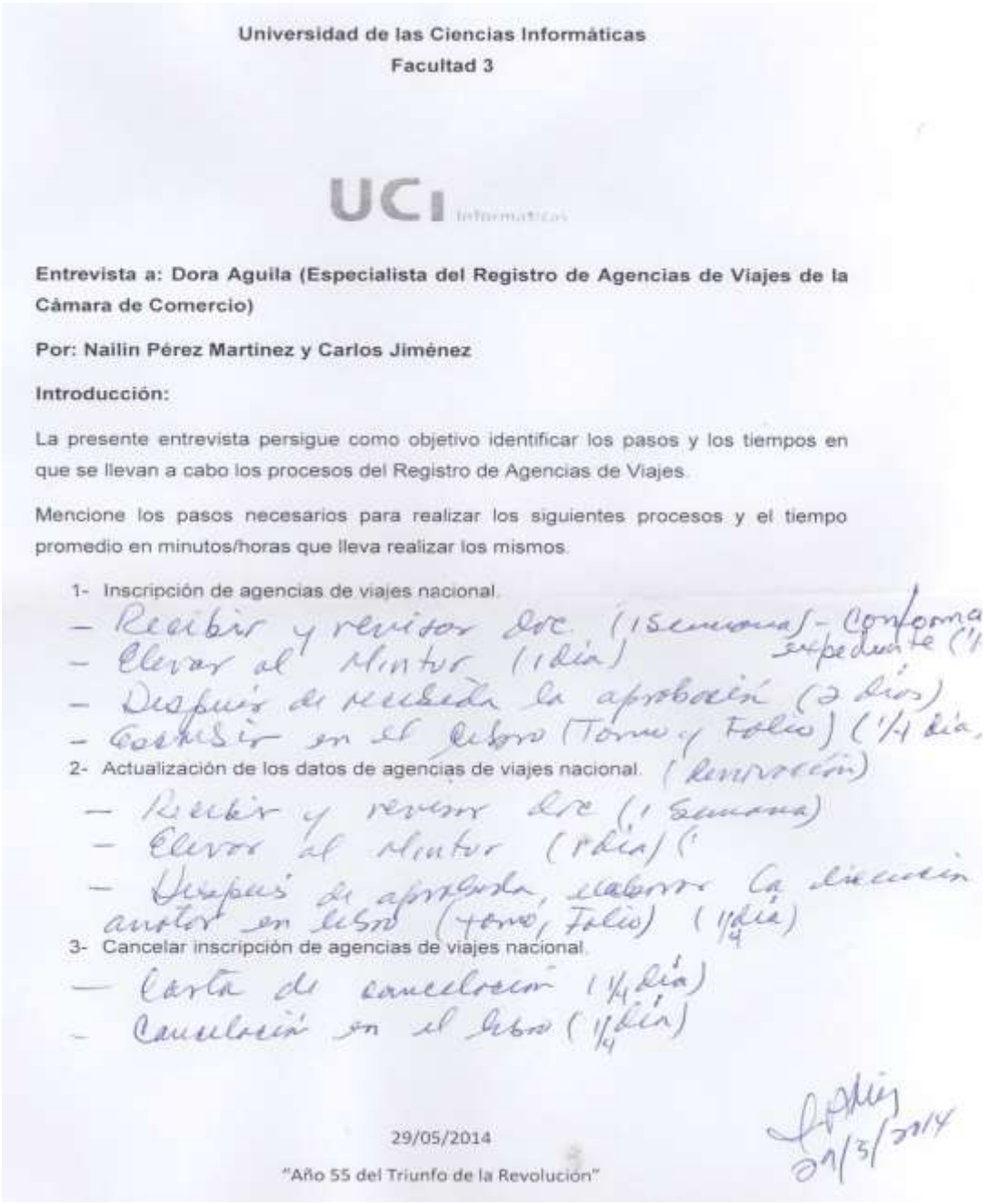
Parámetros	Evaluación	
	Categoría	Criterio
Responsabilidad	B	C < = P
	M	P <C< 2*P
	A	C> 2*P
Complejidad de implementación	B	C < = P
	M	P <C< 2*P
	A	C> 2*P
Reutilización	B	C> 2*P
	M	P <C< 2*P
	A	C < = P

Tabla 12. Medidas del cálculo del dominio para la métrica TOC

Anexo 6. Documento Casos de Pruebas para la prueba de Caja Blanca.

Consultar el documento de nombre —Caso de prueba para el camino básico, en la carpeta Documentos del proceso de Registro de Agencia de Viaje.

Anexo 7. Entrevista realizada a especialista del módulo Agencia de Viajes de la Cámara de Comercio.



Universidad de las Ciencias Informáticas

Facultad 3

UCI informatica

Entrevista a: Dora Aguila (Especialista del Registro de Agencias de Viajes de la Cámara de Comercio)

Por: Nailin Pérez Martinez y Carlos Jiménez

Introducción:

La presente entrevista persigue como objetivo identificar los pasos y los tiempos en que se llevan a cabo los procesos del Registro de Agencias de Viajes.

Mencione los pasos necesarios para realizar los siguientes procesos y el tiempo promedio en minutos/horas que lleva realizar los mismos.

1- Inscripción de agencias de viajes nacional.

– Recibir y revisar doc (1 semana) - Conforma expediente (1
– Elevar al Mintur (1 día)
– Después de recibida la aprobación (2 días)
– Confeccionar en el libro (Tomo y Folio) (1/4 día,

2- Actualización de los datos de agencias de viajes nacional. (Renovación)

– Recibir y revisar doc (1 Semana)
– Elevar al Mintur (1 día) (
– Después de aprobada, elaborar la licencia anotar en libro (Tomo, Folio) (1/4 día)

3- Cancelar inscripción de agencias de viajes nacional.

– Carta de cancelación (1/4 día)
– Cancelación en el libro (1/4 día)

29/05/2014

"Año 55 del Triunfo de la Revolución"

27/3/2014

Figura 17. Respuesta de la entrevista realizada a la especialista de la Cámara de Comercio.

GLOSARIO

Variable: Es un hecho, proceso, fenómeno, atributo, propiedad, concepto que puede cambiar cuantitativa o cualitativamente.

Variable Dependiente: Es el consecuente o efecto que cambia por influencia de la variable independiente.

Variable Independiente: Es el antecedente o causa que genera cambios en otra variable.

Modelo de desarrollo: Son una representación simplificada de un proceso de software, presentada desde una perspectiva específica.

Marco de Trabajo: En términos generales, define un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.