Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Registro de Trámites Migratorios 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

Autores:

Leandro Rodríguez Cedeño

Dalgis Susel Antúnez Ginarte

Tutores:

Ing. Julio Cesar Prieto Alvarez

Ing. Juan David Gómez Amador

La Habana, Junio 2014

"Si no existe organización, las ideas, después del primer impulso, van perdiendo eficacia, van cayendo en la rutina, van cayendo en el conformismo y acaban por ser simplemente un recuerdo".

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

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

Para que así conste, firmamos la presente a los __días del mes de_____ del año 2014.

Firma del Autor	Firma del Autor
Dalgis Susel Antúnez Ginarte	Leandro Rodríguez Cedeño
Firma del Tutor	Firma del Tutor
Ing. Julio César Prieto Álvarez	Ing. Juan David Gómez Amador

AGRADECIMIENTOS

Agradecer en primer lugar a mi mamá y a mi papá, por darme la vida y abrirme el camino para lograr un buen futuro para mi vida, por todo el apoyo y comprensión en todos mis años de enseñanza.

Agradecerle a la gordita de mi corazón, mi hermana, que la quiero mucho y aunque estuvo tres años lejos de mí, los dos anteriores que compartió conmigo significaron mucho y fue el motor impulsor cuando saqué mi primer 2 en la universidad.

A mi compañero de tesis, por compartir conmigo momentos difíciles y saber ponerle el extra como me dice cada vez que el ánimo me venía al piso.

A sus padres que significaron mucho para mí, fueron mis segundos padres y siempre estaban apoyándonos.

A todos mi compañeros de aula y de la universidad, los que compartieron conmigo estos difíciles y maravillosos 5 años y permitirme por un tiempo formar parte de sus vidas.

A mis profesores, por haberme enseñado y soportado todos estos años.

A mis tutores, por ayudarme cuando lo necesitaba.

A mis compañeros del proyecto, por haberme ayudado.

A todos aquellos que me ayudaron a realizar este sueño, muchas gracias.

A todos en general.

Dalgis Susel

Primeramente agradecer a mis padres por su apoyo incondicional en todo momento, por educarme de la mejor manera y ser la razón fundamental de todos mis logros.

A mi familia en general, por siempre estar cuando la necesito.

A mis amigos de Matanzas, por brindarme su ayuda a pesar de la lejanía.

A mis compañeros de la carrera, que tantos momentos buenos y malos pasamos juntos.

A mi compañera de tesis por enfrentar conmigo esta dura batalla.

A los profesores que durante la carrera influyeron en mi preparación como ingeniero y como persona.

A los profesores del laboratorio 315 por su apoyo en la realización de la tesis.

A las técnicas del centro por su paciencia y pasar varias madrugadas despiertas mientras yo trabajaba.

En fin, a todos los que de una forma u otra han influido a que hoy esté cumpliendo el sueño de mi vida: "ser ingeniero".

Leandro

DEDICATORIA

A mi mamá por todo el apoyo y amor brindado.

A mi padre por sus consejos y su experiencia transmitida.

A mi hermana, abuela, tíos y mis primos.

A toda mi familia en general porque son lo que más quiero en el mundo.

A mis amigos y amigas.

A todo el que creyó en mí, el que en algún momento me brindó su ayuda.

A todos.

Dalgis Susel

Dedico este trabajo en especial a mis padres, que son mi razón de ser.

A mi familia.

A mis amigos de siempre.

Leandro

RESUMEN

La Cámara de Comercio de la República de Cuba es una institución vinculada al comercio, la industria y los servicios. Actualmente realiza los procesos de Registros de Empresas Cubanas, Sucursales Extranjeras, Importadores y Exportadores, Agencias de Viajes y Trámites Migratorios. Este último presenta problemas como el manejo manual de la información y la demora en la búsqueda de datos previamente registrados en libros y expedientes físicos. Para mejorar el proceso de registro en esta entidad y como parte de la política de informatización que se lleva a cabo en el país, se determinó desarrollar el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC) compuesto por cinco módulos para la gestión de los procesos mencionados.

El presente trabajo muestra las herramientas, métodos y tecnologías que se utilizan en la implementación del módulo de Registro de Trámites Migratorios. Se expresan las actividades realizadas por cada una de las fases del modelo de desarrollo para obtener una solución consistente. Para la validación del módulo se utilizaron métricas de calidad.

PALABRAS CLAVES

Cámara de Comercio de la República de Cuba, módulo, Registro de Trámites Migratorios.

ÍNDICE GENERAL

Índice de Imágenes

ÍNDICE DE IMÁGENES

Índice de Tablas

INTRODUCCIÓN

Las relaciones económicas necesitan nuevas formas de servicios y requieren de la actualización y empleo de las Tecnologías de la Información y las Comunicaciones (TIC). Con el objetivo de aumentar las relaciones internacionales se hace necesario contar con la disposición tanto de los empresarios como del personal que gestiona la información de encontrar mejores formas de lograr mercados convenientes. Por tal motivo las Cámaras de Comercio pretenden acercar al hombre a la cultura empresarial del mundo, a los esquemas de gestión de la innovación empresarial y a la innovación comercial. En definitiva, la adecuación de la vida económica a los tiempos presentes y venideros.

En la actualidad toda institución debe saber cuál es el rol que le toca cumplir en la sociedad dentro del marco del servicio que brindará. La Cámara de Comercio, no sólo debe ser de excelencia en su servicio, además debe realizar acciones que se acerquen a ese objetivo finalista que es ofrecer a la sociedad caminos y conductas dirigidas al progreso económico, al progreso comercial y empresarial.

La Cámara de Comercio de la República de Cuba es una asociación de empresas vinculadas al comercio, la industria y los servicios, con reconocimiento ante los organismos del Estado, que permite orientar las mejores alternativas para el desarrollo de la actividad empresarial de las entidades que a esa asociación pertenecen. Las acciones que se acometen en ella responden al crecimiento de la sociedad, con un intercambio amplio, con desarrollo de todos los componentes involucrados y donde el país obtenga los mejores resultados socio – económicos. Constituye una herramienta para la reinserción de la economía cubana en el mundo de las relaciones económicas internacionales, pues potencia e intercambia información valiosa en torno a las posibilidades de negocios a escala mundial. (Cámara de Comercio de la República de Cuba, 2009)

La Cámara de Comercio de la República de Cuba en conjunto con el Centro de Gobierno Electrónico (CEGEL), adscrito a la Facultad 3 de la Universidad de las Ciencias Informáticas, desarrolla el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC) para informatizar los diferentes procesos que en ella se realizan. Esta institución actualmente cuenta con cinco procesos para el registro de: Empresas Cubanas, Empresas Importadoras y Exportadoras, Agencias de Viaje, Sucursales Extranjeras y Trámites Migratorios. En este último se gestionan a diario las solicitudes que se reciben de permisos de trabajo, las actualizaciones y renovaciones de los mismos y las solicitudes de trámites migratorios del personal que se encuentra laborando en sucursales extranjeras inscritas en el Registro de Sucursales y Agentes de Sociedades Mercantiles Extranjeras o en el Registro de Sucursales de Agencias de Viajes y de Contratos de representación

turística para extranjeros. Esta entidad hoy día presenta dificultades en el proceso de registro de trámites migratorios las cuales influyen negativamente en el desarrollo económico del país:

- Se realiza de forma manual con la ayuda del Sistema de Gestión de Bases de Datos Microsoft Access 2003 como herramienta para el respaldo digital datos pertenecientes a los trámites registrados.
- Este aspecto influye negativamente en la capacidad de responder el creciente volumen de solicitudes debido a los cambios económicos que se llevan a cabo en el país, pues demora a los funcionarios de 30 a 35 minutos realizar un trámite.
- Estos datos son insertados al final del día obtenido de libros y expedientes que forman parte de este proceso.
- Debido a la cantidad de información almacenada en formato duro la búsqueda de trámites previos registrados se hace compleja, provocando además que los funcionarios puedan tener acceso a información no autorizada.

Teniendo en cuenta las dificultades que presenta el Registro de Trámites Migratorios se identificó el siguiente **problema a resolver**: ¿Cómo contribuir a aumentar la disponibilidad de la información asociada al proceso de Registro de Trámites Migratorios de la Cámara de Comercio de la República de Cuba? En consecuencia el **objeto de estudio** es: el proceso de desarrollo de Sistemas de Informatización Registral.

Para guiar el trabajo se define como **objetivo general:** desarrollar un Módulo para el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba que contribuya a aumentar la disponibilidad de la información asociada al proceso de Registro de Trámites migratorios de la Cámara de Comercio de la República de Cuba. Este objetivo se enmarca dentro del **campo de acción:** Sistema informático para el Registro de Trámites Migratorios de la Cámara de Comercio de la República de Cuba.

Del objetivo general se derivan los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación para determinar las diferentes tecnologías a emplear en la solución.
- Desarrollar el modelo de diseño del módulo en cuestión y a partir de este realizar la implementación del sistema, para alcanzar la propuesta de solución planteada.
- Validar la solución propuesta para evaluar el cumplimiento de los objetivos planteados en la investigación.

Para satisfacer estos objetivos específicos quedan definidas las siguientes **tareas de investigación**:

- Estudio y análisis de los procesos de Registro de Trámites Migratorios de la Cámara de Comercio de la República de Cuba.
- Definición del modelo de desarrollo y herramientas a utilizar para el desarrollo de los componentes.
- Levantamiento de requisitos del proceso de Registro de Trámites Migratorios.
- Validación de los requisitos definidos para el proceso de Registro de Trámites Migratorios.
- Desarrollo del diagrama de paquetes, diagrama de clases y el modelo de datos del módulo.
- Implementación del módulo Registro de Trámites Migratorios.
- Desarrollo del diagrama de componentes y de despliegue.
- Validación de la solución propuesta aplicando pruebas de caja negra y caja blanca.
- Valoración de las variables dependientes e independientes para la solución del problema.

La investigación se sustenta en la siguiente **idea a defender:** con el desarrollo de un Módulo para el Registro de Trámites Migratorios del Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba se contribuirá a aumentar la disponibilidad de la información asociada al proceso de Registro de Trámites Migratorios de la Cámara de Comercio de la República de Cuba.

Operacionalización de las variables

Variable dependiente	Dimensiones	Indicadores	Forma de valoración
Disponibilidad	Tiempo	Tiempo de respuesta de la aplicación (TR)	-TR < 3 segundos de acuerdo al RnF7 de Usabilidad.
	Acceso	Acceso a la información según los privilegios	-Encuesta. -Análisis de la arquitectura del marco de trabajo.

Tabla 1: Operacionalización de la variable dependiente.

Variable Independiente	Dimensiones	Indicadores	Forma de valoración
Módulo de Registro de Trámites Migratorios	Alcance	Cumplimiento de los requisitos	Índice de requisitos implementados = (Cantidad de requisitos implementados) / (Cantidad total de requisitos)
	Calidad	Calidad Externa	Pruebas de liberación

		Calidad Interna	Pruebas Internas (Pruebas de caja negra y caja blanca)

Tabla 2: Operacionalización de la variable independiente.

En el presente trabajo se hace uso de diversos métodos científicos, tanto teóricos como empíricos, propios de la profesión informática y se utilizan durante toda la investigación. Dentro de los métodos teóricos se emplean:

- Analítico-Sintético: En el análisis de los referentes teóricos y la bibliografía en cuestión donde se hizo una extracción y síntesis de los elementos más significativos relacionados con el desarrollo que ha tenido el proceso de Registro de Trámites Migratorios. Esto sirvió como base teórica para el desarrollo de la investigación.
- Modelación: Para crear abstracciones de la solución en aras de comprender la realidad; es usado en el proceso de desarrollo de software para el diseño de diagramas, representar datos y crear otros artefactos.
- Histórico-Lógico: Para conocer la evolución y desarrollo del proceso de Registro de Trámites Migratorios así como las principales etapas de su desenvolvimiento funcional y las conexiones históricas fundamentales; es decir, ofrece la oportunidad de comprender la historia de este proceso.

El método empírico utilizado es:

- Entrevista: Para la recopilación de información mediante conversaciones con los funcionarios de la Cámara de Comercio de la República de Cuba, permitiendo adquirir información relacionada con el funcionamiento del proceso de Registro de Trámites Migratorios, siendo de gran ayuda en la etapa de levantamiento de requisitos.

El presente trabajo está compuesto por tres capítulos como se describe a continuación:

Capítulo 1. Fundamentación teórica: comprende el estudio del estado del arte. Brinda un breve acercamiento a los principales conceptos asociados al dominio del problema. También se describen las tecnologías, metodologías, herramientas, pruebas de software y métricas utilizadas para el desarrollo de la solución propuesta.

Capítulo 2. Descripción de la solución propuesta: se propone el diseño de la solución a partir de la especificación de los requisitos de software. Se exponen los patrones de diseño, los estilos arquitectónicos utilizados y se realiza una breve descripción de la arquitectura para lograr la comprensión del diseño. Se describen los estándares de codificación. Se muestra como resultado del modelo de diseño, los diagramas de clases, el diagrama de paquetes y el modelo de datos

utilizado en el sistema. Se muestran los resultados obtenidos al aplicar las técnicas de validación de requisitos. Se presentan los resultados alcanzados al aplicar las métricas para validar el diseño y la implementación a partir de la realización de los métodos de caja blanca y caja negra realizadas al código del software y a la interfaz del código respectivamente.

Capítulo 3. Valoración de la solución propuesta: Muestra la valoración de las variables dependientes e independientes, dándole solución al problema planteado.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se hace referencia a los principales aspectos y definiciones que contribuyen a la comprensión de los términos que se emplean a lo largo del trabajo de diploma. Se realiza un estudio de los sistemas similares existentes a nivel mundial con el objetivo de tomar ejemplos de arquitecturas, estructuras, lenguajes y demás aspectos que pueden ser de utilidad en el desarrollo la solución propuesta. Se analizan las tendencias actuales para el desarrollo de software con el objetivo de encontrar las herramientas, técnicas y métodos necesarios para llevar un adecuado control y desarrollo del software. Se investigan las características de los diferentes lenguajes de programación y de modelado con el objetivo de seleccionar el adecuado para la implementación del módulo y para el diseño. Se analizan las tecnologías de desarrollo existentes así como el sistema de gestión de base de datos. Se desarrolla el estudio de los posibles patrones de diseño y estilos arquitectónicos a utilizar, las pruebas de software a realizar y las métricas a aplicar en el análisis y el diseño.

1.1. Sistemas similares al Registro de Trámites Migratorios

En este epígrafe se muestran las características de sistemas enfocados al registro de Trámites Migratorios a nivel mundial. Se describe el Sistema Electrónico de Trámites Migratorios (SETRAM), la Dirección Nacional de Migraciones de Argentina, Servicio Autónomo de Identificación, Migración y Extranjería (SAIME) y el Sistema Electrónico de Inmigración USCIS (USCIS ELIS).

1.1.1. Sistema Electrónico de Trámites Migratorios

El Sistema Electrónico de Trámites Migratorios (SETRAM), es una aplicación web implementada en México que registra y da seguimiento a las solicitudes de trámites que ingresan al Instituto Nacional de Migración. A este sistema ingresa la totalidad del registro y control de actividades administrativas, recepción, asignación, despacho y notificación de trámites migratorios. (Migratorios, 2014)

1.1.2. Dirección Nacional de Migraciones de Argentina

El sitio web oficial de la Dirección Nacional de Migraciones de Argentina permite, vía Internet, realizar diversos trámites migratorios que agilizan los procesos de entrada y salida al país. De una forma u otra controlan la estancia de los extranjeros en ese país. Los trámites que se pueden realizar en este sitio son los siguientes: ingreso y egreso al país, residencias, tramitación de ingresos, DNI extranjeros, pasaportes extranjeros, certificaciones e identidad de género. (Argentina, 2014)

1.1.3. Servicio Autónomo de Identificación, Migración y Extranjería

El Servicio Autónomo de Identificación, Migración y Extranjería (SAIME) perteneciente al Gobierno Bolivariano de Venezuela, es el encargado de todo lo relacionado con identificación y extranjería. Para los venezolanos, era muy difícil, casi imposible, obtener pasaporte. Para obtener tan preciado documento, había que apelar al tráfico de influencias o a cierta "mafia" a quienes había que "comprarles" los formularios y "citas" de solicitud. Este sistema venezolano de trámites trae grandes ventajas para la obtención de planillas y citas debido a su fácil acceso por Internet. Se disminuye el tiempo de realización de los trámites, de entrega de documentos y requisitos. (Venezuela, 2014)

1.1.4. Sistema Electrónico de Inmigración USCIS

El Sistema Electrónico de Inmigración USCIS (USCIS ELIS) es un sistema basado en cuentas en línea optimizada, que mejora el servicio al cliente y la calidad de los procesos. A través de USCIS ELIS los clientes pueden ver sus solicitudes de beneficios, recibir la notificación electrónica de las decisiones y las actualizaciones de estado en tiempo real. Permite a los solicitantes una cómoda y segura gestión de sus cuentas y realizar pagos en línea. (Security, 2014)

A partir de la investigación realizada a estos sistemas se realizó un análisis comparativo con los parámetros que deben cumplir para satisfacer las necesidades de la Cámara de Comercio. A continuación se presenta una tabla que muestra los resultados obtenidos de acuerdo a los factores seleccionados. Se tomó en cuenta la cantidad de procesos similares con respecto a los que se desarrollan en la Cámara de Comercio, el total de trámites que desarrolla cada uno, la integración que poseen con otros módulos, su entorno de aplicación, cantidad de campos similares en las planillas generadas por cada sistema y si son aplicables a la legislación cubana.

Nombre del software	SETRAM	Dirección Nacional de Migraciones de Argentina	ELIS	SAIME
Total de Trámites	4	8	5	10
Integración con otros módulos	No	No	No	No
Entorno de aplicación	Aplicación Web	Aplicación Web	Aplicación Web	Aplicación Web
Cantidad de procesos similares	4	5	2	7
Cantidad de campos	15	17	10	20

similares en planillas				
Aplicable a la legislación cubana	No	No	No	No

Tabla 3: Aspectos comparativos entre los sistemas analizados**.**

Luego de lo explicado anteriormente, se llegó a la conclusión de que ninguno de estos sistemas cumple en su totalidad con los parámetros necesarios para satisfacer las necesidades del cliente. El proceso de trámites migratorios que se efectúa en la Cámara de Comercio de la República de Cuba está dirigido a los funcionarios que elaboran los trámites, mientras que en estos sistemas está enfocado a los usuarios que lo realizan. El entorno de aplicación de estos sistemas es web y el sistema que se desea implementar debe ser de escritorio. Existe correspondencia entre las planillas generadas con respecto a las que se elaboran en la entidad cliente y en algunos procesos que se ejecutan en estos sistemas. Los sistemas analizados no son aplicables a la legislación cubana porque todos presentan leyes propias de sus países. Se recomienda tener en cuenta las similitudes en cuanto a planillas y procesos como guía para la confección del módulo.

1.2. Ingeniería de software

La ingeniería del software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después que se utiliza. (Sommerville, 2005)

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

1.2.1. Modelo de desarrollo de software: Programa de mejora

Para el desarrollo de la investigación se utilizó el Programa de Mejora propuesto por la UCI como parte del proceso de reestructuración de la producción en la universidad, este programa define un modelo de desarrollo para representar los procesos asociados al desarrollo de software y la prestación de servicios de las TIC. Estos modelos son conjuntos de buenas prácticas que ayudan a las organizaciones a mejorar sus procesos y proporciona un conjunto completo e integrado de guías para desarrollar productos y servicios estableciendo un Programa de Mejora. Transformando de

procesos de desarrollo a procesos institucionalizados.El Programa de Mejora explota las características fundamentales del Proceso Unificado Relacional (RUP).

Define nueve fases: Estudio Preliminar, Modelo del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas Internas, Pruebas de Liberación, Despliegue y Soporte. Dentro de cada una de estas fases se generan un conjunto de artefactos necesarios para la construcción de un sistema de software con la calidad requerida. Para el desarrollo de la aplicación se tomaron como referencia las fases desde los Requisitos hasta las Pruebas Internas y de Liberación. En cada una se desarrollan artefactos como resultado de aplicarlas en todo el ciclo de vida de desarrollo del software. Además define las herramientas necesarias para la elaboración de diagramas y demás artefactos relacionados con el diseño. (Programa de Mejora, 2011)

A continuación se describen algunas de estas etapas.

1.2.2. Requisitos

La fase de Requisitos tiene como función principal desarrollar un modelo del sistema que se desea implementar. Incluye un conjunto de requisitos y servicios que describen todas las interacciones que tendrán los usuarios con el software, estos responden a los requisitos funcionales y no funcionales del sistema. Para la presente investigación esta fase se realizará en conjunto con los funcionarios de la Oficina de Trámites Migratorios de la Cámara de Comercio de la República de Cuba. (Programa de Mejora, 2011)

1.2.3. Análisis, diseño e implementación

En la fase de Análisis y Diseño, a través de los modelos de análisis se pueden refinar y estructurar los requisitos para conseguir una compresión más precisa y una descripción fácil de mantener. Además es modelado el sistema incluyendo su arquitectura para el soporte de los requisitos funcionales y no funcionales. Son desarrollados el documento de arquitectura, diagrama de clases del diseño, diagrama de paquetes y el modelo de datos.

En la fase de Implementación a partir de los resultados del análisis y del diseño se implementa el sistema en términos de componentes, mediante ficheros de código fuente, scripts, ejecutables y similares. Son elaborados el diagrama de componentes y el diagrama de despliegue. (Programa de Mejora, 2011)

1.2.4. Pruebas Internas y de Liberación

Al concluirse cada una de las fases en el desarrollo de un software se realizan un conjunto de validaciones y pruebas que permiten determinar si el resultado es el esperado.

Las pruebas internas se realizan por los desarrolladores del módulo con el objetivo de verificar el resultado de la implementación. Durante esta fase se desarrollan los diseños de casos de prueba. Las pruebas de liberación son diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software (LIPS)[1] y son aplicadas por el Centro Nacional de Calidad de Software (CALISOFT) y el grupo de calidad de CEGEL a todos los artefactos de los proyectos antes de ser entregados al cliente para su aceptación. (Programa de Mejora, 2011)

Para la validación del diseño se aplican métricas orientadas a las clases del módulo. Los desarrolladores aplican métodos de prueba como: caja blanca y caja negra, confeccionando los Casos de Pruebas (CP) correspondientes para cada requisito funcional.

A continuación se explica el funcionamiento de los métodos de pruebas mencionados.

Prueba de caja blanca

Se centran en evaluar la ejecución por lo menos una vez de cada sentencia del programa. Requieren de conocimiento de la estructura interna del programa. Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones o bucles para determinar si el estado real coincide con el esperado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar. (Pressman, 2007)

Estas pruebas deben garantizar como mínimo que:

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- Se ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez. (EVA, 2014)

En este método se aplica la prueba del camino básico que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Para la prueba del camino básico se utiliza la métrica complejidad ciclomática, empleada para definir el número de caminos independientes del conjunto básico de un programa una vez obtenido el valor

[1] Laboratorio Industrial de Pruebas de Software (LIPS): Protagonista en Cuba en la realización de evaluaciones a las aplicaciones informáticas para establecer las estrategias que le permitan ofertar servicios especializados en pruebas, empleando la norma reconocida internacionalmente NC-ISO/IEC 17025: 2006.

de la métrica facilitando un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. (Pressman, 2007)

Para el desarrollo de la prueba del camino mínimo se siguen una serie de pasos para llegar a obtener una descripción de todo el proceso que realiza el método de caja blanca. Primeramente usando el código base, se dibuja el correspondiente grafo de flujo y luego se determina la complejidad ciclomática del grafo del flujo resultante. Se puede determinar el valor de la complejidad ciclomática denotada por V (G) por tres formas:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V (G), de un grafo de flujo G se define como: V (G)=A-N+2. Donde A es el número de aristas del grafo de flujo y N es el número de nodos del grafo.
- 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.

Después de obtener la complejidad ciclomática se determina un conjunto básico de caminos linealmente independientes. Para terminar se preparan los casos de prueba que forzarán la ejecución de cada camino del conjunto básico. (Pressman, 2007)

Prueba de caja negra

Se centran en verificar el cumplimiento de los requisitos funcionales del software. Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. (Pressman, 2007)

Permiten encontrar:

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

Con el objetivo de detectar fallas en la implantación del sistema se realizan pruebas de aceptación con la participación de los analistas y el cliente en un ambiente de trabajo real, verificando las funcionalidades del sistema. Durante todas las fases del modelo de desarrollo se utilizan herramientas y tecnologías como apoyo al desarrollo de la propuesta de solución.

A continuación se muestra el estudio realizado a determinados elementos claves para el correcto desarrollo de cada una de las fases del ciclo de vida del software, comenzando por los lenguajes a utilizar.

1.3. Lenguajes

Según lo planteado por la (IEEE, 1999), un lenguaje puede ser entendido como un recurso que hace posible la comunicación. Este capítulo se centra es describir las características de los lenguajes de programación, de modelado y de consulta utilizados en la presente investigación.

1.3.1. Lenguajes de programación

En informática, cualquier lenguaje artificial puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Un lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora**.** (Definición. De, 2012)**.** Para el desarrollo de este sistema se decidió utilizar el lenguaje de desarrollo Java debido las siguientes características:

Java es un lenguaje de alto nivel basado en el paradigma Orientado a Objetos cumpliendo con características como encapsulación, herencia y polimorfismo. Es un lenguaje de alto nivel que junto a la Máquina Virtual de Java o JVM (del inglés *Java Virtual Machine*), permite que los proyectos sean ejecutados en diferentes entornos de hardware o software. Sus usos se extienden desde dispositivos móviles, aplicaciones de servidor y clientes, entre otras. Para el desarrollo de proyectos con un grado de complejidad técnica según las necesidades de estos, se puede hacer uso de multihilos, que garantiza la realización de actividades simultáneas logrando un resultado en cuanto a rendimiento y respuestas del sistema. Permite la búsqueda y tratamiento de errores en tiempo de ejecución por lo que se considera un lenguaje robusto. (Oracle Corporation, 2013)

1.3.2. Lenguaje de modelado

El Leguaje Unificado Modelado (UML) es un lenguaje de modelado que se centra en la representación gráfica de un sistema. El lenguaje UML tiene una notación gráfica expresiva que permite representar en mayor o menor medida todas las fases de un proyecto de software: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, entre otros, hasta la implementación y configuración con los diagramas de despliegue. UML en su versión 2.0 sirve para el modelado completo de sistemas complejos, tanto en el diseño del software como para la arquitectura de hardware donde se ejecuten. (Larman, 2005)

UML es además un método formal de modelado y aporta las siguientes ventajas:

- Mayor rigor en la especificación.

- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto. (James Rumbaugh, 2000) (Larman, 2005)

1.3.3. Lenguaje de consulta

Un lenguaje de consulta es un lenguaje informático usado para hacer consultas en bases de datos y sistemas de información. El Java Persistence Query Language (JPQL) es un lenguaje de consulta independiente de plataforma orientado a objetos definido como parte de la especificación Java Persistence API. JPQL se utiliza para hacer consultas en entidades almacenadas en bases de datos relacionales. Está fuertemente inspirado en el Lenguaje de Consulta Estructurado (SQL por sus siglas en inglés), y sus consultas se asemejan a la sintaxis de las consultas SQL, solo que operan contra objetos entidad de Java Persistence API (JPA por sus siglas en inglés) en lugar de hacerlo directamente con las tablas de base de datos. (Java Persistence Query Language (JPQL), 2012)

Una vez mostrado los lenguajes a utilizar se analizan las herramientas necesarias para ponerlos en práctica.

1.4. Herramientas

Con el objetivo de lograr un buen resultado en la solución del software, fue necesario el uso de herramientas. A continuación se describen las características del NetBeans 7.2, Visual Paradigm 8.0, GlassFish 3.1.2, y PostgreSQL 9.3 que son las necesarias a utilizar en la presente investigación propuestas por el Programa de Mejora de la Universidad.

1.4.1. Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE por sus siglas en inglés) es un programa compuesto por un conjunto de herramientas para programar en un lenguaje de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (Fergarciac , 2013)

Características

Un IDE debe tener las siguientes características

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis

- Extensiones y Componentes para el IDE
- Integración con marcos de trabajo
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda (Fergarciac , 2013)

A continuación se describe el IDE de desarrollo utilizado para la implementación de la solución.

NetBeans 7.2

NetBeans es un IDE disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans es de código abierto y permite a los desarrolladores crear rápidamente aplicaciones para móviles, para la web y de escritorio.

Netbeans 7.2 ofrece una mejora significativa del rendimiento y la experiencia de codificación, con nuevas capacidades de análisis de código estático en el editor Java y escaneo de proyecto inteligente. Además cuenta con las siguientes características para la plataforma Java 2 Enterprise Edition (Java 2EE por sus siglas en inglés):

- Finalización de código para las consultas de Java Persistence con nombre y declaraciones JPQL.
- Soporte para GlassFish 3.1.2.
- Clúster e Instancia soporte de implementación para GlassFish.
- Soporta la Java Persistence, servicios web, EJB, entre otros.(Oracle Corporation, 2013)

Se utiliza el IDE de desarrollo Netbeans 7.2 por su adecuado soporte con el servidor de aplicaciones GlassFish y permite utilizar declaraciones JPQL para las consultas de Java Persistence.

1.4.2. Herramienta CASE

Las herramientas CASE ayudan a los gestores y practicantes de la ingeniería del software en todas las actividades asociadas al proceso de desarrollo de software[2]. Automatizan las actividades de gestión de proyectos. Permiten gestionar todos los productos de los trabajos elaborados a través del proceso, y ayudan a los ingenieros en el trabajo de análisis, diseño e implementación.

[2] Proceso de desarrollo de software: Es un conjunto de personas, estructuras de organización, reglas, políticas, actividades y sus procedimientos, componentes de software, metodologías y herramientas utilizadas o creadas específicamente para definir, desarrollar, ofrecer un servicio, innovar y extender un producto de software.

Estas herramientas son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, utilizadas para reducir su costo. Sus productos están centrados en la metodología del Proceso Unificado de Desarrollo (RUP por sus siglas en inglés). Además ayuda en las tareas de diseño de proyectos, documentación o detección de errores y compilación, entre otras acciones desarrolladas en todos los aspectos de ciclo de vida de desarrollo del software. Contienen facilidades para la revisión de aplicaciones, soporte para el desarrollo de prototipos de sistemas y generación de código. (Alarcon, 2004) Para el desarrollo del módulo se seleccionó la siguiente herramienta CASE debido a sus características y a su integración con la herramienta de desarrollo seleccionada Netbeans.

Visual Paradigm 8.0

Visual Paradigm es una herramienta que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad y con un bajo costo. Permite construir diagramas de diversos tipos, generar código desde diagramas y generar documentación. Soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software.

Proporciona ventajas como:

- Dibujo: Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica: Controla que el modelado con UML sea correcto.
- Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Reutilización: Facilita la reutilización, ya que se dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de informe: Permite generar diversos informes a partir de la información introducida en la herramienta. (Lianet Cabrera González, 2012)

1.4.3. Servidor de aplicaciones

GlassFish 3.1.2

GlassFish es una comunidad y un proyecto de Servidor de Aplicaciones que fue iniciado por Sun Microsystems para la plataforma JEE. Es la implementación de referencia de la plataforma mencionada anteriormente y soporta Enterprise Java Beans (EJB por sus siglas en inglés), JPA, Java Server Faces, Java Server Pages, además de otros. GlassFish 3.1.2 es el sucesor de los

lanzamientos 3.0.x anteriores , ofreciendo un tiempo de ejecución modular basado en Open Services Gateway Initiative (OSGi) ahora con la agrupación con todas las funciones con la administración centralizada de varios clústeres y alta disponibilidad de componentes con estado. (Oracle Corporation, 2013)

1.4.4. Marco de trabajo Kairos 1.0

Este marco de trabajo fue desarrollado por el centro CEGEL, el cual está diseñado para la creación de aplicaciones con arquitectura Cliente-Servidor utilizando la plataforma Java 2EE. Consta de dos partes, una 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 una segunda que se encuentra en el lado del servidor donde se encuentra la lógica de negocio. Proporciona a las aplicaciones que lo utilicen funcionalidades y componentes tales como:

- Una interfaz de usuario básica.
- Un mecanismo para la internacionalización de las interfaces de usuario.
- Un mecanismo para el acceso a los procedimientos remotos publicados en un servidor de aplicaciones.
- Un mecanismo de mensajería.
- Un mecanismo para gestionar las excepciones lanzadas por la aplicación que agiliza y simplifica su desarrollo. (Daniel Sarmiento Sastriques, 2012)

1.4.5. Sistema de gestión de base de datos

PostgreSQL es un sistema de gestión de base de datos relacional y de código abierto. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Sus características técnicas la hacen una de las bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. (PostgresSQL, 2014)

1.5. Tecnologías

1.5.1. Java Persistence API

Java Persistence API (JPA) es el estándar para la gestión de la persistencia y provee facilidades para el mapeo objeto / relacional a desarrolladores de aplicaciones, haciendo uso del modelo de dominio de Java para administrar bases de datos relacionales. JPA es parte de la plataforma Java

2EE. Presenta soporte para varios sistemas de Gestión de Bases de Datos, especialmente PostgresSQL y soporta toda la implementación del estándar SQL. (Reinel Muñoz Perez, 2008)

1.5.2. Enterprise Java Beans

Para Java existen tres categorías que encapsulan las tecnologías existentes. Estas categorías son las siguientes:

- Java Standard Edition (JSE): dentro de esta categoría se aglomeran todas las tecnologías existentes en Java para sistemas simples o estándares.
- Java 2 Enterprise Edition (J2EE): esta categoría agrupa todas las tecnologías que proporciona Java para la implementación de aplicaciones empresariales.
- Java Micro Edition (JME): esta categoría contiene las tecnologías que proporciona Java para aplicaciones móviles. (Julio C. Prieto, 2011)

El diseño de la solución estará centrado en el uso de la categoría J2EE, especialmente en la tecnología Enterprise Java Beans (EJB).

Un EJB es un componente del lado del servidor que se debe desplegar sobre un contenedor de EJBs. Está conformado por diversos archivos (interfaces y clases Java). El contenedor provee a los EJBs servicios y controla su ciclo de vida. (Claudia A. Romina Ledesma, 2010)

La tecnología EJB permite el desarrollo rápido y simplificado de aplicaciones distribuidas. Proporciona un modelo de componentes distribuido estándar del lado del servidor. Los componentes encapsulan el comportamiento de la aplicación. Su objetivo es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial, para centrarse en el desarrollo de la lógica de negocio en sí.

Hay tres tipos de componentes EJB: beans de sesión, de mensaje y de entidad. Los beans de sesión y mensaje se usan para implementar la lógica de negocio y los de entidad, la persistencia.

Esta tecnología permite instalar una solución que se compone por un conjunto de módulos, gestionados por distintas máquinas virtuales, facilitando así que los componentes se distribuyan sin restricción alguna. Para el desarrollo de la solución se emplea esta tecnología pues el contenedor implementa un conjunto de servicios que reducen la implementación por parte de los desarrolladores. A continuación se exponen algunos de los servicios antes mencionados:

- Manejo de transacciones: controla las transacciones asociadas a las llamadas a los métodos del bean.
- Seguridad: controla el acceso a los métodos del bean.
- Concurrencia: llamada simultánea a un mismo bean desde múltiples clientes.

- Servicios de red: gestiona la comunicación entre el cliente y el bean en máquinas distintas.
- Gestión de recursos: gestión automática de múltiples recursos.
- Persistencia: sincronización entre los datos del bean y tablas de una base de datos.
- Gestión de mensajes: manejo de Java Message Service (JMS).
- Escalabilidad: posibilidad de constituir clúster de servidores de aplicaciones con múltiples hosts para poder dar respuesta a aumentos repentinos de carga de la aplicación con solo añadir hosts adicionales.
- Adaptación en tiempo de despliegue: posibilidad de modificación de todas estas características en el momento de despliegue del bean. (Julio C. Prieto, 2011)

Los componentes EJB deben ser accedidos a través de sus interfaces, las cuales están divididas en dos variantes: interfaces remotas que permiten que el componente sea accedido remotamente e interfaces locales que posibilitan que los componentes sean accedidos localmente.

1.6. Patrones de diseño

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema. (Larman, 2005) A continuación se explican algunos patrones existentes para asignar responsabilidades.

Patrones de Principios Generales para Asignar Responsabilidades (GRASP, por sus siglas en inglés).

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

El patrón Experto en información es el encargado de asignar las responsabilidades a aquellas clases que contengan la información necesaria para realizar una responsabilidad. Resolviendo el problema de saber a qué clase del sistema le corresponde realizar una o varias operaciones, en dependencia de la información que posea.

El patrón Creador es el responsable de asignarle a una clase la responsabilidad de crear una instancia de otra clase. Resolviendo así el problema de poder crear una nueva instancia de una clase mediante otras que registren, utilicen o contengan instancias de objetos de esa clase.

El patrón Controlador es el encargado de asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema, resolviendo el problema de comunicación de una clase con otra mediante una clase controladora.

El patrón Alta cohesión asigna una responsabilidad de forma tal que la cohesión siga siendo alta. La dificultad de entender, mantener, reutilizar y las afectaciones debido a los contantes cambios son resueltas con un alto acoplamiento debido a que las clases tienen la relación necesaria.

El patrón Bajo acoplamiento asigna una responsabilidad para conservar bajo acoplamiento. Al aplicar este patrón los problemas relacionados con la reutilización, el entendimiento y los cambios en las clases relacionadas serían resueltos pues existiría la relación necesaria de un elemento con otros.

Los patrones Grupo de los cuatro (GoF, por sus siglas en inglés) utilizados son:

Instancia única que es un patrón utilizado para asegurar la existencia de una instancia de una clase y que esta es accesible, visible y con una referencia permanente. Dándole solución al problema de encontrar un único punto de acceso local.

Fachada que es un patrón que dado el problema de la necesidad de una interfaz común unificada para un conjunto de implementaciones o interfaces brinda la posibilidad de un único punto de conexión con el subsistema. Este objeto fachada presenta una única interfaz unificada y es responsable de colaborar con los componentes del sistema. (Larman, 2005)

1.7. Estilos arquitectónicos

Buschmann define estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo. También incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación. (ERIKA CAMACHO, 2004). De acuerdo a las necesidades de una adecuada interacción entre el cliente y el servidor se seleccionaron algunos estilos arquitectónicos que van ayudar a una mejor estructura del sistema.

El estilo arquitectónico Cliente-Servidor, se modela como un conjunto de servicios proporcionados por los servidores y un conjunto de clientes que usan esos servicios. Los principales componentes de este modelo son:

- Un conjunto de servidores que ofrecen servicios a otros subsistemas, ejemplos los servidores de impresoras que ofrecen servicios de impresión, servidores de ficheros que ofrecen

servicios de gestión de ficheros y servidores de compilación que ofrecen servicios de compilación de lenguajes de programación.

- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores. Éstos son normalmente subsistemas en sí mismos. Puede haber varias instancias de un programa cliente ejecutándose concurrentemente.
- Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario ya que los clientes y los servidores podrían ejecutarse sobre una única máquina.

También se seleccionó el estilo N-capas, según Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas. (Carlos Reynoso, 2004), (Sommerville, 2005).

Cada capa está compuesta por varios componentes que van a permitir las relaciones entre los elementos de las capas. Por tal motivo fue seleccionada la arquitectura Basado en componentes el cual se basa en principios definidos por una ingeniería de software específica. (Sommerville, 2005)

- Las interfaces están separadas de las implementaciones, las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de observación, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.
- En cuanto a las restricciones, puede admitirse que una interfaz sea implementada por múltiples componentes.

1.8. Valoración de la solución

En el marco teórico de la investigación fueron definidas en el problema a resolver como variables: variable independiente: Módulo de Registro de Trámites Migratorios y variable dependiente: Disponibilidad. En cuanto a la valoración del Módulo de Registro de Trámites Migratorios, se define conceptualmente hasta lograr un entendimiento de su significado, luego se aplica a la investigación, según los procesos de negocio llevados a cabo en la Cámara de Comercio. En cuanto a la Disponibilidad, se definen dimensiones y a partir de ellas una serie de indicadores, para su posterior valoración.

Para llevar a cabo la valoración correspondiente se aplica una encuesta a especialistas de la Cámara de Comercio, específicamente los funcionarios del departamento de Trámites Migratorios, que son las personas encargadas de trabajar directamente con el proceso de registro de trámites. Luego de aplicada la encuesta se analizan los resultados y se arriba a conclusiones.

Además de esta encuesta se realiza una valoración del tiempo de respuesta obtenido a la hora de registrar información, con la ayuda de las funcionalidades que proporciona la herramienta Netbeans; con respecto al tiempo especificado en el RnF 7, que plantea: “el tiempo de respuesta brindado por el sistema será menor que 3 segundos”.

1.9. Conclusiones del capítulo

En este capítulo se realizó la elaboración del marco teórico de la investigación definiendo las tecnologías, modelo de desarrollo y herramientas necesarias para el desarrollo del sistema. De acuerdo al estudio realizado y teniendo en cuenta las necesidades del sistema se empleará Java como lenguaje de desarrollo apoyado en la herramienta Netbeans en su versión 7.2 debido a que posee buena integración con el contenedor de aplicaciones GlassFish Open Source Edition 3.1.2. Se selecciona el Programa de Mejora como Modelo de desarrollo y se empleará el Visual Paradigm en su versión 8.0 como herramienta CASE, utilizando el lenguaje de modelado UML 2.0 para la generación de diagramas y como sistema gestor de base de datos se empleará el Postgres SQL en su versión 9.3.

Capítulo 2: Descripción y Análisis de la solución propuesta.

Introducción

En el presente capítulo se aborda la solución técnica propuesta para el sistema. Se describen los requisitos funcionales y no funcionales. Se analiza la arquitectura del sistema desarrollado, las principales capas que la componen y sus componentes. Se describen los patrones de diseño a utilizar. Se muestran los diagramas de componentes, de paquetes y de clases, el modelo de datos del módulo y el diagrama de despliegue. Se muestran las técnicas de validación de requisitos así como el resultado de aplicar las métricas para la validación del análisis y el diseño.

2.1. Propuesta del sistema

La presente investigación tiene como objetivo informatizar los procesos asociados a la oficina de Trámites Migratorios de la Cámara de Comercio de la República de Cuba. Actualmente cuenta con los procesos Registro y Actualización de datos, elaborados para el trabajo con los funcionarios asociados a la cámara y para los trámites migratorios que en ella se realizan. Datos personales, identidad, vínculo y observaciones por parte de los funcionarios y tipo de pasaporte, número de acta, número de clave, fecha de solicitud entre otros por parte de los trámites es la información que va a ser gestionada por estos dos procesos.

La propuesta de solución de este trabajo radica en el análisis, diseño, implementación y pruebas de software del Módulo de Registro de Trámites Migratorios. Dicha solución proveerá las funcionalidades necesarias para que exista un correcto funcionamiento de las tareas relacionadas con el registro y control de los trámites asociados a los funcionarios, la cual contribuirá a la gestión de la información de la Cámara de Comercio de la República de Cuba.

2.2. Antecedentes de la investigación

El desarrollo del proyecto estuvo guiado por las etapas planteadas en el modelo de desarrollo aplicado al proyecto SIRECC. En su primera fase se realizó un estudio Preliminar donde se llevaron a cabo actividades relacionadas con el estudio del proceso de Registro de Trámites Migratorios permitiendo obtener información acerca del alcance del proyecto incluyendo la realización de estimaciones de tiempo, costo y esfuerzo.

Una vez terminada esta fase se pasó al Modelado del Negocio, donde se comprendió cómo funciona el proceso de negocio del proceso que se desea informatizar. Fueron generados diferentes entregables como Reglas de Negocio, Modelo de Procesos de Negocio, Glosario de Términos y Minutas de reunión que fueron necesarias realizar con la Cámara de Comercio.

Posteriormente se pasó a la fase de Requisitos, donde fue realizado el levantamiento de requisitos, obteniéndose 40 requisitos funcionales y 33 no funcionales validándose con el cliente la propuesta a partir de los prototipos confeccionados. El Plan de Pruebas es generado para establecer las actividades a realizar por el equipo de desarrollo para comprobar la calidad del producto. El Plan de Desarrollo de Software es donde se define el plan para la ejecución del proyecto. La Especificación de Requisitos de Software detalla todo lo referente a los requisitos funcionales. Estos y otros son artefactos generados para desarrollar un modelo correcto del sistema que se desea construir.

Fueron identificados dos procesos: el registro de funcionarios y el registro de trámites migratorios donde se realiza la tramitación de visados comerciales, cambio de clasificación migratoria, cambio de dirección, prórroga de residencia temporal, solicitud de visados múltiples, prórroga de visado, actualización de permiso de trabajo, renovación de permiso de trabajo, solicitud duplicado de permiso de trabajo y solicitudes de permiso de trabajo. Además permite generar reportes con la información obtenida de cada trámite realizado. Estos artefactos fueron realizados de acuerdo a la información obtenida al aplicar las técnicas de obtención de requisitos mediante entrevistas al personal de la Cámara de Comercio de la República de Cuba. A continuación se muestra un listado de algunos de los requisitos levantados, el resto puede ser consultado en el documento de Requisitos Funcionales del Módulo Registro de Trámites Migratorios.

2.2.1. Requisitos funcionales

RF2. Registrar residencia temporal

RF5. Registrar solicitante

RF6. Modificar solicitante

RF7. Visualizar solicitante

RF11. Registrar trámite para solicitud de entrada de extranjeros

RF12. Modificar trámite para solicitud de entrada de extranjeros

RF19. Registrar trámite para solicitud de permiso de trabajo

RF20. Modificar trámite para solicitud de permiso de trabajo

RF30. Registrar trámite de cambio de clasificación migratoria

RF31. Modificar trámite de cambio de clasificación migratoria

RF32. Realizar el registro migratorio

RF33. Modificar el registro migratorio

2.2.2. Requisitos no funcionales

Usabilidad

RnF7. El tiempo de respuesta brindado por el sistema será menor que 3 segundos. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas en función de los requisitos de hardware necesarios.

Restricciones del diseño

RnF15. Para el montaje del sistema se requerirá del sistema gestor de bases de datos PostgreSQL 9.3 y del servidor de aplicaciones Glassfish 3.1.2.

Requisitos de Licencia

RnF22. 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.

Seguridad

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

RnF26. El sistema brindará la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado.

RnF27. El sistema mostrará las funcionalidades de acuerdo a quién esté autenticado en el mismo.

RnF28. El sistema debe asegurar el almacenamiento de las credenciales de los usuarios utilizando algoritmos criptográficos que oculten la identidad verdadera de los usuarios.

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

Requisitos de Software

RnF30. Instalar en las estaciones de trabajo el software necesario para el correcto funcionamiento del sistema como:

- Visor de documentos PDF.
- Máquina virtual de java versión 1.6 actualización 26.

RnF31. 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 PostgreSQL versión 9.3.

Requisitos de Hardware

RnF32. 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.

RNF33. Para los servidores se deben proporcionar las características mínimas de hardware siguientes:

- 4 GB de RAM.
- 1TB de disco duro para el caso del servidor de Bases de Datos y 500GB para el de aplicaciones.
- Adaptador de red Ethernet 100 Mbps.
- Sistema de Energía Ininterrumpida (UPS) 500 Va.

Para conocer el resto de los requisitos no funcionales, consultar el documento de Requisitos no funcionales del Proyecto SIRECC.

Como resultado de realizar las tres fases anteriormente explicadas, se obtuvo una serie de artefactos, constituyendo las salidas de las fases anteriormente mencionadas y se convirtieron en entradas a la fase de Análisis y Diseño, la cual se explica a continuación.

2.3. Análisis y Diseño del sistema

El modelo de diseño es una abstracción del código fuente y del modelo de implementación empleado para representar y documentar el diseño. Permite describir el posible resultado físico de los componentes mediante los requisitos funcionales y las principales restricciones. Con el diseño se crea una entrada para la actividad de implementación futura ya que se capturan los conceptos de las clases, interfaces y sistemas que serán necesarios para la solución. Está compuesto por varios artefactos como: el diagrama de paquetes, diagramas de clases del diseño y el modelo de datos.

2.3.1. Diagrama de paquetes

Muestran la organización de los paquetes, perfiles y espacios de nombres. (Altova, 2013). Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. Los paquetes pueden estar anidados unos dentro de otros, y unos paquetes pueden depender de otros paquetes. Se pueden utilizar para plantear la arquitectura del sistema a nivel macro. El diagrama muestra como está estructurado el sistema, cada paquete puede contener otros paquetes o clases, que tienen interfaces y realizan cierta funcionalidad. (Gutierrez, 2009). A continuación se muestra el diagrama de paquetes del Módulo de Registro de Trámites Migratorios.

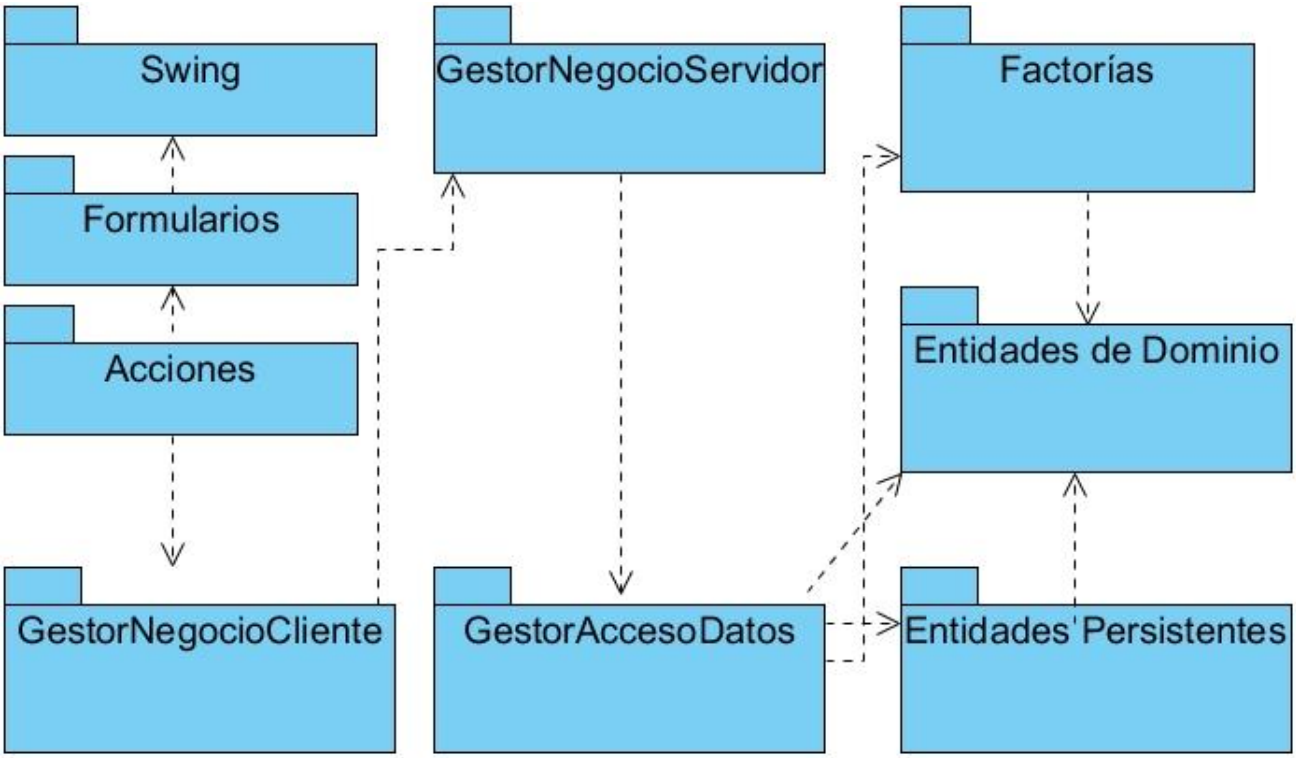


Imagen 1: Diagrama de paquetes del Módulo Registro de Trámites Migratorios**.**

2.3.2. Diagrama de Clases del Diseño

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Se ofrece para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso. (Egdamar, 2012). Para realizar este diagrama se tomó como muestra el requisito Registrar Trámite Entrada Extranjero. El diagrama del sistema completo lo puede observar en el **Anexo 3.**

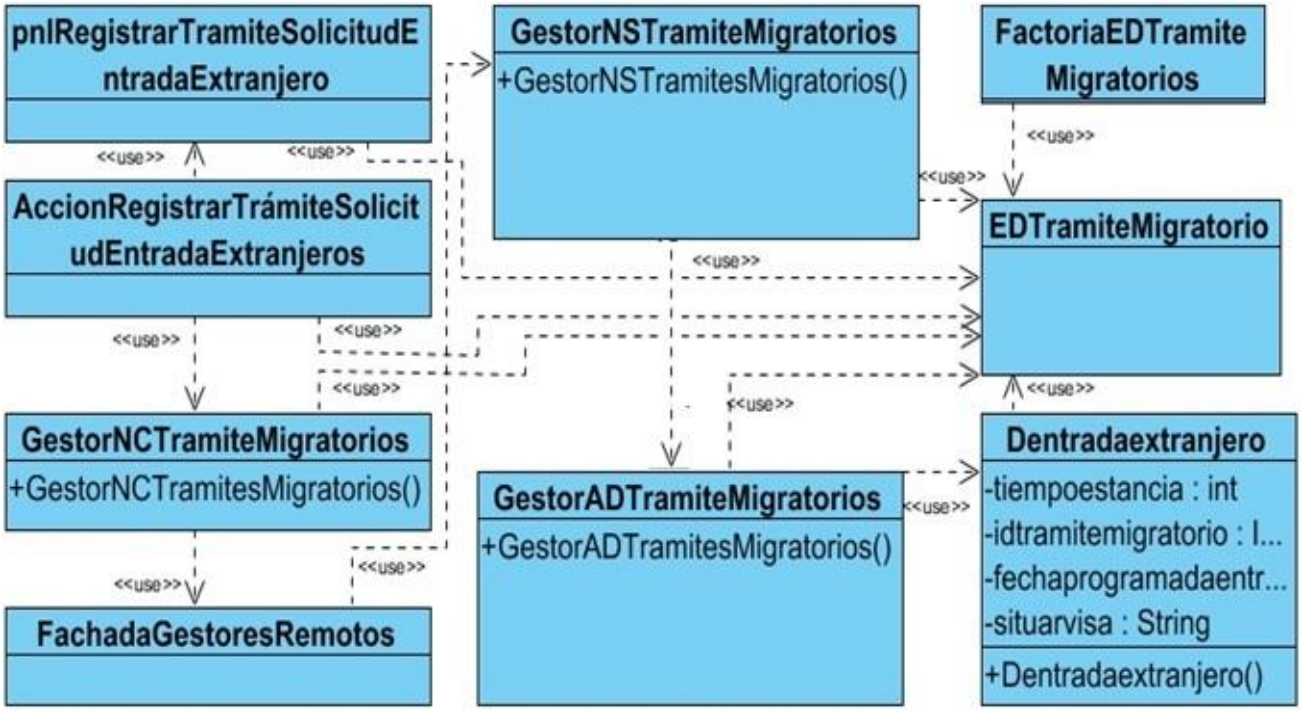


Imagen 2: Diagrama de Clases del Diseño del requisito Registrar Trámite Solicitud de Entrada Extranjero.

2.3.3. Modelo de datos

Un modelo de base de datos es la fundamentación teórica de una base de datos y determina de qué manera los datos van a ser guardados, organizados y manipulados en un sistema de base de datos. De esta forma, define la infraestructura ofrecida por un sistema de base de datos particular. (Fergarciac , 2013)

El modelo correspondiente a los subsistemas de Trámites Migratorios está compuesto por 28 tablas, de ellas 9 nomencladores y el resto para guardar la información correspondiente al sistema a desarrollar.

Para identificar las tablas fueron definidas letras, para el caso de las tablas nomencladoras se utilizó la letra n. Ejemplo: nTramiteMigratorio, para las tablas de datos se utilizó la letra d. Ejemplo: dPersonaTramiteMigratorio y para el caso de las llaves primarias se utilizó la siguiente denotación Id+ nombre de la tabla, ejemplo: IdTramiteMigratorio. A continuación se muestra un fragmento del modelo de datos del Módulo Trámites Migratorios. El modelo de datos del sistema se encuentra más detallado en el **Anexo 4.**

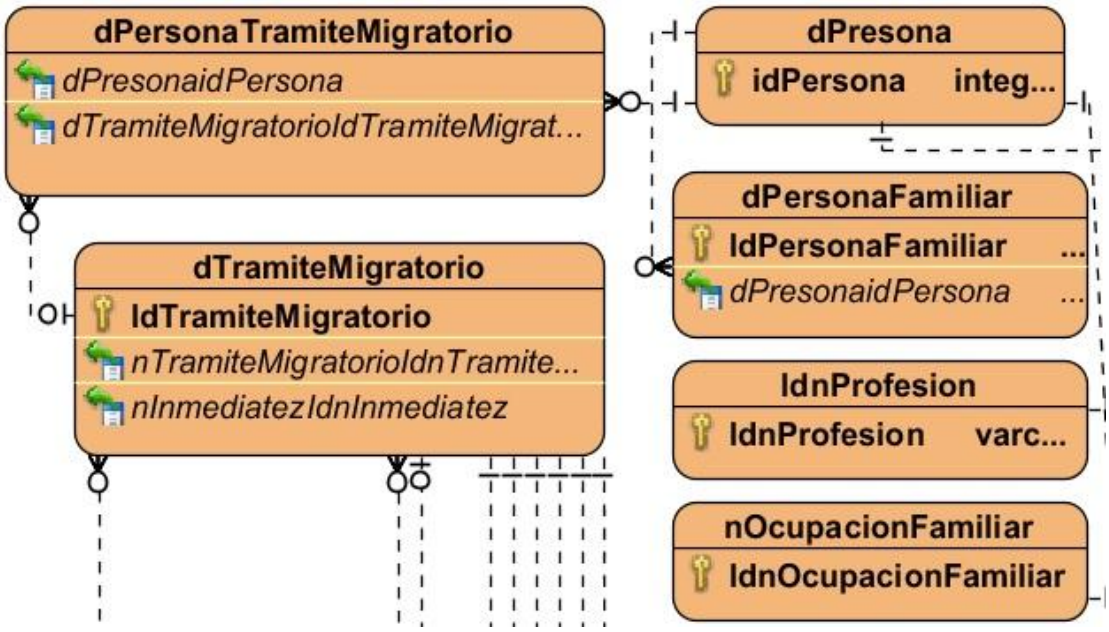


Imagen 3: Fragmento del Modelo de datos del Módulo Trámites Migratorios.

2.4. Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (Ithleovi, 2010)

2.4.1. Diagrama de componentes

El diagrama de componentes ilustra los componentes del software que serán usados para construir el sistema. Estos pueden ser construidos para el modelo de clase y escritos para satisfacer los requisitos del nuevo sistema. (Monografias.com, 2013). A continuación se muestra el diagrama de paquetes del requisito Registrar Trámite Solicitud Entrada Extranjero.

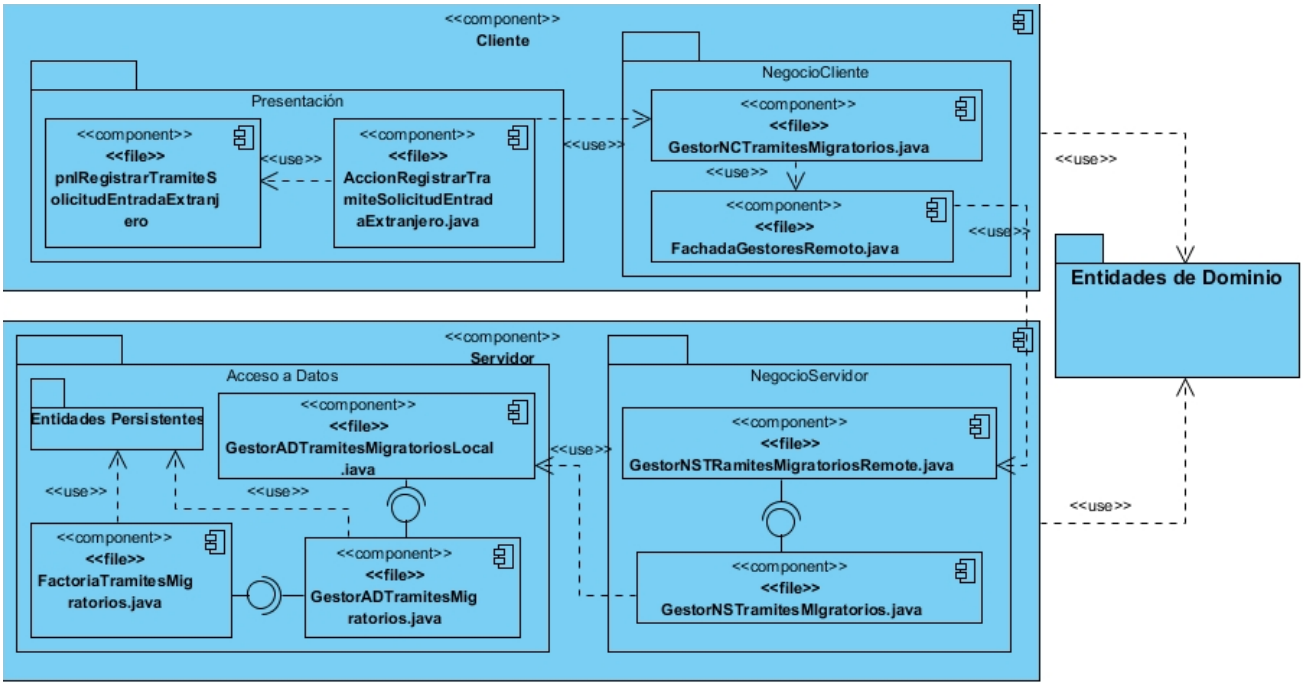


Imagen 4: Diagrama de componentes del requisito Registrar Trámite Solicitud Entrada Extranjero**.**

2.4.2. Diagrama de despliegue

El diagrama de despliegue describe cómo una aplicación se despliega a través de una infraestructura. La intención del modelo de despliegue no es para describir la infraestructura, muestra el camino donde los componentes específicos deben corresponder a una aplicación que despliega a través de él. (Monografias.com, 2013)

La distribución física del sistema estará compuesta por una PC-cliente en la cual estará instalada la máquina virtual de Java donde correrá la aplicación en su lado cliente. A la PC-cliente se le conectará por Universal Serial Bus (USB) 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, donde se encontrará desplegada la aplicación empresarial que se encargará de gestionar las respuestas a las peticiones del cliente usando modelo de descripción de protocolos de red (TCP-IP). La Imagen 5 muestra el modelo de despliegue propuesto para el subsistema Trámites Migratorios.

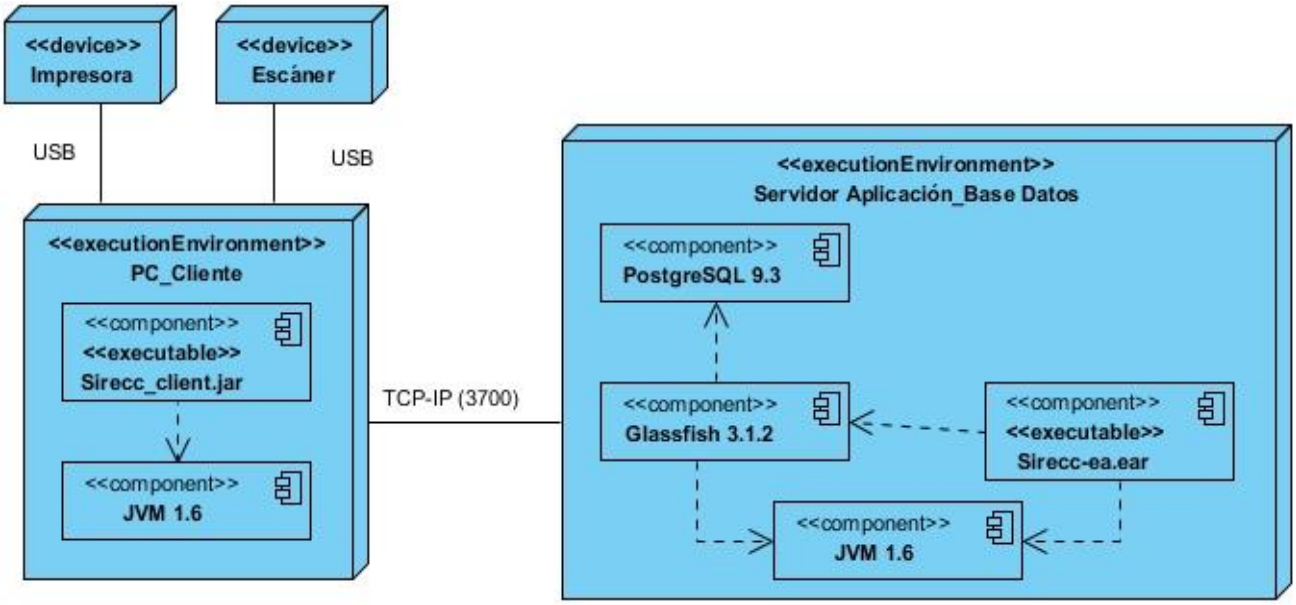


Imagen 5: Diagrama de despliegue del Módulo Trámites Migratorios**.**

2.4.3. Estilos arquitectónicos

El diseño arquitectónico se enfocó en el estilo cliente-servidor con el objetivo de proporcionar un nivel de mantenibilidad adecuado a la solución. Otro estilo a utilizar es de n-capas para organizar mejor el diseño teniendo en cuenta que permite establecer una organización jerárquica entre cada una de las capas, garantizando que cada capa proporcione servicios a la capa inmediatamente superior, específicamente se identificaron cuatro capas: Presentación, Negocio del Cliente, Negocio del Servidor y Acceso a Datos. Además el diseño arquitectónico se orientó a la reutilización de componentes, con el fin de reducir el ciclo de desarrollo y proporcionar un diseño más escalable y refinado.

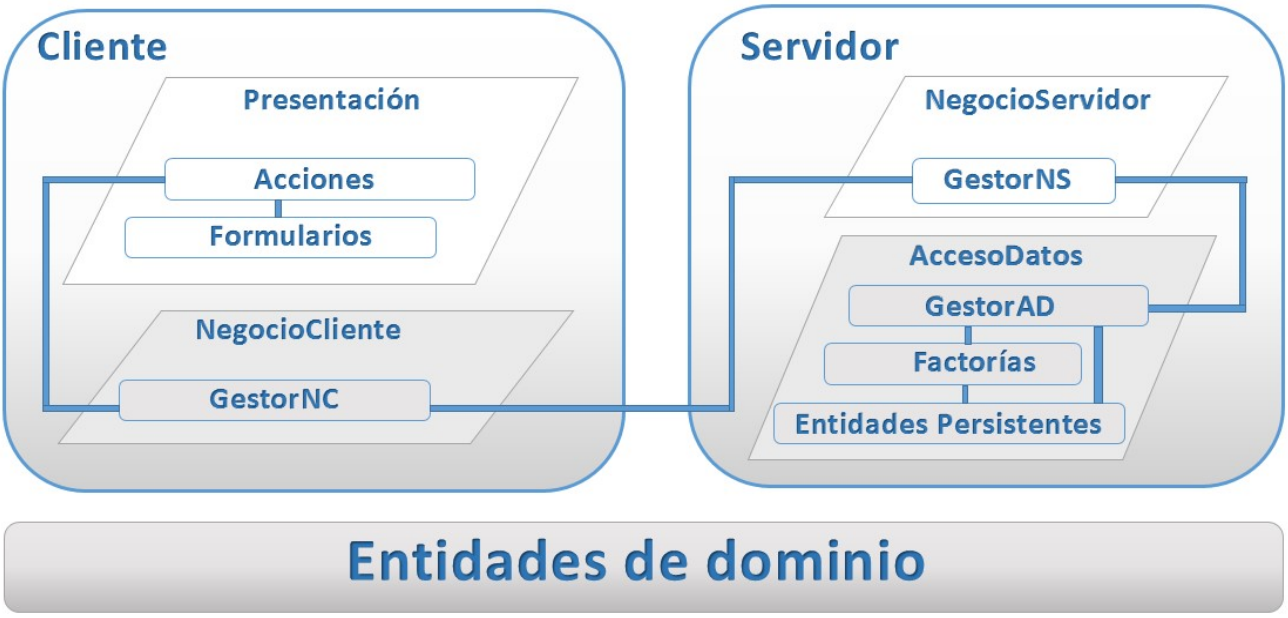


Imagen 6: Arquitectura del sistema.

2.4.4. Patrones de diseño

Patrón Experto: este patrón es utilizado en las entidades de dominio, estas clases contienen toda la información referente a los Trámites Migratorios. A continuación se muestra una parte de la información por la que están compuestas estas clases.

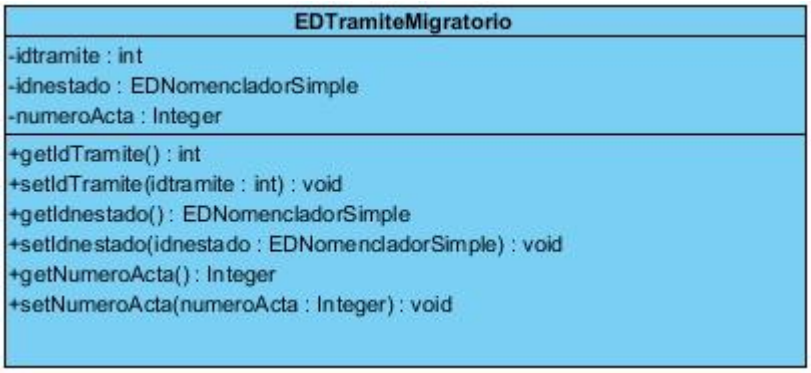


Imagen 7: Aplicación del patrón Experto.

Patrón Instancia única: este patrón se evidencia con la clase FachadaGestoresRemotos () que permite abstraerse de la creación de instancias, no permite que se creen más instancias de una misma clase.

```
static public FachadaGestoresRemotos getInstancia() throws Exception {
    if (instancia == null) {
        instancia = new FachadaGestoresRemotos();
    }
    return instancia;
}
```

Imagen 8: Aplicación del patrón Instancia única.

Patrones Alta cohesión y Bajo acoplamiento: Estos patrones se evidencian en las responsabilidades que le son asignadas a las acciones; ejemplo, en el requisito Registrar Trámite Solicitud Entrada Extranjero. El patrón alta cohesión es el encargado de asignar las responsabilidades de manera que se generen solo las dependencias necesarias en las clases. El patrón bajo acoplamiento proporciona las dependencias mínimas entre las diferentes clases, se puede observar cómo la acción se comunica directamente con el gestor de negocio del cliente, este con el gestor de negocio del servidor y éste a su vez con el gestor de acceso a datos. De esta manera a la hora de realizar modificaciones necesarias, no implica grandes cambios en el negocio, manteniendo un bajo acoplamiento se logra que las distintas "piezas" del software funcionen sin depender demasiado unas de otras y posibilita una mayor reutilización de cada una de las clases.

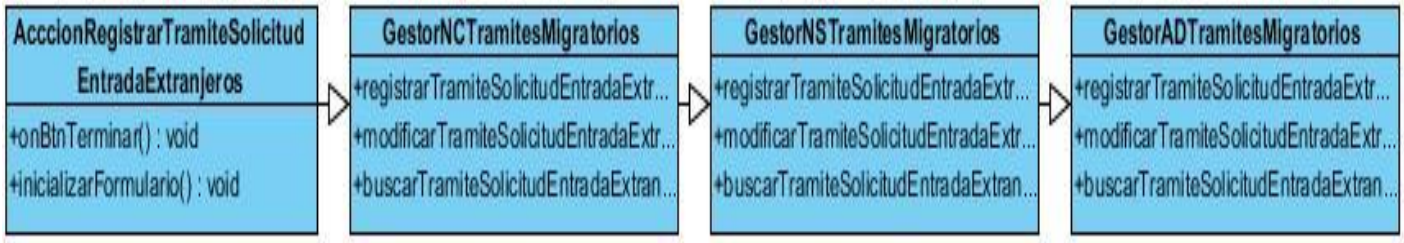


Imagen 9: Aplicación de los patrones alta cohesión y bajo acoplamiento.

Inyección de dependencias: Enterprise JavaBeans (EJB) implementa este patrón para declarar objetos de otras clases. Posibilita inyectar objetos en una clase, en lugar de ser la propia clase quien cree el objeto. En este caso se muestra un fragmento de la clase FactoriaEDTramitesMigratorios () donde se puede visualizar inyecciones para instanciar objetos de sesión bean (son EJB's), utilizados en la ejecución de distintas funcionalidades.

```
@EJB
private FactoriaEDSolicitanteLocal factoriaSolicitante;
@EJB
private FactoriaEDPersonaLocal factoriaPersona;
@EJB
private GestorADDireccionLocal gestorDireccion;
@EJB
private factoriaEDDireccionLocal factoriaDireccion;
@EJB
private FactoriaEDEstructuraLocal factoriaEstrucura;
@EJB
private FactoriaEDPasaporteLocal factoriaPasaporte;
```

Imagen 10: Aplicación del patrón Inyección de dependencias.

Patrón Controlador: Este patrón se evidencia en la capa Presentación, compuesta por los formularios y las acciones, esta última utiliza peticiones de los usuarios, contiene varios objetos de otras clases los cuales van a permitir comunicarse con la capa gestor de negocio del cliente, el cual espera la petición de las acciones para poder responderle.

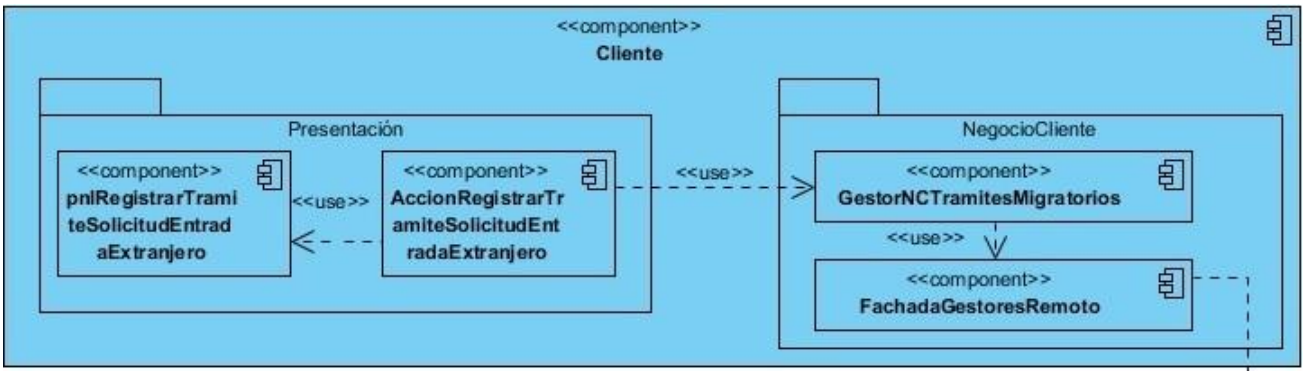


Imagen 11: Aplicación del patrón Controlador.

Patrón fachada: Este patrón es aplicado en la clase FachadaGestoresRemotos pues esta es el punto de acceso común entre el GestorNC (cliente) y el GestorNS (servidor).

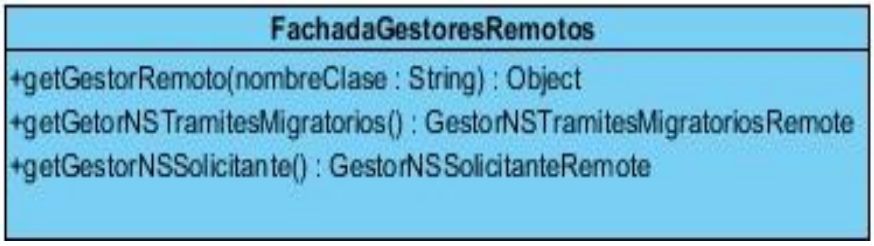


Imagen 12: Aplicación del patrón Fachada.

Patrón Creador: Este patrón se evidencia en las clases Factorías. Tienen la responsabilidad de asignar a determinadas clases la responsabilidad de crear una instancia de otra clase. En este caso se muestra la FactoriaEDTramitesMigratorios, encargada de convertir entidades de dominio en entidades persistentes y viceversa.

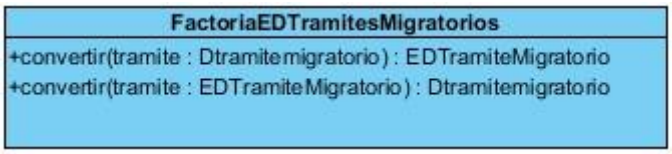


Imagen 13: Aplicación del patrón Controlador

2.4.5. Estándares de codificación

Los estándares de codificación están enmarcados en aspectos que definen la generación de código. Ayudan a que los programadores apliquen en la implementación un estilo correcto y práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Además, si se aplica de forma continua un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y, posteriormente, se

efectúan revisiones del código de rutinas, existen posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener. (Sheyla García, 2013) A continuación se muestran algunos de los estándares definidos. Consultar el documento de Estándares de codificación del proyecto SIRECC.

Organización de los ficheros: Los ficheros fuentes Java tienen la siguiente ordenación.

Comentarios de comienzo: Todos los ficheros fuentes deben comenzar con un comentario en el que se liste el nombre de la clase, información de la versión y fecha.

Sentencias package e import: La primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import.

Declaraciones de clases e interfaces: La siguiente lista describe las partes de la declaración de una clase o interface, en el orden en que deberían aparecer:

- Comentario de documentación de la clase o interface (/**...*/).
- Sentencia class o interface.
- Comentario de implementación de la clase o interface si fuera necesario (/*...*/): Este comentario debe contener cualquier información aplicable a toda la clase o interface que no era apropiada para estar en los comentarios de documentación de la clase o interface.
- Variables de clase (static): Primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
- Variables de instancia: Primero las public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
- Constructores.
- Métodos: Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.

Identación: Se deben emplear cuatro espacios como unidad de identación. La construcción exacta de la identación (espacios en blanco contra tabuladores) no se especifica. Dentro de este están la Longitud de la línea que se usan para evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas. Además se encuentra Rompiendo líneas que son utilizadas para cuando una expresión no entre en una línea, romperla de acuerdo con determinados principios.

Sentencias: Existen varios tipos de sentencias como: las sentencias simples, las sentencias compuestas, las sentencias de retorno, sentencias if, if-else, if else-if else, sentencias for, sentencias while, sentencias do-while, sentencias switch, sentencias try-catch.

2.4.6. Seguridad del Módulo Registro de Trámites Migratorios

El marco de trabajo utilizado en la presente investigación propone dos mecanismos de seguridad: autenticación y autorización, ellos permiten el aseguramiento de la información que manejen los sistemas que utilicen Kairos. Para lograr esto se realizó la implementación de un dominio de seguridad que permiten al servidor interactuar con distintos sistemas de almacenamiento de identidad para llevar a cabo la autenticación de los usuarios.

Para llevar a cabo el proceso de autenticación desde una aplicación cliente contra el servidor GlassFish Open Source se hace uso de una clase que contiene la lógica encargada de publicar la identidad del cliente para su posterior acceso.

Para verificar que quien intente invocar una funcionalidad determinada en el servidor cuenta con los permisos necesarios para hacerlo se realiza el proceso de autorización. El mismo es gestionado por el servidor de aplicaciones y está basado en el uso de roles. Un rol puede ser visto como un privilegio que posee un cliente para invocar cierta funcionalidad o acceder a cierto recurso en un ambiente asegurado.

Para mayor seguridad de la contraseña de los usuarios se propone para la encriptación de las mismas las funciones de resumen md5 y sha-1, las cuales son sólo de ida o irreversibles. Gracias a esto no es posible revertir la encriptación para obtener la contraseña a partir de su valor encriptado. (Daniel Sarmiento Sastriques, 2012)

2.5. Pruebas Internas

A continuación se explican las pruebas internas llevadas a cabo en el desarrollo del software y los resultados obtenidos. Se aplicaron pruebas unitarias y dentro de estas, pruebas estáticas, donde se validó la especificación de requisitos aplicándose una serie de revisiones, listas de verificación, construcción de prototipos y generación de casos de pruebas. Se validó el diseño empleando métricas y por último se realizaron pruebas de codificación cómo las de caja blanca y caja negra.

2.5.1. Validación de los requisitos.

En la fase Requisitos se obtienen los requisitos y se establece el fundamento para el diseño. Las métricas son confiables pues proporcionan una visión interna a la calidad del modelo de análisis. Se propone una lista de características con las cuales puede evaluarse la correspondiente especificación de requisitos: especificidad (falta de ambigüedad), facilidad y comprensión,

corrección, facilidad para modificarse, entre otras en cuanto a la calidad de la especificación. (Pressman, 2007)

Durante el proceso de validación de requisitos se deben llevar a cabo verificaciones sobre el documento de requisitos. Comprenden verificaciones de validez, de consistencia, de completitud, de realismo y verificabilidad.

Pueden utilizarse, en conjunto o de forma individual, varias técnicas de validación de requisitos tales como:

- Revisiones de requisitos: Son analizados sistemáticamente por un equipo de revisores. Es un proceso que involucra a personas tanto de la organización del cliente como de la del contratista.
- Construcción de prototipos: Se muestra un modelo ejecutable del sistema a los usuarios finales y a los clientes, con el objetivo de experimentar con este modelo, para ver si cumple sus necesidades reales. Este enfoque se denomina a veces prototipado desechable, debido a que el prototipo no es estregado al cliente o mantenido por el desarrollador. En el Diseño son utilizados para apoyar el diseño de las interfaces.
- Generación de casos de prueba: Los requisitos deben poder probarse. Si las pruebas para estos se conciben como parte del proceso de validación, a menudo revela los problemas en los requisitos. (Sommerville, 2005)

La validación de requisitos trata de mostrar que éstos realmente definen el sistema que el cliente desea. Coincide parcialmente con el análisis ya que éste implica problemas con los requerimientos, pues permite en caso de ser necesario que los requisitos puedan ser refinados y estructurados para conseguir una comprensión más precisa y una descripción que sea fácil de mantener.

Las técnicas de validación de requisitos utilizadas fueron la construcción de prototipos ayudando en la fase de Requisitos a la validación de los requisitos del Módulo del Registro de Trámites Migratorios. Se utilizaron en la fase del Diseño para apoyar el desarrollo del diseño de las interfaces del sistema en cuestión.

Otra técnica utilizada fue la revisión de requisitos para asegurar que los requisitos levantados son lo que realmente desea el cliente. Son analizados sistemáticamente por un equipo de revisores. Es un proceso que involucra a personas tanto de la Cámara de Comercio como del personal que labora en la confección del módulo.

Otra técnica fue la generación de casos de prueba, elaborados a través de los requisitos levantados para probarlos de acuerdo a lo desarrollado en el módulo, revelando algunos problemas existentes

en los requisitos. Los casos de pruebas contienen la descripción de todos los elementos de cada requisito a implementar, así como las observaciones de cada uno.

2.5.2. Validación del diseño

Con el objetivo de mantener un adecuado aseguramiento y control de la calidad de los artefactos generados en el diseño se utilizan las métricas de software. A continuación se valida el diseño del módulo mediante las métricas, con las que se puede medir el estado de algunos atributos de calidad que permiten brindar criterios valorativos sobre la realización del diseño.

Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo.

En el libro de métricas realizado por Lorenz y Kidd, dividen las métricas basadas en clases. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad.

El Tamaño Operacional de la Clase (TOC), es una métrica que está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- **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.

El tamaño general de una clase se puede determinar empleando las medidas siguientes:

- El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
- El número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.

Para mejor comprensión de los cálculos para evaluar los atributos de calidad consultar el **Anexo 1**.

Si existen valores grandes de TOC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá el grado de reutilización de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Otra de las métricas estudiadas es Relaciones entre Clases (RC), está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

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

Dentro de las métricas de diseño a nivel de componentes se utilizó:

Tamaño Operacional de Clase (TOC):

Complejidad de implementación y responsabilidades:

Promedio Número de Atributos (P (NA)).

Promedio Número de Operaciones (P (NO)).

(P (NA)): 4

Baja: X < 4**, Media:** 4 < X < 8**, Alta:** X > **8**

(P (NO)): 23

Baja: X < 23, **Media:** 23 < X < 46**, Alta:** X > 46

#	Nombre de la clase	NA	NO	Tamaño (NA)	Tamaño (NO)
1	GestorADTramitesMigratorios	3	43	Baja	Media
2	GestorADSolicitante	7	15	Media	Baja
3	GestorNCTramitesMigratorios	0	43	Baja	Media
4	GestorNCSolicitante	0	15	Baja	Baja
5	GestorNSTramitesMigratorios	1	43	Baja	Media
6	GestorNSSolicitante	1	15	Baja	Baja
7	AccionSeleccionarTramite	5	13	Media	Baja
8	AccionBuscarTramite	8	19	Media	Baja
9	AccionRegistrarTramiteSolicitudEntradaExtranjeros	6	12	Media	Baja

10	AccionRegistrarTramiteSolicitudPermisoTrabajo	9	12	Alta	Baja

Tabla 4: Resultados obtenidos mediante la métrica TOC para la complejidad de implementación y las responsabilidades.

Al analizar esta tabla se obtuvo como resultado que para el caso de P (NA) tanto para la complejidad de implementación y las responsabilidades es un 50% Baja, un 40% Media y un 10% Alta. Para el caso de P (NO) un 70% Baja, un 30% Media y un 0% Alta.

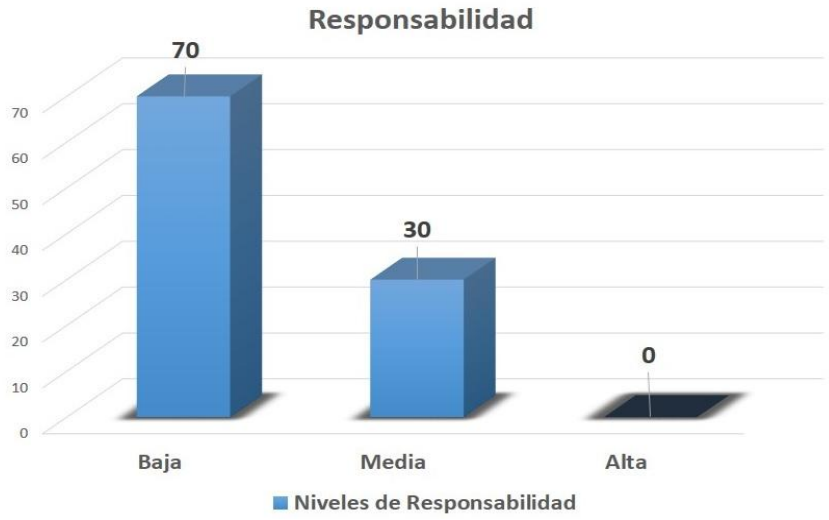


Imagen 14: Resultados obtenidos de la Responsabilidad**.**

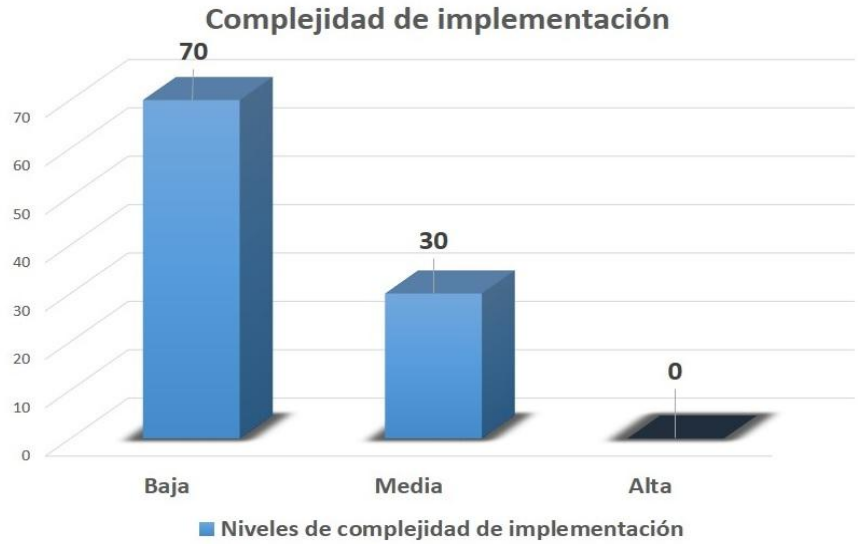


Imagen 15: Resultados obtenidos de la Complejidad de implementación.

Como se puede observar, el análisis mostrado arrojó como resultado que las clases poseen baja complejidad de implementación y de responsabilidad representada por un 70%.

Reutilización:

Promedio (P (NA)): 4

Baja: X > 8**, Media:** 4 < X < 8**, Alta:** X < 4

Promedio (P (NO)): 23

Baja: X > 46, **Media:** 23 < X < 46**, Alta:** X < 23

#	Nombre de la clase	NA	NO	Tamaño (NA)	Tamaño (NO)
1	GestorADTramitesMigratorios	3	43	Alta	Media
2	GestorADSolicitante	7	15	Media	Alta
3	GestorNCTramitesMigratorios	0	43	Alta	Media
4	GestorNCSolicitante	0	15	Alta	Alta
5	GestorNSTramitesMigratorios	1	43	Alta	Media
6	GestorNSSolicitante	1	15	Alta	Alta
7	AccionSeleccionarTramite	5	13	Media	Alta
8	AccionBuscarTramite	8	19	Media	Alta
9	AccionRegistrarTramiteSolicitudEntradaExtranjeros	6	12	Media	Alta
10	AccionRegistrarTramiteSolicitudPermisoTrabajo	9	12	Baja	Alta

Tabla 5: Resultados obtenidos mediante la métrica TOC para la reutilización.

De acuerdo a los resultados obtenidos en la Tabla 5 la P (NA) de manera general es un 10% Baja, un 40% Media y un 50% Alta. Para el caso de P (NO) un 0% Baja, un 30% Media y un 70% Alta.

Imagen 16: Resultados obtenidos de la Reutilización.

Como se puede observar se obtuvo como resultado que las clases poseen una alta reutilización.

Otra métrica utilizada fue Relaciones entre Clases (RC):

#	Nombre de la clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
1	GestorADTramitesMigratorios	2	Medio	Baja	Alta	Baja
2	GestorADSolicitante	1	Bajo	Baja	Alta	Baja
3	GestorNCTramitesMigratorios	2	Medio	Baja	Alta	Baja
4	GestorNCSolicitante	4	Alto	Media	Media	Media
5	GestorNSTramitesMigratorios	4	Alto	Media	Media	Media
6	GestorNSSolicitante	1	Bajo	Baja	Alta	Baja
7	AccionSeleccionarTramite	3	Alto	Media	Media	Media
8	AccionBuscarTramite	1	Bajo	Baja	Alta	Baja

9	AccionRegistrarTramiteSolicitudEntradaExtranjeros	1	Bajo	Baja	Alta	Baja
10	AccionRegistrarTramiteSolicitudPermisoTrabajo	1	Bajo	Baja	Alta	Baja

Tabla 6: Resultados obtenidos mediante la métrica RC.

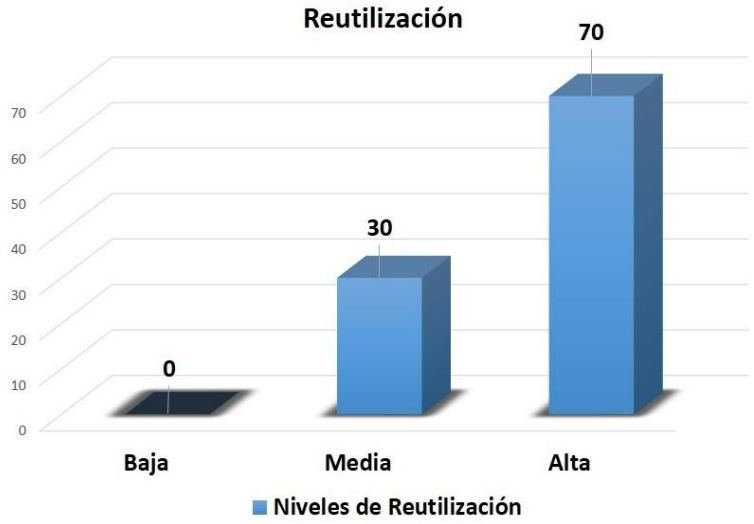


Imagen 17: Resultados obtenidos de la Reutilización.

El resultado obtenido en este atributo muestra un alto nivel de reutilización de las clases del módulo, pues de todos los valores alcanzados es el más alto con un 70 % del total.

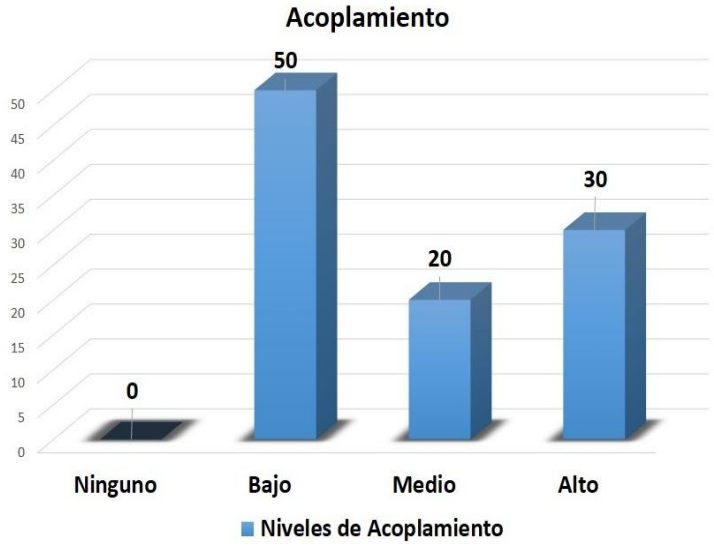


Imagen 18: Resultados obtenidos del Acoplamiento.

El atributo acoplamiento mostró como resultado un 0% ninguno, 50% bajo, 20% medio y un 30% alto. Alcanzando resultados positivos en cuanto a la existencia de un bajo acoplamiento entre las clases del módulo, pues es el mayor resultado de los valores en los niveles de acoplamiento.

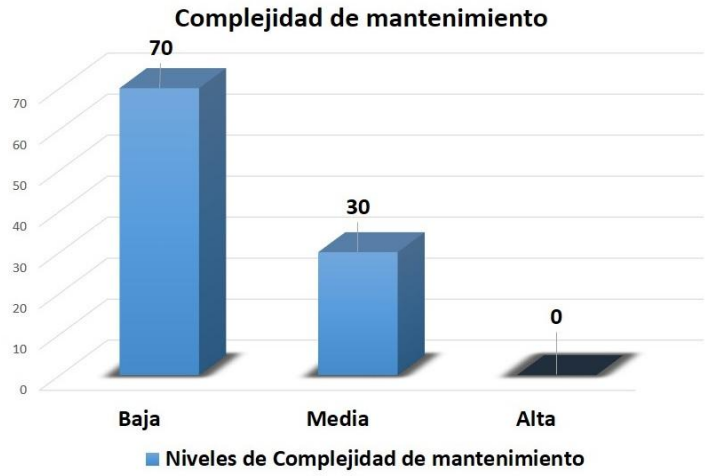


Imagen 19: Resultado obtenido en la complejidad de mantenimiento.

La complejidad de mantenimiento cuenta con un 70% baja, un 30% media y un 0% alta. El módulo no tiene dificultad en cuanto a la complejidad de mantenimiento, pues no es un sistema complejo.

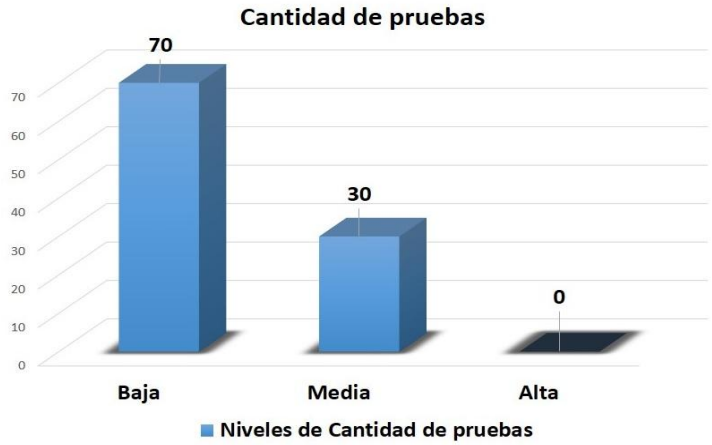


Imagen 20: Resultado obtenido de la cantidad de pruebas.

Por último para el atributo de calidad cantidad de pruebas arrojó como resultado un 70% baja, un 30% media y un 0% alta. El nivel de cantidad de pruebas es bajo.

Otra métrica utilizada fue Profundidad en árbol de herencia.

Esta métrica por su nombre y siglas en inglés (Depth of Inheritance Tree –DIT-) mide el máximo nivel en la jerarquía de herencia. Chidamber y Kemerer proponen esta métrica como medida de la

complejidad de una clase, complejidad del diseño y el potencial de reutilización. Esto es debido a que cuanto más profunda se encuentra una clase en la jerarquía mayor será la posibilidad de heredar un mayor número de métodos. Según Pressman es la longitud máxima desde el nodo hasta la raíz del árbol. Lorenz y Kidd sugieren un umbral de 6 niveles como indicador de un abuso en la herencia. Como resultado de aplicar esta métrica, mostrada en la Imagen 21, se obtuvo 5 niveles, valor que está por debajo del umbral planteado por Lorenz y Kidd, mostrando un bajo nivel de herencia, provocando que no exista una mayor complejidad en el diseño y en los objetos.

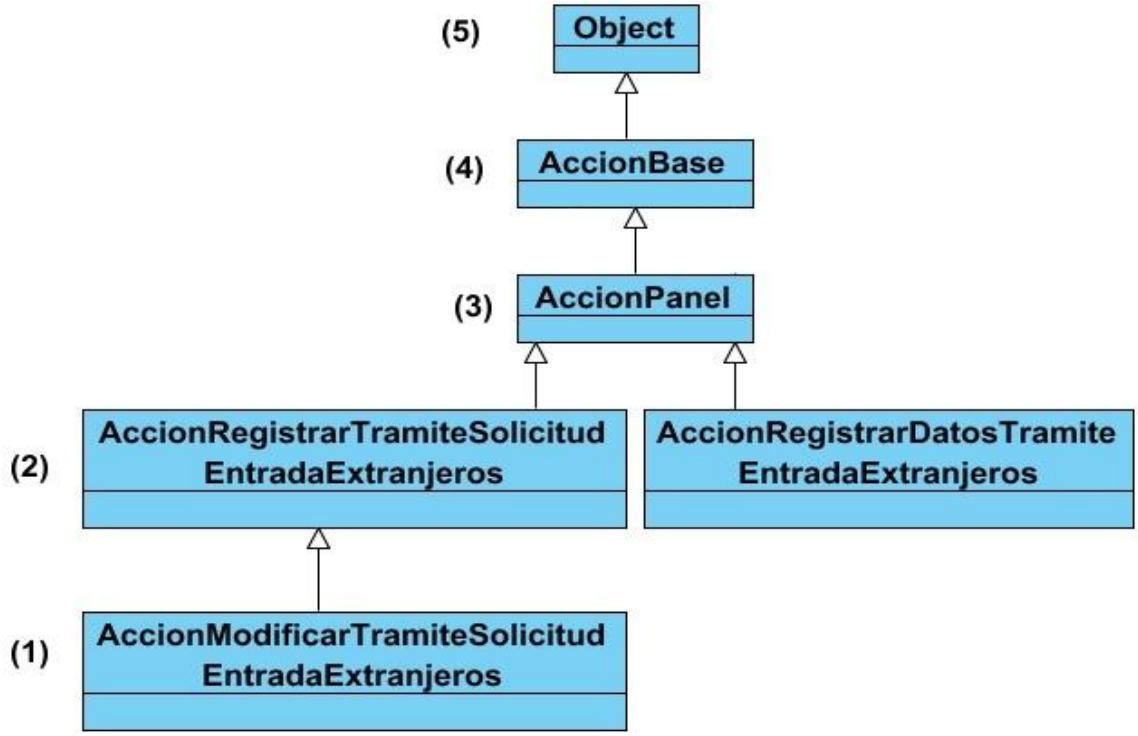


Imagen 21: Niveles del Árbol de profundidad de herencia.

2.5.3. Validación del código fuente

Pruebas de caja blanca

Como parte de la validación de la implementación otros de los métodos de diseño aplicados fue el método de caja blanca con el objetivo de garantizar que se ejercita por lo menos una sola vez los caminos independientes de las funcionalidades de mayor complejidad.

Para obtener el resultado de este método se utilizó la prueba del camino mínimo que es una de las técnicas de prueba de caja blanca. Esta técnica cuenta con 4 pasos fundamentales que son explicados a continuación.

- Seleccionar el método y señalar los posibles nodos a usar.

```
public List<EDTramitePermisoTrabajo> BuscarTramites(Calendar fechaI, Calendar fechaF) {
        List<EDTramitePermisoTrabajo> tramites = new ArrayList<EDTramitePermisoTrabajo>(); (1)
        Query consulta = null; (1)
        String consultaDinamica = STR_QUERY_GENERAL + "WHERE" + STR_QUERY_TIPOTRAMITE_REPORTE; (1)
        if (fechaI != null) { (2)
            consultaDinamica += " AND " + STR_QUERY_FECHAINICIO; (3)
        }
        if (fechaF != null) { (4)
            consultaDinamica += " AND " + STR_QUERY_FECHAFIN; (5)
        }
        consultaDinamica += STR_ORDER_BY; (6)
        consulta = manager.createQuery(consultaDinamica); (6)
        if (fechaI != null) { (7)
            consulta.setParameter("fechaI", fechaI.getTime()); (8)
        }
        if (fechaF != null) { (9)
            consulta.setParameter("fechaF", fechaF.getTime()); (10)
        }
        List<Object[]> resultados = consulta.getResultList(); (11)
        tramites = factoriaTramites.convertirListaReporte(resultados); (11)
        for (int i = 0; i < tramites.size(); i++) { (12)
            String consultaEstado = "SELECT DISTINCT de FROM Destadotramitemigratorio de WHERE de.dtramitem
                    + "AND de.nestadotramitemigratorio.idnestadotramitemigratorio = :idNestado"; (13)
            Query q = manager.createQuery(consultaEstado); (13)
            q.setParameter("idTramite", tramites.get(i).getIdTramite()); (13)
            q.setParameter("idNestado", ENUM_ESTADOS_TRAMITES.APROBADO.getValor()); (13)
            List<Destadotramitemigratorio> list = (List<Destadotramitemigratorio>) q.getResultList(); (13)
            if (!list.isEmpty()) { (14)
                Destadotramitemigratorio e = list.get(0); (15)
                EDNomencladorSimple nestado = new EDNomencladorSimple(e.getNestadotramitemigratorio().getId
                tramites.get(i).setIdnestado(nestado); (15)
            }
        }
        return tramites; (16)
```

Imagen 22: Representación del código y los nodos seleccionados del método BuscarTramitePermisoTrabajo ().

- Determinar la complejidad ciclomática del grafo del flujo resultante.

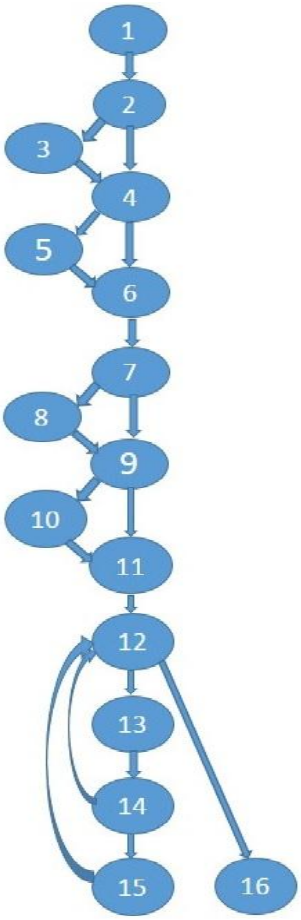


Imagen 23: Grafo del método BuscarTramitePermisoTrabajo ().

Una vez obtenido el grafo correspondiente al método anteriormente mostrado en la Figura 22 se pasa a determinar la complejidad ciclomática del grafo del flujo resultante. Como se había explicado en el Capítulo 1 existen tres maneras de calcularla, a continuación se muestra el resultado obtenido en el cálculo.

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
  Por lo que V (G) = 7.
- V (G) = (A – N) + 2 donde "**A"** es el número de aristas del grafo de flujo y "**N**" es el número de nodos de ese grafo.
  Obteniendo como resultado V (G) = 21 – 16 +2 = **7**
- V (G) =P+ 1 donde "**P**" es el número de nodos predicado contenidos en el grafo. Los nodos 2, 4, 7, 9, 10 y 14 son nodos predicados.
  Por tanto V (G) = 6 + 1 = **7**

Como se puede observar la complejidad ciclomática es 7, comprobándose el mismo resultado en los tres procedimientos por los cuales puede ser obtenida. La cantidad de caminos básicos que puede tomar el algoritmo durante su ejecución es **7** y quedan definidos en la siguiente tabla:

No. camino	Camino
Camino básico #1	1-2-3-4-5-6-7-8-9-10-11-12-16
Camino básico #2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-12-16
Camino básico #3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-12-16
Camino básico #4	1-2-4-6-7-9-10-11-12-16
Camino básico #5	1-2-3-4-5-6-7-8-9-11-12-16
Camino básico #6	1-2-4-5-6-7-9-10-11-12-13-14-12-16
Camino básico #7	1-2-3-4-6-7-8-9-11- 12-13-14-15-12-16

Tabla 7: Resultados de los caminos obtenidos.

Seguidamente se preparan los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

Descripción: este método se encarga de devolver una lista de todos los trámites de permiso de trabajo, dígase de solicitud, de actualización y de renovación de permiso de trabajo que estén registrados entre una fecha inicial y una fecha final.

Caso de prueba para el camino básico # 1: Condición de ejecución: para esta prueba fue necesario que las variables `Calendar fechaI y Calendar fechaF` sean distintas de nulo y que no se hayan registrado trámites de permiso de trabajo en este intervalo de tiempo.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #1 y devuelva una lista vacía.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 2: Condición de ejecución: para esta prueba fue necesario que no sean nulas las variables `Calendar fechaI y Calendar fechaF`, que existan trámites de permiso de trabajo registrados y que ninguno tenga estado aprobado.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #2 y devuelva una lista de trámites cuyo estado sea distinto de aprobado.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 3: Condición de ejecución: para esta prueba fue necesario que las variables `Calendar fechaI y Calendar fechaF` sean distintas de nulo, que existan trámites de permiso de trabajo y su estado sea aprobado.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #3 y devuelva una lista de trámites cuyo estado sea aprobado.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 4: Condición de ejecución: para esta prueba fue necesario que la variable `Calendar fechaI sea` nula, la variable `Calendar fechaF` sea distinta de nula y no tenga trámites de permiso de trabajo registrados antes de la fecha final.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #4 y devuelva una lista de trámites vacía.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 5: Condición de ejecución: para esta prueba fue necesario que la variable `Calendar fechaF` sea nula, la variable `Calendar fechaI` no sea nula y no tenga trámites registrados de permiso de trabajo después de la fecha inicial.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #5 y devuelva una lista de trámites vacía.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 6: Condición de ejecución: para esta prueba fue necesario que la variable `Calendar fechaI sea` nula, la variable `Calendar fechaF` sea distinta de nula y tenga trámites de permiso de trabajo registrados antes de la fecha final.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #6 y devuelva una lista de trámites registrados antes de la fecha final.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 7: Condición de ejecución: para esta prueba fue necesario que la variable `Calendar fechaF` sea nula, la variable `Calendar fechaI` no sea nula y tenga trámites registrados de permiso de trabajo con estado aprobado después de la fecha inicial.

Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino #7 y devuelva una lista de trámites registrados con estado aprobado después de la fecha inicial.

El resultado obtenido fue correcto.

Prueba de Caja Negra

Para la ejecución de este método se realizaron un conjunto de casos de pruebas (CP) para determinar si los requisitos estaban completamente satisfactorios. En total se realizaron 40 CP, probándose cada una de las funcionalidades y evaluándose sus resultados.

Estas pruebas son aplicadas con la intención de descubrir errores que afectan el correcto funcionamiento del software, por lo que es necesario validar que este funcione como fue diseñado y validar que los requisitos fueron implementados correctamente.

Una vez elaborados los casos de pruebas se realizaron comprobaciones al sistema antes de entregar el producto al grupo de calidad de CEGEL y a CALISOFT, para verificar que cumple con el mínimo de calidad necesaria. Se efectuaron 2 iteraciones con el objetivo de encontrar no conformidades (NC), midiendo algunos parámetros para los casos de pruebas y otros para la aplicación, los resultados se muestran en la siguiente tabla.

Tipos NC	1ra Iteración	2da Iteración
Casos de pruebas		
Redacción	29	13
Formato	11	3
Ortografía	37	13
Correspondencia con los requisitos funcionales	24	2
Sistema		
Formato	53	31
Ortografía	27	14
Correspondencia con los casos de pruebas	33	11
Correspondencia con la interfaz de los prototipos	41	2

Tabla 8: Resultado de pruebas internas realizadas al sistema**.**

Luego de aplicadas todas estas validaciones y pruebas, el producto fue entregado al equipo de calidad de CEGEL y a CALISOFT, donde le realizaron también pruebas de caja negra o funcionales, pruebas de liberación, entre otras. Los resultados de estas pruebas se pueden constatar en el (Capítulo 3. Vaoración de la Solución).

2.6. Conclusiones del capítulo

Se utilizaron en la implementación del módulo los patrones de diseño y estilos arquitectónicos. Se definió un conjunto de estándares de codificación para mejor organización, claridad de implementación y sin ambigüedades. Se elaboró el diagrama de paquetes, diagrama de diseño de clases, modelo de datos, diagrama de componentes y diagrama de despliegue a partir de las especificaciones de los requisitos funcionales para el modelo del diseño. Se mostraron los resultados obtenidos a partir de las métricas utilizadas para validar los requisitos, el análisis y el diseño, así como los resultados arrojados a partir de las pruebas internas realizadas.

Capítulo 3: Valoración de la solución

Introducción

En el presente capítulo se muestran los resultados de la propuesta de solución, obtenidos con la valoración de las variables tanto dependiente como independiente identificadas en el diseño teórico de la investigación.

3.1. Valoración de la variable dependiente

La variable dependiente analizada es Disponibilidad, para la valoración de esta variable fueron definidas las dimensiones tiempo y acceso que permiten determinar el nivel de disponibilidad del sistema desarrollado, de acuerdo a estas dimensiones se utilizaron dos indicadores.

Dimensión Tiempo

Indicador No. 1: Tiempo de respuesta de la aplicación.

Para poner en práctica este indicador se utilizó el Netbeans 7.2, donde se ejecutó la aplicación en modo profile para obtener el tiempo de respuesta dado por el IDE al registrar cinco trámites. A continuación se muestra una tabla con los resultados obtenidos:

Tipo de Trámite	Tiempo de respuesta(ms)
Prórroga Residencia Temporal	280
Permiso de Trabajo	649
Cambio de Dirección	9,99
Visados Múltiples	196
Residencia Temporal	79,8

Tabla 9: Muestra el tiempo de respuesta de una muestra de los trámites más complejos del módulo**.**

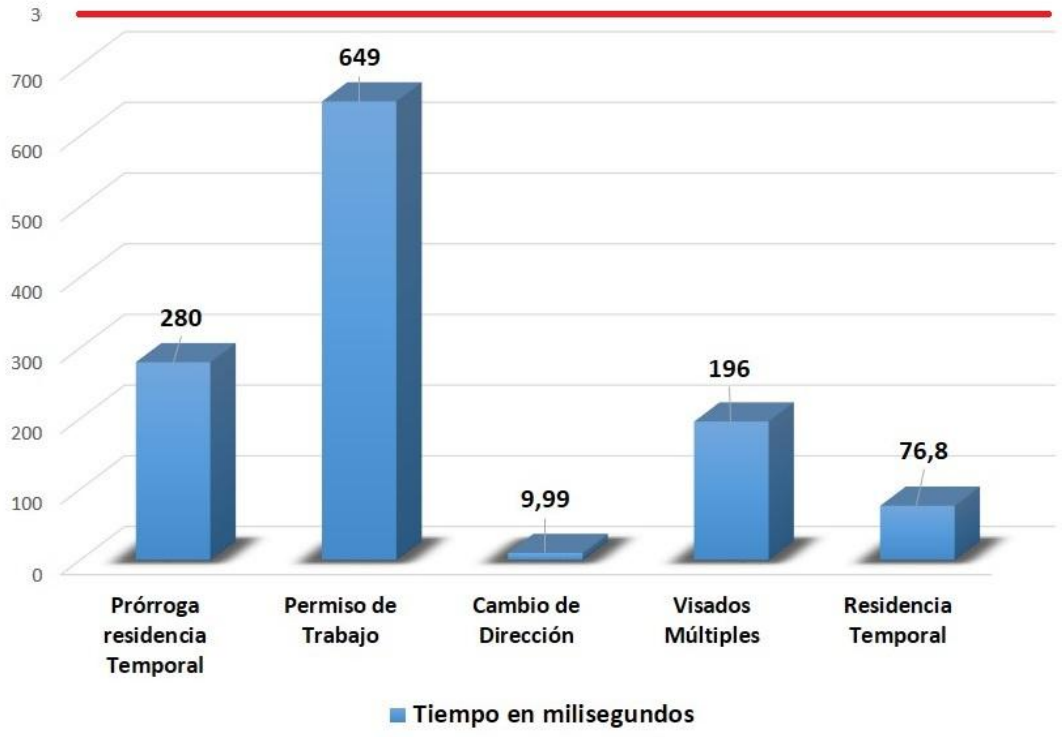


Imagen 24: Representación de resultados obtenidos del tiempo en milisegundos

Los resultados obtenidos muestran la rapidez del sistema en el cumplimiento de las peticiones del usuario. Estos valores se comparan con el tiempo de respuesta que plantea el requisito no funcional Usabilidad, del tipo Ambiente específicamente RnF7, mostrado en el Capítulo 2, epígrafe 2.2.2, el tiempo máximo es de 3 segundos, como en todos los casos es menor que el mínimo de tiempo que debería demorar el sistema en dar una respuesta al usuario se puede concluir que el indicador Tiempo de respuesta de la aplicación brindó resultados satisfactorios. Para una mejor comprensión de los tiempos mostrados remitirse a los Anexos 5, 6, 7, 8 y 9.

Dimensión Acceso

Indicador No. 2: Acceso a la información según los privilegios.

Para la comprobación de este indicador se realizaron dos procedimientos:

Procedimiento No1: Se realizó una encuesta a los funcionarios de la Cámara de Comercio que utilizarán la propuesta (ver Anexo 10), cuyas preguntas están destinadas a comprobar si realmente la información que se gestiona con la propuesta de solución está realmente acorde a los niveles de permiso del sistema y a las responsabilidades de los trabajadores. Fueron encuestados tres funcionarios, obteniendo resultados satisfactorios pues se pudo comprobar que los funcionarios de

la Cámara de Comercio solo tienen acceso a la información que les compete de acuerdo a los roles que ejercen en la institución.

Procedimiento No2: Análisis de la arquitectura del marco de trabajo. Después realizado este análisis centrado en la seguridad de la arquitectura del marco de trabajo se puede comprender la robustez del sistema en cuanto a la autenticación y autorización de usuarios de acuerdo al rol que cumplan. También se identificaron una serie de ventajas que le brinda al sistema la utilización de la arquitectura utilizada. Estas ventajas son:

- El sistema puede ser utilizado solo por usuarios autenticados en el mismo.
- Muestra las funcionalidades de acuerdo a quien esté autenticado en el mismo.
- Establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado.
- Asegura el almacenamiento de las credenciales de los usuarios utilizando algoritmos criptográficos que oculte la identidad verdadera de los usuarios.
- Permite almacenar toda la información de los usuarios sobre el sistema como constancia de las acciones realizadas.

3.2. Valoración variable independiente

La variable independiente analizada es Módulo de Registro de Trámites Migratorios. Para la valoración de esta variable se definieron la dimensión Alcance y Calidad, que permite valorar el módulo desarrollado, de acuerdo a estas dimensiones se utilizaron dos indicadores.

Dimensión Alcance

Indicador No1: Cumplimiento de los requisitos

El alcance de la solución se estimó a partir del indicador cumplimiento de los requisitos determinado por el cálculo del índice de requisitos implementados. El cálculo se realiza a partir de la siguiente fórmula:

- Índice de requisitos implementados = (Cantidad de requisitos implementados) / (Cantidad total de requisitos).

Para la consulta de información sobre los requerimientos tanto funcionales como no funcionales puede consultar en este documento el Capítulo 2: Propuesta del sistema, epígrafe 2.2.

Indicador: Índice de requisitos Implementados P (CR)

Variables:

- Cantidad de requisitos Implementados CRI=43

- Total de requisitos TR=43

P (CR) =CRI/TR

P (CR) = 43/43

P (CR) =1

A partir de los resultados obtenidos podemos afirmar que el alcance del módulo cumple con una probabilidad de valor 1, asegurando que la solución satisface el 100% de las funcionalidades requeridas por el cliente. Se puede afirmar que el sistema tiene un alcance total sobre los requisitos propuestos. Los resultados obtenidos por el grupo de calidad de CEGEL y CALISOFT se muestran a continuación.

Dimensión Calidad

Indicador No2: Pruebas de Liberación

Según lo planteado en el modelo de Integración de modelos de madurez de capacidades o Capability maturity model integration(CMMI por sus siglas en inglés), la calidad de un producto o de un sistema es en su mayor parte consecuencia de la calidad de los procesos empleados en su desarrollo y mantenimiento. (Anisbert Suárez Batista, 2010)

Las Pruebas de Liberación fueron ejecutadas por el equipo de Calidad interna del centro CEGEL como terceros, donde emplean una planilla de No Conformidades (NC) para guardar evidencias de los errores detectados durante las pruebas y corregirlos posteriormente por el Equipo de Desarrollo. Se realizó un proceso de 2 iteraciones, debido a que la fecha de entrada a Calisoft coincidió con las revisiones de calidad centro, en la primera iteración se detectaron 178, de ellas 116 fueron de la aplicación, 58 de caso de prueba y 17 no proceden, en la segunda 38, de ellas 32 de aplicación, 6 de caso de prueba y 8 no proceden.

Imagen 25: Representación de las NC detectadas por grupo de calidad CEGEL.

Las Pruebas de Liberación fueron aplicadas también por CALISOFT a todos los artefactos del proyecto antes de ser entregados al cliente para su aceptación. Fueron detectadas en la etapa de exploración inicial siete NC, de ellas tres de redacción, una de error de interfaz, dos de funcionalidad y una de correspondencia con otro artefacto. En la primera iteración se detectaron 17 NC, de ellas cinco son de validación, diez de funcionalidad, una de correspondencia con otro artefacto y una de redacción. En la segunda iteración se encontraron 12, de ellas ocho fueron de funcionalidad, una de redacción y tres de validación. En la tercera iteración fueron encontradas seis NC, de ellas cinco son de funcionalidad y una de error de interfaz. A continuación se muestra una gráfica como resultado de exponer el comportamiento de las NC detectadas.

Imagen 26: Representación de las No conformidades detectadas por CALISOFT.

3.3. Conclusiones del capítulo

De acuerdo a los resultados mostrados en este capítulo, obtenidos durante la aplicación de las diferentes métricas se arriba a la conclusión de que los objetivos propuestos fueron resueltos. Se valoraron las variables que daban solución al problema planteado y se logró un buen resultado en la implementación de los requisitos levantados satisfaciendo así en su totalidad a las necesidades de la Cámara de Comercio.

Conclusiones

Con la culminación del presente trabajo de diploma se desarrolló el Módulo de Registro de Trámites Migratorios para el Sistema de Informatización Registral de la Cámara de Comercio de la República de Cuba (SIRECC) contribuyendo al aumento de la disponibilidad de los procesos de registro de trámites migratorios. Los resultados obtenidos permiten concluir que:

- Se efectuó un análisis del marco teórico de la investigación, demostrando las bases científicas del trabajo.
- Una adecuada selección del modelo de desarrollo, las tecnologías y de la arquitectura permitió cumplir y satisfacer las necesidades de la Cámara de Comercio.
- La elaboración del modelo de diseño y sus respectivos diagramas constituyeron una guía para las actividades de implementación, al producir una representación técnica del componente desarrollado. Su uso propició la obtención de una entrada adecuada a las posteriores actividades de implementación.
- Se logró el efectivo desarrollo de los requisitos que integran el sistema a partir de la representación física de los componentes, y así proveer una solución a los problemas existentes en la Cámara de Comercio de la República de Cuba.
- Las pruebas realizadas demostraron que las funcionalidades del sistema tienen correspondencia con las definidas inicialmente.

Recomendaciones

Luego de dar cumplimiento al objetivo de este trabajo de diploma a lo largo de esta investigación y teniendo en cuenta las experiencias adquiridas durante el desarrollo, se recomienda:

- Continuar con el mejoramiento del software a partir de nuevos cambios que puedan surgir de acuerdo a nuevas necesidades de los clientes.
- Reutilizar componentes del sistema desarrollado en aplicaciones registrales.

Bibliografía

Java Persistence Query Language (JPQL). 2012. Aprendiendo Java. *Aprendiendo Java.* [En línea] 2012. [Citado el: 12 de 5 de 2014.] http://www.aprendiendojava.com.ar/index.php?topic=59.0.

Alarcon, Pozuelo De. 2004. *CURSO BASICO DE ANALISIS Y DISEÑO.* Madrid : s.n., 2004.

Altova. 2013. Altova. *Altova.* [En línea] 2013. [Citado el: 22 de 3 de 2014.] http://www.altova.com/es/umodel/uml-package-diagrams.html.

Álvarez de Zayas, Dr. Carlos. 1995. *Metodología de la investigación científica.* Santiago de Cuba : s.n., 1995.

Andrew Lee Rubinger, Bill Burke. 2010. *Enterprise Java Beans 3.1.* 2010.

Anisbert Suárez Batista, Maikel Muñoz Rojas, Yaimí Trujillo Casañola,José Manuel Pardo, Lissette Rodriguez, Odannis Enamorado, Osay González, Yenly Pérez, Dagmay Aveleira, Asnier Enrique Góngora, Deborat, Dennis Neuland. 2010. *Implantación de CMMI nivel de madurez 2 en la Universidad de las Ciencias Informáticas: Área de proceso aseguramiento de la calidad a proceso y producto.* La Habana : Grupo Editorial Ediciones Futuro, 2010. 10.

Argentina, Direccion Nacional de MIgraciones de la República. 2014. Dirección Nacional de Migraciones Argentina. *Dirección Nacional de Migraciones Argentina.* [En línea] agosto de 2014. [Citado el: 22 de enero de 2014.] http://www.migraciones.gov.ar/.

Argentina, Direccion Nacional. 2014. Direccion Nacional Argentina. *Direccion Nacional Argentina.* [En línea] 2014. http://www.migraciones.gov.ar/.

Calisoft. 2010. Centro Nacional de Calidad de Software. *Centro Nacional de Calidad de Software.* [En línea] 2010. [Citado el: 9 de 4 de 2014.] https://calisoft.uci.cu.

Cámara de Comercio de la República de Cuba. [En línea] http://www.camaracuba.cu/.

Cámara de Comercio de la República de Cuba. 2009. El Portal del Empresario en Cuba. [En línea] 2009. www.camaracuba.cu.

Carlos Reynoso, Nicolás Kiccillof. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

Carlos, Universidad Rey Juan. 2013. *Arquitecturas Software.* España : s.n., 2013.

CEGEL. 2013. Suite de Gestión de Proyectos (GESPRO 13.05). Laboratorio de Investigaciones en Gestión de Proyectos, UCI. *Suite de Gestión de Proyectos (GESPRO 13.05). Laboratorio de Investigaciones en Gestión de Proyectos, UCI.* [En línea] 2013. [Citado el: 22 de 4 de 2014.] http://gespro.cegel.prod.uci.cu/.

Certificada, Auditor de Sistemas de informacion. 2013. *Confidencialidad, integridad y disponibilidad de la información.* 2013.

Claudia A. Romina Ledesma, y otros. 2010. *Trabajo de investigación: J2EE.* Argentina : s.n., 2010.

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

David Garlan, Mary Shaw. 1994. *An Introduction to Software Architecture.* 1994.

Definición. De. 2012. Definición. De. *Definición. De.* [En línea] 2012. [Citado el: 21 de 3 de 2014.] http://definicion.de/lenguaje-de-programacion/.

Dennis Neuland, Anisbert Suarez Batista, Kariné Ramos Blanco, Ramsés Delgado Martínez, Deborat Pérez Montalván. 2011. *Lecciones aprendidas e implementación de PCM en la definición de procesos de desarrollo de software en la Universidad de las Ciencias Informáticas.* La Habana : Serie Científica de la Universidad de las Ciencias Informáticas, 2011. 2.

Dirección Nacional de Migraciones. [En línea] http://www.migraciones.gov.ar.

2014. EcuRed. *EcuRed.* [En línea] 29 de marzo de 2014. http://www.ecured.cu/index.php/Herramienta_CASE.

EcuRed. 2014. EcuRed. *EcuRed.* [En línea] 28 de marzo de 2014. http://www.ecured.cu/index.php/Lenguaje_de_programaci%C3%B3n.

—. 2014. EcuRed. *EcuRed.* [En línea] 28 de marzo de 2014. http://www.ecured.cu/index.php/Java.

Egdamar. 2012. Diseño UML. *Diseño UML.* [En línea] 2012. [Citado el: 21 de 3 de 2014.] http://egdamar877.blogspot.com/2009/05/expocicion.html.

ERIKA CAMACHO, FABIO CARDESO, GABRIEL NUÑEZ. 2004. *ARQUITECTURAS DE SOFTWARE.* 2004.

EVA. 2014. *Materiales Documentación sobre Pruebas en el Entorno Virtual de Aprendizaje(EVA).* 2014.

Fergarciac . 2013. Entorno de Desarrollo Integrado (IDE). *Entorno de Desarrollo Integrado (IDE).* [En línea] 2013. [Citado el: 24 de 3 de 2014.] http://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

—. 2013. Modelos de Bases de Datos. *Modelos de Bases de Datos.* [En línea] 2013. [Citado el: 9 de 4 de 2014.] http://fergarciac.wordpress.com/2013/01/29/modelos-de-bases-de-datos/.

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.* La Habana : s.n., 2011.

Gutierrez, Demián. 2009. *UML Diagramas de Paquetes.* Universidad de los Andes, Venezuela : s.n., 2009.

HUMBERTO. 2007. *LIBRO DE BIBLIOTECAS.* 2007.

IEEE. 1999. IEEE Advancing Technology for Humanity. *IEEE Advancing Technology for Humanity.* [En línea] 1999. [Citado el: 21 de 3 de 2014.] http://www.ieee.org.

Ithleovi. 2010. Mdelo de implementación. *Mdelo de implementación.* [En línea] 2010. [Citado el: 9 de 4 de 2014.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

James Rumbaugh, Ivar Jacobson, Grady Booch. 2000. *El lenguaje unificado de modelado. Manual de referencia.* Madrid : s.n., 2000.

Julio C. Prieto, Lissuan Fadraga Artiles. 2011. *Documento de la Arquitectura.* La Habana : s.n., 2011.

Krol, Felicita de. 2010. *Plataformas J2SE, J2EE, J2ME.* UNIVERSIDAD TÉCNOLOGICA DE PANAMÁ : s.n., 2010.

Larman, Craig. 2005. *UML y Patrones 2da Edición.* 2005.

—. 1999. *UML y Patrones. Introducción al análisis y diseño de objetos.* 1999.

Lianet Cabrera González, Enrique Roberto Pompa Torres. 2012. *Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.* La Habana : s.n., 2012.

mejora, Programa de. 2008. *CICLO DE VIDA DE PROYECTOS PILOTOS DEL PROGRAMA DE MEJORA.* 2008.

Merson, Paulo. 2009. *Data Model as an Architectural View.* 2009.

Migratorios, Sistema Electónico de Trámites. 2014. SETRAM. *SETRAM.* [En línea] 2014. [Citado el: 19 de febrero de 2014.] http://www.visasmex.com/Esp/documentos_migratorios/setram/setram.html.

Mike Keith, Merrick Schnicariol. 2009. *Pro LPA 2 Mastering the Java Persistence API.* 2009.

2014. monografias.com. *monografias.com.* [En línea] 19 de febrero de 2014. http://www.monografias.com/trabajos28/camara-comercio/camara-comercio.shtml.

Monografias.com. 2013. monografias.com. *monografias.com.* [En línea] 2013. [Citado el: 21 de 3 de 2014.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml.

Oracle Corporation. 2013. Oracle. *Oracle.* [En línea] Oracle, 2013. [Citado el: 12 de 3 de 2014.] http://www.oracle.com/technetwork/java.

—. 2013. Oracle Corporation. NetBeans IDE. The Smarter and Faster Way to Code. *Oracle Corporation. NetBeans IDE. The Smarter and Faster Way to Code.* [En línea] 2013. [Citado el: 21 de 2 de 2014.] https://netbeans.org/..

—. 2013. ORACLE. GlassFish - World's first Java EE 7 Application Server. *ORACLE. GlassFish - World's first Java EE 7 Application Server.* [En línea] 2013. [Citado el: 29 de marzo de 2014.] https://glassfish.java.net.

Paul Clements, Judith Stafford y otros. 2003. *Documenting Software Architectures in an Agile World.* 2003.

Paul Clements, Rick Kazman, Mark Klein. 2001. *Evaluating a Software Architecture .* 2001.

PostgresSQL. 2014. PostgresSQL. *PostgresSQL.* [En línea] 29 de marzo de 2014. http://www.postgresql.org/about/news/1481/.

Pressman, Roger S. 1997. *Ingenieía de Software.* Madrid : s.n., 1997.

Pressman, Roger. S. 2007. *Ingenniería del software. Un enfoque práctico.* 2007.

Programa de Mejora. 2011. *Ciclo de vida d eproyectos pilotos del programa de mejora.* La Habana : s.n., 2011.

Reinel Muñoz Perez, Arianna Devesa Navarro. 2008. *Módulo Hospitalización del Sistema de las HIS.* 2008.

Reynoso, Billy. 2012. *Architect Academy.* Buenos Aires : s.n., 2012.

Reza Rahman, Derek Lane. 2007. *EJB 3 in action.* 2007.

Sandra González Valdés, René Suárez Font. 2011. *Diseño e Implementación de los módulos de Preparación de Documentos.* La Habana : s.n., 2011.

2014. Scridb. *Scridb.* [En línea] 29 de marzo de 2014. http://es.scribd.com/doc/73801129/Ingenieria-de-Software-Basada-en-Componentes.

Security, Official Website of the Department of Homeland. 2014. U.S. Citizentship and Inmigration Services. *U.S. Citizentship and Inmigration Services.* [En línea] 1 de junio de 2014. [Citado el: 1 de junio de 2014.] file:///G:/USCIS%20ELIS%20_%20USCIS.htm.

Server, U.S. Ctizenship and Inmigration. 2014. U.S. Ctizenship and Inmigration Server. *U.S. Ctizenship and Inmigration Server.* [En línea] 2014. http://www.uscis.gov/uscis-elis.

Sheyla García, Felix Gonzalez. 2013. *Extensión del NetBeans IDE para el diseño de Interfaces. Gráficas de Usuario con Ext JS.* La Habana : s.n., 2013.

Sommerville, Ian. 2005. *Ingeniería de Software (Séptima edición).* 2005.

Venezuela, Gobierno Bolivariano de. 2014. Servicio Administrativo de Identificación, Migración y Extranjería. *Servicio Administrativo de Identificación, Migración y Extranjería.* [En línea] 1 de mayo de 2014. [Citado el: 1 de mayo de 2014.] http://pasaporte.saime.gob.ve/.

Wikibooks.org. 2013. Wikilibros. *Wikilibros.* [En línea] 2013. [Citado el: 21 de 2 de 2014.] http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Caracter%C3%ADsticas_del_lenguaje.

Glosario de Términos

Disponibilidad: asegura que el acceso a los datos o a los recursos de información por personal autorizado se produce correctamente y en tiempo. Es decir, la disponibilidad garantiza que los sistemas funcionan cuando se les necesita. (Certificada, 2013)

Patrón: es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. (Pressman, 2007)

Alcance: El alcance de software es la especificación de los requisitos del software estableciendo las metas y objetivos del mismo, describiéndolos en el contexto del sistema basado en computadora. (Sommerville, 2005)

Prueba: Una prueba es una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan y registran, y se realiza una evaluación de algún aspecto. (Pressman, 2007)

Anexos:

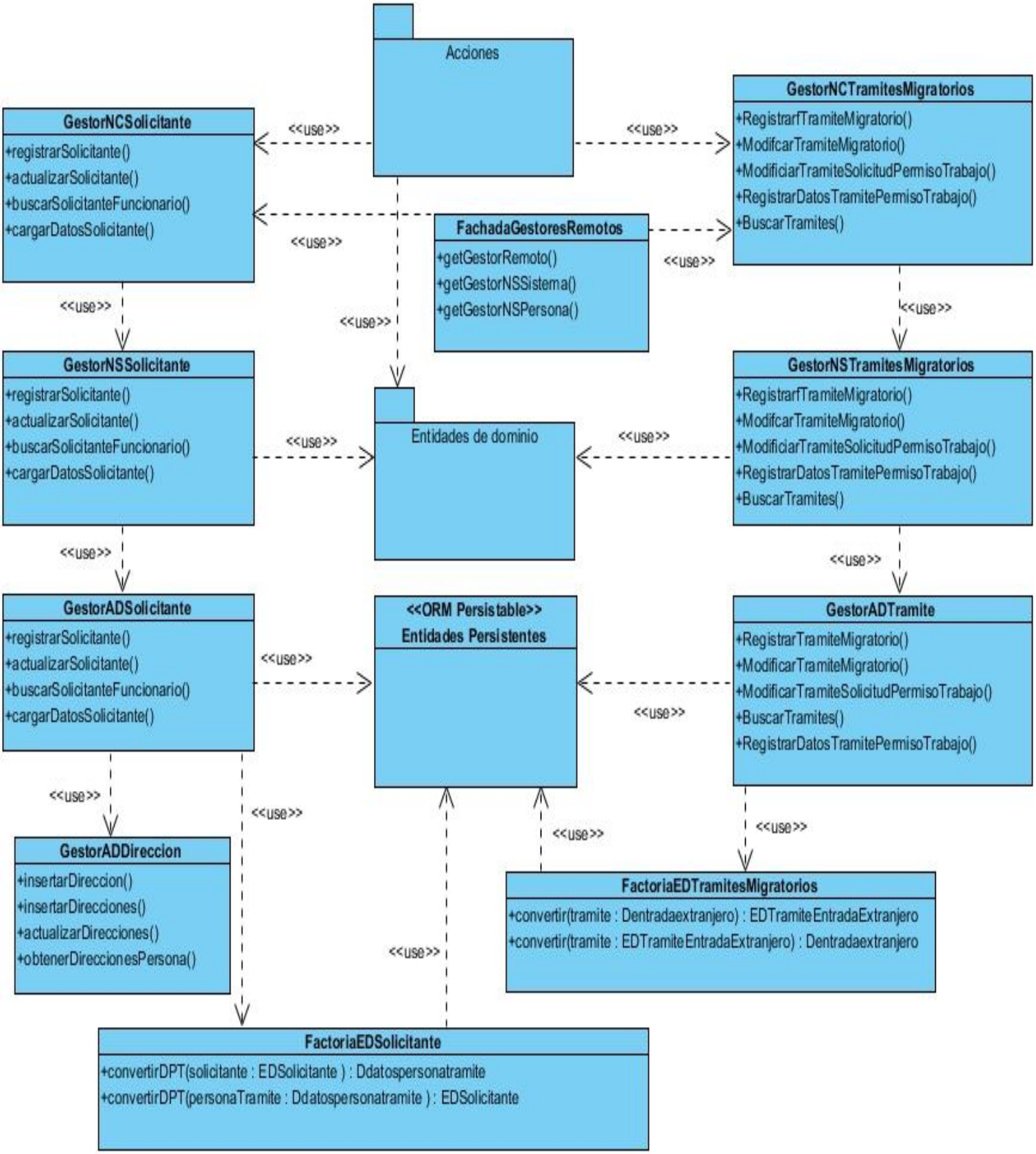
Anexo 1: Evaluación de los atributos de calidad para TOC.

Parámetros	Evaluación	
	Categoría	Criterio
Responsabilidad	Baja	< = Promedio
	Madia	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Complejidad de Implementación	Baja	< = Promedio
	Madia	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Reutilización	Baja	> 2 * Promedio
	Madia	Entre Promedio y 2 * Promedio
	Alta	< = Promedio

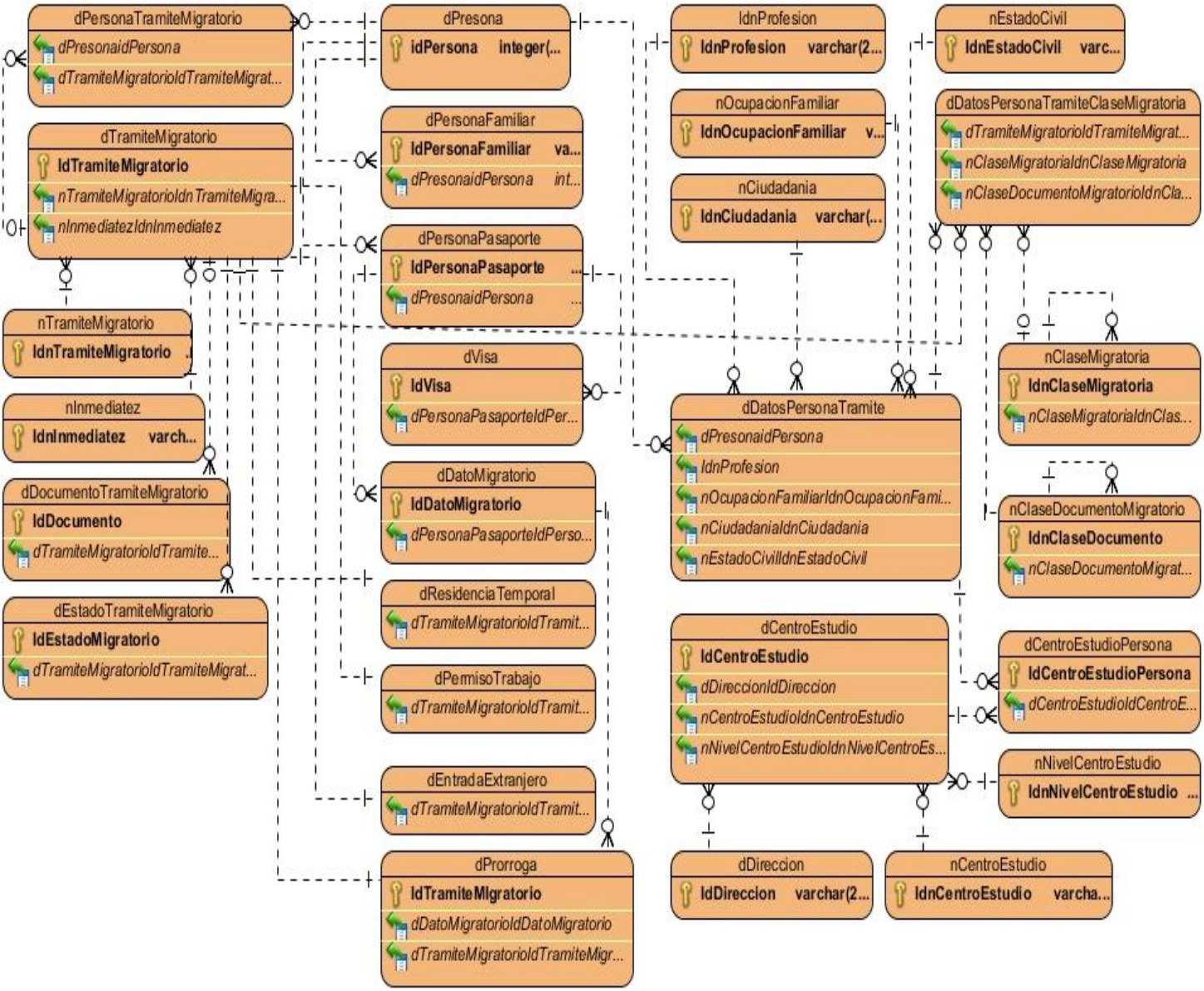
Anexo2: Evaluación de los atributos de calidad para RC.

Parámetros		Evaluación
	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Anexo 3: Diagrama de Clases del Diseño Módulo Registro de Trámites Migratorios.



Anexo 4: Modelo de datos del Módulo Trámites Migratorios.

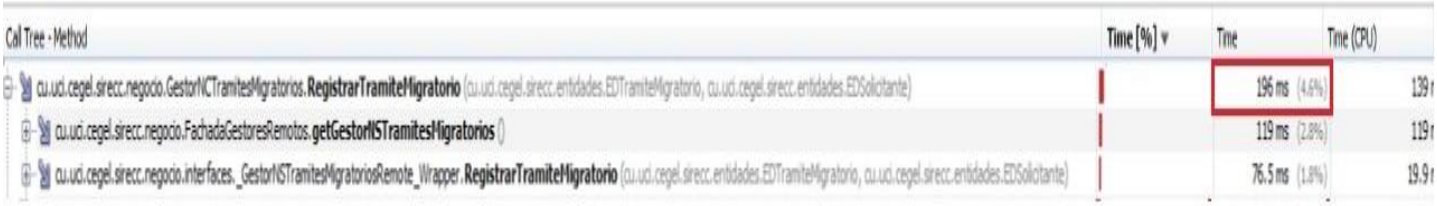


Anexo 5: Tiempo obtenido en el sistema por el trámite Prórroga Residencia temporal.

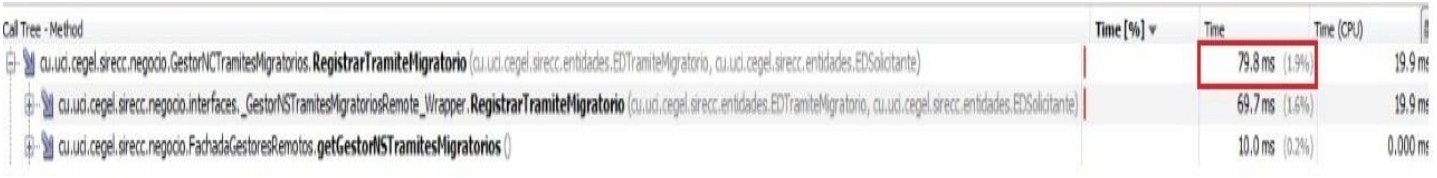
Anexo 6: Tiempo obtenido en el sistema por el trámite Permiso Trabajo.

Anexo 7: Tiempo obtenido en el sistema por el trámite Cambio Dirección.

Anexo 8: Tiempo obtenido en el sistema por el trámite Visados Múltiples.



Anexo 9: Tiempo obtenido en el sistema por el trámite Residencia temporal.



Anexo 10: Encuesta realizada al personal que labora en la Cámara de Comercio, perteneciente a la oficina de Trámites Migratorios.

1- ¿El sistema actual contiene los nomencladores con los conceptos necesarios para trabajar con el sistema?

☐ Si

☐ No

a- En caso de ser No especifique algunas de las siguientes soluciones actuales del problema.

☐ Se escribe textualmente en un campo de texto.

☐ No se tiene en cuenta.

☐ Se copia de otra herramienta (Ejemplo: Excel, Access, Word).

2- Los datos que se gestionan en el sistema actual son:

☐ Insuficientes.

☐ Suficientes.

☐ Suficientes con información de más.

☐ Insuficientes con información de más

a- En caso de ser Insuficientes ¿Cómo se llenan los datos que faltan?

_____

_____

_____

_____

_____

3- ¿El sistema actual permite generar reportes?

☐ Si

☐ No

a- En caso de ser afirmativo:

☐ Son confiables.

☐ No son confiables.

b- En caso de que no sean confiables diga ¿Por qué?

☐ Información actual con errores.

☐ Información incompleta.

☐ No se ajustan al formato establecido.

4- El sistema actual permite consultar las bitácoras de un trámite específico.

☐ Si

☐ No

a- En caso de No ¿Qué se hace para llenarlas según su importancia?

5- ¿El sistema brinda la posibilidad de autenticarse por medio de un usuario y contraseña?

☐ Si

☐ No

a- En caso de ser afirmativo. ¿Sólo tiene acceso a la información referente a Trámites Migratorios?

☐ Si

☐ No

6-¿Brinda el sistema alguna protección en caso de inactividad?

☐ Si

☐ No

a- En caso de ser Si, diga qué tipo de protección:

_____

_____

_____

7-¿Cuándo funcionarios de otras áreas se han autenticado en el sistema han tenido acceso a la información referente a Trámites Migratorios?

☐ Si

☐ No

8-¿Se registran correctamente los trámites realizados a un solicitante?

☐ Si

☐ No