UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Título: Desarrollo de componentes de negocio para el piloto emisión de pasaportes diplomáticos de la República Bolivariana de Venezuela.

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

Autores

Melba Fortunato Brandford

Erick Vega de la Cruz

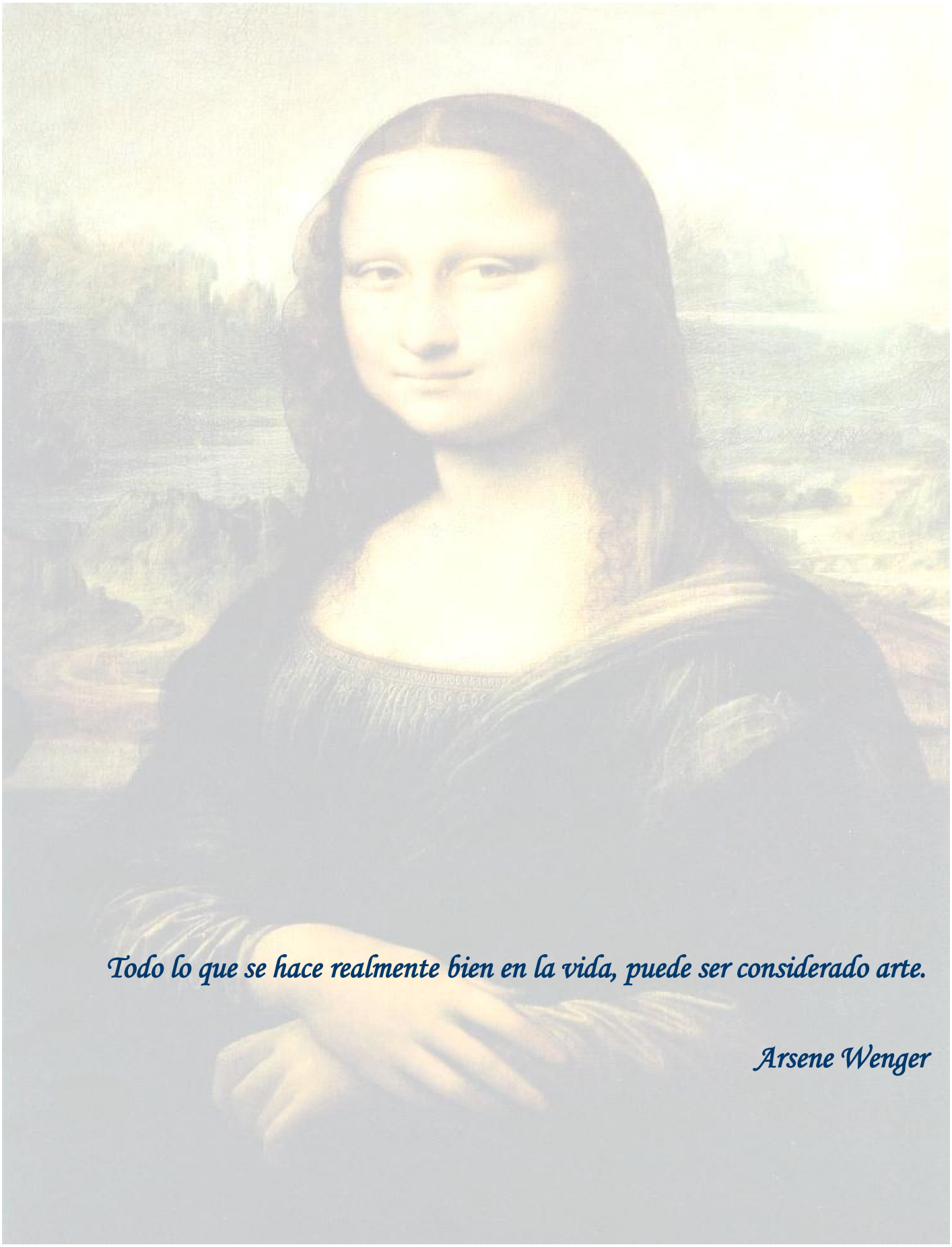
Tutores

Ing. Keytia Quintero Ruiz

Ing. Adiary Hernández Regueiro

Ciudad de La Habana. Junio, 2010

"Año 52 de la Revolución"



Todo lo que se hace realmente bien en la vida, puede ser considerado arte.

Arsene Wenger

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo de diploma y reconocemos al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de junio del año 2010.

Melba Fortunato Brandford

_____

Firma del Autor

Keytia Quintero Ruiz

_____

Firma del Tutor

Erick Vega de la Cruz

_____

Firma del Autor

Adiary Hernández Regueiro

_____

Firma del Tutor

OPINIÓN DEL TUTOR

AGRADECIMIENTOS

A la tierra que nos vio nacer y a la Revolución por ofrecernos la posibilidad de estudiar y realizarnos como profesionales.

A todas las escuelas en las que cursamos las diferentes etapas de enseñanza.

A todos los profesores que de una forma u otra aportaron su granito de arena en nuestra preparación, como personas y como profesionales.

A todo el equipo de trabajo del proyecto, por su pequeño aporte en este trabajo.

A los tutores, por la paciencia y dedicación que tuvieron con nosotros.

Melba:

A mis padres por apoyarme en todo y estar ahí cuando los necesito, mami y papi sin ustedes no sería la persona que soy.

A mi hermana con alma de artista por aguantarme todos estos años.

A mi novio, sólo te quiero decir una cosa, gracias por amarme tanto.

A toda mi familia, gracias por estar.

A todas mis amistades que de una forma u otra me ayudaron a lo largo de estos cinco años, en especial una gracias bien grande para ustedes: Danay, Adianez, Ilies y Yosmel. Y por supuesto a mi mejor amiga, mi segunda hermana, que la quiero con la vida Lisbey.

Erick:

A mi familia, el soporte espiritual de mi vida.

A mi novia, gracias por cambiar el curso de mi vida.

A la familia de mi novia, no sabía que existía tanto cariño en el mundo.

A todos mis amigos, por sus consejos y discusiones de tantos años.

DEDICATORIA

Melba:

Le dedico mi tesis a mi papá, por ser mi ejemplo, mi paradigma a seguir, por ser la luz que ilumina mi camino y por tener tantas cualidades que he admirado y admiraré siempre.

Erick:

A mi familia, a esta tierra que me vio nacer, y a la Revolución, es un orgullo ser un hijo de ella.

RESUMEN

Como parte de las transformaciones que está viviendo la República Bolivariana de Venezuela, surge el Proyecto SAIME (Servicio Administrativo de Identificación, Migración y Extranjería) con el objetivo fundamental de reestructurar los procesos de identificación, migración y extranjería dentro de la ONIDEX (Oficina Nacional de Identificación y Extranjería), dando paso al surgimiento de la Misión Identidad, que ha posibilitado identificar y regularizar a miles de venezolanos e inmigrantes extranjeros. Hasta la fecha se han automatizado los procesos de emisión de cédulas de identidad y pasaportes ordinarios, incorporando a este último proceso tecnologías avanzadas. Sin embargo, el proceso de emisión de pasaportes diplomáticos no está automatizado aún, realizándose éste, hoy en día, de forma engorrosa, burocrática, manual y manuscrito. Por lo que el MPPRE (Ministerio del Poder Popular de Relaciones Exteriores), necesita de un sistema que facilite y automatice el proceso de este tipo de documento. Para llevar a cabo esta tarea se crea el proyecto Pasaporte Diplomático en el Centro de Identificación y Seguridad Digital. Este proyecto se propone desarrollar una solución utilizando herramientas y tecnologías libres de costo e introduciendo otras nuevas en materia de gestión de procesos de negocio, así como el trabajo sobre una arquitectura con principios de arquitecturas orientadas a servicios. Este tipo de solución incluye la utilización de un motor de procesos, que para su correcto funcionamiento necesita del consumo de un conjunto de servicios relacionados con el negocio del sistema. Siendo estos servicios el puente de comunicación entre el motor de procesos y la base de datos del negocio.

Palabras claves: SAIME, ONIDEX, MPPRE, pasaportes diplomáticos, servicios

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La República Bolivariana de Venezuela está inmersa en un complejo proceso de construcción, generación y perfeccionamiento del nuevo Estado Venezolano, al transformar y sustituir los viejos sistemas de orden político y social; por completas y modernizadas instituciones públicas que basan el flujo de sus procesos en los más novedosos adelantos tecnológicos.

Como parte de estas transformaciones surge el Proyecto SAIME (Servicio Administrativo de Identificación, Migración y Extranjería) con el objetivo fundamental de reestructurar los procesos de identificación, migración y extranjería dentro de la ONIDEX (Oficina Nacional de Identificación y Extranjería), dando paso al surgimiento de la Misión Identidad, que ha posibilitado identificar y regularizar a miles de venezolanos e inmigrantes extranjeros (ONIDEX, 2007). Dicha misión demostró que el pueblo y las instituciones en Venezuela estaban listos para iniciar, a una mayor escala, el perfeccionamiento de su sistema de identificación. Dentro de los procesos automatizados se encuentra el de emisión de pasaportes ordinarios, empleando para el mismo, tarjetas de identificación electrónicas, las cuales incorporan lo más novedoso en materia de identidad, permiten además almacenar la información necesaria para comprobar, con alto grado de seguridad, la identidad de su portador, contando también con un mecanismo de verificación biométrico.

Sin embargo y pese a todo esto, el proceso de emisión de pasaportes diplomáticos no está automatizado aún, en el mismo existen inconvenientes en la elaboración de los pasaportes, el cual se maneja de forma manual y con poca seguridad. El registro de la información en los libros se hace de forma manuscrita, generando retrasos y poca confiabilidad en la escritura (errores ortográficos, lectura engorrosa). La transcripción en el sistema actual ocasiona pérdida de tiempo, debido a que el sistema no genera una respuesta rápida a la hora de ingresar información, también existen retrasos a la hora de consultar la misma. El procedimiento actual de obtención de reportes no es el correcto o el más adecuado, lo que atenta con la limpieza y rapidez del mismo, además no se lleva un orden y control sobre la expedición de pasaportes en años anteriores, siendo esto un problema a la hora de realizar búsquedas. Por todos estos inconvenientes el MPPRE, necesita de un sistema que facilite y automatice el proceso de emisión de pasaportes diplomáticos. Para llevar a cabo esta tarea se crea el proyecto Pasaporte Diplomático en el Centro de Identificación y Seguridad Digital. Este proyecto se propone desarrollar una solución utilizando herramientas y tecnologías libres de costo e introduciendo otras nuevas en materia de gestión de

procesos de negocio, así como el trabajo sobre una arquitectura con principios de arquitecturas orientadas a servicios.

Este tipo de solución incluye la utilización de un motor de procesos, que para su correcto funcionamiento necesita del consumo de un conjunto de servicios relacionados con el negocio del sistema. Siendo estos servicios el puente de comunicación entre el motor de procesos y la base de datos del negocio. Precisamente, es en esta parte de la solución, donde se centra la investigación, por lo que el **problema científico** de la misma queda formulado de la siguiente forma: ¿Cómo garantizar el cumplimiento de los servicios necesarios en el proceso de emisión de pasaportes diplomáticos de la República Bolivariana de Venezuela?

En virtud de guiar y dar cumplimiento a la presente investigación se plantea como **objetivo general**: Desarrollar los componentes de negocio para el proceso de emisión de pasaportes diplomáticos de la República Bolivariana de Venezuela.

El mismo contiene los siguientes **objetivos específicos**:

- Crear la lista de funcionalidades de los componentes de negocio.
- Diseñar las funcionalidades de los componentes de negocio.
- Construir las funcionalidades de los componentes de negocio.
- Probar las funcionalidades de los componentes de negocio.

Se define como **objeto de estudio** los procesos de negocio y la arquitectura definida en el Sistema de Emisión de Pasaportes Diplomáticos.

Se especifica como **campo de acción** la capa de componentes de negocio del Sistema de Emisión de Pasaportes Diplomáticos.

Para guiar la investigación se plantea como **hipótesis**: El desarrollo de componentes de negocio proporcionará los servicios necesarios a los procesos de negocio del Sistema de Emisión de Pasaportes Diplomáticos.

De la hipótesis antes mencionada se plantean las siguientes variables de la investigación:

- Independiente: desarrollo de componentes de negocio.
- Dependiente: los servicios necesarios a los procesos de negocio del Sistema de Emisión de Pasaportes Diplomáticos.

Operacionalización de las variables:

Variable conceptual	Dimensión	Indicadores	Unidad de medida
Desarrollo de componentes de negocio	Factibilidad	Tiempo de desarrollo	Corto plazo
			Largo plazo
		Costo de desarrollo	Alto
			Medio
			Bajo
Los servicios necesarios para los procesos de negocio.	Funcionalidad	Funcionalidad	3
		Baja funcionalidad	2
		Funcionalidad nula	1

Tabla 1: Variables de la hipótesis

Como **población** de la investigación se tienen los procesos de negocio, teniendo en cuenta que el tamaño de la población es menor que 100, se toma como muestra para la investigación la totalidad de la población y como unidad de estudio, los servicios que consumen cada uno de los procesos de negocio.

Dentro de las **técnicas de muestreo** existentes se utilizó la No-Probabilística Accidental, ya que se incluye a todos los elementos disponibles, seleccionándolos arbitrariamente sin tener en cuenta ninguna técnica especial, hasta llegar a la totalidad de ellos (LEÓN, 2002).

Para dar cumplimiento a estos elementos se proponen las siguientes **tareas de la investigación**:

- Identificación de la metodología a utilizar.
- Definición y caracterización de los procesos de negocio existentes.
- Caracterización de las herramientas a utilizar.
- Definición de las funcionalidades de los componentes.
- Identificación de los elementos arquitectónicos dentro del sistema.
- Diseño de las funcionalidades de los componentes.
- Implementación de las funcionalidades de los componentes.

- Integración de los módulos del sistema.
- Definición y ejecución de pruebas unitarias al sistema.

Para llevar a cabo estas tareas de utilizarán los siguientes **métodos científicos de investigación**:
Métodos teóricos:

- Método Histórico: Se consultará bibliografía referente al tema de investigación, toda su trayectoria, evolución y comportamiento.
- Método Lógico: Se estructurará la documentación investigada de una manera organizada y cronológica, para así tener un mejor entendimiento de la misma.
- Método de la Modelación: Permitirá la creación de modelos, esto permitirá representar de manera gráfica algún contenido que lo requiera, como los componentes, utilizando las herramientas para ello.
- Método Sistémico: Se utilizará este método para estudiar la integración de las tecnologías utilizadas, mediante la determinación de sus componentes, así como la relación entre ellos. Esta relación determina por un lado la estructura y la jerarquía de cada componente y por otra parte su dinámica, siendo también la expresión del comportamiento del sistema como totalidad en que un componente depende de otro u otros.

Métodos empíricos:

- Método de la observación: Este método se pondrá en práctica cuando, de forma consciente, se planifique la investigación orientada hacia el logro de un objetivo determinado.
- Método experimental: El experimento es el método empírico para el estudio de un objeto en el cual se crearán las condiciones o se adaptarán las existentes para verificar una hipótesis, una teoría o un modelo.

Apoyo sobre los procesos de:

- Análisis: El cual permite la división mental de lo investigado, en relaciones y componentes para una mejor comprensión.
- Síntesis: Se establece mentalmente la unión entre las partes previamente analizadas, resaltando sus principales características, conceptos y relaciones.

Estructura del trabajo de diploma:

- Capítulo 1: Fundamentación teórica: En este capítulo se realizará el estudio del estado del arte de la investigación, tanto a escala nacional, internacional, como en la universidad. Se tratarán los principales conceptos relacionados al campo de acción. Se realizará además una descripción de la metodología, herramientas y lenguajes que serán utilizados.
- Capítulo 2: Características de los componentes: Se analizarán los procesos de negocio que se realizan para la emisión de pasaportes diplomáticos. Se realizará un modelo general y se identificarán los requisitos funcionales y no funcionales de los servicios que consume el proceso. Además de realizar una planificación por iteraciones.
- Capítulo 3: Diseño de los componentes: Se realizará el diseño de los componentes, obteniendo los artefactos para su posterior implementación.
- Capítulo 4: Construcción y pruebas: En este capítulo se describirá la implementación de los componentes de negocio y las técnicas usadas, obteniendo artefactos y los diagramas de componentes. Además de realizar el diseño y ejecución de los casos de prueba a utilizar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

Con el acelerado avance tecnológico de la información, la cantidad y la complejidad de los productos de software se están incrementando considerablemente, así como también la exigencia en su funcionalidad y confiabilidad. Una necesidad sentida en dicho medio es el hecho de que los productos de software deben ser desarrollados con base en la implantación de estándares mundiales, modelos, sistemas métricos, capacitación del recurso humano y otros principios y técnicas de la ingeniería de software, que garanticen la producción de software con calidad y competitividad a nivel local e internacional.

Un factor estratégico para proyectos productivos desarrolladores de software debe ser la aplicación de modelos de mejoramientos de procesos y de nuevas herramientas en el mercado, que una vez adoptados permitan un mejor y más eficiente desarrollo del software, por las grandes ventajas que ofrece. Siguiendo este criterio el proyecto Pasaportes Diplomáticos decidió el uso de nuevas herramientas de desarrollo, además de una metodología ágil.

En este capítulo se presentan los conceptos básicos relacionados con el proceso de tramitación de pasaportes diplomáticos para lograr una mayor comprensión del problema a resolver. También se realiza un estudio de algunos sistemas con una arquitectura similar a la concebida en el proyecto a escala mundial, nacional y en la universidad. Además se describen las tecnologías, herramientas y metodologías a utilizar en el desarrollo del trabajo.

1.2 CONCEPTOS RELACIONADOS AL TEMA

Para un mejor entendimiento del tema resulta necesario comenzar desde la base, la cual sería una definición clara de identidad. La **identidad** es el conjunto de los rasgos propios de un individuo o de una comunidad, en este sentido, la idea de identidad está asociada a algo propio, único, entonces se infiere que identidad civil es el conjunto de atributos confirmados por un ente que permite diferenciar a una persona de forma única del resto. Estos atributos son recogidos en un documento de identificación.

Se conoce como **documento de identidad o de identificación**, al documento que emite el estado para posibilitar la identificación personal de cada ciudadano. Todas las personas deben tener un documento de identidad que acredite quién es y que le permita acceder a los servicios estatales o de otro tipo.

La emisión de un documento de identificación es un proceso complejo, que requiere un alto nivel de seguridad y control, basada en normas legislativas establecidas para el lugar donde el documento tiene efecto legal.
Los documentos aceptados como identificación para Venezuela son los siguientes: Cédula de Identidad, también llamado documento nacional de identidad (DNI) o carné de identidad, la licencia de conducir, la tarjeta de conscripción militar y los pasaportes.
Dentro de los pasaportes existen diferentes categorías:

El **pasaporte ordinario** es el documento de identificación personal que expide el Ministerio del Poder Popular para Relaciones Interiores y de Justicia a los venezolanos. El pasaporte como documento de identidad servirá para comprobar la nacionalidad de su portador confiriéndole las garantías establecidas en la Constitución de la República Bolivariana de Venezuela (VENEZUELA, 2009).

El **pasaporte diplomático** es un documento de viaje y de identificación concedido a quienes desempeñan determinadas funciones o cargos de alta dignidad, de responsabilidad nacional, que en muchos de los casos cumplen misiones oficiales en el exterior. Este tipo de pasaporte según la convención de Viena para las Relaciones Internacionales tiene muchísimas ventajas para el funcionario diplomático, entre las que se destacan: la inviolabilidad de la propiedad, la excepción del pago de impuestos de cualquier índole, la imposibilidad de ser sancionado por las fuerzas policiales, entre otras.

El **pasaporte de servicios** es otorgado a miembros del personal de la misión diplomática, como los agentes diplomáticos (acreditados con este carácter y que gozan de estatuto diplomático, como los agregados o consejeros), los miembros del personal administrativo y técnico (que incluso no siendo acreditados como personal diplomático gozan de ciertas inmunidades y privilegios) y los miembros del personal de servicio de la misión.

1.3 SOLUCIONES INFORMÁTICAS CON ARQUITECTURAS Y/O PROCESOS DE NEGOCIO SIMILARES

Como parte de la investigación y en aras de enriquecer la propuesta de solución que se quiere ofrecer, se estudiaron algunos sistemas encontrados con características similares en el ambiente de desarrollo a las definidas en el marco de trabajo de la investigación. Se estudiaron además algunos sistemas que automatizaran procesos semejantes al proceso de emisión de pasaportes diplomáticos, con el objetivo de

determinar si alguno se ajustaba a las necesidades de la investigación o tomar alguna experiencia de los mismos.

Servidor de aplicaciones biométricas orientado a servicios (BioSP por sus siglas en inglés).

BioSP es un servidor de aplicaciones orientado a servicios utilizado para integrar el procesamiento de los datos biométricos. Es muy adecuado para aplicaciones que requieren de la recolección de datos biométricos a través de una red distribuida, análisis, procesamiento, distribución, autenticación, y el intercambio de estos datos con otros componentes del sistema. Gestiona todos los aspectos del flujo de trabajo de transacciones, incluyendo la mensajería, las comunicaciones y las respuestas. Integra funciones biométricas con otros sistemas empresariales tales como: gestión de identidades, gestión de acceso, administración de tarjetas, y comunicación con el AFIS[1]. Soporta las modalidades de captación de huellas dactilares, reconocimiento de rostro, palma, e iris. Compuesto por 8 módulos. Incorpora lo último en componentes de código abierto y es compatible con J2EE.

Los procesos de negocio están automatizados mediante el lenguaje de ejecución de procesos de negocio *BPEL* (*Busines Process Execution Language).* Las operaciones de bajo nivel definidas en los módulos de BioSP se juntan en un script para formar servicios compuestos, permitiendo el procesamiento sincrónico y asincrónico de las transacciones y datos para satisfacer los requisitos de un amplio y variado escenario de casos de uso. Presenta una arquitectura en capas: presentación, procesos de negocio (donde se utiliza BPEL), capa de componentes y la de acceso a datos.

Identificación, Inmigración y Extranjería de la República de Cuba

Perteneciente al Centro de Identificación y Seguridad Digital de la universidad, entre las principales tareas que llevan a cabo se encuentra la realización de un framework. Es un framework para el desarrollo de aplicaciones web basadas en *workflow*, que permite la orquestación de procesos de negocio con *Windows Workflow Foundation* (WWF)[2]. Su principal objetivo es proporcionar un componente que permita gestionar

[1] Sistema Automático de Identificación de Huellas Dactilares

[2] Fundación de Windows para administrar flujos de trabajo

las instancias del *workflow*. Además encapsula un conjunto de actividades y servicios que le dan mayor dinamismo al desarrollo de sistemas centrado en la orquestación de procesos de negocio, específicamente para un ambiente web.

La propuesta está compuesta por diferentes módulos relacionados entre sí. Cada módulo está diseñado siguiendo un patrón de capas bien definidas y diseñadas para reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Definiéndose una serie de componentes que en su conjunto conforman el framework, estos componentes interactúan entre sí brindando una serie de funcionalidades.

Utiliza en su ambiente de desarrollo las siguientes herramientas: *Microsoft Visual Studio*, como gestor de base de datos Oracle, los frameworks *Windows Workflow Foundation (WWF)***,** Microsoft .NET 3.5, ADO.NET Entity, y como lenguaje de programación C#.

Sistema de Información Hospitalaria alas HIS

Este sistema pertenece a la facultad 7 de la universidad. Para su desarrollado se utilizó Java como lenguaje de programación y Eclipse Ganymede como entorno de desarrollo. Como gestor de base de datos seleccionaron PostgreSQL y como servidor de aplicaciones el JBoss Server. Para la administración de las reglas y procesos del negocio se utilizan Drools y JBoss JBPM respectivamente. Se definió una arquitectura en tres capas, presentándose de la siguiente manera:

Vistas o Interfaz: Esta capa incluye todas las interfaces que interactúan con el usuario.

Controlador o Lógica del negocio: En esta capa se encuentra lo referente a la lógica del negocio del sistema.

Modelo de datos: En esta capa se encuentran las entidades persistentes. Utilizándose la implementación de Hibernate 3.3 para realizar el mapeo objeto relacional.

Además para la integración entre capas se utiliza Seam. (DÍAZ, 2009)

Sistema Informatizado de Cooperación Internacional (SICI)

Dentro de la universidad se encuentra la Dirección de Cooperación Internacional (DCI), esta dirección es la encargada del intercambio con instituciones nacionales y del mundo de temas de intereses entre ambas partes. Para informatizar los procesos de la DCI, nace el proyecto productivo Sistema Informatizado de Cooperación Internacional (SICI). El cual cuenta con tres subsistemas (Trámites, Cooperación,

Investigación y Postgrado). Cooperación se encarga de gestionar becas para profesores, Investigación y Postgrado asigna las becas y el subsistema de Trámite tiene como función principal realizar la tramitación de asuntos migratorios y de extranjería en la universidad. Entre los servicios que brinda el sistema están: recogida de datos, generación de las planillas AO1[3], AO3 [4]y AO4[5], mostrar el estado del trámite, informar si ya la persona cuenta con visa o pasaporte, controlar la estancia en el país al cual se viajó, además de las visitas nacionales e internacionales que vienen a la universidad, entre otros. Fue realizado en Drupal, se utilizó como gestor de base datos PostgresSQL y como lenguaje de programación PHP 5.0.

Realizado un análisis detallado de lo documentado anteriormente se puede inferir que de los sistemas estudiados se pueden tomar experiencias a partir del trabajo y la calidad con que cuentan, probados dos de ellos internacionalmente con grandes resultados; pero teniendo en cuenta que ninguno cumple con las necesidades de MPPRE, ya que contienen términos, clasificaciones, automatizan procesos que no son los mismos que los que allí se desarrollan, no conciben el uso del framework de integración definido en el ambiente de desarrollo y ninguno se encarga de la emisión de los pasaportes.
Estas conclusiones guían la investigación a la obtención de una propuesta de solución que mejore las características que dificultan el uso de estas aplicaciones.

1.4 MIGRACIÓN CUBANA A SOFTWARE LIBRE

Según la Fundación para el software libre[6], el software libre se refiere a cuatro libertades de los usuarios del software: ejecutar el programa para cualquier propósito, redistribuir copias, estudiar cómo trabaja el programa, cambiarlo para lo que se quiera y publicar sus mejoras y versiones modificadas en general, para que se beneficie toda la comunidad.

[3] Modelo de Solicitud de Trámites.

[4] Modelo de solicitud de visa.

[5] Modelo de solicitud de Pasaporte.

[6] Organización creada en octubre de 1985 por Richard Stallman, dedica a eliminar las restricciones sobre la copia, redistribución, entendimiento, y modificación de programas de computadoras.

Sin duda alguna, el uso del software libre es sustentable en Cuba a partir de las ventajas y libertades que ofrece con respecto al software propietario. Por esto, su aplicación como plataforma informática de trabajo adquiere una relevante significación.

En el Consejo de Ministros del 2008, se acordó que la isla tenía que migrar al software libre, que este debía ser un proceso continuo y organizado, teniendo en cuenta las características de cada organización.

El organismo de la Aduana General de la República de Cuba fue el pionero en comenzar su migración hacia software libre en el 2005. En la actualidad se han sumado otras organizaciones, entre ellas los Ministerios de Informática y Comunicaciones (MIC), Educación Superior (MES) y Cultura (MINCULT), así como la Empresa de Telecomunicaciones de Cuba (ETECSA). Es la Universidad de las Ciencias Informáticas (UCI), la rectora en la estrategia cubana para la migración.

En el IV Taller Internacional de Software Libre celebrado en el 2008, se presentó la Guía Cubana para el cambio a código abierto. Dicho documento será el rector mediante el cual las organizaciones y empresas desarrollarán su plan de migración.

Al frente del Grupo Nacional para la Migración a Software Libre se encuentra Héctor Rodríguez, decano de la Facultad 10 de la UCI, quien explica las razones que hacen impostergable el cambio:

"En nuestro país debemos emigrar por tres razones. Una de ellas es la de independencia, pues ningún país soberano debe estar basado en el uso de una herramienta tecnológica que responda a un monopolio, en este caso el de Microsoft, y mucho menos nosotros que somos un país sometido a un férreo bloqueo, de más de 59 años. En cambio el software libre (SL) está basado en la igualdad entre los pueblos, su filosofía es la de compartir el conocimiento y esta es la principal razón, la independencia".

"En segundo lugar, por razones de seguridad. No es para nada confiable basar la informatización del país en un software cerrado, como ocurre hasta ahora ¿Qué garantías se tiene de que los programas hagan únicamente lo que se espera de ellos? ¿Qué garantías tenemos de no construirlos nosotros de que por las famosas puertas traseras se pueden colar para leer nuestra información? ¿Cómo tener la seguridad de que no hay programas ocultos que menoscaben nuestra privacidad?"

Y la tercera razón es económica. Al ser excesivo el costo de las licencias de los sistemas operativos cerrados, y si le sumas el número de PC que lo usan, el gasto sería inmensamente grande. Es un lujo que Cuba no podría darse. Por ejemplo, en la UCI, donde existen seis edificios docentes, el costo superaría

los cinco millones de dólares al año en una sola de esas instalaciones por concepto de licencias"(CUBAVISIÓN, 2009).

En estos momentos la UCI es la que ostenta mayor avance, con toda una facultad que utiliza código abierto y con un área productiva, donde se potencia el desarrollo de la industria cubana del software.

"El 85 por ciento de la exportación está basada en estos sistemas con excelentes resultados y satisfacción de su principal cliente: instituciones venezolanas, país que ha resuelto también utilizar el software libre de manera oficial y que al igual que Cuba ha escogido este camino de independencia en las tecnológicas de información"(CUBAVISIÓN, 2009).

La UCI, como centro de producción de software, juega un papel primordial en la informatización del país; ya sea por la idea de construir un modelo de ciudad digital, como por el número creciente de proyectos que desde la universidad contribuyen a la informatización de los distintos sectores de la isla. En la misma se ha apostado por el uso del software libre y ello requiere construir un software de calidad que cumpla con todos los requerimientos de los usuarios, además de estar en correspondencia con las tendencias de desarrollo de las aplicaciones actuales a nivel internacional.
Es por ello que el Centro de Identificación y Seguridad Digital ha abierto una nueva línea de desarrollo sobre software libre, línea por la cual se guía la presente investigación, utilizando herramientas de código abierto y libres de costo, de esta forma se reducen costos y se garantiza que todo sistema sea multiplataforma, pudiendo modificar el código fuente de acuerdo a las necesidades de cada software.

1.5 HERRAMIENTAS UTILIZADAS EN EL PROCESO DE DESARROLLO

1.5.1 JAVA COMO LENGUAJE DE PROGRAMACIÓN

Java es un lenguaje orientado a objetos, basa mucho su sintaxis en la de los lenguajes de programación C y C++, pero presenta un modelo de objetos mucho más simple que estos. Debido a su característica de ser orientado a objeto soporta los tres pilares propios del paradigma de la orientación a objetos, que son: el encapsulamiento, herencia y polimorfismo.
Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma, presentando las siguientes ventajas:

- Reduce en un 50% los errores más comunes de programación.
- Reducción del tiempo de desarrollo de aplicaciones.
- Las aplicaciones desarrolladas sobre Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus, además es interpretado y dinámico.

1.5.2 JAVA CON EL FRAMEWORK SPRING 2.5.5

Spring es un framework de código abierto, popular y ampliamente utilizado, que ayuda a los desarrolladores a crear aplicaciones de alta calidad de forma mucho más rápida. Spring provee una programación consistente y un modelo de configuración que es bien entendido y utilizado por millones de desarrolladores de todo el mundo. A diferencia de la tradicional plataforma de desarrollo Java EE, Spring ofrece una amplia gama de capacidades para la creación e integración de aplicaciones empresariales desarrolladas en Java (SPRINGSOURCE).

Spring provee una manera consistente de manejar objetos de negocio a través de un contenedor de inversión de control (IoC) como bloque básico, el cual se encargará de inicializar dichos objetos en el código. Teniendo esto como principal potencialidad, Spring fomenta las buenas prácticas de programar orientado hacia interfaces más que a clases, si a esto se le agrega que en la mayoría de los casos es no-intrusivo, o sea, que no fuerza a modificar el código de la aplicación para beneficio de su uso, entonces hace que se le considere un estándar de facto o framework mundialmente adoptado para a partir del cual construir aplicaciones. En otras palabras se está ante “un framework que nos permitirá desarrollar una aplicación robusta, segura y fiable haciendo nuestro código más reusable, manejable y portable” (SPRINGSOURCE).

1.5.3 JAVA CON EL framework HIBERNATE 3

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML), que permiten establecer estas relaciones además de diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se podrán generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySQL, entre otros.

"Hibernate presenta grandes ventajas en cuanto a sus prestaciones y flexibilidad al realizar la correspondencia entre tablas de la base de datos y las clases que representan estas tablas. Soporta diversos tipos de asociaciones, se pueden establecer asociaciones de uno a muchos, de uno a uno, de muchos a muchos y hasta asociaciones de herencia. Unos de los atractivos al utilizar Hibernate es que cuenta con herramientas que permiten generar automáticamente los archivos de mapeo y las clases persistentes a partir de un esquema de base de datos existente, eliminando el proceso de crearlos manualmente por parte del programador. Este framework presenta su propio lenguaje de consulta, el HQL. Las consultas construidas con este lenguaje se realizan en base a las clases y sus atributos, con el uso de este lenguaje se establecen gran portabilidad hacia los distintos SGBDR[7], ya que a partir de este lenguaje el propio Hibernate genera el código SQL nativo del SGBDR que se utilice" (REGUEIRO, 2009).

1.5.4 GESTOR DE BASE DE DATOS. POSTGRESSQL 8.3

PostgreSQL es un poderoso gestor de base de datos relacional de código abierto. Cuenta con más de quince años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación en la comunidad de desarrollo por su confiabilidad, integridad y corrección de datos. Funciona sobre los principales sistemas operativos Linux, UNIX y Windows (*PostgreSQL*).

Entre sus ventajas se encuentran:

- Ejecuta procedimientos almacenados en más de una docena de lenguajes de programación, como Java, Perl, Python, C, C + +, entre otros.
- Su biblioteca de funciones tiene cientos de funciones integradas que van desde matemáticas básicas hasta las operaciones de cadena a la criptografía.
- Alta concurrencia: Mediante un sistema denominado *MVCC* (Acceso concurrente multiversión, por sus siglas en inglés) permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Provee soporte para números de precisión arbitraria, texto de extensión ilimitada, figuras geométricas, direcciones IP, direcciones MAC y arreglos. Además los usuarios pueden crear sus propios tipos de datos.

[7] Sistemas gestores de bases de datos relacionales

1.6 METODOLOGÍA. LENGUAJE DE MODELADO. HERRAMIENTA CASE

La ingeniería de software es una tecnología multicapa en la que se pueden identificar los métodos (indican cómo construir técnicamente un software), el proceso (fundamento de la Ingeniería de Software) y las herramientas (soporte automático o semiautomático para el proceso y los métodos) (PRESSMAN, 1979). Cuyo objetivo es realizar sistemas de software económicos, fiables y que funcionen eficientemente. El proceso de desarrollo de software es la definición de un conjunto de actividades por las cuales se rigen las personas implicadas en el proyecto, las cuales explican los pasos necesarios para la terminación del proyecto. Tiene la misión de transformar los requerimientos del usuario en un producto de software. Lo cual es posible mediante el uso de una metodología de desarrollo.

1.6.1 METODOLOGÍA DE DESARROLLO BASADO EN RASGOS. FDD

Una metodología de desarrollo es un conjunto de técnicas, procedimientos, herramientas y documentos que ayudan a los desarrolladores a la realización de un software. En la actualidad la cantidad y variedad de los procesos de desarrollo de software han aumentado en gran medida. Se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, están los métodos pesados y los métodos ligeros o ágiles. Las metodologías tradicionales, pesadas generalmente, se centran especialmente en el control del proceso, mediante una exhaustiva documentación; definiendo roles, actividades, artefactos, herramientas y notaciones para el modelado y una documentación detallada. Estas metodologías son muy efectivas y necesarias en proyectos grandes. Las metodologías ágiles, dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Con cortos documentos centrados en lo esencial.

La Metodología de Desarrollo Basado en Rasgos, FDD por sus siglas en ingles *Feature Driven Development*, es considerada una metodología ágil. Especialmente preparada para los cambios en el proyecto, ya sea en cuanto a requisitos o herramientas, por lo que debe tener una planificación flexible y abierta. Es aplicable a proyectos cortos, de menos de un año de duración y de pocos integrantes. Tiene un proceso con iteraciones relativamente cortas de no más de dos semanas aproximadamente, permitiendo un mejor monitoreo del mismo. Ayuda a contrarrestar situaciones como el exceso en el presupuesto, fallas en el programa o el hecho de entregar menos de lo deseado. FDD plantea 5 fases, estas se evidencian en la siguiente figura:

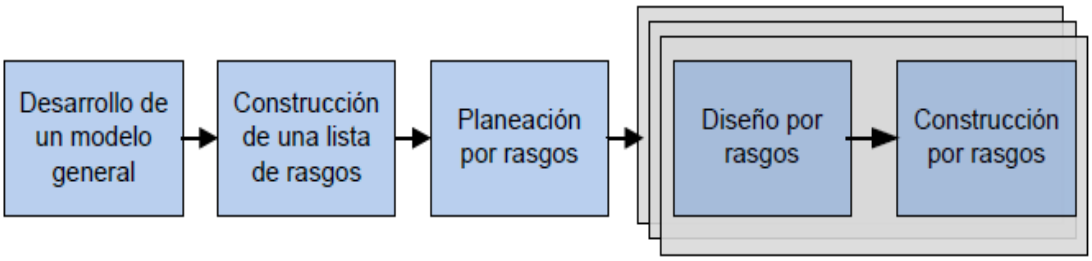


Figura 1: Fases de desarrollo de la metodología FDD

Desarrollo de un modelo general

La primera fase que plantea esta metodología es el desarrollo de un modelo global, al iniciar esta fase, se tiene una idea del contexto y visión del sistema. Se presenta un ensayo del dominio (walkthrough) en el cual los miembros del equipo son informados a través de una descripción del sistema que se quiere construir. El dominio global es dividido en diferentes áreas y se realiza un ensayo del dominio detallado para cada una de las áreas del dominio. A partir de estas descripciones del contexto y visión del sistema se realizan un modelo de objetos para cada área de dominio y simultáneamente, se construye un modelo global del sistema a partir de los modelos por áreas. (CALABRIA, 2003)

Construcción de una lista de rasgos.

Una vez concebido el modelo global se procede a identificar los rasgos o requisitos que resumen las características y comportamiento del sistema que se desea construir. El resultado de esta fase es una lista de rasgos categorizada jerárquicamente. La lista de rasgos se compone por áreas temáticas, actividades del negocio que comprenden estas áreas temáticas y por los rasgos que representan los pasos para dar cumplimiento a cada actividad del negocio. Estos rasgos son pequeñas funcionalidades útiles a los ojos del cliente y los rasgos que requieran de más de diez días se descomponen en otros más pequeños que se puedan cumplir en el tiempo máximo de dos semanas. (CALABRIA, 2003)

Planeación por rasgos.

Teniendo en mano los rasgos por los que se guiará el desarrollo, se procede a incluir la creación de un plan de alto nivel, en el que los conjuntos de rasgos se ponen en secuencia conforme a su prioridad y

dependencia. Esta planificación permite establecer que rasgos se incluyen en cada iteración de los dos últimos procesos que establece esta metodología. (CALABRIA, 2003)

Diseño por rasgos y construcción por rasgos.

Este proceso se realiza de forma iterativa. En cada iteración se selecciona un pequeño conjunto de rasgos. Se identifican las clases involucradas por cada rasgo y se diseñan e implementan estas clases a partir del conjunto de rasgos que están en la iteración. Se procede luego iterativamente hasta que se producen todos los rasgos identificados. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. El proceso iterativo está definido por los hitos: un ensayo del dominio del rasgo a desarrollar, diseño de las clases, inspección del diseño, codificación, pruebas unitarias, inspección de código y por último promoción del rasgo que se construyó. (CALABRIA, 2003)

En las iteraciones iniciales las tres primeras fases son las que ocupan más tiempo, pero a medida que avanza el proyecto entonces son las de diseño y construcción las que pasan a ocupar la mayoría del tiempo, quedando las tres primeras fases sólo en un proceso de refinamiento.

Analizando lo anteriormente documentado, se pueden resumir un conjunto de ventajas que proporciona el uso de esta metodología desarrollo:

- El trabajo, tanto de modelado como de construcción es realizado en grupos, esto garantiza que todos trabajen juntos como un equipo y que los menos experimentados aprendan de las experiencias de los demás.
- En cuanto a la relación con el cliente, al principio de las fases se define el modelo global que produce un marco en el cual puede moverse el proyecto, no siendo necesario su cambio, a no ser casos especiales que lo requieran.
- Ayuda al equipo a producir resultados tangibles de forma periódica, haciendo énfasis en la obtención de estos cada dos semanas como máximo.
- Asegura en gran parte la calidad del software entregado.
- Pone al individuo y a las interacciones del equipo de desarrollo sobre el proceso y las herramientas (BECK, 2001). En otras palabras ofrece la libertad de conformar el equipo de trabajo y que sean ellos los que definan su propio entorno de acción, no al revés.

- Antepone la importancia de hacer énfasis en el desarrollo del software por encima de conseguir una buena documentación (BECK, 2001). Se persigue que solamente la creación de los documentos en caso de que se requiera, también que sea un documento conciso y fácil de entender.

1.6.2 LENGUAJE DE MODELADO. UML

Para dar soporte a dicha metodología de software se usó UML (Lenguaje Unificado de Modelado). Se trata de un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. (IVARJACOBSON 2000). Se ha convertido en un estándar con las siguientes características:

- UML permite la creación de los diferentes modelos que ofrecen las vistas necesarias para la construcción de un software de calidad.
- Permite la compresión del sistema que se quiere realizar tanto por parte de los usuarios finales, como de los desarrolladores que implementarán la solución.
- Se puede usar para modelar distintos tipos de sistemas, ya sean sistemas de software o sistemas de hardware, y organizaciones del mundo real.
- Utilizando UML se pueden modelar sistemas bajo el paradigma orientado a objetos (OO).
- Permite especificar las decisiones de análisis y diseño, construyéndose modelos precisos y completos.
- Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, además cuenta con reglas para combinar dichos elementos.
- Es independiente del lenguaje de programación y de las características de los proyectos, ya que fue diseñado para modelar cualquier tipo de proyecto.
- Integra las mejores prácticas de los lenguajes de modelación existentes.
- A pesar de ser un lenguaje potente, es fácil de aprender y de usar.
- Permite documentar los artefactos de un proceso de desarrollo.
- Posee la capacidad de modelar toda la gama de sistemas que se necesite construir.

1.6.3 HERRAMIENTA CASE. VISUAL PARADIGM 3.4

La herramienta utilizada para el modelado del sistema es Visual Paradigm. La cual es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Con la utilización de la misma se obtienen una serie de ventajas como:

- Una rápida construcción de aplicaciones con calidad y a un menor costo.
- Ofrece capacidades de ingeniería directa e inversa.
- Es una poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
- Permite la sincronización entre el código fuente y el modelo en tiempo real.
- Cuenta con una abundante documentación, como son: tutoriales, demostraciones interactivas y proyectos UML.
- Es una herramienta colaborativa, es decir, por lo que permite múltiples usuarios trabajando sobre el mismo proyecto.
- Permite control de versiones.
- Se puede diseñar la documentación del sistema con un diseñador de plantilla.
- Se pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto.
- Genera código Java.

1.7 CONCLUSIONES

Como conclusiones parciales del capítulo se tiene que de de los sistemas investigados ninguno se ajusta a las necesidades del MPPRE, debido a que automatizan procesos que no son los que este ministerio maneja. No obstante se pueden tomar algunas que otras experiencias de ellos para el desarrollo de una solución a pesar de que algunos de estos sistemas fueron desarrollados con herramientas que no son las que se concibieron para el desarrollo del sistema, o no son compatibles con estas. Basado en esto, se propone el desarrollo de una solución utilizando herramientas libres de costo a excepción de Visual Paradigm, que es freeware, herramientas con una sólida y probada reputación a nivel internacional que ayudarán a simplificar el desarrollo de la solución. Además con la utilización de la metodología seleccionada se ganará en tiempo y solidez en el desarrollo, sobre todo ante posibles cambios que

pudiesen surgir en el mismo, dándole además mayor valor al desarrollador y a la colaboración con el cliente.

CAPÍTULO 2: CARACTERÍSTICAS DE LOS COMPONENTES

2.1 INTRODUCCIÓN

En este capítulo partiendo del estudio de los procesos del negocio, se realiza un modelo general y un modelo por cada una de las áreas o módulos existentes, exponiéndose los principales conceptos o clases y sus relaciones. Se documenta en detalle la propuesta de solución, construyéndose una lista de rasgos que representará las características o funcionalidades que los componentes deben cumplir. Una vez creada la lista de rasgos se planifica cada rasgo para la posterior fase iterativa de diseño, construcción y prueba.

2.2 FLUJO ACTUAL DEL PROCESO DE NEGOCIO INVOLUCRADO EN EL OBJETO DE ESTUDIO

Un proceso de negocio es un conjunto de tareas relacionadas de forma lógica, llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas (DEFINICIÓN.DE, 2009).

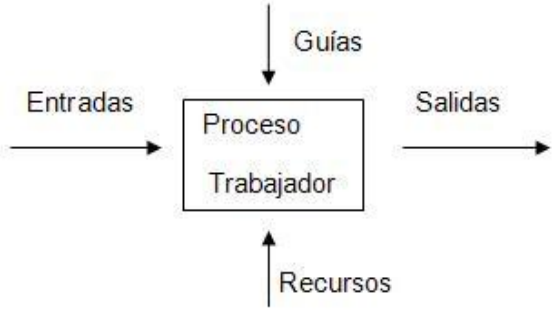


Figura 2: Descripción visual de un proceso

Entrada: Representa el material o la información que es consumida o transformada por el proceso con el objetivo de producir las salidas.

Salida: Información que, producto del proceso, se devuelve como resultado.

En el desarrollo del proceso se necesitan:

Guías: Establecen la forma en que los procesos van a desarrollar sus actividades.

Recursos: Son los recursos que el proceso necesita para poder desarrollarse de manera correcta, como son los recursos informáticos.

Trabajador: Persona encargada de llevar a cabo el proceso.

Teniendo la idea general de qué es un proceso a continuación se describe el involucrado en el campo de acción.

Proceso de emisión de pasaportes diplomáticos

El proceso comienza cuando un ente gubernamental envía a la Dirección General de Protocolo (DGP) los siguientes recaudos:

- Oficio de solicitud de pasaporte (Nota de Elaboración)
- Planilla de Pasaporte Diplomático
- 1 Fotocopia de la cédula del usuario.
- 3 fotos vigentes.

En la cual se revisan, se verifican los documentos recibidos, se elabora el Oficio de Notificación y se remite a la Dirección de Inmunidades y Privilegios para la correspondiente tramitación del pasaporte diplomático.

La Dirección de Inmunidades y Privilegios recibe, revisa los recaudos y gira las instrucciones para el área de trabajo de pasaportes de la Dirección de Inmunidades y Privilegios, donde obtenido los documentos, los funcionarios encargados proceden a registrar en el sistema los datos personales del usuario solicitante, asimismo los datos del organismo, cargo del titular y las observaciones. Una vez registrados los datos de la Planilla de Pasaporte Diplomático, se le asigna un número, fecha de expedición, fecha de vencimiento y tipo de pasaporte correspondiente. Según el tipo de pasaporte, se elige el material y se transcriben los datos del usuario (menciones, inscripciones y demás especificaciones que deban contener). Una vez culminado el proceso manual de elaboración del pasaporte se procede a enviar el lote a la Dirección de Inmunidades y Privilegios para su revisión y firma, además se le aplica un sello húmedo y personalizado, lo que autentica el documento, luego se remite al escritorio de trabajo de pasaportes para

su entrega, donde se revisa y entrega al usuario correspondiente según el chequeo de su cédula el pasaporte diplomático.

Al momento de entregar el pasaporte, se registra la firma autógrafa, nombre y cédula del titular en el cuaderno de entrega de pasaportes; en su defecto, una autorización si no es retirado por el titular, o copia del oficio de solicitud si es retirado por la oficina del ente solicitante. Se entrega el pasaporte y culmina el proceso. **Ver diagrama del proceso: Anexo 1 Fig 1**.

2.3 DOCUMENTOS ESPECÍFICOS QUE SE PROCESAN

Nota de elaboración: Documento que emite el organismo solicitante como oficio de solicitud de pasaporte.

Planilla de pasaporte diplomático: Planilla de pasaporte que se identifica por su color amarillo.

Cédula de identidad: Dato para almacenar en el cuaderno de entrega de pasaportes, identificador de usuario.

Fotos: Fotos del usuario para el cual se emitirá el pasaporte.

Oficio de notificación: Es la constancia de que la revisión de los recaudos fue exitosa y por tanto se puede continuar con el trámite.

Sello húmedo y personalizado: Indicador que refleja la disponibilidad final del pasaporte en cuestión.

Cuaderno de entrega de pasaportes: Donde se registran los datos del usuario que recogió su pasaporte, en caso de no estar presente el mismo se registra el autorizo o la copia del oficio de solicitud.

Copia del oficio de solicitud (Copia de la nota de elaboración): Documento que se presenta en caso de que sea el ente gubernamental quien se presenta a recoger el pasaporte.

Autorización: Se refiere al documento que debe presentarse en caso de que no sea el titular de pasaporte quien se presenta a recoger el mismo.

2.4 MODELO GENERAL

El modelo general consiste en la construcción de un diagrama de clases que representa los tipos de objetos más importantes dentro del dominio del problema y las relaciones entre ellos. Este diagrama de clases es de carácter estructural y luce como el tradicional diagrama entidad-relación de las bases de datos relacionales, pero presenta dos grandes diferencias ya que puede incluir relaciones de herencia, generalización y especialización, y las operaciones no reflejan conveniencias de programación sino que se describen en formas de rasgos especificando cómo debe comportarse el objeto. (PALMER, 2002)

El modelo general ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. El proceso para su elaboración consta de tres pasos:

- Identificar las clases conceptuales.
- Modelarlas en un diagrama de clases.
- Añadir relaciones.

El modelo general que se muestra a continuación define todos los conceptos del negocio: un ciudadano (que puede ser un funcionario, un personal_servicio, o un menor) tiene información biométrica (huellas dactilares, foto, firma) perteneciente a un organismo (estatal, internacional o una misión diplomática) realiza una solicitud. Si se aprueba o deniega la solicitud se envía una notificación al ciudadano informándole la respuesta, después de haber consultado previamente la agenda de citas. En caso que se apruebe la solicitud, se solicita el documento de identificación requerido (pasaporte_diplomatico, pasaporte_servicio, acreditación_diplomatica, acreditación_servicio), y comienza la realización del trámite. Cuando finaliza el trámite se personaliza el documento de identificación. En cada una de las solicitudes y trámites se emite un documento de recaudo.

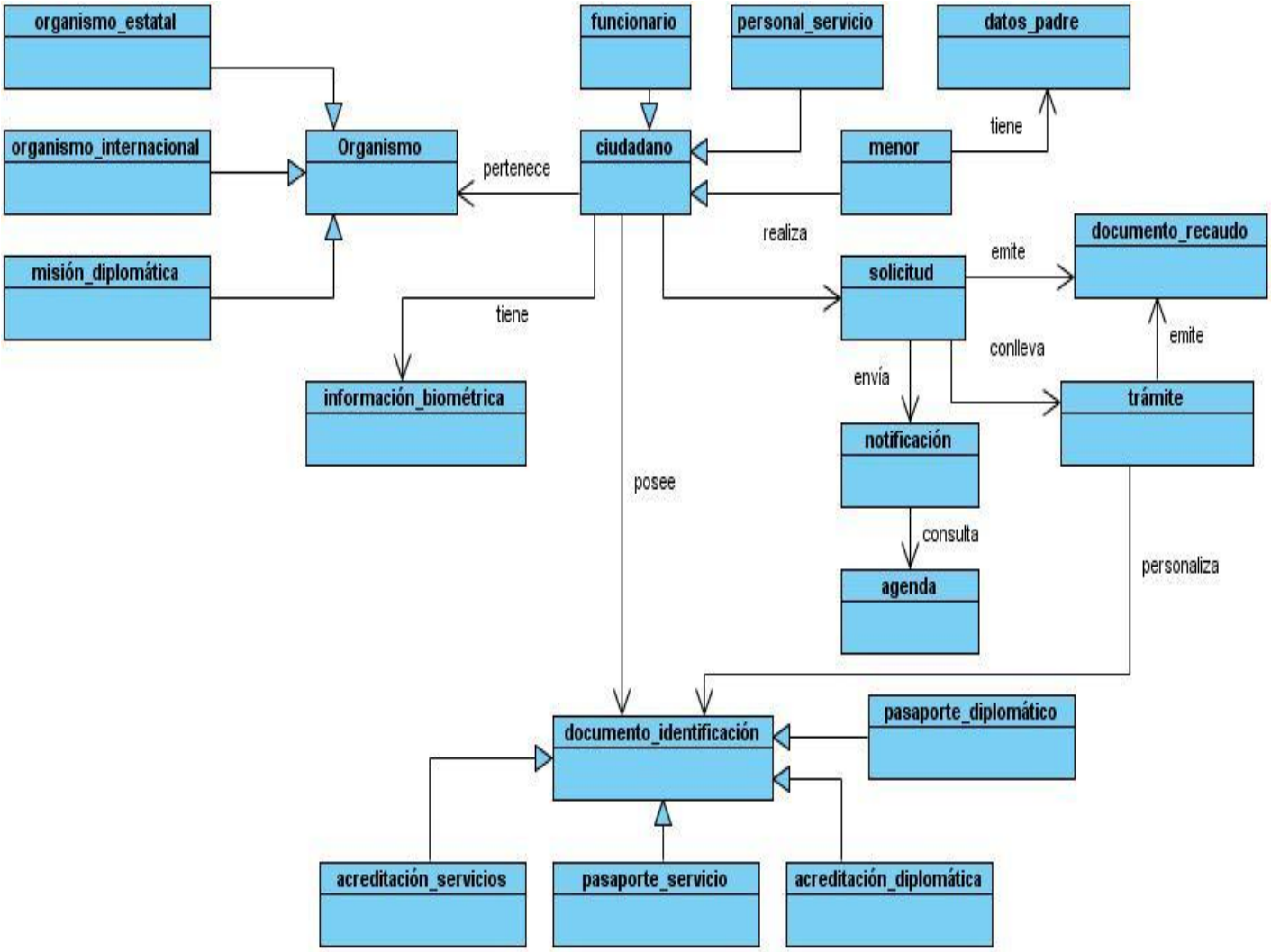


Figura 3: Modelo general

Conceptos fundamentales tratados:

Ciudadano: Es la persona que solicita o se le realiza la tramitación del pasaporte.

Información biométrica: Conjunto de información que identifica al ciudadano (huellas dactilares, firma, fotos).

Organismo: Organismo que solicita la tramitación de un pasaporte para un ciudadano.

Solicitud: Petición realizada por el ente gubernamental para la realización del pasaporte diplomático.

Notificación: Aviso que envía al organismo para cumplir un proceso determinado dentro del negocio.

Documento de identificación: Documento que se expide al finalizar el proceso de tramitación que pueden ser acreditación_diplomática, acreditación_servicios, pasaporte_servicio y pasaporte_diplomático.

Trámite: Proceso que se realiza para el enrolamiento de los datos de un ciudadano, con el fin de expedir un documento de identificación.

Documento_recaudo: Planillas oficiales que se conciben durante el proceso de emisión de pasaportes.

Como ya se especificó anteriormente; la presente investigación forma parte de un sistema mucho más complejo, orientado a procesos de negocio, los cuales para su correcto funcionamiento necesitan de un conjunto de servicios. Dichos servicios serán brindados mediante componentes. Los componentes se distribuyeron por módulos agrupados por funcionalidades para una mejor organización y concepción del sistema.

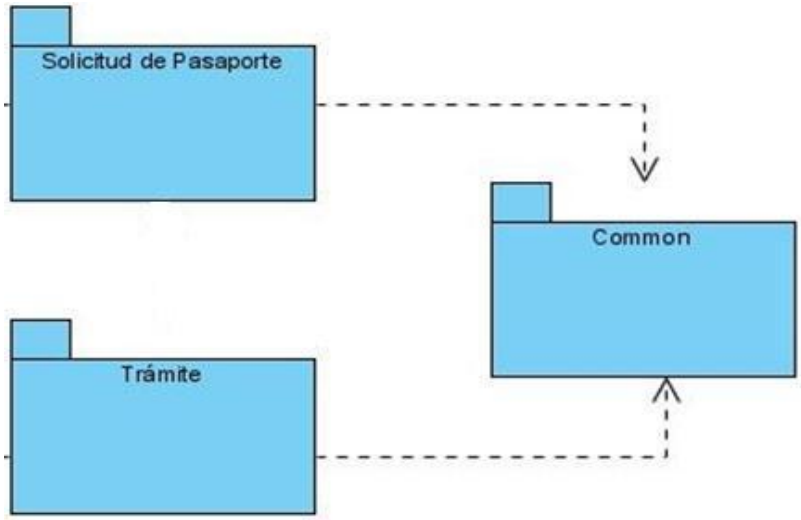


Figura 4: Diagrama de los módulos

2.4.1 MODELO GENERAL DEL MÓDULO SOLICITUD DE PASAPORTES

La metodología utilizada plantea que una vez realizado un modelo general se procede a realizar un modelo general por cada una de las áreas temáticas o módulos definidos.

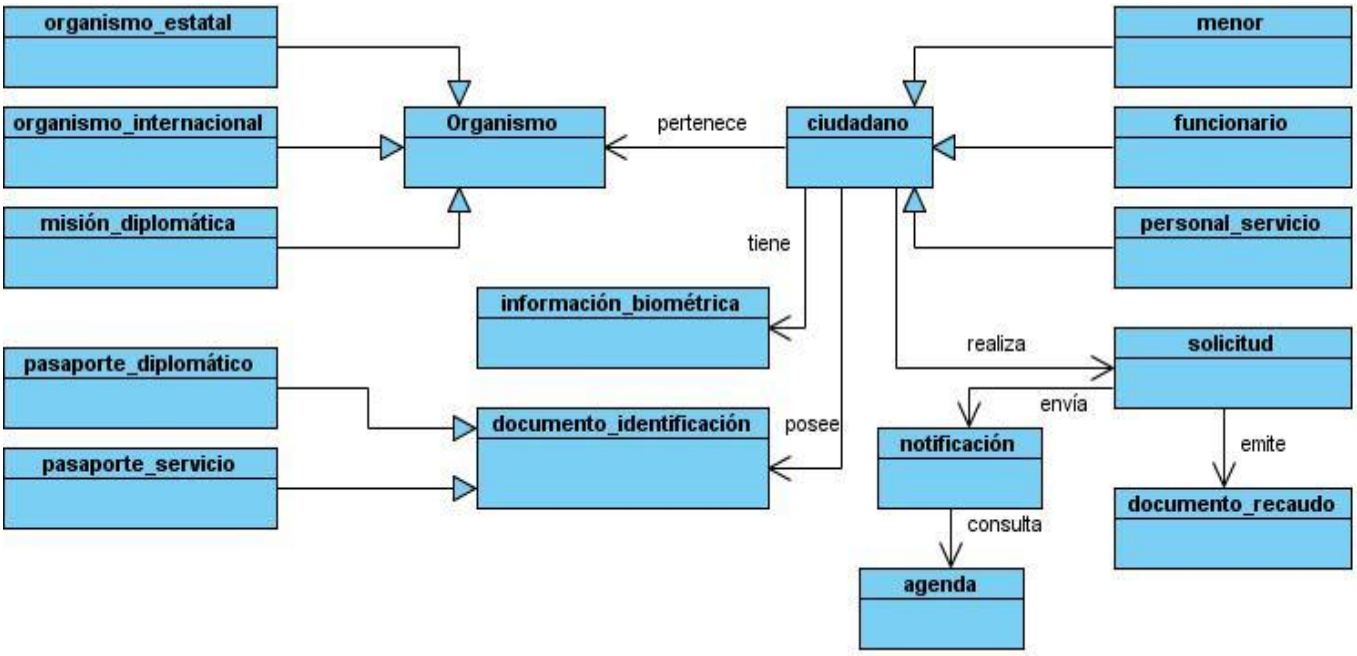


Figura 5: Modelo general del módulo Solicitud de Pasaportes

2.4.2 MODELO GENERAL DEL MÓDULO COMÚN

Figura 6: Modelo general del módulo Común

2.4.3 MODELO GENERAL DEL MÓDULO TRÁMITE

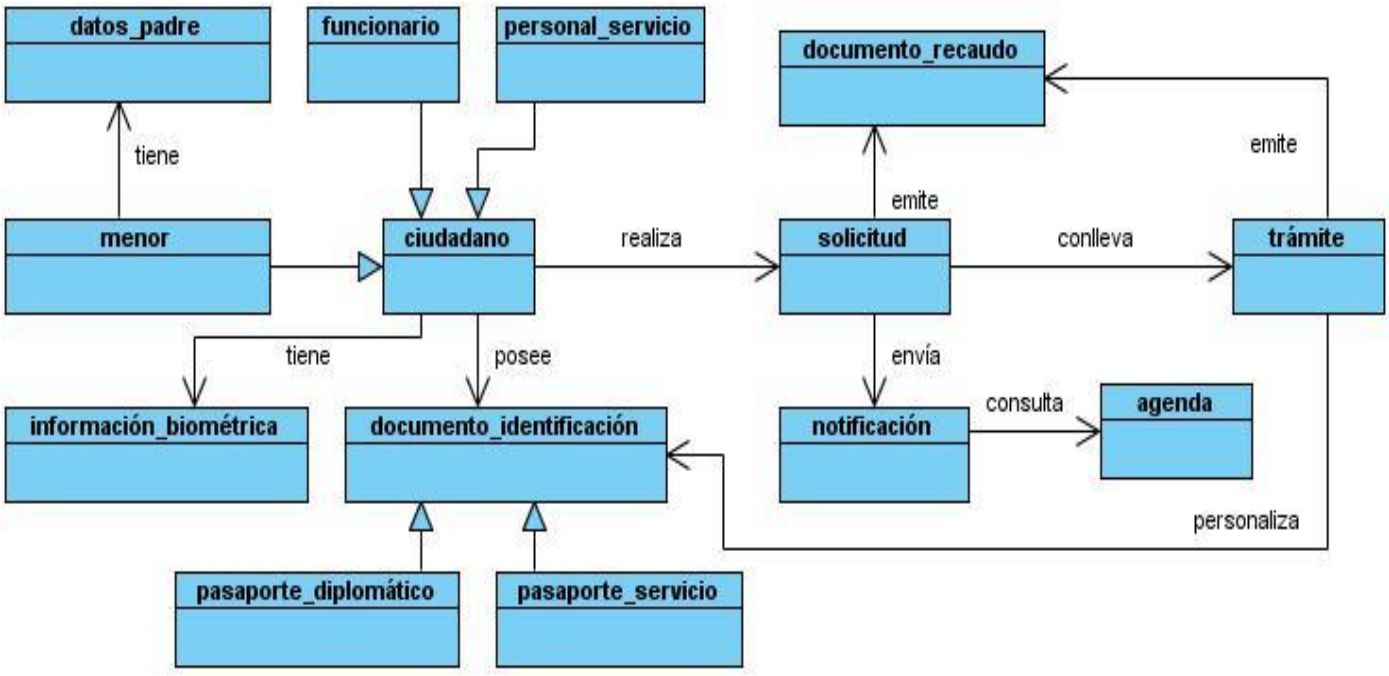


Figura 7: Modelo general del módulo Trámite

2.5 CONSTRUCCIÓN DE LA LISTA DE RASGOS

A partir del modelo general de cada uno de los módulos, se identificaron un conjunto de rasgos o características categorizadas jerárquicamente que resumen el comportamiento del sistema en general, de los cuales se extrajeron los vinculados directamente a los componentes. La lista de rasgos se descompone en áreas temáticas, actividades del negocio que comprenden estas áreas temáticas y por los rasgos. En la Tabla 2 se muestran los rasgos identificados con sus correspondientes actividades del negocio dentro de su área temática.

Lista de rasgos		
Área temática	**Actividad del negocio**	**Rasgos**
módulo Solicitud de Pasaporte	Realizar solicitud	**R 1. Crear solicitud de pasaporte.** 1.1 Mostrar formulario de solicitud de pasaporte para introducir los datos de la solicitud. 1.2 Registrar datos de la solicitud de pasaporte. **R 2. Modificar solicitud de pasaporte.** 2.1 Mostrar listado de las solicitudes del usuario autenticado en el sistema. 2.1.1 Mostrar los datos de la solicitud de pasaporte seleccionada por el usuario. 2.2 Actualizar los datos de la solicitud de pasaporte. **R 3. Cancelar solicitud de pasaporte.**
módulo Solicitud de Pasaporte	Realizar solicitud	**R 4. Crear planilla de pasaporte.** 4.1 Mostrar un formulario para

		introducir los datos de pasaporte diplomático o de servicio según seleccione el usuario. 4.3. Registrar datos de la planilla de pasaporte diplomático o de servicio. 4.4. Mostrar planilla con los datos introducidos. 4.4.1. Modificar datos de la planilla de pasaporte diplomático o de servicio.
módulo Solicitud de Pasaporte	Revisar la solicitud y verificar los datos	**R 5. Registrar estado de una solicitud.** 5.1. Mostrar listado de solicitudes realizadas en una fecha determinada con el estado pendiente a revisión. 5.2. Permitir la denegación de solicitudes. 5.2.1. Registrar causas de denegación de solicitudes. 5.2.2. Cambiar estado de la solicitud a denegada. 5.3. Permitir la aprobación de solicitudes**.** 5.3.1. Registrar fecha de cita, el número de serie de la solicitud. 5.3.2. Cambiar estado de la solicitud a aprobada.
módulo Común	Notificar aceptación de solicitud	**R 6. Emitir notificaciones.** 6.1. Enviar vía email al organismo

	Notificar denegación de solicitud	solicitante la notificación de denegación de la solicitud. 6.2. Enviar vía email la notificación de aceptación que contenga la fecha en que el ciudadano debe ir a tramitar su pasaporte (fecha de cita), el número de serie de la solicitud.
módulo Trámite	Iniciar trámite	**R 7. Registrar trámite de titular.** 7.1. Mostrar listado de nuevos titulares sin trámites definidos (nombre, número de cédula). 7.2. Listar el (los) número(s) de (los) pasaporte(s) del titular. 7.3. Registrar el formato del pasaporte a confeccionar (hoja, normal, pequeño). 7.4. Registrar el tipo de trámite (confección).
módulo Trámite	Iniciar trámite	**R 8. Mostrar estado del trámite.** 8.1. Mostrar listado de los trámites realizados con el estado de trámite correspondiente.
módulo Trámite	Iniciar trámite	**R 9. Generar planilla de control.** 9.1. Imprimir la Planilla de Control.
módulo Trámite	Supervisar trámite	**R 10. Permitir revisar los datos del trámite.** 10.1. Mostrar un listado con los pasaportes que están en proceso de supervisión. 10.1. Mostrar todos los datos de un

		pasaporte seleccionado para ser revisado.
módulo Trámite	Generar irregularidad Notificar cancelación de trámite	**R 12. Generar irregularidad.** 12.1. Cancelar Trámite. 12.1. Enviar notificación de la cancelación del trámite.
módulo Trámite	Personalizar documento	**R 13. Controlar estado del trámite durante el proceso de personalización.**
módulo Trámite	Controlar calidad	**R 14. Permitir revisión de calidad.** 14.1. Mostrar listado por lotes de los documentos personalizados.
módulo Trámite	Entregar pasaporte	**R 15. Registrar la entrega del pasaporte.** 15.1. Crear planilla de Acta de Entrega. 15.1. Imprimir el Acta de Entrega.

Tabla 2: Listado de rasgos

En el **Anexo 2** se encuentra la descripción detallada de cada uno de los rasgos anteriores.

2.6 PLANEACIÓN POR RASGOS

En las siguientes tablas se encuentran planificados los rasgos conforme a su prioridad y dependencia. A la hora de planear cada rasgo se tuvo en cuenta que dentro de una iteración no pueden existir rasgos que dependan de otros que no han sido implementados, con la excepción de que puede depender de rasgos no implementados, pero que todos estos se encuentren en la misma iteración. Otro aspecto a tener en cuenta es que se definió la realización de 6 iteraciones, planificadas de la siguiente manera:

Iteración 1: módulo Solicitud de Pasaporte.

Iteración 2: módulo Común.

Iteración 3, 4 y 5: módulo Trámite.

Iteración 6: Pruebas de integración.

Área temática: módulo Solicitud de Pasaporte (Iteración 1)

Realizar solicitud					Rasgos: 4
Rasgos	Modelo general	Diseño	Implementación	Pruebas	Despliegue
	Plan	Plan	Plan	Plan	Plan
Crear solicitud de pasaporte.	01/02/10	02/02/10	04/02/10	10/02/10	12/02/10
Modificar solicitud de pasaporte.	01/02/10	02/02/10	05/02/10	10/02/10	12/02/10
Cancelar solicitud de pasaporte.	01/02/10	02/02/10	05/02/10	10/02/10	12/02/10
Crear planilla de Pasaporte.	01/02/10	02/02/10	08/02/10	10/02/10	12/02/10

Tabla 3: Planeación de los rasgos: Realizar solicitud

Revisar la solicitud y verificar los datos					Rasgos: 1
Rasgos	Modelo general	Diseño	Implementación	Pruebas	Despliegue
	Plan	Plan	Plan	Plan	Plan
Registrar estado de una solicitud.	01/02/10	02/02/10	09/02/10	10/02/10	12/02/10

Tabla 4: Planeación de los rasgos: Revisar la solicitud y verificar los datos

Área temática: módulo Común (Iteración 2)

Notificar aceptación y denegación de solicitud					Rasgos: 1
Rasgos	Modelo general	Diseño	Implementación	Pruebas	Despliegue
	Plan	Plan	Plan	Plan	Plan

Emitir notificaciones.	15/02/10	16/02/10	18/02/10	24/02/10	26/02/10

Tabla 5: Planeación de los rasgos: Notificar aceptación y denegación de solicitud

Área temática: módulo Trámite (Iteración 3)

Iniciar trámite					**Rasgos: 1**
Rasgos	**Modelo general**	**Diseño**	**Implementación**	**Pruebas**	**Despliegue**
	Plan	**Plan**	**Plan**	**Plan**	**Plan**
Registrar Trámite de titular.	01/03/10	02/03/10	04/03/10	10/03/10	12/03/10

Tabla 6: Planeación de los rasgos: Iniciar trámite

Capturar huellas, firma y fotos					**Rasgos: 2**
Rasgos	**Modelo general**	**Diseño**	**Implementación**	**Pruebas**	**Despliegue**
	Plan	**Plan**	**Plan**	**Plan**	**Plan**
Registrar información biométrica	01/03/10	02/03/10	05/03/10	10/03/10	12/03/10
Mostrar estado del trámite.	01/03/10	02/03/10	09/03/10	10/03/10	12/03/10

Tabla 7: Planeación de los rasgos: Capturar huellas, firma y fotos

La continuación de las iteraciones se puede encontrar en el **Anexo 3.**

2.7 RASGOS NO FUNCIONALES

Existen un conjunto de propiedades o cualidades que se deben cumplir:

Eficiencia

1. Rendimiento del sistema.
   1.1. Solicitudes/día 50
2. Recursos de Hardware.
   2.1. 1G de RAM para su funcionamiento.

2.2. 120 GB de espacio en disco.

Restricciones de diseño

3. Plataforma de desarrollo.
   3.1. J2EE.
   3.2. Framework Spring para la integración de toda la aplicación.
   3.3. Framework Hibernate para el manejo y control de los datos persistentes.
4. Ambiente de desarrollo.
   4.1. Netbeans para la creación de los clientes de escritorio y eclipse para la automatización e implementación de los componentes de software.
5. Arquitectura del sistema.
   5.1. Componentes desarrollados embebidos dentro del sistema y se accede a ellos mediante interfaces.
   5.2. Los componentes utilizados desde las líneas de desarrollos del centro son externos al sistema y se establece comunicación con ellos mediante los estándares definidos de servicios web.

Requisitos de Licencia

6. Licencia de los servidores.
   6.1. El gestor de base de datos se encuentra bajo licencia Artistic License.
7. Licencia de frameworks.
   7.1. Framework Spring bajo la licencia Apache v2.0 para la integración del Sistema.

2.8 CONCLUSIONES

Se realizó las tres primeras fases secuenciales usando la metodología de desarrollo basada en rasgos, donde se logró determinar las principales clases del sistema, con su respectiva lista de rasgos. Además se determinó la realización de 6 iteraciones para la entrega total de los componentes, con fecha tope a finales del mes de abril. Se identificaron un conjunto de rasgos no funcionales en cuanto a: disponibilidad, eficiencia, restricciones de diseño y requisitos de licencias del sistema.

CAPÍTULO 3: DISEÑO DE LOS COMPONENTES

3.1 INTRODUCCIÓN

En el presente capítulo se describe cómo deben ser los componentes, a través de los diagramas de clases del diseño de cada uno de ellos. Se define el estilo arquitectónico y los patrones a utilizar. Se muestra también el modelo de datos propuesto, describiéndose cada una de sus tablas y se representa el modelo de despliegue.

3.2 DEFINICIÓN DE LA ARQUITECTURA

"La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, la confiabilidad, escalabilidad, portabilidad, y disponibilidad" (KRUCHTEN, Noviembre 1995) .

La arquitectura del Sistema de Emisión de Pasaportes Diplomáticos se encuentra dividida en tres capas: capa de presentación en la cual se encuentran los clientes que interactúan con el núcleo de la aplicación, ya sean de escritorio o web, capa de procesos de negocio donde se automatizan los procesos de negocio identificados y por último la capa de componentes encargada de implementar las funcionalidades utilizadas dentro de la capa de procesos.

Las capas están físicamente distribuidas, lo cual significa que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Esta arquitectura en capas es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse. A continuación la figura 8 muestra una vista general de la arquitectura del sistema:

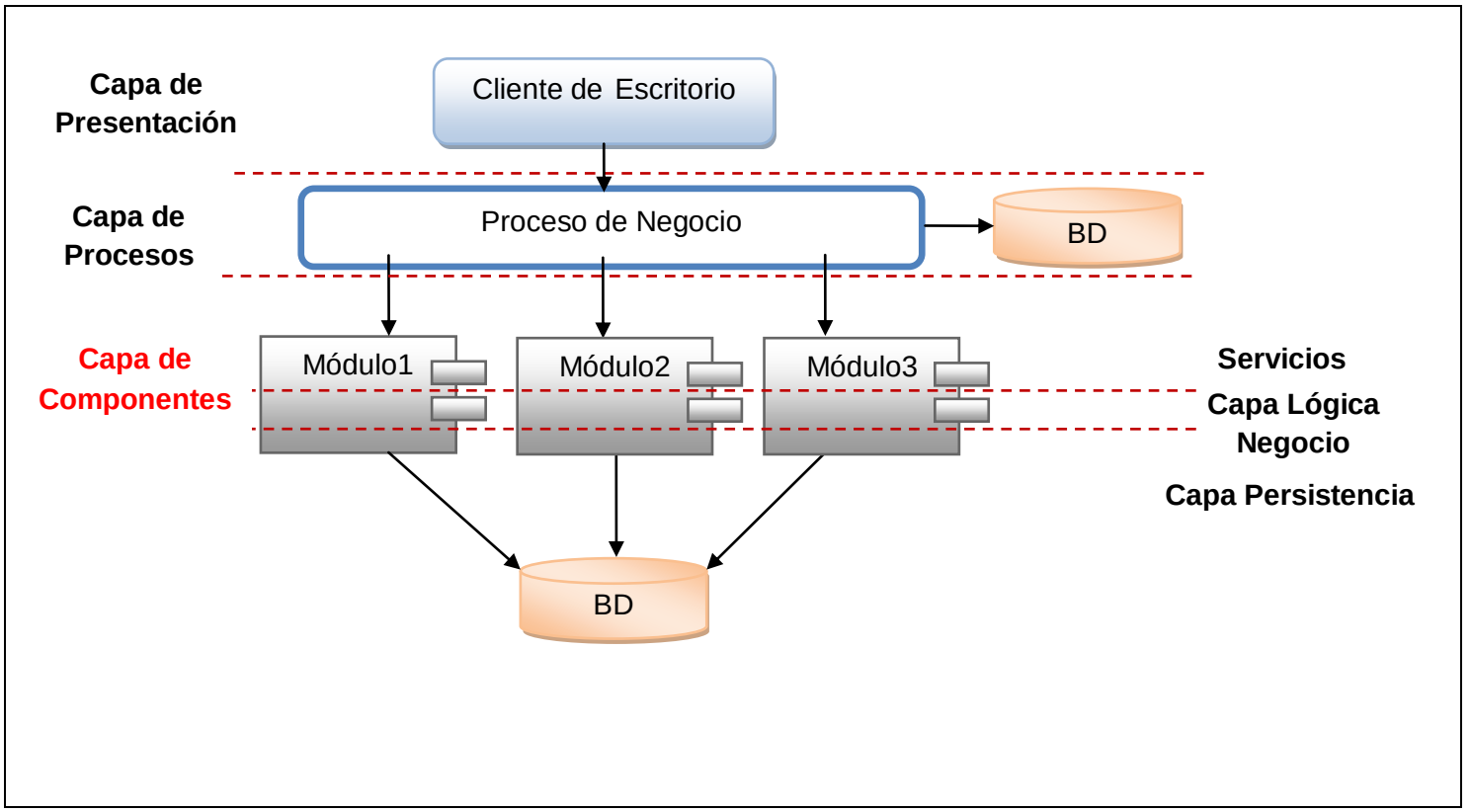


Figura 8: Vista general del sistema

Descripción por Capas:

Capa de Presentación: En esta capa se encuentran los clientes de la aplicación, se sirve de las funcionalidades expuestas por la capa inferior (capa de procesos) mediantes estándares definidos de servicios web, los clientes pueden ser tanto de escritorios como web de acuerdo a las posibilidades de conexión existentes y las necesidades del negocio.

Capa de Negocio: En esta capa se encuentran automatizados los procesos del sistema, la misma tiene la responsabilidad de brindar sus funcionalidades a la capa superior y reutiliza las expuestas por la capa de componentes, dispone de una base de datos para la información de ejecución de los procesos de negocio.

Y como última capa, la de componentes, en la cual se encuentran enmarcados los componentes a desarrollar, y tiene a su vez una arquitectura en capas:

Capa de Componentes: Esta capa por las necesidades de funcionalidades de la capa de procesos se ha subdividido en 3 capas, la capa de servicios como primera capa (superior) que ofrece las funcionalidades implementadas dentro del módulo a los procesos de negocio, la capa lógica, en la cual se implementa el negocio para el componente de acuerdo a las necesidades identificadas en el negocio y brinda sus funcionalidades a la capa de servicios, y por último la capa de persistencia encargada de controlar toda la persistencia de los datos.

Dicha arquitectura presenta principios de un paradigma orientado a servicios, o una arquitectura orientada a servicios (SOA).

Una arquitectura SOA plantea una estrategia general de organización, donde sistemas, aplicaciones y datos puedan transformarse en una red de recursos integrados, simplificada y sumamente flexible. Microsoft la define como "la Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios" (CORPORATION, Diciembre del 2006).

"Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella" (CORPORATION, Diciembre del 2006).

La arquitectura utilizada sólo contiene principios de una arquitectura orientada a servicios, porque todo el sistema se encuentra desplegado en un mismo servidor, es decir, no es un sistema distribuido, pero todas las funcionalidades que se ofrecen son expuestas mediante servicios.

3.3 PATRONES DE DISEÑO

Los patrones son un amplio repertorio de principios generales basados en la experiencia que guían la creación de un software. Cada patrón es específico a un problema recurrente en el diseño e implementación de un sistema de software. Los patrones de diseño pretenden fundamentalmente proporcionar elementos reusables en el diseño de sistemas software.

Los patrones del diseño utilizados en el desarrollo de los componentes son:

- **Patrones para asignación de responsabilidades (Grasp)**

Bajo acoplamiento: Uno de los principios para proteger al software frente al cambio es mantener bajo el acoplamiento entre clases. El acoplamiento de una clase es el conjunto de dependencias que tiene con otras clases, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios. En la solución que se ofrece, cada clase se relaciona sólo con quien lo necesita para realizar sus procedimientos (o métodos). Como ejemplo se tiene que: para realizar las acciones del proceso de solicitud, la clase **SolicitServiceImpl** recurre a las interfaces **SolicitFacade** y **OrganismFacade** solamente, pues son estas las contienen los procedimientos de los cuales se apoya para realizar los suyos.

Alta cohesión: Al asignar responsabilidades en el diseño, se buscan soluciones que asignen los métodos a las clases de forma coherente, completa y relacionada. De esta forma, se obtienen clases cohesionadas. Las ventajas son evidentes. Una clase cohesionada facilita el cambio. Al realizar un cambio en una clase muy cohesionada, todos los métodos que pueden verse afectados, toda la información que necesitamos controlar, estará a la vista, en el mismo fichero. Este patrón se relaciona con el de bajo acoplamiento, porque un diseño cohesionado tendrá un bajo acoplamiento entre clases. Ejemplos de este patrón se pueden encontrar en todas las clases de las capas de negocio y servicio, donde cada clase realiza los métodos que le competen, según su concepción y finalidad.

Experto: Se tiene en consideración a qué clase debe pertenecer un método, este principio sugiere que se asigne a la clase que más sepa del método (es decir al experto). Esto es una consecuencia del principio de alta cohesión, ya que si se asignan los métodos a las clases que tienen la información necesaria para ejecutarlos, se están creando clases altamente cohesionadas. Al igual que el patrón anterior, ejemplos de este están plasmados en todas las clases de las capas de negocio y servicio. Un ejemplo sería a clase **CitizenFacadeImpl,** la cual es la responsable de realizar todas las operaciones acerca de un ciudadano del negocio, por lo que los métodos relacionados con las operaciones básicas cobre ciudadanos estarán en la misma.

Creador: Permite decidir cuáles serán las clases creadoras de otras clases. Este patrón se tiene muy en cuenta a la hora de estructurar las clases según la arquitectura, donde cada clase de una capa superior crea su similar en la inferior (en conjunto con el framework, ya que Spring es quién maneja los objetos de

negocio), asimismo se tiene de ejemplo que la clase **NotificationServiceImpl** es quien concibe (y en conjunto con el framework) crea la clase **NotificationFacadeImpl**, la cual utilizará para sus operaciones.

3.4 MODELO DE CLASES DEL DISEÑO DE LOS COMPONENTES

3.4.1 DIAGRAMAS DE CLASES DEL DISEÑO DE LOS COMPONENTES

Los diagramas de clases del diseño muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Son utilizados para modelar principalmente la vista de diseño estática de un sistema. Esto incluye modelar el vocabulario del sistema, las colaboraciones o esquemas. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.
El lenguaje utilizado para especificar una clase de diseño es el mismo que el lenguaje de programación. Consecuentemente las operaciones, parámetros, atributos, tipos y demás son especificados utilizando la sintaxis del lenguaje de programación elegido. Las relaciones de aquellas clases de diseño implicadas con otras clases, a menudo tienen un significado directo cuando la clase es implementada. Los métodos tienen correspondencia directa con el correspondiente método en la implementación de las clases.

A continuación se muestran los diagramas de clases del diseño para los componentes más significativos dentro del sistema, para visualizar el resto ver **Anexo 4**.

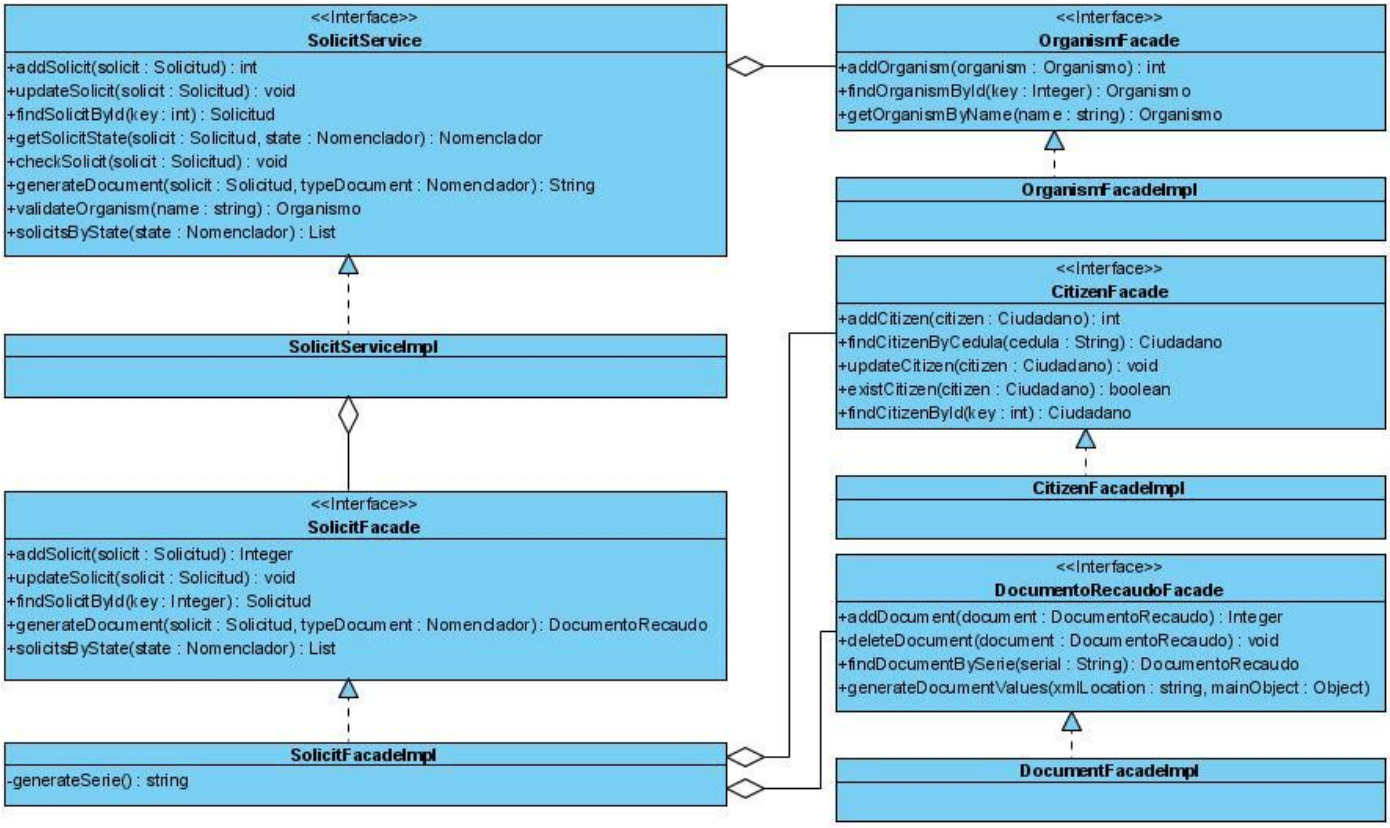


Figura 9: Diagrama de clases del diseño del componente Solicitud del módulo Solicitud de pasaporte

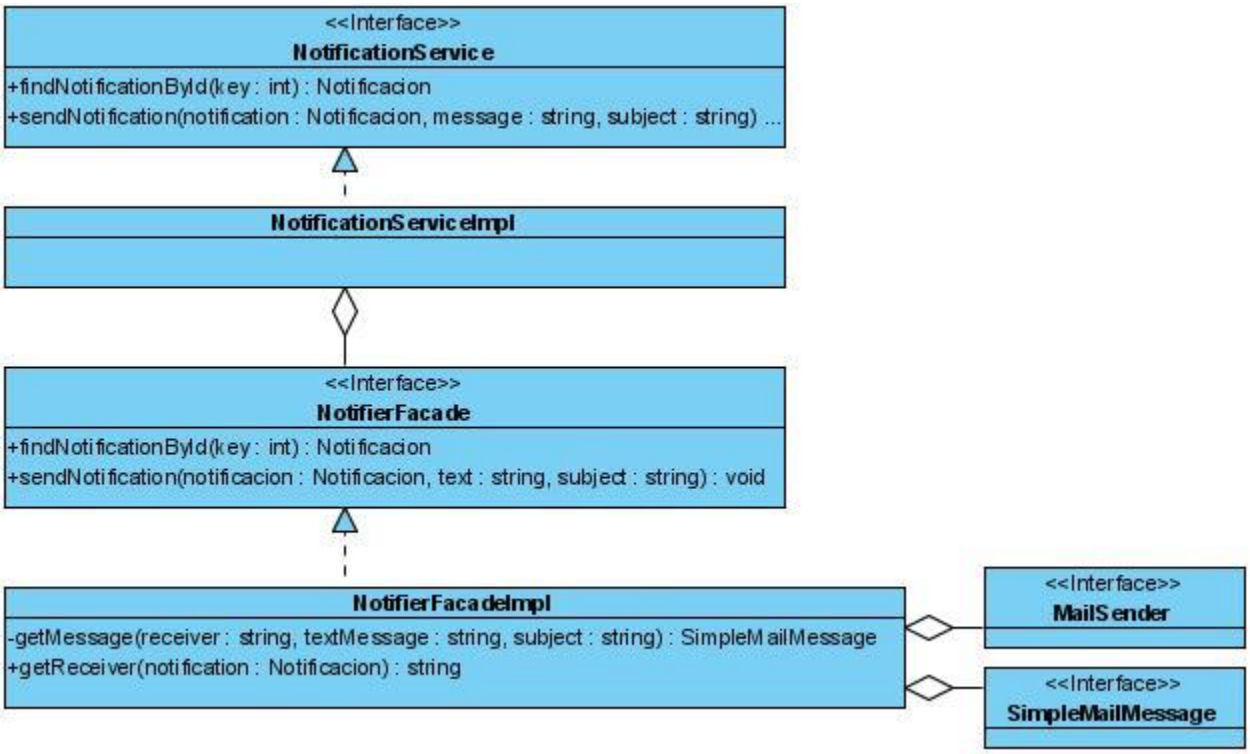


Figura 10: Diagrama de clases del diseño del componente Notificación del módulo Común

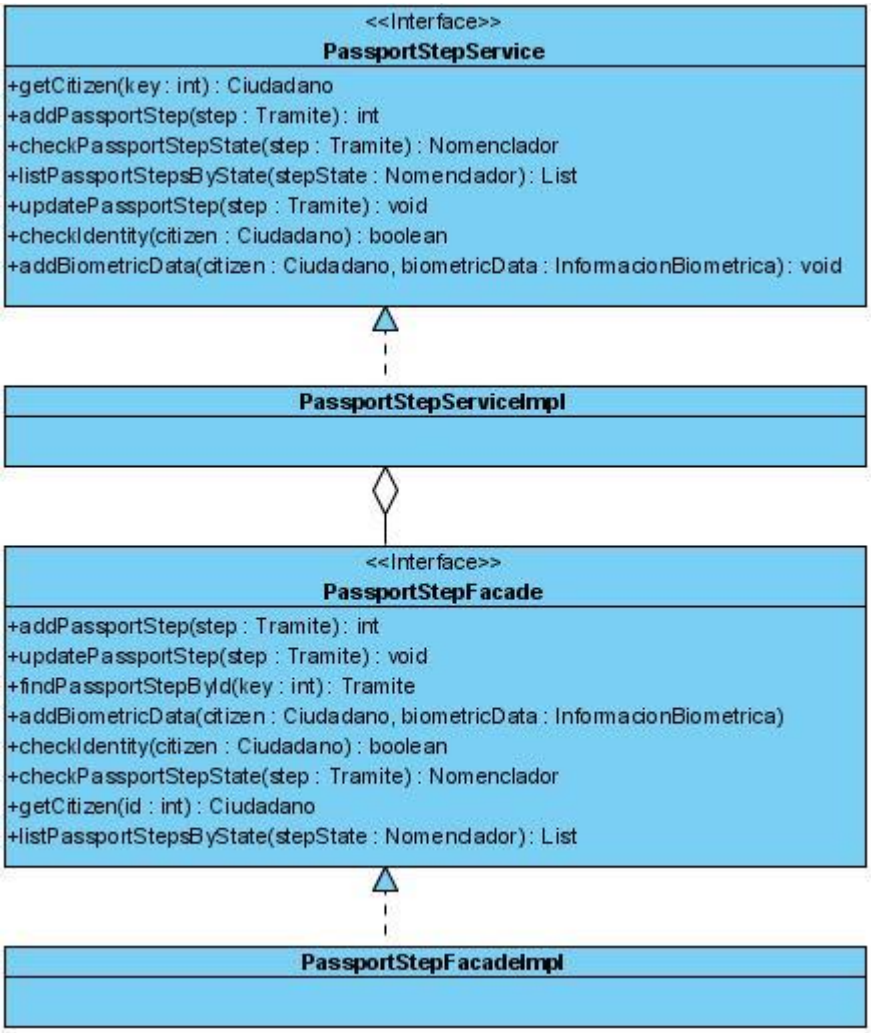


Figura 11: Diagrama de clases del diseño del componente Trámite del módulo Trámite

3.4.2 DESCRIPCIÓN DE LAS CLASES DEL DISEÑO

En la presente sección se describen cada una de las clases presentes en los diagramas de clases del diseño anteriores. Para cada una de ellas se muestra su nombre, una breve descripción de su función. Se expone además el objetivo y forma de funcionamiento de todas las operaciones contenidas por la clase, exceptuando aquellas comúnmente conocidas como “get” y “set”.

Clases contenidas en el diagrama de clases del diseño del componente solicitud perteneciente al módulo Solicitud de pasaporte:

Nombre: SolicitService	
Tipo de clase: Interfaz Está diseñada para brindar el servicio de gestionar todo lo que ocurre con la solicitud de un pasaporte en el contexto de la aplicación. En el Módulo Solicitud de pasaporte se realizan todos los pasos correspondientes a la solicitud de un pasaporte, y es este componente perteneciente a la capa de servicio, el que se encarga de brindar a los procesos todas las funcionalidades que se necesiten para las operaciones antes mencionadas.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**addSolicit (Solicitud solicit)**
Descripción:	Registra (o inserta) la solicitud que se le pasa como parámetro en la base de datos.
Nombre:	**updateSolicit (Solicitud solicit)**
Descripción:	Realiza la función de actualizar los datos de una solicitud que se le haya pasado por parámetro. Este método suele ser utilizado cuando se le cambia el estado a una solicitud.
Nombre:	**findSolicitById (int key)**
Descripción:	Realiza la función de devolver los datos de una solicitud, cuyo identificador se le haya pasado por parámetro.

Nombre:	**checkSolicit (solicit Solicit, Nomenclador state)**
Descripción:	Chequea la solicitud pasada como parámetro cambiando su estado al especificado también como parámetro del método.
Nombre:	**generateDocumentoRecaudo (Solicitud solicit, Nomenclador typeDocument)**
Descripción:	Genera (registrar en la base de datos) el documento de recaudo perteneciente a la solicitud que se le pasa como parámetro según el tipo de planilla que se quiere (el tipo de plantilla lo da el nomenclador que también se le es pasado como parámetro).
Nombre:	**validateOrganism (String name)**
Descripción:	Retorna un objeto Organismo al cual pertenece el nombre que se le es pasado como parámetro al método (considerando que le nombre es un atributo único para cada organismo existente).
Nombre:	**solicitByState (state Nomenclador)**
Descripción:	Dado un objeto de tipo Nomenclador con el identificador de un estado de solicitud determinado, devuelve una lista con todos los objetos de tipo Solicitud que se encuentran registrados en la base de datos con dicho estado.

Tabla 8: Descripción de la clase del diseño SolicitService

Nombre: SolicitFacade	
Tipo de clase: Interfaz Está concebida para la gestión y manejo de las acciones básicas sobre el objeto **Solicitud**, esta interfaz es la que interactúa directamente (o sea a través de DAOs) con la base de datos, por lo que es la que de alguna forma le da soporte a la capa de servicio correspondiente a sus funciones.	
Atributo	**Tipo**
Para cada responsabilidad: (Ver la descripción de las responsabilidades en SolicitService).	

Nombre:	
Descripción:	

Tabla 9: Descripción de la clase del diseño SolicitFacade

Nombre: CitizenFacade	
Tipo de clase: Interfaz Está pensada para la realización de las cuestiones transversales con relación al objeto de dominio **Ciudadano**. Brinda además como las demás clases de la capa de negocio independientemente del módulo al que pertenezca soporte para los servicios que se puedan ofrecer en un futuro con dicho objeto de dominio.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**addCitizen (Ciudadano citizen)**
Descripción:	Se le pasa por parámetro un objeto de tipo Ciudadano y el método se encarga de registrar sus datos en la Base de datos del sistema.
Nombre:	**findCitizenByCedula (String cedula)**
Descripción:	Se encarga de realizar una consulta a la base de datos del sistema en busca de un objeto persistente que tenga como número identificativo de cédula, la cadena de caracteres que recibe como parámetro el método.
Nombre:	**updateCitizen (Ciudadano citizen)**
Descripción:	Está pensado para modificar los datos de un Ciudadano específico (pasado como parámetro) con respecto al modelo de datos.
Nombre:	**existCitizen (Ciudadano citizen)**
Descripción:	Verifica que pasado un Ciudadano como parámetro este exista en la base de datos del sistema.

Nombre:	**findCitizenById (key int)**
Descripción:	Devuelve un objeto de tipo **Ciudadano** al cual pertenece el identificador que se especifica como parámetro al método.

Tabla 10: Descripción de la clase del diseño CitizenFacade

Nombre: DocumentoRecaudoFacade	
Tipo de clase: Interfaz Está concebida para la gestión y manejo de acciones elementales sobre los documentos de recaudo que son generados al gestionar una solicitud, esta interfaz es la encargada de garantizar su persistencia en la base de datos para un posterior uso.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**addDocument (DocumentoRecaudo document)**
Descripción:	Se le pasa por parámetro un objeto de dominio de Documento de Recaudo para posteriormente insertarlo en la base de datos en caso de que no se encuentre. Devuelve el identificador con que se registró el documento en la base de datos.
Nombre:	**deleteDocument (DocumentoRecaudo document)**
Descripción:	Elimina de la base de datos del sistema los datos del objeto DocumentoRecaudo que se le es pasado como parámetro.
Nombre:	**findDocumentBySerie (String serial)**
Descripción:	Espera por parámetro una cadena de caracteres que sería la serie que representa a un documento de recaudo, el cual se buscará en la Base de Datos y se retornará como objeto, de no existir, el método devolverá **null**.
Nombre:	**generateDocumentValues (String xmlLocation, Object mainObject)**

Descripción:	A través del objeto principal especificado como parámetro (**mainObject**), extrae todos los datos necesarios para completar los campos que se necesitan en la confección de un reporte. Cada campo a completar en el reporte debe estar relacionado con los atributos del objeto principal.

Tabla 11: Descripción de la clase del diseño DocumentoRecaudoFacade

Para ver el resto de las descripciones de las clases remitirse al **Anexo 5**.

3.4.3 ESTRUCTURA POR PAQUETES DE DISEÑO

Los diagramas de paquetes se usan para reflejar la organización de los paquetes y sus elementos. Este diagrama muestra cómo los componentes se dividen en agrupaciones lógicas, de igual manera muestra las dependencias que existen entre estas agrupaciones.

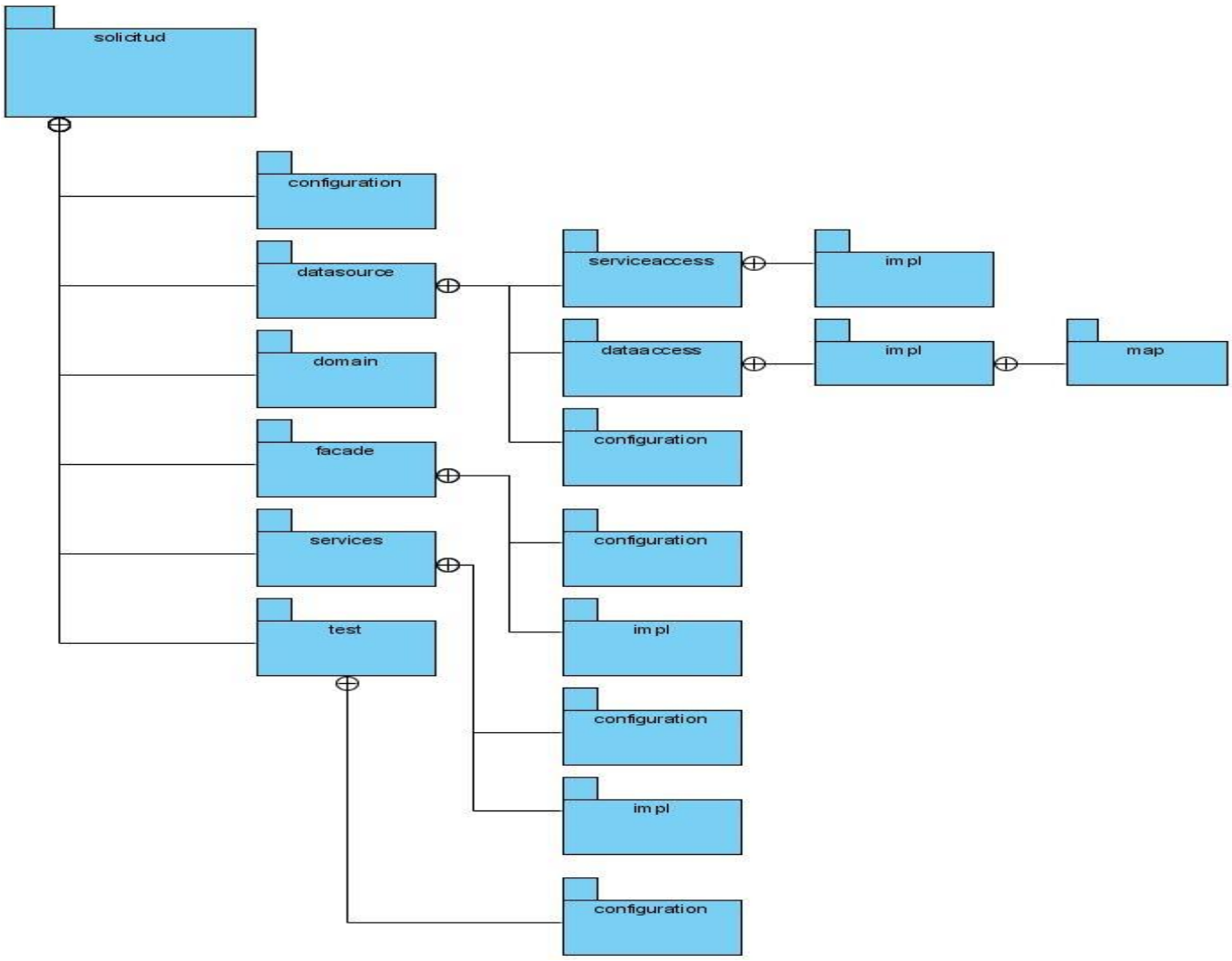


Figura 12: Diagrama por paquetes del módulo Solicitud de Pasaporte

Paquete **configuration**: Contiene los archivos de configuración de los servicios web, conexión a la base de datos y demás propiedades que necesitan los componentes de negocio para su funcionamiento.

Paquete **datasource**: Contiene todos los archivos de configuración, clases de consumo de servicios y acceso a datos.

Paquete **domain**: Contiene todas las clases de dominio o persistentes con que se trabajan en el módulo.

Paquete **facade**: Contiene todas las clases de negocio con que se trabajan en el módulo.

Paquete **services**: Contiene todas las clases que exponen los servicios que se ofrecen en el módulo.

Paquete **test**: Contiene todas las clases de pruebas para las diferentes capas de abstracción que se tienen en el módulo.

Los módulos Común y Trámite contienen un similar diagrama de paquetes, ambos se muestran en el **Anexo 6.**

3.5 Concepción de la Base de datos

Uno de los pasos fundamentales en la construcción de los componentes, es sin duda el diseño de la base de datos. En el **Anexo 7** se presenta el modelo entidad – relación necesario para el proceso de emisión de documentos de identificación de la República Bolivariana de Venezuela.

3.5.1 Descripción de las tablas de la Base de datos

La descripción de una tabla de la base de datos posee su nombre, atributos, tipo y descripción, como se muestra a continuación, pero sólo se documentan las más importantes, el resto se puede encontrar en el **Anexo 8**:

Nombre de la tabla: ciudadano		
Descripción: En esta tabla persistirán los datos de los ciudadanos con que se interactúa en el negocio.		
Atributo	**Tipo**	**Descripción**
+id_ciudadano	Int	Identificador (autoincrementado) del ciudadano.
Primer_nombre	Text	Primer nombre.
Segundo_nombre	Text	Segundo nombre (en caso de tenerlo).
Primer_apellido	Text	Primer apellido.
Segundo_apellido	Text	Segundo apellido.
Sexo	Text	Sexo.
Cedula	Text	Número identificativo de su cédula de identidad.
Fecha_nacimiento	Date	Fecha de nacimiento
observaciones	Text	Algunas observaciones de importancia que sean válidas a tener en cuenta.
Direccion	Text	Dirección donde reside.
Telefono	Text	Teléfono donde localizarlo (en

		caso de tenerlo).
Profesion	Text	Profesión en que se desempeña.
#id_organismo	Int	Identificador del organismo al que pertenece.
#nomenclador_estado_ciudadano	Int	Identificador del estado que posee el ciudadano en el negocio.
#nomenclador_nacionalidad	Int	Identificador de la nacionalidad que posee.
#nomenclador_tipo_nacionalidad	Int	Identificador del tipo de nacionalidad que posee.
#nomenclador_estado_civil	Int	Identificador del estado civil en que se encuentra.

Tabla 12: Descripción de la tabla ciudadano

Nombre de la tabla: solicitud		
Descripción: En esta tabla es donde persistirán los datos de las solicitudes que se llevarán a cabo a través del sistema.		
Atributo	**Tipo**	**Descripción**
+id_solicitud	Int	Identificador (autoincrementado) de las tuplas de la tabla.
Fecha_solicitud	Date	Fecha en la que se realizó la solicitud de pasaporte.
Observacion	Text	Alguna observación importante que deba ser guardada.
#ciudadano_solicitante	Int	Identificador del ciudadano que hace la solicitud (llamado “ente” en el contexto del negocio).
#ciudadano_receptor	Int	Identificador del ciudadano a quien se le hace la solicitud (llamado “titular” en el contexto del negocio).
#nomenclador_tipo_solicitud	Int	Identificador del tipo de solicitud que se hizo.
#nomenclador_estado_solicitud	Int	Identificador del estado en que está la solicitud.

Tabla 13: Descripción de la tabla solicitud

Nombre de la tabla: notificacion		
Descripción: En esta tabla es donde persistirán los datos de las notificaciones que envíe el sistema.		
Atributo	**Tipo**	**Descripción**
+id_notificacion	Int	Identificador (autoincrementado) de las tuplas de la tabla.
#id_cita	Int	Identificador de la cita (agenda) que se planificó en la notificación.
#id_solicitud	Int	Identificador de la solicitud por la cual se envió la notificación.
Mensaje	Text	Texto de la notificación.

#nomenclador_tipo_notificacion	Int	Identificador del tipo de notificación que se hizo.

Tabla 14: Descripción de la tabla notificación

Nombre de la tabla: tramite		
Descripción: En esta tabla se persistirán todo los datos referentes al proceso de tramitación del pasaporte.		
Atributo	**Tipo**	**Descripción**
+id_tramite	Int	Identificador (autoincrementado) de las tuplas de la tabla.
numero_serie	Int	Número que seriado que se le asigna a cada proceso de trámite.
#id_solicitud	Int	Identificador de la solicitud por la cual se está realizando el trámite.
#nomenclador_estado_tramite	Int	Identificador del estado en que se encuentra el trámite.
#nomenclador_tipo_tramite	Int	Identificador del tipo de trámite.
#nomenclador_formato_documento	Int	Identificador del formato de documento que se va a expedir.
#id_documento_identificacion	Int	Identificador del documento de identificación que se expidió a partir del trámite.

Tabla 15: Descripción de la tabla trámite

3.6 MODELO DE DESPLIEGUE

Un modelo de despliegue representa la distribución física del sistema en términos de cómo las funcionalidades se distribuyen entre los nodos de computación sobre los que se va a instalar el sistema. Como principales características tiene:

- Cada nodo representa un recurso de computación.
- Los nodos tienen relaciones que representan los medios de comunicación que hay entre ellos.
- La funcionalidad de un nodo está dada por los componentes que se ejecutan en él.
- Representa un mapeo claro entre la arquitectura de software y la de hardware.

3.6.1 DIAGRAMA DE DESPLIEGUE

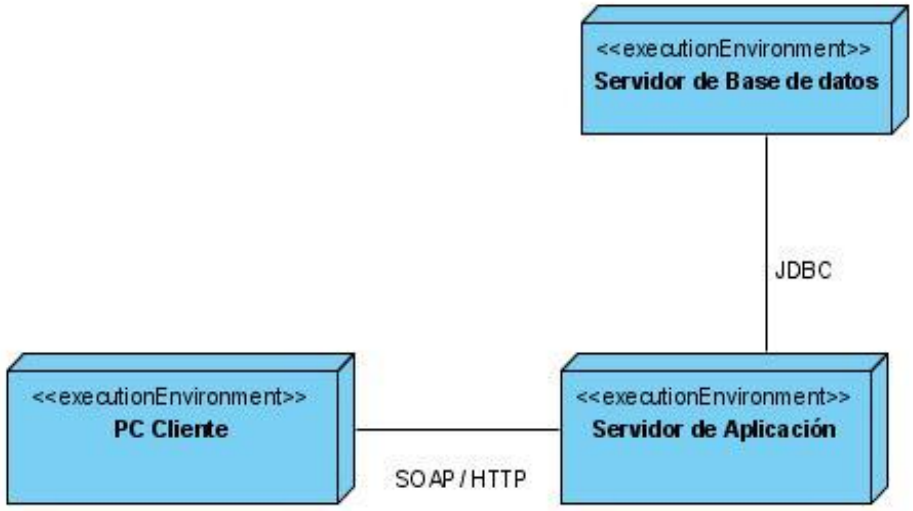


Figura 13: Diagrama de despliegue de los componentes

Un nodo representaría la PC Cliente en la cual estaría desplegado el Sistema de Emisión de Pasaportes Diplomáticos, comunicándose mediante el protocolo SOAP/HTTP al servidor de aplicaciones. El protocolo SOAP/HTTP se utiliza para la comunicación a través de servicios web.

El nodo Servidor de Aplicación representa un servidor que se utiliza en el Sistema de Emisión de Pasaportes Diplomáticos, donde se encuentran los componentes que implementan las funcionalidades de dicho sistema, cada componente desarrolla como una de sus capas la de persistencia, utilizando así el servidor de base de datos especificado.

El nodo Servidor de Base de Datos, representa un servidor utilizado para el almacenamiento de los datos y para lograr la conexión del sistema con la base de datos se utiliza JDBC como protocolo de comunicación.

El protocolo de comunicación JDBC, es la API[8] de Java que utiliza la herramienta Hibernate para lograr la persistencia de los objetos. JDBC es una especificación de un conjunto de clases y métodos

[8] Aplication Programming Interface.

de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. La aplicación de Java debe tener acceso un controlador JDBC adecuado. Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API, JDBC y la base de datos real.

3.7 CONCLUSIONES

Se utiliza para el desarrollo de los componentes una arquitectura en capas con principios de una arquitectura orientada a servicios, además se especifican cada uno de los patrones de diseño usados mediante ejemplos prácticos. Se presentó el modelo de datos del Sistema Emisión de Pasaportes Diplomáticos. En el diagrama de despliegue se tienen 3 nodos físicos: la PC Cliente y dos servidores, uno para la base de datos y otro de aplicación.

CAPÍTULO 4: CONSTRUCCIÓN Y PRUEBAS

4.1 INTRODUCCIÓN

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la de pruebas. Esta fase del desarrollo de un software es una de las que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta. En esta fase se le añade valor al producto, ya que todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade.

En el presente capítulo se documenta el diagrama de componentes realizado como artefacto que se genera en la fase de construcción. Además se define el modelo de pruebas que se llevará a cabo, especificándose cada una de las pruebas a realizar y las herramientas utilizadas, para así ir formando el programa global a medida que se comprueba cómo los distintos componentes interaccionan y se comunican libres de errores.

4.2 DIAGRAMA DE COMPONENTES

El diagrama de componentes que se muestra representa cómo el sistema de software es dividido en componentes así como las dependencias entre estos. Los componentes pueden agruparse en paquetes siguiendo un criterio lógico y con vista a simplificar la implementación. Son paquetes estereotipados en subsistemas (o módulos).

Como se definió en la arquitectura cada uno de los módulos del sistema está compuesto por capas, representadas como subpaquetes dentro de los mismos, la capa de servicios (services) y la capa de negocio (facade), donde se implementan cada uno de los componentes. En la **figura 15** se muestra el diagrama de componentes general del sistema y en la **figura 16** específicamente el del módulo Solicitud de pasaportes.

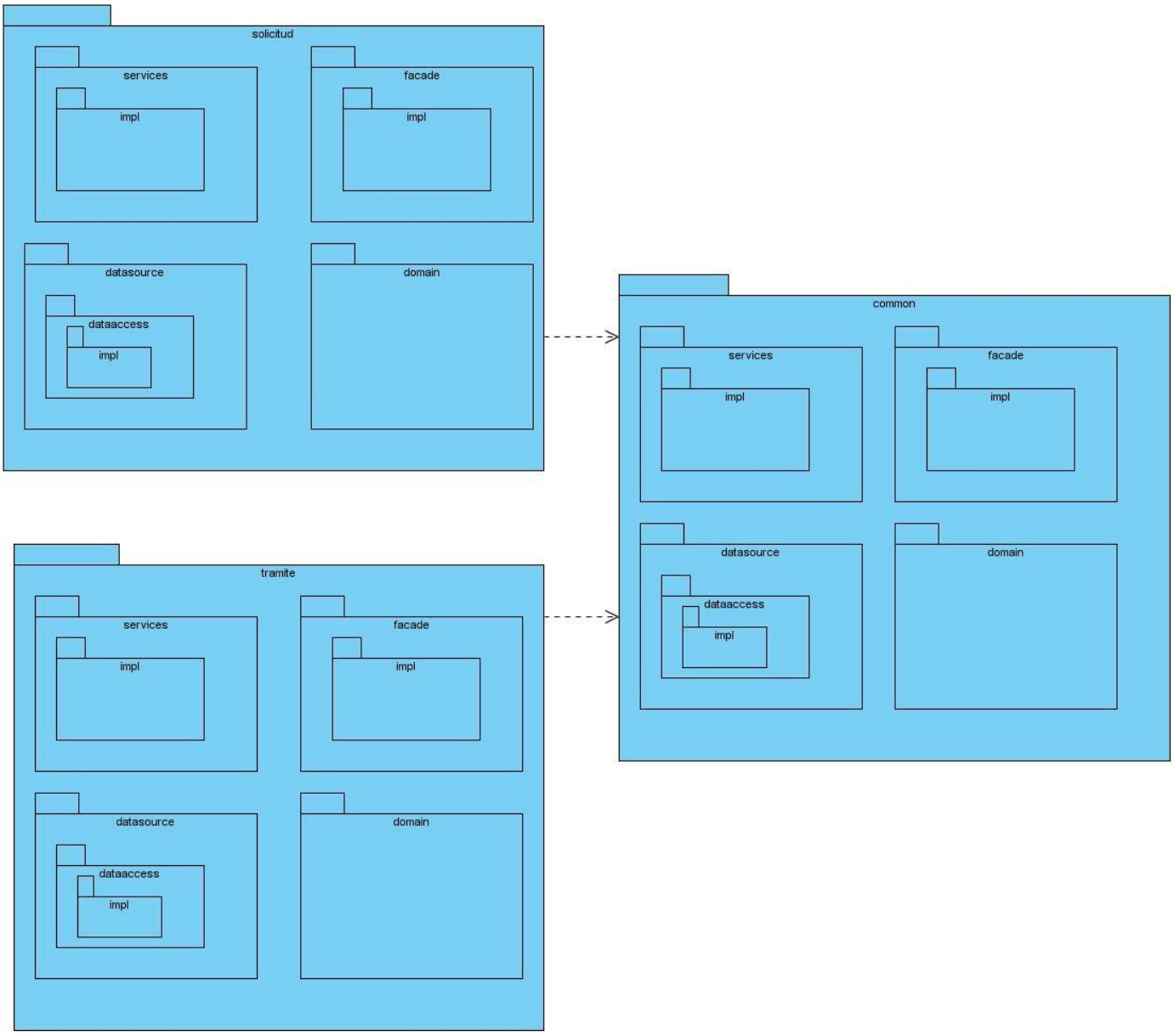


Figura 14: Diagrama de componentes general

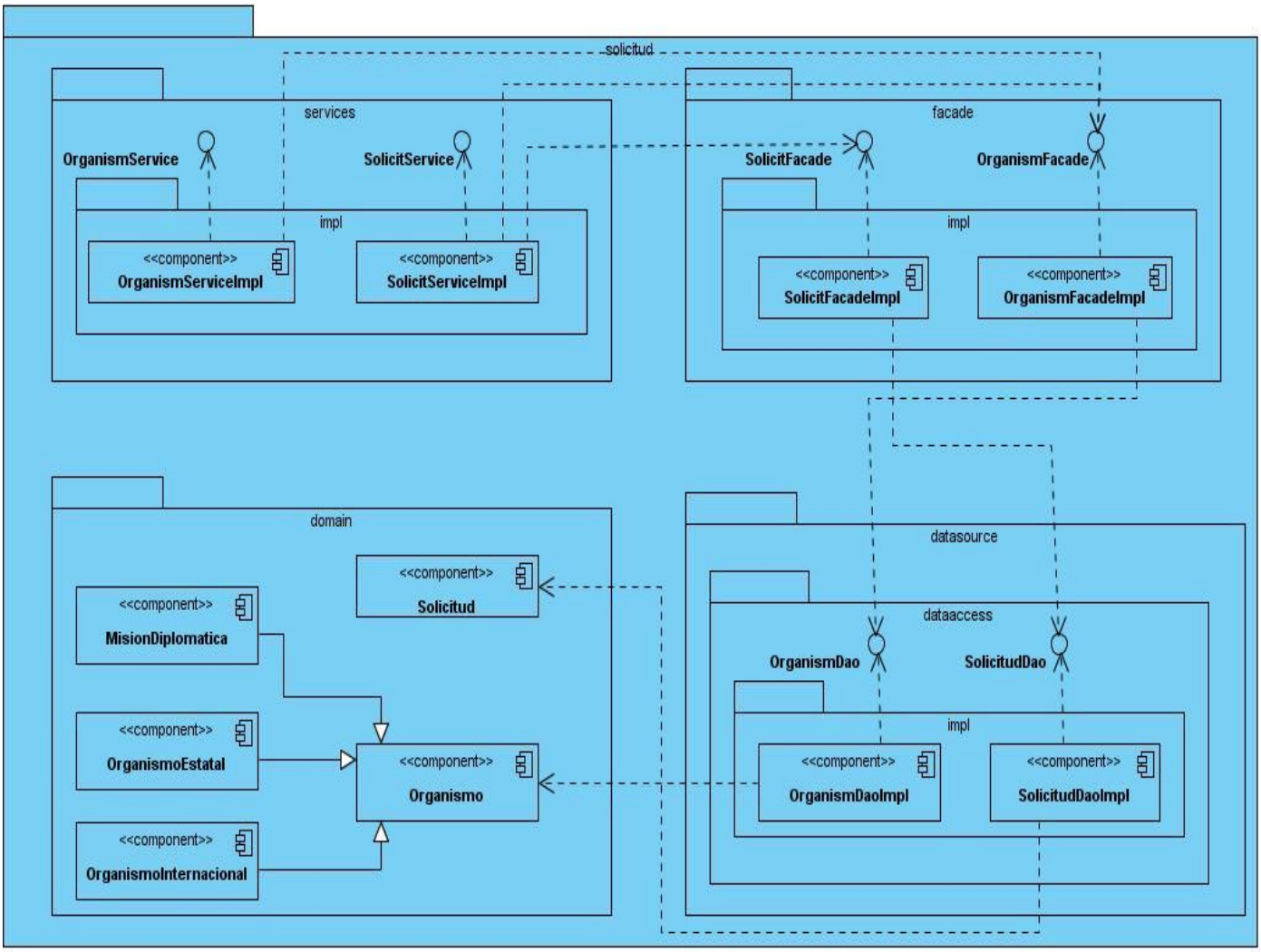


Figura 15: Diagrama de componentes del modulo Solicitud de pasaportes

Los diagramas de componentes de los módulos Común y Trámite se encuentran documentados en el **Anexo 9**.

4.3 MODELO DE PRUEBAS

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto de software. Básicamente es la fase del desarrollo de software donde se prueban las aplicaciones construidas con el objetivo de identificar posibles fallos de implementación, calidad, o usabilidad de un sistema, demostrando así la validez de la hipótesis planteada.

El modelo de pruebas que se concibió para los componentes incluye:

- Pruebas unitarias: Para asegurar que cada uno de los módulos definidos funcione correctamente por separado. Como parte de estas pruebas se aplicará el método de caja blanca.
- Pruebas de integración: Tienen como objetivo fundamental probar el software cuando los módulos individuales de software son combinados como un grupo, para probar el correcto funcionamiento de los mismos en su conjunto.
- Prueba para comprobar la funcionalidad de los servicios ofrecidos por los componentes desarrollados.

Todas las pruebas definidas serán con el apoyo de la herramienta JUnit[9], herramienta que se explica más adelante.

4.3.1 MÉTODO DE CAJA BLANCA

Las pruebas de caja blanca realizan un seguimiento del código fuente según va ejecutando los casos de prueba, de manera que se determinan concretamente las instrucciones, bloques, en los que existen errores **(ver la siguiente figura).**

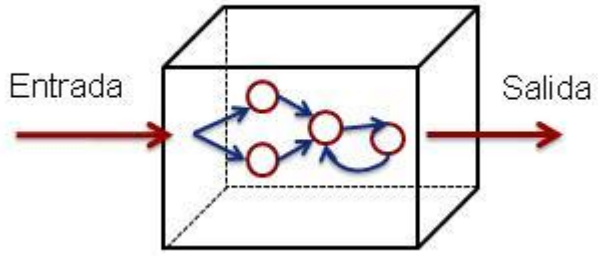


Figura 16: Método de caja blanca

[9] Conjunto de bibliotecas para hacer pruebas unitarias a aplicaciones Java.

Estas pruebas aisladas proporcionan cinco ventajas básicas:

1. **Fomentan el cambio**: Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
2. **Simplifica la integración**: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
3. **Documenta el código**: Las propias pruebas son documentación del código, ahí se puede ver cómo utilizarlo.
4. **Separación de la interfaz y la implementación**: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock[10] para simular el comportamiento de objetos complejos.
5. **Los errores están más acotados y son más fáciles de localizar**: Dado que se tienen pruebas unitarias que pueden desenmascararlos (RODRÍGUEZ, 2006).

En las pruebas de caja blanca, se definen la prioridad de cada uno de los métodos de acuerdo a la significación que tenga la funcionalidad que implementa. Una vez definidos los métodos más importantes para el funcionamiento de la aplicación, serán creados los casos de prueba asociados a éstos, definiendo los valores a los que deberá responder correctamente el sistema.

De acuerdo a la porción de código correspondiente al rasgo **Crear solicitud de pasaportes**, perteneciente a la clase SolicitFacade, del componente solicitud, módulo Solicitud de pasaportes, se le realizó la prueba de caja blanca:

```
public int addSolicit(Solicitud solicit) {                    1
        int key = -1;                                         1
        if(solicit != null)                                   2
```

[10] Objetos simulados

```
        {
                Ciudadano ente = solicit.getCiudadanoSolicitante();                    3
                Ciudadano receptor = solicit.getCiudadanoReceptor();                   3
                if(!citizenFacade.existCitizen(ente))                                  4
                {
                        int keyEnte = citizenFacade.addCitizen(ente);                  5
                        ente.setIdCiudadano(keyEnte);                                  5
                 }
                        else {
                        Ciudadano same = citizenFacade.findCitizenByCedula(ente.getCedula()); 6
                        ente.setIdCiudadano(same.getIdCiudadano());                    6
                            }
                int keyReceptor = citizenFacade.addCitizen(receptor);                  7
                receptor.setIdCiudadano(keyReceptor);                                  7
                key = solicitudDao.save(solicit);                                      7
        }
        return key;                                                                    8
}
```

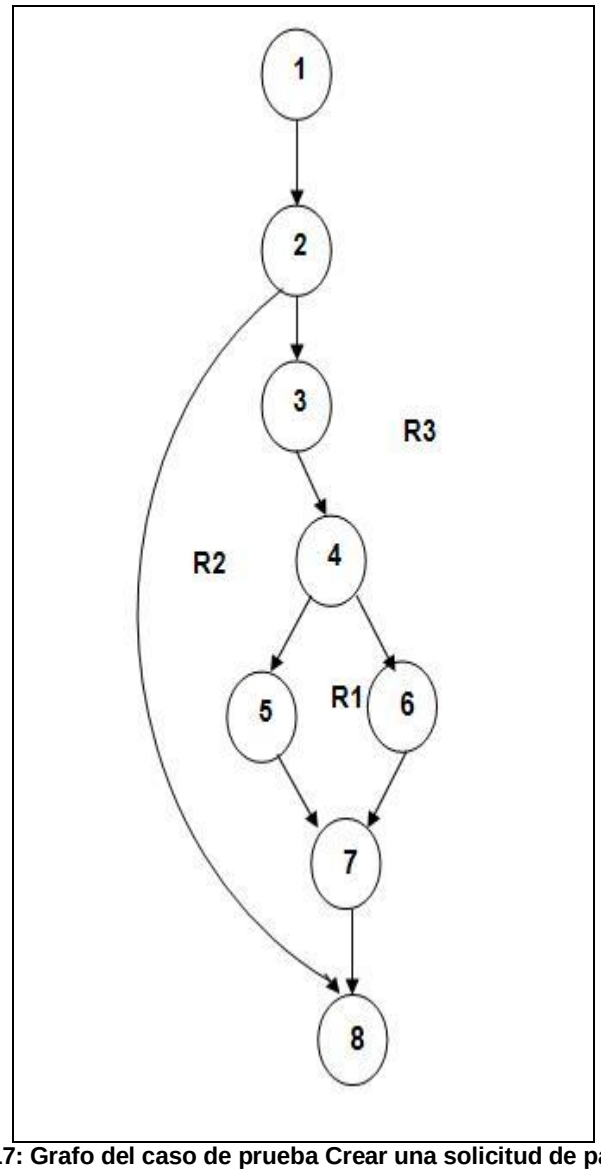


Figura 17: Grafo del caso de prueba Crear una solicitud de pasaporte

Complejidad Ciclomática:

V (G): Número de regiones del grafo

V (G) = A – N + 2

V (G) = P + 1

A: Número de aristas del grafo.

N: Número de nodos.

P: Número de nodos predicados.

V (G) = 3

Caminos: 1-2-3-4-5-7-8, 1-2-3-4-6-7-8, 1-2-7-8

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

Caso de prueba: Crear una solicitud de pasaporte.

Entrada: Recibe una solicitud válida.

Resultado: Se chequea que la solicitud sea válida, se obtienen el ente y el receptor de dicha solicitud, si no existe el ente se adiciona en la base de datos, se adiciona también el receptor y por último se registra (o inserta) la solicitud.

Condiciones: Haber pasado o no por parámetro una solicitud.

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

Caso de prueba: Crear una solicitud de pasaporte.

Entrada: Recibe una solicitud válida.

Resultado: Se chequea que la solicitud sea válida, se obtienen el ente y el receptor de dicha solicitud, si existe el ente se obtiene de la base de datos, se adiciona el receptor y por último se registra (o inserta) la solicitud.

Condiciones: Haber pasado o no por parámetro una solicitud.

Camino: 1-2-7-8

Caso de prueba: Crear una solicitud de pasaporte.

Entrada: Recibe una solicitud no válida o nula.

Resultado: Se chequea que la solicitud sea válida, como no lo es devuelve -1.

Condiciones: Haber pasado o no por parámetro una solicitud.

De acuerdo a la porción de código correspondiente al rasgo **Emitir notificaciones**, perteneciente a la clase DateFacade, del componente fecha, módulo Común, se le realizó la prueba de caja blanca:

```
public Calendar getAppointmentDate() {                    1
        Agenda diary = findAvailableDiary();              1
        if(diary != null)                                 2
        {
            Calendar result = diary.getFecha();           3
            return result;
        } else {                                          4
            throw new RuntimeException("No existen citas planificadas o todas están consumidas");
        }                                                 5
    }
```

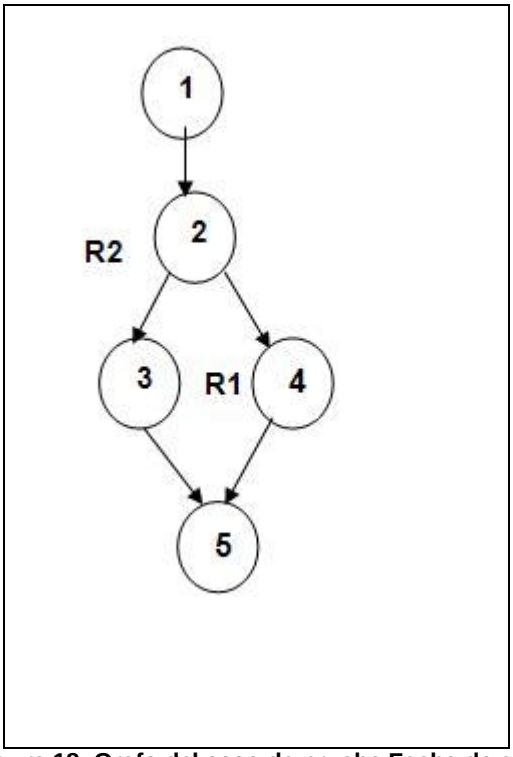


Figura 18: Grafo del caso de prueba Fecha de cita

V (G) = 2

Caminos: 1-2-3-5, 1-2-4-5

Camino: 1-2-3-5

Caso de prueba: Fecha de cita.

Entrada: Ninguna.

Resultado: Se obtienen las citas más cercanas a la fecha, si existen citas, se obtiene la fecha de la cita.

Condiciones: Que el método sea llamado.

Camino: 1-2-4-5

Caso de prueba: Fecha de cita.

Entrada: Ninguna.

Resultado: Se obtienen las citas más cercanas a la fecha, como no existen, se muestra un mensaje de error.

Condiciones: Que el método sea llamado.

Se le realizó además la prueba de caja blanca al método **Registrar trámite de titular** del módulo Tramite, presentada en el **Anexo 10**.

4.3.2 PRUEBAS DE INTEGRACIÓN

Las pruebas de integración buscan probar la combinación de las distintas partes de la aplicación, para determinar si funcionan correctamente en conjunto. Se realizan después de concluir las pruebas unitarias.
La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman a un sistema suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual.
Para realizar dicha prueba se escoge una porción de código que incluye alguna llamada a otro método perteneciente a diferente módulo, se le realiza la prueba de caja blanca y además se comprueba mediante la herramienta JUnit, demostrando así la integración exitosa entre componentes.

4.3.3 HERRAMIENTA JUNIT

Para la realización de estas pruebas se utilizó la herramienta JUnit. JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente (ESPAÑA, 2009).

En las figuras que se muestran se evidencia mediante una barra de color cuándo el método pasó satisfactoriamente la prueba y cuándo no.

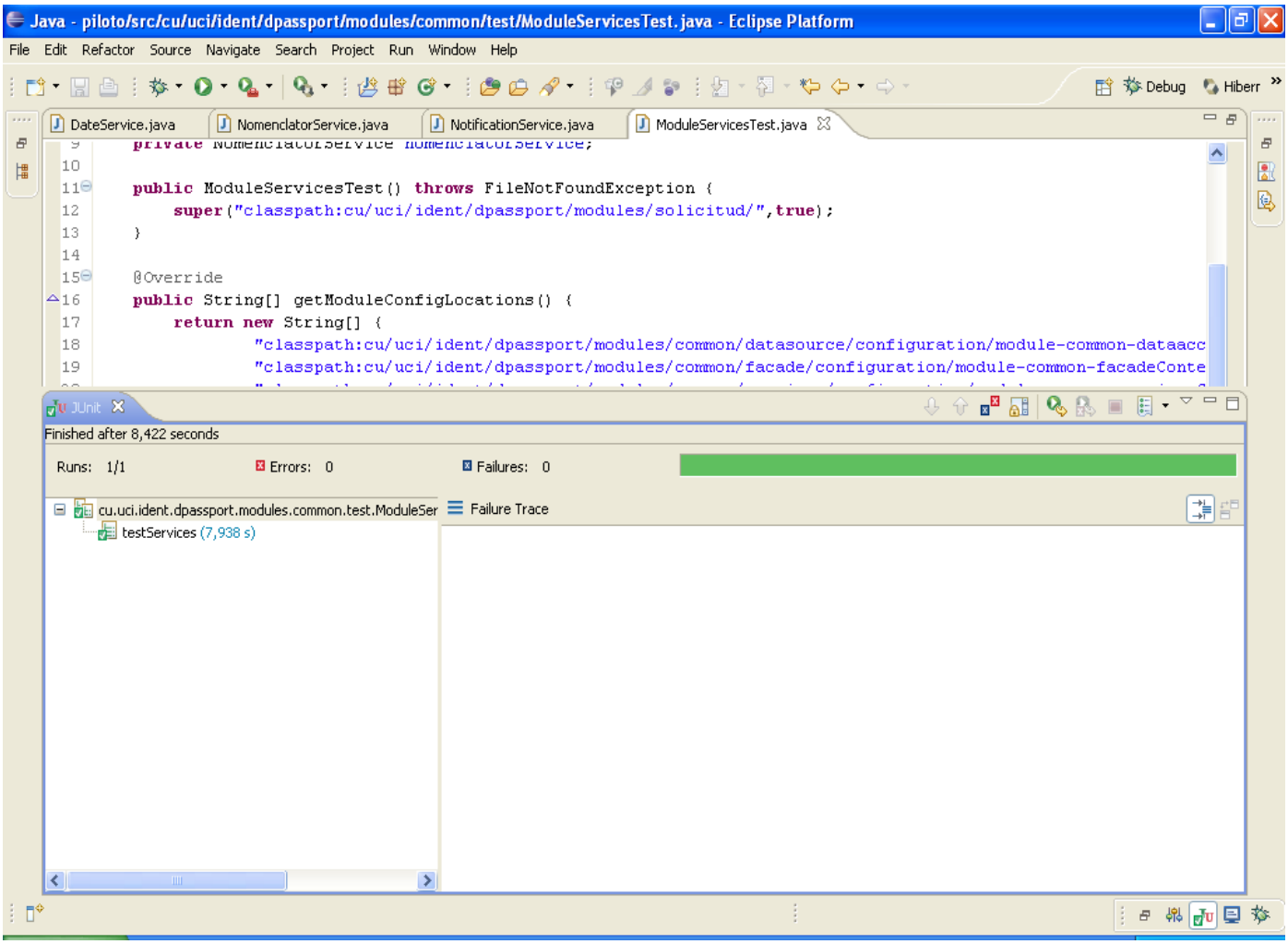


Figura 19: Prueba exitosa

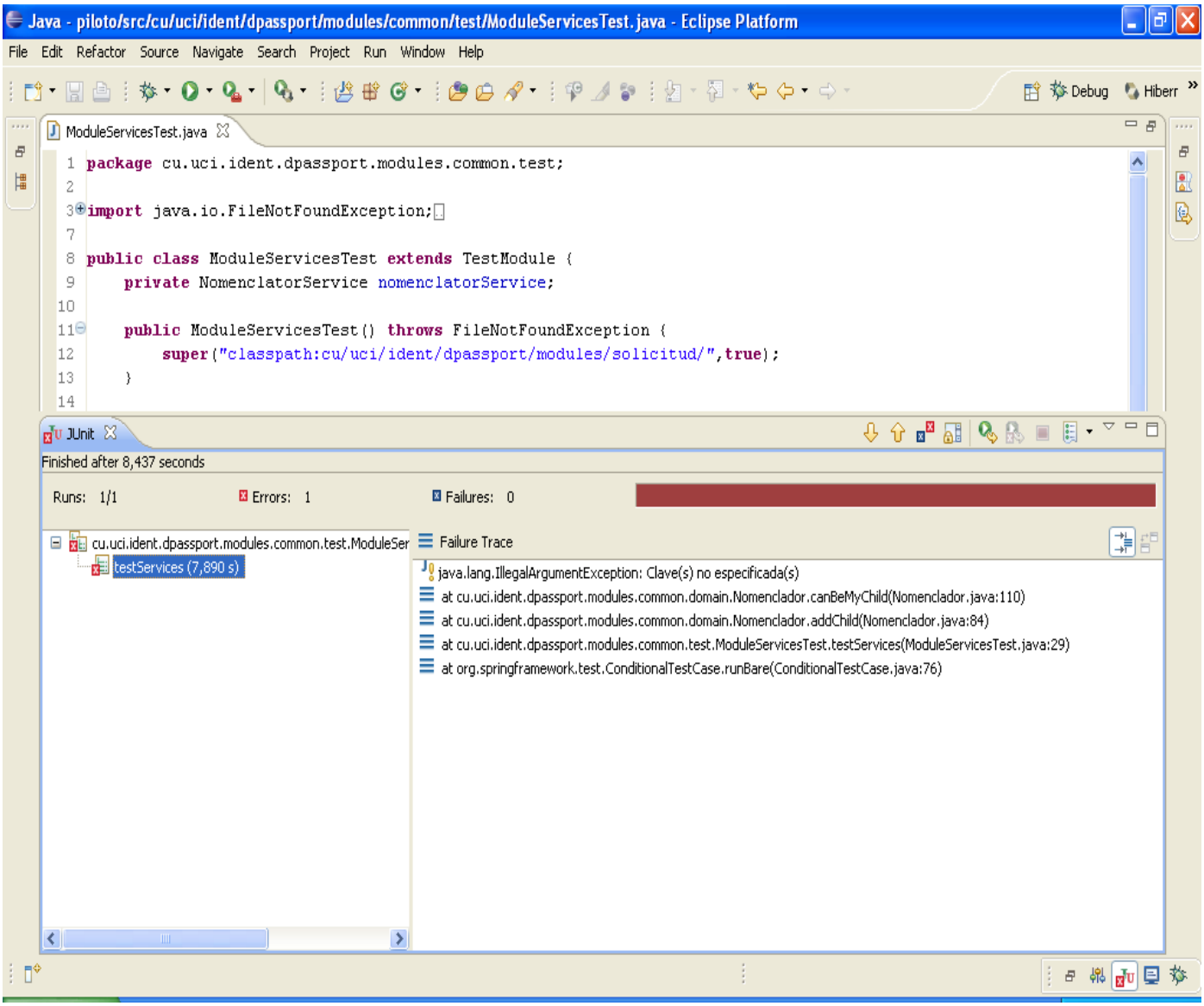


Figura 20: Prueba fallida

4.4 ESTIMACIÓN DE ESFUERZO: FACTIBILIDAD ECONÓMICA

El Modelo Constructivo de Costes o COCOMO, por su acrónimo del inglés *Constructive Cost Model*). Pertenece a la categoría de modelos de subestimaciones basados en estimaciones matemáticas. Está orientado a la magnitud del producto final, midiendo el "tamaño" del proyecto, en líneas de código principalmente. Incluye tres submodelos: básico, intermedio y detallado.

A la vez, cada submodelo también se divide en **modos** que representan el tipo de proyecto, y puede ser:

- **modo orgánico**: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio).

- **modo semilibre** o **semiencajado**: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.

- **modo rígido** o **empotrado**: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

De acuerdo a las características en particular de la investigación presentada se clasificó en el modo rígido y se calcula la estimación mediante el submodelo básico.

Modelo básico: Se utiliza para obtener una primera aproximación rápida del esfuerzo.

Estos valores son para las fórmulas:

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

- **Personas necesarias por mes para llevar adelante el proyecto (MM) = a*(KI^{b}), donde KI es un aproximado de las líneas de código de la aplicación.**

Módulos	Cantidad de líneas de código
Solicitud de pasaportes	1348
Común	2938
Trámite	1446
Gestor de documentos	420
Total	6152 SLOC (Líneas de Código Fuente) 6009
Conversión	6152/1000=6,152 KLOC (Miles de Líneas de Código Fuente).

MM= 3,60*(6,15 ^{1,20})

MM= 31,82

- **Tiempo de desarrollo del proyecto (TDEV) = c*(MM^{d})**

TDEV= 2.50*(31,82 ^{0,32})

TDEV= 7,55 horas/hombre

- **Personas necesarias para realizar el proyecto (CosteH) = MM/TDEV**

CosteH= 31,82 / 7,55

CosteH= 4 personas

- **Costo total del proyecto (CosteM) = CosteH * Salario medio entre los programadores y analistas.**

CosteM= 4*100

CosteM = $ 400

Análisis de la factibilidad económica:

El desarrollo de esta aplicación no supone grandes gastos, ya sean monetario o de tiempo. Además las tecnologías utilizadas para el desarrollo de los componentes del sistema son basadas en software libre. Por lo que haciendo un balance del costo-beneficio de la investigación y atendiendo al resultado que esta tributa se puede concluir que es factible de realizar.

4.5 CONCLUSIONES

En este capítulo con la realización de las pruebas de unidad a los componentes por separado, las de integración y con la utilización de la herramienta JUnit, se pudo comprobar que los componentes funcionan correctamente, brindando los servicios requeridos por los procesos de negocio, evidenciándose la calidad del producto final. Por lo que queda probada la hipótesis planteada en la investigación, otorgando los siguientes valores:

- Tiempo de desarrollo de los componentes a corto plazo.
- Costo de desarrollo bajo.
- Funcionalidad con su máximo indicador, el 3.

CONCLUSIONES GENERALES

El sistema de emisión de pasaportes diplomáticos, formado por los componentes que tributan a los diferentes módulos, la arquitectura definida y el avanzado ambiente de desarrollo, constituyen un importante aporte al desarrollo de la informatización en Venezuela, además del aporte en cuanto a conocimientos que brinda. En este trabajo se demostró la necesidad de diseñar e implementar los componentes que brindarían los servicios necesarios a los procesos de negocio. La propuesta de solución ayuda al sistema a introducir mejoras en los procesos de emisión de los pasaportes diplomáticos en pos de humanizar el trabajo del personal encargado de la realización de los mismos, disminuir el tiempo de elaboración y con ello aumentar la calidad, lo que permitirá aumentar la eficiencia de la organización.

Además para darle cumplimiento a estos objetivos a partir de la metodología utilizada se realizó un modelo general basado en el negocio estudiado, que permitió identificar una lista de rasgos o funcionalidades, a partir de la cual se logró diseñar e implementar los componentes.

Para la realización de las pruebas definidas de los componentes se utilizó la herramienta JUnit. La cual permitió realizar las pruebas tanto a nivel de unidad, como a nivel de los componentes en general, comprobando su buen funcionamiento.

RECOMENDACIONES

Los objetivos de este trabajo han sido logrados teniendo en cuenta que se cumplieron todos los requisitos. Sin embargo se proponen las siguientes recomendaciones:

- Se propone continuar el desarrollo de los componentes y la incorporación de nuevas funcionalidades.
- Aplicar la solución en otras áreas relacionadas con procesos de identificación y seguridad digital.

REFERENCIAS BIBLIOGRÁFICAS

1. BECK, K. Manifiesto de Metodologías ágiles de desarrollo. 2001,
2. CORPORATION, M. La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real. Diciembre del 2006, 24 p.
3. CUBAVISIÓN, P. D. El portal de la cultura matancera Disponible en: http://www.atenas.cult.cu.
4. DEFINICIÓN.DE. Definición.de. 2009, Disponible en: http://definicion.de.
5. DÍAZ, Y. M. Módulo Citas del Sistema de Información Hospitalaria alas HIS. 2009.
6. ESPAÑA, E. S. D. I. D. C. R. E. Tutorial de JUnit de la Escuela Superior de Informática de Ciudad Real en España. 2009,
7. IVARJACOBSON , G. B. Y. J. R. El Proceso Unificado de Desarrollo de Software. Madrid: Addison Wesley, 2000.
8. KRUCHTEN, P. Architectural Blueprints--The 4+1 View Model of Software Architecture. Noviembre 1995,
9. LEÓN, R. A. H. EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA 2002.
10. PALMER, S. R. A Practical Guide to Feature-Driven Development 2002, Disponible en: http://www.amazon.com/exec/obidos/ASIN/0130676152#reader
11. PostgreSQL. Disponible en: http://www.postgresql.org.
12. PRESSMAN, R. S. Ingeniería de Software. 1979.
13. REGUEIRO, L. Componente de acceso a datos para soluciones de emisión de documentos de identificación 2009.
14. RODRÍGUEZ, J. Pruebas unitarias. 2006,
15. SPRINGSOURCE. Disponible en: http://www.springsource.com.
16. VENEZUELA, M. D. P. P. P. R. E. D. Ministerio del Poder Popular para Relaciones Exteriores de Venezuela Disponible en: http://www.misionvenezuela.org.

BIBLIOGRAFÍA

1- INTERNET. Manual de Java. [Online] 2003-2008, Disponible en: http://www.webtaller.com/manual-java/caracteristicas-java.php.

2- SERVICIOS, P. E. D. P. D. Y. D. Documento Metodología Ágil. 2009

3- CALABRIA, L. Metodología FDD. Uruguay: 2003, Disponible en: http://athenea.ort.edu.uy/publicaciones/ingsoft/investigacion/ayudantias/metodologia_FDD.pdf.

4- MAIKEL DE LA TORRE, D. P. Aplicación de Gestión de Interfaces de Usuario para el Sistema de Emisión de Documentos de Identificación del Centro de Identificación y Seguridad Digital. 2008-2009.

5- RENIER SOTES, S. M. Propuesta de integración para los servicios web del Sistema de Emisión de Documentos de Identificación del Centro de Identificación y Seguridad Digital. 2008-2009.

6- CLUB, L. Linux Club Disponible en: http://linux.jovenclub.cu.

7- ERL, T. Service-Oriented Architecture: Concepts, Technology, and Design. 2005

8- TECSISA, E. Tecsisa Madrid: Disponible en: http://www.tecsisa.com.

9- GARIMELLA, K. Introducción a BPM para Dummies. Software AG 2008, Disponible en: http://www.softwareag.es/bpm

10- FILENET. The Synergy between BPM & SOA 2008.

11- GÓMEZ, G. G. MENSAREX: SISTEMA DE MENSAJERÍA DEL MINREX

12- CUBA, M. D. R. E. E. CubaMinrex. nº Disponible en: http://www.cubaminrex.cu.

13- Enciclopedia jurídica. 2009, Disponible en: http://www.enciclopedia-juridica.biz14.com.

14- ZETES. Zetes Disponible en: http://www.zetes.es.

15- ONIDEX. Sistema Autónomo de Identificación, Migración y Extranjería. Disponible en: <http://www.saime.gob.ve/>.

16- MONTERO, M. Análisis y Diseño de un Sistema de Gestión y Control de Trámites para el Ministerio del Turismo (MINTUR). 2009.

17- AWARE. Biometric Services Platform (BioSP) Disponible en: http://www.aware.com.

18- El modelo COCOMO. 2008.

GLOSARIO DE TÉRMINOS

A

1- **Arquitectura:** Organización de los diversos elementos constitutivos de un sistema informático.

2- **Arrays:** Arreglo o alineación es un conjunto o agrupación de variables del mismo tipo.

C

3- **Componentes:** Son todo aquel recurso desarrollado para un fin concreto y que puede formar solo, o junto con otros, un entorno funcional requerido por cualquier proceso predefinido.

D

4- **Direcciones MAC:** En redes de computadoras las direcciones MAC (siglas en inglés de Media Access Control o control de acceso al medio), es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una Ethernet de red.

F

5- **Framework:** Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

6- **Freeware:** Define un tipo de software de computadora que se distribuye sin costo, disponible para su uso y por tiempo limitado, en el que la meta es lograr que un usuario pruebe el producto durante un tiempo y si le satisface, pague por él, habilitando toda su funcionalidad. El freeware suele incluir una licencia de uso, que permite su redistribución pero con algunas restricciones, como no modificar la aplicación en sí, ni venderla, y dar cuenta de su autor.

H

7- **Hardware:** Corresponde a todas las partes físicas y tangibles de una computadora.

8- **HTML** (Lenguaje de Marcado de Hipertexto): Lenguaje de marcado predominante para la construcción de páginas web.

J

9- **JBoss:** Es un servidor de aplicaciones J2EE de código abierto implementado en Java puro.

10- **jBPM:** Es un motor de flujo de trabajo escrito en Java que puede ejecutar procesos descritos en su proceso de lenguaje de definición JPDL.

O

11- **Objetos simulados:** Los objetos simulados se usan para simular el comportamiento de objetos complejos cuando es imposible o impracticable usar al objeto real en la prueba.

P

12- **Paradigma:** Es un modelo o patrón en cualquier disciplina científica.

13- **Periféricos:** En informática, se denominan periféricos a los aparatos o dispositivos auxiliares e independientes conectados a la unidad central de procesamiento de una computadora.

14- **Proceso de negocio**: Ordenación lógicamente interrelacionada de tareas desarrolladas en tiempo y espacio (con comienzo y fin, con entradas y salidas definidas) y que se orienta al logro de un objetivo de negocio, generando un output de valor (total o parcial) para el cliente del proceso.

15- **Protocolo:** Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.

S

16- **Seam:** Es un framework de integración desarrollado por JBoss.

17- **Servidor de aplicaciones:** Se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones.

18- **Software:** Se refiere al equipamiento lógico o soporte lógico de una computadora digital.

W

19- **Workflow:** Un Flujo de Trabajo o Workflow constituye en esquema de tareas (simples o complejas) definidas.

X

20- **XML:** Por sus siglas en inglés *E**x**tensible **M**arkup **L**anguage* (lenguaje de marcas extensible), es un metalenguaje (es un lenguaje que se usa para hablar acerca de otro lenguaje) extensible de etiqueta.