Universidad de las Ciencias Informáticas
Facultad 3

Tema: "Diseño e implementación del proceso de solicitud de copias del sistema informático para los Registros Principales de la República Bolivariana de Venezuela".

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores:

Yanisleidy Torres Puga
Juan Pablo Matos Hernández

Tutores:

Ing. Yunier Raúl Vega Rodríguez
Ing. Armando Esteban Pacheco Iglesias

La Habana, Cuba
Junio 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Yanisleidy Torres Puga

Autor

Juan Pablo Mato Hernández

Autor

_____ _____

Yunier Raúl Vega Rodríguez

Tutor

Armando Esteban Pacheco Iglesias

Tutor

AGRADECIMIENTOS

A mi mamá por todo su apoyo y amor, durante estos 5 años, por ser la mejor madre del mundo. A mi abuelo Chichi por ser un padre para mí, por todo tu esfuerzo, tu lucha, por guiarme por el buen camino, eres mi guía, mi ejemplo a seguir. A mi linda abuela Clara por tanto amor, y cariño.

A mi familia: A mi papá por todos sus concejos, a mi tía Lourdes, a Juan Carlos, Albertico, gracias por su ayuda, apoyo ustedes también forman parte de este triunfo que es de todos.

A mi abuelo Juan, mi abuela Esperanza, a mi tía Yolanda, Ileana, Marbelis, a todos mis primos gracias por ser parte de nuestra linda familia.

A mi hermano, ese que quiero tanto en la vida, y que siempre ha sido incondicional. Espero yo sea un ejemplo para ti a seguir.

A mis compañeros de tesis Yanisleidis y Yuliesky, sin su ayuda y apoyo esto no sería posible.

A Manuel Días, al Kiki, Pedro, Mangly, Earles, Mauro, Papito, Ángel, Juan, Héctor, todos los que de una forma u otra compartimos juntos tanto momentos malos como buenos.

A mi tutor Armando por su guía y apoyo para la realización de este trabajo.

A Liennys por todo el tiempo que pasamos juntos, por su apoyo, por su comprensión, por ser una amiga por encima de todo, de verdad gracias.

A todos, incluyendo los que no he mencionado aquí y me han ayudado de alguna forma, mis más profundo e infinito agradecimiento.

Juan Pablo.

RESUMEN

En el presente documento se realiza un análisis acerca de los principales problemas que presenta el proceso de solicitud de copias en los Registros Principales de la República Bolivariana de Venezuela de acuerdo con lo estipulado en la Ley de Registro Público y del Notariado promulgada en diciembre del 2006.

Además se describe la metodología de desarrollo de software, las herramientas, la plataforma y los frameworks de desarrollo utilizados para realizar la solución propuesta.

Se presenta el diseño e implementación de un sistema de gestión que entre otras funcionalidades incluye el proceso de solicitud de copias que se realiza en los Registros Principales contribuyendo de esta forma a la estandarización de dicho proceso y al aumento de los niveles de seguridad jurídica y de rapidez requerida por el ciudadano venezolano. Además se describen los aspectos más importantes del mismo.

Por último se realiza la validación de la solución propuesta, quedando de esta forma el sistema listo para su despliegue en las oficinas de los registros principales de la República Bolivariana de Venezuela.

INTRODUCCIÓN

En la República Bolivariana de Venezuela una vez que el presidente Hugo Rafael Chávez Frías asumió su primer mandato se comenzaron a realizar una serie de transformaciones de la sociedad y de las estructuras públicas venezolanas que se ven reflejadas en la nueva constitución del año 1999. De esta manera se abrió paso a la promulgación de nuevas leyes como la Ley del Registro Público y del Notariado, la cual en su artículo 10 establece la creación del Servicio Autónomo de Registros y Notarías (SAREN). SAREN es el órgano encargado de planificar, organizar y controlar todos los procesos que se llevan a cabo en las oficinas de los Registros Mercantiles, Públicos, Principales y las Notarías Públicas.

Inicialmente este órgano no llevaba un control efectivo, pues existía poca seguridad y alta probabilidad de intentos de acceso, con la finalidad de sustraer o sabotear documentos legales, información, equipamiento técnico o causar daños a las personas; además los Registros y Notarías existentes funcionaban con total autonomía en sus servicios. Se decide entonces como parte de los acuerdos de cooperación entre Cuba y la República Bolivariana de Venezuela desarrollar un proyecto que tendría como objetivo crear un sistema que informatizara todos los procesos registrales para lograr la ejecución de servicios más eficaces en los Registros Mercantiles, Públicos, Principales y las Notarías Públicas.

El proyecto de desarrollo de la solución tecnológica integral para la automatización y modernización de los Registros y Notarías de la República Bolivariana de Venezuela en su primera fase, comprendió los Registros Públicos, Mercantiles y la Administración Financiera del Servicio de Registros y Notarías, mientras que los Registros Principales y las Notarías Públicas aún no se encuentran informatizados.

Los Registros Principales rigen su funcionamiento de acuerdo con la Ley de Registro Público y del Notariado[1] la cual establece en su artículo 65, que corresponde al

[1] Gaceta Oficial Nº 5.833 Extraordinaria del 22 de diciembre de 2006

Registro Principal efectuar la inscripción de los actos siguientes (Gobierno Bolivariano de Venezuela, 2006):

1. La separación de cuerpos y bienes, salvo que se trate de bienes inmuebles y derechos reales, los cuales se harán por ante el Registro de Propiedad.

2. Las interdicciones e inhabilitaciones civiles.

3. Los títulos y certificados académicos, científicos, eclesiásticos y los despachos militares.

Igualmente, corresponde al Registro Principal recibir y mantener los duplicados de los asientos de los registros públicos, registros civiles municipales y parroquiales y expedir copias certificadas y simples de los asientos y duplicados de los documentos que reposan en sus archivos. (Gobierno Bolivariano de Venezuela, 2006).

El Registro Principal a través del proceso de solicitud es el responsable de realizar los trámites para la emisión de las copias simples y certificadas de los documentos:

- Actas
  - Acta de nacimiento
  - Acta de matrimonio
  - Acta de defunción
- Documento duplicado
- Documentos protocolizados
  - Sentencias
  - Títulos Universitarios
  - Actas Constitutivas y de Asambleas

En la actualidad en los Registros Principales y específicamente dentro del proceso de solicitud de copias se presentan problemas que afectan la calidad de los servicios que se ofrecen al ciudadano, como son:

- Existe baja calidad y lentitud en los servicios de solicitud.
- Falta de uniformidad en los procedimientos de trabajo de las oficinas de Registros Principales haciéndole difícil a SAREN el control y administración de los registros.
- En la mayoría de las oficinas el trabajo se realiza de forma manual, provocando atrasos en la entrega de los documentos.
- La información documental generalmente se encuentra almacenada en soporte físico, lo que provoca que tienda a deteriorase con el tiempo y que disminuya la calidad de la información que se resguarda.
- En los Registros Principales se guardan los duplicados de sus oficinas y de otras, por lo que si ocurre una catástrofe, puede perderse toda o parte de la información que estos almacenan.

Todo lo anteriormente planteado muestran la situación real - actual de los Registros Principales de la nación venezolana y por ende, el por qué de la necesidad de realizar cambios generales en las oficinas de los Registros Principales.

Debido a la situación problemática antes expuesta se plantea el **problema a resolver** a través de la siguiente interrogante: ¿Cómo contribuir a estandarizar y perfeccionar el proceso de solicitud de copias que se realiza en los Registros Principales de la República Bolivariana de Venezuela en relación con los niveles de seguridad jurídica y la rapidez requerida en los servicios que se brindan al ciudadano venezolano?

Siendo el **objeto de estudio** de esta investigación el proceso de desarrollo de software y el **campo de acción** se centra en el diseño e implementación del proceso de solicitud de copias que se realiza en los Registros Principales de la República Bolivariana de Venezuela.

Para dar solución al problema planteado, se traza como **objetivo general**: Realizar el diseño e implementación del proceso de solicitud de copias para los Registros Principales de la República Bolivariana de Venezuela, que contribuya a estandarizar y

perfeccionar dicho proceso en relación con los niveles de seguridad jurídica y la rapidez requerida en los servicios que se brindan al ciudadano venezolano.

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

- Elaborar el marco teórico de la investigación.
- Realizar el diseño de la solución informática.
- Realizar la implementación de la solución informática diseñada.
- Validar los resultados obtenidos.

Como **idea a defender** de esta investigación se tendrá: El diseño e implementación del proceso de solicitud de copias del sistema informático para los Registros Principales de la República Bolivariana de Venezuela, contribuirá a estandarizar y perfeccionar dicho proceso en relación con los niveles de seguridad jurídica y la rapidez requerida en los servicios que se brindan al ciudadano venezolano.

Para dar cumplimiento a los objetivos específicos trazados, se realizarán las siguientes **tareas de investigación**:

- Estudio de las leyes que rigen el funcionamiento del proceso de solicitud de copias en los Registros Principales de Venezuela.
- Estudio del proceso de solicitud de copias en el sistema actual de Registros Principales en la República Bolivariana de Venezuela.
- Investigación sobre los sistemas existentes actualmente en los Registros Principales en la República Bolivariana de Venezuela.
- Estudio de los artefactos generados durante el modelado del negocio y la captura de requisitos.

- Diseño de los diagramas de clases de los casos de usos del proceso de solicitud de copias.
- Diseño de los diagramas de secuencia de los casos de uso del proceso de solicitud de copias.
- Realización del modelo de implementación.
- Implementación del proceso de solicitud de copias.
- Validación del modelo de diseño a través de métricas.
- Validación del código fuente mediante pruebas de unidad y pruebas de aceptación.

Con este trabajo quedarán informatizadas las funcionalidades correspondientes al proceso de solicitud de copias en los Registros Principales de Venezuela. Entre los principales resultados esperados con esta investigación se encuentran: solución integral para la configuración y gestión de las solicitudes que se realizan en los Registros Principales, incremento sustancial de la seguridad jurídica y completamiento de la plataforma informática para las oficinas de los Registros Principales.

El presente trabajo consta de tres capítulos en los cuales se desarrollan aspectos de importancia para lograr los objetivos que se persiguen.

Capítulo 1: Aborda aspectos generales sobre los Sistemas Registrales y los principios que rigen el Sistema Registral Venezolano. También se analiza lo que le corresponde al Registro Principal según la Ley de Registro Público y del Notariado, y se describe el proceso de solicitud de copias. Además se realiza un estudio acerca de los principales paradigmas de programación, patrones de diseño, métricas y pruebas de calidad, así como la descripción de la metodología, plataforma, frameworks y herramientas a emplear.

Capítulo 2: Se presenta la propuesta de solución del sistema, explicando la arquitectura sobre la cual se desarrollará el mismo. Además se hace alusión a los patrones que fueron empleados en la construcción del sistema y se describen los artefactos generados durante el diseño y la implementación.

Capítulo 3: Muestra el análisis de la validación del sistema. Se exponen los resultados obtenidos de la medición del software por algunas de las principales métricas utilizadas en la actualidad para determinar la calidad del diseño de Sistemas Orientados a Objetos. Además se presentan las pruebas de unidad realizadas al código y las pruebas de aceptación aplicadas para verificar que el software cumple los requisitos establecidos por los clientes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Con el fin de alcanzar los objetivos trazados, se realiza en el presente capítulo una serie de estudios, de temas como: los Sistemas Registrales y principios que rigen el Sistema Registral Venezolano, lo expresado en la Ley de Registro Público y del Notariado en cuanto a los Registros Principales y los subprocesos que conforman el proceso de solicitud de copias. Se muestran además las tendencias y tecnologías empleadas que facilitan y permiten la realización del diseño e implementación del sistema informático.

1.1 Sistema Registral

Por Sistema Registral se entiende el conjunto de normas que en un determinado país regulan las formas de publicidad de los derechos reales sobre bienes inmuebles a través del Registro de la Propiedad, también incluye el régimen y organización de esta institución o más sintéticamente, el conjunto de normas reguladoras de la institución del Registro de la Propiedad, tanto desde el punto de vista sustantivo (valor de los asientos, como forma de constitución de publicidad de aquellos Derechos) como desde un punto de vista formal, organización y régimen del Registro. (Fernández, 1947)

Los Sistemas Registrales varían de acuerdo al país en que se apliquen, siguiendo las normas y derechos que se establecen en esta institución. Todo Sistema Registral consta de un grupo de Principios Registrales, que deben ser de obligatorio cumplimiento, y a raíz de esto, se conforman sus peculiaridades y diferencias con respecto a otros países.

1.1.1 Principios Registrales

"*Los principios registrales explican el contenido y función del Registro Público de la Propiedad. Asimismo están totalmente entrelazados unos de otros de tal manera que no existen en forma independiente".* (Pérez Fernández del Castillo, 1990).

La solución informática que se desea implementar automatizará el proceso de solicitud de copias que se realiza en las oficinas de los Registros Principales de la República Bolivariana de Venezuela, la solución además de ser un sistema de gestión y control va a estar enmarcada en un contexto jurídico, por lo tanto se debe ajustar a la Ley y no puede violar ninguno de los principios registrales que esta establece, siendo la seguridad jurídica un eslabón fundamental dentro de los objetivos del desarrollo de este sistema.

En la Ley del Registro Público y del Notariado, redactada en la Asamblea Nacional de la República Bolivariana de Venezuela, se enumeran los principios registrales que rigen el trabajo en las oficinas de los Registros Principales venezolanos. A continuación se analizarán estos principios con el objetivo de comprender sus implicaciones sobre el sistema a desarrollar.

1.1.1.1 Principio de Rogación

"La presentación de un documento dará por iniciado el procedimiento registral, el cual deberá ser impulsado de oficio hasta su conclusión, siempre que haya sido debidamente admitido". (Gobierno Bolivariano de Venezuela, 2006)

Este principio establece que el centro de atención en un registro es el documento. El procedimiento registral será iniciado por la presentación de un documento en un registro; es decir, salvo algunas pocas excepciones como pudieran ser los documentos remitidos por alguna autoridad, todo documento debe ingresar al registro mediante una solicitud realizada por un usuario.

1.1.1.2 Principio de Prioridad

"Todo documento que ingrese al Registro deberá inscribirse u otorgarse con prelación a cualquier otro presentado posteriormente, salvo las excepciones establecidas en esta Ley". (Gobierno Bolivariano de Venezuela, 2006).

Este principio establece que el primer documento que se ingrese en tiempo sea el primero en derecho. En caso de algún litigio jurídico el documento que mayor peso o prioridad tiene es el presentado con mayor antelación salvo excepciones establecidas por la ley.

1.1.1.3 Principio de Especialidad

"Los bienes y derechos inscritos en el Registro, deberán estar definidos y precisados respecto a su titularidad, naturaleza, contenido y limitaciones". (Gobierno Bolivariano de Venezuela, 2006).

El principio de especialidad establece que los bienes y derechos a ser inscritos estén bien definidos y precisados por cada registro, de manera tal que ninguno pueda asumir las funciones de otro. Esto no significa que un documento sólo pueda ser inscrito en un registro o que dos oficinas no puedan compartir el mismo espacio físico, pero en materia jurídica y registral su espacio estará bien delimitado.

1.1.1.4 Principio de Consecutividad

"*De los asientos existentes en el Registro, relativos a un mismo bien, deberá resultar una perfecta secuencia y encadenamiento de las titularidades del dominio y de los demás derechos registrados, así como la correlación entre las inscripciones y sus modificaciones, cancelaciones y extinciones".* (Gobierno Bolivariano de Venezuela, 2006).

Este principio expresa el encadenamiento de los títulos y la perfecta secuencia de los derechos registrales, así como la correlación que debe existir entre las inscripciones y los demás procesos posteriores que puedan surgir.

1.1.1.5 Principio de Legalidad

"Sólo se inscribirán en el Registro los títulos que reúnan los requisitos de fondo y forma establecidos por la ley". (Gobierno Bolivariano de Venezuela, 2006).

Este principio expresa que para que se inicie el proceso de inscripción los documentos deberán ser revisados por la persona o conjunto de personas calificadas para la revisión de estos requisitos de fondo y forma, verificando de esta forma el cumplimiento de todas las normas y leyes establecidas. El fondo se refiere a lo que está escrito en el documento y es necesario leer para verificar que esté correcto, la forma se refiere al formato, estructura y organización de la información en el documento inscrito.

1.1.1.6 Principio de Publicidad

"La fe pública registral protege la verosimilitud y certeza jurídica que muestran sus asientos. La información contenida en los asientos de los registros es pública y puede ser consultada por cualquier persona". (Gobierno Bolivariano de Venezuela, 2006).

Fe pública: Es la autoridad legítima que se le brinda a notarios, escribanos, agentes de cambio y bolsa, cónsules y secretarios de juzgados, tribunales y otros institutos oficiales, para que los documentos que autorizan sean considerados como auténticos y lo contenido en ellos sea verdadero mientras no se pruebe lo contrario . (Real Academia Española)

Este principio establece que los documentos inscritos en los registros pueden ser consultado por cualquier persona que lo desee, constituyendo un derecho de la persona poder ver cualquier documento. Además la fe pública registral garantiza la veracidad y certeza jurídica de los documentos que en el registro se expiden.

Después de explicar los principios que rigen el funcionamiento de los registros, se pueden analizar los Registros Principales en particular. Todos estos principios se manifiestan en los procesos u operaciones que se desarrollan en los Registros Principales.

En el proceso de solicitud específicamente se tendrá en cuenta los principios registrales que en él se deben aplicar a la hora de automatizar dicho proceso.

1.2 Registros Principales

Los Registros Principales son oficinas que están adscritas a SAREN y pertenecen a su vez al Ministerio del Poder Popular para Relaciones Interiores y Justicia (MPPRIJ). Actualmente existen 21 Registros Principales distribuidos por todo el país. Estos registros se encargan de realizar distintos tipos de trámites como son inscripción de documentos, solicitud de copias, legalizaciones de firmas y notas marginales.

En la Ley de Registro Público y del Notariado se establece en su artículo 65, que corresponde al Registro Principal efectuar la inscripción de los actos siguientes:

1. La separación de cuerpos y bienes, salvo que se trate de bienes inmuebles y derechos reales, los cuales se harán por ante el Registro de Propiedad.
2. Las interdicciones e inhabilitaciones civiles.
3. Los títulos y certificados académicos, científicos, eclesiásticos y los despachos militares.

Igualmente, corresponde al Registro Principal recibir y mantener los duplicados de los asientos de los registros públicos, registros civiles municipales y parroquiales y expedir copias certificadas y simples de los asientos y duplicados de los documentos que reposan en sus archivos. (Gobierno Bolivariano de Venezuela, 2006)

Luego de analizar lo que le corresponde al Registro Principal se centrará la atención en el proceso de solicitud de copias para dar cumplimiento a los objetivos propuestos.

1.2.1 Proceso de Solicitud de Copias en los Registros Principales

El proceso de solicitud de copias se encuentra organizado en fases o subprocesos bien definidos por los que se pasa hasta llegar a la entrega de la copia del documento solicitado. A través de este proceso se realizan las copias simples y certificadas de documentos como:

- Actas
  - Acta de nacimiento
  - Acta de matrimonio
  - Acta de defunción
- Documento duplicado
- Documentos protocolizados
  - Sentencias
  - Títulos Universitarios
  - Actas Constitutivas y de Asambleas

En la Fig. 1 se muestra una imagen que ilustra el proceso general de solicitud de copias y se explican los subprocesos que comprenden a dicho proceso.

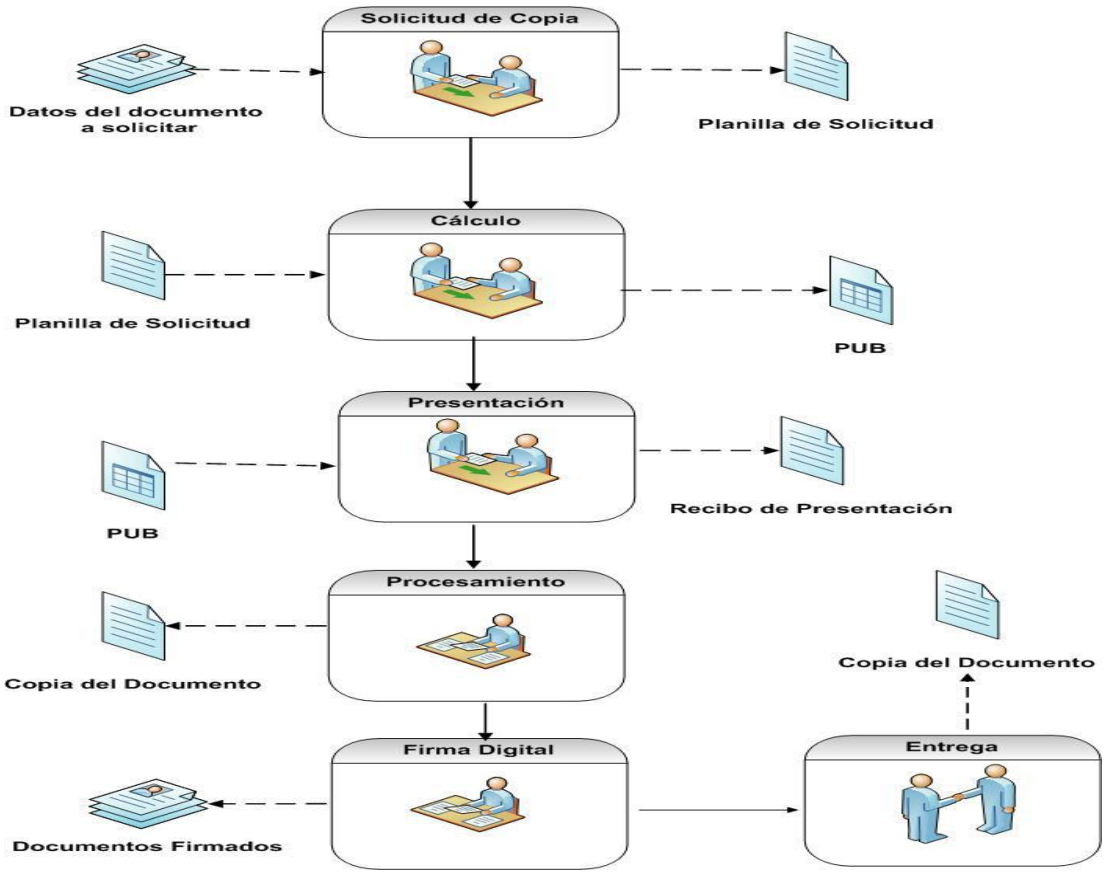

Fig. 1: Diagrama del flujo de procesos que se realiza en el proceso de solicitud de copias en los Registros Principales.

Solicitud de Copia: Es la fase inicial del proceso de solicitud de copias. En este proceso es donde el usuario solicita a que documento hacerle la copia y qué tipo de copia. Posteriormente se crea la planilla de solicitud y se le da paso a la siguiente fase, por lo que el usuario se dirige a taquilla para realizar el cálculo del documento.

Cálculo: En esta fase es donde se le informa al usuario las Unidades Tributarias[2] que debe pagar para la realización del servicio solicitado. Después se elabora la Planilla Única Bancaria (PUB) y se le entrega.

Presentación: Para comenzar esta fase el usuario entrega los recaudos y la PUB cancelada al funcionario de Presentación. El funcionario establece la fecha y la hora de la posible entrega. Luego se elabora un recibo que el usuario debe firmar. El funcionario también lo firma y le entrega una copia del recibo al usuario, el cual se retira del registro.

Procesamiento: En esta fase el funcionario de Procesamiento realiza la copia al documento solicitado y elabora y revisa la nota de certificación en correspondencia al acto seleccionado. Posteriormente el funcionario de Revisión revisa la nota y si está correcta coloca la firma y el sello de revisado, en caso contrario envía la nota para su corrección.

Firma digital: En esta fase el registrador principal es el encargado de firmar digitalmente tanto el documento original como la nota de certificación o mecanografiada, en correspondencia con el tipo de acto seleccionado garantizando de esta forma la autenticidad e integridad de los mismos y evitando que sean falsificados por terceras personas.

Entrega: En esta fase el registrador revisa la copia del documento y la nota de certificación. Si los documentos están incorrectos se envían para procesamiento,

[2] Es el monto de ley sobre el cual se hacen los cálculos de tributos, multas y demás prestaciones dinerarias debidas al estado.

donde el documento es nuevamente procesado. En caso que los documentos estén correctos el registrador firma y entrega la copia del documento al usuario.

1.3 Soluciones Informáticas existentes

En la actualidad en los Registros Principales de la República Bolivariana de Venezuela existen disímiles aplicaciones informáticas que automatizan los procesos que se realizan en estos registros. Estas aplicaciones generalmente automatizan procesos aislados y no tienen relación entre ellas, por lo que no cuentan en las oficinas con un sistema único que permita realizar la totalidad del proceso. El principal inconveniente es que las soluciones existentes son independientes y no se subordinan a un ente controlador, por lo que los registros funcionan con total autonomía en los servicios.

Debido a los problemas que presentan las aplicaciones existentes hoy en los Registros Principales es necesario hacer un análisis riguroso para que la aplicación a desarrollar erradique estos problemas, facilite un mejor servicio en los Registros Principales y contribuya a garantizar la seguridad jurídica al ciudadano venezolano.

1.4 Tendencias

1.4.1 Paradigmas de programación

Un paradigma de programación representa un enfoque particular o filosofía para la construcción del software. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas. (López, 2009).

Entre los paradigmas de programación se encuentran la programación imperativa y la declarativa. La imperativa describe la programación como una secuencia de instrucciones o comandos que cambian el estado de un programa. Los programas imperativos son un grupo de procedimientos que le muestran al computador cómo efectuar una tarea. (López, 2009)

El paradigma declarativo se enfoca en describir las propiedades de la solución buscada, dejando indeterminado el conjunto de instrucciones usado para encontrar esa solución. La solución se logrará mediante mecanismos internos de inferencia de información a partir de la descripción realizada. (López, 2009)

Por otro lado tenemos el paradigma de programación orientada a objetos (POO) que es una nueva forma de programar, que se acerca a como se ven las cosas en la vida real. Este paradigma desde sus inicios cambió la forma de pensar a la hora de programar y trajo consigo muchos beneficios, que dieron gran impulso a la construcción de sistemas cada vez más complejos. (López, 2009)

1.4.1.1 Programación Orientada a Objetos

La POO se acerca más a lo que refleja la realidad, lo que hace que esta sea fácil de comprender y que ayude a dominar la complejidad del software. Dicha programación se basa en la simulación del mundo real mediante objetos, clases, métodos, herencia, entre otros. A continuación se muestran algunos de los elementos básicos de la POO:

Objetos: Los objetos se utilizan para modelar elementos reales o abstractos del ámbito del problema a resolver. Estos tienen responsabilidades y comportamientos definidos.

Clase: *"Una clase se puede definir como una descripción abstracta de un grupo de objetos*"**.** (Aguilar, 1996). Los objetos creados por una clase contienen los mismos atributos y operaciones, pero toman valores diferentes.

Método: Está formado por un conjunto de operaciones que pueden producir cambio en las propiedades de un objeto o generar un evento.

Características principales de la POO:

Abstracción: La abstracción permite ignorar los aspectos de la realidad que no influyen en el ámbito del problema que se modela. Esta se centra en los elementos comunes, pero permite variaciones.

Encapsulación: Permite ocultar los detalles internos del comportamiento de una clase, dándole acceso al programador sólo a los detalles que necesite para el resto del programa.

Herencia: La herencia posibilita que se concentren conceptos comunes de varias clases en una superclase y que estos conceptos puedan ser heredados a varias subclases.

Polimorfismo Es la propiedad que indica, literalmente, la posibilidad de que una entidad tome muchas formas. En términos prácticos, el polimorfismo permite referirse a objetos de clases diferentes mediante el mismo elemento de programa y realizar la misma operación de diferentes formas, según sea el objeto que se referencie en ese momento". (Aguilar, 1996)

Entre las ventajas que ofrece la POO se encuentra que (Budd, 1991):

- Promueve la reutilización, es decir permite utilizar el mismo código en varios programas sin necesidad de volver a escribir este. Esto trae consigo un ahorro considerable en tiempo de implementación y por lo tanto se agiliza el desarrollo del software.

- Reduce la complejidad en el desarrollo de software ya que los objetos muestran sólo su interfaz pública, ocultando los detalles de la implementación.

- Al permitir relacionar el sistema con el mundo real es generalmente más fácil de comprender y modelar, posibilitando que se creen sistemas complejos con mayor facilidad.

1.4.1.2 Programación Orientada a Aspectos

La programación orientada a aspectos (POA) se encarga principalmente de separar competencias y disminuir las dependencias entre estas, con lo que se consigue que los diferentes conceptos que forman una aplicación se encuentren en el lugar que le corresponde (Pollack, y otros, 2007). A continuación se muestran algunos de los conceptos básicos de la POA:

Aspecto (Aspect) (Pollack, y otros, 2007)**:** Es una funcionalidad transversal que se va a implementar de forma modular y separada del resto del sistema. Un ejemplo común de un aspecto es la administración de transacciones en aplicaciones empresariales.

Punto de Cruce o de Unión (Join point) (Pollack, y otros, 2007)**:** Es un punto de ejecución dentro del sistema donde un aspecto puede ser conectado, como una llamada a un método, el lanzamiento de una excepción o la modificación de un campo. El código del aspecto será insertado en el flujo de ejecución de la aplicación para añadir su funcionalidad.

Consejo (Advice) (Pollack, y otros, 2007)**:** Es la implementación del aspecto, es decir, contiene el código que implementa la nueva funcionalidad. Se insertan en la aplicación en los puntos de cruce.

Puntos de Corte (Pointcut) (Pollack, y otros, 2007)**:** Define los consejos que se aplicarán a cada punto de cruce. Se especifica mediante expresiones regulares o mediante patrones de nombres (de clases, métodos o campos), e incluso dinámicamente en tiempo de ejecución según el valor de ciertos parámetros.

Introducción (Introduction) (Pollack, y otros, 2007)**:** Permite añadir métodos o atributos a clases ya existentes.

Destinatario (Target): Es la clase aconsejada, la clase que es objeto de un consejo. Sin AOP, esta clase debería contener su lógica, además de la lógica del aspecto. (Pollack, y otros, 2007).

Entre las ventajas de la POA se encuentra que (Pollack, y otros, 2007):

- El código que se tiene es menos confuso y más reusable.
- Elimina la dispersión del código, lo que posibilita que las implementaciones se vuelvan más comprensibles y adaptables.
- Facilita la modificación del código, porque al concentrar cada concepto en un lugar es más fácil de localizar donde es necesario realizar el cambio.
- Permite que grandes cambios en la definición de una materia tenga un pequeño impacto en las demás.

1.4.2 Patrón de diseño

Según Erich Gamma[3] un patrón de diseño es *"Una solución (probada) a un problema en un determinado contexto*". (Gamma, 2002)

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en problemas específicos a la hora de realizar una aplicación orientada a objetos. Son soluciones surgidas por las experiencias pasadas y que se ha demostrado que funcionan.

Entre los patrones de diseño se encuentran los Patrones de Software para la Asignación General de Responsabilidad (GRASP[4]) y los patrones Gang of Four[5] (GOF).

[3]Erich Gamma: uno de los autores del libro Design Patterns.
[4]Acrónimo de General Responsability Assignment Software Patterns
[5] Es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns.

Los patrones GOF se clasifican respecto a su ámbito en Clase y Objetos; mientras que según su propósito se dividen en Creacionales, Estructurales y de Comportamiento. A continuación se describen algunos de los patrones GOF que van a ser utilizados en la aplicación.

Fábrica Abstracta (Abstract Factory): *"El patrón fábrica abstracta provee una interfaz para crear una familia de objetos similares o dependientes sin especificar sus clases concretas".* (Gamma,2002).

Solitario (Singleton): Su objetivo es garantizar que para cada clase existe una única instancia y proveer un mecanismo global de acceso a ella.

Fachada (Facade): Consiste en ofrecer una sencilla interfaz que oculte uno o varios sistemas más complejos y sus interacciones.

Los patrones GRASP son utilizados para asignar responsabilidades. Estos describen los aspectos esenciales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. En el grupo de los patrones GRASP se encuentran los siguientes patrones: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Variaciones Protegidas, Controlador, Fabricación Pura, Indirección, No hables con extraños y Polimorfismo.

Los patrones de diseño promueven la reutilización, facilitan la modificación y la comprensión del software construido. Además permiten solventar problemas de optimización de código, reutilización de componentes y simplicidad de las soluciones.

1.5 Metodologías y herramientas de modelado para el diseño de software

Metodologías de Desarrollo de Software

El desarrollo de software no es una tarea fácil, por lo que si no se lleva una metodología de por medio, sólo se obtendrán clientes insatisfechos.

Dado que los requisitos de un software a otro son tan variados y cambiantes, existen una gran variedad de metodologías para la creación de software. Por un lado tenemos las metodologías tradicionales que se centran especialmente en el control del proceso y por otro lado las metodologías ágiles que están orientadas a la interacción con el cliente y el desarrollo incremental del software.

Tanto las metodologías tradicionales como las ágiles han demostrado ser efectivas en un gran número de proyectos, pero también han presentado problemas en otros muchos; por lo que hay que tener especial cuidado a la hora de seleccionar la metodología a utilizar en un software determinado.

1.5.1.1 Metodologías Tradicionales

Las metodologías tradicionales o pesadas son aquellas que implantan una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de desarrollar software más eficiente. Estas se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada.

Este tipo de metodologías son más eficaces y necesarias cuando:

- El proyecto es de gran envergadura respecto a tiempo y recursos que son necesarios emplear.
- El equipo de desarrollo es grande.
- Los requisitos están bien definidos.

1.5.1.1.1 Rational Unified Process (RUP)

RUP es una metodología de desarrollo de software que brinda a los equipos de desarrollo guías, estándares, y un enfoque disciplinario para la asignación de actividades y responsabilidades dentro del equipo de desarrollo.

El ciclo de vida de RUP presenta tres aspectos claves que son los que hacen único al proceso unificado:

Dirigido por los Casos de Uso: *"Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante*". (Jacobson, 2000). Se dice que RUP está dirigido por los casos de usos porque el proceso de desarrollo avanza a través de una sucesión de flujos que parten de los casos de usos.

Centrado en la Arquitectura: RUP hace un importante énfasis en la definición temprana de una adecuada arquitectura que sea flexible durante todo el proceso de desarrollo, para que no se vea fuertemente impactada ante los cambios.

Iterativo e Incremental: Posee una secuencia de iteraciones y cada una de estas iteraciones aborda una porción de la funcionalidad total del software.

RUP divide el proceso de desarrollo en cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada una de estas fases se divide en varias iteraciones, dentro de las que se desarrollan actividades de los nueve flujos de trabajo, en mayor o menor medida dependiendo de la iteración y la fase en que se encuentre. Dentro de los nueve flujos de trabajo con que consta RUP, seis son ingenieriles y tres de apoyo, entre los ingenieriles se encuentran: modelado del negocio, requisitos, análisis, diseño, implementación y pruebas, dentro del grupo de apoyo se encuentran: administración del proyecto, configuración de control y de cambio y entorno.

Se ha seleccionado la metodología RUP porque presenta varios aspectos favorables para el desarrollo del proyecto:

- Es una metodología que ha proporcionado resultados satisfactorios en proyectos con características similares al nuestro.
- Existe gran documentación de la misma.

- El equipo de desarrollo está familiarizado con esta metodología, por lo que la introducción de una nueva atrasaría el cronograma.

- Genera documentación que es exigida por el cliente y necesaria por la dinámica del proyecto, ya que en el mismo el equipo de trabajo varía y se necesita tener documentación de lo que se ha hecho.

Una vez analizada la metodología de desarrollo de software, es necesario estudiar las herramientas de diseño que se utilizarán para el modelado del mismo.

1.5.2 Herramientas de modelado para el diseño de software

Las herramientas CASE[6] son diversas aplicaciones informáticas que sirven de apoyo en todo el ciclo de vida del proceso de desarrollo de software. Estas herramientas además de agilizar el trabajo, mejorar y estandarizar la documentación y facilitar la realización de prototipos; contribuyen a mejorar la calidad y la productividad en el desarrollo de software.

Actualmente existen disimiles herramientas CASE que presentan diversas utilidades en cuanto a la gestión de proyectos, la semi-automatización de la generación de código en diferentes lenguajes, la automatización de la documentación y el mantenimiento del código.

1.5.2.1 Enterprise Architect for UML[7] v2.1

Enterprise Architect es una herramienta CASE de modelado de todos los aspectos del ciclo de desarrollo para sistemas de negocio y tecnologías de la información, ingeniería de software y sistemas, desarrollo en tiempo real y embebido. Esta herramienta soporta la especificación de UML 2.1 y provee soporte para pruebas, mantenimiento y control de cambio.

[6] Computer Assisted Software Engineering, Ingeniería de Software Asistida por Ordenador
[7] Unified Modeling Language, Lenguaje Unificado de Modelado

Algunas de las características del Enterprise Architect son (Sparx Systems, 2000-2010):

- Genera y realiza ingeniería inversa de lenguajes como Actionscript, C++, C#, Delphi, Java, PHP, Python y Visual Basic.
- Permite modelar base de datos y generar scripts DDL[8].
- Provee trazabilidad completa desde los modelos de requisitos, análisis y diseño, hasta la implementación y despliegue.
- Genera clases de prueba NUnit y JUnit desde clases origen.
- Recolecta incidencias del proyecto, tareas y el glosario del sistema.
- Asigna recursos a los elementos del modelo y compara el esfuerzo que lleva con el esfuerzo requerido.
- Integración con los entornos de desarrollo Eclipse y Visual Studio .NET.
- Genera reportes detallados con calidad en formatos RTF [9]Y HTML[10].
- Control de versiones con proveedores SCC[11], CVS[12] y configuraciones del control de versiones de subversión.
- Guarda y descarga diagramas completos como patrones UML.
- La verificación y validación efectiva y el análisis del impacto inmediato son posibles a través del ciclo de vida completo, usando capacidades tales como la matriz de relaciones y la vista de jerarquía de Enterprise Architect.

[8] Data Definition Language, Lenguaje de Definición de Datos
[9] Rich Text Format
[10] Hypertext Markup Language, Lenguaje de Marcado de Hipertexto
[11] Source Code Control
[12] Control Version System

1.5.2.2 ER/Studio

ER/Studio es una herramienta profesional de modelado de datos fácil de usar para el diseño y construcción de bases de datos a nivel lógico y físico. Esta herramienta permite la sincronización bidireccional entre los diseños lógicos y físicos, lo que posibilita que si se hace algún cambio en uno de los dos diseños antes mencionados, se refleje de forma automática en el otro.

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Entre las características del ER/Studio se encuentran (Data Modeling Enterprise Software| Embarcadero ER/Studio):

- Permite realizar diagramas con un desempeño claro y rápido.
- Mantiene automáticamente todas las dependencias entre sub-vistas y el diagrama completo.
- Da la posibilidad de hacer reingeniería inversa de bases de datos.
- Soporta un gran número de gestores de base de datos.
- Permite la creación de reportes de forma sencilla.
- Genera vistas, procedimientos almacenados, reglas, y tipos de datos de usuario, lo cual ayuda a la auto-ordenación de tipos de objetos para eliminar errores de dependencia al construir la base de datos.

1.6 Plataformas y Frameworks de desarrollo

1.6.1 Plataformas de Desarrollo

Con el fin de hacer eficiente la capacidad de desarrollo y hacer el trabajo más rápidamente, con menor coste, más eficiencia y rapidez en las tareas de mantenimiento, han surgido las plataformas de desarrollo de aplicaciones de gestión

y aplicaciones web que constituyen el entorno de software común en el cual se desenvuelve la programación de un grupo de aplicaciones. Comúnmente se encuentran relacionadas directamente con un sistema operativo, sin embargo, también es posible encontrarlas ligadas a una familia de lenguajes de programación o a una interfaz de programación de aplicaciones[13].

1.6.1.1 Plataforma de desarrollo Microsoft .NET

Microsoft .NET es una plataforma de software que conecta información, sistemas, personas y dispositivos. Esta plataforma proporciona a los desarrolladores las herramientas y tecnologías para desarrollar aplicaciones de manera rápida pero a la vez segura y robusta; logrando así una integración más rápida entre empresas y un acceso más simple a todo tipo de información desde cualquier tipo de dispositivo (.Net Framework Developer Center).

El .NET Framework es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en entornos altamente distribuidos. Este framework contiene tres componentes fundamentales (.Net Framework Developer Center):

Entorno Común de Ejecución de Lenguajes: El Entorno Común de Ejecución (CLR) constituye el núcleo del framework. Este proporciona funciones como la gestión de memoria, la seguridad y un sólido sistema de control de errores, a cualquier lenguaje que se integre en .NET Framework. Gracias al CLR, todos los lenguajes .NET pueden usar varios servicios de ejecución sin que los programadores tengan que preocuparse de si su lenguaje particular admite una función de ejecución.

Biblioteca de clases de .Net: Las bibliotecas de clases .NET Framework son sumamente importantes para proporcionar interoperabilidad al lenguaje porque

[13] API por sus siglas en inglés.

permiten a los desarrolladores usar una sola interfaz de programación para toda la funcionalidad expuesta por el CLR.

Lenguajes de compilación: Este framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes.

Esta plataforma presenta disimiles ventajas, dentro de las que se encuentran:

Código administrado: El CLR impone seguridad en el acceso al código, mejora el rendimiento, administra la memoria y controla los servicios del sistema para que la aplicación se ejecute correctamente.

Interoperabilidad multilenguaje: Puede coexistir en una misma aplicación código escrito en diferentes lenguajes compatible con .Net, ya que siempre se compila en código intermedio (MSIL).

Recolector de Basura: El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente, de esta forma el programador no tiene que liberar la memoria de forma explícita.

Seguridad de acceso al código: Permite especificar el conjunto de operaciones que debe permitirse al código, así como las operaciones que nunca deben permitírsele.

1.6.1.1.1 Visual Studio .NET 2008

Visual Studio .NET es un Entorno de Desarrollo Integrado (IDE) especialmente diseñado para facilitar la construcción y el desarrollo de servicios Web XML, aplicaciones para Windows y soluciones Web.

Dentro de las características del Visual Studio 2008 se puede mencionar que (Microsoft,2008):

- Facilita la creación de aplicaciones, ya que utiliza las funcionalidades brindadas por el framework como librerías de clases y recolector de basura.
- Soporte Multilenguaje, que le permite integrar en una misma aplicación código escrito en diferentes lenguajes (Visual Basic .NET, Visual C# .NET, Visual C++ .NET, Visual J# .NET, soporte para la instalación de otros lenguajes).
- Permite escoger con que framework trabajar (.Net Framework 2.0, 3.0 o 3.5).
- Presenta un completo depurador y permite crear reportes con el Crystal Reports.

1.6.2 Frameworks de Desarrollo

"*Un framework, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto*". (CodeBox)

Los framework son diseñados con la intención de evitar construir aplicaciones desde cero. Estos facilitan el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

1.6.2.1 NHibernate

NHibernate es un framework de Mapeo Relacional de Objetos (ORM[14]) de alto rendimiento para la persistencia y consulta de objetos. Permite el desarrollo de clases persistentes siguiendo el paradigma orientado a objetos; incluye asociación, herencia, composición y polimorfismo.

[14] Acrónimo de Object-Relational-Mapping

Este framework es adecuado para aplicaciones centradas en los datos o con modelos de negocios complejos. Entre sus ventajas se puede mencionar (Microsoft, 2007):

- Puede aceptar consultas SQL directas.
- Reduce el tiempo de desarrollo de grandes sistemas orientados a objetos que usan bases de datos relacionales, aliviando al desarrollador de gran parte de las tareas comunes de programación relacionadas con la persistencia de datos.
- Al ser un framework de ORM soporta el mapeo de todos los tipos de relaciones que pueden existir en un modelo de objeto del dominio.
- Posee independencia completa del tipo de base dato utilizado o lo que es lo mismo la aplicación puede persistir sus datos en SQL Server, en Oracle, en MySQL, simplemente cambiando la configuración correspondiente.

1.6.2.2 NUnit

Es un framework de código abierto utilizado para efectuar pruebas unitarias de sistemas realizados con la plataforma .Net.

NUnit se encarga de analizar ensamblados generados por .NET, interpretar las pruebas inmersas en ellos y ejecutarlas. Utiliza atributos personalizados para interpretar las pruebas y provee métodos para implementarlas. En general este framework compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, caso contrario la prueba es exitosa.

Ventajas del Framework NUnit (Hunt, y otros, 2007):

- Pruebas unitarias automatizadas, por lo cual se hacen repetibles.
- Fomenta el cambio, ya que permite probar cambios en el código y asegurar que en éstos no se hayan introducido errores funcionales.

- Simplifica la integración, de manera que permite llegar a la fase de integración con un grado alto de confiabilidad sobre el código.
- Los defectos están acotados y fáciles de localizar.

Desventajas y limitaciones (Hunt, y otros, 2007):

- No descubrirán todos los defectos del código.
- No permite determinar problemas de integración o desempeño.
- No es trivial anticipar todos los casos especiales de entradas.
- Las pruebas unitarias determinan la presencia de defectos, no la ausencia de éstos. Son efectivas al combinarse con otras actividades de pruebas.

1.6.2.3 Spring.Net

Spring.NET es un framework de código abierto que facilita la construcción de aplicaciones empresariales en .NET. Este framework proporciona un medio consistente y transparente para configurar la aplicación e integrar la programación orientada a aspectos (AOP) en el software; permitiendo separar conceptos y minimizar la dependencia entre ellos.

Entre las ventajas de Spring.Net se puede mencionar su modularidad, pudiendo usar algunos de los módulos sin comprometerse con el uso del resto. Además brinda otras ventajas como (Mark Pollack, 2008):

- Provee componentes basados en patrones de diseño como Fábrica (Factory), Fábrica abstracta (Abstract Factory) y Constructor (Builder) que pueden ser integrados en todas las capas de la arquitectura de una aplicación.
- Soporta inyección de dependencia, permitiendo suministrar objetos a una clase en lugar de ser la propia clase quien cree el objeto.

- Brinda componentes de integración con librerías populares de mapeo relacional de objetos (ORM), como NHibernate e IBatis para .NET.

- Permite realizar las pruebas de unidad con NUnit.

1.7 Arquitectura de software

Según el documento de IEEE Std 1471-2000[15], la arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (IEEE, 2000).

La arquitectura del software se centra en los requerimientos no funcionales como la calidad del sistema, rendimiento, reutilización y capacidad de evolución.

La definición temprana de una arquitectura flexible, que no se vea fuertemente impactada ante los cambios, influye en gran medida en el éxito de un proyecto.

1.7.1 Arquitectura en Capas

La arquitectura en capas permite separar el sistema en capas, de manera que se separen las funcionalidades de cada capa de las demás. En esta arquitectura cada capa provee servicios a su capa inmediatamente superior y utiliza las prestaciones que le ofrece la inmediatamente inferior.

Ventajas (César de la Torre Llorente, 2010):

- Los cambios sólo afectan a las capas vecinas, lo que permite que soporte fácilmente la evolución del sistema.

- Se puede cambiar la implementación de una capa sin cambiar las interfaces, de manera que no se afecten las demás capas.

[15] Recommended Practice for Architectural Description of Software-Intensive Systems, Práctica recomendada para descripción arquitectónica de sistemas intensivos de software

- Al proveer una organización lógica de la aplicación facilita el mantenimiento.
- Permite aislar componentes.

Desventajas (César de la Torre Llorente, 2010):

- La separación en capas ayuda a controlar y encapsular la complejidad de aplicaciones complejas, pero agrega complejidad a las aplicaciones simples.
- Los cambios en las interfaces de una capa pueden ocasionar cambios en cascada.

1.8 Sistemas Gestores de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas que pretenden ordenar un conjunto de datos de manera clara y sencilla, de forma que posteriormente estos datos puedan ser consultados y mantenidos.

Un SGBD debe permitir (Tascón, 2006):

- Especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

1.8.1 Oracle

Oracle es considerado uno de los SGBD más confiable y seguro, lo que hace que sea muy utilizado en aplicaciones que manejan información crítica como entidades de servicios financieros y del sector gobierno. Se destaca fundamentalmente por el alto rendimiento en transacciones, su excepcional disponibilidad, escalabilidad y por ser

multiplataforma. Además puede utilizarse para el almacenamiento de todo tipo de datos como documentos, imágenes, multimedia, XML, entre otros.

Este potente gestor de base de dato brinda beneficios como (Tascón, 2006):

- Acceso más rápido y seguro a la información.
- Manejo de grandes volúmenes de información.
- Permite implementar diseños con disparadores y procedimientos almacenados, con una integridad referencial declarativa bastante potente.
- Cuenta con un buen soporte por parte de la corporación.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso admite la administración de bases de datos distribuidas.
- Gran capacidad de réplica.

Además Oracle elimina los riesgos de inseguridad al proteger los datos transmitidos desde el cliente hasta el servidor de aplicaciones y permite crear soluciones tecnológicas diseñadas para contingencias no planificadas, como fallos de sistemas.

Uno de los principales inconvenientes del uso de Oracle es el elevado costo de las patentes y licencias. Sin embargo estas licencias son más costosas al comienzo de la utilización del producto pues luego solamente cabe reactivarlas, proceso que es mucho más económico.

Oracle 10g R2 Enterprise Edition RAC: Se requiere este potente gestor dada la gran cantidad de información que se maneja y la seguridad que requiere esta información.

1.9 Calidad de Software

1.9.1 Métricas

La IEEE Std 610.12-1990[16] define que una métrica es *"una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado*". (IEEE, 1990).

Las métricas para ser efectivas deben ser simples y calculables, consistentes y objetivas, persuasivas e independientes del lenguaje de programación. Estas proporcionan una manera sistemática de evaluar la calidad a partir de un conjunto de reglas definidas con claridad, permitiendo determinar en qué medida se ajusta el software a los requisitos implícitos y explícitos del cliente.

Existe un gran número de métricas encaminadas a evaluar la calidad del software, a continuación se muestran las áreas más importantes de las métricas (Pressman, 2005).

Métricas para el modelo de análisis: Estas métricas examinan el modelo de análisis con la intención de predecir el tamaño del sistema, la funcionalidad entregada con el software y el grado en que se ha completado la especificación de los requisitos. Entre estas métricas se encuentran: funcionalidad entregada, tamaño del sistema y calidad de la especificación.

Métricas para el modelo de diseño: Estas métricas para el diseño cuantifican los atributos del diseño permitiendo medir la calidad de este. Entre las métricas para el modelo de diseño se pueden mencionar las métricas arquitectónicas, las métricas al nivel de componentes, las métricas de diseño de la interfaz y las métricas especializadas en diseño orientado a objeto.

[16] Standard Glosary of Software Engering Terms, Glosario Estándar de Términos de Ingeniería de Software

Métricas para el código fuente: Se encargan de evaluarla complejidad del código fuente y también miden la facilidad con que se mantiene y prueba el código. Entre estas se encuentran las métricas de halstead, las métricas de complejidad y las métricas de longitud.

Métricas para pruebas: Facilitan el diseño de casos de prueba efectivos y permiten evaluar la eficacia de las pruebas. Estas incluyen las métricas de cobertura de instrucciones y ramas, las métricas relacionadas con los defectos, la efectividad de la prueba y las métricas en el proceso.

1.9.2 Pruebas de Caja Blanca y Caja Negra

"*La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba*". (Pressman, 2005).

Las pruebas de caja blanca intentan garantizar que se ejecuten al menos una vez todos los caminos independientes de cada módulo, que se utilicen las decisiones en su parte verdadera y en su parte falsa, y que se utilicen todas las estructuras de datos internas.

"Las pruebas de caja negra se concentran en los requisitos funcionales del software". (Pressman, 2005). Estas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa.

Las pruebas de caja negra tienen la intensión de descubrir funciones incorrectas o ausentes, errores en la interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento, además de errores de inicialización y de terminación. Estas pruebas pretenden demostrar que las funciones del software son operativas, que las entradas se aceptan de forma correcta y que se produce una salida correcta.

1.10 Conclusiones

Luego del análisis realizado a lo largo de este capítulo se han desarrollado los principales aspectos abordados durante la investigación, se han dejado claros los conceptos más importantes a tener en cuenta para la continuidad del trabajo, por lo que se considera que los objetivos propuestos con el desarrollo de este capítulo han sido cumplidos.

De manera general, se analizaron los principios registrales que definen el sistema registral venezolano. También se han abordado las funciones fundamentales de los Registros Principales, correspondientes al proceso de solicitud de copias que se realiza en los mismos. Se seleccionó RUP como metodología de desarrollo a utilizar. Como herramientas para el modelado del diseño y para el modelado de datos se usarán el Enterprise Architect y el ER/Studio respectivamente. Como plataforma de desarrollo Microsoft .NET para el desarrollar aplicaciones de manera rápida pero a su vez segura y robusta. Además del framework .Net se utilizarán el Spring.Net, el NHibernate y el NUnit.

CAPÍTULO 2: DESCRIPCIÓN DEL SISTEMA PROPUESTO

En el presente capítulo se muestra la propuesta de solución del problema planteado, realizada con la utilización de las herramientas, metodología y tendencias analizadas en el apartado anterior. Con el fin de describir el sistema se presenta la arquitectura sobre la que estará implementado y como parte de la propuesta de solución se exponen los patrones utilizados, la descripción del proceso de solicitud de copia, el modelo de diseño, el modelo de datos y el modelo de implementación.

2.1 Arquitectura del sistema

La arquitectura utilizada para el desarrollo de la aplicación es la arquitectura en capas, esta se seleccionó teniendo en cuenta las características del software a desarrollar, así como las cualidades o atributos de calidad que se desean alcanzar.

La arquitectura en capas empleada presenta dos enfoques, uno horizontal y otro vertical. La perspectiva horizontal muestra que los sistemas de registros principales y notarías públicas se integrarán con la solución general de SAREN, conjuntamente con las soluciones que se encuentran en explotación (Registros Públicos y Mercantiles).

La perspectiva vertical abarca la distribución de los componentes que dan vida a la arquitectura para cada módulo de manera individual. En este sentido se puede apreciar en la figura 2 que se presenta un modelo multicapa, donde las capas tienen un orden lógico de procedencia, es decir las funcionalidades y recursos están encapsulados en cada nivel de acuerdo con la responsabilidad que se establece para cada uno de ellos. (Proyecto Registros y Notarías, 2011). Este modelo multicapa está constituido por las capas Presentación, Negocio, Acceso a Datos y Dominio, como se ilustra a continuación.

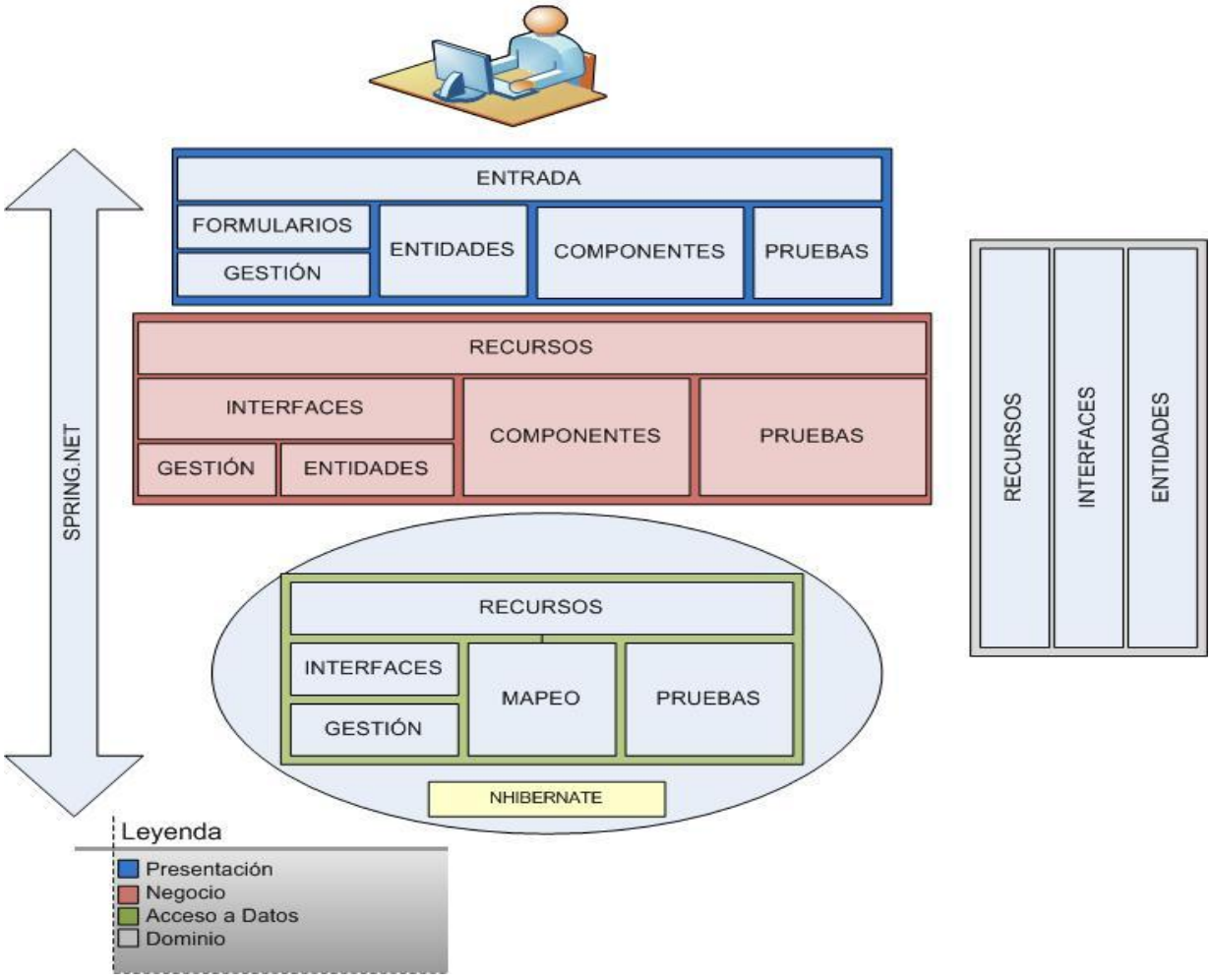

Fig. 2: Representación del enfoque vertical de la arquitectura.[17]

A continuación se describen brevemente cada una de las capas que conforman la arquitectura.

Nivel Presentación: Es la capa que contiene los componentes con los que interactuará el usuario. Esta tiene acceso a las funcionalidades que brinda el negocio y puede mostrar o capturar la información a través de los diferentes elementos que incluye. En este nivel se ubican los mecanismos de inicialización y configuración de algunos componentes radicados en la Arquitectura Base, los formularios, las acciones que se asocian a estos formularios, clases para facilitar el trabajo en esta capa, así

[17] Extraído del documento Arquitectura de Software de RN, elaborado por los arquitectos del proyecto.

como clases diseñadas para realizar pruebas de unidad a los diferentes bloques de código de este nivel.

Nivel de Negocio: Esta capa se encarga de recibir una petición de la capa superior, gestionar o procesar la misma, de ser necesario solicitándola a la capa de Acceso a Datos y finalmente envía la respuesta a quien realizó la petición. Contiene clases que son útiles para el negocio (Entidades), clases encargadas de resolver determinadas funcionalidades correspondientes a los casos de usos de un módulo (Gestores), las interfaces para ofrecer funcionalidades que van a representar un nivel de abstracción y clases diseñadas para realizar pruebas de unidad a los diferentes bloques de código de este nivel.

Nivel de Acceso a Datos: Es la capa inferior, por lo que los componentes que contiene desconocen los niveles superiores. Este nivel se limita al manejo de la información, ya sea para persistirla u obtenerla para su procesamiento y propagación por la aplicación. Contiene las clases que implementan la totalidad de las operaciones de persistencia y obtención de datos (DAOs[18]), las interfaces que representan las funcionalidades que brinda esta capa, el mapeo, así como las configuraciones y estructuras que presenta la capa para el trabajo correcto de determinadas funcionalidades para la persistencia.

Nivel de Dominio: Esta capa está distribuida verticalmente, ya que contiene todas las entidades y recursos que se manejan en el dominio completo del sistema.

Después de describir brevemente cada una de las capas que componen el sistema, se representa en la figura 3 la vista lógica para mayor entendimiento de la arquitectura.

[18] Siglas en ingles de Objetos de Acceso a Datos.

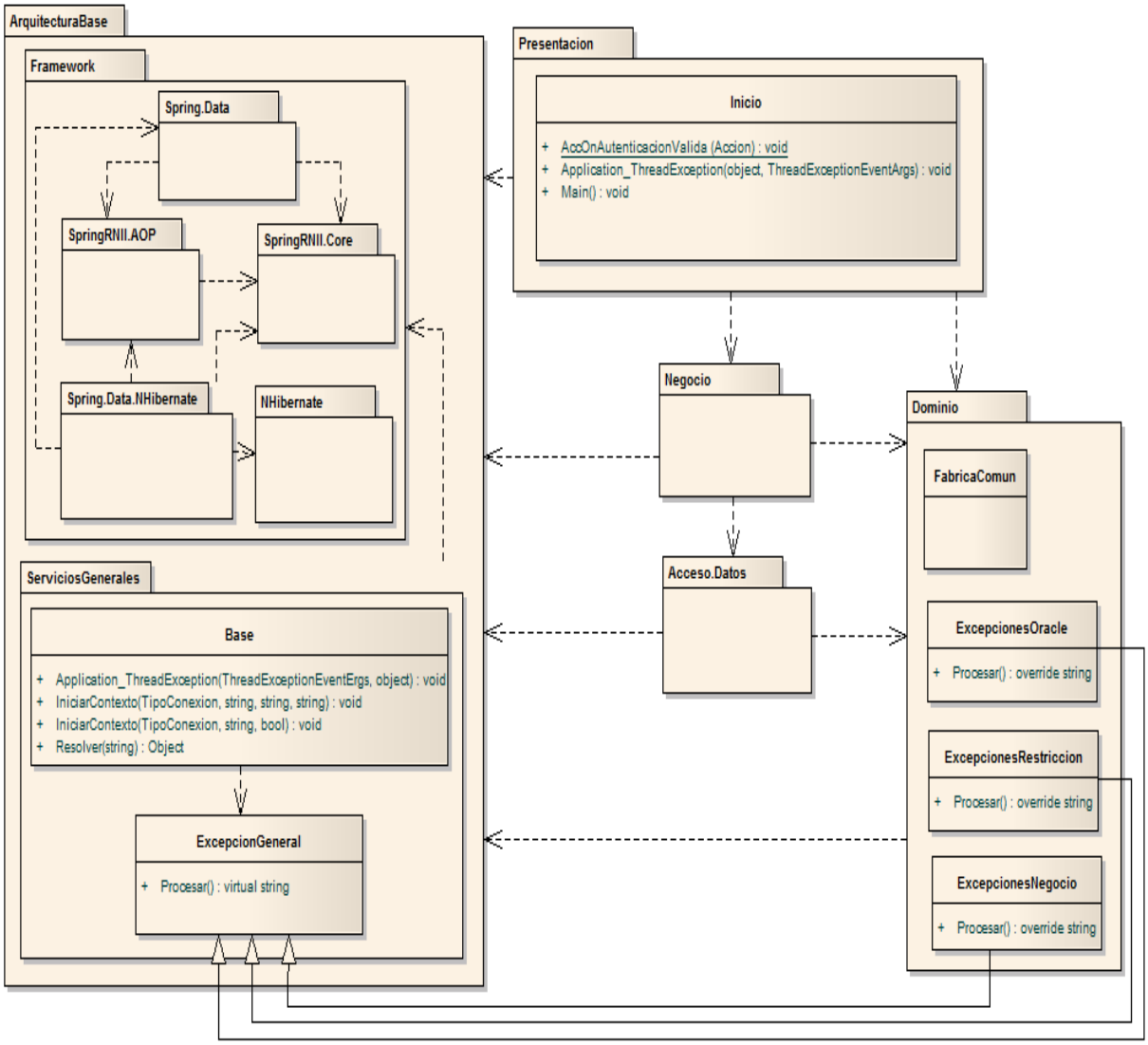

Fig. 3: Vista lógica general.

En la vista lógica se representan las relaciones existentes entre las diferentes capas y la Arquitectura Base, así como entre los componentes del framework Spring.Net y NHibernate.

La Arquitectura Base utiliza varios componentes de Sprint.Net para aspectos como inicializar el contexto, administrar transacción, realizar la integración con NHibernate e inyectar dependencias. Algunos de estos componentes de Spring.Net fueron

modificados con el objetivo de adecuar sus funcionalidades a las necesidades del sistema, obteniendo como resultado: "*SpringRNII.AOP*" y "*SpringRNII.Core*".

En los "*ServiciosGenerales*" se encuentra la clase "*Base"* la cual en integración con Spring.Net, gestiona el manejo de excepciones ("*Application_ThreadException"*), además dicha clase se encarga de cargar toda la configuración para que la arquitectura del sistema trabaje adecuadamente ("*Iniciar Contexto"*) y la funcionalidad "*Resolver"* permite realizar la instanciación de los diversos objetos mediante el patrón Fábrica Abstracta.

Además se evidencia el trabajo con los diferentes tipos de excepciones en la capa de dominio, las cuales heredan de la clase "*ExcepcionGeneral*" de la Arquitectura Base.

2.2 Descripción del proceso Solicitud de Copias

2.2.1 Solicitud de Copias

Al funcionario comenzar este proceso en el sistema, aparece una primera interfaz con la opción de crear un nuevo trámite o buscar uno ya existente para su posterior modificación. Si se desea crear un nuevo trámite se debe buscar en el archivo digital el documento al que se desea hacer la copia o adicionar el documento del archivo físico, en caso que el documento no se encuentre digital.

El sistema da la posibilidad de que se realice la solicitud de copia de varios documentos a la vez. Además se debe elegir para cada documento el tipo de acto (Copia Simple, Copia Certificada Fotostática, Copia Certificada Mecanografiada), los recaudos correspondientes al acto seleccionado y la cantidad de copias que se desean realizar.

A continuación en caso que alguno de los documentos a los que se le va a realizar copia sea físico, debe ser digitalizado. Posteriormente se debe seleccionar si es una solicitud realizada por una Persona Natural o por Oficio Institucional. En caso que sea una solicitud realizada por oficio se deben introducir los datos requeridos, de lo

contrario debe asociarse una persona al trámite que se está creando, para ello en el sistema se deben introducir la cédula o el pasaporte para realizar la búsqueda del ciudadano en la Base de Datos.

Luego el sistema muestra una interfaz de usuario que permite seleccionar las exenciones a aplicar al trámite en curso, en caso que este tenga exenciones asociadas.

Finalmente el sistema crea un trámite por cada documento que se solicitó copiar, generando para cada uno el número del trámite y la fecha de creación, por último se actualiza el estado de cada uno de los trámites, además el sistema genera y muestra la planilla de solicitud de copia.

2.2.2 Cálculo

Para realizar el cálculo lo primero que se debe hacer es seleccionar el trámite al que se le va a realizar el cálculo, del que se muestra posteriormente un resumen del acto, el solicitante, las exenciones aplicadas y los recaudos. Se visualiza entonces los datos de la persona u oficio asociado al trámite y se da la posibilidad de cambiarlo en caso necesario. El funcionario de cálculo especifica los conceptos de pago del trámite según el tipo de acto y luego el sistema genera la PUB y la Planilla de Cálculo.

2.2.3 Presentación

Primeramente se debe seleccionar el trámite y posteriormente se procede a gestionar el pago de la PUB, donde se visualizan los datos de la planilla y se permite la gestión del pago de la misma, ya sea en efectivo o cheque, por internet o por punto de venta. Posteriormente el sistema muestra una interfaz que visualiza todos los recaudos asociados al trámite, dando la posibilidad de gestionar y digitalizar estos. Se muestra entonces los datos de la persona u oficio asociado al trámite y se da la posibilidad de cambiarlo en caso necesario. El sistema genera la fecha de presentación, la posible fecha de otorgamiento y el recibo de presentación.

2.2.4 Procesamiento

Para comenzar este proceso debe elegirse el trámite a ser procesado. Luego se muestra una interfaz con los datos del documento a copiar. Estos datos dependen del tipo de documento al que se le vaya a realizar la copia, los mismos pueden ser:

- Actas
  - Acta de nacimiento
  - Acta de defunción
  - Acta de matrimonio
- Documento duplicado
- Documentos protocolizados
  - Sentencias
  - Títulos Universitarios
  - Actas Constitutivas y de Asambleas

Posteriormente el sistema puede realizar varias operaciones en dependencia del tipo de acto:

- Copia Simple: Se muestra el documento al que se le va a realizar la copia.
- Copia Certificada Fotostática: Se muestra el documento al que se le va a realizar la copia y se genera la nota de certificación.
- Copia Certificada Mecanografiada: Se muestra el documento, se da la posibilidad de transcribir el documento al que se le va a realizar la copia y se genera la nota de la copia mecanografiada.

2.2.5 Revisión de Procesamiento

El funcionario de revisión selecciona el trámite y posteriormente revisa el documento. El sistema si el acto es Copia Certificada Fotostática muestra la nota de certificación y si es Copia Certificada Mecanografiada la nota de la copia mecanografiada. A

continuación se muestra un resumen del trámite y se permite dar el documento por revisado o dejarlo pendiente.

2.2.6 Firma Digital

Para realizar la firma digital lo primero que se debe hacer es seleccionar el trámite al que se le va a firmar sus documentos. A continuación se muestra un resumen del trámite y se permite firmar digitalmente el documento que contiene los recaudos y el documento original.

2.2.7 Otorgamiento/Entrega

Primeramente se selecciona el trámite a continuación se muestra la nota de certificación en caso de que el acto seleccionado haya sido Copia Certificada Fotostática de lo contrario si es una nota mecanografiada se muestra la Copia Certificada Mecanografiada. Para culminar el documento se entrega al usuario o se deja pendiente de entrega si existe algún problema con el mismo.

2.3 Diseño e Implementación

En el diseño se modela el sistema, de forma que soporte todos los requisitos y las restricciones que se le suponen. El modelo de implementación describe cómo los elementos obtenidos del modelo de diseño se implementan en términos de componentes, proporcionándole una ubicación a estos en los nodos físicos en los que funcionará la aplicación.

A continuación se explican de manera general dichos modelos y se ilustran algunos diagramas generados durante el diseño.

2.3.1 Modelo de Diseño

El modelo de diseño es un modelo de objetos que surge a través de los resultados obtenidos del modelo de análisis, el cual proporciona una comprensión detallada de

los requisitos y que a su vez sirve como abstracción del modelo de implementación y su código fuente. Incluye los diagramas de clases y de interacción.

2.3.1.1 Diagramas de clases

Durante el flujo de trabajo de análisis y diseño los diagramas de clases desempeñan un papel fundamental ya que los mismos proporcionan un mejor entendimiento de la estructura estática del sistema mostrando las clases, sus atributos y las relaciones que se establecen entre ellas.

Dentro de los principales elementos que se pueden encontrar en un diagrama de clases de diseño se encuentran: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de atributos, navegabilidad y dependencias.

2.3.1.1.1 Diagramas de clases del proceso de solicitud de copias

Los diagramas de clases del proceso de solicitud de copias se dividen en la aplicación en siete procesos o subprocesos diferentes como se explicó en el epígrafe *2.2*-"*Descripción del proceso Solicitud de Copias*". Cada uno de estos procesos se divide en varios casos de usos y para cada caso de uso se realiza un diagrama de clases.

El diagrama de clases que se muestra a continuación es el correspondiente al caso de uso "*Gestionar Trámite de Solicitud*". Para mayor entendimiento de este diagrama se mostrará en la figura 4 las entidades e interfaces de dominio relacionadas directamente con este caso de uso, en la figura 5 los DAOs e interfaces de la capa acceso a datos, en la figura 6 los gestores e interfaces del negocio y en la figura 7 se mostrarán las relaciones establecidas entre las diferentes clases de la aplicación para realizar este caso de uso.

A continuación se evidencian las entidades e interfaces del dominio que permiten manejar los datos relacionados con el proceso de solicitud de copias, representadas

por <*EDNombreEntidad*> y <*IDNombreInterfaz*>. Además se muestran las relaciones que se establecen entre dichas entidades e interfaces.

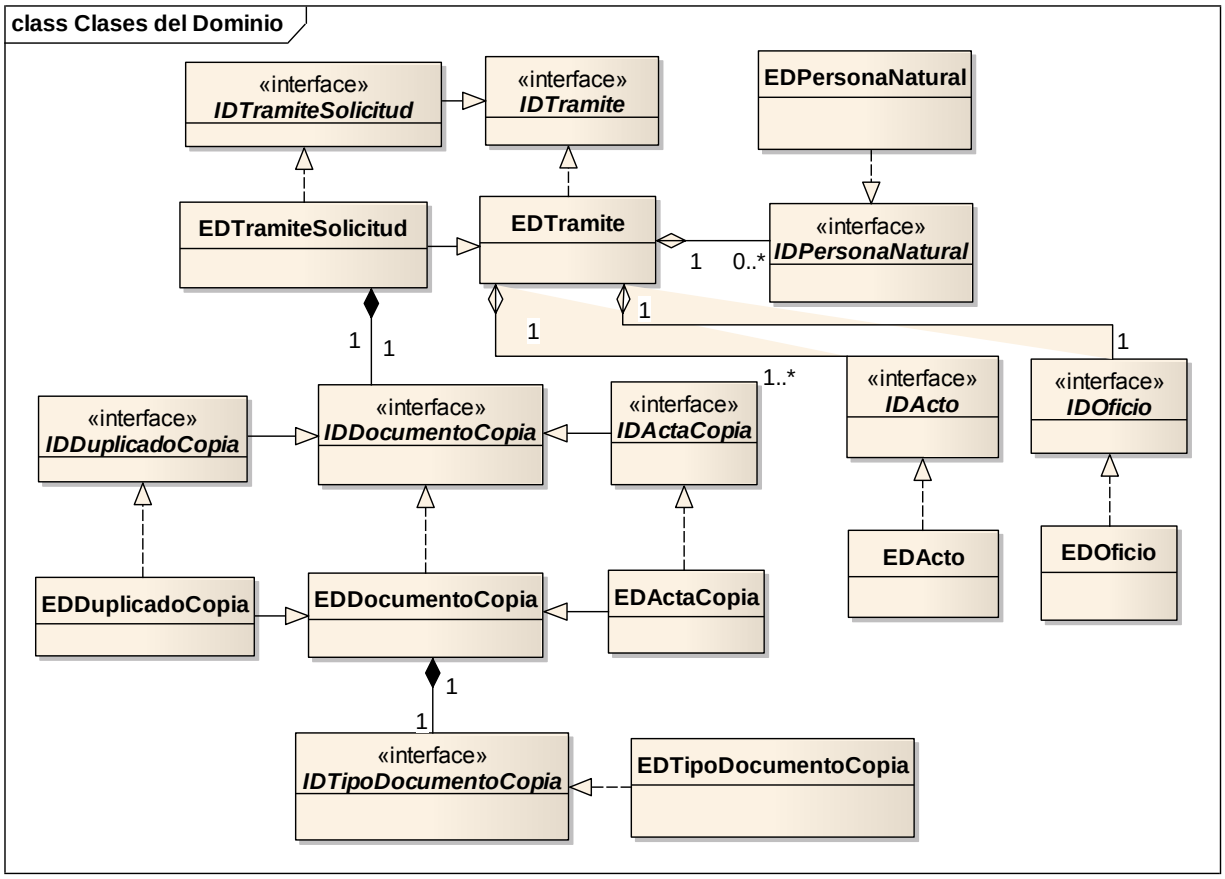

Fig. 4: Clases del dominio relacionadas con el caso de uso Gestionar Trámite de Solicitud.

Las entidades e interfaces mostradas anteriormente van a permitir manejar los datos relacionados con los trámites de solicitud, como son: el solicitante que puede ser una persona natural o un oficio y el acto asociado a dicho trámite. Además a través de estas se podrán manipular los datos del documento al que se desea realizar una copia, ya sea este documento un acta, un documento duplicado o un documento protocolizado. Es importante aclarar que al estar estas clases ubicadas en el capa dominio, se tendrá acceso a ellas desde toda la aplicación como indica la arquitectura del sistema.

En la siguiente imagen se muestran las clases que implementan la totalidad de las operaciones de persistencia y obtención de datos (<*DaoNombreDao*>) y sus respectivas interfaces (<*IDaoNombreDao*>).

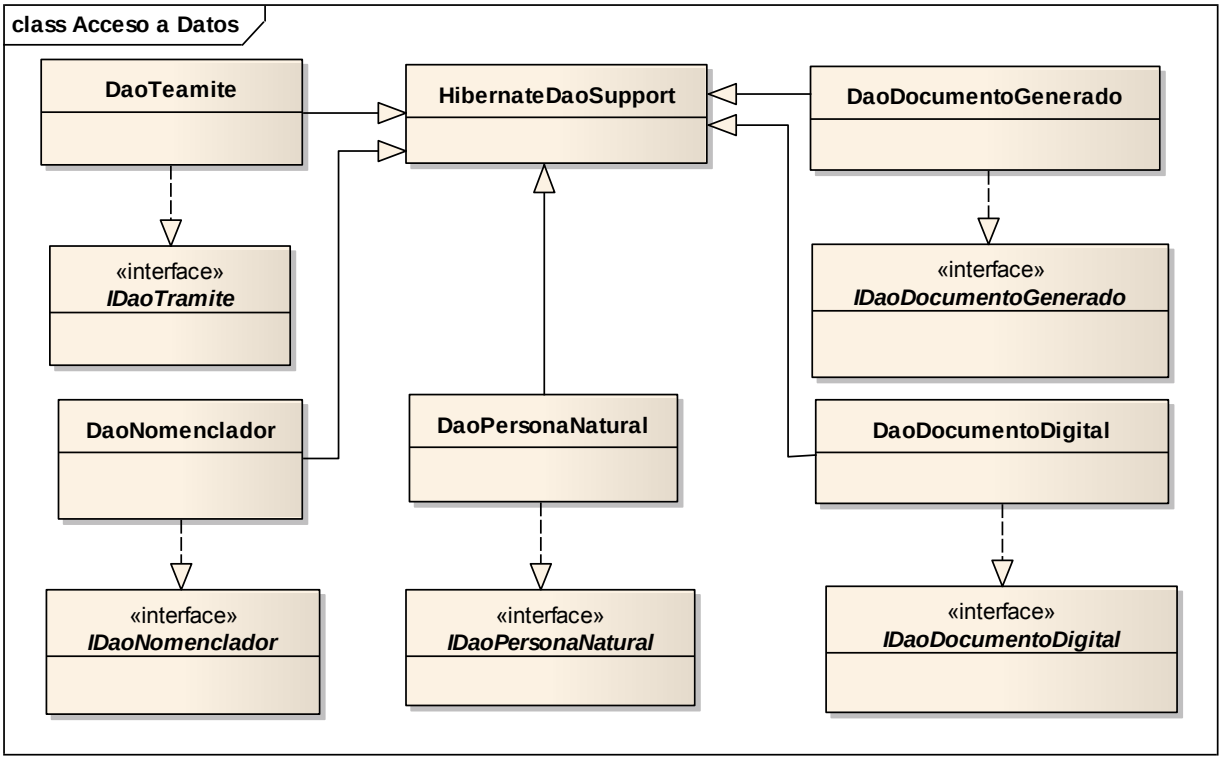

Fig. 5: Clases de la capa Acceso a Datos relacionadas con el caso de uso Gestionar Trámite de Solicitud.

Se puede apreciar en la imagen anterior que todos estos DAOs heredan de "*HibernateDaoSupport*", que es una clase que provee Spring.Net para brindarle a los DAOs soporte para NHibernate. Esta clase ofrece varias utilidades para el manejo adecuado de transacciones.

Además se evidencia que las operaciones de persistencias están divididas en varios DAOs según su función, es decir existe el "*DaoTramite*" que es el encargado de la persistencia de los datos relacionados con el trámite, el "*DaoNomenclador*" que se utiliza para obtener los valores de los nomencladores que se le mostrarán al usuario, el "*DaoPersonaNatural*" que es utilizado para la búsqueda y actualización de los datos

de las personas, el "*DaoDocumentoDigital*" que se encarga de todo lo relacionado con la digitalización y obtención de documentos digitales y por último el "*DaoDocumentoGenerado*" que permite salvar y cargar las notas según sea el caso. Estas clases se encuentran en la capa de Acceso a Datos, pues son las que controlan todo lo concerniente a la información que se encuentra en la fuente de almacenamiento.

A continuación se muestran los gestores (<*GtrNombreGestor*>), que son las clases que tienen como objetivo resolver determinadas funcionalidades de los casos de uso y sus interfaces (<*IGtrNombreInterfazGestor*>).

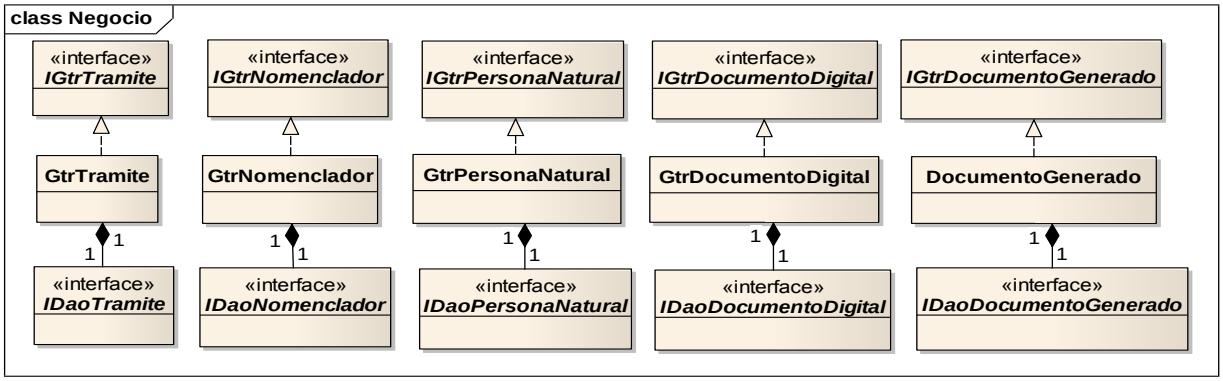

Fig. 6: Clases de la capa Negocio relacionadas con el caso de uso Gestionar Trámite de Solicitud.

Los gestores mostrados anteriormente están creados según la necesidad del diseño y al igual que los DAOs están divididos de forma que cada uno realice sólo las funciones para las que fue creado. Además se puede evidenciar que cada gestor contiene un DAO al que de ser necesario le puede realizar peticiones, mostrándose de esta forma que la capa de Negocio puede realizar peticiones a la capa de Acceso a Datos para responder a las solicitudes realizadas por la capa de Presentación.

En la figura 7 se reflejan las relaciones entre las clases existentes en el diagrama de clases del caso de uso "*Gestionar Trámite de Solicitud"*, el cual está compuesto por las entidades de dominio y sus interfaces, las clases DAOs y sus interfaces, las clases gestoras de negocio y sus interfaces, las clases controladoras de interfaz (*<AccNombreAccion>)* y las clases interfaz (<*FrmNombreFormulario*>).

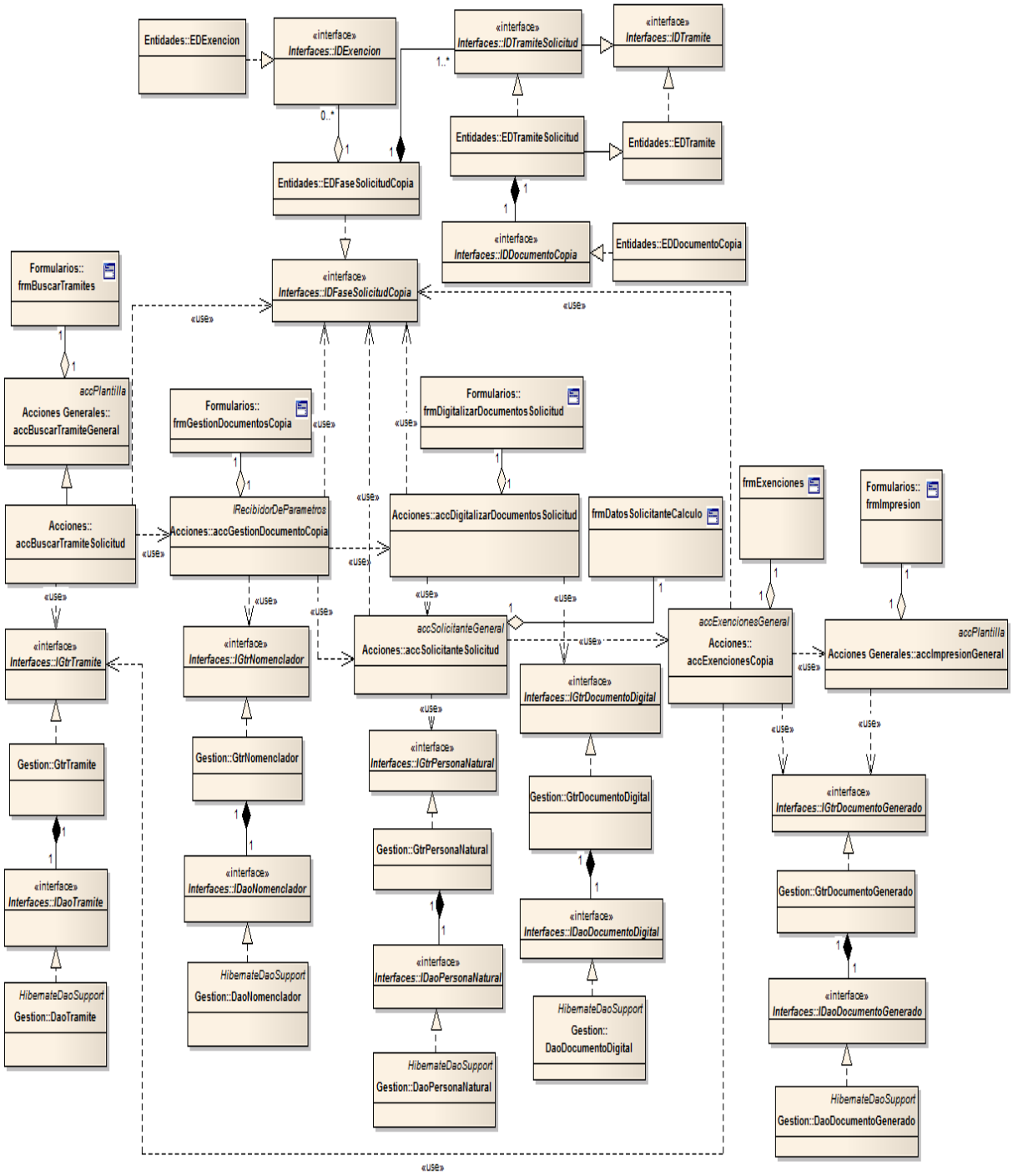

Fig. 7: Diagrama de clases del caso de uso Gestionar Trámite de Solicitud.

En esta imagen se puede apreciar que existe la entidad "*EDFaseSolicitudCopia*", que será la que contendrá los datos del trámite de solicitud. Esta clase va a ser utilizada

por todas las acciones, ya que dicha clase es la que va a navegar a través de las acciones conteniendo los datos relacionados con el trámite en curso.

Además se observa que la lógica de presentación de los formularios de interfaz de usuario se encuentra separada por acciones asociadas a los mismos, brindando así mayor flexibilidad, modificabilidad y mantenibilidad al software.

Dichas acciones podrán realizar peticiones a los gestores, quienes gestionarán o procesarán la misma, de ser necesario haciéndole una solicitud al DAO y finalmente enviarán una respuesta a la acción que realizó dicha petición. Esto evidencia la relación existente entre la capa de presentación (Acciones), la capa de negocio (Gestores) y la capa de acceso a datos (DAOs).

Debido a la gran cantidad de diagramas desarrollados durante el diseño se decidió no abarrotar este documento con imágenes por lo que sólo se mostró el diagrama de clases del caso de uso "*Gestionar Trámite de Solicitud"*. Para más información de los diagramas de clases existentes en el proceso de solicitud de copias se recomienda ver el documento Modelo de Diseño que debe estar adjunto a este documento.

2.3.1.2 Diagramas de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. Estos diagramas se pueden expresar en diagramas de secuencia y en diagramas de colaboración. Los primeros muestran las interacciones entre los objetos mediante transferencias de mensajes entre objetos o subsistemas (Jacobson, 2000) y los segundos destacan la organización de los objetos que envían y reciben mensajes. En este caso se utilizará el diagrama de secuencia.

A continuación se muestra el diagrama de secuencia correspondiente al escenario "*Adicionar del Archivo Físico"* del caso de uso "*Gestión Documento Copia"*, que es incluido del caso de uso "*Gestionar Trámite de Solicitud"*, de este fue del que se mostraron los diagramas de clases anteriormente.

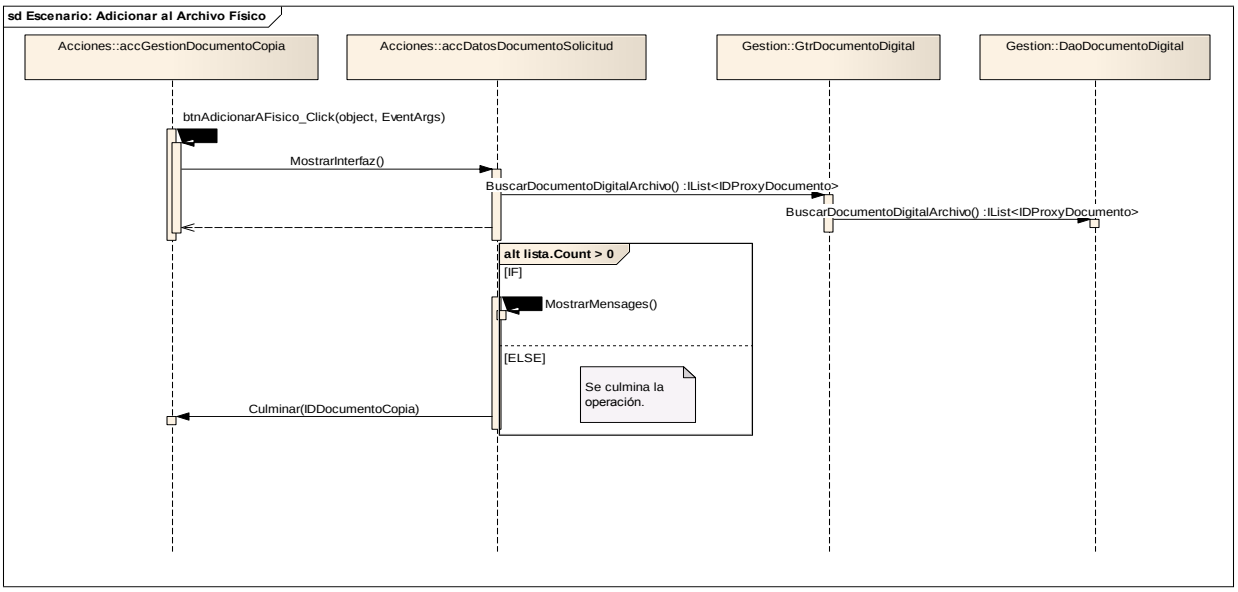

Fig. 8: Diagrama de secuencia correspondiente al escenario Adicionar del Archivo Físico.

Este diagrama de secuencia muestra como adicionar un documento del archivo físico al archivo digital, en caso que este no exista en el archivo digital. Luego de ser adicionado el documento al archivo digital se podrá proceder a realizarle una copia.

Para más información de los diagramas de secuencia existentes en el proceso de solicitud de copias se recomienda ver el documento Modelo de Diseño que debe estar adjunto a este documento.

2.3.2 Modelo de Datos

El modelo de datos es usado para describir la lógica y física de la información persistente manejada por el sistema. Este es creado a partir de las clases persistentes y contienen tres elementos fundamentales: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos; y las relaciones, que son las que enlazan a dichos objetos entre sí.

A continuación de muestra una porción del modelo de dato del proceso de solicitud de copia.

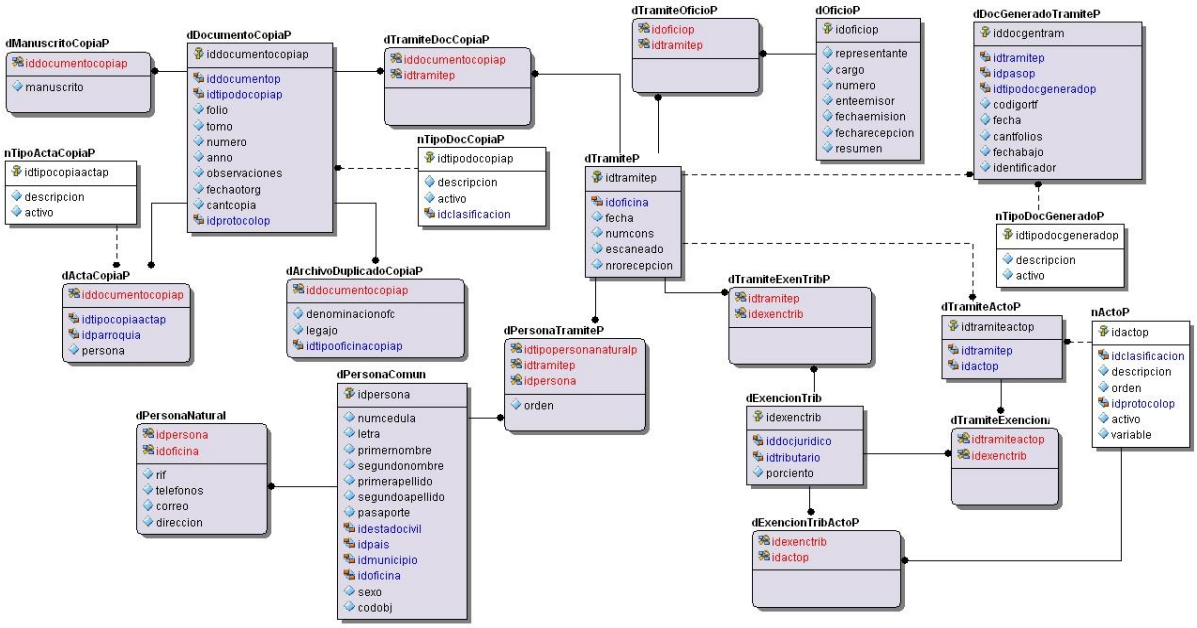

Fig. 9: Modelo de datos del proceso de Solicitud de Copias.

En la imagen anterior de muestran las tablas de la base de dato que almacenarán la información del trámite de solicitud, es decir trámite, persona, oficio, acto, exenciones y documentos generados. También se reflejan las tablas en las que se almacenarán los datos del documento a copiar y la relación de este con el trámite.

2.3.3 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes y la estructura del sistema en ejecución. Este modelo está conformado por los diagramas de despliegue y componentes.

2.3.3.1 Diagramas de Componentes

El diagrama de componente muestra la organización y las dependencias entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, binarios o ejecutables. Este diagrama es utilizado para representar la vista estática de un sistema.

En el sistema propuesto se elaboraron varios diagramas de componentes para facilitar el entendimiento del mismo, es decir se creó un diagrama de componentes para cada capa y uno que evidencia de forma general el sistema.

A continuación se representa el diagrama de componentes que muestra de forma general el sistema propuesto:

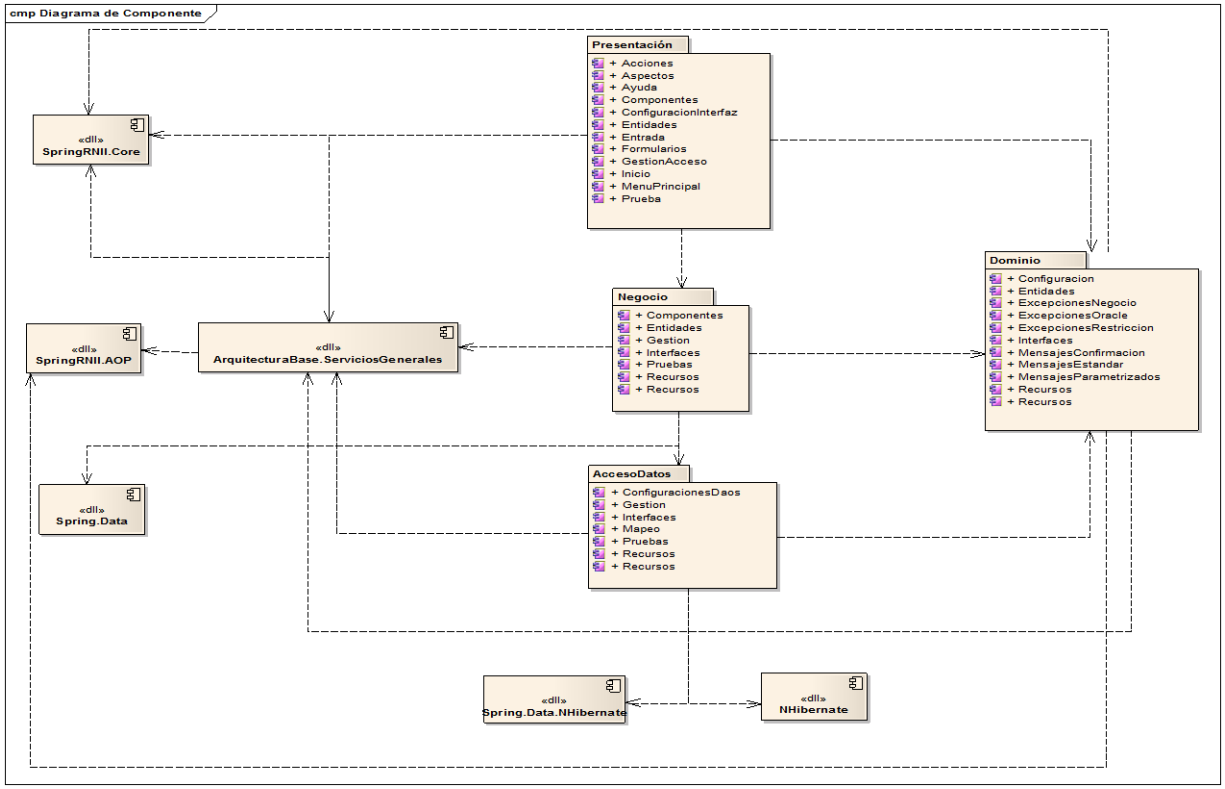

Fig. 10: Diagrama de componentes del sistema propuesto.

Como se ve en la figura todas las capas de la aplicación utilizan los servicios generales de la Arquitectura Base, así como estos utilizan el "*SpringRNII.Core"* y "*SpringRNII.AOP"*, para aspectos como inicializar el contexto, manejar las excepciones e inyectar dependencias. Además el dominio utiliza directamente el "*SpringRNII.Core" y "SpringRNII.AOP"* para inicializar el contexto de las mensajerías y el trabajo con las excepciones; la capa de presentación "*SpringRNII.Core" para* inicializar el contexto de la aplicación; la capa de negocio el "*Spring.Data"* para el manejo de transacciones y la capa acceso a datos depende del "*NHibernate"* y el

"*Spring.Data.NHibernate"* para realizar la persistencia de los datos. El resto de las relaciones se obtienen de la representación vertical de la arquitectura.[19]

A continuación se muestran los diagramas de componentes pertenecientes a cada una de las capas.

En la capa de presentación se encuentra la clase "*Inicio"* que constituye el punto de partida de la aplicación, en esta se establecen las configuraciones necesarias para el correcto funcionamiento del sistema, como son el manejo de excepciones a través de la Arquitectura Base, las configuraciones de los aspectos reflejadas en el componente "*Aspectos.xml*" y el acceso al sistema según lo establecido en "*GestionAcceso.xml*". Además se realizan las configuraciones para el correcto funcionamiento de la ayuda en *"Ayuda.xml"* y se establecen las opciones del menú en "*MenuPrincipal.xml*".

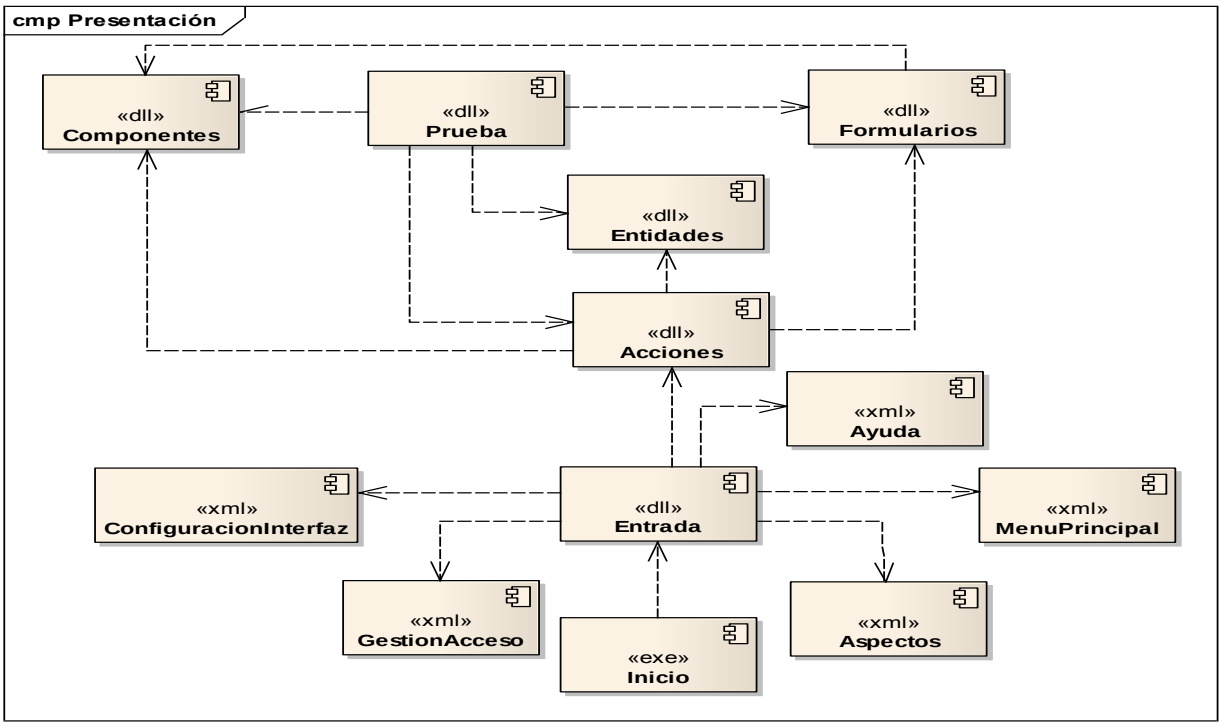

Fig. 11: Diagrama de componentes de la capa presentación.

[19] En el diagrama de componente no se representaron las relaciones existentes entre los componentes de los frameworks, para facilitar el entendimiento de este diagrama, ya que estas relaciones se evidenciaron en la vista lógica de la arquitectura.

En la capa de negocio se ubica "*Recursos.xml*" que es donde se configura la instanciación de los DAOs a través de la inyección de dependencia.

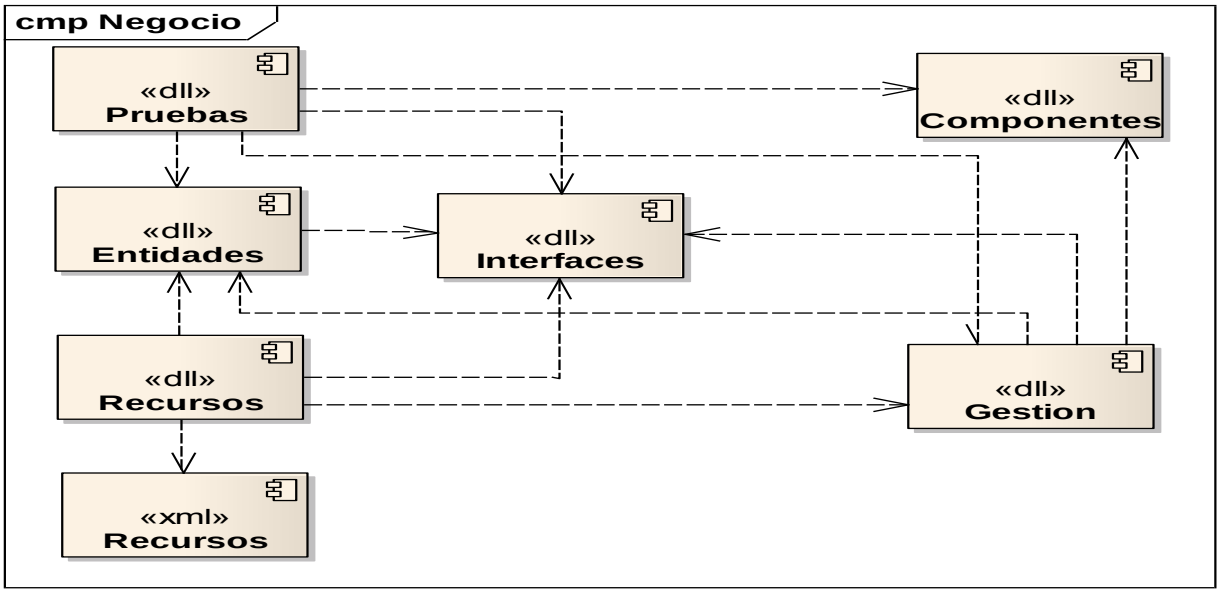

Fig. 12: Diagrama de componentes de la capa negocio.

La capa acceso a datos contiene a "*ConfiguracionesDaos.xml*", en el que se establece las configuraciones para la integración de NHibernate con Spring.Net. Además en esta se observa "*Recursos.xml*" que es donde se realiza la inyección del "*HibernateTemplate*".

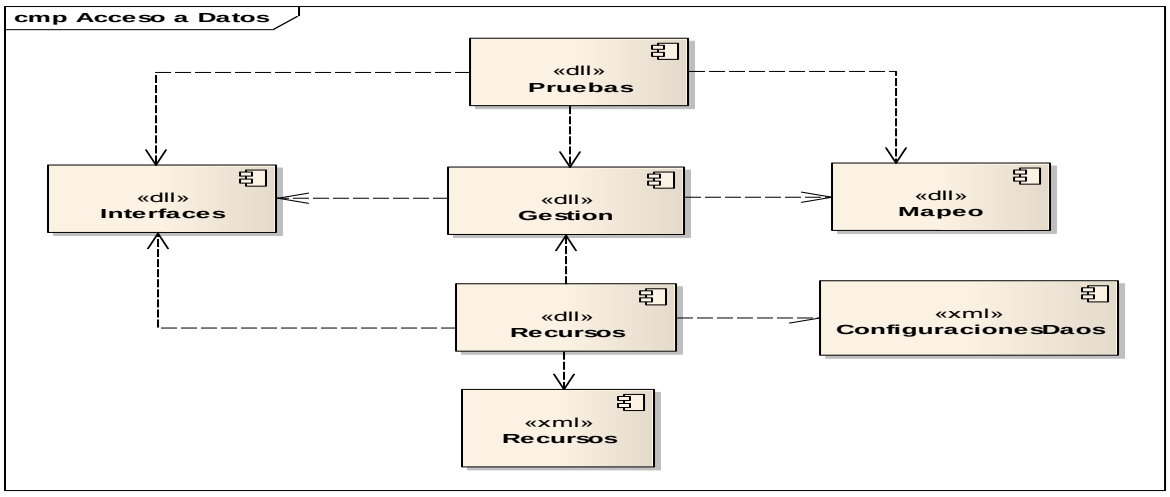

Fig. 13: Diagrama de componentes de la capa acceso a datos.

En el dominio se encuentra "*Configuracion.xml*" que es donde se realizan las configuraciones necesarias para el funcionamiento de las mensajerías ("*MensajesConfirmacion.resx*", "*MensajesParametrizados.resx*" y "*MensajesEstandar.resx*") y "*Recursos.xml*" que es donde se configura porqué fábrica y porqué método va a ser creada cada entidad. Además en este nivel se encuentran ficheros que contienen los diferentes tipos de excepciones, como son: "*ExcepcionesNegocio.resx*", "*ExcepcionesOracle.resx*" y "*ExcepcionesRestriccion.resx*".

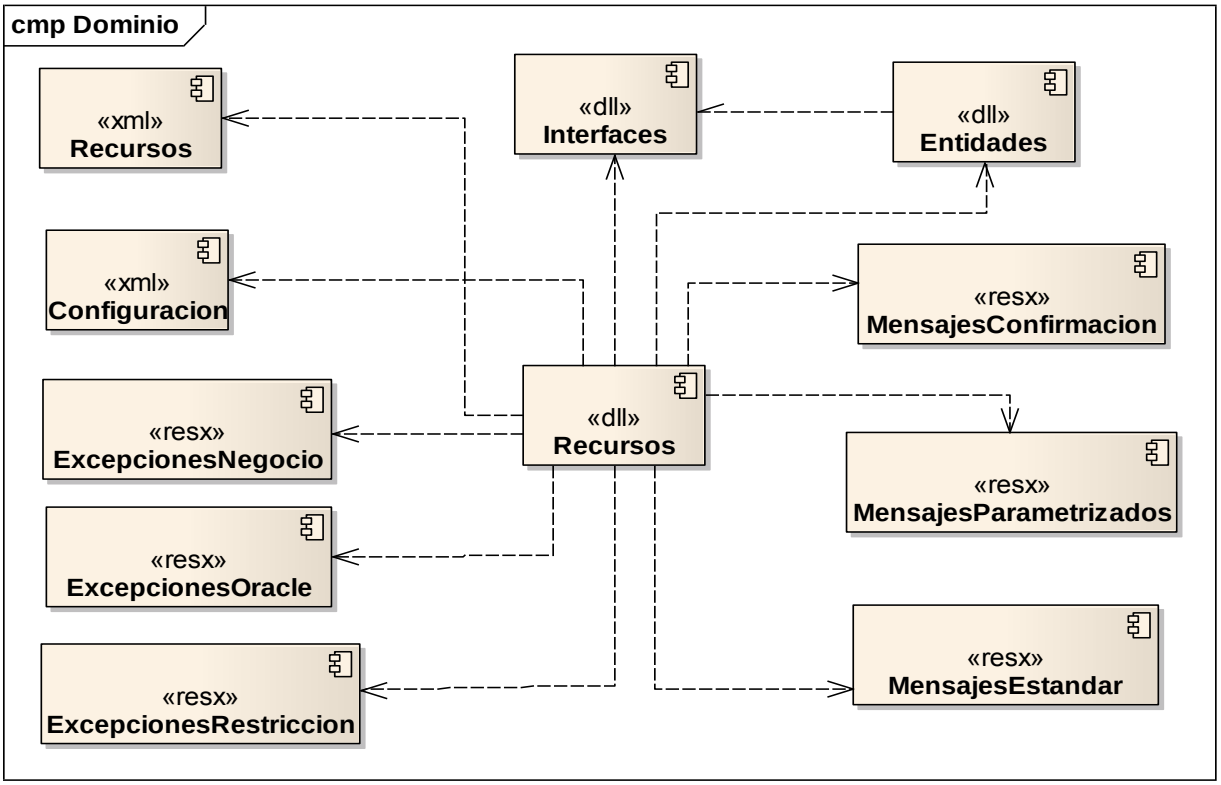

Fig. 14: Diagrama de componentes de la capa dominio.

2.3.3.2 Diagrama de Despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue, representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo

es un recurso de ejecución tal como un computador, un dispositivo o memoria.

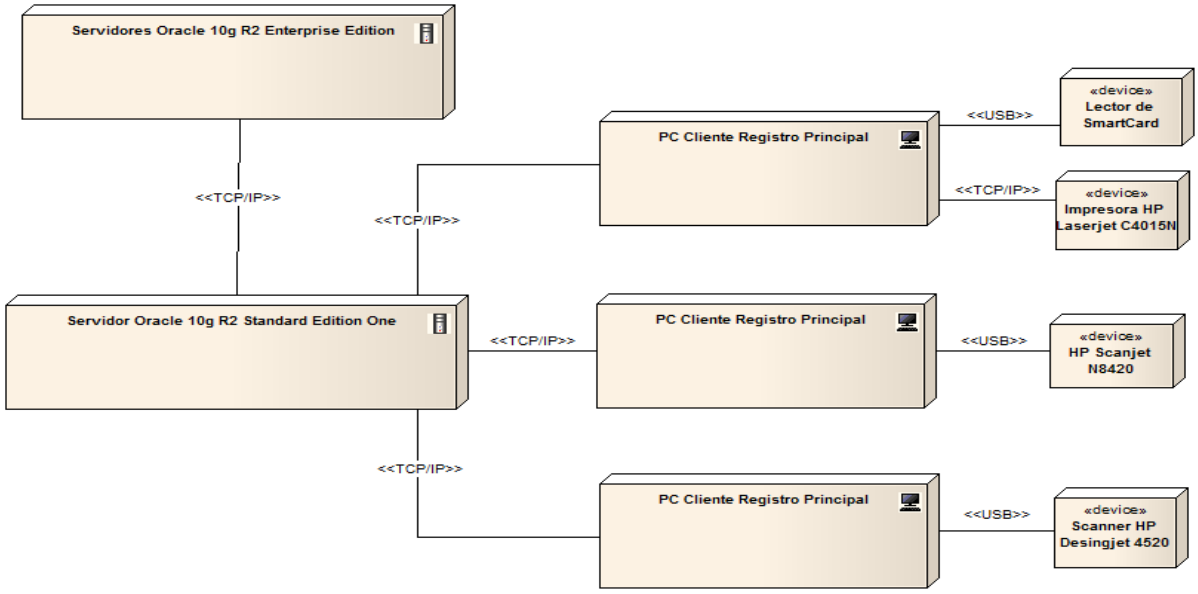

Fig. 15: Diagrama de despliegue del sistema de Registros Principales. (Ver anexo 1)

En el diagrama de despliegue se observa que el servidor de base de datos central se encuentra ubicado en el Centro de Datos el mismo tendrá un alcance a nivel nacional, que permitirá el intercambio de información con los servidores locales de las oficinas. Las máquinas clientes se conectan al servidor local mediante red inalámbrica, estas tendrán a su disposición un Scanner, un lector de SmartCard, un Plotter o una impresora láser. Para aumentar la seguridad del sistema se realizará filtrado MAC[20] en el AP[21] y por direcciones IP en el servidor de base de datos.

A continuación se muestra una breve descripción de los dispositivos que intervienen en el diagrama de despliegue ilustrado anteriormente.

Impresora: Este dispositivo garantiza las funcionalidades de impresión de reportes y documentos que genera la aplicación dependiendo de las funcionalidades de la máquina cliente que la utilice a través de la red.

[20] Media Access Control, Control de Acceso al Medio
[21] Access Point, Punto de Acceso

PC Cliente: Este nodo se corresponde con la máquina donde se encuentra instalado el sistema de Registros Principales con las funcionalidades que se correspondan a cada puesto de trabajo.

Servidor Oracle 10g R2 Standard Edition One: Este es el servidor de base de datos que se encuentra en cada Registro Principal. Está montado sobre el Sistema Operativo Windows Server 2008 y es el nodo al que la máquina cliente encuesta y solicita información.

Servidores Oracle 10g R2 Enterprise Edition: Este es el clúster de servidores de base de datos que se encontrará en el Centro de Datos. Está montado sobre el Sistema Operativo RedHat AS v4.0 y en este es donde se encontrará almacenada la totalidad de la información del sistema de todas las oficinas.

Scanner HP Designjet 4520: Dispositivo que se utiliza en los escaneos de mapas, títulos y documentos de gran tamaño.

HP Scanjet N8420: Dispositivo que se utiliza en los Registros Principales para la digitalización de documentos de tamaño normal.

2.4 Patrones utilizados

Durante el desarrollo de la aplicación se utilizaron patrones, con el objetivo de promover la reutilización, facilitar la modificación y aumentar la comprensión del software construido. A continuación se realiza una breve descripción de la aplicación de algunos de estos patrones.

Fachada: De manera general se trabaja con este patrón en todo momento, lógicamente haciendo uso del framework Spring.Net pero a través de la Arquitectura Base. Se utiliza en la aplicación para abstraer al programador de la complejidad de las configuraciones requeridas por Spring.Net para aspectos como el manejo de excepciones y la instanciación de objetos. De esta manera la instanciación de un objeto por el programador quedaría de la siguiente forma:

```
gtrTramite = Base.Resolver<IGtrTramite>();
```

Fig. 17: Ejemplo de instanciación de objetos en la aplicación.

Donde *"Base"* es la clase principal radicada en la Arquitectura Base diseñada para resolver los objetos y sus configuraciones en el contexto.

Interceptor: En la aplicación se utilizan los interceptores del framework Spring.Net con el objetivo de realizar el análisis de métodos para determinar si necesitan algún tratamiento especial antes, durante y después de su ejecución. Este patrón se utiliza en la administración de transacciones, de forma que se establece un punto de cruce especificando que método es transaccional y por tanto necesita un tratamiento especial. A continuación se muestra la imagen de un método transaccional.

```
[Transaction]
public IDPersonaJuridica SalvarActualizarPersonaJuridica(IDPersonaJuridica objPersonaJuridica)
{
      return DaoPersonaJuridica.SalvarActualizarPersonaJuridica(objPersonaJuridica);
}
```

Fig. 18: Ejemplo de método transaccional.

Patrón Método Plantilla: Según (Gamma, 2002), este patrón define el esqueleto de un algoritmo dejando las especificidades a las subclases y permitiendo que estas redefinan ciertos pasos del algoritmo.

Este patrón es uno de los más utilizados para la reutilización de una misma vista por uno o más controladores, acciones según las define el framework. De esta forma el controlador base podría definir el comportamiento para las operaciones comunes y dejaría a las subclases que respondieran con sus propias implementaciones a las que debería realizar.

Patrones Grasp aplicados.

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades. En nuestra aplicación podemos observar que tanto en la capa de acceso a datos como en la capa de negocio se diseñaron varios DAOs, y gestores respectivamente con el objetivo de garantizar que cada cual se encargara de realizar sus funciones y operaciones específicas en dependencia de la información que cada cual poseía, soportando de esta forma un bajo acoplamiento y una alta cohesión.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. Durante el diseño de nuestra aplicación podemos observar que para cada una de las fases o subprocesos existe una clase que recoge toda la información que se genera durante la ejecución de dicha fase. Por ejemplo en la fase de Solicitud podemos decir que la clase EDFaseSolicitud es un creador ya que la misma recoge toda la información del trámite en cuestión y de las exenciones aplicadas al mismo.

Alta cohesión: Con el objetivo de lograr un alto nivel de cohesión se decide agrupar las clases relacionadas en módulos de forma tal que cada módulo realice sus funciones de forma independiente a los demás, es decir, cada cual podrá realizar sus funciones específicas sin tener una relación de dependencia muy fuerte con los demás módulos. Por lo tanto, las clases que serán las encargadas de mantener la comunicación con la base de datos se organizaron en el módulo acceso a datos, las clases que encargadas de manejar la lógica de negocio se agruparon en el módulo de negocio, las clases encargadas de mantener la interacción con el usuario se ubicaron en el módulo presentación, y las clases encargadas de contener los datos persistentes de nuestra aplicación se ubicaron en el módulo dominio.

Bajo acoplamiento: Para la comunicación entre los módulos se diseñaron interfaces para cada una de las clases de nuestra aplicación de forma tal que lo que viaje entre los módulos sean las interfaces y no las clases, así logramos que los cambios que se

realicen en un módulo afecten de la menor forma posible a los demás que se comunican con él, además de que se reduce de forma sustancial la complejidad del sistema y mejora su mantenibilidad.

2.5 Conclusiones

El análisis de la arquitectura en capas a utilizar en el desarrollo del sistema, las funcionalidades de cada nivel en particular y las características fundamentales; permitió adquirir el conocimiento necesario para la adecuada utilización de las funcionalidades que esta ofrece.

Como parte de la solución se obtuvieron los modelos de diseño e implementación, los cuales incluyen los diagramas de clases, de secuencia, de componentes y de despliegue.

El uso de patrones como Fábrica Abstracta, Solitario, Fachada e Interceptor, permitieron que se creara una aplicación flexible y mantenible.

Se considera que los objetivos trazados para el presente capítulo han sido cumplidos y como resultado se obtuvo el sistema informático para el proceso de solicitud de copias de los Registros Principales; cumpliendo el mismo con lo estipulado en la Ley de Registro Público y del Notariado.

CAPÍTULO 3: VALIDACIÓN DE LOS RESULTADOS

La validación de la solución propuesta es una actividad sumamente importante, dado que permite determinar la calidad del diseño y de la implementación del sistema. En el presente capítulo se muestra el análisis de los resultados de la validación del sistema, ya sea mediante las métricas para medir la calidad del diseño de Sistemas Orientados a Objetos, como al aplicar las pruebas de unidad al código con NUnit y al realizar las pruebas de aceptación.

3.1 Métricas orientadas a objetos aplicadas al diseño del software

3.1.1 Tamaño de clase

"Las métricas orientadas al tamaño para las clases orientadas a objetos se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema orientado a objeto como un todo".(Pressman, 2005).

Esta métrica establece que el tamaño general de una clase puede medirse a través de la medida del número total de operaciones y atributos (heredados y privados de la interfaz), que se encapsulan dentro de la clase.

Según Pressman los valores grandes para esta métrica, indican que la clase debe tener bastante responsabilidad. Esto reducirá la reutilización de esta clase y complicará la implementación y las pruebas. (Pressman, 2005).

Además se pueden calcular los promedios del número de atributos y operaciones de las clases, de manera que mientras menor sea el valor promedio, mayor será la posibilidad de reutilizar la clase.

A continuación se muestra la relación de los intervalos de valores para determinar si el valor de TC[22] se considera pequeño, medio o grande.

[22]Tamaño de clase

Umbral	TC (Total de Atributos y Operaciones)
Menor o igual que 20 (TC <= 20)	Pequeño
Entre 20 y 30 (20 < TC <= 30)	Medio
Mayor que 30 (TC > 30)	Grande

Tabla 1: Umbrales para la Métrica TC. (Pacheco Iglesias, y otros, 2008)

Al aplicar esta métrica al proceso de solicitud de copias (Ver anexo 2) se obtuvo que de 108 clases con que cuenta este proceso, 6 son de tamaño grande, 10 de tamaño medio y 92 de tamaño pequeño; para un promedio de cantidad de atributos de 4.30 y un promedio de cantidad de operaciones de 7.04. A continuación se muestra el porciento de clases según los tamaños definidos.

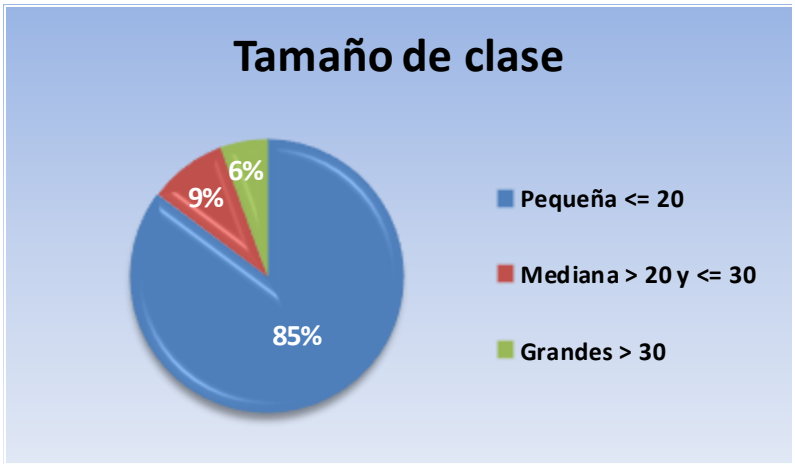

Fig. 19: Porciento de clases por tamaños

Una vez aplicada la métrica TC y analizados los resultados obtenidos se puede concluir que la mayor parte de las clases se clasifican de tamaño pequeño (85%) y mediano (9%), mientras que la minoría se clasifican en grandes (6%), por lo que se considera un resultado positivo para la métrica según los parámetros establecidos. Esto indica que las clases del sistema por lo general son reusables y de fácil mantenimiento.

3.1.2 Árbol de Profundidad de Herencia

La métrica Árbol de Profundidad de Herencia se utiliza para medir la complejidad de una clase, complejidad del diseño y la posibilidad de reutilización. Esta se define como la máxima longitud del nodo a la raíz del árbol, el valor obtenido se denomina APH[23]. (Pressman, 2005).

Se puede decir que mientras mayor sea el APH, superior será la probabilidad de que las clases de más bajos niveles hereden muchos métodos y mayor será la complejidad del diseño.

A continuación se muestra las jerarquías de clases presentes en la capa de dominio del flujo de solicitud de copias.

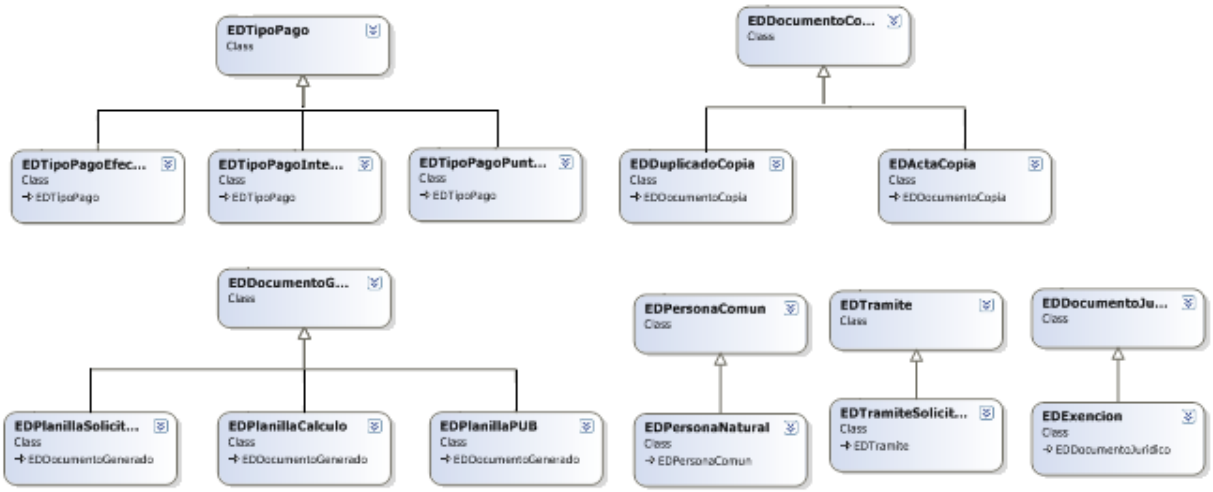

Fig. 20: Jerarquías de clases de la capa de dominio del proceso solicitud de copias.

Al analizar las jerarquías presentes en esta capa se obtuvo que el nivel más alto de APH entre las clases del diseño sea dos, lo que significa que el diseño tiene poca complejidad, por lo que es fácil de entender y comprobar.

[23] Árbol de Profundidad de Herencia

3.1.3 Número de descendientes

"*Las subclases inmediatamente subordinadas a una clase en la jerarquía de clases se denominan sus descendientes*". (Pressman, 2005). Esta métrica establece que si crece el NDD[24] aumenta la reutilización, pero la abstracción representada por las clases predecesoras puede disolverse.

Al aplicar la métrica NDD a la jerarquía de clases del dominio representada en la figura 20, se obtiene que el máximo NDD es tres; pero es necesario aclarar que en la capa de presentación los formularios y las acciones que los controlan heredan todos de *frmPlantilla* y *accPlantilla* respectivamente, puesto que todos los formularios tienen una estructura y comportamiento común. Esto hace que se obtenga un elevado NDD y que aumente la reutilización del código.

3.2 Pruebas del NUnit

Con el objetivo de evaluar la calidad de la implementación del sistema informático se realizaron varias pruebas de caja blanca. Estas pruebas fueron ejecutadas con la utilización del NUnit, de manera que se logró verificar que el código funcione adecuadamente.

Las pruebas se realización en las capas de acceso a datos y negocio; aprovechando las facilidades que brinda la arquitectura del sistema para practicar las mismas.

A continuación se muestra una porción de las pruebas efectuadas en la capa de acceso a datos, donde se validó que se obtengan e inserten correctamente los datos de la base de dato.

[24] Número de descendientes.

Resultados obtenidos de las pruebas aplicadas en la capa de acceso a datos

Este método constituye el punto de partida de las pruebas de unidad, en él se declaran las inicializaciones de la base de datos de Registros Principales, que son la base para la realización de las futuras pruebas.

```
[SetUp]
public void IniciarContexto()
{
    string cadenaConeccion =
       "Data Source=RP213;
        User Id=usuario2_213;
        Password=Data Source=RP213;
        User Id=usuario2_213;
        Password=CslNMWU73oQb0Ny1dFos3361tW0=;;";

    Base.IniciarContexto(cadenaConeccion, Base.TipoConexion.Oracle, true);
 }
```

Fig. 21: Método de inicio de las pruebas.

A continuación se ilustran varios métodos de pruebas realizadas a los procedimientos que se encuentran en las clases DAOs de trámite, de persona natural y de documento digital.

El siguiente método se encarga de probar que el trámite se cree correctamente, para ello verifica que el número del trámite se haya generado adecuadamente.

```
[Test]
public void ProbandoCrearTramite()
{
    obj = Base.Resolver<IDaoTramite>();
    var tramite = Base.Resolver<IDTramite>();
    obj.CrearTramite(tramite);

    Assert.IsNotNullOrEmpty(tramite.NumeroTramite);
}
```

Fig. 22: Prueba al método Crear Trámite.

Esta funcionalidad dado el identificador del trámite y del tipo de persona natural pasados por parámetro, se encarga de validar que funcione correctamente el método

"*Cargar Persona Trámite"*, comprobando que se devuelvan las personas naturales asociadas a ese trámite.

```
[Test]
public void ProbandoCargarPersonasTramite()
{
    obj = Base.Resolver<IDaoTramite>();
    var persona = obj.CargarPersonasTramite(2130000000000000034, 3);

    Assert.AreNotEqual(0,persona.Count);
}
```

Fig. 23: Prueba al método Cargar Personas Trámites.

Este método se encarga de validar que la búsqueda de la persona sea correcta, para ello se pasa por parámetro un pasaporte real *"111"* y se comprueba que la persona obtenida tenga como primer nombre "*FISCALIA*".

```
[Test]
public void ProbandoBuscarPersonaNatural()
{
   daoPersonaNatural = Base.Resolver<IDaoPersonaNatural>();
   var letra = char.MinValue;
   IDPersonaNatural pers =
      daoPersonaNatural.BuscarPersonaNatural(-1,letra,-1,"111");

   Assert.AreEqual("FISCALIA", pers.PrimerNombre);
}
```

Fig. 24: Prueba al método Buscar Persona Natural.

Este procedimiento verifica que se actualice correctamente el teléfono de una persona natural que se encuentra en la base de datos.

```
[Test]
public void ProbandoActualizarPersonaNatural()
{
    daoPersonaNatural = Base.Resolver<IDaoPersonaNatural>();
    var letra = char.MinValue;
    var persona = daoPersonaNatural.BuscarPersonaNatural(-1, letra,-1,"111");
    persona.Telefono = "0412-2092248";
    daoPersonaNatural.ActualizarPersonaNatural(persona);

    var persona2 = daoPersonaNatural.BuscarPersonaNatural(-1,letra,-1,"111");

    Assert.AreEqual("0412-2092248", persona2.Telefono);
}
```

Fig. 25: Prueba al método Actualizar Persona Natural.

El siguiente método se encarga de probar que dado el identificador del tipo de documento (en este caso 1) se realice correctamente la búsqueda de los documentos que se encuentran en el archivo digital, para ello se verifica que la búsqueda devuelva resultados.

```
[Test]
public void ProbandoBuscarDocumentoDigitalArchivo()
{
    daoDocumentoDigital = Base.Resolver<IDaoDocumentoDigital>();
    IList<IDProxyDocumento> documentos =
    daoDocumentoDigital.BuscarDocumentoDigitalArchivo("", "", "", "", "",
                                        -1, "", -1, -1, -1,-1, -1, 1, -1);
    Assert.AreNotEqual(0,documentos.Count);
}
```

Fig. 26: Prueba al método Buscar Documentos del Archivo Digital.

Este método verifica que se carguen correctamente la cantidad de copias a realizar, dado el identificador del trámite.

```
[Test]
public void ProbandoCargarCantidadCopias()
{
    daoCalculo = Base.Resolver<IDaoCalculo>();
    int cantidadCopias =daoCalculo.CargarCantidadCopias(2130000000000000006);

    Assert.AreEqual(1,cantidadCopias);
}
```

Fig. 27: Prueba al método Cargar Cantidad de Copias.

El siguiente método se encarga de validar que se cargue correctamente el documento de tipo duplicado, para ello se pasa por parámetro el identificador del documento y se comprueba que el documento obtenido sea el esperado.

```
[Test]
public void ProbandoCargarDupCopiaDoc()
{
    daoDocumentoDigital = Base.Resolver<IDaoDocumentoDigital>();
    var doc = daoDocumentoDigital.CargarDupCopiaDoc(2130000000000000308);

    Assert.AreEqual("Registro",doc.TipoOficina.Descripcion);
}
```

Fig. 28: Prueba al método Cargar Documento Duplicado Copia.

A continuación se muestra una imagen que representa la última iteración de las pruebas realizadas con el NUnit en la capa de acceso a datos, donde se puede evidenciar que fueron satisfactorias. Se efectuaron 24 pruebas, de manera que se abarcaron métodos relacionados con todos los DAOs y se comprobó que el código funciona correctamente. De forma general se realizaron un total de 3 iteraciones de forma tal que se fueron corrigiendo los defectos encontrados durante las primeras iteraciones. (Ver anexo 5)

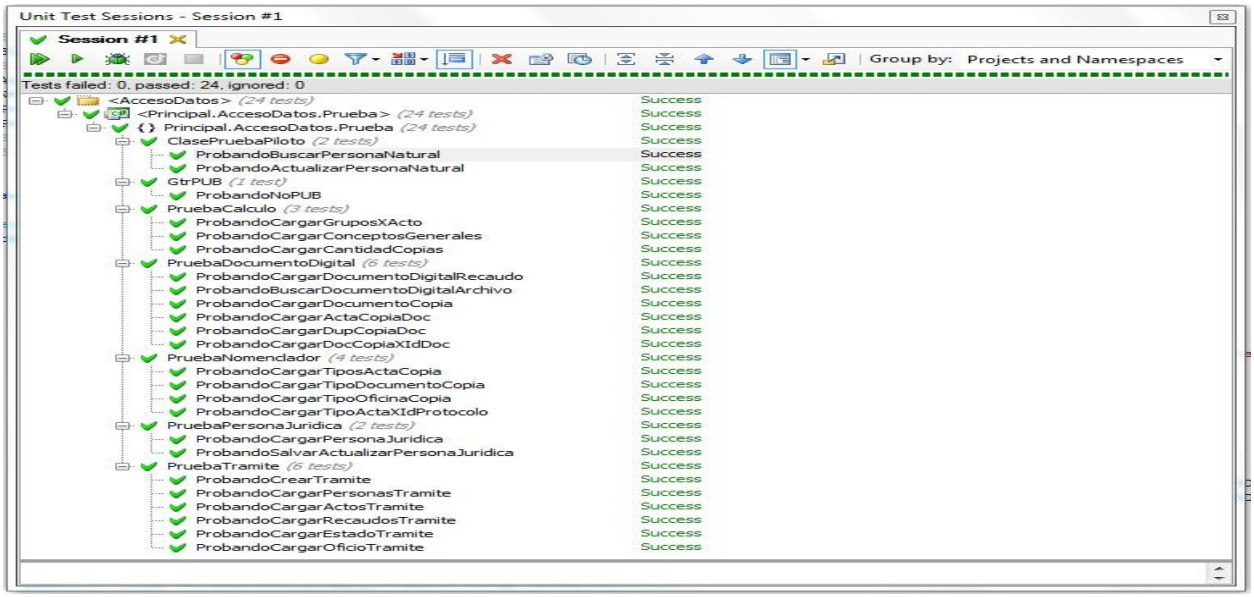

Fig. 29: Resultados de las pruebas con el NUnit.

3.3 Pruebas de aceptación

Las pruebas de aceptación las realiza el usuario final en lugar del responsable del desarrollo del sistema. Estas permiten que el cliente valide todos los requisitos del software. (Pressman, 2005)

3.3.1 Prueba alfa

"La prueba alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso". (Pressman, 2005)

Las pruebas alfa aplicadas a la solución propuesta se realizaron a través del laboratorio de aceptación, el que contó con un ambiente de trabajo colaborativo y se evaluaron las respuestas del sistema en un escenario simulado desde la perspectiva del cliente. En este laboratorio participaron varios funcionarios del SAREN, en conjunto con miembros del equipo de desarrollo y una especialista de pruebas de CALISOFT (Centro Nacional de Calidad del Software).

Durante el laboratorio de aceptación se detectaron dos no conformidades relacionadas con el proceso de solicitud de copias, ambas con un nivel de importancia alta y que fueron resueltas por el equipo de proyecto y aprobadas por el cliente inmediatamente.

Además el cliente solicitó doce pedidos de cambio de los cuales tres constituían necesidades para lograr un correcto funcionamiento de la aplicación y nueve representaban mejoras que optimizarían las prestaciones del entregable según los intereses del cliente. En la gráfica que se muestra a continuación se evidencia que uno de los pedidos de cambios no procedió, nueve fueron resueltos por él equipo de desarrollo y aprobados por parte del cliente durante el laboratorio, mientras que dos por mutuo acuerdo entre todos los involucrados quedaron pendientes a solucionar. Evidenciándose de esta forma que el 75% de los pedidos de cambios fueron realizados, mientras que sólo un 17% quedaron pendientes a solucionar.

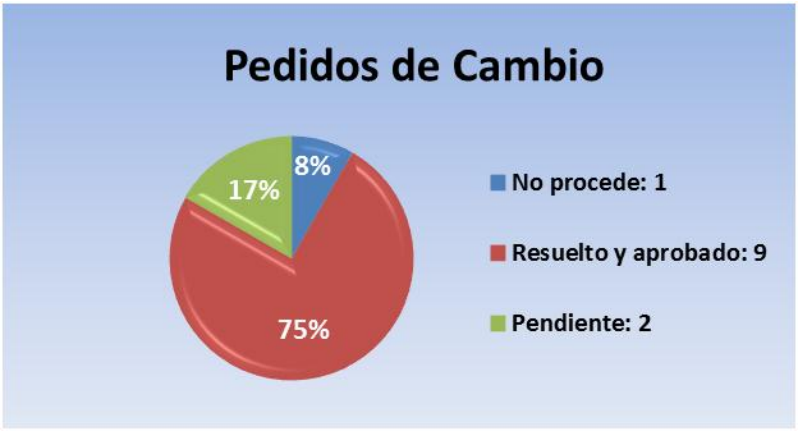

Fig. 30: Resultados de las pruebas.

Al concluir el laboratorio teniendo en cuenta que las no conformidades fueron debidamente resueltas por el equipo de desarrollo, validada la eficacia de la corrección por los clientes y gestionadas los pedidos de cambios adecuadamente, se aceptó la solución de Registros Principales por las partes involucradas. Posteriormente se resolvieron los pedidos de cambios pendientes, encontrándose de esta forma el sistema listo para entrar en funcionamiento en un ambiente similar al ambiente para el que fuera diseñado.

3.3.2 Prueba beta

"La prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados durante estas, el desarrollador del software lleva a cabo modificaciones y así prepara una versión final del producto de software". (Pressman, 2005).

Con el fin de comprobar el desempeño de la aplicación en el entorno real, se seleccionó un Registro Principal en el que se realizan un gran número de trámites diariamente. En este registro se realizaron la prueba beta con la presencia de una analista del equipo de desarrollo; para ayudar al adiestramiento de los funcionarios en la aplicación y registrar con más detalles las incidencias detectadas. Además dichas pruebas fueron supervisadas por una especialista de pruebas de CALISOFT.

Durante la prueba beta los funcionarios involucrados con la ayuda de la analista capturaron las incidencias detectadas, las que fueron resueltas por el equipo de desarrollo y posteriormente actualizadas en la oficina para su aprobación por parte del cliente. En total se realizaron 107 trámites de solicitud de copias y fueron detectadas seis incidencias, de las cuales dos no procedían y cuatro fueron resultas inmediatamente por el equipo de desarrollo; de manera que se le dio solución al 100% de estas incidencias de forma oportuna.

Estas pruebas permitieron comprobar el nivel de calidad y terminación del sistema, además de que contribuyeron a que se realizaran las modificaciones necesarias para preparar la versión final del producto de software. (Ver anexo 3)

3.4 Análisis del cumplimiento de los objetivos

Para verificar el cumplimiento de los objetivos de la investigación en el presente epígrafe se analizarán los aspectos que contribuyen a alcanzar los resultados esperados.

Seguridad jurídica

Con el objetivo de aumentar los niveles de seguridad jurídica, de manera que no se puedan modificar los derechos del ciudadano venezolano, al no ser a través de procedimientos establecidos en la ley; en la aplicación se establecerán varios mecanismos de seguridad, que contribuyan a que los procesos sean ejercidos según lo establecido legalmente. El sistema se regirá por la Ley de Registro Público y del Notariado, de manera que cumplirá cabalmente cada uno de los artículos establecidos en la misma.

En el sistema se firmarán digitalmente los documentos, garantizando de esta forma la autenticidad e integridad de los mismos y evitando que sean falsificados por terceras personas. Los documentos serán firmados digitalmente por el Registrador, que será la persona que certificará que dichos documentos tienen valides legal, dando así mayor confianza y legalidad al proceso. Con la firma digital se fortalece la Fe Pública Registral, establecida en la Ley de Registro Público y del Notariado.

También se establecerá una adecuada gestión de roles, donde cada funcionario de las oficinas de Registros Principales, contará con un usuario que tendrá sólo los permisos necesarios según el proceso que realice; con esto se logrará que estos procesos sean ejecutados únicamente por el personal autorizado. La información que se gestiona en cada proceso, sólo podrá ser modificada por dichos funcionarios en

dependencia del nivel de permisos que le hayan sido otorgados, manteniendo con ello la confidencialidad e integridad de dicha información; con el objetivo de aumentar el nivel de confianza del ciudadano venezolano y evitar que ocurran eventos adversos, que puedan afectar la imagen y el compromiso del personal que trabaja en los registros. Además las contraseñas de dichos usuarios serán cifradas para impedir que puedan ser descubiertas y utilizadas por terceros.

Estandarización

Con la utilización de la aplicación se logrará que el proceso de solicitud de copias se realice de manera uniforme en todas las oficinas de Registros Principales; muestra de esto es que anteriormente en el proceso de cálculo por un mismo acto se cobraban montos distintos en cada una de las oficinas, mientras que el sistema calculará el monto a pagar automáticamente, según lo establecido en la Ley de Registro Público y del Notariado, por lo que el monto a pagar será el mismo en todas las oficinas. Al usuario se le entregará la PUB junto con la Planilla de Cálculo, a través de la cual verificará cada uno de los conceptos calculados y el monto total a pagar.

De forma general la aplicación obligará a que se realicen los procesos en las oficinas de manera uniforme según lo establecido en la ley y que puedan ser supervisados los mismos constantemente por funcionarios del SAREN; contribuyendo a que disminuya el fraude y evitando que se cometan errores, dando esto una gran confianza al ciudadano venezolano cuando vaya a realizar algún trámite de solicitud en cualquier Registro Principal.

Con la estandarización del proceso de solicitud de copias en todas las oficinas de Registros Principales, se favorecerá el cumplimiento del Principio de Legalidad planteado en la Ley de Registro Público y del Notariado.

Rapidez

En los Registros Principales en muchas ocasiones se producen atrasos en la entrega de los documentos, de manera que uno de los propósitos que se persigue con el

sistema es aumentar los niveles de rapidez de los servicios que se brindan al ciudadano venezolano. Con el objetivo de verificar que con la utilización del sistema disminuye el tiempo necesario para realizar un trámite de solicitud, se observó el tiempo que se emplea para hacer cada uno de los subprocesos de solicitud con el sistema y sin este. (Ver anexo 4). Para ello se realizó la observación de los tiempos utilizados para 20 trámites de solicitud de copias, obteniéndose los valores promedios que se muestran en la tabla siguiente:

Proceso	**Duración sin el sistema**	**Duración con el sistema**	**Diferencia de tiempo**
Solicitud de copias	4 min	3 min	1 min
Cálculo	1,30 min	30 s	1 min
Presentación	6 min	2 min	4 min
Procesamiento	3 min	2 min	1 min
Revisión de Procesamiento	1 min	1 min	0 min
Firma digital		20 s	-20 s
Entrega	1,30 min	30 s	1 min
Total	**17 min**	**9,20 min**	**7,40 min**

Tabla 2: Resultados obtenidos con las observaciones realizadas.

Al analizar la tabla anterior se pudo constatar que con la utilización del sistema disminuye considerablemente el tiempo necesario para realizar un trámite de solicitud, pudiéndose apreciar que se demora aproximadamente 7 min menos realizar un trámite. De forma general se puede decir que en el tiempo que se realizaban anteriormente 20 trámites, con la utilización del sistema se podrán hacer 36 trámites, por lo que se considera que aumentará la rapidez con que se elabora un trámite de solicitud de copias.

El acta de aceptación obtenida al culminar el laboratorio de aceptación es evidencia de que el sistema cumple con los objetivos para los cuales fue creado, de manera que se puede decir que se logró elevar los niveles de seguridad jurídica, estandarizar

el proceso de solicitud de copias y aumentar los niveles de rapidez en dicho proceso; dando cumplimiento a los problemas definidos en el proyecto técnico del proyecto y por tanto a los objetivos de esta investigación.

3.5 Conclusiones

Al aplicar las métricas para medir la calidad del diseño de sistemas orientados a objetos, se pudo constatar que el diseño propuesto tiene poca complejidad. Además las clases que lo componen por lo general son reusables y fáciles de comprobar, de manera que se puede concluir que el sistema para la gestión de las funcionalidades del proceso de solicitud de copias posee un diseño con la calidad requerida.

La validación del código se realizó mediante el framework NUnit, posibilitando que los procedimientos de pruebas se realizaran fácilmente y en menor tiempo. Las pruebas efectuadas arrojaron resultados satisfactorios, comprobándose así el correcto funcionamiento del código. Además al realizar las pruebas de aceptación se dio respuesta oportuna a las incidencias detectadas, quedando de esta forma todas las funcionalidades del proceso de solicitud de copias listo para su despliegue y utilización en las oficinas de los Registros Principales.

CONCLUSIONES

Con la realización del presente trabajo se arribó a las siguientes conclusiones:

- Se logró dominar los aspectos más relevantes relacionados con el proceso de solicitud de copias, de manera que se adquirió el conocimiento necesario para que el sistema de gestión para el proceso de solicitud de copias cumpla con las leyes que rigen el funcionamiento de los Registros Principales.
- Se construyó el modelo de diseño y el modelo de implementación, desarrollándose los diagramas de interacción, diagramas de clases, el modelo de datos, los diagramas de componentes y de despliegue.
- Se implementó el sistema informático para la gestión del proceso de solicitud de copias de acuerdo con lo estipulado en la Ley de Registro Público del Notariado, dando esto una gran confianza al ciudadano venezolano a la hora de realizar un trámite de solicitud de copias.
- Se realizó la validación tanto del diseño como de la implementación del sistema que gestiona el proceso de solicitud de copias a través de la aplicación de diferentes métricas y la realización de varios tipos de pruebas que ayudaron a corroborar el nivel de calidad del software.
- El sistema estandarizará el proceso de solicitud de copias en todas las oficinas de Registros Principales, dando mejor cumplimiento al Principio de Legalidad planteado en la Ley de Registro Público del Notariado vigente.
- El sistema de gestión para el proceso de solicitud de copias incrementará los niveles de seguridad jurídica con el uso de la firma digital a los documentos de los registros; garantizando de esta forma la autenticidad e integridad de dichos documentos, de manera que se fortalece la Fe Pública Registral de los archivos digitales de las oficinas. Además se gestiona el acceso al mismo, a través de una adecuada asignación de roles, favoreciendo la confidencialidad e integridad de la información que se maneja.

BIBLIOGRAFÍA

.Net Framework Developer Center. Información general y conceptual sobre .NET Framework. *Información general y conceptual sobre .NET Framework.* [En línea] [Citado el: 10 de febrero de 2011.] http://msdn.microsoft.com/library/zw4w595w.aspx.

Aguilar, Luis Joyanes. 1996. *Programación Orientada a Objetos.* Madrid : s.n., 1996.

Budd, Timothy. 1991. *An introduction to Object-Oriented Programming.* 1991. 0-201-54709-0.

César de la Torre Llorente, Unai Zorrilla Castro,Javier Calvarro Nelson,Miguel Ángel Ramos Barroso. 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .Net.* 2010.

CodeBox. CodeBox Glosario. [Online] [Cited: 02 15, 2011.] http://www.codebox.es/glosario.

Data Modeling Enterprise Software| Embarcadero ER/Studio. Data Modeling Enterprise Software| Embarcadero ER/Studio. [En línea] www.embarcadero.com/products/er-studio-xe.

Fernández, Ángel Sanz. 1947. *Instituciones de Derecho Hipotecario.* Madrid : s.n., 1947.

Gamma, Erich, y otros. 2002. *Design Pattern: elements of reusable object-oriented software.* s.l. : Addison Wesley, 2002.

Gobierno Bolivariano de Venezuela. 2006. *Ley del Registro Público y del Notariado.* Caracas : s.n., 2006.

Gómez Vicente, Miguel Ángel and Martín Mohedano, Manuel. 2005. *Introducción a C#.* s.l. : Departamento de Informática y Automática Universidad de Salamanca, 2005.

Hunt, Andy and Thomas, Dave. 2007. *Pragmatic Unit Testing.* Texas : s.n., 2007.

IEEE. 2000. *Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000. Std 1471-2000.

—. 1990. *Standard Glossary of Software Engineering Terminology.* New York : s.n., 1990. Std 610.12-1990.