UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 15

Diseño e implementación de una solución informática para el proceso de inscripción de documentos en los Registros Principales de la República Bolivariana de Venezuela.

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

Autores:

Bárbara Inés González Jorge

Arturo Díaz Rivas.

Tutores:

Ing. Armando Esteban Pacheco Iglesias.

Ing. Yunier Raúl Vega Rodríguez

Ciudad de La Habana, Cuba

Junio 2010

Declaración de Autoría

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

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

Bárbara Inés González Jorge Autor	Arturo Díaz Rivas Autor
Yunier Raúl Vega Rodríguez Tutor	Armando Esteban Pacheco Iglesias Tutor

<<…Universidad de Excelencia

es la idea de hacer lo mejor que pueda hacerse,

de desarrollar la mejor universidad que se haya

desarrollado jamás, buscando lo óptimo,

lo más perfecto posible dentro de las cosas humanas,

lo más nuevo, lo más creativo, algo que no solo

sirva a los intereses de nuestro país,

sino sirva de ejemplo para el resto del mundo…>>

Agradecimientos

A mi familia: mi mamá, por el eterno apoyo y dedicación en todos estos años de estudio. A mi padre por guiarme y por sus grandes consejos de la vida. A mi linda hermana, la cual quiero con todas las fuerzas de mi corazón, aunque sea mayor que yo siempre será mi hermanita, que sirva esta tesis como ejemplo de sacrificio profesional, estoy seguro que ella lo puede hacer mejor que yo.

Al equipo de desarrollo: Juan Pablo, Rodolfo, Rene, Iosmel, Diana, Susana, Yaumi, Yunier Pérez del Proyecto Registros y Notarias; su cooperación y esfuerzo fue decisivo para la realización de este Trabajo de Diploma.

A mis jefes, amigos y tutores Armando y Yunier Raúl por guiarme y asesorarme en todo el desarrollo del trabajo.

A mi compañera de tesis, Barbie por soportarme tanto en la confección de este trabajo, por su amistad sin reservas durante todo este tiempo y por la confianza depositada para que saliera esta gran tesis.

A la Universidad de las Ciencias Informáticas, que sin esta maravillosa universidad no se pudieran profundizar los amplios conocimientos de universitario.

Arturo

A mi familia: por el apoyo y dedicación en todos estos años de estudio, en especial a mi tía María, a mis hermanos Frank y Michel, a Yaine, mi cuñada, a todos: los quiero mucho y siempre estaré ahí cuando me necesiten. Muy en especial a mi madre por educarme como soy y guiarme por este camino que siempre soñó para mí, sé que si pudiera verme en este momento estaría muy orgullosa.

A mis amigas del alma: Danise, Yudelkys, Lisset, Leyanis, Tais, por brindarme su amistad en los buenos y en los malos momentos.

A mis tutores y amigos Armando y Yunier Raúl: por su guía y apoyo para la realización de este trabajo.

A mi compañero de tesis Arturo: por soportarme en la realización de este trabajo y estar ahí para lo que se necesitara.

A mi MuTiKo lindo por llegar a mi vida, aunque un poco retrasado, pero como suele decirse, nunca es tarde si la dicha es buena.

A mis maestros y profesores que me han enseñado a aprender, en especial a Vicente, Maira, Zarate, Yasel, Libertad, Yvonne…

A esta maravillosa universidad que me ha dado la posibilidad de formarme como profesional y ser mejor como ser humano.

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

Bárbara Inés

Dedicatoria

A mis padres que siempre están cuando los necesito.

A mi hermana, sé que serás una gran profesional.

A toda mi familia que siempre me han apoyado de una forma u otra.

A todos mis mejores amigos que son como mis hermanos.

A Fidel y la Revolución.

Arturo

Dedicado a la memoria de mi adorada madre.

Bárbara Inés

Resumen

En el presente documento se realiza un análisis acerca de los principales problemas que presenta el Sistema Registral vigente en cuanto a la Inscripción de Documentos 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.

Se realiza un estudio de las tendencias en la industria del software, las principales plataformas y metodologías de desarrollo de software, así como una descripción de las herramientas que van a ser utilizadas.

Se presenta una propuesta de solución de software para el Proceso de Inscripción de Documentos para su despliegue en los Registros Principales del país, posibilitando la estandarización de dichos procesos en los mismos. Incluye además las etapas de Diseño e Implementación, describiendo los elementos más importantes de la misma.

Por último, recoge las acciones realizadas con el fin de validar el sistema propuesto, junto con los resultados obtenidos durante la validación.

Tabla de contenido

Introducción

Desde hace ya algún tiempo entre Cuba y Venezuela han proliferado relaciones de cooperación mutua entre ambas naciones, en el marco de la Alternativa Bolivariana para las Américas (ALBA). El presidente venezolano ha llevado a cabo muchas reformas en los diferentes sectores de la sociedad con el fin de mejorar la calidad de vida del ciudadano. Uno de los sectores que necesitaba remodelación de sus procesos era el sector público en Venezuela.

Ya desde 1993 se había promulgado la Ley de Registro Público y del Notariado, donde se autorizó la digitalización y automatización del Sistema Registral y Notarial venezolano, pero no se habían dado pasos significativos en este sentido. Posteriormente, en 1999, con La Constitución Bolivariana, se abrió un camino para modernizar las instituciones del Sector Público. El 13 de noviembre de 2001, mediante Decreto No 1554, se promulga la Ley del Registro Público y del Notariado, la cual en su artículo 14 establece la creación de la Dirección Nacional de Registros y del Notariado, perteneciente al Ministerio del Poder Popular para las Relaciones Interiores y Justicia (MPPRIJ) siendo el órgano encargado de ordenar, supervisar y controlar todos los procesos que se llevan a cabo en los registros, tanto Mercantiles, Públicos, Principales, y las Notarías. Se decide entonces realizar un proyecto que tendría como objetivo desarrollar un sistema que informatizara todos los procesos registrales, así surgió el Servicio Autónomo de Registros y Notarías (SAREN). SAREN, en su primera fase, que se encuentra actualmente en funcionamiento, tiene como alcance el universo de Registros Inmobiliarios y Mercantiles del Servicio de Registros y Notarías, mientras que los Registros Principales y las Notarías no se encuentran automatizados por esta solución informática.

Los Registros Principales son oficinas que están adscritas a SAREN y se encuentran distribuidos por todo el país, donde existen un total de 21 Registros Principales. Según la Ley de Registro Público y del Notariado[1], en su artículo 65, corresponde al Registro Principal efectuar la inscripción de los actos siguientes (1):

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.

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

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, expedir copias certificadas y simples de los asientos y duplicados de los documentos que reposan en sus archivos (1).

La inscripción de documentos es el proceso fundamental que se lleva a cabo dentro de una oficina del Registro Principal, y actualmente se ve afectado por la siguiente situación problemática.

- La mayoría de las oficinas no cuentan con una plataforma informática adecuada, por lo que el trabajo generalmente es manual, provocando atrasos en la entrega de los documentos.

- La mayor parte de la información documental se encuentra en soporte físico, que tiende a deteriorarse con el paso del tiempo, ocasionando que disminuya la calidad de la información que se preserva y que su recuperación sea en ocasiones un proceso difícil.

- No existe un proceso centralizado de inscripción de documentos y de organización de la información, originando cierta autonomía en las oficinas y haciéndole difícil a SAREN el control y administración de los registros. Esta descentralización afecta la publicidad registral en gran medida ya que los documentos solo pueden ser consultados en el lugar donde se originaron.

- Los registros principales guardan sus duplicados en sus mismas oficinas, además de los duplicados de otras, por lo que si ocurre una eventualidad como inundaciones o incendios, pueda perderse toda o parte de la información que estos archivan.

- La mínima parte de los patrimonios de las oficinas que se encuentran en soporte digital, no explotan las potencialidades que brinda el uso de la firma digital, la cual ofrece mayor seguridad a los documentos digitales generados.

Debido a todos los problemas anteriormente planteados, se evidencia la necesidad de cambios generales en las oficinas de los Registros Principales, tanto en los procesos registrales como en la infraestructura tecnológica en la que se llevarán a cabo.

El problema que enfrenta este trabajo consiste en cómo centralizar y gestionar el proceso de inscripción de documentos, de manera que facilite un mejor servicio y funcionamiento de los Registros Principales de Venezuela, según lo estipulado en la Ley de Registro Público y del Notariado.

Es por ello que se hace necesaria la introducción de un sistema informático integrado que gestione la inscripción de documentos en los Registros Principales de la República Bolivariana de Venezuela.

Se define como objeto de estudio de esta investigación el proceso de desarrollo de software.

Después de ser visto el problema planteado y el objeto de estudio se determina como campo de acción de esta investigación, el diseño e implementación como fases específicas dentro del proceso de desarrollo de software.

Constituye entonces el objetivo general del trabajo desarrollar el diseño y la implementación de una solución de software en el proyecto Registros y Notarías (RN), para informatizar el Proceso de Inscripción de Documentos en los Registros Principales de la República Bolivariana de Venezuela.

Para dar cumplimiento al objetivo general y obtener así los resultados esperados se han trazado varios objetivos específicos, los cuales se listan a continuación:

1. Estudiar las leyes que rigen el funcionamiento de los Registros Principales de Venezuela.
2. Analizar el Proceso de Inscripción de Documentos en los Registros Principales de Venezuela.
3. Analizar las metodologías disponibles para el diseño e implementación del software.
4. Estudio de los artefactos generados del negocio y requerimientos para facilitar la comprensión del proceso.
5. Estudiar el documento Línea Base de la Arquitectura generado por los arquitectos del proyecto.
6. Realizar los diagramas de clases, de interacción, de componentes y de despliegues para la solución informática.

7. Implementar la aplicación informática a partir de los diagramas obtenidos en el diseño.

8. Validar los modelos de diseño por las distintas métricas de calidad que existen.

9. Validar la solución de software mediante pruebas de unidad, ya sean pruebas de caja blanca y de caja negra, para lograr así una mejor calidad del producto.

Con este producto quedarán informatizadas las funcionalidades correspondientes al Proceso de Inscripción de Documentos en los Registros Principales.

Como hipótesis de trabajo se encuentra: si se desarrolla un sistema que informatice el proceso de inscripción de documentos en los registros principales de la República Bolivariana de Venezuela, entonces se logrará una centralización y una mejor gestión de dicho proceso.

Entre los principales resultados esperados con esta investigación se encuentran:

- Solución integral para gestionar el Proceso de Inscripción de Documentos en los Registros Principales de la República Bolivariana de Venezuela en su primera iteración.

- Estandarizar el Proceso Registral en todas las Oficinas de Registros Principales dando mejor cumplimiento al Principio de Legalidad planteado en la Ley de Registro Público del Notariado vigente, facilitando la organización y control de las operaciones que se realizan en los mismos.

- Lograr que la información obtenida de las oficinas registrales sea recibida en SAREN de manera más confiable y rápida, de modo que se pueda ofrecer un mejor servicio, organización y administración de los Registros.

- Lograr un incremento en la Seguridad Jurídica con el uso de la Firma Digital a los documentos generados en los registros, fortaleciéndose la Fe Pública Registral con la creación de los archivos digitales en las oficinas.

El presente documento se estructura por tres capítulos fundamentales:

- El Capítulo 1 aborda la fundamentación teórica para la realización de la solución propuesta y aspectos generales sobre los Sistemas Registrales, principios que lo rigen, características y particularidades vigentes en la República Bolivariana de Venezuela. También incluye su funcionamiento,

objetivos y servicios que brindan, así como una pequeña descripción del proceso de inscripción de los documentos. Se concluye el capítulo con un breve recorrido por el estado del arte de las distintas técnicas de programación, arquitectura de sistemas, metodologías, plataformas y herramientas existentes.

- El Capítulo 2 aborda la propuesta de solución del sistema, donde se explica la arquitectura sobre la cual se desarrollará el diseño e implementación, resaltando sus principales características. Además se describen las principales características de los frameworks de desarrollo utilizados y los artefactos generados como parte de la solución. Finalmente se hace un resumen de los principales elementos del diseño que incluye los patrones que fueron empleados en la construcción del sistema y los componentes personalizados utilizados.

- El Capítulo 3 aborda el análisis de los resultados de la validación del sistema. Se expone el resultado obtenido 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. Es un objetivo también de este capítulo mostrar los resultados obtenidos de las pruebas unitarias.

Capítulo 1. Fundamentación Teórica

1.1. Sistemas Registrales

Según Sanz Fernández[2], un sistema registral es "… el conjunto de normas que en un determinado país regulan las formas de publicidad de los derechos reales sobre los bienes inmuebles a través del Registro de la Propiedad, así como el régimen y organización de esta institución…" (2)

Dentro de los rasgos básicos de un Sistema Registral se puede observar que varían en su organización de acuerdo a los sistemas elegidos por cada país. Cada sistema registral debe estar dotado de un grupo de Principios Registrales, que deben ser de estricto cumplimiento, y a raíz de esto, se conforman sus particularidades y diferencias con respecto a otros países.

1.1.1. Principios registrales

La solución informática que se desea desarrollar automatizará el proceso de inscripción que tiene lugar en las oficinas de los Registros Principales de Venezuela. La misma además de ser un sistema de gestión y control, está enmarcado en un contexto jurídico, por lo que debe garantizar el debido cumplimiento de la ley vigente. Para ello tendrá que dar cumplimiento a todos los principios por los que se rige la Ley del Registro Público y del Notariado: los Principios Registrales.

Dentro de la Ley del Registro Público y del Notariado, redactada en la Asamblea Nacional de la República Bolivariana de Venezuela el 15 de Enero de 2007, se identifican los principios que rigen el trabajo en las oficinas de los registros principales venezolanos. A continuación se abordarán dichos principios con el objetivo de comprender sus implicaciones sobre el sistema a desarrollar.

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". (1)

Este principio establece que el procedimiento registral será iniciado por la presentación de un documento formal en un registro y la solicitud del presentante para que practique la inscripción y debe llegar a su conclusión, a menos que no sea admitido.

[2] Ángel Sanz Fernández: estudioso español de las materias registrales.

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”. (1)

Este principio es fundamental y expresa el ordenamiento el cual debe tener en cuenta que el primero en tiempo es el primero en derecho, salvo en excepciones establecidas por la ley.

Principio de Especialidad

“Los bienes y derechos inscritos en el Registro, deberán estar definidos y precisados respecto a su titularidad, naturaleza, contenido y limitaciones”. (1)

Este principio regula que cada registro debe tener bien definido el tipo de bien o derecho que inscribe de acuerdo con su titularidad, naturaleza, contenido y limitaciones.

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”. (1)

Es necesario tener en cuenta este principio en la implementación de la solución informática, pues este expresa la correlación que debe existir entre las inscripciones y los demás procesos posteriores que puedan surgir como las modificaciones, cancelaciones, así como otros documentos registrados asociados al mismo.

Principio de Legalidad

“Sólo se inscribirán en el Registro los títulos que reúnan los requisitos de fondo y forma establecidos por la ley”. (1)

Este principio expresa que para que se inicie el proceso de inscripción, el documento ha de ser previamente revisado, y estar conforme con los requisitos de fondo y forma establecidos por la ley.

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”. (1)

Fe pública: Es la autoridad legítima atribuida a notarios, escribanos, agentes de cambio y bolsa, cónsules y secretarios de juzgados, tribunales y otros institutos oficiales, para que los documentos que autorizan en debida forma sean considerados como auténticos y lo contenido en ellos sea tenido por verdadero mientras no se haga prueba en contrario. (3)

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

Al haberse explicado los principios que rigen el funcionamiento de los registros, se pueden analizar los Registros Principales en particular. Todos estos principios en mayor o menor medida estarán reflejados en los procesos u operaciones que se desarrollan en los Registros Principales.

1.2. Registros Principales

Los Registros Principales son oficinas que están adscritas a SAREN y se encuentran distribuidos por todo el país, donde existen un total de 21 Registros Principales. En él se realizan distintos tipos de trámites como son inscripción de documentos, solicitudes de copia, legalizaciones de firmas y notas marginales.

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

- 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.
- Las interdicciones e inhabilitaciones civiles.
- 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. En el caso de los registros civiles municipales están

obligados a remitir al Registro Principal, cada quince días, la información actualizada de los asientos relativos a:

- Nacimientos.
- Matrimonios.
- Defunciones.
- Las sentencias de divorcio.
- La nulidad del matrimonio.
- Los reconocimientos de filiación.
- Las emancipaciones.
- Las adopciones.
- Los actos relativos a la adquisición, modificación o revocatoria de la nacionalidad.
- La sentencia que declare la ausencia o presunción de muerte.
- Las constancias de no presentaciones.

Son responsables en su jurisdicción de informar al Registro Principal los nacimientos, matrimonios, defunciones y todo hecho que afecte el estado civil de las personas las alcaldías y los fiscales de niños, niñas y adolescentes, el Tribunal de Protección del Niño, Niña o Adolescente y los Consejos de Protección del Niño, Niña y Adolescente". (1)

Para dar cumplimiento de los objetivos trazados con este trabajo de investigación se analizará solamente el proceso de inscripción de documentos.

1.2.1. Proceso de Inscripción de Documentos en los Registros Principales

El proceso de inscripción de documentos se encuentra dividido en fases o subprocesos bien definidos por los que pasa el documento hasta que queda archivado en el registro. A continuación se presenta una imagen que muestra el proceso general de inscripción.

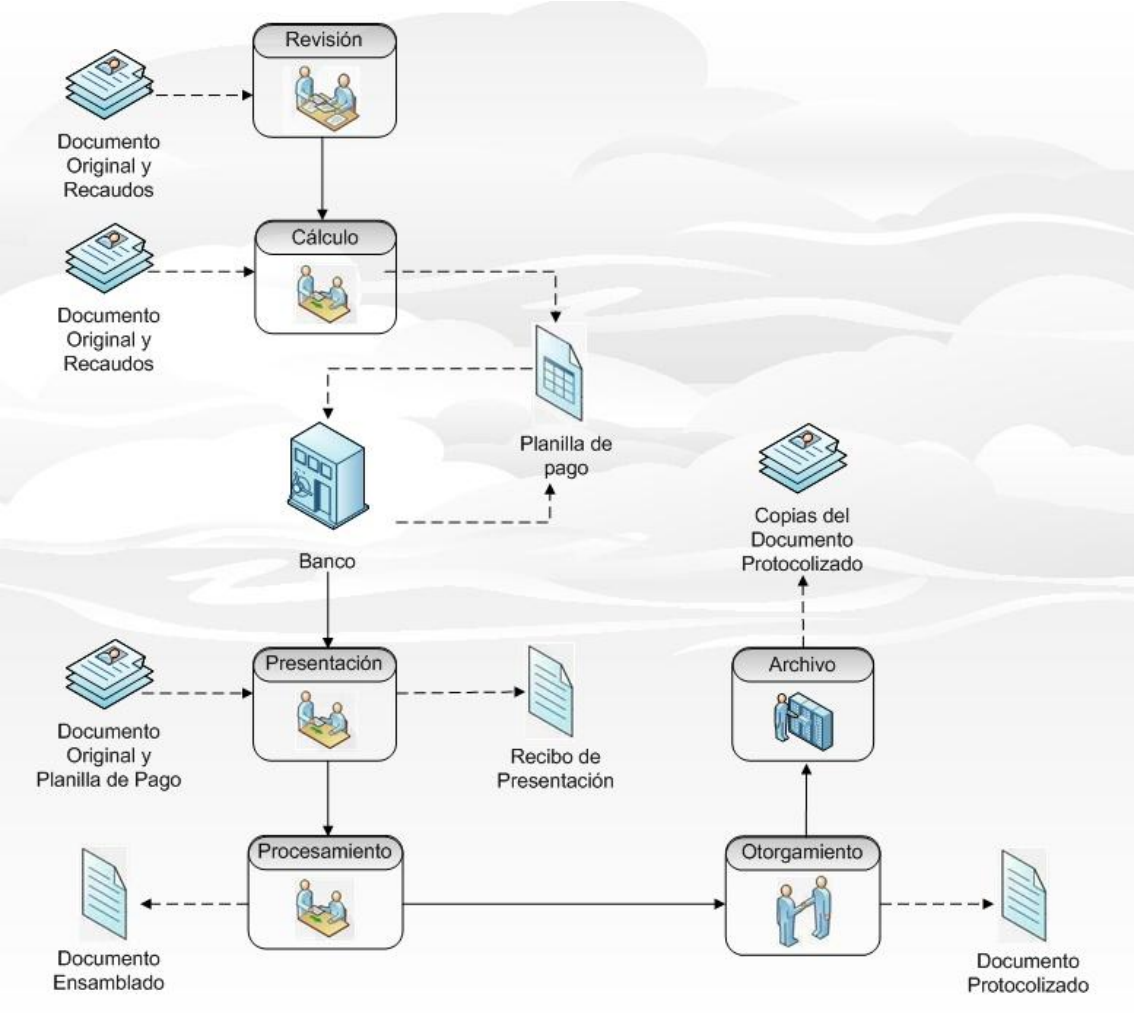

Figura 1: Diagrama de flujo de procesos de un documento para su inscripción en un Registro Principal.

Revisión: es la fase inicial por la que transita el documento. Primeramente el presentante debe entregar el documento original, entonces se lleva a cabo la revisión exhaustiva del mismo por parte del Funcionario de Revisión. El presentante procede a entregar los recaudos que le son solicitados. Si alguno de los documentos está incorrecto el funcionario se los devuelve, explicándole los errores y el presentante se retira del registro. En caso contrario, se le da paso a la siguiente fase, por lo que el presentante se dirige a taquilla para realizar el cálculo del documento.

Cálculo: es la fase en donde el registro precisa las unidades tributarias que debe pagar el usuario por la realización del trámite solicitado. Para ello, una vez en la taquilla, el usuario además de presentar los documentos, indica el tiempo de habilitación que desea. El Funcionario de Cálculo identifica el acto y los conceptos de pagos contenidos en el documento. Después elabora la Planilla Única Bancaria (PUB), si el acto está exento de pago el monto a pagar aparece en cero, en caso opuesto, con

la suma calculada. Entrega entonces la PUB y los documentos al presentante y este se retira.

Presentación: se corresponde a la fase en donde se hace formal la presentación del documento a ser inscrito, después que ha pasado por una revisión y el presentante ha cancelado la PUB. Para ello el presentante entrega el documento, los recaudos y la PUB cancelada al Funcionario de Presentación. El funcionario asigna al documento la ubicación en el archivo, establece la fecha y la hora del posible otorgamiento, elabora y asienta el resumen de las operaciones en el libro de presentación, y elabora un recibo que el presentante debe firmar. El funcionario también lo firma y le entrega una copia del recibo al presentante, el cual se retira del registro.

Procesamiento: se elaboran y se revisan las notas de registro del trámite y se ensamblan conjuntamente con el documento. El Funcionario de Nota elabora la nota de registro del trámite y la ensambla junto con el documento. Posteriormente el Funcionario de Revisión revisa la nota y si esta correcta coloca la firma y el sello de revisado. En caso contrario, envía la nota para su revisión al Funcionario de Nota.

Otorgamiento: se llevan a cabo las firmas de todas las partes involucradas y se le entrega el documento al usuario. Se le indica al usuario que debe firmar y plasmar su huella dactilar en la nota de registro. Además, los testigos firman y plasman sus huellas dactilares en la nota de registro. El registrador revisa el documento original y la nota de registro. Si los documentos están incorrectos se envían para el paso procesar documento, donde el documento es nuevamente procesado. Si los documentos están correctos el registrador firma y sella el documento original y la nota de registro. El registrador entrega el documento protocolizado. El usuario recibe el documento protocolizado y se retira del registro.

Archivo: corresponde con la fase final del proceso de inscripción de cualquier documento de los registros principales. Durante este proceso se archivan las copias del documento protocolizado, se archiva una en el protocolo principal y la otra copia en el protocolo duplicado. Para ello el funcionario de archivo recibe copias del documento protocolizado después del otorgamiento y agrega los recaudos al Cuaderno de Comprobantes, para llevar el control de todos los recaudos que se procesan. Ingresa al Libro Índice los datos del otorgamiento, lo cual sirve para llevar el control de los documentos otorgados y agilizar la búsqueda posterior del documento. El funcionario guarda la copia del documento protocolizado en el protocolo principal y en el duplicado y se folia el documento.

Debido a la complejidad del proceso explicado, su importancia en la gestión registral actual en la República Bolivariana de Venezuela y las facilidades que brinda la informática, es que se decide realizar un sistema informático integrado, que unifique y estandarice los procesos de Inscripción de Documentos en todas las Oficinas de Registro. Para desarrollar el sistema es necesario conocer además las metodologías de desarrollo de software, las tendencias en programación, las tecnologías actuales de desarrollo.

1.3. Desarrollo de software. Tendencias y tecnologías

1.3.1. Paradigmas de programación.

A lo largo de la evolución del desarrollo de software, los lenguajes de programación han ido atravesando por varios paradigmas de programación.

Entre estos paradigmas se encuentra la programación imperativa que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea.

Otro paradigma es el declarativo, el cual le dice al ordenador qué hacer, pero no cómo hacerlo, o sea, se describe el problema que se quiere solucionar, pero no las instrucciones necesarias para solucionarlo. La solución se logrará mediante mecanismos internos de inferencia de información a partir de la descripción realizada.

Pero sin lugar a dudas, el paradigma de Programación Orientada a Objetos (POO) desde su aparición revolucionó la forma de pensar a la hora de programar y trajo consigo muchos beneficios, que dieron gran impulso a la construcción de software cada vez más complejo. “La programación orientada a objetos es una nueva forma de pensar acerca de lo que significa computar, acerca de cómo podemos estructurar la información dentro de un computador”. (4)

1.3.2. Programación Orientada a Objetos

Una de las características principales de la POO es que sigue con frecuencia el mismo método que se aplica en la resolución de problemas de la vida diaria. El diseño está dirigido por las responsabilidades. Para lograrlo se basa en la simulación del mundo real mediante objetos, clases, mensajes, métodos, herencia, entre otros. A continuación se explicarán estos conceptos para su mejor comprensión.

Objetos: El concepto fundamental de la POO es el objeto. Los objetos representan cosas reales o abstractas del universo del problema a resolver y poseen un nombre que los identifica. Tienen responsabilidades y comportamientos bien definidos. Son consistentes, coherentes y completos; y pertenecen a una categoría o clase.

Clases: "Todos los objetos son ejemplares de una clase". (4). La clase define los atributos y las operaciones que la caracterizan. Es decir, las clases son como plantillas para crear objetos. Todos los objetos creados a partir de una misma clase, tienen los mismos atributos y operaciones, aunque son únicos, debido a que toman valores y estados diferentes.

Mensajes y métodos: Cualquier acción se inicia mediante la transmisión de un mensaje a un objeto que es responsable de la acción, o sea, de ejecutar determinado método.

Abstracción: La abstracción permite realizar generalizaciones, simplifica la realidad ignorando los detalles que no tienen relevancia en el universo del problema que se modela y se enfoca en las cosas comunes, pero permitiendo las variaciones.

Herencia: Fomenta la semejanza del modelado de la solución al mundo real, mediante la agrupación de los conceptos comunes a varias clases en una superclase, permitiendo la organización en categorías, donde el conocimiento de una categoría superior, puede ser aplicado a una categoría inferior.

Polimorfismo: "Es la habilidad de dos o más clases de objetos de responder a un mismo mensaje, cada una a su propia forma". (5).

Encapsulación. Es un principio en que se basa la POO que consiste en mostrar solo el qué y no el cómo, exponiendo solo la vista del cliente de las responsabilidades del objeto.

¿Cuáles son los beneficios de la POO y por qué ha ganado tanta popularidad?

Los sustantivos son las palabras primarias que más se usan (objetos), los cuales son calificados por los adjetivos (atributos) y finalmente se asocian con verbos (métodos). Por lo tanto, la POO es frecuentemente más fácil de comprender y modelar, ya que se asocia directamente con las situaciones del mundo real, como se había dicho anteriormente.

Logra una reducción de la complejidad a la hora de construir el software ya que los objetos exponen solo su interfaz pública, escondiendo por lo tanto, los detalles de implementación y evitando las complejas interdependencias en el código.

Las interfaces promueven un código flexible y robusto y son ideales para particionar responsabilidades individuales y del equipo, logrando así un incremento de la productividad y la eficiencia.

La pérdida de acoplamiento y la modularidad facilitan la extensibilidad, la flexibilidad, la escalabilidad y la reusabilidad, así como que los cambios sean más fáciles de implementar y mantener.

Un buen diseño del conjunto de objetos permite que nuevas funcionalidades sean añadidas incrementalmente y de forma no invasiva.

1.3.3. Programación Orientada a Aspectos

A pesar de las grandes ventajas que ofrece la POO, esta no evita que sigan quedando funciones dispersas en el código del programa, teniendo el inconveniente de tener que modificar varias clases para realizar la integración de varias funciones por lo que produce un enmarañamiento de la funcionalidad del sistema.

La programación orientada a aspectos (POA) viene a resolver esa problemática aspirando a soportar la separación de competencias para los aspectos de la aplicación. Pero, ¿qué se entiende por aspecto? “Un aspecto es una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa” (6)

Le sigue al paradigma de la orientación a objetos, pero, a pesar de esto, la POA no se puede considerar como una extensión de la POO, ya que puede utilizarse con los diferentes estilos de programación.

Entre los objetivos que se ha propuesto la programación orientada a aspectos están principalmente el de separar conceptos y el de minimizar las dependencias entre ellos. De la consecución de estos objetivos se pueden obtener las siguientes ventajas:

- Un código menos enmarañado, más natural y más reducido.

- Una mayor facilidad para razonar sobre las materias, ya que están separadas y tienen una dependencia mínima.
- Más facilidad para depurar y hacer modificaciones en el código.
- Se consigue que un conjunto grande de modificaciones en la definición de una materia tenga un impacto mínimo en las otras.
- Se tiene un código más reusable y que se puede acoplar y desacoplar cuando sea necesario.

Limitaciones:

Hasta ahora los lenguajes orientados a aspectos específicos y de propósito general han elegido utilizar un lenguaje base existente. Aparte de las ventajas de esta elección, esto trae consigo una serie de problemas. En el caso de los lenguajes de dominio específico no se es completamente libre para diseñar los puntos de enlace (recurso que utiliza la POA) sino que se restringe a aquellos que pueden ser identificados en el lenguaje base.

También los entornos de programación orientada a aspectos que han habido hasta ahora no soportan una separación de funcionalidades suficientemente amplia, o bien el lenguaje de aspecto soporta el dominio de aspecto parcialmente, o no se sostiene bien la restricción del lenguaje base.

1.3.4. Patrones de diseño orientado a objetos

Según Christopher Alexander: “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma” (7). Existen muchos tipos de patrones, aunque esta investigación solo se centrará en los patrones de diseño orientados a objetos.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los mismos identifican las clases, instancias, roles, colaboraciones y la distribución de responsabilidades para darle solución a los problemas.

Entre los patrones de diseño se pueden mencionar los patrones de asignación de responsabilidades GRASP y los patrones GOF, siglas de Gang of Four, que es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns (8)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro de este grupo de patrones se encuentran los siguientes: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Fabricación Pura, Indirección, Variaciones Protegidas, No hables con extraños y Polimorfismo.

Los patrones de diseño GOF son 23 y se clasifican según su propósito en Creacionales, Estructurales, y de Comportamiento. Y según su ámbito en Objeto y de Clase. Para ver más detalladamente estas clasificaciones, debe consultar el Anexo 1.

1.3.5. Metodologías de desarrollo de software

Durante el desarrollo de un proyecto de software una metodología define quién debe hacer qué, cómo y cuándo debe hacerlo, o sea, provee una guía para llevar a cabo el proyecto, con la calidad requerida y en el tiempo planificado.

A lo largo del desarrollo de la Ingeniería de Software, se han propuesto muchas metodologías, que han sido efectivas en su momento, pero con el intenso revolucionar de las tecnologías y los paradigmas, han ido surgiendo nuevas que dan respuesta a los problemas actuales. Ningún proyecto es igual a otro por lo que las metodologías deben ser configurables. Tampoco existe una metodología universal adaptable a todos los proyectos. Lo cierto es que cada una tiene sus características principales, que se han de tener en cuenta en el momento de escoger cuál se debe utilizar para desarrollar el software.

La comparación y clasificación entre las diferentes metodologías existentes no es una tarea fácil debido a la gran variedad de propuestas, el grado de detalle, información disponible y el alcance de cada una. A grandes rasgos, se pueden dividir las metodologías en dos grandes grupos: las ágiles y las tradicionales.

1.3.5.1. Metodologías ágiles

Las metodologías ágiles son, sin dudas, uno de los temas recientes en Ingeniería de Software que están acaparando gran interés. Surgen por la necesidad de reducir drásticamente los tiempos de desarrollo y mantener aún así la calidad.

Están pensadas para grupos pequeños de desarrollo donde el cliente es un miembro más que participa e influye directamente. Son efectivas en proyectos con requisitos poco definidos y cambiantes, ya que están especialmente preparadas para el cambio. Existen pocos roles y durante el desarrollo se producen la mínima cantidad de

artefactos, disminuyendo drásticamente el volumen de documentación y centrándose en obtener entregas pequeñas en iteraciones cortas. No es un proceso riguroso y cuenta con pocos principios, por lo que son sencillas de aprender.

Dentro de las metodologías ágiles se pueden mencionar:

- Extreme Programming.
- Scrum.
- Familia de Metodologías Crystal.
- Feature Driven Development.
- Proceso Unificado Rational, una configuración ágil.
- Dynamic Systems Development Method.
- Adaptive Software Development.
- Open Source Software Development.

Una vez analizadas las características generales de las metodologías de desarrollo ágiles, se deduce que no se ajustan a las características del proyecto, por lo que no se profundizará en ninguna en específico. Las razones por las que se plantea lo anterior son las siguientes:

- El proyecto a desarrollar es de gran magnitud y repercusión.
- El equipo de desarrollo está compuesto por más de 60 miembros.
- El cliente es extranjero (venezolano) por lo que no puede estar presente durante el desarrollo del mismo, así que todos los requisitos deberán tenerse en cuenta antes de comenzar a implementar.

1.3.5.2. Metodologías tradicionales

Las metodologías tradicionales se diferencian de las ágiles en muchos aspectos.

Son necesarias en proyectos de gran envergadura y equipos de desarrollo grandes, donde la aplicación de una metodología ágil no tendría cabida, debido a las características antes mencionadas. Están enfocadas a los procesos y requieren de normas y estándares para el manejo oportuno de dichos procesos. Tienen cierta resistencia al cambio por lo que tratan de capturar lo más tempranamente posible lo que desea el cliente. Generan gran cantidad de artefactos y existen muchos más roles que en una metodología ágil.

Dentro de las metodologías tradicionales se pueden citar:

- OPEN

- Métrica 3
- RationalUnifiedProcess (RUP)

OPEN

Object-oriented Process, Environment, and Notation (OPEN) es una metodología de desarrollo de software que abarca el ciclo completo de desarrollo, está orientada a procesos, diseñada para programación orientada a objetos, y desarrollo basado en componentes.

Es una metodología altamente flexible, que cuenta con un marco de procesos adaptables a individuos y proyectos, teniendo en cuenta las habilidades personales, cultura organizacional y requerimientos particulares. Esto se logra mediante la instanciación de los procesos definidos en el marco de procesos, escogiendo las actividades, tareas y técnicas específicas y sus configuraciones.

Provee un fuerte soporte para todo el ciclo de vida del software, incluyendo gestión de proyectos, modelado de procesos del negocio y guías estratégicas para la migración. La primera preocupación de OPEN es la calidad del software y el uso de métricas, teniendo sus procesos estrecha relación con el Modelo de Madurez y Capacidades del Instituto de Ingeniería de Software (SEI, por sus siglas en inglés).

Métrica 3

Es una metodología de desarrollo de software concebida para abarcar el ciclo completo de desarrollo, asegurándose que el proyecto cumpla sus objetivos en términos de calidad, costes y plazos. Está basada en los últimos estándares de Ingeniería de Software y calidad y en la experiencia de los usuarios de las versiones anteriores para solventar los problemas o deficiencias detectados.

Tiene un enfoque orientado a procesos centrándose en la clasificación y definición de los mismos. Los procesos se descomponen en actividades, y éstos en tareas. La automatización de las tareas propuestas está soportada por una amplia gama de herramientas y cubre tanto el desarrollo orientado a objetos, como la programación estructurada.

Los procesos de la estructura principal de Métrica son los que aparecen a continuación. El proceso de desarrollo de sistemas de información, por ser más complejo se encuentra subdividido en cinco procesos.

- Planificación de Sistemas de Información.

- Desarrollo de Sistemas de Información.
    - Estudio de viabilidad del sistema.
    - Análisis del Sistema de Información.
    - Diseño del Sistema de Información.
    - Construcción del Sistema de Información.
    - Implantación y Aceptación del Sistema
- Mantenimientos de Sistemas de Información.

Tiene además cuatro procesos de apoyo u organizativos: Gestión de Proyectos, Gestión de Configuración, Aseguramiento de Calidad y Seguridad.

Rational Unified Process (RUP)

El Proceso Unificado de Rational es el resultado de varios años de desarrollo y uso práctico, el cual sigue una historia desde el Proceso Objectory (primera publicación 1987), pasando por Proceso Objectory de Rational (1997), hasta el Proceso Unificado del Rational (publicado en 1998).

Como proceso define quién (trabajadores) está haciendo qué (artefactos), cómo (actividades) y cuándo (flujos de trabajo). La vida del sistema transcurre a través de ciclos de desarrollo. Cada ciclo se divide en cuatro fases fundamentales: Inicio, Elaboración, Construcción y Transición; y concluye con una versión del producto. Cada fase se subdivide en iteraciones que desarrollan actividades de nueve flujos de trabajo, lo que cada uno en mayor o menor medida dependiendo de las iteraciones y la fase en que se encuentre. De los nueve flujos, seis son ingenieriles y tres de apoyo.

1.3.5.3. RUP como metodología a utilizar.

Una vez analizadas las metodologías anteriores se ha seleccionado la metodología RUP por tener varios aspectos a su favor. En RUP se genera gran cantidad de documentación que es requerida por la dinámica del proyecto y exigida por parte del cliente, ya que en el mismo el personal varía y se necesita tener documentado lo que se ha hecho anteriormente por otros miembros del proyecto que pueden no encontrarse presente. Además, el equipo está altamente capacitado en la metodología, y la introducción de una nueva metodología produciría un atraso del cronograma de ejecución, ya que a diferencia de las metodologías ágiles, las metodologías tradicionales requieren de un estudio previo a su utilización. Otras

razones son que RUP es una metodología altamente configurable que ha dado buenos resultados a lo largo de su utilización, y existe bastante documentación de la misma.

Una vez elegida la metodología de desarrollo de software, es necesario determinar sobre qué plataforma se va desarrollar el software y las herramientas que se utilizarán a lo largo del desarrollo del software.

1.3.6. Plataformas de desarrollo

Una plataforma es un ambiente de hardware o software en el cual los programas se ejecutan. Algunas de las plataformas más populares son Microsoft Windows, Linux, Solaris OS[3], y Mac OS. La mayoría de estas plataformas pueden ser descritas como una combinación de sistema operativo y el hardware subyacente.

1.3.6.1. Plataforma Java

La plataforma Java, como las que se analizarán más adelante, difiere del resto de las plataformas mencionadas anteriormente en que es una plataforma de solo software, que corre sobre otras plataformas basadas en hardware como las que se mencionan al inicio de este epígrafe. Esta plataforma, unida con las potencialidades ofrecidas por el lenguaje de programación Java, forman las tecnologías Java, provistas por Sun Microsystem, que actualmente ha sido comprada por Oracle Corporation.

Esta plataforma está compuesta por dos bloques fundamentales:

- Máquina virtual de java (JVM)
- Interfaz de Programación de Aplicaciones de Java (API)

La máquina virtual de Java es la base de la plataforma y está adaptada a diferentes plataformas basadas en hardware, permitiendo a los programas desarrollados sobre ella ejecutarse en diferentes sistemas operativos. Esto se logra ya que cualquier programa escrito en Java (extensiones .java) se compila a extensiones .class que en lugar de tener código nativo del procesador, contiene *bytecodes,* que es el lenguaje que interpreta la máquina virtual de Java.

[3] Operating System

Figura 2: Funcionamiento de la plataforma Java

Como la JVM está disponible para los diferentes sistemas operativos, un fichero .class es capaz de ejecutarse tanto en Windows como en Linux, etc.

Las API son una gran colección de componentes de software listos para usarse que proveen muchas funcionalidades útiles. Están agrupadas en librerías de clases e interfaces relacionadas, estas librerías son conocidas como paquetes.

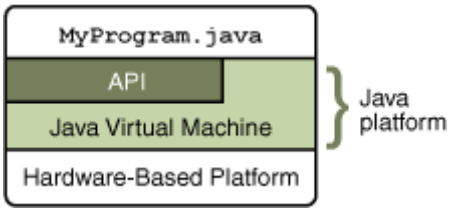

Figura 3: Plataforma Java

Ventajas de la plataforma Java:

- Es una plataforma muy robusta, y que cuenta con muchas herramientas integradas que permiten compilar, ejecutar, depurar, y documentar las aplicaciones.
- Lenguaje de programación relativamente fácil.
- Las API son el núcleo de funcionalidad del lenguaje de programación Java, incluyendo un amplio conjunto de clases útiles que se expanden desde los objetos básicos hasta funcionamiento en red, seguridad, generación de XML, y acceso a bases de datos entre otros.
- El código una vez escrito, puede ser ejecutado consistentemente en cualquier plataforma Java.
- No hay dependencia del código con las plataformas bases.
- Posee recolector de basura evitando desbordamientos de memoria.
- Los ejecutables son pequeños porque las librerías de clases vienen proporcionadas junto a la JVM de la plataforma concreta.

Desventajas:

Como un ambiente independiente de las plataformas, la plataforma Java puede ser un poco más lenta que el código nativo. Además, la plataforma permite desarrollo de aplicaciones solo en lenguaje de programación Java.

1.3.6.2. Plataforma .NET

La plataforma .NET de Microsoft tiene como principal objetivo conectar información, personas y dispositivos, bajo el principio de los servicios integrados que puedan ser accedidos remotamente, por ejemplo, a través de Internet. Intenta ofrecer una manera rápida pero a la vez segura y robusta de desarrollar aplicaciones permitiendo a su vez una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

La base de la plataforma.NET la constituye el framework o marco de trabajo, y este denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

Los principales componentes del framework son:

- El conjunto de lenguajes de programación.
- El Entorno Común de Ejecución para Lenguajes o CLR por sus siglas en inglés.
- La Biblioteca de Clases Base o BCL.

Conjunto de lenguajes de programación

La plataforma .NET es independiente del lenguaje utilizado, al contrario de la plataforma Java, pudiéndose realizar aplicaciones programadas en distintos lenguajes como pudieran ser: C++, Visual Basic, JScript, J#. Básicamente se puede integrar cualquier lenguaje al resto una vez que se construya un compilador que convierta dicho código a un lenguaje intermedio, que se explicará a continuación.

Entorno Común de Ejecución de Lenguajes.

El Entorno Común de Ejecución de Lenguajes o CLR (Common Language Runtime) es el verdadero núcleo del framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes. Esta herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL,

Microsoft Intermediate Lenguaje). A continuación se muestra una figura que describe el CLR.

Figura 4: Estructura Interna del CLR

Biblioteca de clases de .Net

La Librería de Clases Base (BCL, Base Class Library) es una librería incluida en el framework de .NET formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente cuando se está programando, a la vez este organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que resulta bastante sencillo encontrar lo que se necesita

La biblioteca de clases del framework de .NET incluye, entre otros, tres componentes clave:

- ASP.NET: para construir aplicaciones y servicios Web.
- Windows Forms: para desarrollar interfaces de usuario.
- ADO.NET: para conectar las aplicaciones a bases de datos.

Esta plataforma tiene varias ventajas, a continuación se resumen las más importantes:

Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL).

Compilación just-in-time (JIT): El compilador JIT incluido en el framework compila el código intermedio (MSIL) generando el código máquina propio para cada plataforma.

Recolector de Basura: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura. El CLR detecta cuando el programa deja de utilizar la memoria y la libera automáticamente.

Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.

La plataforma .Net presenta varias desventajas entre las que se encuentran que solo se puede ejecutar en entornos de la familia Windows, es decir, no es multiplataforma. También está bajo licencia propietaria, y el costo de desarrollar en ella es muy alto. Para correr aplicaciones desarrolladas en la misma se necesita instalar el framework, porque el ejecutable no es nativo, tiene que ser interpretado por el mismo. Además, por ser exclusiva de Microsoft, solo esta empresa es la única que puede añadir y quitar características según crean necesario.

1.3.6.3. Plataforma MONO

Mono es una plataforma de software diseñada para permitir a los desarrolladores crear aplicaciones multiplataforma. Es una implementación código abierto del framework .NET de Microsoft que está basado en las especificaciones definidas en ECMA[4].

En los puntos siguientes se comentan algunas de las partes de las que consta esta plataforma:

- Mono Runtime

Sería el equivalente al Common Language Runtime o entorno virtual de ejecución. Implementa un compilador JIT para el CIL de la máquina virtual, un compilador Ahead-of-Time (AOT), un cargador de clases, un recolector de basura, el sistema de hilos y las librerías de acceso a los metadatos.

- Librerías de clases base

Se ha buscado una compatibilidad total con la implementación .Net de Microsoft.

- Librerías de clases mono

[4] **ECMA International** es una organización internacional basada en membresías de estándares para la comunicación y la información

Mono también cuenta con muchas clases que van más allá de las librerías de clases bases ofrecidas por Microsoft. Estas proveen una funcionalidad adicional que son muy útiles, especialmente desarrollando aplicaciones Linux.

- Compilador C#

El compilador de C# de la plataforma Mono en estos momentos se considera que es un producto relativamente maduro.

Ventajas:

Esta plataforma trata de implementar las funcionalidades del framework de .NET por lo que reporta los mismos beneficios de las características implementadas del framework como gestión de memoria automática que incluye el recolector de basura, un entorno seguro de ejecución, interoperabilidad entre varios lenguajes, entre otras. A estos beneficios se le agrega el hecho de ser multiplataforma y código abierto, al contrario de .NET. La principal desventaja de esta plataforma es que es un tanto inmadura todavía, para la realización de aplicaciones de gran envergadura como complejos sistema de gestión.

1.3.6.4. Plataforma .NET para el desarrollo de la solución

De las plataformas analizadas se selecciona .NET puesto que la aplicación a realizar debe integrarse con la que controla los Registros Públicos y Mercantiles, la cual está desarrollada sobre esta plataforma.

La plataforma .NET tiene como entorno de desarrollo integrado IDE (Integrated Development Enviroment) el Visual Studio .NET. A continuación se explicarán las características del mismo como parte de esta plataforma.

1.3.6.5. Visual Studio .NET

Como ya se había dicho, Visual Studio .NET es el entorno de desarrollo integrado de Microsoft, que forma parte de la plataforma .NET por lo que tiene incorporado muchas de las funcionalidades específicas del framework.

Como cualquier otro buen entorno de desarrollo, Visual Studio .NET incluye un administrador de proyectos, un editor de código, diseñadores de interfaz de usuario, una buena cantidad de asistentes, compiladores, depuradores, herramientas y utilidades.

Soporta además la construcción de aplicaciones tanto como para plataformas de 32 como de 64 bits. Permite la construcción de aplicaciones sobre distintos lenguajes como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET. Facilita la programación de aplicaciones ya que se utilizan muchas funcionalidades brindadas por el framework como librerías de clases, recolector de basura, entre otras. O sea, permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET.

Existen ya varias versiones de este IDE que van desde Visual Studio 5.0 hasta Visual Studio 2010, contando con: Visual Studio 6.0, Visual Studio 2002, Visual Studio 2003, Visual Studio 2005 y Visual Studio 2008. La versión que se utilizará para el desarrollo de la aplicación es la 2008.

1.3.7. Frameworks de desarrollo

"En general, un framework o marco de trabajo es una estructura conceptual o real destinada a servir de apoyo o guía para la construcción de algo que expande la estructura en algo útil. En los sistemas informáticos, el framework es a menudo una estructura en capas que indican qué tipo de programas puede o debe ser construido y cómo se interrelacionan". (9)

Básicamente los frameworks son diseñados con la intención de facilitar el desarrollo de software, evitando tener que construir una aplicación desde cero.

1.3.7.1. Framework NHibernate

"Hibernate es un servicio poderoso, de alto rendimiento, para la persistencia y consulta relacional y de objetos". (10)

Permite el desarrollo de clases persistentes siguiendo el paradigma orientado a objetos, incluyendo asociación, herencia, composición, polimorfismo y colecciones. Se pueden realizar consultas tanto en lenguaje SQL[5] nativo, o en su propia extensión SQL portable.

NHibernate no es más que la implementación de Hibernate (Java), para .NET. Maneja la persistencia de objetos desde y hasta bases de datos relacionales. Dado una descripción en XML de las entidades y sus relaciones, NHibernate automáticamente genera código SQL para cargar y almacenar dichos objetos.

[5] Structured Query Language

Es un framework adecuado para aplicaciones centradas en los datos o para aplicaciones con modelos de procesos complejos de negocio. Su objetivo y ventaja fundamental de su utilización es la reducción del 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 persistencia de datos. Como desventaja fundamental se tiene que al trabajar con mecanismos de reflexión afecta el rendimiento del sistema y se obtienen tiempos de respuesta mayores, por lo tanto, ralentiza la aplicación.

1.3.7.2. Framework NUnit

NUnit es un framework para desarrollar pruebas unitarias para todos los lenguajes de .NET. Inicialmente portado de JUnit, es una herramienta que se encarga de analizar ensamblados generados por .NET, para interpretar las pruebas inmersas en ellos y ejecutarlas. Utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. Está escrito enteramente en C #, su licencia es basada en Open Source y ha sido completamente rediseñado para aprovechar ventajas de otros frameworks. En general, NUnit compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, caso contrario la prueba es exitosa.

Ventajas del framework NUnit:

- Código abierto.
- Pruebas unitarias automatizadas, por lo cual se hacen repetibles.
- Fomentan el cambio: ya que permiten probar cambios en el código y asegurar que en éstos no se hayan introducido errores funcionales; habilitan la refactorización o descomposición del código.
- Simplifican la integración: permiten llegar a la fase de integración con un grado alto de seguridad sobre el código.
- Documenta el código.
- Separa la interfaz y la implementación.
- Los defectos están acotados y fáciles de localizar.

Desventajas y limitaciones:

- 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.

1.3.7.3. Framework Spring.NET

Spring.NET es un framework de aplicación que proporciona apoyo de infraestructura completa para el desarrollo en .NET. Permite eliminar la complejidad del código utilizando clases base de las bibliotecas.

El diseño de Spring.NET se basa en la versión Java de Spring, que ha demostrado beneficios en el mundo real y se utiliza en miles de aplicaciones empresariales en todo el mundo. Spring proporciona una ligera solución para la construcción de aplicaciones, haciéndolo de una forma coherente y transparente para configurar e integrar la AOP (Aspect Oriented Programming) dentro del software.

El framework Spring tiene las mejores prácticas que han demostrado a lo largo de los años en numerosas aplicaciones y formalizado como patrones de diseño.

1.3.7.4. Framework para la Gestión de la Capa de Presentación.

Es un framework que se ha venido utilizando desde que comenzó el proyecto Registro y Notarías en su primera fase. Es importante destacar que este había comenzado su desarrollo en otro proyecto de software con características arquitectónicas similares, pero ya desde esa fase fue transformado para las características específicas de este proyecto. Este tiene elementos del patrón Modelo-Vista-Controlador (MVC) que quedan encapsuladas en la capa de presentación por ser la que está compuesta por los elementos de este patrón. El uso de este framework no influye en ningún caso en el estilo escogido para el desarrollo de la aplicación, su utilización se limita a la capa de presentación.

Entre las ventajas del framework escogido están las siguientes:

- Permite separar la lógica de presentación de los datos de los formularios, mediante la definición de Acciones que son las clases que gestionan el flujo de información desde y hasta los formularios.
- Seguridad a nivel de aplicación. Mediante un fichero XML de configuración se definen roles y en base a estos se gestiona el acceso a las acciones, y por lo tanto, el acceso a la información que contienen los formularios.
- Colores de interfaz configurable. Se puede cambiar el color de toda la aplicación con solo unos pocos cambios en la configuración de éste.
- Configuración del Menú de Opciones. Mediante un fichero XML de configuración se puede cambiar la estructura del menú de opciones, donde cada opción puede contener opciones y así sucesivamente por varios niveles.

Entre las desventajas del framework escogido se pueden ver las siguientes:

- Complejidad. Incrementa la naturaleza de basada a eventos del código de la interfaz de usuario, lo cual puede ser más fácil de depurar.
- Interfaz. El framework posee una interfaz al estilo Web donde en la parte derecha se muestra el menú con todas las opciones, si se desea cambiar esta apariencia puede conllevar un gran esfuerzo por el equipo de desarrollo, partiendo desde el punto que se cuente con el código del mismo, de lo contrario es imposible.

1.3.8. Herramientas CASE[6]

"La ingeniería de sistemas asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo". (11)

Existe actualmente gran variedad de dichas herramientas las cuales cubren distintas fases del proceso de desarrollo: análisis y diseño, planificación, modelado de procesos, y tienen diversas utilidades como semi-automatización de la generación de código en diferentes lenguajes, automatización de la documentación y mantenimiento del código.

1.3.8.1. Enterprise Architect

Es una herramienta CASE de modelado completo del ciclo de vida para sistemas de negocio y Tecnologías de la Información, ingeniería de software y sistemas, desarrollo en tiempo real y embebido. Abarca análisis, diseño, implementación, pruebas y mantenimiento, usando UML[7] 2.1 y otros estándares abiertos para modelado. Entre sus funcionalidades están:

- Soporta la generación e ingeniería inversa de código fuente para muchos lenguajes como son ActionScript, Ada, C y C++, C#, Java, Delphi, Verilog, PHP, VHDL, Python, System C, VB.Net, Visual Basic, entre otros.
- Integración con los entornos de desarrollo Eclipse y Visual Studio .NET.
- Permite depurar, compilar y visualizar código ejecutable incluyendo capacidades de depuración para Java, .NET y Microsoft Native (C++, C y VB).
- Genera clases de prueba NUnit y JUnit desde clases origen, integrando el proceso de compilación, prueba, ejecución y despliegue.

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

- Con respecto al modelado de bases de datos extiende el UML para agregar funcionalidades como disparadores, restricciones, entre otras, pudiendo además generar automáticamente scripts DDL para nueve gestores de bases de datos.
- 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.
- Provee trazabilidad completa desde los modelos de requisitos, análisis y diseño, hasta la implementación y despliegue.
- Es una herramienta potente de generación de documentación ya que cuenta con un editor de plantillas WYSIWYG[8] que permite la generación de reportes y documentos de manera fácil. Además de que permite generar versión HTML de los modelos realizados para su posterior publicación en la red.
- Para el modelado de procesos de negocio Enterprise Architect complementa UML 2.1 con soporte BPMN[9] y elementos de extensión para análisis, administración de requisitos y administración de procesos.
- Permite una administración efectiva del proyecto mediante la asignación de recursos a los elementos, la medición del esfuerzo y riesgos, la estimación del tamaño y complejidad del proyecto y la implementación del control de cambios y de procedimientos de mantenimiento.

1.3.8.2. Visual Paradigm for UML

Visual Paradigm for UML es uno de los productos contenidos en la suite Visual Paradigm. Es una herramienta de diseño multiplataforma que soporta varios estándares para modelar visualmente la aplicación y generar documentación, código fuente y bases de datos.

¿Qué provee Visual Paradigm for UML?

- Permite visualizar, diseñar, comunicar y documentar diagramas UML 2.2 con un intuitivo y sofisticado entorno de modelado visual.
- Para el modelado de negocio incluye mapas de procesos, diagramas de flujos de datos y gráficos organizacionales permitiendo visualizar, entender, analizar y mejorar los procesos del negocio, usando BPMN.

[8]Siglas de What You See Is What You Get que significa “Lo que ves es lo que obtienes” Se aplica a los procesadores de texto y otros editores de texto con formato que permiten escribir un documento viendo directamente el resultado final.
[9] Business Process Modeling Notation, Notación para el Modelado de Procesos de Negocio

- Permite diseñar y generar bases de datos para muchos gestores.
- Para el modelado de requerimientos permite capturar, organizar, administrar y realizar requerimientos usando diagramas de requerimientos SysML[10].
- Provee de matrices de dependencia y diagramas de análisis para llevar la trazabilidad de los elementos de los modelos.
- Permite el trabajo en equipo mediante la instalación de VP Teamwork Server, Subversion, Perforce o CVS[11]
- Permite generar reportes del sistema en formato HTML, PDF, y MS Word, las plantillas son configurables y se puede publicar en la red. También implementa técnicas para el análisis de impacto como matrices de dependencia y diagramas de análisis.

1.3.8.3. Rational Rose Enterprise Edition 2003

El producto Rational Rose Enterprise Edition de IBM es uno de los más integradores en la familia de Rational Rose. Está basado en UML. Entre sus principales características se encuentran las siguientes.

- Soporte a patrones de Análisis, ANSI C++, Rose J and Visual C++.
- Basado principalmente en el nivel de integración que tiene este con el resto de las herramientas que lo acompañan en la Suite entre las que aparecen:
- Rational Clear CASE, para el control de versiones.
- Rational Clear Quest, para el control de cambios.
- Rational Model Integrator, para la integración de los artefactos.
- Rational Requisite Pro, herramienta de administración de requerimientos.
- Brinda la posibilidad de generar y realizar ingeniería inversa en una buena cantidad de lenguajes de programación.
- Incluye un agregado para el modelar aplicaciones Web.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurable.
- Permite el diseño de bases de datos.

1.3.8.4. ER/Studio

Es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades

[10] Systems Modeling Language, Lenguaje de Modelado de Sistemas
[11] Concurrent Versioning System, Sistema Concurrente de Versiones

diarias de los administradores, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de las bases de datos grandes y complejas.

ER/Studio (ERS) está equipado para crear y manejar diseños de bases de datos funcionales y confiables, ofreciendo las siguientes funcionalidades:

- Capacidad fuerte en el diseño lógico.
- Sincronización bidireccional de los diseños lógico y físico.
- Construcción automática de base de datos.
- Reingeniería inversa de bases de datos.
- Documentación basada en HTML.

Genera otros objetos de base de datos: vistas, procedimientos almacenados, defaults, 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. Además soporta un gran número de gestores de base de datos.

El ERS tiene la posibilidad de realizar diagramas con un desempeño claro y rápido. También es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Es posible crear sub-vistas para separar y manejar áreas importantes. El ERS automáticamente mantiene todas las dependencias entre sub-vistas y el diagrama completo.

1.3.8.5. Selección de las herramientas a utilizar

Las herramientas elegidas para la realización del proyecto son el Enterprise Architect y el ER/Studio. A continuación se explicarán las razones de su elección.

De las herramientas analizadas, Rational Rose, Visual Paradigm for UML y Enterprise Architect cubren casi todo el ciclo de vida del desarrollo de software, mientras que el ER/Studio se centra en el diseño de bases de datos, aunque las otras tres tienen funcionalidades para el modelado de datos, se eligió el ER/Studio ya que es una herramienta más profesional y especializada en todo lo referente al diseño de bases de datos.

Analizando sus entornos de trabajo se puede decir que Rational Rose se ha quedado atrás con respecto a las facilidades de diseño que brindan, ya que tanto el Enterprise Architect y el Visual Paradigm for UML son más amigables y vistosos. La generación de documentación es posible tanto en el Visual Paradigm for UML como en el Enterprise Architect, mientras que en Rational Rose lo que se hace es publicar el proyecto en HTML. En cuanto a los requerimientos del sistema, el más pesado en

cuanto a necesidades de memoria, tanto RAM como espacio en disco, es el Visual Paradigm, y el que menos recursos consume es el Enterprise Architect. Además el Enterprise Architect tiene una integración perfecta con el Visual Studio, que es el entorno de desarrollo perteneciente a la plataforma .NET, escogida para desarrollar el proyecto.

1.3.9. Sistemas gestores de Bases de Datos

Un Sistema de Gestión de Bases de Datos Relacionales (SGBDR), permite el almacenamiento de datos en tablas formadas por filas y columnas, y su posterior consulta y mantenimiento mediante un sencillo y potente lenguaje de consulta estructurado (SQL).
Bajo las siglas SGBDR se oculta toda la complejidad informática necesaria para gestionar:

- El acceso controlado de los procesos de los usuarios a los datos.
- La gestión del almacenamiento de los datos.
- La gestión de las comunicaciones entre procesos clientes y servidores.
- Interconexión con distintos protocolos y sistemas operativos.
- Acceso a los recursos conectados a la red.

1.3.9.1. PostgreSQL

PostgreSQL es un sistema gestor de bases de datos relacionales orientado a objetos con características de los mejores sistemas de bases de datos comerciales. PostgreSQL es libre y su código fuente completo está disponible.

El desarrollo de PostgreSQL es realizado por un equipo de desarrolladores en su mayoría voluntarios extendido por todo el mundo, que se comunican vía Internet. Se trata de un proyecto comunitario y no está controlado por compañía alguna.
Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones.

PostgreSQL ofrece muchas ventajas entre ellas:

- Código fuente disponible.
- Estabilidad y confiabilidad.
- Extensible.
- Multiplataforma.

- Diseño para ambientes de alto volumen.
- Herramientas gráficas de diseño y administración de bases de datos.

Una de las grandes desventajas de PostgreSQL es que al no ser desarrollado por una compañía, sino por una comunidad de desarrolladores, estos no se hacen responsables de los fallos que pueda presentar y como tal, no existe soporte formal. Hay algunas compañías comerciales que se dedican a esto, pero no son precisamente las creadoras del software. Esto es un aspecto clave que requirió el cliente y por lo cual este gestor no se eligió para la implementación de la solución.

1.3.9.2. Oracle

Oracle es uno de los SGBDR más conocidos en el mundo profesional, por sus características que lo hacen confiable y seguro. Se puede utilizar para el almacenamiento de todo tipo de datos como documentos, imágenes, multimedia, XML entre otros.
Es considerado como uno de los sistemas de bases de datos más completos, destacándose fundamentalmente: soporte de transacciones, estabilidad, escalabilidad, multiplataforma.

- Soporta todas las funciones que se esperan de un RDBMS: con un lenguaje de diseño de bases de datos muy completo, el (PL/SQL) que permite implementar diseños con disparadores y procedimientos almacenados, con una integridad referencial declarativa bastante potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso admite la administración de bases de datos distribuidas.
- Una característica importante de Oracle es la agrupación de conexiones mediante un clúster, mejorando así notablemente de forma significativa el rendimiento y la escalabilidad de una aplicación.
- Cuenta con un buen soporte por parte de la corporación.

Una de las desventajas que presenta la utilización de este potente gestor es el elevado costo de sus licencias por lo que su uso está limitado fundamentalmente a empresas grandes y multinacionales.

Una vez analizados los gestores de Bases de Datos Relacionales anteriores se ha seleccionado el gestor Oracle por tener varios aspectos a su favor. En Oracle se puede trabajar de manera más eficiente con el motor de base de datos en todo el

entorno de trabajo, obteniendo un acceso más rápido y seguro a la información. También es el único que ofrece soluciones que se ajustan a las características y las necesidades que tiene el proyecto de consultar y consolidar grandes volúmenes de información que maneja la oficina SAREN, lo que ayuda a resolver los problemas con mayor rapidez. Otra razón por la que se escogió es la buena reputación que tiene la compañía en cuanto a soporte.

Otra funcionalidad de Oracle es la réplica de Bases de Datos, usado para mejorar la calidad de servicio y desempeño de la base de datos primaria, pudiendo así ser utilizada durante horas no pico como una base de datos de prueba, y también puede servir en cualquier momento como una solución para la recuperación ante desastres.

1.3.10. Arquitectura de software

“Arquitectura es un término que mucha gente intenta definir, con poco acuerdo. Hay dos elementos comunes: una es el desglose de nivel más alto de un sistema en sus partes; la otra, decisiones que son difíciles de cambiar. También progresivamente nos hemos dado cuenta de que no existe una única forma para indicar la arquitectura de un sistema; más bien hay arquitecturas múltiples en un sistema, y la vista de es arquitectónicamente significativo puede cambiar sobre la tiempo de vida de un sistema”. (12)

Este documento se regirá por la definición que se ha tomado como oficial por todas las instituciones que desarrollan software a nivel mundial, como ha sido por ejemplo Microsoft y es la brinda el documento de la IEEE Std 1471-2000 que plantea: “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”.

1.3.11. Estilos Arquitectónicos

No es objetivo de este epígrafe hacer un análisis de los distintos estilos arquitectónicos en su totalidad, sino solo aquellos que por su estructura y solución podrían considerarse para la construcción del sistema, analizando las ventajas y desventajas que estos poseen para determinar si debe ser o no aceptado para el desarrollo del software.

Se identifican los estilos arquitectónicos como un “conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la

forma en que se lleva a cabo la composición. Es en gran medida la interacción entre los componentes, mediados por conectores, lo que confiere a los distintos estilos sus características distintivas". (13)

1.3.11.1. Estilos de Flujo de Datos

Los estilos de flujos de datos como su nombre lo sugiere se refiere a sistemas que implementan transformaciones de datos en pasos sucesivos, así como un flujo, va en un solo sentido. Ejemplos de estas se encuentran las arquitecturas de proceso secuencial por lotes, red de flujos de datos, y tuberías y filtros.

A pesar de que es fácil de entender e implementar, al igual que enfatiza la reutilización y la modificabilidad, este estilo de arquitectura no se recomienda para la realización del sistema, puesto que es demasiado simplista y no permite el desarrollo de aplicaciones con lógica de negocios complicadas, como es el caso del presente sistema.

1.3.11.2. Estilos Centrados en Datos

Esta familia de estilos enfatiza en la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento. Sub-estilos característicos de la familia serían los repositorios, las bases de datos, las arquitecturas basadas en hipertextos y las arquitecturas de pizarra. (13)

1.3.11.2.1. Arquitecturas de Pizarra o Repositorio

El estilo de repositorio o pizarra tiene dos componentes fundamentales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él.

Se han utilizado principalmente en aplicaciones de inteligencia artificial como reconocimiento de patrones, reconocimiento de habla, en exploraciones recientes de inteligencia artificial distribuida o cooperativa, en robótica, en modelos multi-agentes, en programación evolutiva, en gramáticas complejas, en modelos de crecimiento afines a los L-Systems de Lindenmayer, entre otros. Muchos más sistemas de los que se cree están organizados como repositorios: bibliotecas de componentes reutilizables, grandes bases de datos y motores de búsqueda.

Por lo que se ha podido apreciar, este estilo arquitectónico no es afín con el tipo de aplicación que se desea desarrollar, que es una aplicación de gestión.

1.3.11.2.2. Estilos Peer-to-peer

Esta familia, también llamada de componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes. Cada entidad puede enviar mensajes a otras entidades, pero no controlarlas directamente. Los mensajes pueden ser enviados a componentes nominados o propalados mediante broadcast. Miembros de la familia son los estilos basados en eventos, en servicios y en recursos.

Esta independencia entre los procesos promueve el desarrollo en paralelo lo que resulta en una mejora del rendimiento, pudiéndose reemplazar componentes y agregarlos registrándolo para los eventos del sistema. Esta independencia a su vez trae dificultades como que un componente no puede utilizar los datos o el estado de otro componente para efectuar su tarea y que cuando un componente anuncia un evento, no tiene idea sobre qué otros componentes están interesados en él, ni el orden en que serán invocados, ni el momento en que finalizan lo que tienen que hacer. Esto trae como consecuencia que el estilo no permita construir respuestas complejas a funciones de negocios, por lo que no se adapta a la solución que se desea desarrollar, ya que la misma tiene un negocio bastante complejo.

1.3.11.3. Estilos de Llamada y Retorno

Los estilos de Llamada y Retorno enfatizan la modificabilidad y la escalabilidad, son los estilos más utilizados o más generalizados por los sistemas a gran escala. Dentro de esta familia se encuentran el Modelo-Vista-Controlador (MVC), la Arquitectura Orientada a Objetos y la Arquitectura en Capas.

1.3.11.3.1. Modelo-Vista-Controlador

Reconocido como estilo arquitectónico por Taylor y Medvidovic, considerado una micro-arquitectura por Robert Allen y David Garlan, en ocasiones definido más bien como un patrón de diseño o como práctica recurrente, tratado a veces en términos de un estilo decididamente abstracto, el Modelo-Vista-Controlador intenta dar solución al problema que se presenta cuando se unen en una sola pieza el almacenamiento de datos, la lógica de la información y la presentación de la misma, con el objetivo ingenuo de reducir la cantidad de código y optimizar el rendimiento. El problema está precisamente en el hecho de que la interfaz suele cambiar, o acostumbra depender de

distintas clases de dispositivos (navegadores, PDA[12]); la programación de interfaces de HTML, además, requiere habilidades muy distintas de la programación de lógica de negocios.

Este patrón separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: el modelo, la vista y el controlador, respectivamente. Las ventajas de utilizar este patrón radican en el soporte de múltiples vistas y la adaptación al cambio, ya que los requisitos de interfaz pueden cambiar, mientras que las reglas del negocio no suelen hacerlo.

Para el desarrollo del sistema se escogió un framework creado en la primera fase del Proyecto RN el cual se basa en el patrón MVC, aprovechando así las ventajas de dicho patrón y permitiendo que la aplicación se pueda desarrollar con otra arquitectura que soporte mejor los complejos procesos de negocio que se desean implementar.

1.3.11.3.2. Arquitectura en Capas

La división en capas es una de las técnicas más comunes que utilizan los diseñadores de software para separar un sistema de software complicado. Se puede ver en arquitecturas de máquina, donde las capas descienden de un lenguaje de programación con llamadas de sistema operativo en los controladores de dispositivos y conjuntos de instrucciones del microprocesador y en las puertas lógicas dentro de los chips.

En este esquema, las capas superiores se sirven de los servicios brindados por las capas inferiores, pero las capas inferiores desconocen de las capas superiores. Aun más, cada capa usualmente esconde sus capas inferiores a las capas superiores por lo tanto la capa 4 usa los servicios de capa 3, que utiliza los servicios de capa 2, pero la capa 4 no es consciente de la capa 2. (No todas las arquitecturas en capas son opacas como esta, pero la mayoría lo son).

Entre los beneficios de esta arquitectura se encuentran:

- Puede comprender una sola capa como un todo coherente sin saber mucho sobre las otras capas.
- Puede sustituir las capas con implementaciones alternativas de los mismos servicios básicos.
- Minimizar las dependencias entre las capas.

[12] Personal Digital Assistent, asistente digital personal.

- Una vez que tenga una capa construida, se puede utilizar para muchos servicios de nivel superiores.
- Mantenibilidad: provee una organización lógica de aplicación y desarrollo.
- Escalabilidad y rendimiento: permite distribuir una aplicación, cada capa puede residir en un computador distinto y agregar máquinas mejora el rendimiento.
- Seguridad: permite aislar componentes.
-

También presentan sus desventajas como son:

Las capas encapsulan algunas cosas bien, pero no todas. Como resultado, se obtienen a veces cambios en cascada. El ejemplo clásico de esto en una aplicación empresarial en capas es agregar un campo que necesita para mostrar en la interfaz de usuario, debe estar en la base de datos y por lo tanto debe agregarse a cada capa intermedia.

Las capas adicionales pueden perjudicar el rendimiento. En cada capa, la información normalmente necesita ser transformada de una representación a otra.

1.3.12. Calidad de Software

1.3.12.1. Métricas

La medición es un elemento clave en cualquier proceso de ingeniería. Las medidas se emplean para comprender mejor los atributos de los modelos que se crean y evaluar la calidad de los productos de la ingeniería o de los sistemas que se construyan.

Aunque las métricas no suelen ser absolutas, proporcionan una manera sistemática de evaluar la calidad a partir de un conjunto de reglas definidas con claridad. También proporcionan al ingeniero información inmediata y en el sitio, no posterior al hecho, teniendo la ventaja de describir y corregir problemas potenciales antes de que se conviertan en efecto catastróficos.

Para que resulte útil en realidad una métrica debe ser simple y calculable, persuasiva, consistente y objetiva. Debe ser independiente del lenguaje de programación y proporcionar una retroalimentación efectiva al ingeniero.

Se han propuesto gran variedad de clasificaciones para las métricas, a continuación se contemplan las áreas más importantes (14):

1. Métricas para el modelo de análisis.

Las métricas para el modelo de análisis se concentran en la función, los datos y el comportamiento, que son elementos del modelo de diseño e incluyen: funcionalidad entregada, tamaño del sistema y calidad de la especificación.

2. Las métricas para el modelo de diseño.

Las métricas para el diseño cuantifican los atributos del diseño para poder evaluar la calidad del mismo. Entre las mismas se incluyen:

- ❖ Métricas arquitectónicas: consideran los aspectos estructurales del modelo de diseño.
- ❖ Métricas al nivel de componentes: indican la calidad del módulo al establecer medidas indirectas para la cohesión, el acoplamiento y la complejidad.
- ❖ Métricas de diseño de la interfaz: proporcionan un indicio de la facilidad con la que se usa la interfaz gráfica del usuario.
- ❖ Métricas especializadas en diseño orientado a objeto: se concentran en la medición que puede aplicarse a las características de clase y diseño (localización, encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos) que convierten a una clase en única.

3. Métricas para el código fuente.

Estas métricas, como su nombre lo indica, miden el código fuente y evalúan su complejidad, entre las que se encuentran:

- ❖ Métricas de Halstead: controversiales pero fascinantes, estas métricas proporcionan medidas únicas de un programa de cómputo.
- ❖ Métricas de complejidad: miden la complejidad lógica del código fuente.
- ❖ Métricas de longitud: proporcionan un indicio del tamaño de software.

4. Métricas para pruebas

Ayudan a diseñar casos de prueba efectivos y a evaluar la eficacia de las pruebas, se incluyen:

- ❖ Métricas de cobertura de instrucciones y ramas: conduce al diseño de casos de prueba que proporcionan cobertura del programa.

- Métricas relacionadas con los defectos: se concentran en encontrar defectos y no en las propias pruebas.
- Efectividad de la prueba: proporcionan un indicio en tiempo real de la efectividad de las pruebas aplicadas.
- Métricas en el proceso: métricas relacionadas con el proceso que se determinan a medida que se aplican las pruebas.

1.3.12.2. Pruebas de Caja Blanca y Caja Negra

"Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software". (14)

Se centran principalmente en los requisitos funcionales del software. Permiten encontrar: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a las bases de datos externas y errores de rendimiento, errores de inicialización y terminación.

La prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de condiciones, bucles o ambos. (14)

Requieren del conocimiento de la estructura interna del programa. Estas pruebas deben garantizar como mínimo que:

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

1.4. Conclusiones del capítulo

Luego del análisis realizado a lo largo de este capítulo se han desarrollado los principales aspectos abordados durante la investigación, se han dejado claros muchos de 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, haciendo énfasis en el Proceso de Inscripción de Documentos que se realiza en los mismos. Se realiza un análisis de las metodologías, tendencias y herramientas de desarrollo a utilizar en la solución propuesta destacando sus características esenciales, aquí se incluye todo lo referente a entornos de desarrollo, gestores de base de datos, metodologías y las principales herramientas a emplear.

Capítulo 2. Solución propuesta.

2.1. Arquitectura del sistema

La arquitectura seleccionada para la realización de la aplicación es la arquitectura en capas por todas las ventajas antes mencionadas, y además de que la mayoría de las aplicaciones empresariales no triviales utilizan una arquitectura en capas de alguna forma, porque es la más útil en este tipo de aplicaciones.

A continuación se presenta una imagen de la arquitectura propuesta por el equipo de arquitectura del proyecto.[13]

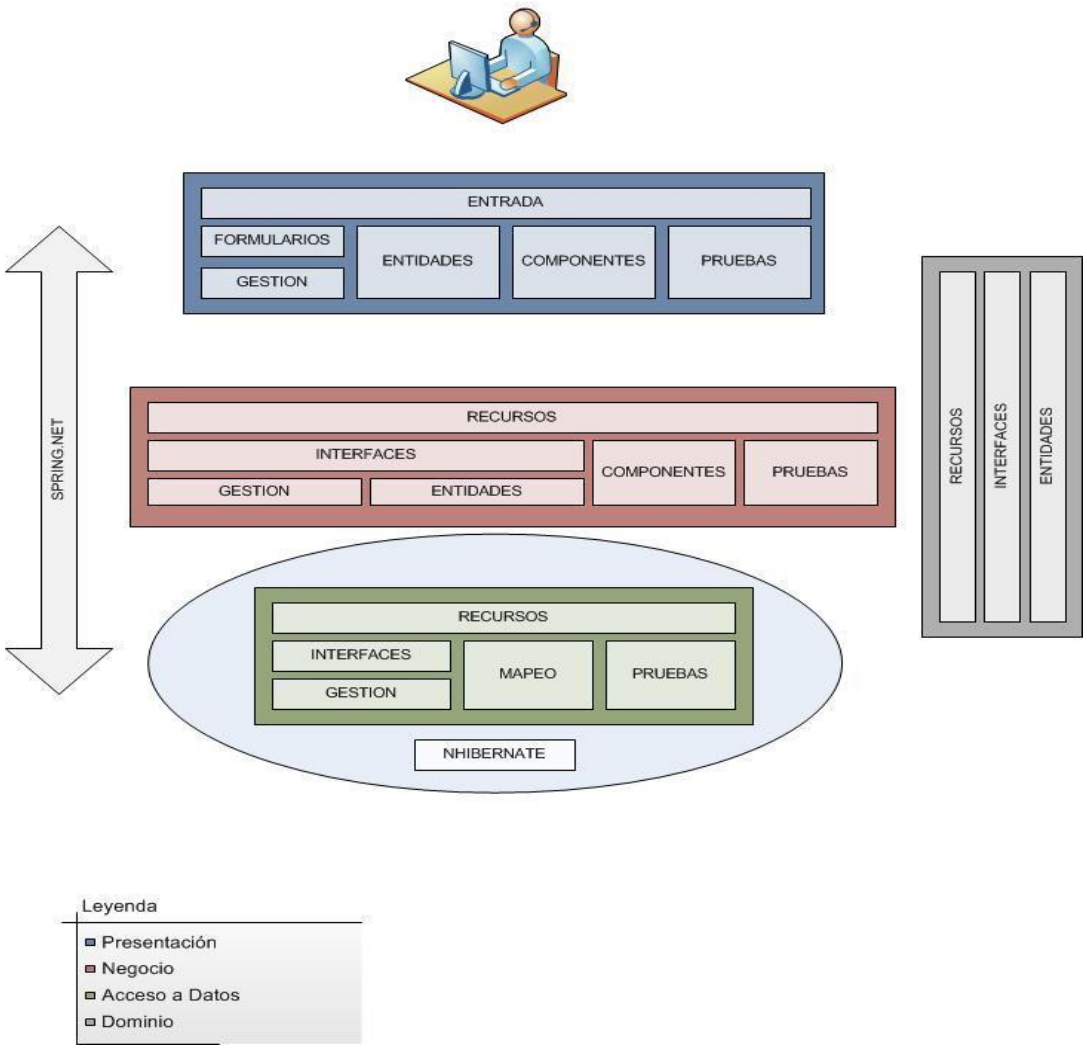

Figura 5: Arquitectura del sistema

[13] La información sobre la arquitectura de la solución fue extraído del documento Arquitectura de Software de RN, desarrollado por los arquitectos del proyecto.

Teniendo en cuenta las características del software que se está desarrollando así como las cualidades o atributos de calidad que se desean alcanzar, la Arquitectura sustenta su base en los componentes, siendo estos bloques de construcción que conformarán las partes del mismo. Dichos componentes se orientan en dos perspectivas Vertical y Horizontal según se ve en la figura anterior, asimismo se establecen los mecanismos de comunicación o conexión entre ellos para satisfacer los objetivos del sistema.

El enfoque vertical abarca la distribución de los componentes que dan vida a la Arquitectura para cada módulo de manera individual. En este sentido, se presenta un modelo multicapa según se ve 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.

Nivel Presentación

Es la capa superior, contiene los componentes con las que va a interactuar el usuario. Desde este nivel se pueden alcanzar las funcionalidades que brinda el negocio y mostrar o capturar la información a través de los diferentes elementos que comprende.

Nivel de Negocio

Su diseño depende directamente del negocio específico al que se refiera cada módulo. Esta capa recibe una petición del nivel superior, gestiona o procesa la misma, de ser necesario solicitándola a la capa de Acceso a Datos y finalmente envía una respuesta

Nivel de Acceso a Datos

Controla todo lo concerniente a la información que se encuentra en la fuente de almacenamiento. Al ser la capa inferior los componentes que contiene desconocen los niveles superiores, únicamente se limitan al manejo de la información, ya sea para persistirla u obtenerla para su procesamiento y propagación por la aplicación.

Nivel de Dominio

Se encuentra distribuido verticalmente teniendo en cuenta que contiene todas las entidades y recursos que representan los valores correspondientes a los datos o servicios que se manejan en el dominio completo del módulo.

La arquitectura se sustenta en el uso de frameworks de desarrollo que facilitan enormemente el desarrollo de la aplicación

2.1.1. Frameworks

Los frameworks definidos por la Arquitectura juegan un papel muy importante en la mayoría de los servicios y recursos que satisfacen las funcionalidades del sistema.

- **Spring.net:**

Sobre este framework se construyen la mayoría de los elementos significativos que se pueden encontrar, tomando como premisa sus principales potencialidades las cuales tributan significativamente a la arquitectura, tales como Inyección de Dependencia, Programación Orientada a Aspectos (AOP por sus siglas en ingles), interoperabilidad e integración con otros framework utilizados, así como el hecho de ser un framework de software libre dando la posibilidad de tener acceso a su código fuente.

Administrador de Transacciones:

Una de las razones que justifican el uso de Spring.net es precisamente su trabajo con las transacciones. Dicho framework proporciona un Administrador de Transacciones el cual tiene las siguientes ventajas:

1. Garantiza un amplio manejo transaccional a través de las diferentes APIs de este tipo tales como ADO.NET, Enterprise Services, System.Transactions, and NHibernate.

2. Cuenta con instrucciones para el manejo de transacciones que soportan cualquiera de las tecnologías antes mencionadas.

3. Proporciona una simple interfaz con las funcionalidades que abarcan la mayoría de los eventos que se pueden encontrar durante el trabajo con transacciones.

- **NHibernate:**

La utilización de este framework en la solución está supeditada al uso de Spring.net puesto que ambos se encuentran integrados. En este sentido únicamente resultan significativos los ficheros de mapeo y algunas configuraciones independientes teniendo en cuenta que las restantes son absorbidas por Spring.net.

2.2. Descripción de los procesos

Se aborda a continuación la descripción de los procesos de inscripción antes descritos en el Capítulo 1, atendiendo a su solución con el sistema propuesto. Remítase al epígrafe 1.2.1 para ver más información sobre los mismos.

2.2.1. Revisión

Cuando el funcionario de revisión se dispone a comenzar dicho 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 desea crear un nuevo trámite el sistema permite asociar un usuario al trámite que se está creando. Para ello se pueden introducir campos para realizar la búsqueda del ciudadano en el centro de datos. En caso de no aparecer, se procede a su incorporación al sistema solo en caso de que sea extranjero, si es una persona venezolana debe primero inscribirse en el centro de datos. También permite la elección del tipo de acto a inscribir y los recaudos correspondientes al acto seleccionado. Una vez hecho lo anterior, se puede también opcionalmente asociar un representante jurídico al trámite. Después el sistema muestra otra interfaz que permite gestionar las exenciones, en caso de que existan, en el trámite en curso. Después se muestra una pantalla con el resumen del trámite. Si todo está correcto el funcionario de revisión elige finalizar la operación y el sistema crea el trámite, generando el número del trámite, la fecha de creación y actualizando el estado del mismo.

2.2.2. Cálculo

Para realizar el proceso de cálculo de un documento, lo primero que se necesita hacer es realizar una búsqueda del documento en cuestión, introduciendo datos en los campos de búsqueda. Se selecciona el trámite y se muestra el resumen de revisión obtenido en el proceso anterior. Se visualiza entonces los datos del usuario asociado al trámite y se da la posibilidad de cambiarlo en caso necesario. El funcionario de cálculo especifica entonces los conceptos de pago del trámite según el tipo de acto. Si todos los datos son introducidos correctamente el sistema genera la Planilla Única Bancaria (PUB) y actualiza el estado del trámite.

2.2.3. Presentación

Para realizar el proceso de presentación de un documento, lo primero que se necesita hacer es realizar una búsqueda del documento en cuestión, introduciendo datos en los campos de búsqueda. Una vez seleccionado el trámite se procede a gestionar el pago de la PUB, en donde se especifica los datos relacionados con el banco y el tipo de

pago, y se procede a agregar el pago en el sistema. Se visualiza entonces los datos del usuario 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 asigna protocolo, tomo y número al documento dándole de esta forma la ubicación en el archivo. El sistema guarda además los datos en el Libro de Presentación.

2.2.4. Procesamiento

El funcionario de procesamiento elige procesar documento de inscripción. El sistema realiza una búsqueda de los trámites y se selecciona el que va hacer procesado. Una vez seleccionado, el funcionario continúa la operación registrando los datos del documento a inscribir. Una vez insertados los datos el funcionario de procesamiento pasaría a elaborar la nota de registro del trámite, para eso se lanza una interfaz que permite gestionar la nota. Después de concluida la elaboración de la nota, se almacenan todos los datos y genera la fecha de procesamiento. Posteriormente el funcionario de revisión revisa el documento y la nota de registro, para eso selecciona el trámite deseado después de que el sistema haya realizado la búsqueda. Una vez seleccionado, el sistema visualiza la nota de registro y el funcionario puede revisar la nota y si esta correcta procede a continuar la operación, el sistema muestra un resumen del trámite permitiendo aceptar o denegar el documento, selecciona a todos los otorgantes como posibles firmantes y actualiza el estado del trámite terminado así el procesamiento del trámite.

2.2.5. Otorgamiento

El sistema permite seleccionar el archivo que se va a otorgar, después de que se haya realizado una búsqueda previa. El sistema muestra un resumen del trámite en curso, además de los testigos y los posibles firmantes. Si se desea adicionar un nuevo testigo, el sistema muestra una interfaz donde aparecen las personas asociadas al trámite como posibles testigos, se seleccionan los testigos deseados o se puede agregar nuevos testigos al sistema. También se puede agregar un nuevo otorgante, o representación jurídica. Opcionalmente se puede también agregar una nota aclaratoria. También se pueden seleccionar los otorgantes que pasaran a ser firmantes.

2.2.6. Archivo

El proceso inicia con la opción de buscar trámite. Una vez seleccionado el trámite se procede a su digitalización. Se permite además seleccionar recaudos y proceder a su

digitalización. Para finalizar el sistema muestra un resumen del trámite, así como la ubicación en el archivo del documento en trámite y la ubicación de los recaudos en el Cuaderno de Comprobantes.

2.3. Diseño e implementación

El diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos. El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

A continuación se explica de manera general el contenido de dichos modelos. Debido a la gran cantidad de diagramas generados durante el diseño se decidió no abarrotar este documento con imágenes. Las mismas se pueden encontrar en el documento Modelo de Diseño, que debe estar adjunto a este documento.

2.3.1. Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización de los casos de usos y sirve como abstracción del modelo de implementación y su código fuente. Es usado tanto para concebir como para documentar el diseño de un sistema de software. Incluye los diagramas de interacción, de clases y el modelo de datos.

2.3.1.1. Diagramas de clases.

Los diagramas de clases son muy utilizados durante el diseño para la representación de la estructura estática del sistema, mostrando las clases, sus atributos y las relaciones que se establecen entre las mismas.
Un diagrama de clases del diseño contiene la siguiente información: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de atributos, navegabilidad y dependencias.

2.3.1.2. Diagramas de Interacción

Muestran gráficamente la relación entre los objetos a fin de cumplir con los requerimientos. Estos diagramas se pueden expresar en diagramas de secuencia y en diagramas de colaboración. Los primeros muestran las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto y son los que encontrará en el Modelo de Diseño, los segundos destacan la organización de los objetos que participan en una interacción.

2.3.1.3. Modelo de Datos.

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos. Básicamente el modelo de datos está formado por tres elementos fundamentales que son: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos antes mencionados; y las relaciones, que son las que enlazan a dichos objetos entre sí.

2.3.2. Modelo de implementación

Un modelo de implementación muestra las dependencias entre las partes de código del sistema y la estructura del sistema en ejecución. Como parte del Modelo de Implementación se encuentran los Diagramas de Componentes y los Diagramas de Despliegue.

2.3.2.1. Diagramas de componentes

El diagrama de componentes es un diagrama que muestra un conjunto de elementos del modelo, tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes software, sean estos componentes de código fuente, librerías, binarios o ejecutables.

2.3.2.2. 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.

Por su importancia en la comprensión del sistema en su totalidad se incluyó una imagen del diagrama de despliegue.

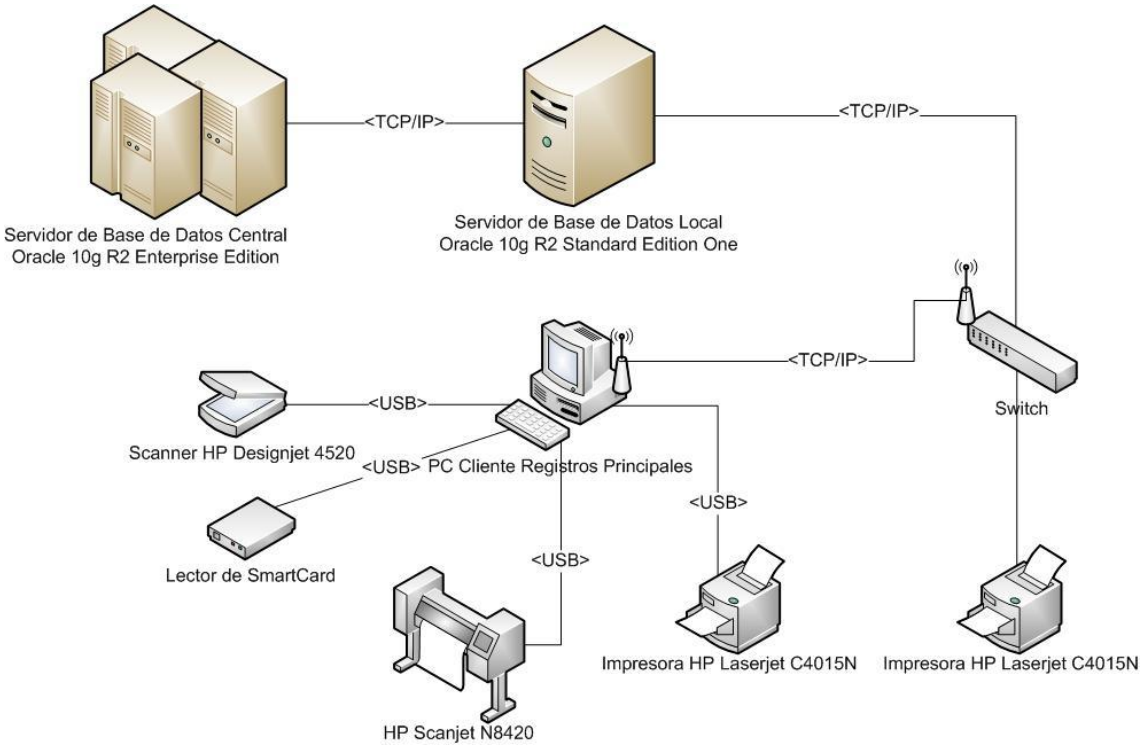

Figura 6. Diagrama de despliegue de la solución informática

Se observa un servidor de base de datos central que está ubicado en el centro de datos y tiene alcance a nivel nacional, el mismo provee la información a los servidores locales de las oficinas, las máquinas clientes se conectan al servidor local mediante red inalámbrica, estas tendrán a su disposición una scanner, un lector de SmartCard, un Plotter y una impresora láser. La impresora podrá estar también conectada directamente a la red mediante un Switch.

El scanner se utilizará para el escaneo de los documentos a inscribir como por ejemplo títulos universitarios. El plotter se utilizará en caso de que el documento a inscribir sea demasiado grande y no se pueda escanear en un escáner común. Las impresoras se utilizarán, por ejemplo, para imprimir la Planilla Única Bancaria, generada durante el proceso de Cálculo. Cada registrador tendrá una tarjeta digital para firmar digitalmente los documentos que se inscriban, por esa razón es que se incluye en el diagrama un lector de SmartCard.

2.4. Resumen de los principales elementos del diseño e implementación

A continuación se presenta un resumen de los principales elementos del diseño y la implementación del sistema, con el objetivo de abarcar de manera global los elementos más importantes de la solución sin entrar en detalles específicos.

- Se confeccionaron modelos conceptuales para cada una de las fases de la inscripción de documentos, con el fin de tener un modelo basado en conceptos que sirva de punto de partida para el trabajo posterior.
- Se diseñaron los gestores implementando interfaces. Esto permite un mayor desacoplamiento entre las capas, ya que una capa puede variar sin afectar a las capas superiores, mientras que siga fiel a las interfaces definidas.
- La instanciación de clases se realiza a través de Spring, que tiene varias opciones para ello:
  - Crear objetos singleton, lo cual te permite tener una sola instancia de una clase sin tener que implementar dicho patrón.
  - Decidir quién será el responsable de la creación de los objetos (uso de fábricas).
- Para el diseño del paquete de acceso a datos se tuvo en cuenta el uso del NHibernate, necesitando solo diseñar los gestores, ya que dicho framework enlaza mediante un fichero de mapeo, cada una de las entidades de dominio con las tablas de la base de datos. Posee métodos propios para salvar y cargar dichos objetos, aunque permite también el uso de procedimientos almacenados que se encuentren en la base de datos.
- El diseño de la navegación de la aplicación se ha hecho en forma de asistente, debido a que el flujo de información en los procesos así lo permite. No obstante, un botón en una pantalla puede llevar a otra que no está en el flujo básico, pero una vez en esa pantalla cuando se culmina la operación se regresa a la pantalla que la originó.
- El framework común se utiliza en el paquete de Presentación para darle un diseño visual estandarizado a la Aplicación y además se utiliza para la gestión de seguridad de acceso basado en roles.
- Los formularios heredan de una plantilla, que contiene lo básico para todas pantallas: los botones Anterior, Siguiente y Cancelar, para la navegación. Los mismos no implementan métodos, ni se accede directamente a ellos. La lógica de presentación de los datos se realiza en otras clases apartes, llamadas acciones. Básicamente los formularios son como cascarones, y al no contener nada, se pueden reutilizar más de una vez.

2.4.1. Componentes personalizados utilizados.

Se reutilizaron componentes desarrollados en la primera fase del proyecto cuando se informatizaron los registros Mercantiles y Públicos. Estos se realizaron con el objetivo de facilitar la implementación recogiendo en componentes las principales

funcionalidades que se hacían repetitivamente en distintas partes del código, ganando en reusabilidad. A continuación se muestra una imagen de los componentes personalizados que se utilizarán para el desarrollo de la aplicación.

Figura 7: Componentes personalizados

2.4.2. Patrones de diseño empleados.

Singleton (solitario): su propósito es asegurar que una clase tenga una sola instancia y proveer un punto global de acceso a ella. Es un patrón bastante simple de implementar pero a la vez potente. En este caso, como la instanciación de clases se hace a través de Spring, no se hizo necesario implementar dicho patrón, aunque sí se utiliza en los gestores de acceso a datos, y de negocio. Esto es posible gracias a que Spring crea los objetos de las clases de tipo singleton por defecto.

Abstract Factory (fábrica abstracta): su propósito es definir una interfaz para la creación de familias de objetos relacionados o dependientes sin tener que especificar la clase concreta. La fábrica abstracta se utiliza para la creación de los objetos entidades. Se potencia el encapsulamiento, puesto que se aísla a los clientes de las implementaciones y se obtiene un incremento de la flexibilidad del diseño.

Proxy (intermediario): su objetivo es proporcionar un representante o delegado que se encargue de controlar el acceso a un objeto, generalmente por motivos de eficiencia o seguridad. En este caso, se utiliza para la creación de objetos costosos por demanda, que son los trámites. Luego de realizar una búsqueda de trámites, no se devuelven objetos de tipo trámite en sí, debido a que son muy grandes y el cliente solo podrá seleccionar uno. En este caso de devuelve un proxy trámite que muestra una mínima parte del objeto trámite que se utiliza en la visualización de los resultados de búsqueda. Una vez que el usuario selecciona el trámite que gestionará, se manda a crear el verdadero objeto trámite.

State (estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Está motivado por aquellas clases en que, según su estado actual, varía su comportamiento ante los diferentes mensajes. Es utilizado para manejar los diferentes estados por los que pasa el trámite, ya sea por cada uno de los flujos dentro de la inscripción de documentos, así como dentro del mismo subproceso.

2.5. Conclusiones del capítulo

Durante el desarrollo de este capítulo se abordaron los aspectos concernientes a la solución propuesta. Se analizó la arquitectura del sistema a desarrollar, que es una arquitectura en capas, se analizaron las funciones de cada capa en particular y las características fundamentales. La arquitectura está basada en la utilización de algunos frameworks, de los cuales se analizaron sus funciones y su contribución al desarrollo. Se realizó también una descripción de los procesos del sistema a grandes rasgos. Como parte de la solución se obtuvieron los modelos de diseño e implementación, los cuales incluyen diagramas de clases, de secuencia, de componentes y de despliegue. En este documento solo se presenta una breve caracterización de los modelos, incluyéndose un resumen de los principales elementos contenidos en ellos, que incluyen los patrones de diseño empleados para lograr una solución más flexible. Se obtuvo además el código fuente de la aplicación en su primera iteración, el cual fue validado mediante pruebas unitarias. Los resultados de la validación se exponen en el capítulo siguiente.

En el Anexo 2 puede observar una imagen de la aplicación en ejecución.

Capítulo 3. Análisis de resultados

3.1. Métricas Aplicadas

3.1.1. Tamaño de clase

La clase es la unidad fundamental del diseño orientado a objetos, por lo que es importante tener una forma de saber si las clases que se han diseñado tienen buena calidad o no. Para ello se miden las clases individuales, las jerarquías de clases y sus colaboraciones.

El tamaño general de una clase se determina con las siguientes medidas:

- El número total de operaciones (de instancia heredada y privada) que están encapsuladas dentro de la clase.
- El número de atributos (de instancia heredada y privada) que están encapsuladas por la clase.

Los valores grandes de TC indican que tal vez una clase tenga demasiada responsabilidad, haciendo difícil su implementación y prueba, mientras que valores pequeños permiten que la clase sea altamente reutilizable.

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

A continuación se presenta el resultado de aplicar la métrica.

Total de clases	TC			Promedio de atributos	Promedio de operaciones
	Pequeño	Mediano	Grande		
144	139	4	1	2,51	4,71

Los resultados obtenidos indican que el diseño de las clases es bueno ya que las clases son reusables, gracias al bajo nivel de responsabilidades y son fáciles de comprobar.

3.1.2. Árbol de Profundidad de Herencia

Esta métrica se aplica a una jerarquía de clases, permitiendo conocer el nivel de complejidad en el momento de predecir el comportamiento de alguna de sus clases y la complejidad del diseño realizado. Esta métrica se define como la longitud máxima

desde el nodo padre hasta la raíz más lejana en el árbol que representa a la jerarquía y el valor obtenido se denomina APH[14].

Se puede determinar que si el valor del APH crece entonces las clases de niveles inferiores heredan muchos métodos y conlleva a una mayor complejidad del diseño pero a su vez implica que muchos de estos métodos puedan ser reutilizados.

Se analizaron todas las jerarquías presentes en la solución resultando que la mayor tiene APH 3, lo que se significa que no tiene mucha complejidad.

3.1.3. Número de descendientes

"Un descendiente es una subclase que se encuentra inmediatamente subordinada a otra en la jerarquía de clases. A medida que crece el número de descendientes (NDD) se incrementa la reutilización pero podría diluirse la abstracción que presenta la clase predecesora si alguno de los descendientes no es un miembro apropiado de la clase padre". (14)

En la solución propuesta las entidades tiene NDD máximo de 4, mientras que los formularios y las acciones que los controlan heredan todos de frmPlantilla y accPlantilla respectivamente, por lo que se obtiene un número elevado de NDD aumentando la reutilización del código, puesto que todos los formularios tienen una estructura y comportamiento común en forma de asistente.

3.1.4. Complejidad ciclomática

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

En Acceso a Datos, Gestión, todos los métodos tienen CC 1 excepto el método BuscarPersonaJuridica(string, string, IDEntidadFederal) : IList<IDPersonaJuridica> que tiene 3.

Dentro de Negocio, Gestión, la CC varía entre 1y 7, predominando los valores más bajos.

[14] Árbol de Profundidad de Herencia

3.2. Resultados obtenidos de las pruebas del NUnit

Se realizaron varias iteraciones de pruebas de caja blanca mediante el NUnit, que luego de corregir los errores obtenidos resultaron ser satisfactorias.

A continuación se muestra las pruebas que se le hicieron a métodos relacionados con los casos de uso Gestionar Persona Natural y Gestionar Persona Jurídica.

Pruebas a la implementación de los casos de uso Gestionar Persona Natural y Gestionar Persona Jurídica

Este es el método de partida de las pruebas de unidad, en él se declaran las inicializaciones de la Base de Datos de Registros Principales con el objetivo de realizas las pruebas posteriores.

```
[SetUp]
public void IniciarContexto()
{
    string conxString =
    "Data Source=rp213p.saren.mij.gov.ve;
     User Id=usuario_213;
     Password=Data Source=rp213p.saren.mij.gov.ve;
     User Id=usuario_213;
     Password=CslNMWU73oQb0Ny1dFos3361tW0=;;";

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

Figura 8. Método de inicio de las pruebas.

Este método se encarga de validar que la búsqueda de la persona sea correcta, para ello se pasa por parámetro un pasaporte real "57584" y comprueba si la persona "BARBARA" es igual al valor esperado de la persona.

```
[Test]
public void ProbandoBuscarPersonaNatural1()
{
    persona = Base.Resolver<IGtrPersonaNatural>();
    var letra = char.MinValue;
    IDPersonaNatural pers= persona.BuscarPersonaNatural(letra,-1, "57584");

    Assert.AreEqual("BARBARA", pers.PrimerNombre);

}
```

Figura 9. Prueba al método de Buscar Persona Natural.

Este método al igual que el anterior se encarga de validar en caso de que el valor esperado sea distinto del valor pasado por parámetro.

```
[Test]
public void ProbandoBuscarPersonaNatural()
{
     persona = Base.Resolver<IGtrPersonaNatural>();
     var letra = char.MinValue;
     IDPersonaNatural pers = persona.BuscarPersonaNatural(letra, -1, "57584");

     Assert.AreNotEqual("BARBAR", pers.PrimerNombre);

}
```

Figura 10. Prueba al método de Buscar Persona Natural con parámetros erróneos.

Este método dado una persona pasada por parámetros se encarga de valida que funcione correctamente comparando que la persona salvada se encuentre en la Base de Datos.

```
[Test]
public void ProbandoSalvarActualizarPersonaJNatural()
{
     persona = Base.Resolver<IGtrPersonaNatural>();
     per = Base.Resolver<IDPersonaNatural>();

     per.PrimerNombre = "Jose";
     per.PrimerApellido = "Zamora";
     per.SegundoNombre = "Gregorio";
     per.SegundoApellido = "Sanchez";
     per.CedulaLetra = 'v';
     per.CedulaNumero = 18051521;
     per.IdPersonaComun = 18102098;

     persona.SalvarPersonaNatural(per);
     per2 = persona.BuscarPersonaNatural(per.CedulaLetra, per.CedulaNumero, per.Pasaporte);

     Assert.AreEqual(18102098, per2.IdPersonaComun);
}
```

Figura 11. Prueba al método de Salvar Actualizar Persona Natural

Este método se encarga de validar que la búsqueda de la persona jurídica sea correcta, para ello se pasa por parámetro tres atributos y comprueba si la persona jurídica es igual al valor esperado.

```
[Test]
public void ProbandoBuscarPersonaJuridica()
{
    persona = Base.Resolver<IGtrPersonaJuridica>();

    IList<IDPersonaJuridica> aux = persona.BuscarPersonaJuridica("BNV", "V1234", 10500);
    Assert.AreNotEqual(10500, aux[0].EstadoDomicilio);
}
```

Figura 12. Prueba al método de Buscar Persona Jurídica.

Esta funcionalidad dado una persona jurídica pasada por parámetro, se encarga de validar que funcione correctamente el método Salvar Actualizar Persona Jurídica, comparando que la persona salvada se encuentre en la base de datos.

```
[Test]
public void ProbandoSalvarActualizarPersonaJuridica()
{
    per1 = Base.Resolver<IDPersonaJuridica>();
    per2 = Base.Resolver<IDPersonaJuridica>();

    per1.Denominacion = "BNV";
    per1.IdPersonaJuridica = 18015027;
    per1.Objeto = "Buscado";
    per1.Rif = "V1234";

    persona.SalvarActualizarPersonaJuridica(per1);
    IList<IDPersonaJuridica> aux = persona.BuscarPersonaJuridica("BNV", "V1234", 10500);
    for (int i = 0; i >= aux.Count; i++)
    {
        if (aux[i].IdPersonaJuridica == per1.IdPersonaJuridica)
        {
            per2 = aux[i];
        }
    }

    Assert.AreEqual(18015027, per2.IdPersonaJuridica);
}
```

Figura 13. Prueba al método de Salvar Actualizar Persona Jurídica.

Este método verifica que se genere el número del trámite.

```
[Test]
public void ProbandoGenerarNumeroTramite()
{
    obj = Base.Resolver<IGtrTramite>();
    string num = obj.GenerarNumeroTramite("1");

    Assert.AreNotEqual("", num);
}
```

Figura 14. Prueba al método Generar Número Trámite.

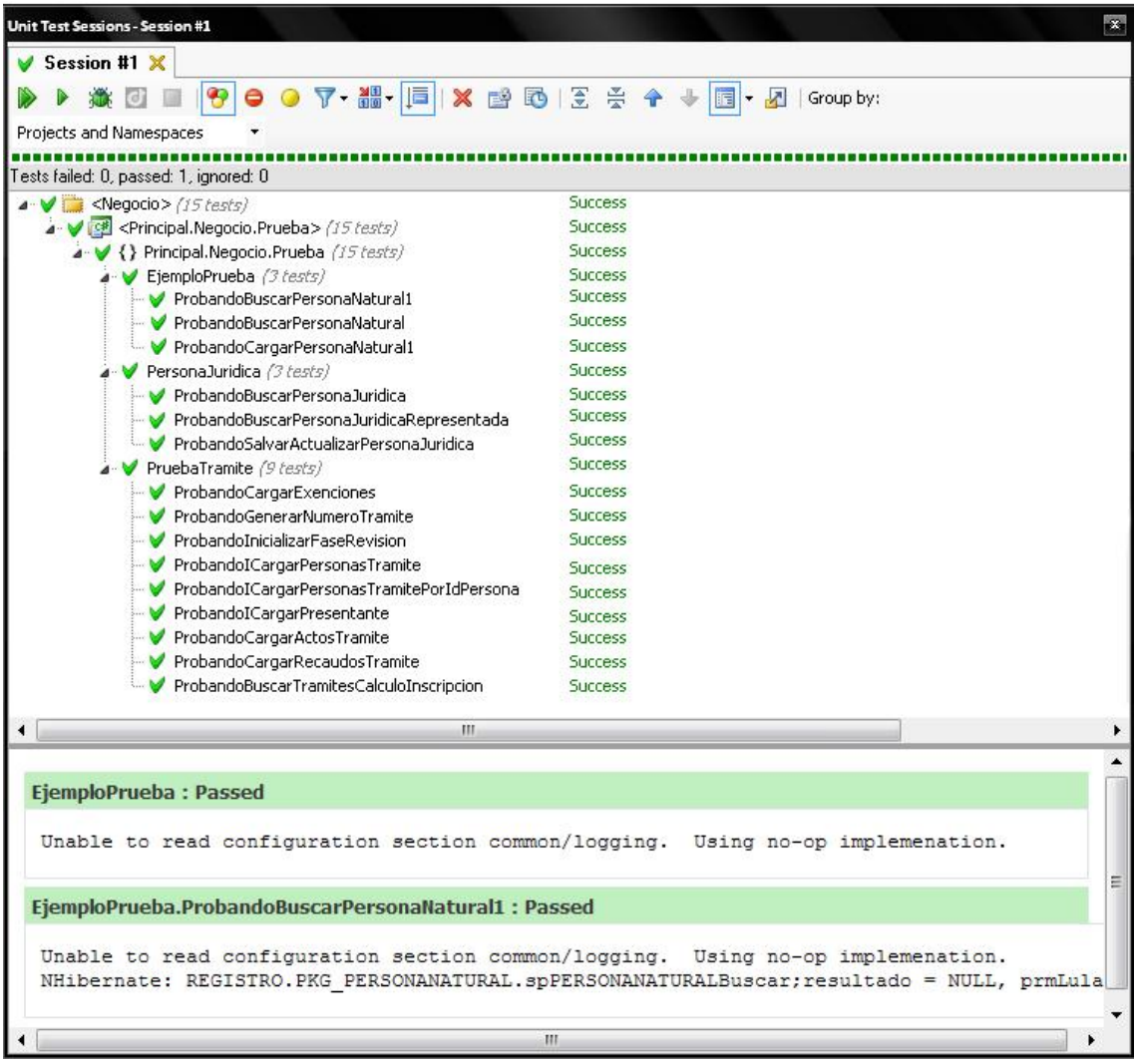

Figura 15. Imagen del resultado de las pruebas con el NUnit

3.3. Conclusiones del capítulo

A lo largo del capítulo se abordó todo lo referente a la validación del sistema propuesto. La validación se llevó a cabo a través de la medición de algunos aspectos del software como el tamaño de clase, la profundidad de la herencia, y la complejidad ciclomática. Los valores numéricos obtenidos mediante la aplicación de las distintas métricas proveen un valor cuantitativo acerca de la calidad del diseño del software. Además, se plasmó el resultado de las pruebas unitarias realizadas al sistema. Después de varias iteraciones de pruebas, tanto de caja negra como las realizadas con el NUnit, donde se detectaron errores y se corrigieron hasta que no se detectaron otros errores, se obtuvo un sistema de alta calidad.

Conclusiones

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

- Mediante el estudio realizado sobre los procesos que tienen lugar en los Registros Principales, se logró dominar los aspectos más relevantes del Proceso de Inscripción de Documentos para la automatización del mismo.
- Se construyó el Modelo de Diseño y Modelo de Implementación del Sistema, desarrollándose los Diagramas de Interacción, Diagramas de Clases, el Modelo de Datos, los Diagramas de Componentes y de Despliegue.
- Se implementó un sistema informático para la gestión de los procesos registrales que se apoya en lo estipulado en la Ley para su correcto funcionamiento y que además:
  - Estandariza el Proceso Registral en todas las Oficinas de Registros Principales dando mejor cumplimiento al Principio de Legalidad planteado en la Ley de Registro Público del Notariado vigente.
  - Logra que la información obtenida de las oficinas registrales sea recibida en el MPPRIJ de forma más confiable y rápida, de manera que se pueda ofrecer un mejor servicio, organización y administración de los registros.
  - Logra un incremento en la Seguridad Jurídica con el uso de la Firma Digital a los documentos generados en los registros, fortaleciéndose la Fe Pública Registral con la creación de los archivos digitales en las oficinas.
- Se realizó un análisis y validación de la calidad tanto del diseño como de la implementación del sistema a través de la aplicación de diferentes métricas de calidad y la realización de varios tipos de pruebas que ayudaron a corroborar el estado de los artefactos obtenidos durante el desarrollo.
- Como resultado general del trabajo se obtuvo el Diseño e Implementación de la aplicación informática para la Inscripción de Documentos en los Registros Principales de la República Bolivariana de Venezuela.

Recomendaciones

Con la conclusión del presenta trabajo se recomienda:

- Migrar la solución propuesta hacia alguna plataforma de software libre según lo dispuesto en el Decreto 3390 del 23 de diciembre del 2004.
- Continuar con el diseño y la implementación de los demás procesos que tienen lugar en los Registros Principales.
- Documentar el Framework Común para que sea posible su utilización por otros desarrolladores y proyectos.

Bibliografía

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

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

3. Real Academia Española. [Online] [Cited: 02 24, 2010.] http://www.rae.es/rae.html.

4. **Budd, Timothy.** *An introduction to Object-Oriented Programming.* s.l. : Addison Wesley, 1991. ISBN: 0-201-54709-0.

5. **LeMaster, Ron and Leberknight, Dave.** *Object-Oriented Programming and Design.* s.l. : University of Colorado, 2001.

6. **Kiczales, C., et al.** *Aspect-Oriented Programming.* s.l. : Xerox Palo Alto Research Center, 1997.

7. **Christopher Alexander, et al.** *A Pattern Language.* New York : Oxford University Press, 1977.

8. **Gamma, Erich, et al.** *Design Pattern: elements of reusable object-oriented software.* s.l. : Addison Wesley.

9. **http://whatis.techtarget.com/.** Computer Glossary, Computer Terms. *Computer Glossary, Computer Terms.* [Online] [Cited: 03 03, 2010.] http://whatis.techtarget.com/.

10. **https://www.hibernate.org/.** Hibernate. [Online] [Cited: 03 03, 2010.] https://www.hibernate.org/.

11. **Kendall, Kenneth and Kendall, Julie.** *Análisis y Diseño de Sistemas de Información.* s.l. : Pearson Educación, 1997. ISBN: 978-970-26-0577-5.

12. **Flower, Martin, et al.** *Patterns of Enterprise Application Architecture.* s.l. : Addison Wesley, November 05, 2002. 0-321-12742-0.

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

14. **Pressman, Roger S.** *La Ingeniría del Software, un enfoque práctico.* 2005.

15. **Richter, Jeffrey.** *Applied Microsoft .NET Framework.* s.l. : Microsoft Press, 2002.

16. .Net Framework Developer Center. [Online] [Cited: 02 10, 2010.] http://msdn.microsoft.com/en-us/netframework/default.aspx.

17. Consejo Superior de Administración Electrónica. *MÉTRICA Versión 3.* [Online] [Cited: 02 16, 2010.] http://www.csi.map.es/csi/metrica3/.

18. *Cultura Informática.* **Instituto Nacional de Estadística e Informática.**

19. Enterprice Architect. [Online] [Cited: 02 10, 2010.] http://www.sparxsystems.com.ar/.

20. IBM. [Online] IBM. [Cited: 02 19, 2010.] http://www.ibm.com/us/en/.

21. Microsoft Visual Studio .Net 2003. [Online] [Cited: 02 10, 2010.] http://support.microsoft.com/ph/3040.

22. Mono. [Online] [Cited: 03 02, 2010.] http://www.mono-project.com/Main_Page.

23. The OPEN website. [Online] [Cited: 02 16, 2010.] http://www.open.org.au/.

24. **Visual Paradigm International Company.** Visual Paradigm. [Online] [Cited: 02 18, 2010.] http://www.visual-paradigm.com/.

25. **Becker, Kent.** *Extreme Programming Explained, embrace change.* 1999.

26. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación S.A., 2000.

27. **Prosise, Jeff.** *Programming Microsoft.NET.* s.l. : Microsoft Press, 2002. 0-7356-1376-1.

28. **Reina Quintero, Antonia.** *Visión General de la Programación Orientada a Aspectos.* Sevilla, Espeña : s.n.

29. **Shannon, Bill, et al.** *Java™ 2 Platform, Enterprise Edition: Platform and Component Specifications.* s.l. : Addison Wesley, May 26, 2000. 0-201-70456-0.

30. Java.sun.com. *The Source for Java Developers.* [Online] http://java.sun.com/docs/books/tutorial/getStarted/index.html.

Glosario de Términos

A

Asiento Registral

Un Documento luego de ser inscrito es archivado o protocolizado ubicándose en Protocolo, Tomo, Folio, lo cual constituye un Asiento Registral.

Acto

Son las operaciones que vienen definidas en el documento a inscribir. Ejemplos de actos se pueden mencionar algunas de los más importantes como:

- Registro de Título Universitario.
- Registro de Sentencia.
- Registros de Actas de Asociaciones Civiles, Fundaciones y Cooperativas.

B

Bytecode

El bytecode es un código intermedio más abstracto que el código máquina.

E

Exención

Figura establecida en las Leyes de la República mediante la cual el estado otorga ventaja a los beneficiados en el no pago total o parcial de impuestos, tasas, o contribuciones.

F

Foliatura

Folio inicial y final desde y hasta donde se extiende la inscripción de un documento, comprobante o cualquier otro instrumento que debe asentarse en un tomo.

Folio

Página, hoja de un documento o libro donde este se asienta. Se numeran consecutivamente, lo cual sirve para referenciar dónde exactamente, en un tomo, está inscrito un documento.

Framework

Conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software. Un framework ofrece una guía arquitectónica partiendo del diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones.

Funcionario

Personal que labora en el Registro Principal, los mismos pueden cumplir diferentes roles, tales como:

- Funcionario de Cálculo.
- Funcionario de Presentación.
- Funcionario de Revisión.
- Funcionario de Archivo.
- Funcionario de Otorgamiento.
- Funcionario de Procesamiento.

H

Habilitar

Pagar por realizar el otorgamiento en fecha anticipada a los 10 días que por defecto han de transcurrir entre la presentación y el otorgamiento.

L

Libro de Presentaciones

Libro donde se recogen consecutivamente a diario las notas de presentación.

O

Oficio

Se refiere al documento oficial que se envía de una institución a otra.

Otorgante

Persona involucrada en las operaciones que se realizan en la inscripción de un documento.

P

Planilla Única Bancaria

Planilla emitida durante la fase de Cálculo donde se asientan los conceptos por los cuales se debe pagar al servicio autónomo y el monto a pagar por cada uno, según el cálculo realizado al documento.

Presentante

Persona que presenta el documento.

Protocolo

Clasificación que se hace de los tomos según el tipo de operaciones que están inscritas en él.

R

Recaudos

Documentos, comprobantes, avales, certificaciones y constancias que deben acompañar a los documentos a la hora de presentarlos, para conferirle valor legal al proceso y respaldar las operaciones contenidas en el mismo.

RIF

El Rif (Registro de Información Fiscal) es una identificación tributaria de todas las entidades jurídicas existentes.

Anexos

Anexo 1

Criterio de clasificación		Propósito		
		Creacional	**Estructural**	**Comportamiento**
Ámbito	**Clase**	Método de fábrica	Adaptador	Intérprete Método plantilla
	Objeto	Fábrica abstracta Constructor Prototipo Solitario	Adaptador Puente Composición Decorador Fachada	Cadena de responsabilidades Acción Iterador Mediador Recuerdo Peso ligero Observador Estado Estrategia Visitante

Anexo 2