UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Análisis y Diseño de un Sistema de compilación genérica para desplegar sistemas informáticos en el ámbito Jurídico en la República Bolivariana de Venezuela

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora:

Daylenis Ortíz Franco

Tutores:

Ms C. Yarina Amoroso Fernández

Ing. Alien Rodríguez López

Co-Tutor

Ing. Jorge Yuniel Jorrin Perdomo

CIUDAD DE LA HABANA
Junio, 2009

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a___ días del mes de _____ del año 2009.

_____ _____

Autora: Daylenis Ortíz Franco Tutora: Ms C. Yarina Amoroso Fernández

_____ _____

Tutor: Ing. Alien Rodríguez López Co-Tutor: Ing. Jorge Yuniel Jorrin Perdomo

DEDICATORIA

A la mujer más fuerte y tierna que en la vida conocí.
Quien fue un ejemplo para mí durante toda mi niñez, mi adolescencia y parte de mi juventud.
La dicha de disfrutar juntas este momento que tanto anheló no nos acompañó, pero hoy está aquí en cada paso, en cada decisión que tome, tan fuerte como antes, guiándome como lo hizo siempre. Por ella fue posible que llegara hasta aquí. Aún sin estar físicamente, fue quien me impulsó a seguir.
Cuando las fuerzas me faltaron fue su recuerdo el que me salvó.

A mi madre, la mejor de mis guías, mi estrella y mi paradigma
Gracias

AGRADECIMIENTOS

Agradecerle en primer lugar a mi familia. A mi padre que ha sido siempre mi ejemplo.

A mis abuelos: Mima, mami Enma, Papi Santo y mi abuelito Hernán que ya no está conmigo y que fue un eslabón importante en mi formación.

A mis tíos, Merilainy, Lice y Joel, siempre preocupados por mis resultados.

A mi otra abuela Nilda y mi otro papá el Furo, mi familia de la Habana.

A la tita Yvonne y al tito Cheo, mis padres en la escuela.

Agradecerle de forma especial a mi purri (mi novio), quien estuvo en los buenos y malos momentos de mi tesis apoyándome y animándome siempre.

A mis primas, las locas de Bayamo, que han sido mis hermanas del alma.

A mi hermana por quien me esfuerzo para darle un ejemplo a seguir cada día.

A la loca de mi madrastra, que me ayudó tanto con mi mamá y me apoyó en la universidad.

A mis tutores: Yarina y Alien, por ayudarme a comprender el difícil mundo de los documentos jurídicos

Un agradecimiento especial a mi cotutor, el moreno conversador (Jorrín) por asumir la tarea de tutorar la tesis como suya.

A mis hermanas del apartamento, el grupo de psiquiátricas: Aida, Marilidia, Gleydis, Yelena, Ivis que aunque no nos vamos a ver más siempre las recordaré.

A las profes del proyecto que siempre estuvieron dispuestas a ayudar en cualquier momento.

A Yoandris por su ayuda con aquel diseño teórico.

A todos los que hicieron posible que de una forma u otra se cumpliera el sueño de graduarme de Ingeniera en Ciencias Informáticas.

GRACIAS

RESUMEN

Para el despliegue de cualquier sistema informático se necesita de la Carga Inicial (CI), un proceso que recopila los datos necesarios para que la solución funcione correctamente en un ambiente real. La CI, no sólo permite los sistemas de gestión automaticen la información en las entidades, disminuyendo el trabajo manual, aligerando los procesos, aumentando la excelencia de los servicios que puedan prestar, sino que también disminuye el tiempo destinado para el despliegue de las soluciones. En el ámbito jurídico este proceso está relacionado con la recopilación de los datos documentales jurídicos y las características de las instituciones donde se instalará el software. Estos datos documentales están referidos a los tipos documentales y documentos jurídicos que manejan en la oficina y que son de consulta obligatoria a la hora de realizar cualquier trámite legal.

Este trabajo está enfocado a la realización del Análisis y Diseño de un sistema de recopilación de CI para desplegar sistemas informáticos en el ámbito jurídico en la República Bolivariana Venezuela. En el mismo se desarrollan los artefactos de esta disciplina con el objetivo de lograr una eficiente implementación de un sistema que permita capturar la información inicial referente a los datos documentales jurídicos y a las oficinas donde se desplegará la solución, permitiendo además que se introduzcan documentos de forma genérica, dándole gran flexibilidad al sistema para que pueda ser instalado en cualquier institución jurídica.

Para desarrollar el análisis y el diseño se utilizó: UML como lenguaje de modelado y Visual Paradigm for UML Enterprise Edition como herramienta CASE, aplicando la metodología de desarrollo Rational Unified Process (RUP). El uso de estas herramientas permitió un desarrollo sencillo, comprensible y robusto de los diferentes artefactos generados, entre los que se pueden mencionar, los diagramas de clases tanto del análisis como del diseño, diagramas de interacción, subsistemas de diseño, diagrama de despliegue, diagrama de clases persistentes, entre otros propuestos por la metodología.

Palabras claves: sistemas informáticos, ámbito jurídico, instituciones jurídicas, Carga Inicial, proceso, despliegue, datos documentales jurídicos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

El despliegue de una solución informática, al igual que el resto de las disciplinas desarrolladas en el proceso de creación de un software, depende de un conjunto de actividades que garantizan la rapidez en la instalación del sistema de manera general, entre éstas podemos citar la capacitación de los usuarios que trabajarán con el software, la configuración del equipamiento que se utilizará, los servicios de entrega y la organización del soporte que se dará al mismo.

En el ámbito jurídico no basta con que se hayan desarrollado estas actividades, es necesario además que se cuente con las características de las entidades donde la solución será desplegada y los documentos jurídicos que manejan, que fungen como la CI para el despliegue de sistemas informáticos en este ámbito.

En la República Bolivariana de Venezuela el ente encargado de esgrimir la información relacionada con las instituciones jurídicas es el Ministerio del Poder Popular para las Relaciones Interiores y de Justicia (MPPRIJ), que tiene entre otras misiones, promover la seguridad jurídica de la población a través de los órganos encargados de la administración de justicia, registros y notarías, documentos, identificación y derechos humanos de los ciudadanos.

El MPPRIJ cuenta con tres soluciones para la CI, pero sólo para los registros públicos y mercantiles, de modo que no tiene control sobre los datos de ubicación oficial, teléfono, categoría, libros en archivo, Estado, Municipio, parroquias, documentos jurídicos como exenciones, denominaciones, disposiciones legales y prohibiciones, del resto de las instituciones jurídicas.

Las soluciones están desarrolladas en Microsoft Access, lo que limita sus funcionalidades debido a que esta herramienta está diseñada para proyectos pequeños y su funcionamiento puede ser inestable cuando maneja grandes volúmenes de información.

La obtención de la información resulta engorrosa, debido a que las aplicaciones se envían desde el Centro de Datos (CD) a las oficinas, se carga la información y luego se remiten, por correo postal o por correo electrónico, al CD nuevamente para ser procesadas. Una vez que llegan, se asigna a un especialista para descargar las aplicaciones y revisar de forma manual los datos que contienen; luego se entregan a otro especialista para que las introduzca en otra aplicación que también hace revisiones y posteriormente se guardan en la BD. El proceso de revisión en dos lugares diferentes, en ocasiones es

lento de modo no siempre se obtienen reportes diarios de los documentos enviados por las oficinas, además de lo incómodo que resulta la revisión manual de un gran número de documentos.

Resulta difícil para los usuarios en las instituciones introducir los datos en tres aplicaciones diferentes que además no se instalan, por tal motivo si se pierde alguna de ellas, no hay forma de introducir la información si no se solicita nuevamente la aplicación al CD.

Las aplicaciones no capturan como dato obligatorio la dirección de correo electrónico del encargado de de la oficina o del responsable de la CI en ésta, esto trae como consecuencia que una vez que llegue la información por correo electrónico, el especialista encargado de hacer la revisión manual debe guardar esa información en el mismo correo o en un bloc de notas para enviar posteriormente las notificaciones de documentos con errores. En caso de la información sea enviada por correo postal no se pueden enviar de forma sencilla dichas notificaciones a las instituciones.

Atendiendo a la situación anteriormente descrita el ***Problema Científico*** de la investigación se formula de la siguiente forma:

¿Cómo desarrollar un sistema de compilación genérica, que capture toda la información referente a los documentos jurídicos y las características de las instituciones jurídicas, en aras de facilitar el despliegue de sistemas informáticos en este ámbito?

Como **Objeto de Estudio** se ha determinado:

El Proceso de Desarrollo de Software.

Para dar solución a esta problemática se determinó el siguiente ***Objetivo General***:

Realizar el análisis y diseño de un sistema de compilación genérica para desplegar sistemas informáticos en el ámbito jurídico en la República Bolivariana de Venezuela.

Del Objetivo General se desglosan los siguientes ***Objetivos específicos:***

- ✓ Estudiar los conceptos vinculados al entorno jurídico y las herramientas relacionadas con aplicaciones informáticas en ese ámbito.
- ✓ Realizar el análisis y el diseño de un sistema que garantice la recopilación de información en el ámbito jurídico.

- ✓ Evaluar el diseño del mencionado sistema.

A partir del objeto de estudio planteado se toma como ***Campo de acción*:**

Análisis y Diseño de sistemas informáticos vinculados a la informática Jurídica.

La respuesta anticipada al problema se formula en la siguiente ***Hipótesis:***

Si se realiza el análisis y diseño de un sistema de compilación genérica para desplegar sistemas informáticos en el ámbito jurídico en la República Bolivariana de Venezuela, se contribuye a desarrollar un sistema de compilación genérica, que capture toda la información referente a los documentos jurídicos y las características de las instituciones jurídicas, en aras de facilitar el despliegue de sistemas informáticos en este ámbito.

Tareas a desarrollar:

- ✓ Estudio de los conceptos vinculados a la Informática Jurídica.
- ✓ Estudio de organizaciones vinculadas a la Informática Jurídicas en la región de América Latina.
- ✓ Análisis de la convergencia del sistema al desarrollo del Gobierno Electrónico en la República Bolivariana de Venezuela.
- ✓ Estudio de la evolución del diseño de soluciones informáticas y de sistemas informáticos de información jurídica.
- ✓ Determinación de las herramientas y metodología a emplear para desarrollar el análisis y el diseño de de la solución planteada.
- ✓ Realización del análisis de un sistema para la recopilación de información jurídica.
- ✓ Realización del diseño de un sistema para la recopilación de información jurídica.
- ✓ Aplicación de métricas de evaluación al diseño de la solución.

Métodos Científicos

Métodos Teóricos

Método histórico – lógico.

El método histórico estudia la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia. El método lógico investiga las leyes generales del funcionamiento y desarrollo de los fenómenos.

Este último para poder descubrir las leyes fundamentales de los fenómenos, se basa en los datos que le proporciona el método histórico, de manera que no constituya un simple razonamiento especulativo. De igual forma el método lógico descubre las leyes y la lógica objetiva del desarrollo histórico del fenómeno y no se limita a la simple descripción de los hechos.

Este método sirvió para estudiar todo el estado del arte de los fenómenos relacionado no solo con el análisis y el diseño de una aplicación como resultado final, sino que permitió el desplazamiento a otras esferas vinculadas con el tema que contribuyen a comprender mejor el fenómeno, dígase la información referente a la Informática Jurídica, Gobierno Electrónico, Proceso de Desarrollo de Software, así como las herramientas de modelado.

Método Analítico – Sintético:

Este método permite analizar y estudiar los factores de un todo en su relativa independencia, dividiéndolo en partes más pequeñas con sus múltiples relaciones y componentes de forma que el fenómeno sea más comprensible para el investigador. Una vez estudiadas cada una de las partes se integran a través de la síntesis que permite se descubran las relaciones existentes entre un factor y otro así como las características generales entre los elementos de la realidad.

El método permitió estudiar individualmente los lenguajes de modelado, las herramientas cases y las metodologías de desarrollo de software para obtener las mejores herramientas para modelar el sistema.

Método Inductivo – deductivo:

Este método refleja la lógica objetiva de los fenómenos y procesos de la realidad en una forma de razonamiento, mediante la cual se pasa de un conocimiento general a otro de menor nivel de generalidad, esto está expresado en el estudio de la Informática Jurídica en la actualidad como nivel de mayor generalidad y el estudio de la Informática Jurídica en América Latina y en Venezuela como niveles más específicos.

Métodos Empíricos

Observación:

Este método permite en síntesis obtener el registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los acontecimientos pertinentes de acuerdo con algún

esquema previsto. Para esto se observó cómo se recopilan los datos actualmente para desplegar sistemas informáticos en entornos jurídicos en Venezuela.

Para una mejor comprensión del trabajo se estructuró en 3 capítulos.

Capítulo 1: Fundamentación teórica. En este capítulo se hace un estudio del estado del arte de la Informática Jurídica en América Latina, el Gobierno Electrónico en Venezuela. Además se abordan los temas relacionados con el Proceso de Desarrollo de Software haciendo énfasis en el flujo de trabajo de Análisis y Diseño de sistemas informáticos y las herramientas y metodologías que se emplearán.

Capítulo 2: Se describe el análisis y el diseño del sistema para la recopilación de datos en el ámbito Jurídico en Venezuela.

Capítulo 3: Se aplican algunas métricas para evaluar el diseño desarrollado y se analizan los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hará referencia a algunos conceptos relacionados con los términos Informática Jurídica y Gobierno Electrónico, además del comportamiento de éstos en el área de América Latina. Igualmente se hará alusión a la evolución del diseño de aplicaciones informáticas. Se analizarán algunos ejemplos de soluciones informáticas en entornos jurídicos, así como algunos sistemas gestores de Base de Datos. Por último se expondrán algunas metodologías de desarrollo de software, lenguajes de modelado y herramientas CASE conocidos para finalmente determinar las más apropiadas para el análisis y el diseño de la solución.

1.2 Informática Jurídica

La relación derecho e informática tiene dos líneas de investigación: los aspectos normativos del uso de la informática desarrollados bajo el derecho de la informática, y la aplicación de esta última en el tratamiento de la información jurídica conocida como Informática Jurídica (IJ). Para el desarrollo de la IJ es necesario considerar ciertos elementos de origen como son la aplicación de la lógica del derecho, análisis del discurso jurídico, la teoría de los sistemas, aplicación de la teoría de la información entre otros; tales elementos constituyen la base fundamental para cumplimentar el objeto mismo de la IJ. (1)

Existen varias definiciones del término IJ, Julio Téllez[1] la define como la técnica interdisciplinaria que tiene por objeto el estudio e investigación de conocimientos de la informática general, aplicables a la recuperación de información jurídica, así como la elaboración y aprovechamiento de los instrumentos de análisis y tratamiento de la información jurídica necesarios para lograr dicha recuperación. (2)

La IJ es el "Tratamiento automatizado de las fuentes de conocimiento jurídico (sistemas de documentación legislativa, jurisprudencial y doctrinal), de las fuentes de producción jurídica y su organización (funcionamiento de organismos legislativos y judiciales) y de las decisiones judiciales (informática jurídica decisional)".[2]

[1] Doctor en informática jurídica y derecho de la informática por la Universidad de Montpellier, Francia. Asesor de la Organización Mundial de Derecho Informático, con sede en Venezuela.

[2] Antonio Enrique Pérez Luño: Catedrático de Filosofía del Derecho y director del Departamento de esa disciplina en la Universidad de Sevilla.

La configuración de cada subsistema está determinada por el género de información que debe procesar y su ámbito de aplicación. La IJ puede estar referida a las fuentes del derecho, a los procesos administrativos de las fuentes de producción jurídica o los datos requeridos para las decisiones y dictámenes. A partir de las esferas donde está representada se puede clasificar en Informática Jurídica Documental (IJD), Informática Jurídica de Gestión (IJG), e Informática Jurídica Meta documental o Decisional (IJMD).

IJD: Procesamiento automático de documentos jurídicos, proveniente de cualquiera de las fuentes formales del derecho: Legislativa, jurisprudencial (producción de los órganos jurisdiccionales, comprendidos los individuales y los colectivos) y doctrinaria (conceptos de los expertos en derecho).

Ubica a los sistemas de tratamiento documental de creación, gestión y recuperación de información en bancos contenedores de datos rigurosamente jurídicos tales como leyes, doctrina, jurisprudencia u otros de interés jurídico.

IJG: Denominada también ofimática o burótica, se refiere a la automatización de procedimientos en las oficinas de los operadores jurídicos. Además de procesar información administrativa, provee herramientas auxiliares para las labores que se desarrollan en tales despachos. Se utiliza para realizar actos jurídicos como certificadores, atribuciones de juez competente, sentencias.

IJMD: Comprende el campo de la inteligencia artificial (IA), concretamente de los sistemas expertos jurídicos (SEJ) que reproducen la actividad del jurista, como auxiliar en la adopción de decisiones para problemas concretos.

En nuestra época, existe un flujo incesante e inexorable de normas. De ahí que se pueda considerar que las bases de datos jurídicos y los sistemas de información legal, son esfuerzos por reunir las normas jurídicas a fin de facilitar la labor de consulta a quienes deben intervenir en su elaboración, estudio o aplicación y actualmente, también a quien se le aplican dichas normas. (3) (4)

1.2.1 Informática Jurídica en América Latina

En la actualidad en América Latina existen organizaciones relacionadas con el campo de la IJ, ejemplo de ello es la Federación Iberoamericana de Asociaciones de Derecho e Informática (FIADI) una de las Organizaciones no Gubernamentales (ONG) pioneras en la región en cuanto a promoción, estudio y desarrollo de la IJ y el Derecho Informático. Casi coetáneamente nace el Instituto Latinoamericano de Alta

Tecnología, Informática y Derecho (ILATID), organización que desde 1984 reúne a los países de América Latina y el Caribe a profesionales del Derecho que cooperan en proyectos de investigación y desarrollo conexos con aspectos legales de las Tecnologías de la Información y las Comunicaciones (TIC).

Asimismo Alfa-Redi, es una de las más activas entidades de investigación y promoción de temas de políticas y regulación de Sociedad de la información en América Latina y el Caribe. Es una organización de la Sociedad Civil, dedicada a la investigación, discusión, formulación de propuestas y acción en temas de Políticas de Sociedad de la Información, además de la formación y capacitación en estas temáticas.

1.2.2 Sistemas de información jurídica

Una de las experiencias más brillantes de la aplicación de la IJD a nivel europeo, la constituye, el sistema CELEX (Communitatis Europea Lex), creado y desarrollado por la Comisión de las Comunidades como banco de datos legislativo, jurisprudencial y documental del derecho comunitario.

El CELEX es un sistema de documentación automatizada del derecho comunitario, que permite conocer, no sólo las normas, sino la jurisprudencia, actos internos, trabajos preparatorios y actos parlamentarios, cuestiones parlamentarias, dictámenes e, Incluso, la doctrina jurídica. Es un instrumento que se adapta a dos de los aspectos fundamentales del derecho europeo: el gran número y la dispersión de las fuentes de derecho, y el alto nivel de atipicidad y falta de Institucionalidad en el desarrollo del ordenamiento de las Comunidades, que avanzan hacia la consecución de una sociedad integrada, apoyándose en manifestaciones jurídicas de diversa naturaleza.

Por otro lado INDILEX es otro de los sistemas desarrollados a partir de la experiencia en la IJD. Este ha permitido constituir una base de referencias legislativas o de índices de legislación. Funciona como una serie de referencias integradas en el ordenamiento jurídico estatal de las disposiciones contenidas en el periódico. Constituye un producto directo de la mecanización de los índices, y un pequeño banco de datos de referencia legislativa. Es una de las primeras aplicaciones de IJD, en el ámbito de la legislación con la realización práctica de datos bibliográficos de carácter general, que incorpora un elevado número de referencias legales, desarrolladas en España. (5)

El sistema de Servicio Autónomo de Registros y Notarias (SAREN) SAREN es un sistema de IJD e IJG, desarrollado, en su primera fase, para automatizar los procesos en los Registros públicos y mercantiles en Venezuela, sirve de tratamiento electrónico a las actividades que se desarrollan en estas oficinas e incluye operaciones como el procesamiento de los datos, análisis, búsqueda y recuperación de información.

Un ejemplo de sistemas dedicados a la IJMD es sistema artificial de experto legal *SHYSTER*, desarrollado en Estados Unidos, que consiste en un sistema de casos legalmente basados. Proporciona consejos a través de la relación de casos similares. Se diseñó para analizar jurisprudencia y elaborar argumentos legales en procesos judiciales relativos al derecho de secreto comercial, combina jurisprudencia y reglas positivas. (6)

1.3 Gobierno Electrónico

El Gobierno Electrónico o E-Government, se ha caracterizado como una nueva forma de relacionamiento entre el Gobierno y la sociedad (7). La implantación del Gobierno Electrónico permite el reconocimiento por parte de los Estados del derecho de los ciudadanos a relacionarse electrónicamente con sus Gobiernos y Administraciones Públicas, lo que supone que las Administraciones estén interrelacionadas entre sí a fin de simplificar los procedimientos. Las leyes de acceso a la información pública establecidas en algunos países de la región de Iberoamérica apuntan en esa dirección.

Este reconocimiento debe ser tan amplio como lo permita la naturaleza del trámite y pretensión de que se trate. Los ciudadanos podrán vincularse electrónicamente con los Gobiernos y las Administraciones Públicas, entre otros, a efectos tales como los siguientes:

- ✓ Dirigir por vía electrónica todo tipo de escritos, recursos, reclamaciones y quejas a los Gobiernos y las Administraciones Públicas, quedando éstos igualmente obligados a responder o resolver como si dichos escritos, reclamaciones y quejas se hubieran realizado por medios tradicionales.
- ✓ Realizar por medios electrónicos todo tipo de pagos, presentar y liquidar impuestos y cualquier otra clase de obligaciones.
- ✓ Recibir por medios electrónicos notificaciones cuando tal medio sea aceptado por el ciudadano o si el ciudadano así lo solicita.
- ✓ Acceder por medios electrónicos a la información administrativa general con igual grado de fiabilidad que la que es objeto de anuncio en diarios o boletines oficiales o la que se publica en anuncios oficiales por cualquier medio.
- ✓ Acceder a los expedientes para conocer el estado en que se encuentra la tramitación de los mismos.
- ✓ Acceder por medios electrónicos a información pública de alto valor agregado que sirva a aumentar la competitividad de los países, lo que supone garantizar estándares consensuados entre los Estados

respecto al modo en que esa información debe ser procesada y difundida con la ayuda de las nuevas tecnologías disponibles.

- ✓ Utilizar y presentar ante el Gobierno o las Administraciones Públicas las resoluciones administrativas en soporte electrónico, así como los documentos administrativos electrónicos en las mismas condiciones que si fueran documentos en papel, así como poder remitirlas por medios electrónicos a la Administración de que se trate.
- ✓ Evitar la presentación reiterada ante la Administración de documentos que ya obren en poder de la misma o de otra, especialmente si son electrónicos, todo ello en el supuesto de que el ciudadano de su consentimiento para la comunicación de tales documentos entre Administraciones y entre distintas dependencias de la misma Administración, lo que supone acciones de Interoperabilidad y Simplificación Registral. (8)

No se puede hablar del Gobierno electrónico sin hacer referencia a la administración electrónica como uno sus componentes.

1.3.1 Administración electrónica

La Administración electrónica (AE) -también llamada ciber Administración", "Administración virtual", "Administración digital", "Administración online" o "tele Administración"- comprende y designa todos aquellos mecanismos e infraestructuras informáticas y telemáticas que permiten la prestación de servicios, tanto a los ciudadanos como a las empresas. (7)

En la construcción de este nuevo modelo organizacional se asiste a un momento decisivo de reforma estatal, en la cual se fincan grandes expectativas: desde la Administración, por advertir el inmenso potencial de las tecnologías de la información y las comunicaciones para mejorar la prestación de todo tipo de servicios y la propia gestión administrativa, y desde la ciudadanía, por la posibilidad de acceder a más y mejor información, controlar a las autoridades, obtener trato igualitario y aumentar la eficiencia en el uso del tiempo y de más recursos.

Ventajas de la Administración Electrónica

- ✓ Desde el punto de vista jurídico, el avance a través de los nuevos estadios del progreso tecnológico permite una nueva forma de descentralización operativa, comprensiva desde una dimensión social y desde aspectos de gestión con el objetivo de posibilitar el acercamiento de las diversas Administraciones a todos los habitantes independientemente del lugar en que se encuentren.

- ✓ Posibilita un modelo de arquitectura más horizontal que vertical, que vincula y permite el acceso y la interoperabilidad sistémica de la información de las diferentes instituciones entre sí y con los integrantes de la sociedad.
- ✓ Se potencia la participación ciudadana que avanza hacia una gestión más transparente en la que el actuar de la Administración refiere a la diafanidad del obrar público, permitiendo ver con claridad el actuar de esta en la disposición y uso de los fondos públicos y en el obrar de sus funcionarios; esto constituye una consecuencia de la muy elemental presunción de que el gobierno pertenece al pueblo, quien tiene derecho a saber qué hacen los servidores públicos, por qué y cómo lo hacen. Evitando así el secretismo que ambienta la corrupción.

1.3.2 Gobierno Electrónico en Venezuela

El Gobierno Electrónico en Venezuela se plantea como meta la transformación del Estado Venezolano, de la Administración Pública, de las estructuras y de los procesos de gobierno a fin de favorecer el acercamiento e intercambio entre el gobierno y el ciudadano y el gobierno y el gobierno con el apoyo de las tecnologías de información y comunicación.

El objetivo fundamental del Gobierno Electrónico en Venezuela es:

- ✓ Apoyar la constitución de un nuevo modelo de Estado definido en el nuevo marco constitucional y el nuevo modelo de gestión en el proceso de transformación del Estado.
- ✓ Propiciar el control social y establecer la corresponsalía como un nuevo esquema de relación entre el ciudadano y el Estado.
- ✓ Contribuir mediante el uso intensivo de las TIC a la racionalización de las tramitaciones públicas, logrando a tal efecto una mayor celeridad y funcionalidad.
- ✓ Reducir los gastos operativos en que incurren los organismos públicos y obtener así ahorros presupuestarios que permitan cubrir insuficiencias de carácter fiscal, mejorando las relaciones administración pública-ciudadano.
- ✓ Establecer un modelo de arquitectura más horizontal, empírico y endógeno, que vincule y permita el acceso y la interoperabilidad sistémica de la información de las diferentes instituciones del gobierno hacia el ciudadano.
- ✓ Proveer de mayor acceso a la información gubernamental.
- ✓ Sistematizar la responsabilidad y transparencia en los procesos de la Administración Pública.

- ✓ Ser un país integrado, eficiente y competitivo en el ámbito regional e internacional, que garantice a todos los ciudadanos en el territorio nacional el acceso democrático a los beneficios y oportunidades que la sociedad de la información, las comunicaciones y las tecnologías generen.

Para el Dr. Carlos Figueira, presidente del Centro Nacional de Tecnologías de Información (CNTI) de Venezuela, el gobierno venezolano reconoce la importancia de un instrumento que posibilite la interacción de los diferentes actores de la sociedad con el Estado, lo que se traduce en servicios eficientes y procedimientos transparentes, reducción del burocratismo y optimización de sus recursos financieros, tecnológicos y humanos. Además planteó "Cuando se habla de Gobierno Electrónico en Venezuela y en el mundo, típicamente se habla de la relación Gobierno-Gobierno, Gobierno-Ciudadano". (Directorio del Estado).

En Venezuela se está trabajando para lograr la automatización de los servicios y procedimientos del Estado, lo que involucra aspectos como la creación de un único centro de datos para todas las instituciones gubernamentales, “lo que es más eficiente que tener pequeños centros por cada una de ellas”. La idea principal del Gobierno Electrónico es que el Estado tenga una puerta de entrada única, un portal que permita la interacción con el ciudadano y derivar de ahí, a servicios que pueden ser meramente informativos o transaccionales para el ciudadano, las empresas, las comunidades organizadas y todo aquel que requiera comunicarse u obtener algo del Gobierno.

Venezuela ha desarrollado varios proyectos con el objetivo de modernizar el país automatizando gran cantidad de procesos, los cuales logran mejorar de forma considerable el nivel de vida de los habitantes. Dentro de los ámbitos institucionales a modernizar se encuentra el sistema registral y notarial venezolano. (9)

1.3.3 ¿Cómo converge un sistema de recopilación genérica de datos al Gobierno Electrónico en Venezuela?

Para Venezuela se desarrolló una solución informática que moderniza el sistema registral venezolano y que a la vez garantiza la seguridad jurídica de los documentos. Esta solución como muchas de las desarrolladas actualmente necesita de información inicial para garantizar un buen funcionamiento. El sistema que se propone a partir del análisis y diseño a pesar de ser una aplicación de escritorio tiene funcionalidades que hacen posible la interacción de las oficinas con el MPPRIJ. Una vez que se hayan creado las condiciones de redes necesarias para desarrollar los portales Web, se podrá adaptar el sistema para que el proceso de obtención de la CI se haga por esta vía. Así los responsables en cada institución

accederían a un sitio Web para insertar o corregir algún documento o los datos de la oficina. Por esta razón podemos afirmar que el análisis y diseño de la solución propuesta permitirán crear un sistema que converja al desarrollo del Gobierno Electrónico en Venezuela lográndose un cambio radical en las relaciones gobierno-ciudadano.

1.4 Evolución del diseño de sistemas informáticos

La evolución del diseño del software es un proceso continuo que ha abarcado las últimas cuatro décadas. El primer trabajo de diseño se concentraba en criterios para el desarrollo de programas modulares y métodos para refinar las estructuras del software de manera descendente.

Los aspectos procedimentales de la definición de diseño evolucionaron en una filosofía denominada programación estructurada. Un trabajo posterior propuso métodos para la conversión del flujo de datos o estructura de datos en una definición de diseño. Enfoques de diseños más recientes hacia la derivación de diseño proponen un método orientado a objetos. Hoy en día, se ha hecho hincapié en un diseño de software basado en la arquitectura del software.

La programación orientada a objetos, como paradigma es una filosofía de la que surge una cultura nueva que incorpora técnicas y metodologías diferentes, en ella el universo computacional está poblado por objetos, cada uno responsable de sí mismo, y comunicándose con los demás por medio de mensajes. Cada objeto representa una instancia de alguna clase, y estas clases son miembros de una jerarquía de clases unidas vía relaciones de herencia. La evolución histórica de los paradigmas de programación se podría presentar de la siguiente forma:

- ✓ Programación estructurada.
- ✓ Programación modular.
- ✓ Programación orientada a objetos.
- ✓ Programación orientada a aspectos.
- ✓ Programación orientada a componentes.

Independientemente del modelo de diseño que se utilice, un ingeniero del software deberá aplicar un conjunto de principios fundamentales y conceptos básicos para el diseño a nivel de componentes, de interfaz, arquitectónico y de datos. (10)

1.5 Metodologías de Desarrollo de Software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto y guías para uso de herramientas de apoyo. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño. (11)

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea estructurada, Orientada a Objeto (OO), tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

Metodología estructurada: Estas metodologías son particularmente apropiadas en proyectos que utilizan para la implementación lenguajes de 3ra y 4ta generación. Ejemplos de metodologías estructuradas de ámbito gubernamental son MERISE2, MÉTRICA3, SSADM4. Ejemplos de propuestas de métodos estructurados en el ámbito académico son Gane & Sarson5, Ward & Mellor6, Yourdon & DeMarco7 e Information Engineering8.

Metodología orientada a objeto: Surge a partir de la idea de Booch y Rumbaugh de unir sus métodos y notaciones que posteriormente da lugar al Unified Modeling Languaje (UML), la notación OO más popular de la actualidad. Algunos métodos OO con notaciones predecesoras de UML son: OOAD (Booch), OOSE (Jacobson), Coad & Yourdon, Shaler & Mellor y OMT (Rumbaugh).

Metodología tradicional: Las metodologías tradicionales o formales se focalizan en documentación, planificación y procesos, plantillas, técnicas de administración y/o revisiones. Rational Unified Process (RUP) y Microsoft Solution Framework **(**MSF) asoman como las principales metodologías tradicionales.

Metodología Ágil: Se basan en retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. Estas metodologías ponen en relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Como parte de ésta aparecen Extreme Programing (XP), SCRUM y Ágil Unified Process (AUP). (12)

1.5.1 Rational Unified Process (RUP)

RUP es una metodología de desarrollo de Software que proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es garantizar la producción de alta calidad de software que satisfaga las necesidades de sus usuarios finales, dentro de un calendario y el presupuesto previsible.

Aumenta la productividad del equipo aportando a cada miembro el acceso fácil a una base de conocimientos con las directrices, modelos y herramientas, para el desarrollo de las actividades fundamentales. Teniendo a todos los miembros del equipo accediendo a la misma base de conocimientos, no importa si se trabaja con requisitos, diseño, prueba, gestión de proyecto, o gestión de configuración, se asegura que todos los miembros del equipo compartan un lenguaje común, el proceso y vista de cómo desarrollar el software.

Las actividades de RUP crean y mantienen modelos. En lugar de centrarse en la producción de gran cantidad de documentos en papel, el Proceso Unificado hace hincapié en el desarrollo y mantenimiento de los modelos con representaciones semánticamente del sistema en desarrollo.

Es una guía para saber cómo utilizar efectivamente el Lenguaje Unificado de Modelado (UML). UML es un lenguaje estándar de la industria que permite comunicar claramente los requisitos, arquitecturas y diseños.

Es apoyado por herramientas que automatizan gran parte del proceso. Se utilizan para crear y mantener los diversos artefactos del proceso de ingeniería de software: modelado visual, programación, pruebas, etc.

Es un proceso configurable. Ningún proceso es adecuado para todos los desarrollos de software. El Proceso Unificado se inscribe en pequeños equipos de desarrollo y en grandes organizaciones de desarrollo. Contiene un kit de desarrollo, que prevé el soporte a la prestación para la configuración de proceso que se adapten a las necesidades de una organización determinada.

Recoge muchas de las mejores prácticas en desarrollo de software moderno de una forma adecuada para una amplia gama de proyectos y organizaciones. El despliegue de estas buenas prácticas mediante ofrece una serie de ventajas al equipo de desarrollo. (19)

Dentro de los elementos de modelado que propone RUP se encuentran los trabajadores que definen el quién, los artefactos que definen el qué, el flujo de actividades el cuándo, y las actividades el cómo.

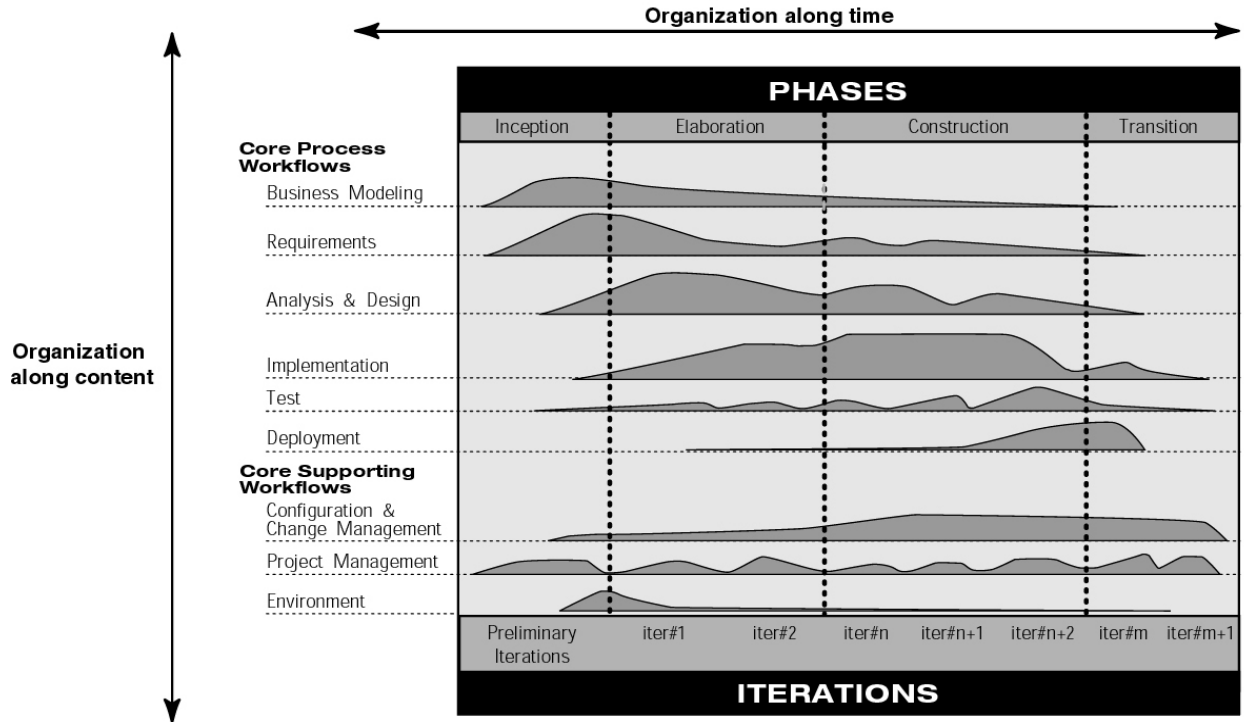

Figura 1 Flujos de trabajo y fases de RUP

1.5.2 Agile Unified Process (AUP) (13)

AUP es una adaptación de Unified Process (UP) ágil formalizada por Scott W. Ambler[3], describe de forma simple y fácil la creación de software funcional usando técnicas ágiles y aplicando las mejores características de RUP. AUP aplica técnicas comunes a otros enfoques ágiles como el diseño conducido por pruebas, desarrollo dirigido por modelos ágiles, gestión ágil de los cambios, y técnicas de refactorización de la base de datos para mejorar la productividad.

[3] Scott W. Ambler: Escritor de varios libros sobre el desarrollo de software orientado a objetos y sobre metodologías de desarrollo agiles como Agile Model Driven Development (AMDD), Agile Database Techniques, Agile UP y Enterprise Unified Process (EUP). Actualmente trabaja con el Grupo de Software de la IBM.

En AUP se establecen cuatro fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados:

Concepción (Inception): El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

AUP tiene solamente siete disciplinas:

Modelo: Entender el negocio del cliente, el dominio del problema que trata el proyecto, e identificar una solución viable para tratar el problema.

Implementación: Transformar el modelo en código ejecutable y realizar pruebas a un nivel básico.

Prueba: Realizar una evaluación objetiva para asegurar la calidad. Esto incluye encontrar defectos, validar que el sistema trabaja según lo diseñado, y verificar que los requisitos están resueltos.

Despliegue: Plan para la entrega del sistema haciendo que esté disponible para los usuarios.

Gestión de Configuración: Manejar el acceso a los artefactos del proyecto. Esto incluye no solamente seguir versiones de los artefactos sino controlando los cambios que se producen en ellos.

Gestión de Proyecto: Dirigir las actividades que ocurre dentro del proyecto. Esto incluye riesgos asumidos, dirección y coordinación del personal (asignación de tareas, seguimiento del proyecto, etc.), y gestión de los recursos externos para asegurar la entrega del producto dentro del tiempo y presupuesto.

Ambiente: Apoyar el esfuerzo asegurando que el proceso, guías (estándares y pautas), y las herramientas apropiadas (hardware, software, etc.) estén disponibles para el equipo según lo necesiten.

AUP define algunos productos ágiles de trabajo tales como:

El código fuente para el sistema.

Una colección completa de pruebas (unidad, sistema y aceptación), y el código que las ejecuta en el orden adecuado.

Instalación de Sistema y scripts de instalación.

Documentación de Sistema y Notas del release entregadas como parte del sistema para ayudar a trabajar a las partes interesadas y los desarrolladores.

Un modelo de requisitos mínimos (por ejemplo, el modelo de CU, modelo de dominio, glosario, etc) sólo lo que sea suficiente para comprender y, a continuación, construir lo que los usuarios necesitan.

Un mínimo del Modelo Diseño (por ejemplo, un modelo de objeto y / o modelo de datos, un modelo de despliegue, y una vista de la documentación del sistema)

1.5.3 SCRUM

SCRUM es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. SCRUM se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming.

SCRUM se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente.

En SCRUM, el equipo se focaliza en una única cosa: construir software de calidad. Por el otro lado, la gestión de un proyecto SCRUM se focaliza en definir cuáles son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en remover cualquier obstáculo que pudiera

entorpecer la tarea del equipo de desarrollo. Se busca que los equipos sean lo más efectivos y productivos que sea posible.

SCRUM tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. (20)

1.5.4 Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y equipos pequeños. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Esta metodología se basa en pruebas unitarias, la refabricación y la programación en pares.

Propone que se empiece a desarrollar desde pequeño y se añadan funcionalidades con retroalimentación continua, que el manejo del cambio se convierta en parte sucesiva del proceso, que el costo del cambio no dependa de la fase o etapa, que no se introduzcan funcionalidades antes de ser necesario y que el cliente o usuario se convierta en miembro del equipo. (21)

Ventajas

Apropiado para entornos volátiles

Estar preparados para el cambio, significa reducir su coste.

Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio

Permitirá definir en cada iteración cuales son los objetivos de la próxima.

Permite tener realimentación de los usuarios muy útil.

La presión esta a lo largo de todo el proyecto y no en una entrega final.

Desventajas

Delimitar el alcance del proyecto con nuestro cliente. (12)

1.5.5 Selección de la Metodología de desarrollo.

Se seleccionó RUP como metodología de desarrollo debido a los conocimientos que se tienen en la Universidad de las Ciencias Informáticas (UCI) sobre la misma, además entrega una forma disciplinada de asignar tareas y responsabilidades. Su meta es asegurar la producción de software de alta calidad, que cumpla las necesidades de los usuarios, dentro de las restricciones. Utiliza el desarrollo iterativo para enfrentar el riesgo. Además es una metodología que genera gran cantidad de documentación.

1.6 Proceso de Desarrollo de Software.

Un proceso determina quien está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. Un proceso efectivo proporciona normas para el desarrollo eficiente del software de calidad, captura y presenta las mejores prácticas que el estado actual de las tecnologías permite. En consecuencia reduce el riesgo y hace el producto más predecible. El proceso debe estar ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentran implicados. (14)

El proceso de desarrollo de software es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo. Es el conjunto de actividades necesarias para transformar los requisitos un usuario en un producto de software. (14)

El proceso de desarrollo de software no es único. No existe un proceso universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos (15)

- ✓ Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
- ✓ Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.
- ✓ Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- ✓ Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Además de estas actividades fundamentales, existe un conjunto de actividades protectoras, que se aplican a lo largo de todo el proceso del software (16). Ellas se señalan a continuación:

- ✓ Seguimiento y control de proyecto de software.
- ✓ Revisiones técnicas formales.
- ✓ Garantía de calidad del software.
- ✓ Gestión de configuración del software.
- ✓ Preparación y producción de documentos.
- ✓ Gestión de reutilización.
- ✓ Mediciones.

1.6.1 Flujo de Trabajo de Análisis y Diseño definido por RUP.

El flujo de Análisis y Diseño es el flujo que mayor peso tiene en la fase de elaboración. Durante el análisis se analizan los requisitos levantados en la captura de requerimientos, refinándolos y estructurándolos, el objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que sea fácil de estructurar el sistema entero, incluyendo su arquitectura.

El análisis se ocupa de los Casos de Uso (CU) significativos para la arquitectura. Por lo general esta proporción es menos del 10 % del total. También se analizan los CU para entenderlos de forma más precisa y discernir la interferencia de unos con otros. (14)

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales.

Los trabajadores en esta disciplina son el arquitecto, que responde por la integridad del modelo de análisis y por la arquitectura del modelo de análisis; el ingeniero de casos de uso (CU), que responde por la integridad de una o más realizaciones de CU y el ingeniero de componentes, que define y mantiene las clases y las relaciones entre ellas y la integridad de uno o varios paquetes del análisis. Los artefactos que construyen estos trabajadores son el modelo de análisis, las clases de análisis, la realización de los CU del análisis, los paquetes del análisis y la descripción de la arquitectura.

Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código. Al principio de la fase de de la elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos. El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas. Otro producto importante de este flujo es la documentación de la arquitectura software, que captura varias visiones arquitectónicas del sistema. (17)

Los trabajadores de esta disciplina son el arquitecto de software, que desarrolla el modelo de despliegue, el documento de la arquitectura; el diseñador, que determina las clases de análisis, la realización de los CU, los subsistemas de diseño; el diseñador de interfaz de usuario, que realiza el prototipo de interfaz de usuario; el diseñador de Base de Datos, que realiza el modelo de datos y el diseñador de pruebas que se encarga de incorporar las decisiones referentes a los elementos estructurales de pruebas y sus colaboraciones

1.6.2 Métricas de Evaluación.

Una métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Incluye el método de medición que constituye el proceso por el cual se obtiene una medida que es a su vez el valor asignado a un atributo de una entidad mediante una medición.[4]

Los objetivos de las métricas en general son: entender y mejorar la calidad del producto, evaluar y mejorar la efectividad del proceso y mejorar la calidad del trabajo realizado en el proyecto.

Las métricas son la maduración de una disciplina que van a ayudar a la evaluación de los modelos de análisis y de diseño, donde proporcionarán una indicación de la complejidad de diseños procedimentales y

[4] Concepto emitido por la IEEE (The Institute of Electrical and Electronics Engineers) en el año 1993.

de código fuente, y ayudaran en el diseño de pruebas más efectivas. El proceso de medición se caracteriza por cinco actividades:

- ✓ *Formulación*: La obtención de medidas y métricas del software apropiadas para la representación de software en cuestión.
- ✓ *Colección*: El mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.
- ✓ *Análisis*: El cálculo de las métricas y la aplicación de herramientas matemáticas.
- ✓ *Interpretación*: La evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.
- ✓ *Realimentación*: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

Características medibles de un diseño OO

Tamaño, se mide a través de:

- ✓ Población: Recuento de entidades OO, clases u operaciones.
- ✓ Volumen: Ídem, pero dinámicamente en un instante de tiempo dado.
- ✓ Longitud: Se mide una cadena de elementos de diseño interconectados (ej: la profundidad de un árbol de herencia).
- ✓ Funcionalidad: proporcionan una indicación indirecta del valor entregado al Cliente por la aplicación OO.

Complejidad: medir en términos de características estructurales, examinando cómo se interrelacionan las clases.

Acoplamiento: se miden conexiones físicas entre los elementos del diseño (número de colaboraciones entre clases o número de mensajes intercambiados entre objetos).

Suficiencia: un componente de diseño (clase) es suficiente si refleja completamente todas las propiedades del objeto dominio de la aplicación que se modela; es decir, la clase posee los rasgos imprescindibles.

Integridad: ¿Qué propiedades se requieren para representar completamente el objeto dominio del problema?

Cohesión: un componente OO debe poseer todas las operaciones trabajando conjuntamente para alcanzar un propósito único y bien definido. La cohesión de una clase se determina examinando el grado en que "el conjunto de propiedades que posee sea parte del diseño o dominio del problema".

Originalidad: grado en que "una operación es atómica", es decir, no puede ser construida fuera de una secuencia de otras operaciones contenidas en la clase.

Similitud: grado en que dos o más clases son similares en términos de estructura, comportamiento, función o propósito.

Volatilidad: la volatilidad de un componente de diseño OO mide la probabilidad de que un cambio ocurra.

Las métricas han de ser utilizadas para el control de los proyectos. No son ni estándares ni universales, sino que cada proyecto debe seleccionar sus propias métricas en dependencia de sus características. Estas se pueden clasificar de manera general como:

- ✓ Métricas orientadas a la función
- ✓ Métricas orientadas al tamaño.
- ✓ También se pueden clasificar según la información que entregan:
- ✓ Métricas de productividad: Las que se centran en el rendimiento del proceso de ingeniería de software.
- ✓ Métricas de calidad: Proporcionan una indicación de cómo se ajusta el software a los requisitos explícitos e implícitos del cliente.
- ✓ Métricas técnicas: Se centran más en el software que en el proceso a través del cual se ha desarrollado (por ejemplo grado de modularidad o grado de complejidad lógica).

Métricas de validación para Diseños OO.

Métricas de alto nivel: Ayudan a localizar los módulos más complejos y, por lo tanto, aquellos en los que se debe poner especial atención. También es utilizada para saber el número de módulos asignados a cada trabajador.

Algunas de estas métricas son:

- ✓ Profundidad de herencia
- ✓ Descendientes
- ✓ Acoplamiento
- ✓ Invalidación en la herencia
- ✓ Añadidas por subclase

Métricas de bajo nivel: Las métricas de bajo nivel, también llamadas métricas de caja blanca, ayudan a conocer las interioridades del sistema, estas son:

- ✓ Métodos por clase
- ✓ Respuesta
- ✓ Cohesión
- ✓ Tamaño

1.6.3 Patrones de diseño

"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 ni siquiera dos veces de la misma forma".[5]

Un patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre como asignar las responsabilidades a los objetos ante determinada categoría de problemas. (18)

Patrón = <**problema, contexto, solución**> +<**recurrencia, enseña, nombre**>

Los patrones de diseño proporcionan esquemas para refinar subsistemas o componentes de un sistema, ayudan a los diseñadores a reutilizar con éxito los diseños. Su objetivo es guardar la experiencia en diseños de programas orientados a objetos y hacer más fácil la reutilización de los diseños y arquitecturas; ayudan a elegir entre diseños alternativos, hacen a un sistema reutilizable y evitan alternativas que comprometen la reutilización.

[5] Christopher Alexander: Arquitecto, reconocido por sus diseños destacados de edificios en California, Japón y México.

Cualidades de un patrón de diseño

Encapsulamiento y abstracción: Cada patrón encapsula un problema bien definido y su solución en un dominio particular.

Extensión y variabilidad: Cada patrón debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solucionar un gran problema.

Generatividad y composición: Cada patrón una vez aplicado genera un contexto resultante, el que concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo.

Equilibrio: Cada patrón debe realizar algún tipo de balance entre sus efectos y restricciones

1.7 Lenguajes de Modelado.

Un Lenguaje de modelado es la notación, mayoritariamente gráfica, que utilizan los métodos para expresar los diseños. (11)

1.7.1 Unified Modeling Language (UML)

El Lenguaje Unificado de Modelado, en adelante UML (Unified Modeling Language), es el resultado más integrador de una serie de métodos de análisis y diseño orientado a objetos.

Posee una semántica bien definida lo que hace posible la integración de la tecnología de verificación y validación formal, tecnología que precisa descripciones formales de los sistemas, en los ciclos de desarrollo software orientado a objetos. Permite pues la definición de modelos sin ambigüedad cuya completitud y consistencia semántica pueden ser comprobadas con el apoyo de herramientas software diseñadas al respecto, que asimismo pueden ofrecer la propia ejecución, validación y simulación automática de los modelos. La posibilidad de conectar dichos modelos con diversos lenguajes de programación (tales como Java, C++ o Visual Basic) permite también el diseño de herramientas para la generación automática de código y recíprocamente, para soportar procesos de ingeniería inversa (es el caso de la herramienta Rational Rose).

UML promueve particularmente procesos de análisis iterativos, increméntales y en diferentes niveles de abstracción. De este modo resulta muy útil para definir modelos genéricos que se especializan en dominios concretos. (27)

1.7.2 Goal-oriented Requirement Language (GRL)

GRL es un lenguaje para apoyar el modelado orientado a objetivos y de razonamientos con requisitos no funcionales. (28)

GRL forma parte de la notación URN (User Requirements Notation), que ha sido propuesta como estándar de la ITU-T (International Telecommunication Union –Telecommunication Standardization Sector). Distingue tres categorías de conceptos principales: elementos intencionales, relaciones intencionales y actores que no admiten especializaciones. GRL ofrece constructores para establecer relaciones con elementos externos al modelo, elementos no-intencionales y atributos de conexión, y posee elementos adicionales de justificación y contextualización como correlaciones, tipos de contribuciones y etiquetas de evaluación para especificar los estados de satisfacción, ampliando así los tipos y rango de calificaciones de las relaciones intencionales. (29)

1.7.3 BPMN

Notación de Gestión de Procesos de Negocio (del inglés BPMN). Este es un lenguaje gráfico basados en diagramas de flujo, permite a los analistas de negocio modelar los procesos de manera intuitiva. No obstante, no es ejecutable, y requiere de herramientas que realicen la transformación a lenguajes ejecutables como BPEL. Esta transformación no suele ser fiable ni correcta, por lo que se requiere del trabajo de validación que se realiza con personal técnico especializado. (30)

1.7.4 Selección del lenguaje de modelado

A partir del estudio realizado se determinó utilizar como lenguaje de modelado UML porque es el resultado más integrador de los métodos de análisis y diseño orientado a objetos, no se limita solo a etapas determinadas del desarrollo del software como BPMN y GRL, sino que además de permitir modelar el negocio y los requerimientos permite desarrollar el análisis y el diseño que son las etapas que se manejan en el trabajo. Además el desarrollo de los procesos de forma iterativa, incremental y en diferentes niveles de abstracción.

1.8 Herramientas CASE

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de

software. Una herramienta CASE (Computer- Aided Software Engineering) se puede definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. (22)

1.8.1 Visual Paradigm for UML Enterprise Edition

Visual Paradigm para UML (VP) es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este es un software propietario, pero se ha desarrollado una versión para software libre: Visual Paradigm UML Community Edition que corre en Windows 95, Windows Me, Windows NT, Windows 2000, Windows XP, Windows 98.

VP permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (23)

Facilita el modelado, construcción y despliegue de software. Soporta un conjunto de lenguajes de generación de código y de ingeniería inversa en Java, C + +, CORBA IDL, PHP, XML Schema, Ada y Python. Además, apoya la generación de código C #, VB. NET, Objeto Definition Language (ODL), Flash ActionScript, Delphi, Perl, Objective-C, y Ruby. Ingeniería inversa también apoya clase Java, .NET .dll y .exe, JDBC, y los archivos de mapeo Hibernate. Además soporta la importación y exportación de XMI de versiones 1.0, 1.2 y 2.1. Puede importar archivos con extensiones .MDL y .CAT del Rational Rose a través del Rose Importer.

Para aprovechar al máximo la interoperabilidad de productos de Visual paradigma con otras aplicaciones, se ha introducido la importación/exportación de modelado de proyectos desde / a un formato XML abierto. Usuarios y proveedores de tecnología pueden integrar modelos paradigma visual en sus soluciones con un mínimo esfuerzo. (24)

1.8.2 Rational Rose

Rational Rose es una herramienta propietaria con un elevado precio (7,485.48 €). Soporta construcciones de modelado incluidas la ejecución de modelos y la generación completa de código ejecutable.

Es una solución sólida de desarrollo dirigida por el modelo. Una conversión de diseño a código totalmente automatizada para Java, C y C++. Crea automáticamente unidades, adaptadores y scripts de prueba reales. Optimizada para aplicaciones simultáneas, dirigidas por sucesos y distribuidas. Construcciones de

modelado avanzado para satisfacer los estrictos requisitos de latencia, rendimiento y fiabilidad. Diseñada para las aplicaciones que suponen un mayor reto tecnológico. Incluye IBM Rational Rose para UNIX/Linux e IBM Rational Rose Enterprise para Windows a la integración de nivel de diseño con Java, C++ y Ada (25)

Rational Rose Enterprise Edition

Rational Rose Enterprise Edition es la versión de Rational Rose para Windows. Es una herramienta que soporta la generación de código a partir de modelos en Ada, ANSI C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado que facilita la creación de software de calidad.

Características adicionales incluidas:

- ✓ Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software"
- ✓ Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos
- ✓ Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5
- ✓ La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables
- ✓ Soporte Enterprise Java Beans™ 2.0
- ✓ Capacidad de análisis de calidad de código
- ✓ El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web
- ✓ Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos
- ✓ Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación
- ✓ Integración con otras herramientas de desarrollo de Rational
- ✓ Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase

- ✓ Publicación web y generación de informes para optimizar la comunicación dentro del equipo. (26)

1.8.3 Selección de la Herramienta Case

Se seleccionó como herramienta para el modelado el Visual Paradigm for UML Enterprise Edition debido a que es una herramienta fácil de utilizar, además de soporta un conjunto de siete lenguajes de generación de código entre los que se encuentran los que utiliza el Rational Rose y algunos adicionales. Además apoya otros lenguajes de generación de código y de ingeniería inversa y puede importar archivos XML y al igual que el Rational .MDL y CAT.

1.9 Sistemas Gestores de Base de Datos

Los sistemas de Gestión de Bases de Datos, son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. Los SGBD es la aplicación que interactúa con los usuarios de los programas de aplicación y la base de datos.

Algunos de los SGBD más conocidos son: SQL, DB2, SLQ/DS, ORACLE, INGRES, INFORMIX, SYBASE, PARADOX, DBASE, ACCESS, FOXPRO, R, RM/T y RM/V2. (31)

1.9.1 PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos, de software libre, publicado bajo la licencia BSD. Su desarrollo no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Ventajas

- ✓ Es un SGBD de código abierto: No tiene costo asociado a la licencia del software.
- ✓ Ahorros considerables en costos de operación: Ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.
- ✓ Estabilidad y Confiabilidad Legendarias: En varios años no ha presentado caídas de operación de alta actividad.
- ✓ Extensible: El código fuente está disponible para todos sin costo. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo.

- ✓ Multiplataforma: PostgreSQL está disponible en Linux, en casi cualquier Unix, y ahora en versión nativa para Windows.
- ✓ Diseñado para ambientes de alto volumen: PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes.
- ✓ Herramientas gráficas de diseño y administración de BD: Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect). (32)

1.9.2 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario creado por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL y de la marca. Tiene un esquema de licenciamiento dual.

MySQL es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, que en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

Ventajas

- ✓ Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- ✓ Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- ✓ Facilidad de configuración e instalación.
- ✓ Soporta gran variedad de Sistemas Operativos
- ✓ Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- ✓ Conectividad y seguridad

Desventajas

- ✓ Un gran porcentaje de las utilidades de MySQL no están documentadas.
- ✓ No es intuitivo, como otros programas (ACCESS). (33)

1.9.3 Oracle

Es un sistema propietario de gestión de base de datos relacional considerada como uno de los sistemas de bases de datos más completos, destacado por su soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Oracle se encuentra disponible en una variedad de ediciones: Express Edition, Standard Edition One, Standard Edition, y Enterprise Edition. Todas las ediciones fueron creadas utilizando la misma base de código, lo cual significa que sus aplicaciones de base de datos pueden fácilmente escalar de servidores de procesadores sin cambiar ninguna línea de código. La única edición gratuita de Oracle es la Express Edition, compatible con las demás ediciones.

La seguridad de las aplicaciones y bases de datos de Oracle permiten la administración completa de los usuarios, el monitoreo de potenciales ataques y la asignación de privilegios de acceso a través de toda la empresa. Por sus altos niveles de seguridad, estas bases de datos han sido implementadas por organizaciones que manejan información crítica como entidades de servicios financieros y del sector gobierno.

Además elimina los riesgos de inseguridad al proteger los datos transmitidos desde el cliente hasta el servidor de aplicaciones, encriptando los datos sensibles dentro de la base de datos, restringiendo el acceso de los usuarios en el nivel de filas, ofreciendo un único punto de ingreso a todas las aplicaciones autorizadas y detectando rápidamente, el uso inadecuado de los datos. De modo que las aplicaciones que utilizan Oracle son soluciones tecnológicas diseñadas para contingencias no planificadas, como fallos de sistemas, y capaces de soportar todas las aplicaciones empresariales brindando la mejor escalabilidad y disponibilidad de la industria. (34)

1.9.4 Selección del Sistema gestor de Base de Datos.

Se determinó a PostgreSQL como sistema gestor de Base de Datos porque es un sistema de software libre cuyo código fuente está disponible para todos sin costo, diseñado para ambientes de alto volumen, muy estable y confiable, que al igual que MySQL y Oracle es multiplataforma y seguro.

1.10 Conclusiones

Como parte de este capítulo se analizó cómo influye el sistema al que se le desarrolla el análisis y el diseño en el Gobierno Electrónico en Venezuela.

Se estudiaron las metodologías, lenguajes de modelado y herramientas Case y se determinó utilizar RUP, UML y Visual Paradigm for UML Enterprise Edition respectivamente debido a la experiencia que se tiene y las facilidades que brindan.

Se analizaron algunos sistemas gestores de BD y se determinó usar PostgreSQL, cuya utilización permitirá que el sistema, una vez implementado, cuente con una BD potente, segura y multiplataforma, lo que contribuye al desarrollo de los países tecnológicamente dependientes.

CAPITULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

2.1 Introducción

En este capítulo se realizará el análisis y el diseño de la solución que se propone, exponiendo los actores del sistema, los requisitos que debe tener el mismo, el diagrama de Caso de Uso (CU) del sistema, así como la descripción de los Casos de Uso, los diagramas de clases del análisis y los diagramas de interacción. Además se exponen otros artefactos que define la metodología seleccionada para documentar la solución.

2.2 Descripción del sistema propuesto

El Análisis y Diseño que se propone permitirá desarrollar un sistema de recolección genérica de datos para desplegar cualquier sistema informático en el ámbito jurídico. La solución CISJUR (Carga Inicial para Sistemas Jurídicos), podrá ser utilizada no sólo en la digitalización del sistema registral sino también en la modernización de cualquier entidad pública con el fin de mejorar la vida de los ciudadanos. Permitirá capturar cualquier tipo de información inicial en cuanto a datos documentales jurídicos y características de las instituciones, así como enviar documentos con errores desde una misma aplicación y desde un mismo puesto.

CISJUR deberá cumplir con algunas funcionalidades. Para esto se tuvo en cuenta que el mismo se dividirá en dos partes fundamentales: el acceso a la aplicación por los funcionarios de las instituciones jurídicas y por el administrador.

Inicialmente se mostrarán activas las acciones a las que accederá el funcionario de la institución jurídica, en el caso de las acciones del administrador permanecerán inhabilitadas hasta que el éste introduzca los datos de acceso a la BD.

Accediendo el funcionario al sistema podrá acceder en la interfaz principal a las opciones adicionar tipo documental, buscar documentos, generar reportes, gestionar datos de la institución jurídica, insertar o corregir documentos.

La opción buscar documentos permite hacer búsquedas a partir de: el número del documento, la fecha de emisión y el rango de fechas de inserción. La opción generar reportes brinda algunas funcionalidades como la generación de los reportes por fecha de emisión, fecha de inserción, fecha de modificación y un reporte general de documentos. Esta opción dará además la posibilidad de imprimir los reportes. La

opción gestionar datos de la institución jurídica incluye las opciones insertar y modificar datos de la institución. El sistema además incluirá funcionalidades a las que sólo podrá acceder el administrador, con el objetivo de llevar un control general de la información. El administrador podrá, una vez autenticado como tal, buscar documentos, este procedimiento además de las funcionalidades que se le muestran al funcionario, permite al administrador buscar documentos por Institución Jurídica. Puede además generar reportes a partir de las mismas funcionalidades que se le brindan al funcionario y se le adicionan las opciones de generar reportes por entidades jurídicas y reporte de documentos con errores. El administrador podrá además importar archivos a la Base de Datos y eliminar alguna Institución Jurídica en caso que deje de funcionar en un momento determinado.

2.2.1 Descripción de los actores del sistema

Los actores del sistema son terceros fuera del sistema que interactúan con él, representan el rol que juega una o varias personas, un equipo o un sistema automatizado en el sistema, generalmente los trabajadores del negocio son parte de estos. Los actores del sistema no son parte de él, pueden intercambiar información con él y pueden ser un recipiente pasivo de información.

Actor	Descripción
Usuario	El usuario es un actor genérico de quien heredan el responsable de la institución jurídica y el administrador.
Responsable Institución Jurídica	Es un usuario que maneja la información relacionada con la institución donde se instala el software.
Administrador	Es un usuario que tiene la responsabilidad de insertar a la Base de Datos la información generada por las instituciones jurídicas y enviar a estas los documentos que contienen errores, así como otras actividades de administración.

Tabla 1 Descripción de los actores del sistema

2.2.2 Casos de Uso del Sistema

Un Caso de Uso (CU) del sistema es un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. A continuación se muestran algunas de las descripciones detalladas de los CU significativos arquitectónicamente.

Descripción expandida de los casos de uso

Caso de Uso:	Autenticar Acceso a Base de Datos (BD)
Actores:	Administrador
Resumen:	El CU inicia cuando el administrador introduce los datos para acceder a la Base de Datos.
Referencias	R1; R2
Pre-condiciones:	El sistema debe estar instalado y ejecutándose correctamente.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona opción de autenticarse en la Interfaz principal.	2. El sistema verifica disponibilidad de conexión del usuario. 3. Si hay conexión muestra interfaz de autenticación.
4. Introduce su usuario y contraseña	5 Verifica datos. 6 Si los datos son correctos habilita opción de importar datos a la Base de datos en la interfaz principal, la funcionalidad de generar reportes por institución en la interfaz de reporte de documentos y la opción Gestionar Institución Jurídica. 7 Finaliza el Caso de Uso
Flujo Alterno al paso 3 " No hay disponibilidad de conexión del usuario"	
	2.a Muestra un mensaje informándole al usuario que no hay disponibilidad de conexión 2.b Finaliza el caso de uso.
Flujo Alterno al paso 3 "datos incorrectos"	
Acción del Actor	**Respuesta del Sistema**
	4.a Muestra mensaje de error: "No se encontró el usuario". 4.b Ir al paso 2 del FNE

Pos-condiciones	El administrador queda autenticado y queda habilitada la interfaz de importar datos a la Base de datos y reportes por institución en la interfaz de reporte de documentos.

Tabla 2 Descripción expandida del CU: Autenticar Acceso a BD

Caso de Uso:	Gestionar datos de la Institución Jurídica
Actores:	Responsable de la Institución Jurídica
Resumen:	El caso de uso se inicia cuando el responsable de la Institución Jurídica selecciona la opción gestionar datos de la institución. Consiste en registrar los datos de las instituciones jurídicas para mantener un control de las mismas. Muestra la interfaz de gestionar datos vacía si no se han insertado y llena si los datos se insertaron en algún momento. Permite además modificar estos datos. El caso de uso termina con la información insertada o modificada.
Pre-condiciones:	El sistema debe estar instalado y ejecutándose correctamente.
Referencias	R3; R4
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción gestionar datos de la institución.	2. Muestra la interfaz de gestionar datos de la Institución Jurídica. 3. Busca si los datos se insertaron con anterioridad. 4. Si no se insertaron muestra los campos vacios para insertar los datos de la institución. Si se insertaron, muestra campos llenos con datos insertados previamente y habilita opción cancelar.
5. Inserta o modifica los datos y salva los datos.	7. Valida que estén todos los datos y que estén correctos. 8. Actualiza base de datos 9. Muestra mensaje de confirmación 10. Finaliza el CU
Flujo Alterno al paso 5 "Cancela Acción "	
	5.a Muestra interfaz principal y Finaliza el CU.

Pos-condiciones	El sistema inserta y modifica los datos de la institución.

Tabla 3 Descripción expandida del CU: Gestionar datos de la Institución Jurídica

El resto de la descripciones verlas en los Anexos en la sección ***(Descripción detallada de los Casos de Uso)***

2.3 Modelo de Análisis

Contiene clases del análisis y sus objetos organizados en paquetes que colaboran. Contribuye a tener un mejor control sobre el problema a resolver ya que modela la solución de los mismos. En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis.

2.3.1 Clases de Análisis

Las clases de análisis son una abstracción de una o varias clases y/o subsistemas del diseño del sistema que se centra en el tratamiento de los requisitos funcionales, ofrecen una interfaz en términos de operaciones, definen atributos normalmente conceptuales y reconocibles en el dominio del problema, participan en relaciones más conceptuales que en el diseño y la implementación. Los estereotipos de las clases de análisis son interfaz, control y entidad, cada uno expresa una semántica específica que constituye un método potente para identificar y describir las clases del análisis. (14)

Las clases interfaz, modelan la interacción Actor-Sistema. Muestran ventanas, formularios y la comunicación con otros sistemas o dispositivos; las clases control, coordinan el trabajo de las clases, realizan funciones complejas y encapsulan el comportamiento de un CU; y las clases entidad, modelan la información del Sistema y el comportamiento asociado a una información.

2.3.2 Diagrama de clases del análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas.

A continuación se muestran los algunos de los diagramas de clases desarrollados, el resto verlos en la sección de Anexos ***(Diagramas de clases de Análisis).***

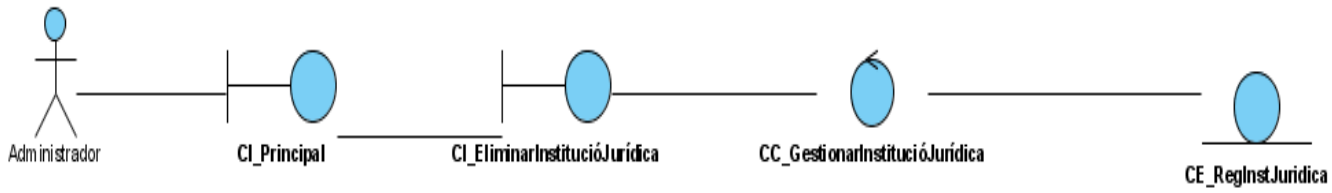

Figura 2 Diagrama de Clases de Análisis: Eliminar Institución Jurídica

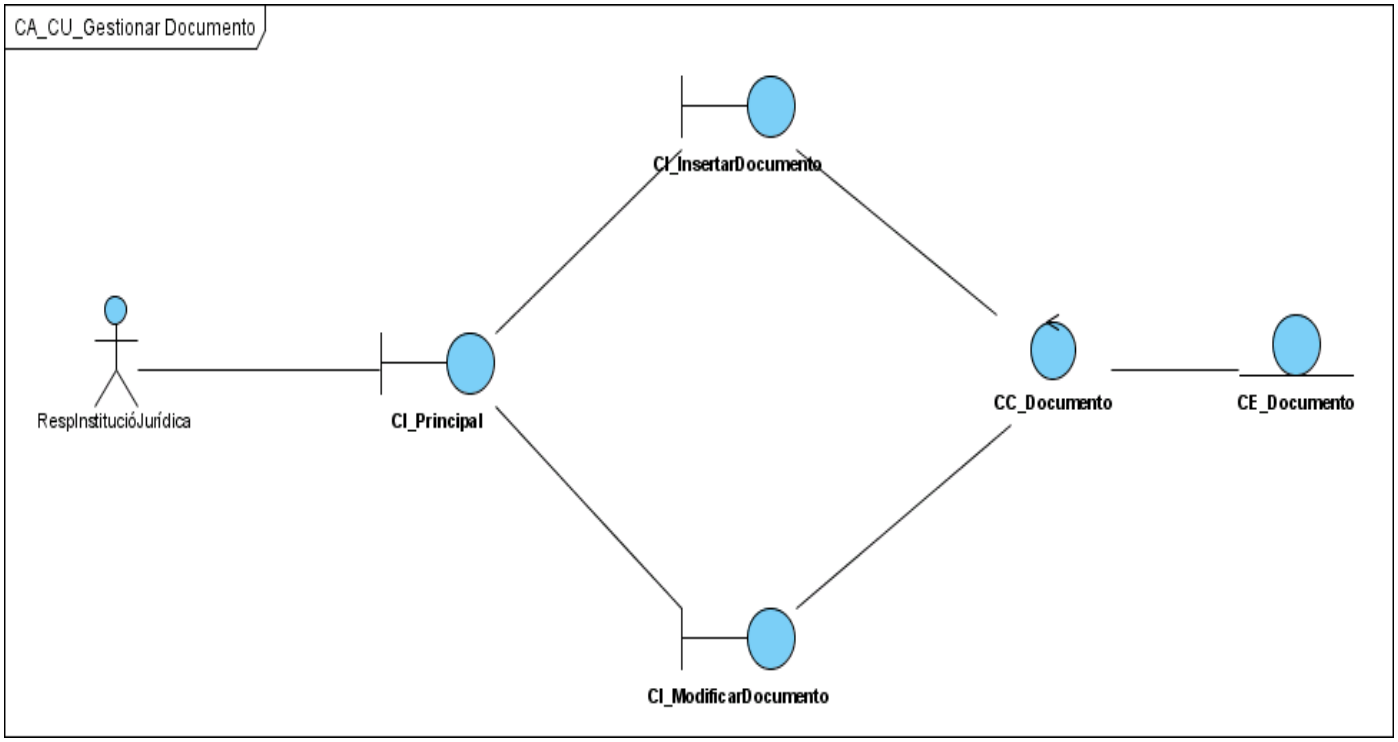

Figura 3 Diagrama de Clases de Análisis: Gestionar Documento

2.3.3 Diagramas de Interacción.

Diagramas de colaboración del análisis

Son una forma alternativa a los diagramas de secuencia. Destaca la organización de los objetos que participan en la interacción. A continuación se muestran los diagramas de los Caso de Uso (CU) Autenticar Acceso a Base de datos y Generar reportes de documento. Para ver el resto de los diagramas dirigirse a la Sección de Anexos ***(Diagramas de Colaboración del análisis).***

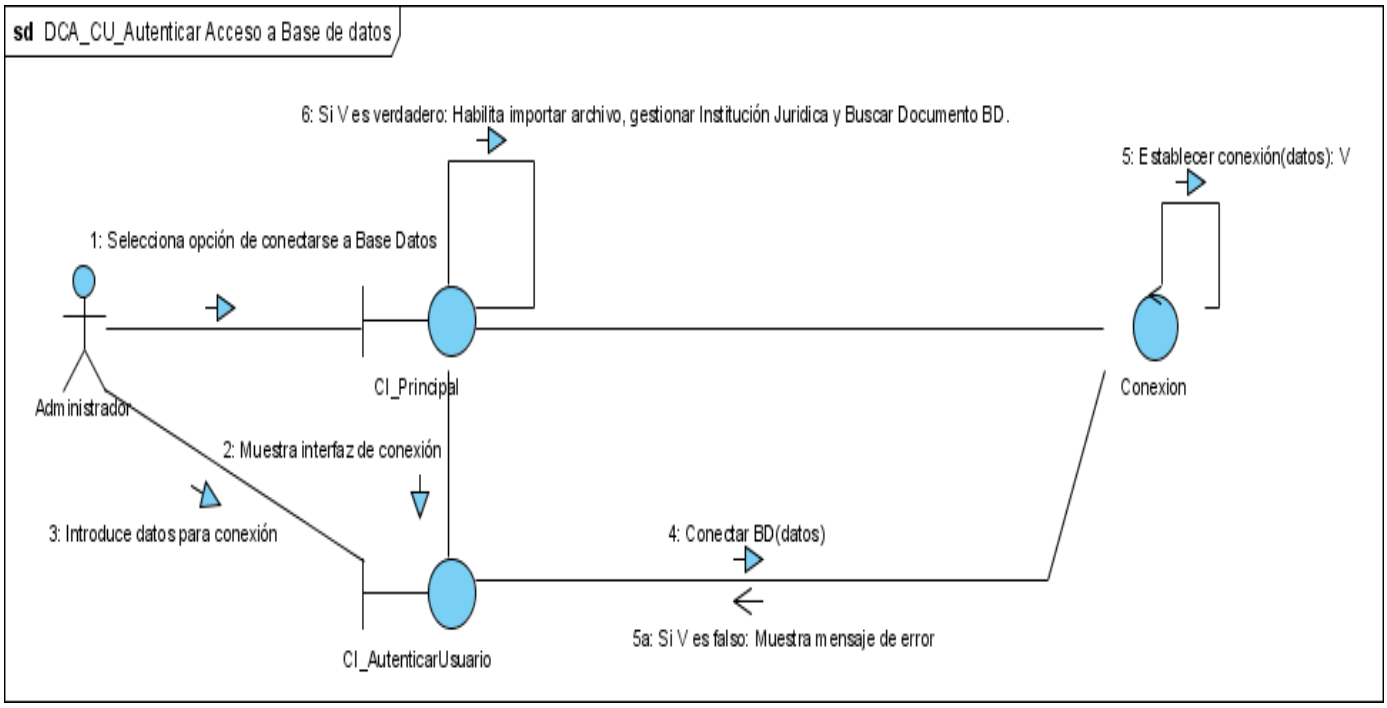

Figura 4 Diagrama de Colaboración del CU: Autenticar acceso a BD

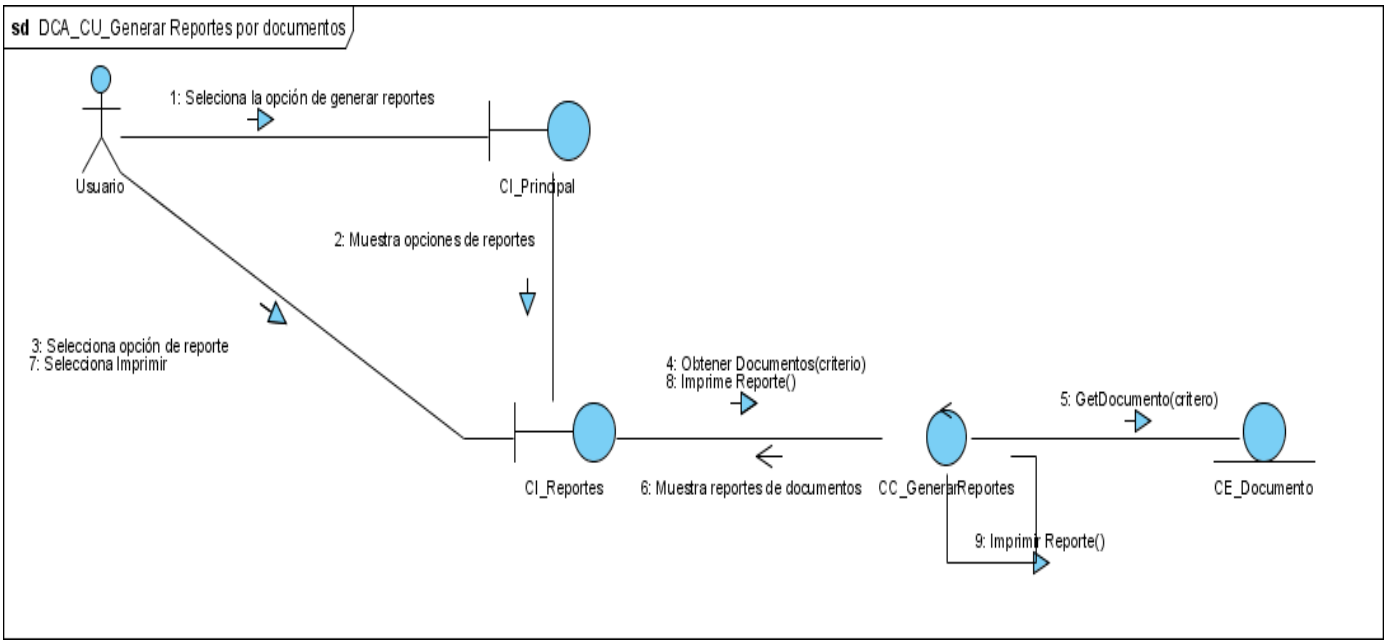

Figura 5 Diagrama de Colaboración del CU: Generar Reportes por documentos

2.4 Vista vertical de la arquitectura

"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"[6]. Esta es la definición que se ha tomado como oficial por todas las instituciones que desarrollan software a nivel mundial, como ha sido por ejemplo Microsoft.

2.4.1 Estilo arquitectónico aplicado

Estilo 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. (36)

De los tres se aplicó la Arquitectura en capas. Este estilo según, funciona como una organización jerárquica tal que cada capa proporciona servicios a cada capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Esta arquitectura puede decirse que su finalidad es abstraer las funcionalidades de una capa de manera tal que pueda ser totalmente remplazada. Dentro de este estilo el más comúnmente aplicado es el estilo en tres capas: presentación, negocio y acceso a datos. De esta forma se puede remplazar cualquier capa sin afectar a las otras, sólo se cambian las referencias de las implicadas en el cambio.

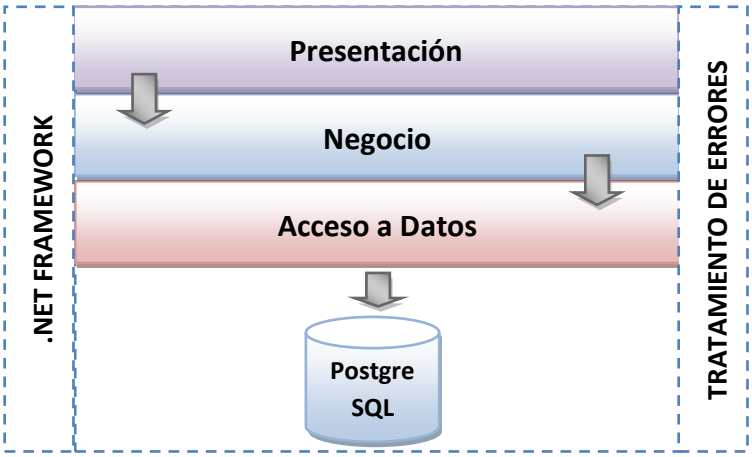

Figura 6 Arquitectura Vertical

[6] Definición de arquitectura de software dada por el documento de la IEEE Std 1471-2000

Capa de Presentación

Esta capa estará compuesta por la capa de Interfaz: Su uso se basa generalmente en la explotación de los componentes de interfaz, se corresponde con funcionalidades de visualización de datos a través de los formularios de presentación. Además provee el framework, como esqueleto sobre el cual varios objetos son integrados para una solución dada, con que se trabajará facilitado el desarrollo del sistema basado en acciones.

Capa de Negocio

La capa intermedia, capa de empresa o lógica de negocio, es básicamente el código al que recurre la capa de presentación para recuperar los datos deseados. La capa de presentación recibe entonces los datos y los formatea para su presentación. Esta separación entre la lógica de aplicación de la interfaz de usuario añade una enorme flexibilidad al diseño de la aplicación. Así se construyen y despliegan múltiples interfaces de usuario sin cambiar en absoluto la lógica de aplicación siempre que ésta presente una interfaz claramente definida a la capa de presentación. Esta capa engloba todas las clases que son principalmente persistentes en la base de datos, contenedores de información y las encargadas la realización de funciones para que el negocio funcione correctamente.

Capa de Acceso a Datos

Esta capa garantiza que la información fluya desde el negocio hasta la base de datos y viceversa. Está formada por dos tipos de clases principalmente, las entidades mapeadas de la Base de Datos, y las clases que gestionan todas las funcionalidades que se realizan en la base de datos sobre las entidades, ya sean procedimientos almacenados o vistas.

2.5 Modelo de diseño

El modelo de diseño es un modelo que describe la realización física de los casos de uso. Se centra en el impacto que tienen los requisitos funcionales y no funcionales en el sistema. RUP propone que el artefacto Modelo de Diseño contenga una Introducción, que no es más que una descripción textual que sirve como breve introducción al modelo; paquetes y subsistemas de diseño; contiene también diagramas, específicamente los diagramas de clases del diseño y diagramas de interacción del diseño, estos últimos también llamados realización de casos de uso; además contiene clases, interfaces y relaciones contenidas en los paquetes.

2.5.1 Subsistemas de Diseño

Los subsistemas de diseño son una forma de organizar los artefactos del diseño en piezas más manejables. Puede contener clases de diseño, realizaciones de CU, Interfaces y otros subsistemas. Por otro lado puede proporcionar interfaces que representan las funcionalidades que exportan en término de operaciones. Un subsistema debe ser altamente cohesivo, o sea, que sus contenidos se encuentren fuertemente asociados, y debe ser también débilmente acoplados, de forma que las dependencias entre unos y otros, o entre sus interfaces, debería ser mínima.

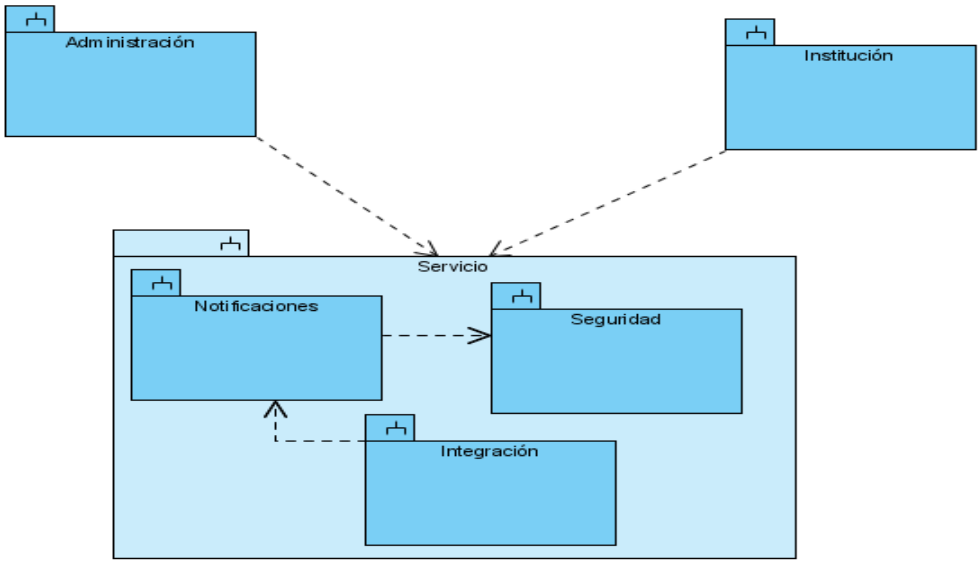

Figura 7 Subsistemas de diseño

2.5.2 Clases de Diseño

Son una abstracción sin costura de una clase o construcción similar en la implementación del sistema. El lenguaje utilizado en una clase de diseño es el mismo lenguaje de programación. Se especifican la visibilidad de los atributos y las operaciones. La relación entre las clases generalmente tiene un significado directo cuando la clase es implementada, los métodos tienen correspondencia directa con los métodos utilizados en la implementación. Puede posponer el manejo de algunos requisitos para las subsiguientes actividades de implementación, indicándolos como requisitos de implementación de la clase. Generalmente aparece con un estereotipo sin costura que se corresponde con una construcción en el lenguaje de programación dado. Puede realizar y proporcionar interfaces si es necesario en el lenguaje

de programación. Puede activarse indicando que objetos de la clase mantengan su propio hilo de control y se ejecuten concurrentemente con otros objetos activos. (14)

2.5.3 Diagrama de clases de diseño

Es una representación más concreta que el diagrama de clases del análisis. Representa la parte estática del sistema, así como las clases y sus relaciones. A continuación se muestran los diagramas de clases correspondientes a los Casos de Uso Gestionar documento y gestionar institución jurídica, estos reflejan las clases con sus relaciones, ya sean clases de interfaz, representadas por <frmNombreInterfaz>, las clases controladoras, representadas por <ccNombreClase>, las clases de acceso a datos representadas por <accNombreGestor>, y las clases de entidades del negocio, representadas por<NombreEntidad>.

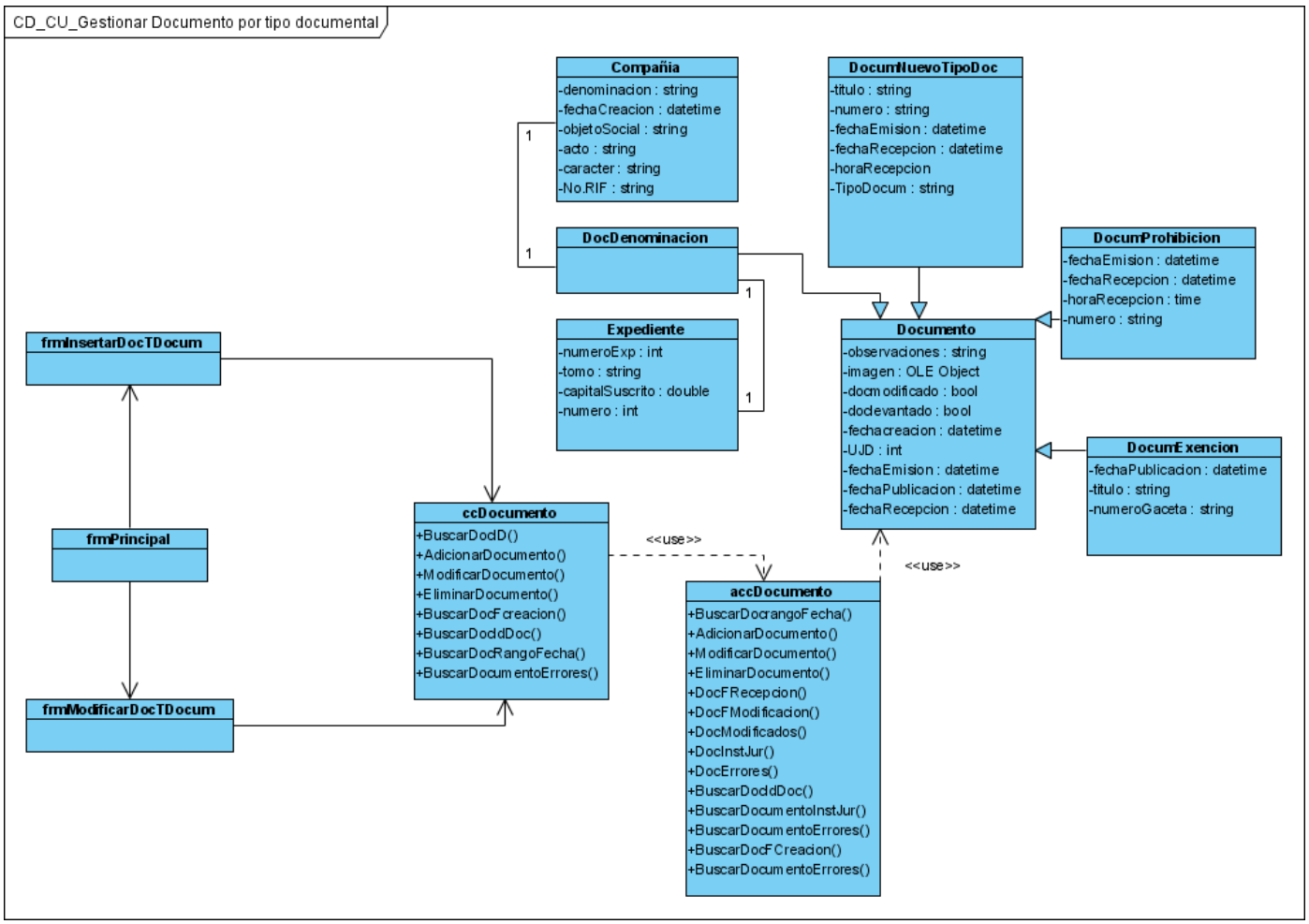

Figura 8 Diagrama de clases de diseño del CU: Gestionar documento

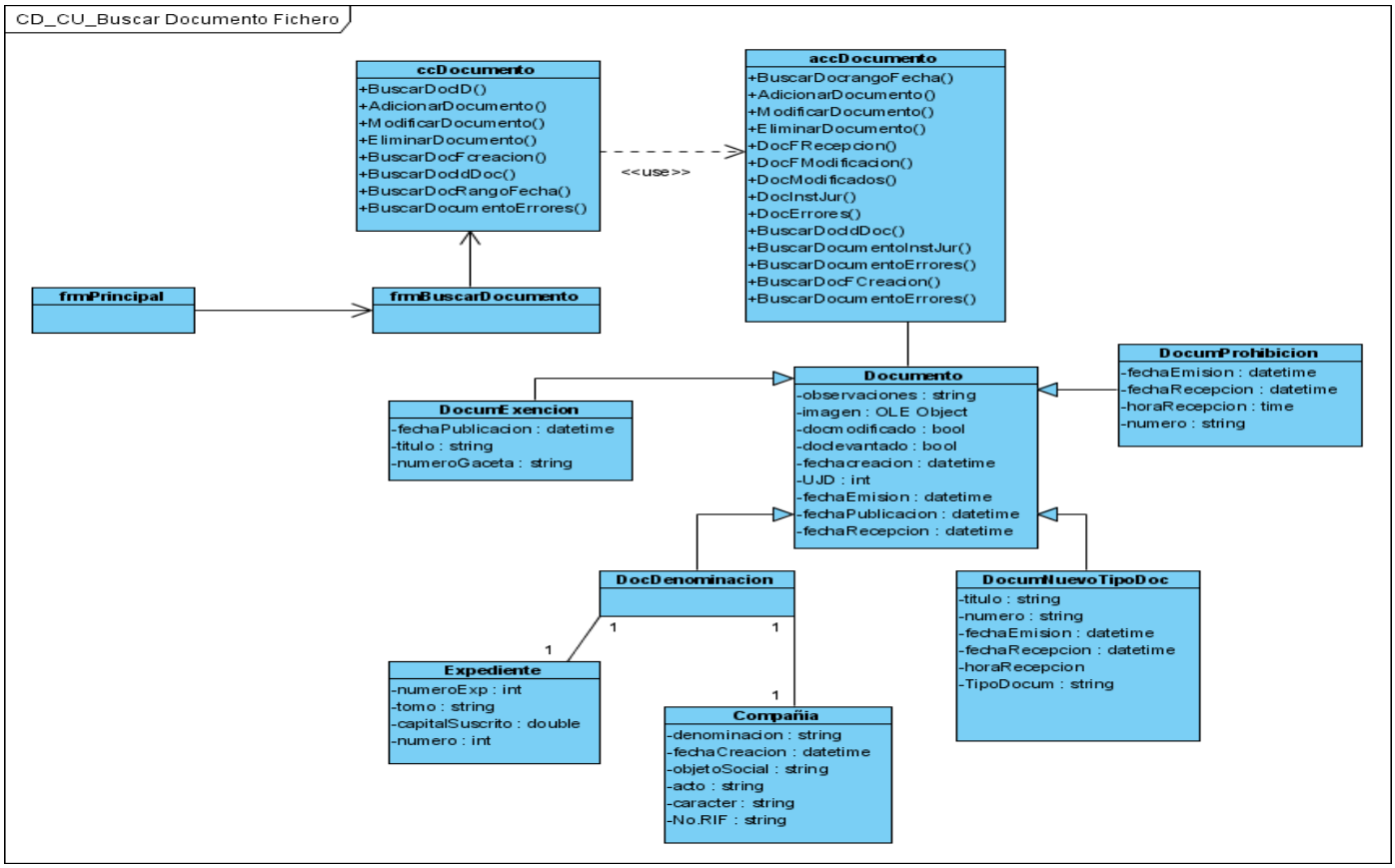

Figura 9 Clase de Diseño del CU: Buscar Documentos fichero

El resto de los diagramas verlos en la sección de Anexos ***(Diagramas de clases de diseño)***

2.5.4 Diagramas de interacción del diseño

Explica gráficamente las interacciones existentes entre las instancias y las clases del modelo de estas. El punto de partida de las interacciones es el cumplimiento de las poscondiciones de los contratos de operación. UML define dos tipos de estos diagramas, secuencia y colaboración; ambos sirven para expresar interacciones semejantes o idénticas de mensaje.

Diagramas de Secuencia del diseño

Un diagrama de secuencia destaca la ordenación temporal de los mensajes. Estos tienen dos características fundamentales que los distinguen de los diagramas de colaboración: la línea de vida, que representa la existencia de un objeto a lo largo de un período de tiempo, y el foco de control que

representa el período de tiempo durante el cual el objeto ejecuta la acción, bien sea directamente o a través de un procedimiento subordinado, también puede encontrarse un anidamiento del foco de control que puede estar causado por recursión, una llamada a una operación propia o una llamada desde otro objeto. A continuación se muestran los diagramas de secuencia de los CU Enviar documentos con errores e Importar archivo a BD. Los diagramas correspondientes al resto de los CU verlos en Anexos ***(Diagramas de Secuencia del diseño)***

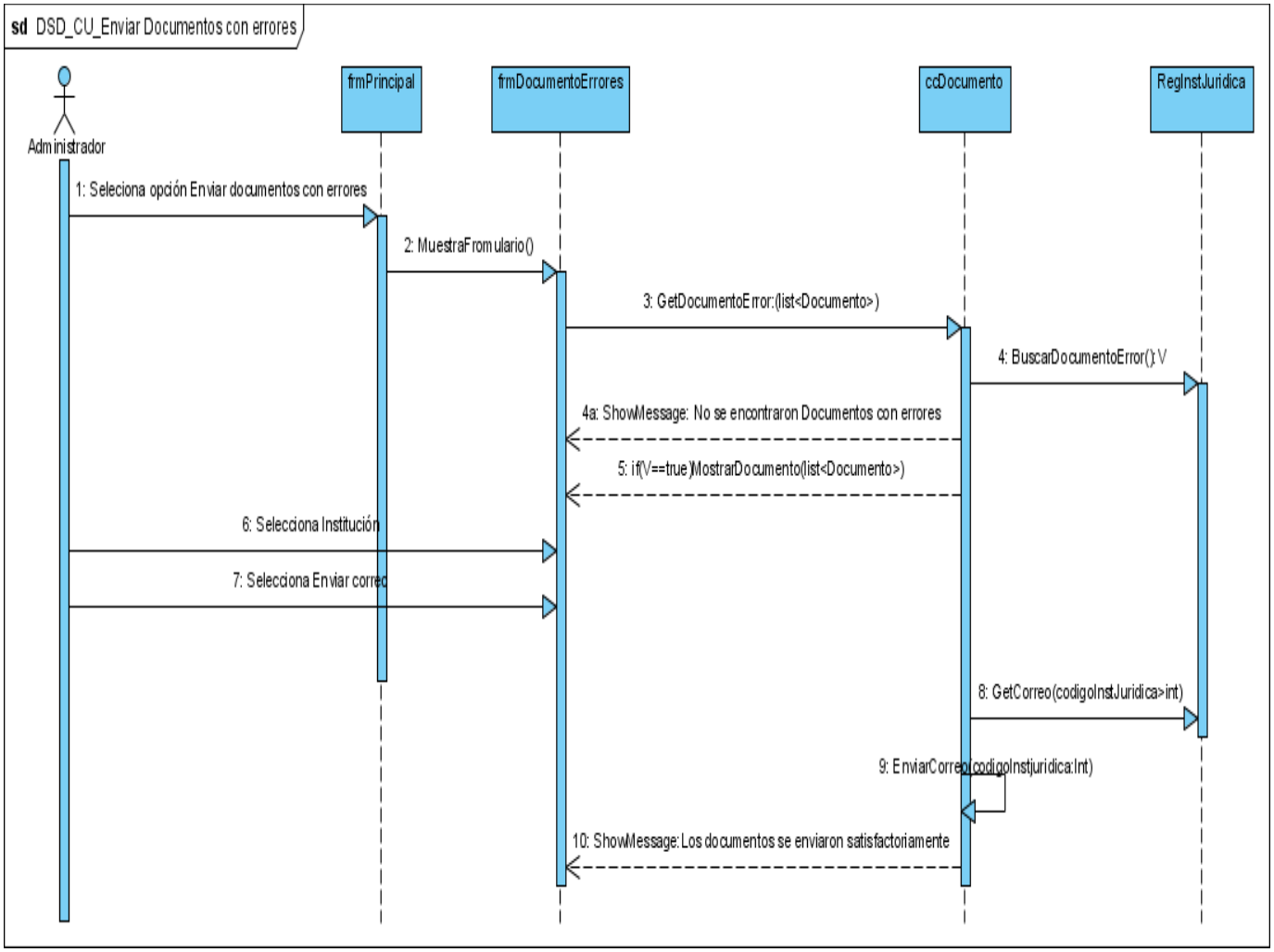

Figura 10 Diagrama de Secuencia del diseño del CU: Enviar Documentos con errores

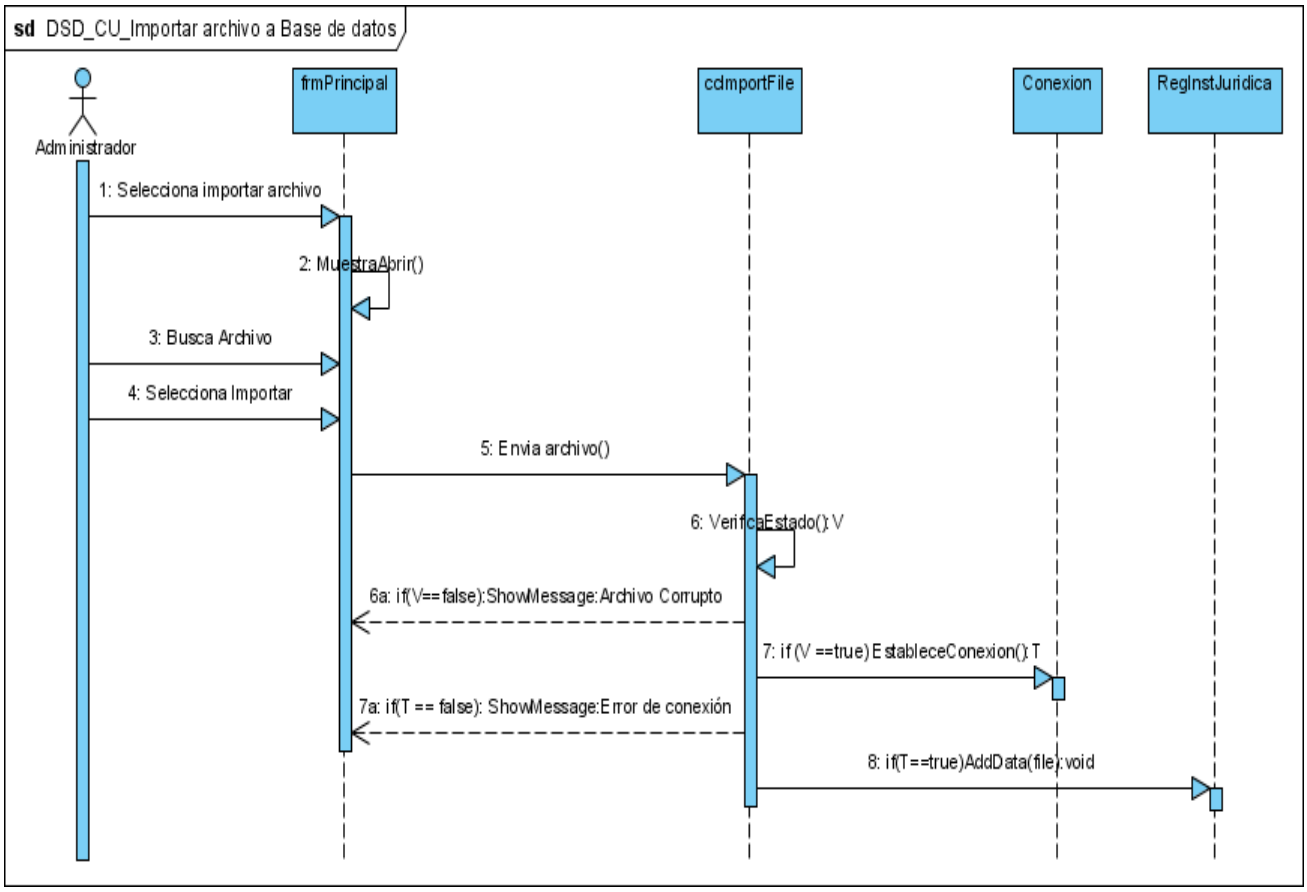

Figura 11 Diagrama de Colaboración del diseño del CU: Importar archivo a Base de Datos

2.5.5 Prototipo de interfaz de Usuario.

El proceso global para el diseño de la interfaz de usuario comienza con la creación de diferentes modelos de funcionamiento del sistema. Cuando se va a diseñar la interfaz de usuario entran en juego cuatro modelos diferentes. El ingeniero del software crea un *modelo de diseño*; cualquier otro ingeniero (o el mismo ingeniero del software) establece un *modelo de usuario,* el usuario final desarrolla una imagen mental que se suele llamar *modelo de usuario* o *percepción del usuario,* y los que implementan el sistema crean una *imagen de sistema*. Un modelo de diseño de un sistema completo incorpora las representaciones del software en función de los datos, arquitectura, interfaz y procedimiento. El modelo de usuario representa el perfil de los usuarios finales del sistema. La percepción del usuario es la imagen final del sistema que el usuario tiene en su mente. Por último, la imagen del sistema es una combinación de fachada externa del sistema basado en computadora (la apariencia del sistema) y la información de

soporte (libros, manuales, cintas de video, archivos de ayuda) todo lo cual ayuda a describir la sintaxis y la semántica del sistema.

Cuando la imagen y la percepción del sistema coinciden, los usuarios generalmente se sienten a gusto con el software y con su funcionamiento. Para llevar a cabo esta mezcla de modelos, el modelo de diseño deberá desarrollarse con el fin de acoplar la información del modelo de usuario, y la imagen del sistema deberá reflejar de forma precisa la información sintáctica y semántica de la interfaz.

A continuación se muestran algunas de las interfaces de usuario desarrolladas como parte de la solución propuesta, el resto verlas en la sección de Anexos (Interfaz de Usuario). (35)

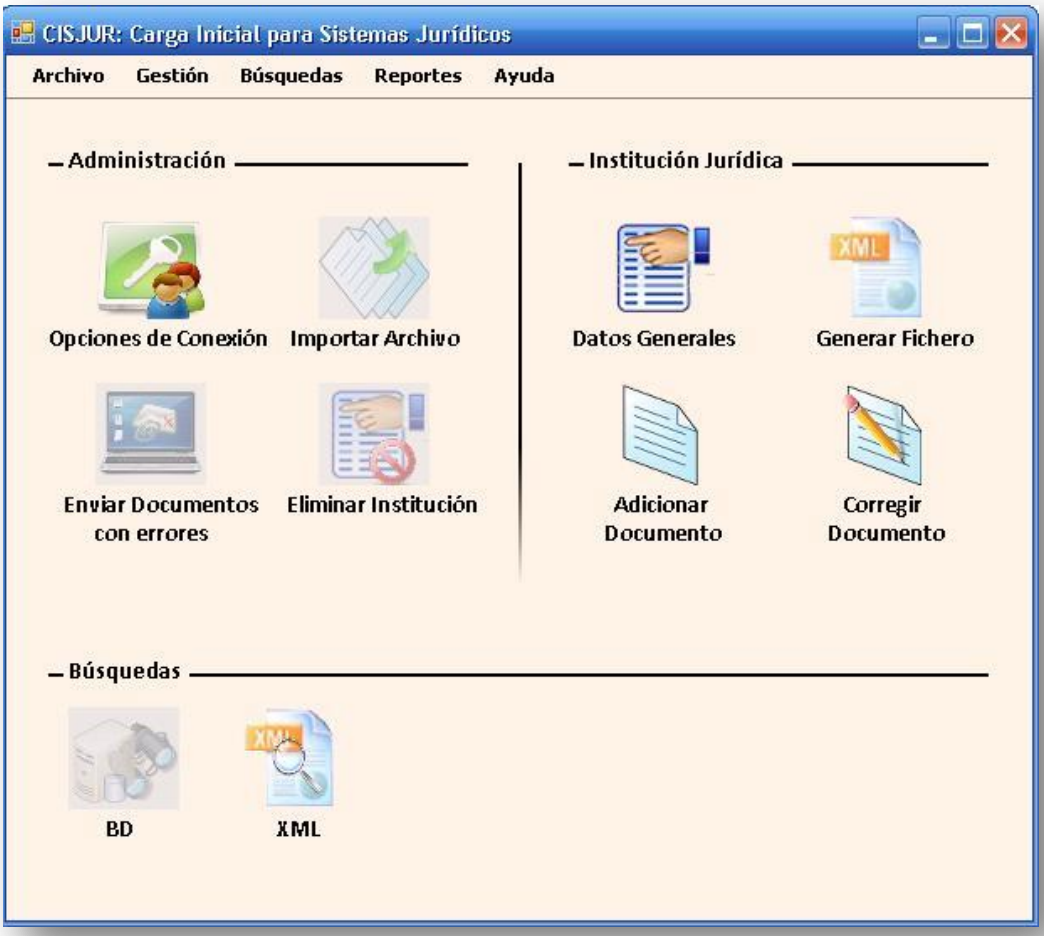

Figura 12 Interfaz principal

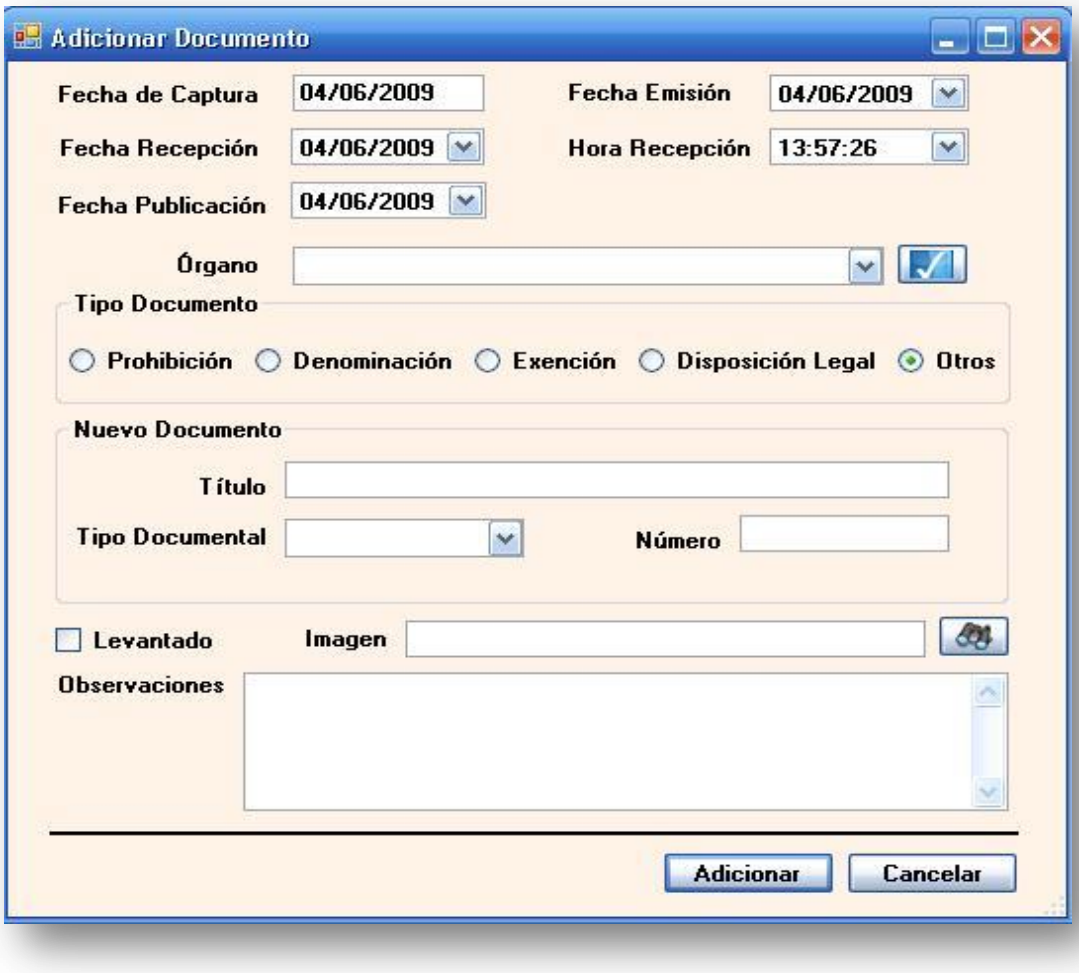

Figura 13 Adicionar Nuevo Documento

2.6 Patrones de Diseño

Los diseñadores expertos en orientación a objetos van formando un amplio repertorio de principios generales y de expresiones que los guían al crear software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su soluci6n, y si se les asigna un nombre.

- ✓ Patrones de Arquitectura:

Describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Ejemplos: Capas (Layers), Aplicaciones: JVM, API,

Windows NI, Pipes and Filters, Aplicaciones: UNIX; Pizarrón (Blackboard) así como Aplicaciones: Hearsay e Inteligencia Artificial.

- ✓ Tipos de patrones de diseño

Patrones de **creación**. Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resultas dinámicamente decidiendo que clases instanciar o sobre que objetos un objeto delegará responsabilidades

Patrones **estructurales**. Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

Patrones de **comportamiento**. Se utilizan para organizar, manejar y combinar comportamientos.

- ✓ Patrones GRASP:

Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos. 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 podemos encontrar los siguientes: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Fabricación Pura, Indirección, Variaciones Protegidas, No hables con extraños y Polimorfismo.

En el diseño del sistema se aplicaron los siguientes patrones de diseño:

2.6.1 Patrones de diseño aplicados en la solución

Experto:

El patrón GRASP Experto asigna una responsabilidad al experto en información o sea a la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Resuelve el problema sobre cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos.

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Si se hace en forma adecuada, los sistemas tienden a ser más fáciles de

entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Por ejemplo en la solución propuesta la clase que se encarga de contener toda la información referente a la Institución Jurídica es RegInstJurídica. La clase Documento es la experta en lo que respecta a los datos relacionados con los documentos que manejan las instituciones jurídicas.

Clase	Responsabilidades
RegInstJur	Contiene datos de la institución jurídica tales como el nombre, la dirección y el número de fax.
Documento	Maneja datos de los documentos tales como el tipo de documento.

Tabla 4 Aplicación del patrón GRASP Experto en la solución

Entre los beneficios que brinda este patrón se encuentran:

Conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Creador:

Este patrón asigna a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- ✓ B agrega los objetos A.

- ✓ B contiene los objetos A.
- ✓ B registra las instancias de los objetos A.
- ✓ B utiliza específicamente los objetos A.
- ✓ B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
- ✓ B es un creador de los objetos A.

Resuelve el problema de definir quién debería ser responsable de crear una nueva instancia de alguna clase. La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reusabilidad.

En la solución propuesta, quién debería encargarse de crear una instancia Documento. Desde el punto de vista de este patrón, se debe buscar una clase que agregue, contenga y realice otras operaciones sobre este tipo de instancias. En la figura siguiente se muestra un diagrama de clases parcial que refleja que una clase RegInstJurídica contiene muchos objetos Documento. Por ello, el patrón Creador sugiere que RegInstJur es idónea para asumir la responsabilidad de crear las instancias Documento.

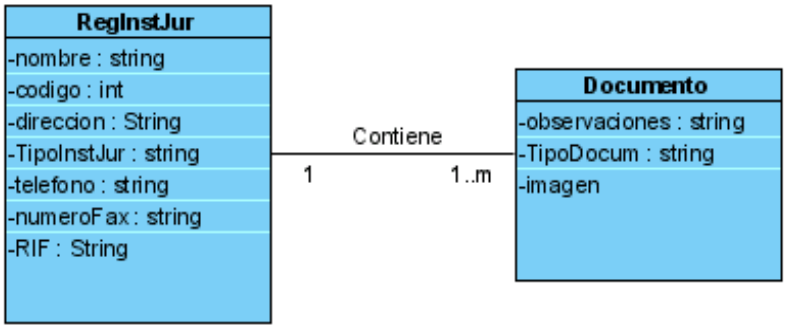

Figura 14 Diagrama de clases parcial

Singleton: Permite garantizar que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Se aplica en la clase RegIntJuridica que tendrá el método singleton de modo que si se va a

ralizar cualquier acción sobre algún dato de ésta, sea a través de este método actualizándose los datos directamente.

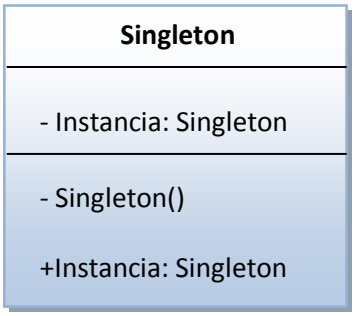

Tabla 5 Estructura del patrón Singlenton

2.7 Diagrama de clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo por tanto las clases persistentes necesitan guardar su estado en un medio permanente ya que estas proporcionan un almacenamiento físico permanente de la información de la clase para la copia de seguridad en caso de caída del sistema, o para el intercambio de información.

Algunas de las reglas que se proponen para el proceso de selección de las clases persistentes son:

- ✓ Cuando una clase que está conformada por otra clase es persistente, automáticamente las clases componentes también son persistentes.
- ✓ Cuando una clase hija de una jerarquía es persistente, automáticamente son persistentes sus ancestros en el árbol de jerarquía.
- ✓ Cuando se define como persistente a una clase que agrupa a objetos del mismo tipo de clase base (clases listas, colecciones y registros), entonces son automáticamente persistentes todas las clases hijas a partir de la clase base, incluyendo a ésta.
- ✓ Si el medio de almacenamiento no forma parte del concepto de clases persistentes y existe herencia múltiple, esta debe ser resuelta con antelación. (37)

Estas por lo general tienen como origen las clases clasificadas como entidad porque modelan la información y el comportamiento asociado a algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso. A continuación se muestra el diagrama de clases persistentes:

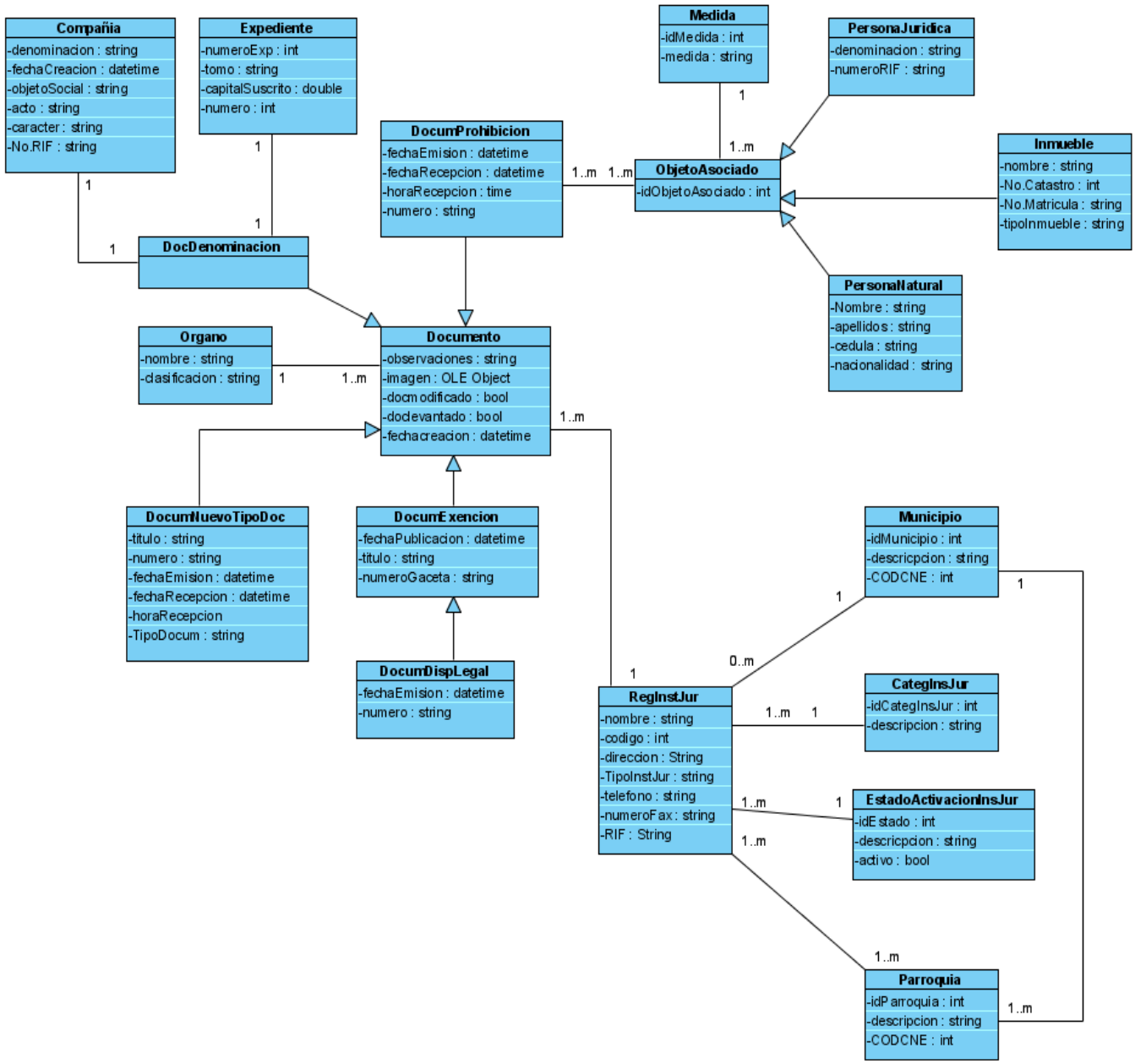

Figura 15 Diagrama de clases persistentes

2.8 Diagrama de Despliegue

Los diagramas de despliegue muestran la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos, lo que significa que relaciona las PC clientes con los servidores y los dispositivos que se utilizarán en el despliegue de la solución.

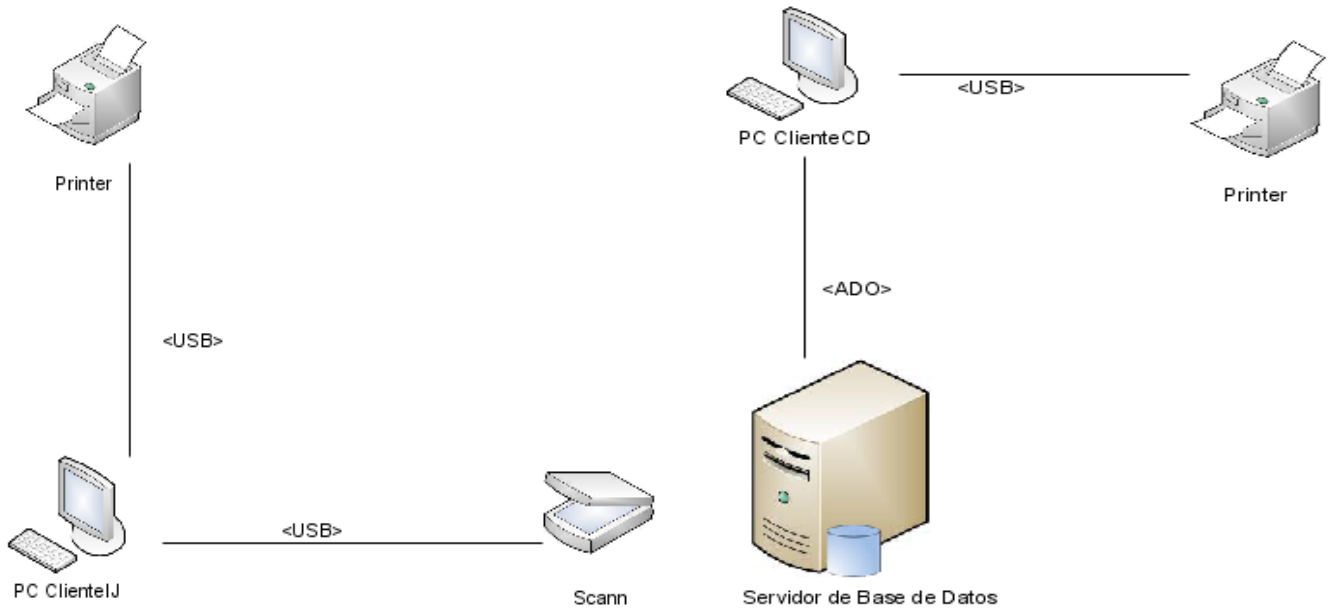

Figura 16 Diagrama de despliegue

2.9 Conclusiones

En el trabajo se desarrolló el análisis de la solución describiendo los procesos realizados y documentando los artefactos definidos por la metodología de desarrollo seleccionada; obteniendo como resultado los diagramas de clases, de colaboración, secuencia y la descripción de los Casos de Uso arquitectónicamente significativos.

Se definió la arquitectura que se aplicará en el sistema, demostrando la independencia de cada una de las capas definidas a partir del estilo arquitectónico aplicado. Vertical a las capas se encuentra el framework de .NET como marco de trabajo y el tratamiento de errores.

Se realizó el diseño del sistema que arrojó como resultado los diagramas de clases, de interacción, de despliegue y de clases persistentes con los cuales se da paso a la fase de implementación.

CAPÍTULO 3: MÉTRICAS PARA LA EVALUACIÓN DEL DISEÑO

3.1 Introducción

Uno de los objetivos fundamentales del uso de métricas en sistemas software es evaluar los mismos con el fin de controlar y mejorar su calidad. Por otra parte, cuestiones como el tiempo estimado para el desarrollo de un proyecto y los costes de mantenimiento una vez que el producto está acabado y funcionando no pueden ser fácilmente resueltas sin la ayuda de la medición.

El Software OO, como el que se pretende construir, es fundamentalmente distinto del software que se desarrolla utilizando métodos convencionales. Las métricas para sistemas OO deben de ajustarse a las características que distinguen el software OO del software convencional como son el encapsulamiento, la herencia, complejidad de clases y polimorfismo.

En este capítulo se estará abordando la validación del diseño de la solución propuesta a través de métricas. Se hace referencia a algunos aspectos generales de las éstas y se presentan los resultados obtenidos una vez aplicadas algunas de ellas al diseño de la solución.

3.2 Métricas para la evaluación del diseño.

En la etapa del diseño la calidad aumenta en la medida en que se realiza una alta especificación de los procesos y se propone una estrecha tolerancia a la modificación, estableciendo los métodos correctivos a las desviaciones ocurridas. (38)

La *calidad de diseño* se refiere a las características que especifican los ingenieros de software para un elemento. El grado de materiales, tolerancias y las especificaciones del rendimiento contribuyen a la calidad del diseño. Cuando se utilizan materiales de alto grado y se especifican tolerancias más estrictas y niveles más altos de rendimiento, la calidad de diseño de un producto aumenta, si el producto se fabrica de acuerdo con las especificaciones.

3.2.1 Tipos de Métricas OO

Métricas Orientadas a Clases: Existen varios tipos de series aplicadas en este sentido, se pueden citar algunas como las de Chidamber y Kemner (CK), Lorentz y Kidd (LK), Li y Henry, Henderson-Sellers.En el diseño desarrollado se aplicaron algunas de las métricas propuestas por las series CK y LK, por ser las más reconocidas y las que más se ajustan al diseño desarrollado.

Serie CK

- ✓ Métodos ponderados por Clase (MPC).
- ✓ Árbol de Profundidad de Herencia (APH)
- ✓ Número de Descendientes (NDD)
- ✓ Acoplamiento entre Clases Objeto (ACO)
- ✓ Respuesta Para una Clase (RPC)
- ✓ Carencia de Cohesión de los Métodos (CCM)

Serie LK

- ✓ Tamaño de Clase (TC)
- ✓ Número de Operaciones redefinidas por una subclase (NOR)
- ✓ Número de Operaciones Añadidas por una subclase (NOA)
- ✓ Índice de Especialización (IE)

Métricas Orientadas a las Operaciones:

- ✓ Tamaño medio de Operación (TO_{medio})
- ✓ Complejidad de Operación (CO)
- ✓ Número medio de Parámetros por operación (NP_{media})

Métricas para Pruebas OO: Son las ***m***étricas con influencia directa en la comprobación de sistemas OO***.***

Encapsulamiento

- ✓ Carencia de Cohesión de Métodos (CCM)
- ✓ Porcentaje Público y Protegido (PPP)
- ✓ Acceso público a datos miembros (APD)

Herencia

- ✓ Número de Clases Raíz (NCR)
- ✓ Número de Padres Directos (NPD)
- ✓ Número de Descendientes (NDD) y Profundidad del Árbol de Herencia (APH)

Métricas sobre el tamaño del software: El gerente debe estimar el tamaño del software, y el tamaño es directamente proporcional al esfuerzo y duración.

- ✓ Número de Escenarios (NE)
- ✓ Número de Clases Clave (NCC)
- ✓ Número de Subsistemas (NSUB)

3.2.2 Consideraciones en la etapa del diseño

Como el análisis, el diseño de software posee una serie de métodos para apoyarlo. Cada uno de ellos posee su propia notación y heurística para acompañar el desarrollo de este proceso, pero todos ellos se basan en una serie de principios.

- ✓ Los Datos y los algoritmos que los manipulan deben crearse como un conjunto de abstracciones interrelacionadas.
- ✓ Los detalles internos del diseño de las estructuras de datos y los algoritmos deben ocultarse de otros componentes software que hacen uso de dichas estructuras de datos o algoritmos.
- ✓ Los módulos deben exhibir independencia. Los módulos deben estar muy poco acoplados con otros módulos y del ambiente externo, además deben poseer cohesión funcional.
- ✓ Los algoritmos deben diseñarse utilizando un conjunto restringido de constructores lógicos (Programación estructurada) (39)

Un paso previo al establecimiento de las métricas es definir qué atributos o propiedades de qué elementos son los que se medirán. Posteriormente se establece si para medir dichos atributos es necesario medir otros y derivar los que son de interés de ellos o se pueden hacer directamente.

La ISO 9126 da una definición de calidad de producto software en base a atributos en la que no se habla de calidad, en general, sino de calidad interna y calidad externa. Tanto la primera como la segunda se definen como un conjunto de 6 características: (40)

- ✓ Funcionalidad: la capacidad del software de proveer las funciones que cumplen con las necesidades implícitas y explícitas cuando el mismo es utilizado bajo ciertas condiciones.
- ✓ Fiabilidad: la capacidad del software de mantener un nivel específico de rendimiento bajo determinadas condiciones de uso.

- ✓ Usabilidad: la capacidad del producto software de ser entendido, aprendido, usado y atractivo al usuario, cuando se usa bajo ciertas condiciones.
- ✓ Eficiencia: la capacidad del software de ofrecer el rendimiento apropiado con respecto a la cantidad de recursos utilizados, bajo condiciones prefijadas.
- ✓ Mantenibilidad: la capacidad del producto de ser modificado. Dichas modificaciones pueden incluir correcciones, mejoras o adaptaciones a cambios en el entorno y en los requisitos y especificaciones funcionales.
- ✓ Portabilidad: la capacidad del software de ser trasladado de un entorno (informático) a otro.

3.3 Métricas aplicadas en el diseño de la solución

Para la validación del diseño del software se escogieron dos de las series de métricas más conocidas y divulgadas a nivel mundial, la primera fue la serie de métricas CK, la cual de sus seis métricas plantadas, se escogieron dos en particular, por su importancia.

Árbol de Profundidad de Herencia (APH): La métrica APH de una clase A es su profundidad en el árbol de herencia. Si A se encuentra en situación de herencia múltiple la longitud máxima hasta la raíz será el APH. Esta métrica se define como la máxima longitud del nodo a la raíz del árbol. (40)

Número de Descendientes (NDD): NDD de una clase es el número de subclases inmediatamente subordinadas a una clase en la jerarquía. Estas se denominan descendientes. (40)

La segunda serie que se escogió fue la LK, de las cuatro métricas que proponen se escogieron 2 de ellas para la medición de este diseño.

Tamaño de Clases (TC): Las clases pueden medirse determinado el total de operaciones, tanto heredadas como privadas de la instancia que se encapsulan dentro de una clase, más el total de atributos, atributos tanto heredados como privados de la instancia encapsulados por la clases. (36

Número de Operaciones Redefinidas para una Sub-Clase (NOR): Es el caso de que una subclase reemplaza una operación heredada de su superclase por una versión especializada para su propio uso, a esto se le denomina redefinición.

A continuación se explicará el análisis de resultados del diseño planteado anteriormente por cada una de las métricas explicadas.

3.3.1 Árbol de Profundidad de Herencia (APH)

Como se ha visto esta métrica se define como la máxima longitud del nodo a la raíz del árbol. A medida que el APH crece, es posible que clases de más bajos niveles hereden muchos métodos, esto conlleva dificultades potenciales, cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda, APH largo, también conduce a una complejidad del diseño mayor. Por el lado positivo, los valores de APH grandes implican un gran número de métodos que se reutilizarán. (35)

Por su parte, algunos autores sugieren que un umbral de 6 niveles como indicador es un abuso en la herencia en distintos lenguajes de programación.

Esta métrica arrojó como resultado que el nivel más alto de herencia entre las clases del diseño es 3, lo cual se encuentra dentro del umbral definido por los autores.

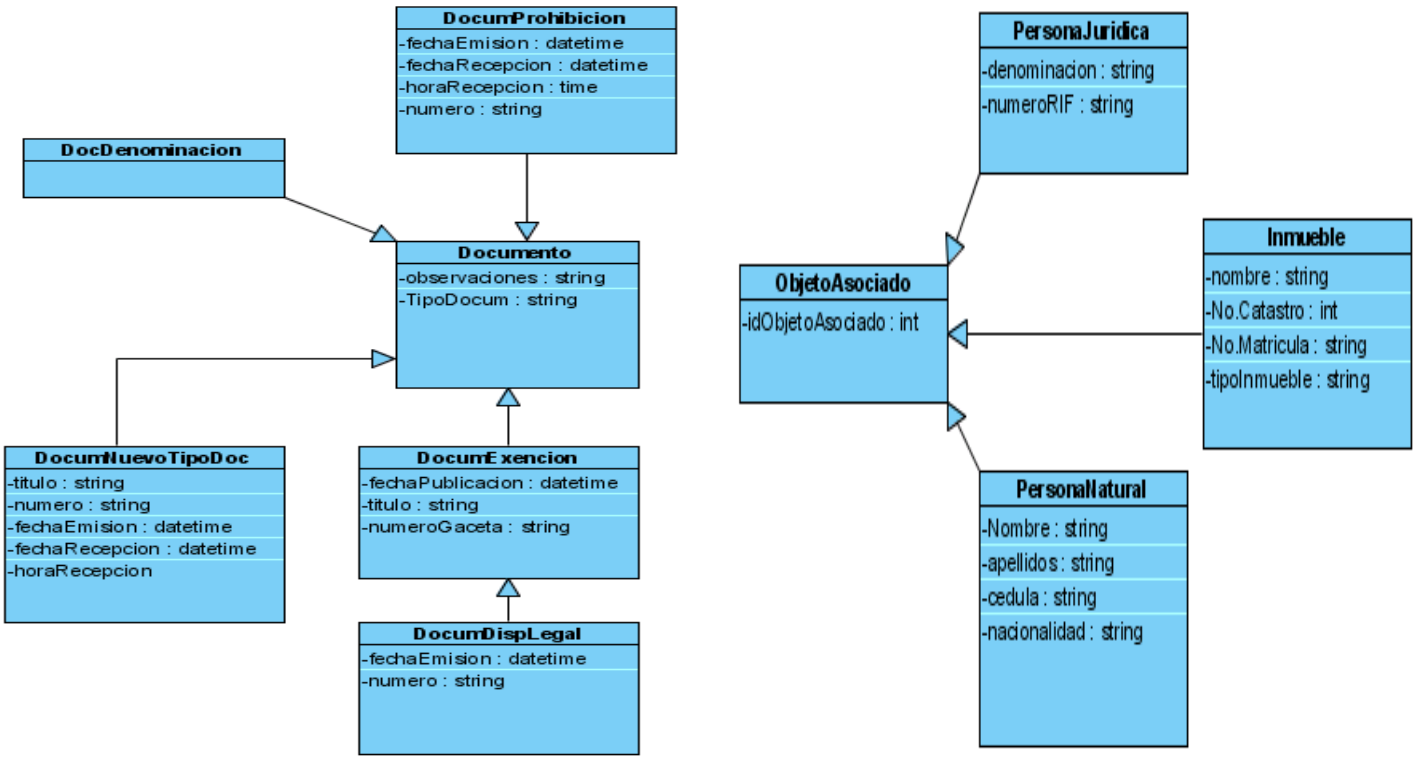

Figura 17 Representación del APH para el diseño propuesto

Clase	Clase padre	Niveles del árbol de herencia
Documento		1
DocDenominacion	Documento	2
DocumNuevoTipoDoc	Documento	2
DocumProhibicion	Documento	2
DocumExencion	Documento	2
DocumDispLegal	DocumExencion	3
ObjetoAsociado		1
PersonaNatural	ObjetoAsociado	2
Inmueble	ObjetoAsociado	2
PersonaJuridica	ObjetoAsociado	2

Tabla 6 Clases y niveles de herencia. Métrica Árbol en profundidad de herencia

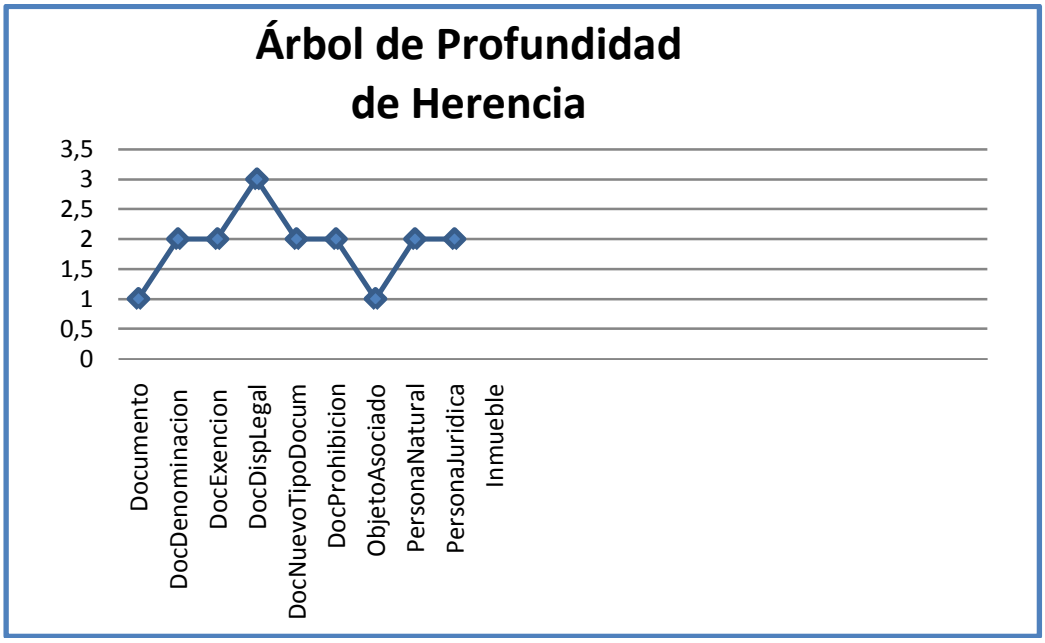

Figura 18 Representación gráfica de los resultados obtenidos al aplicar la métrica PH

3.3.2 Número de Descendientes (NDD)

A medida que el número de descendientes crece la reutilización se incrementa, pero además cuando el NDD crece, la abstracción representada por la clase predecesora puede diluirse. Esto significa que existe la posibilidad de que algunos descendientes no sean miembros realmente apropiados de la clase predecesora. A medida que el NDD crece, la cantidad de pruebas se incrementará también. (35)

	Categoría	Criterio
Reutilización	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoría	Criterio
Abstracción	Indefinida	> 5
	Afectada	Entre 2 y 5
	Definida	<= 2

	Categoría	Criterio
Cohesión	Baja	> 5
	Media	Entre 2 y 5
	Alta	<= 2

	Categoría	Criterio
Cantidad de Pruebas	Baja	<= 2
	Media	Entre 2 y 5
	Alta	> 5

Tabla 7 Umbrales de evaluación de las clases aplicando la métrica NDD

Esta métrica arrojó como resultado que el número máximo de descendientes de una clase es 4 y el promedio general es de 2,67. A continuación se muestra una tabla con los resultados arrojados por esta métrica.

Clase	Clase padre	NDD	Reutilización	Abstracción de la clase base	Cohesión	Cantidad de Pruebas
Documento	Documento	4	Alta	Afectada	Media	Media
DocumExencion	Documento	1	Media	Definida	Alta	Baja
ObjetoAsociado	ObjetoAsociado	3	Alta	Afectada	Media	Media

Tabla 8 Resultado del NDD para el diseño propuesto

3.3.3 Tamaño de Clases (TC)

El tamaño general de una clase puede medirse determinando las siguientes medidas:

El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase y el número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.

Los valores grandes para esta métrica, indican que la clase debe tener bastante responsabilidad. Esto reducirá la reutilización de esta clase y complicará la implementación y las pruebas. Se pueden calcular los promedios para el número de atributos y operaciones de clase. Cuando menor sea el valor del

promedio para el tamaño será más posible que las clases dentro del sistema puedan ser reutilizadas. Las medidas o umbrales para los parámetros de calidad han sido una polémica a nivel mundial en el diseño de sistemas.

El tamaño general de una clase se puede determinar empleando medidas para saber el número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase así como encontrando el número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase. Si existen valores grandes de TC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reusabilidad de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

No.	Clase	No. Atributos	No. Operaciones	Responsabilidad	Complejidad	Reutilización
1.	RegInstJur	12	12	Alta	Alta	Baja
2.	Documento	9	9	Media	Media	Media
3.	DocDenominacion	4	4	Baja	Baja	Alta
4.	DocProhibicion	7	7	Media	Media	Media
5.	DocExencion	5	5	Baja	Baja	Alta
6.	DocDisposicionLegal	7	7	Media	Media	Media
7.	DocNuevoTipoDoc	7	7	Media	Media	Media
8.	Organo	2	2	Baja	Baja	Alta
9.	Compañía	6	6	Media	Media	Media
10.	Expediente	4	4	Baja	Baja	Alta
11.	ObjetoAsociado	2	2	Baja	Baja	Alta
12.	PersonaJuridica	4	4	Baja	Baja	Alta
13.	PersonaNatural	6	6	Media	Media	Media
14.	Inmueble	6	6	Media	Media	Media
15.	Medida	2	2	Baja	Baja	Alta

16.	Municipio	3	3	Baja	Baja	Alta
17.	Parroquia	5	5	Baja	Baja	Alta
18.	CategInsJur	2	2	Baja	Baja	Alta
19.	EstadoActivacionInsJur	3	3	Baja	Baja	Alta
20.	accDocumento	0	14	Alta	Alta	Baja
21.	ccDocumento	2	8	Media	Media	Media
22.	ccGenerarReportes	0	5	Baja	Baja	Alta
23.	ccRegInstJuridica	1	4	Baja	Baja	Alta
24.	accRegInstJuridica	0	4	Baja	Baja	Alta
25.	ccImportFile	1	1	Baja	Baja	Alta
26.	accImportFile	0	1	Baja	Baja	Alta
27.	ccGenerarFichero	0	3	Baja	Baja	Alta
28.	accGenerarFichero	0	2	Baja	Baja	Alta
29.	Conexión	2	3	Baja	Baja	Alta

Tabla 9 Representación de las clases de la solución propuesta

A continuación se representan los parámetros de calidad evaluados con esta métrica.

	Categoría	Cantidad de clases	Criterio
Responsabilidad	Baja	19	< =Prom.
	Media	8	Entre Prom. y 2* Prom.
	Alta	2	> 2* Prom.

Tabla 10 Resultado de la aplicación de la métrica TC para el parámetro Responsabilidad.

	Categoría	Cantidad de clases	Criterio
Complejidad implementación	Baja	19	< =Prom.
	Media	8	Entre Prom. y 2* Prom.
	Alta	2	> 2* Prom.

Tabla 11 Resultado de la aplicación de la métrica TC para el parámetro Complejidad en la implementacion.

	Categoría	Cantidad de clases	Criterio
Reutilización	Baja	2	> 2*Prom.
	Media	8	Entre Prom. y 2* Prom.
	Alta	19	<= Prom.

Tabla 12 Resultado de la aplicación de la métrica TC para el parámetro Reutilización.

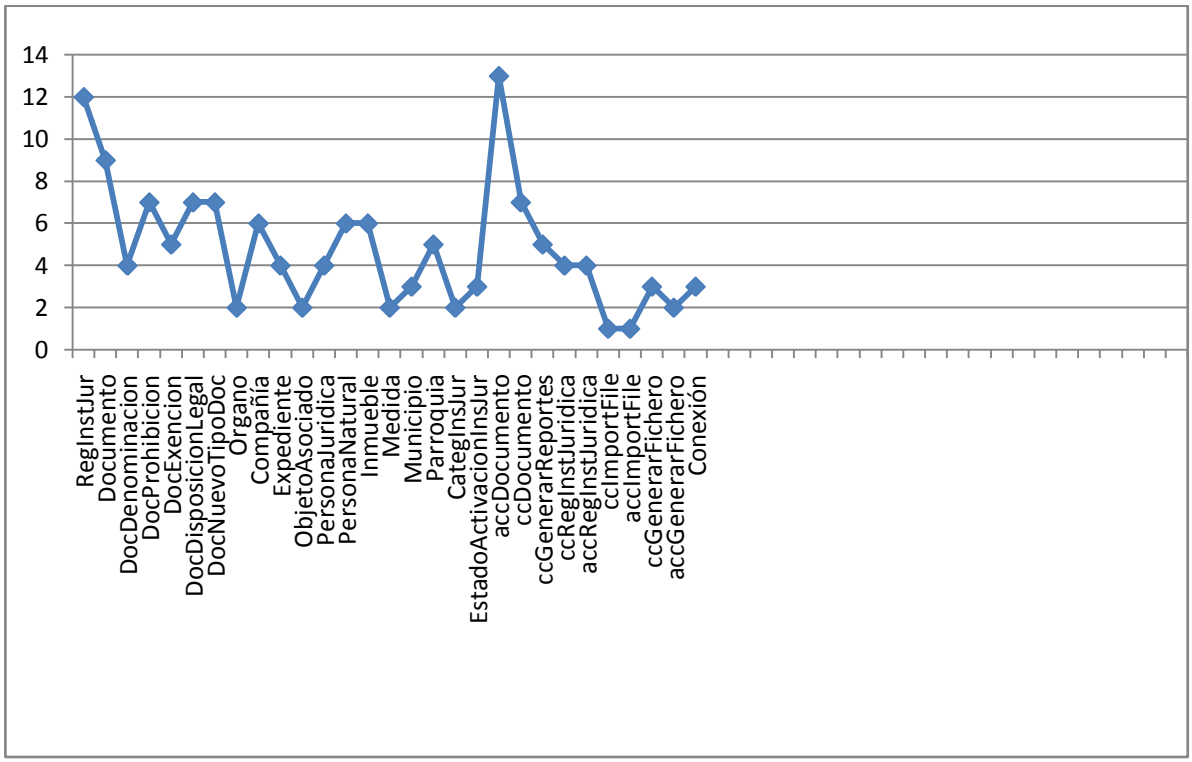

Figura 19 Cantidad de Operaciones por clase

Criterio	Cantidad de clases	Promedio
Entre 1 y 5 operaciones	19	65,51724138
Entre 6 y 10 operaciones	8	27,5862069
Entre 11 y 15 operaciones	2	6,896551724
Total	29	100

Tabla 13 Clases según cantidad de operaciones

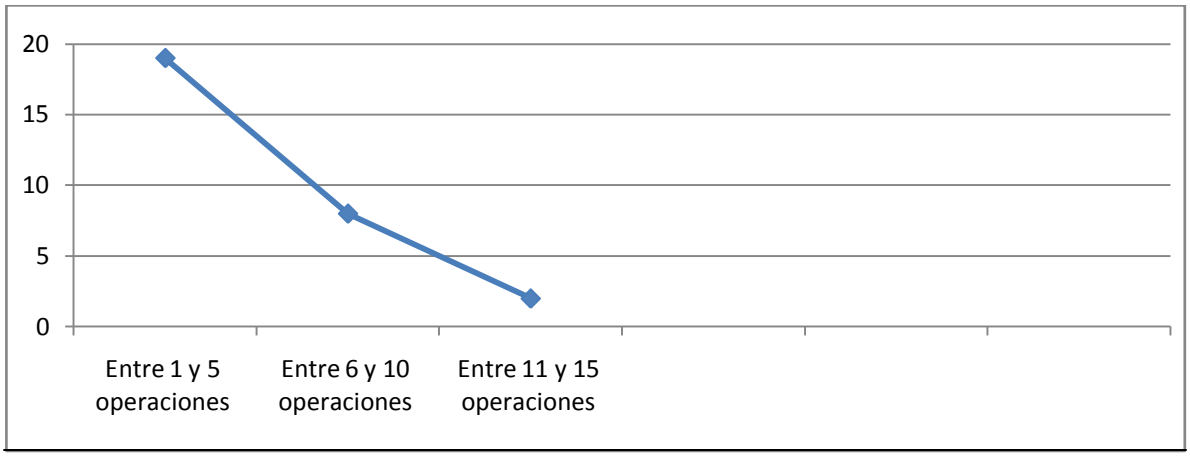

Figura 20 Cantidad de clases por rango de operaciones

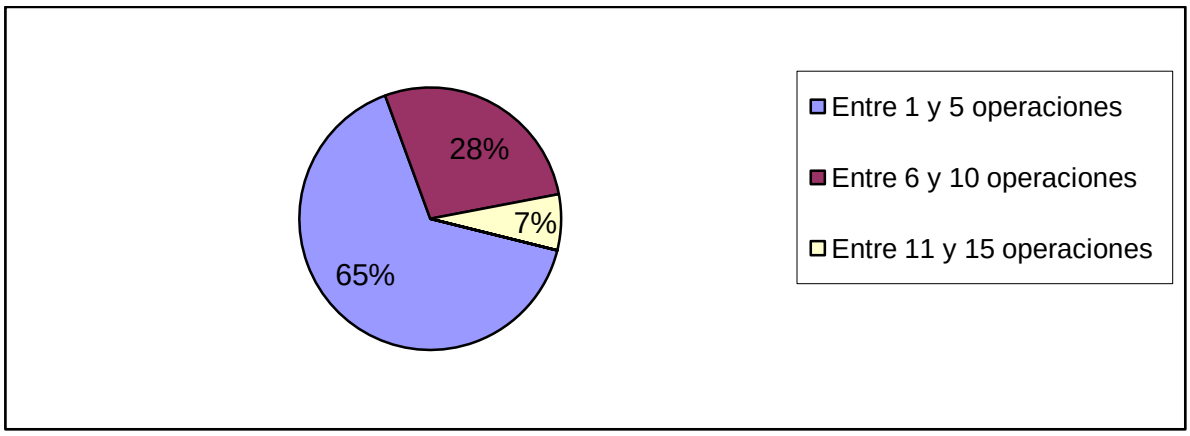

Figura 21 Porciento de clases según rango de operaciones

Teniendo en cuenta las medidas o umbrales de referencia en lo que respecta al número de operaciones y/o atributos de las clases se establece que un tamaño de clase pequeño es aquel que tiene un valor menor o igual que 20. Un tamaño de clase medio es aquel cuyos valores exceden a 20 y son menores o incluyen a 30 y un tamaño de clase grande es aquel que es mayor que este último valor (30). Después de aplicar esta métrica al diseño de la solución propuesta se llegó a las siguientes conclusiones.

La solución cuenta con un total de 29 clases, con un promedio de cantidad de atributos de 3.52 y un promedio de cantidad de operaciones de 4.79.

Los valores de tamaño quedan distribuidos de la siguiente manera:

Umbral	Tamaño	Cantidad de clases
Pequeño	<=20	28
Medio	>20 y <=30	1
Grande	>30	0

Tabla 14 Distribución de valores de tamaño de las clases

Como se puede observar en las tablas que muestran los resultados, de 29 clases que contiene la solución no hay clases de tamaño grande, solo una de tamaño mediano y el resto son clases pequeñas lo cual brinda un resultado positivo según los parámetros de calidad propuesto para la métrica tratada en este sub-epígrafe.

3.3.4 Número de Operaciones Redefinidas para una Sub-Clase (NOR)

Existen casos en que una subclase reemplaza una operación heredada de su superclase por una versión especializada para su propio uso. A esto se le llama redefinición. Los valores grandes para el NOR, generalmente indica un problema en el diseño, o sea si el NOR es grande el diseñador ha violado la abstracción representada por la superclase. Esto provoca una débil jerarquía de clases y un software orientado a objetos, que puede ser difícil de probar y modificar (35).

A partir de los datos obtenidos después de aplicarle al sistema la métrica NOR se obtuvo que de 29 clases que tiene el sistema ninguna subclase reemplaza operaciones definidas en las superclases.

Luego de realizar las pruebas y valorar los resultados que arroja la métrica en cuestión podemos arribar a la conclusión de que existe una adecuada jerarquía y el software puede ser fácil de probar y modificar sin afectar tanto el tiempo que requieran los cambios.

3.4 Matriz de cubrimiento de los indicadores de calidad

La matriz de cubrimiento o matriz de inferencia de indicadores de calidad es el resumen de los resultados obtenidos al aplicar las métricas mencionadas en el epígrafe anterior. Esta matriz es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño de la

solución propuesta. La misma permite conocer si el resultado obtenido de la relación atributo/métricas es positivo o negativo.

Llevando estos resultados a una escalabilidad numérica se determinó que si los resultados son positivos tendrán un valor entre 0,5 y 1, si son negativos tendrán un valor entre 0 y 0,5 si no existe relación alguna tomará el valor -1. Los valores fueron tomados evaluando el grado de afectación o no de los parámetros de calidad en las métricas aplicadas.

En la métrica NDD algunos parámetros fueron medios o afectados, a partir de ahí se valoró, teniendo en cuenta el promedio, si se acercaba más al límite inferior o superior, los valores siempre se acercaron al límite inferior, de modo que de manera general, los valores siempre estuvieron por encima de 0,5 demostrando que los parámetros de calidad tuvieron un impacto positivo en el diseño.

Una vez completado los datos de dicha relación se determina el promedio de los valores obtenidos de la relación atributo/métrica (solo se toman en consideración las que arrojan un resultado distinto de -1). Este valor representa el impacto que tiene cada atributo en el diseño de la solución determinando si su impacto fue bueno, regular o malo. Al desarrollar este procedimiento con los resultados que se obtuvieron una vez se aplicaron las métricas en el diseño propuesto se obtuvo la matriz de cubrimiento que se representa a continuación (41)

Calificativo	Valor
Positivo	>0.5 y <=1
Negativo	>0 y <=0.5
Nulo	-1

Figura 22 Rangos de evaluación de los parámetros de calidad

Calificativo	Rango
Bueno	>0.7
Regular	>0.4 y <=0.7
Malo	<0.4

Figura 23 Clasificación de los parámetros de calidad evaluados en el diseño

Atributos/Métricas	TC	APH	NDD	NOR	Promedio
Responsabilidad	1	-1	-1	-1	**1**
Complejidad del diseño	-1	1	-1	-1	**1**
Complejidad de implementación	1	1	-1	-1	**1**
Reutilización	1	-1	1	-1	**1**
Complejidad del mantenimiento	-1	1	-1	1	**1**
Cantidad de pruebas	-1	-1	0,7	1	**0,9**
Nivel de cohesión	-1	-1	0,7	-1	**0,7**
Abstracción del diseño	-1	-1	0,7	1	**0,9**

Tabla 15 Matriz de cubrimiento para los parámetros de calidad evaluados con las métricas aplicadas al diseño propuesto

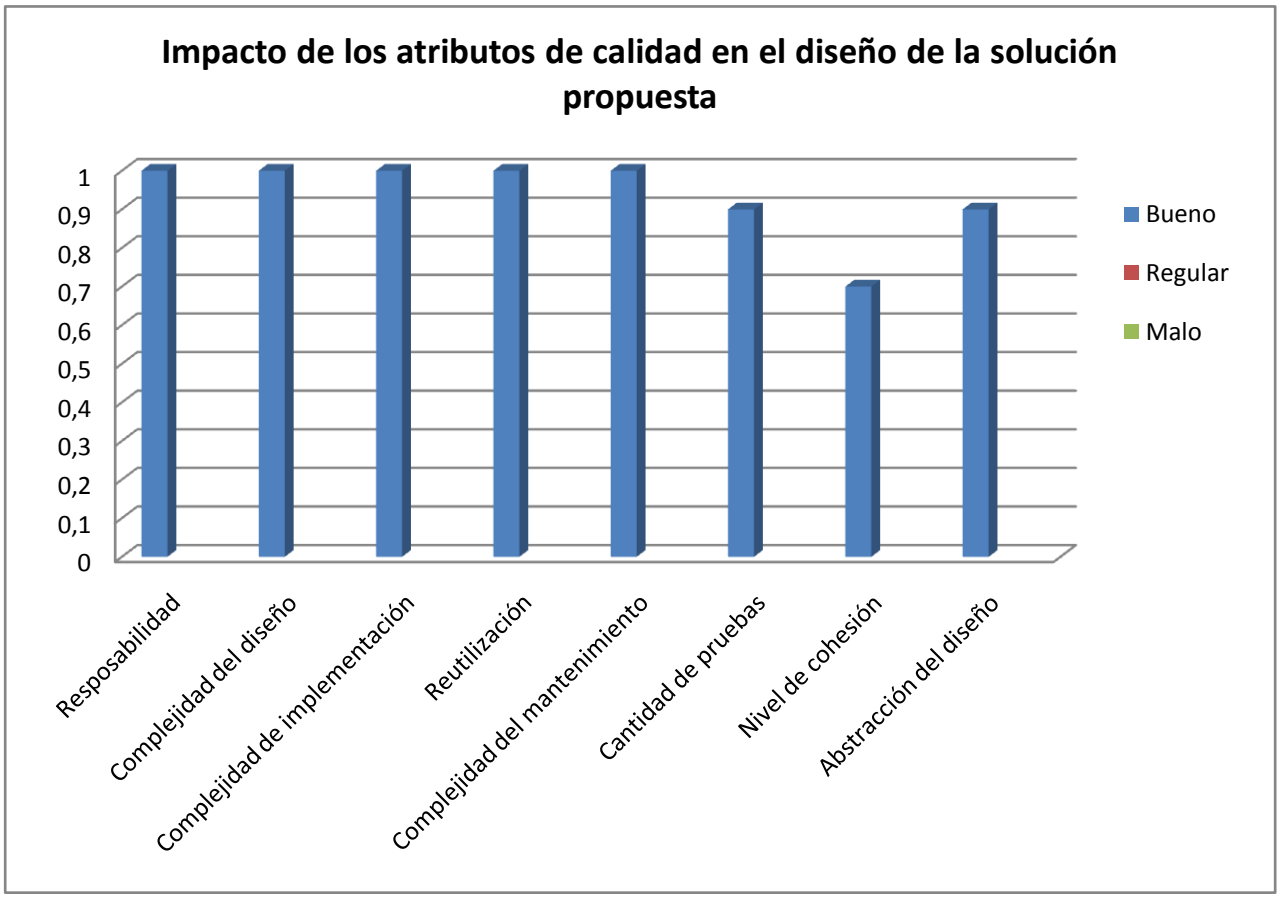

Figura 24 Impacto de los atributos de calidad en el diseño de la solución propuesta

3.5 Conclusiones

La mayoría de las estrategias de medida, como por ejemplo los atributos directamente mensurables denominados métricas de calidad, tienen por objetivo evaluar las diferentes características de calidad del software, tales como la fiabilidad, la facilidad de uso, la facilidad de mantenimiento y la robustez. Estas permiten definir unos atributos en términos de otros más fáciles de medir. Para esto existen umbrales y guías se detectan cuáles clases o métodos se encuentran dentro de los valores medios. El hecho de que una clase no se encuentre con valores muy altos significa que no existen abstracciones mal concebidas y implementaciones no tendrán problemas en el futuro.

En el presente capítulo se analizaron de manera general las características de las métricas para validar el diseño de sistemas informáticos. Se aplicaron algunas de las métricas de las series CK y LK, unas de las más conocidas internacionalmente.

Así la métrica APH arrojó que el diseño no es complejo, existe un bajo acoplamiento entre las clases y no es difícil su mantenimiento.

La aplicación de la métrica NDD mostró que el diseño permite la reutilización y mantiene la abstracción representada por la clase predecesora.

La métrica TC arrojó como resultado que las clases tienen poca responsabilidad, que son de poca complejidad para la implementación y proporcionan una alta reutilización.

Por último la métrica NOR demostró que el diseño es fácil de modificar y probar y que las clases poseen una jerarquía adecuada.

CONCLUSIONES GENERALES

Se realizó el estudio de los conceptos relacionados con la investigación lo que permitió comprender el negocio para el cual está destinado el sistema a implementar. Además se analizó la convergencia que éste puede tener en el desarrollo del Gobierno Electrónico en Venezuela.

Se desarrolló el Modelo de Análisis que sirvió como primera aproximación al Modelo de Diseño. A través de los diferentes artefactos desarrollados en el modelo de análisis se pudo comprender de una forma más precisa los requisitos relacionados con los datos documentales jurídicos y con las características de las instituciones para las cuales está destinado el sistema.

Se desarrolló el diseño de la solución lo que permitió refinar el análisis y garantizar una guía para la implementación de la misma. Los diagramas de clases, la arquitectura, el diagrama de despliegue y el resto de los diagramas desarrollados permitieron determinar el comportamiento estático y dinámico del sistema, así como las relaciones físicas entre los componentes de hardware y software en el sistema final.

Se aplicaron métricas para evaluar el diseño lo que permitió constatar que éste presenta una adecuada jerarquía, no es complejo para la implementación, es de fácil mantenimiento y es reutilizable en un gran porciento.

RECOMENDACIONES

Se recomienda para trabajos futuros:

Determinar los casos de uso no críticos que harán posible que el sistema tenga mayores funcionalidades.

Implementar la solución propuesta basándose en el análisis y el diseño desarrollados.

REFERENCIA BIBLIOGRÁFICA

1. **Ríos Estavillo, Juan José.** *"Informática Jurídica". Primera Edición. Instituto de Investigaciones Jurídicas de la UNAM, Pág . 45.* México, DF : s.n., 1997.

2. **Téllez, Julio.** *"Derecho Informático.* México : Editorial McGraw Hill, 1996. Pág. 26. 2º Edición.

3. INFORMÁTICA JURÍDICA. [En línea] [Citado el: 27 de febrero de 2007.] http://www.gratisweb.com/gmontoya/archivos/dere01.htm.

4. **Fernández, Yarina Amoroso.** LA INFORMATICA JURIDICA QUE NECESITAMOS. [En línea] [Citado el: 27 de febrero de 2009.] http://www.congreso-info.cu/UserFiles/File/Info/Info97/Ponencias/191.pdf.

5. **Martin, Diego José Martínez.** EL SISTEMA COMUNITARIO DE INFORMÁTICA JURÍDICA «CELEX»Y LA EXPERIENCIA «INDILEX»DEL BOLETÍN OFICIAL DEL ESTADO. [En línea] [Citado el: 20 de febrero de 2009.]

6. **Borja, Dra. Anabel.** Organización del conocimiento para la traducción jurídica a través de sistemas expertos basados en el concepto de género textual. [En línea] [Citado el: 24 de febrero de 2009.] http://www.gentt.uji.es/Publicacions/Borja_Ontolog.pdf.

7. **Delpiazzo, Carlos E.** DESAFIOS JURIDICOS RELACIONADOS AL GOBIERNO ELECTRÓNICO Con especial referencia a la Administración electrónica. [En línea] [Citado el: 25 de febrero de 2009.] http://www.lacnic.net/documentos/egov/egov-desafios-juridicos.pdf.

8. CARTA IBEROAMERICANA DE GOBIERNO ELECTRÓNICO. *Aprobada por la IX Conferencia Iberoamericana de Ministros de Administración Pública y Reforma del Estado.* [En línea] 31 de mayo y 1° de junio de 2007. [Citado el: 25 de febrero de 2009.]

9. **López, Alien Rodríguez.** Diseño e Implementación de la solución informática para la Gestión de las Negativas Registrales en la Dirección General de Registros y Notarías de la República Bolivariana de Venezuela. *Trabajo de Diploma para optar por el título de Ingeniero Informático.* Caracas-Venezuela : s.n., 2008.

10. **Leyet Fernández, Osmar y Rodríguez Lorenzo, Iosmel.** Desarrollo de una herramienta generadora de ficheros de mapeo para la persistencia de objetos relacionales basada en NHibernate. . La Habana : s.n., 2008.

11. **Lovelle, Juan Manuel Cueva.** Introducción a UML Lenguaje para modelar objetos. [En línea] [Citado el: 21 de febrero de 2009.]

12. **Roberth G. Figueroa, Camilo J. Solís y Armando A. Cabrera.** METODOLOGÍAS TRADICIONALES VS METODOLOGÍAS ÁGILES. [En línea] [Citado el: 24 de febrero de 2009.]

13. Agil Unified Process . [En línea] http://www.ambysoft.com/unifiedprocess/agileUP.html.

14. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Person Education S.A, 2000.

15. **Sommerville, I.** *Ingeniería de Software.* s.l. : Pearson Educación, 2002.

16. **Pressman, Roger.** *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw Hill, 2001.

17. **Martínez, Alejandro y Martínez, Raúl.** *Guía a Rational Unified Process – Universidad de Castilla la Mancha.*

18. **Larman, Craig.** UML y patrones. Introducción al análisis y diseño orientado a objetos. [En línea] [Citado el: 14 de febrero de 2009.]

19. Rational Unified Process. Best Practices for Software Development Teams. [En línea] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

20. ¿Qué es Scrum? [En línea] [Citado el: 2 de marzo de 2009.] http://www.baufest.com/spanish/scrum/scrumconference2006/Que_es_scrum.pdf.

21. **Sanchez, María A. Mendoza.** Metodologías De Desarrollo De Software. [En línea] [Citado el: 21 de febrero de 2009.] http://www.willydev.net/InsiteCreation/v1.0/descargas/cualmetodologia.pdf.

22. **Informática, Instituto Nacional de Estadística e.** Herramientas Case. [En línea] Noviembre de 1999. [Citado el: 21 de febrero de 2009.] http://informatica.gonzalonazareno.org/file.php/8/case.pdf.

23. Free download manager. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

24. Visual Paradigm, build quality applications faster, better and cheaper. [En línea] [Citado el: 21 de febrero de 2009.] http://www.visual-paradigm.com/product/vpuml/.

25. Rational Rose Technical Developer. [En línea] [Citado el: 21 de febrero de 2009.] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=H515804V91308R94.

26. Grupo de Soluciones GSInnova. [En línea] [Citado el: 21 de febrero de 2009.] http://www.rational.com.ar/herramientas/roseenterprise.html.

27. **Lagos, Pedro Salcedo.** CommonKADS y el Lenguaje de Modelado Unificado – UML. [En línea] [Citado el: 21 de febrero de 2009.]

28. **L. Chung, B.A. Nixon, E. Yu, J.** *Mylopoulos Non-Functional Requirements in Software Engineering.* s.l. : Kluwer Academic Publishers, 2000.

29. *Z.151 (GRL). International Telecommunication Union (ITU).* September 2003.

30. OMG, Business Process Modeling Notation Specification, OMG Final Version Adopted, . [En línea] Febrero de 2006. http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf..

31. **Fidel Gil, Javier Albrigo, Javier Do Rosario.** SISTEMAS DE GESTIÓN DE BASE DE DATOS . *SGBD / DBMS.* [En línea] 14 de febrero de 2005. [Citado el: 20 de febrero de 2009.] http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones/1/SGBD.pdf.

32. **Espinoza, Humberto.** PostgreSQL. Una Alternativa de DBMS Open Source. [En línea] [Citado el: 20 de febrero de 2009.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

33. Enrique Toledo alma. *MySQL.* [En línea] [Citado el: 20 de febrero de 2009.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf.

34. **Prieto, Maribel Silva Muñoz y Sándor Rodríguez.** Diseño e Implementación de un sistema informático integrado para la Gestión de Compras de Bienes y Contratación de Servicios en los Registros y las Notarías de la República Bolivariana de Venezuela. *Trabajo de Diploma para optar por el título de Ingeniero Informático.* Junio 2008.

35. **Pressman, Roger S.** *Ingenería de Software. Un enfoque práctico.* s.l. : Quinta Edicion, febrero 2002.

36. **Mutis., Armando Esteban Pacheco Iglesias. Alejandro Casanova.** Diseño e implementación de funcionalidades que se llevan a cabo en los registros mercantiles: solicitudes de expedientes, copias de documentos y sellado de libros. Ciudad de la Habana : s.n., Junio del 2008.

37. **Gornzález, Anaisa Hernández.** Un método para el diseño de la Base de Datos a partir del modelo Orientado a Objeto. [En línea] Abril-Junio de 2004. [Citado el: 15 de Marzo de 2009.] http://redalyc.uaemex.mx/redalyc/pdf/615/61570402.pdf.

38. **Departamento de Informático de la Universidad e Vigo.** Sitio Web del Departamento de Informático de la Universidad e Vigo. *Departamento de Informático de la Universidad e Vigo.* [En línea] [Citado el: 19 de 12 de 2008.] http://www.lsi.uvigo.es/lsi/erosello/imo/articulos/rendimiento.pdf.

39. **Informática, Departamento de Control de la calidad y auditoría.** Departamento de Control de la calidad en los Sistemas. [En línea] [Citado el: 5 de febrero de 2009.] http://bibliodoc.uci.cu/pdf/controldecalidad.pdf.

40. **Arregui, Juan José Olmedilla.** Revisión Sistemática de Métricas de Diseño Orientado a Objetos. [En línea] Septiembre 2005. [Citado el: 7 de Abril de 2009.] http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Olmedilla.pdf.

41. **Ochoa, Raykenler Yzquierdo Herrera y René Lazo.** Trabajo de Diploma "El modelo de diseño del sistema HyperWeb. Módulos de Tratamiento Farmacológico y Configuración". Mayo de 2007.

42. RapidRapp. [En línea] https://rapidrabb.it/.

43. TunaWeb Design Blog. [En línea] http://www.tunawebdesign.com/blog/category/diseno-web/.

GLOSARIO DE TÉRMINOS

Análisis: Flujo de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la captura de requisitos, mediante su refinamiento y estructuración. El objetivo es lograr una compresión más precisa de estos, y obtener una descripción que sea fácil de mantener y que ayude a dar estructura al sistema en su conjunto, incluyendo su arquitectura.

Artefacto: Producto tangible del proyecto, resultado de una o varias actividades sobre el mismo. Pieza de información producida, modificada y utilizada en un proceso.

Calidad de software: La calidad del software es una compleja mezcla de factores que varían a través de diferentes aplicaciones y según los clientes que las pidan. (35)

Carga inicial: Proceso de recopilación inicial de datos de una institución que facilitan el correcto funcionamiento de un sistema informático en ésta.

Clases de análisis: Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. Las clases de análisis se clasifican en: clases interfaz, clases control y clases entidad

Clases de Diseño: Se especifican utilizando la sintaxis del lenguaje de programación elegido, se especifica la visibilidad de los atributos y operaciones: public, protected, private. Se implementan las relaciones (referencias entre objetos) que tienen un significado en la implementación. Se especifican los métodos. Tienen correspondencia directa con los métodos en la implementación. A menudo aparece con un estereotipo que se corresponde con una construcción de lenguaje de programación. Puede realizar interfaces si tiene sentido en el lenguaje de programación, una clase que proporcione una interfaz debe proporcionar métodos que realicen las operaciones de la interfaz, las interfaces constituyen una forma de separar la especificación de funcionalidad de operaciones, de su implementación en término de métodos.

Derecho informático: Conjunto de normas positivas referidas al tratamiento automatizado de la información en sus múltiples aspectos.

Despliegue: Es uno de los flujo de trabajo de ingeniería. En la metodología utilizada se define el despliegue como el plan para la entrega del sistema haciendo que esté disponible para los usuarios,

teniéndose en cuenta actividades de empaque, instalación, asistencia a usuarios para entregar el software a los usuarios finales.

Diagrama de Colaboración: Es uno de los diagramas de interacción. Destaca la organización de los objetos que participan en una interacción.

Diagramas de Interacción: Se utilizan para modelar los aspectos dinámicos de un sistema. Consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se puedan enviar entre ellos.

Diagrama de secuencia: Es uno de los diagramas de interacción. Destaca el ordenamiento temporal de los mensajes.

Diseño: Flujo de trabajo fundamental cuyo propósito principal es el de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución, y que prepara para la implementación y pruebas del sistema.

Documento jurídico: Los documentos jurídicos se dividen, por su autor, en públicos y privados; documentos públicos son aquellos de los que es autor un funcionario público actuando como tal, y, por tanto, dentro de su competencia y con sujeción a las reglas que rigen su función pública. Pueden ser también por su autor, notariales (Escrituras Notariales, judiciales (Sentencias, Autos o cualquier documento que emita un órgano Judicial o de Arbitraje) se reconocen también como Judiciales; administrativo (Resoluciones y documentos emanados por las Administraciones), normativos (Leyes, Decretos Leyes, Reglamentos o cualquier Acto Normativo) , y, por su contenido, se clasifican en Documentos de Derecho público y de Derecho privado; al Derecho civil interesan especialmente los documentos notariales (instrumentos públicos). Son documentos privados todos los demás, incluso los públicos defectuosos por incompetencia del funcionario o por falta de forma.

Flujo de trabajo: Un flujo de trabajo muestra la secuencia de actividades que se desarrollan dentro de uno o varios Casos de Uso, como una pieza de funcionalidad concreta que satisface los requerimientos de un Actor. Los flujos de trabajo pueden ser de ingeniería y de apoyo.

Informática Jurídica: Es la aplicación de la informática en el tratamiento de la información jurídica, las informática jurídica está dividida en dependencia de las áreas donde opera en informática jurídica Documental (técnicas de documentación científica aplicada a la información jurídica: legislación, jurisprudencia y doctrinas), de Gestión (automatización de procedimientos en las oficinas de los

operadores jurídicos. Provee herramientas auxiliares para las labores que se desarrollan en tales despachos) o Meta documental (también conocida como Decisional. Comprende el campo de la inteligencia artificial, de los sistemas expertos jurídicos).

Instituciones jurídicas: Dentro de la tradición romanista, el concepto de institución aparece vinculado a la práctica de los juristas que impartían la enseñanza del Derecho. El concepto romanista identificaba las instituciones jurídicas con los conjuntos de situaciones, relaciones, actuaciones y reglas que estaban unidos por una cierta homogeneidad funcional en torno a un elemento jurídico dotado de autonomía dentro de la organización. Finalmente, se ha generalizado la caracterización de las instituciones jurídicas como núcleos o figuras jurídicas estables que vienen delimitadas por el conjunto de normas que regulan el modo en que han de ser realizadas las respectivas relaciones. La tradición institucionalista cambió esta perspectiva de análisis, marginando el estricto enfoque jurídico normativo y adhiriéndose al punto de vista sociológico. Así, el elemento definitorio de las instituciones jurídicas es la propia agrupación social en la que se integran las personalidades y los intereses de los diferentes miembros. Una institución jurídica es, pues, para la doctrina institucionalista, una realidad o ente social complejo que está dotado de organización interna, de modo que la actividad de todos sus miembros se realiza según el orden exigido por la idea directriz que los aglutina.

Métricas de calidad: Proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. Es decir cómo medir el sistema para que se adapte a los requisitos que pide el cliente.

Métricas de diseño: Las métricas de diseño proporcionan una predicción de la calidad del diseño, y una indicación general de la cantidad de esfuerzo de pruebas necesario para aplicarlo en un sistema Orientado a Objetos.

Procesos: Proviene del latín *processus* y se refiere al conjunto de actividades o eventos que se realizan o suceden alternativa o simultáneamente con un fin determinado.

Proceso de desarrollo de software: Define un conjunto de actividades para transformar los requisitos del cliente en un producto de software.

Sistema informático: Conjunto de partes interrelacionadas, hardware, software y de recursos humanos. Emplea computadoras que utilizan dispositivos programables para capturar, almacenar y procesar datos.

ANEXOS

Descripción detallada de los Casos de Uso (CU) arquitectónicamente significativos.

Caso de Uso:	Importar archivo a la Base de Datos	
Actores:	Administrador	
Resumen:	El caso de uso se inicia cuando el administrador accede al sistema para importar a la base de datos central los datos de las instituciones. Cosiste e importar el archivo generado en las entidades jurídicas y actualizar la Base de Datos del Centro de Datos con esta información. Como resultado se obtiene la actualización de la información existente en la base de datos.	
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con los permisos necesarios.	
Referencias	R5	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1. Selecciona en la interfaz principal la opción importar datos		2. Muestra la ventana de abrir archivo de los datos que serán importados
3. Busca el archivo en el directorio. 4. Selecciona abrir archivo.		5. Verifica que esté en la extensión correcta y que no esté corrupto. 6. Importa los datos a la Base de Datos. 7. Muestra mensaje de confirmación. 8. Finaliza CU.
Flujo Alterno al paso 4 "Archivo Incorrecto"		
Acción del Actor		**Respuesta del Sistema**
		4.a Muestra mensaje de error: "No se pudo importar el archivo. No está en la extensión correcta o está corrupto. 4.b Finaliza el CU.
Pos-condiciones	Quedan los datos de las instituciones guardados en la Base de datos.	

Tabla 16 CU: Importar archivo a la Base de Datos

Caso de Uso:	Generar Reportes por documentos
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario accede al sistema para generar algún reporte sobre las acciones realizadas. Se basa en la generación de algunos reportes que se necesitan tanto en las entidades como en el Centro de Datos. Permite generar reportes sobre los documentos de manera general o especificando una fecha determinada ya sea de emisión, inserción o modificación. Además permite generar reportes de documentos por entidad jurídica y de documentos con errores una vez activadas estas opciones.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • Debe existir al menos un documento en la Base de Datos. • El usuario debe estar autenticado con los permisos necesarios.
Referencias	R6, R7
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario selecciona la opción Reporte de documentos en la interfaz principal.	2. Muestra una interfaz con las opciones de Reporte de documentos por: **Fecha de emisión, fecha de inserción, fecha de modificación** y **Reporte general de documentos.** En caso de que el usuario sea el administrador se le muestra además las opciones **Reporte de documentos por Entidades Jurídicas** y **Reporte de documentos con errores**.
3. Selecciona opción de reportes	4. Busca datos relacionados con el criterio de búsqueda. 5. Genera el reporte generado dependiendo del criterio de búsqueda
6. Selecciona imprimir o salir.	7. Si es imprimir imprime reporte. 8. Finaliza el CU
Pos-condiciones	El sistema muestra o imprime los reportes.

Tabla 17 CU: Generar Reportes por documentos

Caso de Uso:	Eliminar Institución Jurídica
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador entra al sistema para eliminar alguna Institución Jurídica. Termina una vez que se actualice la Base de datos
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con los permisos necesarios.
Referencias	R8; R9; R10
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona la opción Eliminar Institución Jurídica en la interfaz principal.	2 Muestra la interfaz Eliminar institución con los criterios de búsqueda y listado de instituciones
3. Introduce criterios de búsqueda.	4. Muestra resultado de la búsqueda.
5. Selecciona la Institución a eliminar.	6. Muestra los datos de la Institución seleccionada
7. Selecciona Eliminar	8. Actualiza la Base de Datos 9. Muestra mensaje de confirmación 10. Finaliza el Caso de Uso.
Pos-condiciones	Se elimina una institución Jurídica de la BD.

Tabla 18 CU: Eliminar Institución Jurídica

Caso de Uso:	Generar fichero
Actores:	Responsable de la Institución Jurídica
Resumen:	El caso de uso se inicia cuando una vez que la Responsable de la Institución Jurídica selecciona la opción generar fichero en la interfaz principal o al cerrar la aplicación. Consiste en generar un fichero *.XML con la información insertada en las instituciones jurídicas para enviarlo al Centro de Datos y registrar dicha información. El caso de uso termina con el fichero creado.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • Debe existir al menos un documento en la Base de Datos.

Referencias	
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción generar fichero en la interfaz principal.	2. Verifica existencia de datos de la institución 3. Verifica actualizaciones. 4. Genera el fichero. 5. Finaliza el CU.
Flujo Alterno al paso 2 "No hay datos de la institución"	
	2.a Si no se han insertado los datos de la institución Ir al CU Gestionar datos de la institución 2.b Ir al paso 3.
Flujo Alterno al paso 4 "no hay actualizaciones"	
	4.a Muestra mensaje de error. 4.b Muestra interfaz principal. 4.c Finaliza el CU.
Pos-condiciones	El sistema salva el fichero.

Tabla 19 CU: Generar fichero

Caso de Uso:	Gestionar documento por tipo documental
Actores:	Responsable de la Institución Jurídica
Resumen:	El caso de uso se inicia cuando el responsable de la Institución Jurídica selecciona la opción de Gestionar documento por tipo documental. Permite insertar y modificar documentos teniendo en cuenta su tipo documental. Termina cuando queda insertado o modificado un documento de la aplicación.
Pre-condiciones:	El sistema debe estar instalado y ejecutándose correctamente.
Referencias	R11; R12; R13; R14; R15; R16; R17; R18
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. Selecciona en la interfaz principal las opciones insertar o modificar documento.	2. Si selecciona Insertar, ir a la Sección "Insertar Documento". Si Selecciona Modificar, ir a la Sección "Modificar Documento". Si selecciona eliminar, ir a la Sección "Eliminar Documento"
Sección "Insertar Documento"	
Acción del Actor	**Respuesta del Sistema**
	3. Muestra una interfaz para insertar los datos del documento.
4. Introduce los datos del documento y selecciona Aceptar.	5. Valida que estén todos los datos y que estén correctos. 6. Inserta los datos en la Base de Datos. 7. Finaliza el CU.
Flujo Alterno al paso 5 "Datos Incorrectos"	
	5.a Muestra mensaje de error: "Los datos introducidos son incorrectos"
Sección "Modificar Documento"	
Acción del Actor	**Respuesta del Sistema**
	3. Muestra la interfaz de modificar documentos con los criterios de búsqueda de documentos y listado de documentos insertados.
4. Introduce criterios de búsqueda	5. Muestra resultado de la búsqueda.
6. Selecciona el documento a modificar	7. Muestra los datos del documento seleccionado.
8. Modifica los datos y selecciona salvar	9. Actualiza la Base de Datos 10. Muestra mensaje de confirmación 11. Finaliza el Caso de Uso
Flujo Alterno al paso 10 "Selecciona Cancelar"	
	10.a Ir al paso 3. 10.b Finaliza el CU.
Pos-condiciones	1. El sistema queda con los documentos insertados, o modificados.

Tabla 20 CU: Gestionar documento por tipo documental

Caso de Uso:	Buscar Documento en fichero
Actores:	Responsable de la institución Jurídica
Resumen:	El caso de uso se inicia cuando el Responsable de la institución Jurídica selecciona la opción Buscar documento, está relacionado además con los casos de uso: Gestionar documento por tipo documental y de Generar

	Reportes. Consiste en realizar una búsqueda de documentos por parte del responsable de la institución Jurídica, ya sea para modificar algún documento o para generar algún reporte. Muestra un conjunto de documentos que podrán ser seleccionados.
Pre-condiciones:	El sistema debe estar instalado y ejecutándose correctamente.
Referencias	R17
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción buscar documento en la interfaz principal.	2. El sistema muestra una interfaz para buscar documentos. En dependencia del usuario son las opciones de búsqueda: • ID documento. • Fechas de creación. • Rango de fechas
3. Llena los criterios de búsqueda y acepta o rechaza acción.	4. Si acepta el sistema verifica que los datos introducidos sean correctos y que el campo Institución Jurídica (obligatorio en el caso del administrador) no esté vacío 5. Busca Documentos según criterios de búsqueda 6. Muestra Listado de documentos. 7. Finaliza el Caso de Uso.
Flujo Alterno al paso 5 "No se encontraron documentos"	
Acción del Actor	**Respuesta del Sistema**
	5.a El sistema Muestra mensaje de error: "No hay documentos disponibles".
Pos-condiciones	1. El sistema se mantiene con el mismo estado.

Tabla 21 CU: Buscar Documento en fichero

Caso de Uso:	Buscar Documento en BD
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador selecciona la opción Buscar documento en BD. Consiste en realizar una búsqueda directa en la BD de los documentos a partir de un conjunto de criterio. El acceso será a través de consultas a la BD. Finaliza con los documentos mostrados o mensajes de error en caso de que no existan.

Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El administrador debe haber autenticado el acceso a la BD previamente
Referencias	R20, R21
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción buscar documento en BD en la interfaz principal.	2. El sistema muestra una interfaz para buscar documentos. Las búsquedas se pueden hacer por: • Institución Jurídica • ID documento • Fechas de creación • Rango de fechas
3. Llena los criterios de búsqueda y acepta o rechaza acción.	4. Si acepta el sistema abre la conexión a la BD y hace una consulta en dependencia de criterio de búsqueda seleccionado. 5. Muestra Listado de documentos. 6. Finaliza el Caso de Uso.

Flujo Alterno al paso 5 "No se encontraron documentos"	
Acción del Actor	**Respuesta del Sistema**
	5.a El sistema Muestra mensaje de error: "No hay documentos disponibles".

Pos-condiciones	2. El sistema se mantiene con el mismo estado.

Tabla 22 Buscar Documento en BD

Caso de Uso:	Enviar documentos con errores
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador selecciona la opción enviar documentos con errores. Consiste en enviar un correo a las instituciones que han enviado documentos con algún error. Termina una vez que se haya enviado la confirmación.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con los permisos necesarios.
Referencias	R22
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1 Selecciona la opción enviar documentos con errores.	2 Muestra la interfaz documentos con errores 3 Muestra listado de documentos con errores por institución.
4. Selecciona la institución. 5. Selecciona la opción Enviar correo.	6. Busca la dirección de correo de la institución. 7. Envía correo de errores. • Documento sin objeto asociado • Documentos sin imágenes. • Archivo corrupto. 8. Muestra mensaje de confirmación.
Flujo Alterno al paso 3 "No existen documentos con errores"	
	3.a Muestra mensaje de error: No existen documentos con errores.
Pos-condiciones	El sistema queda con los documentos insertados o eliminados en la BD.

Tabla 23 CU: Enviar documentos con errores

Diagrama de clases del análisis

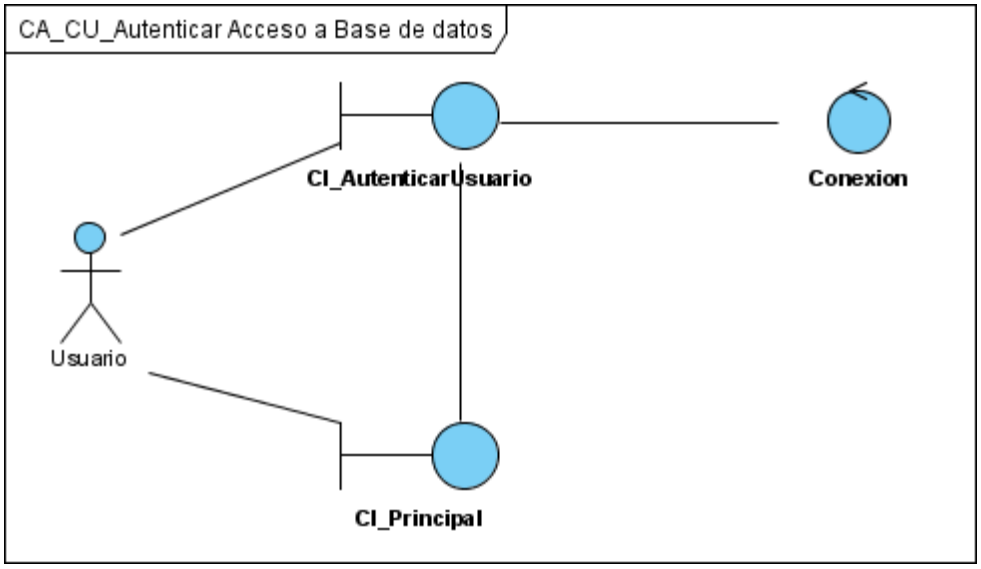

Figura 25 CU: Autenticar Acceso a Base de datos

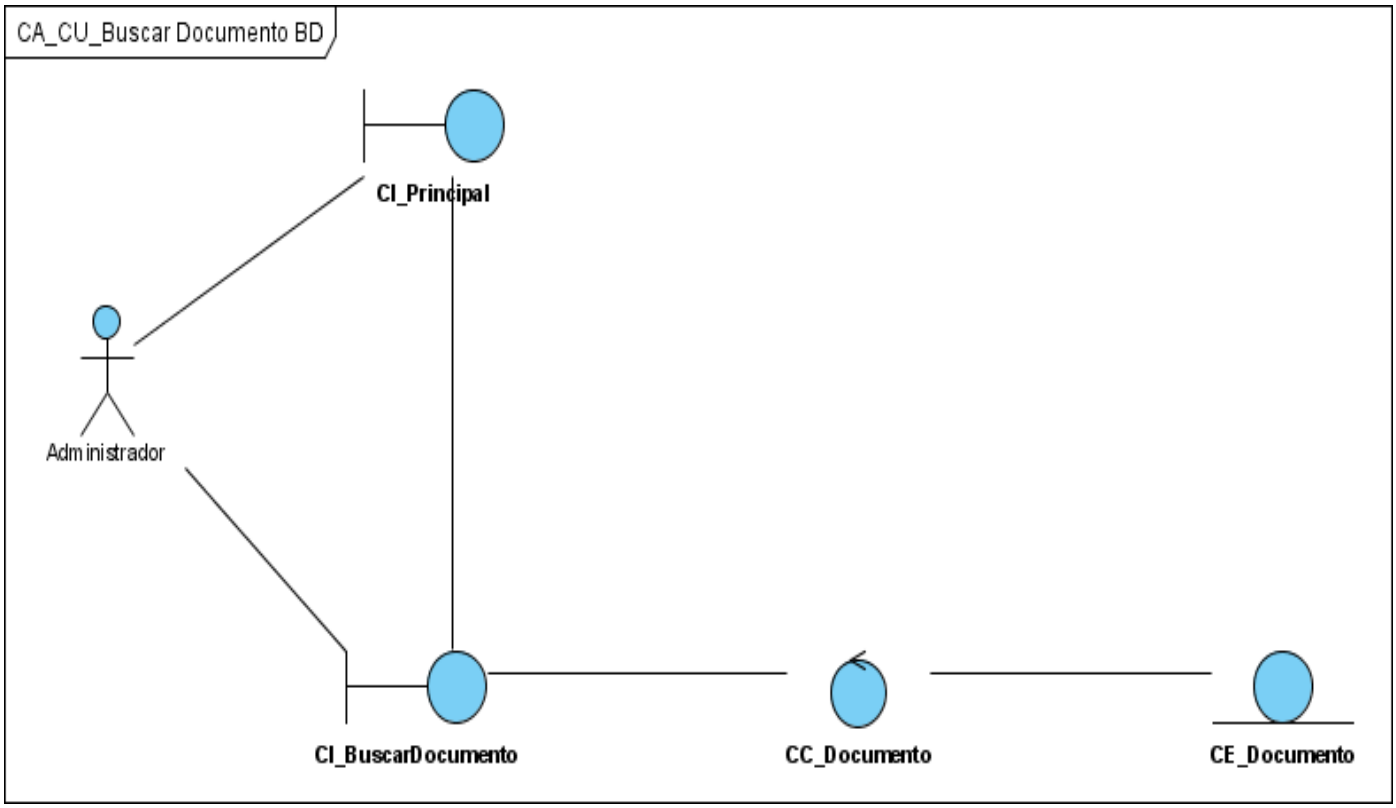

Figura 26 CU: Buscar Documento Fichero

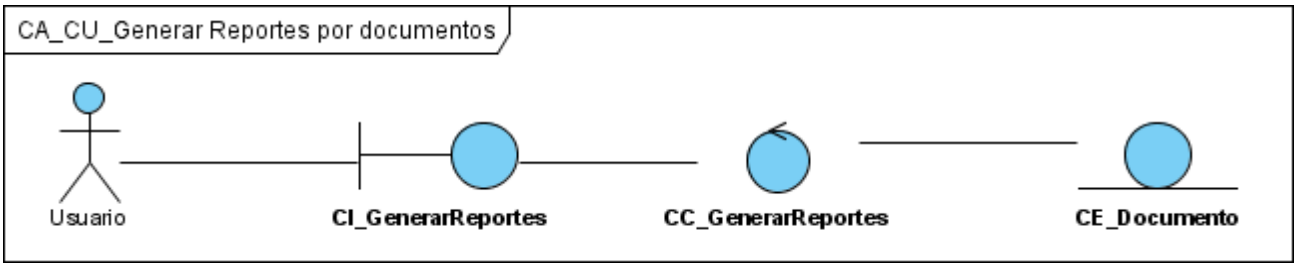

Figura 27 CU: Generar Reportes por documentos

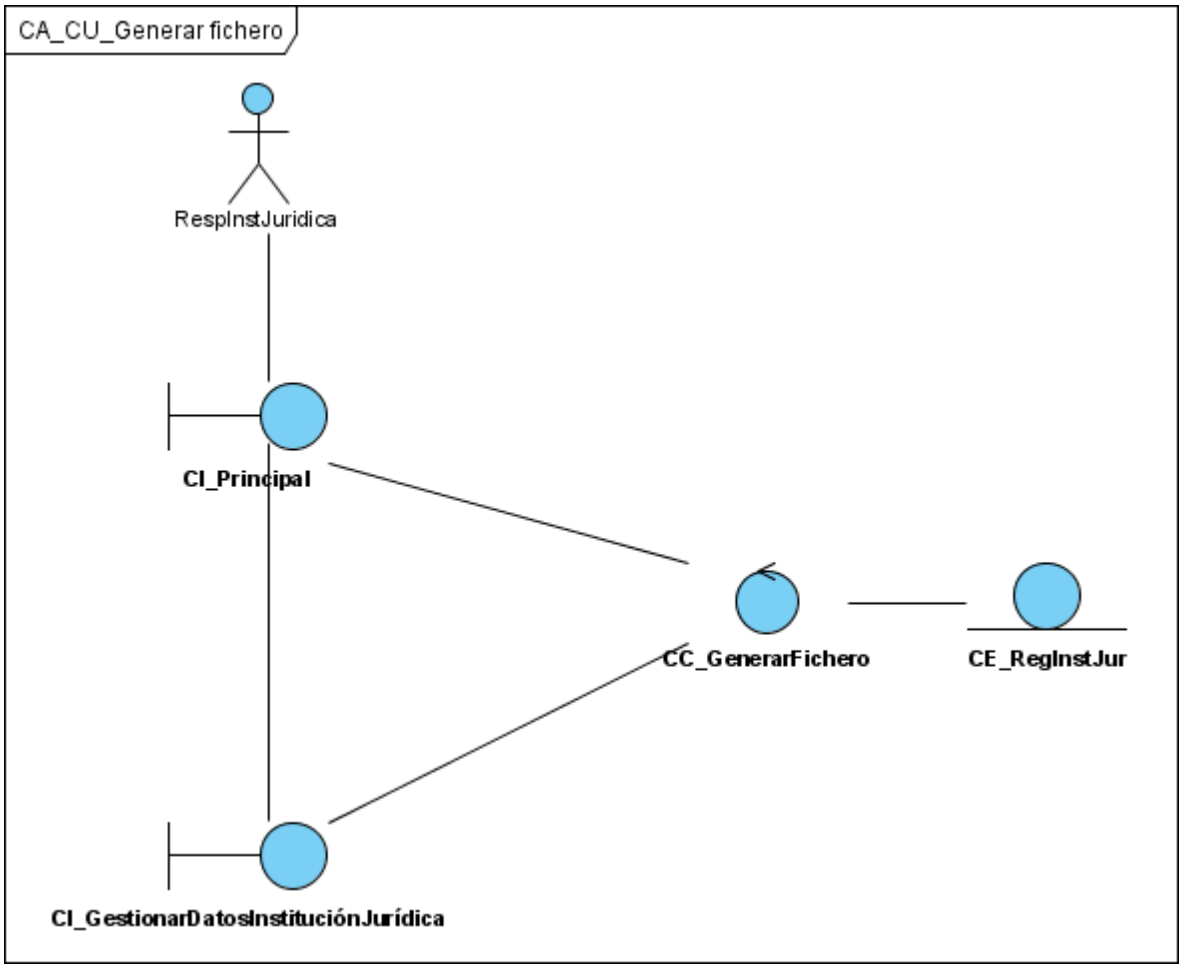

Figura 28 CU: Generar fichero

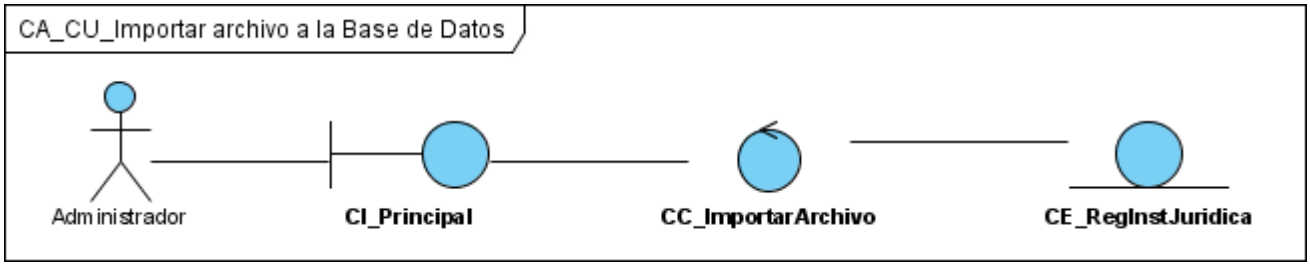

Figura 29: Importar archivo a la Base de Datos

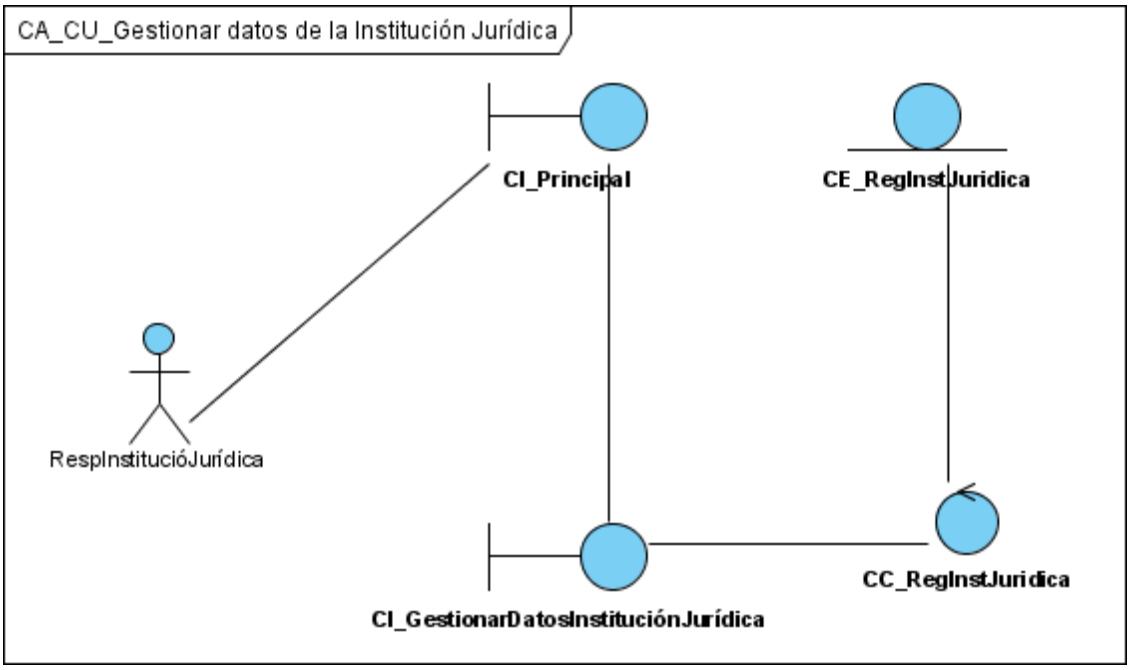

Figura 30 CU: Gestionar datos de la Institución Jurídica

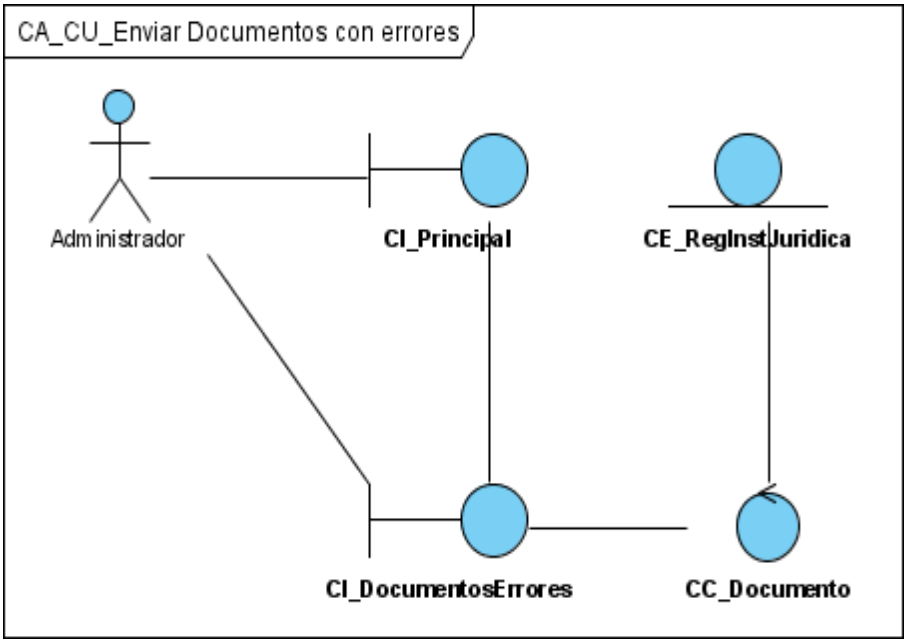

Figura 31 CU: Enviar Documentos con errores

Diagrama de clases de diseño

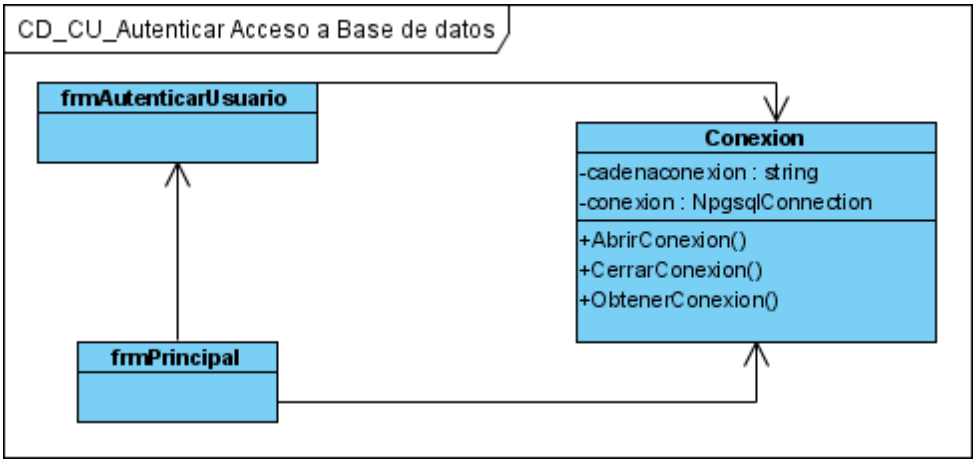

Figura 32 CU: Autenticar Acceso a Base de datos

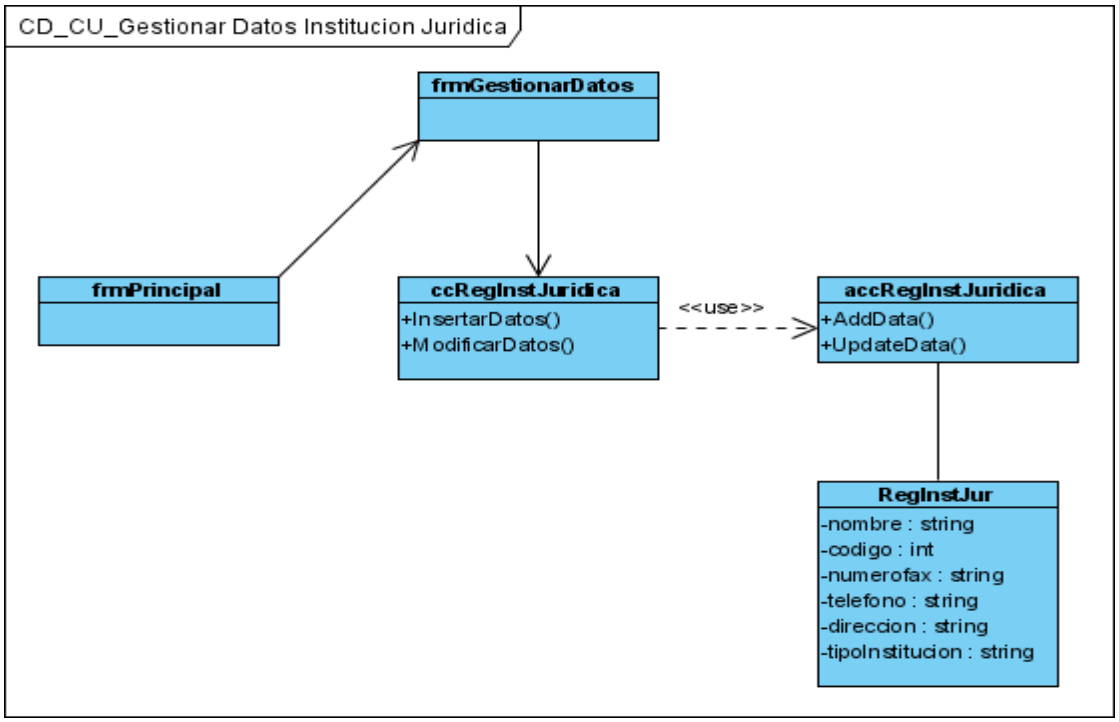

Figura 33 CU: Gestionar Datos Institución Jurídica

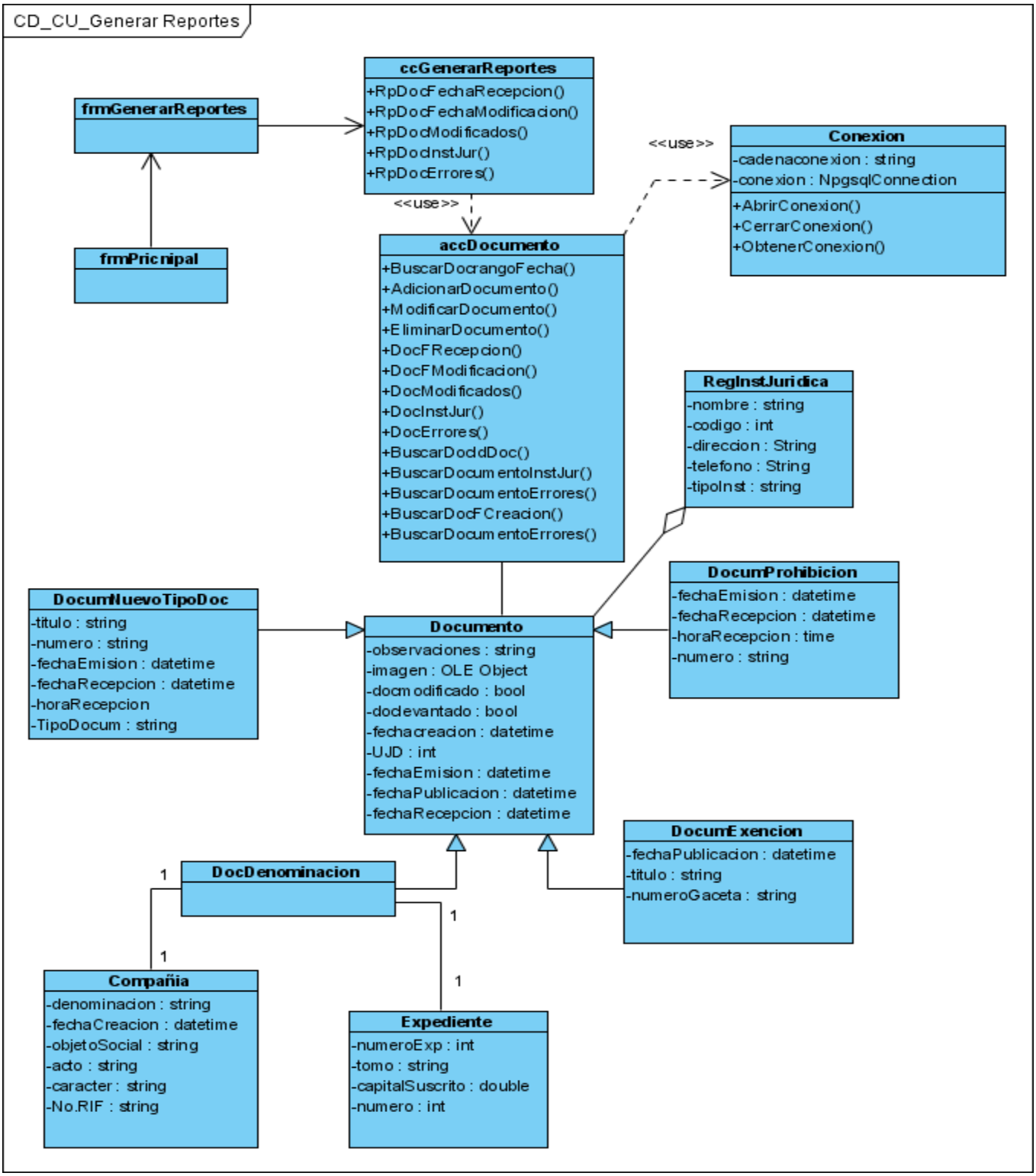

Figura 34 CU: Generar Reportes

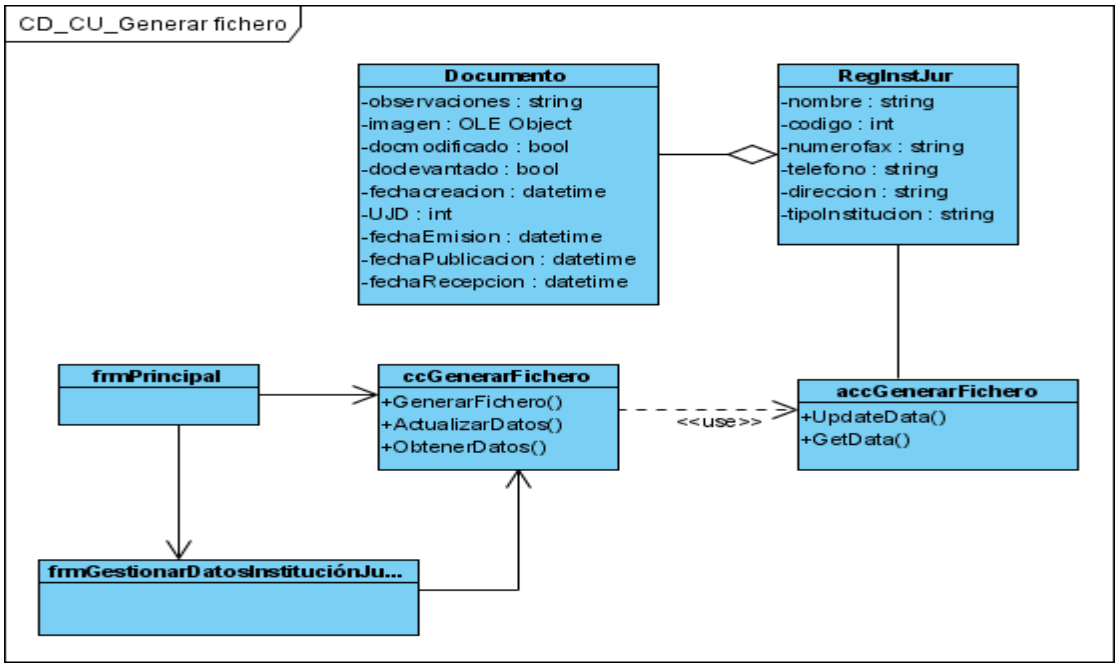

Figura 35 CU: Generar fichero

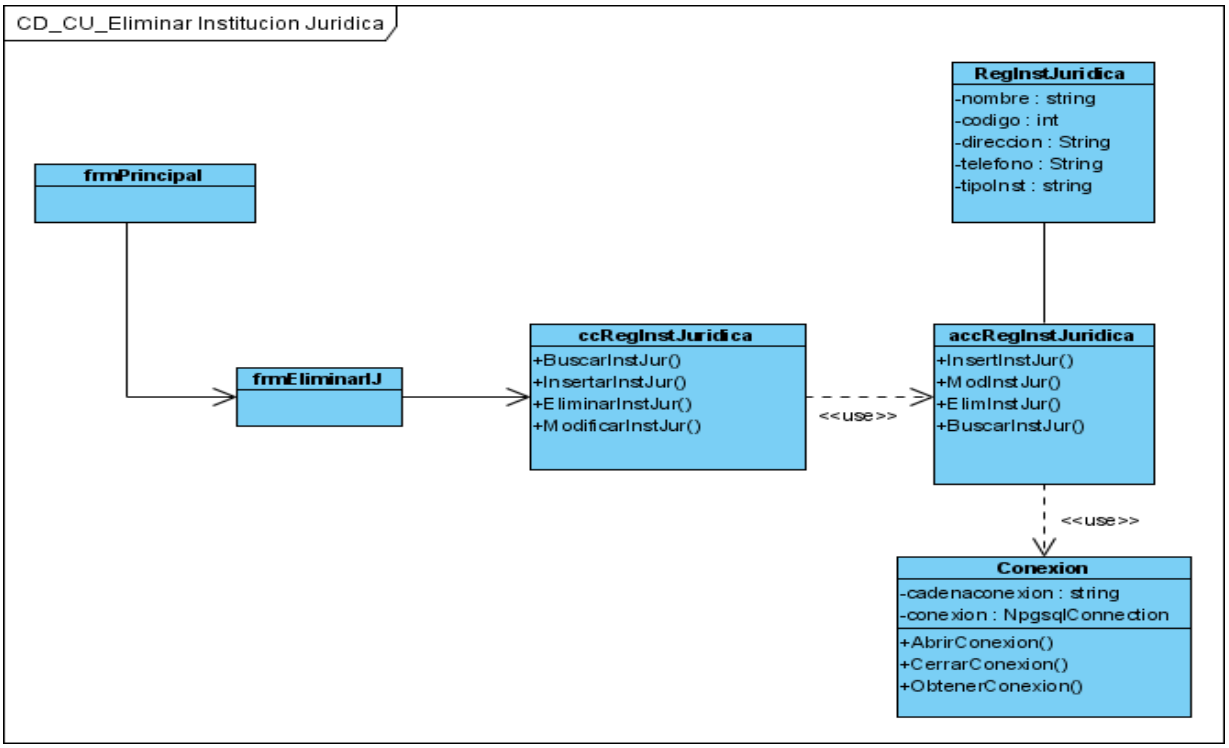

Figura 36 CU: Eliminar Institución Jurídica

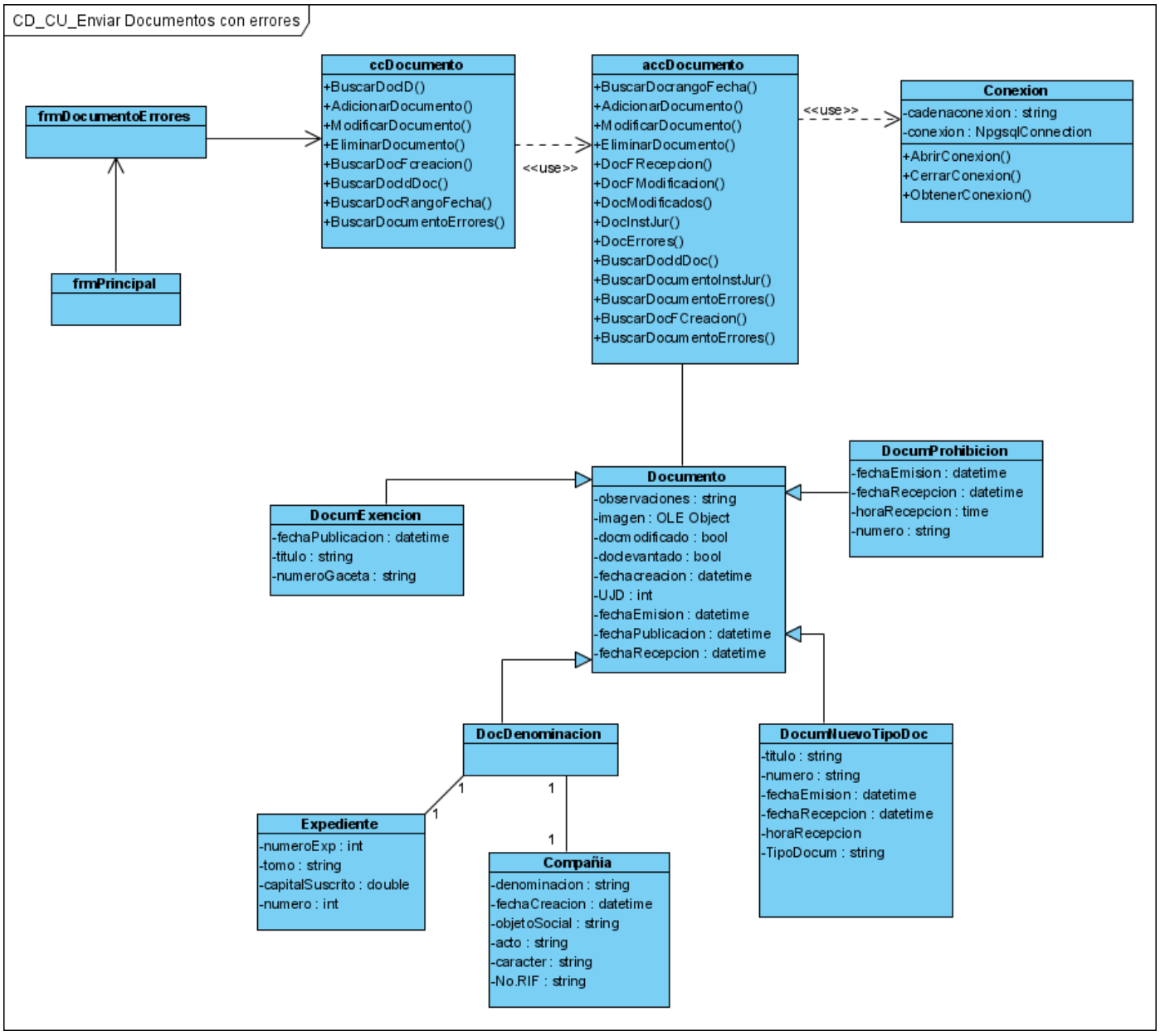

Figura 37 CU: Enviar Documentos con errores

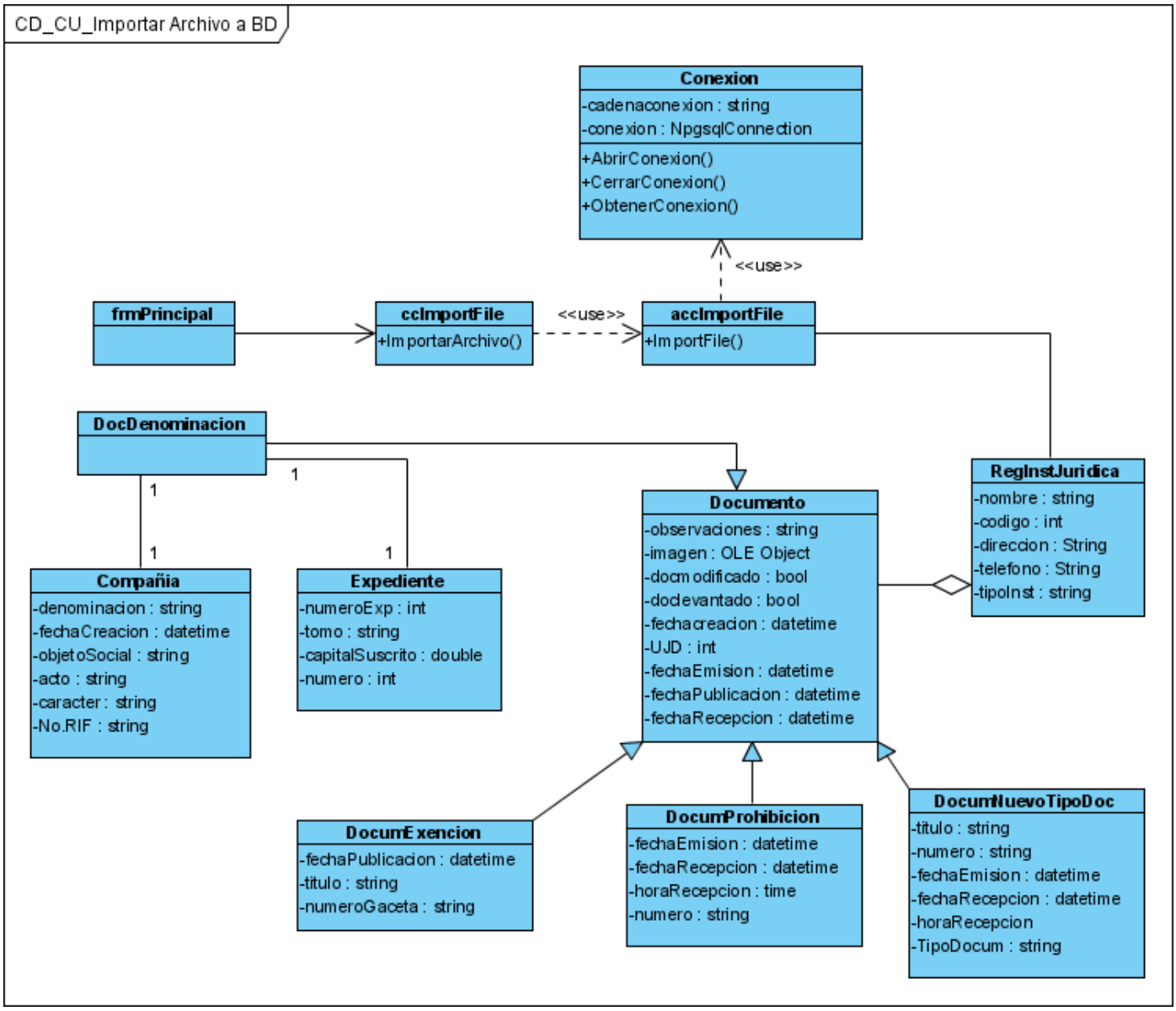

Figura 38 CU: Importar Archivo a BD

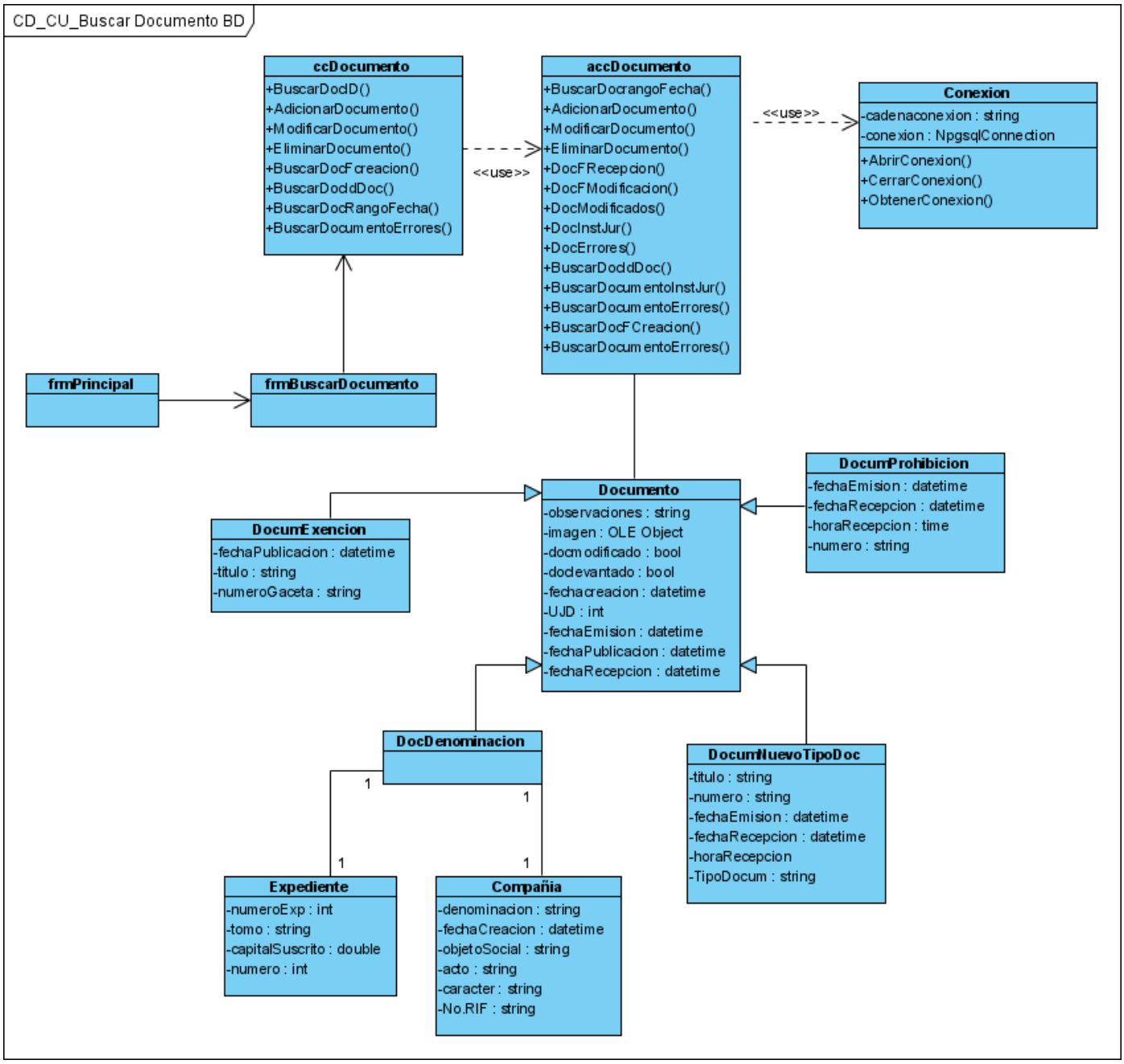

Figura 39 CU: Buscar Documento BD

Diagramas de Colaboración del Análisis

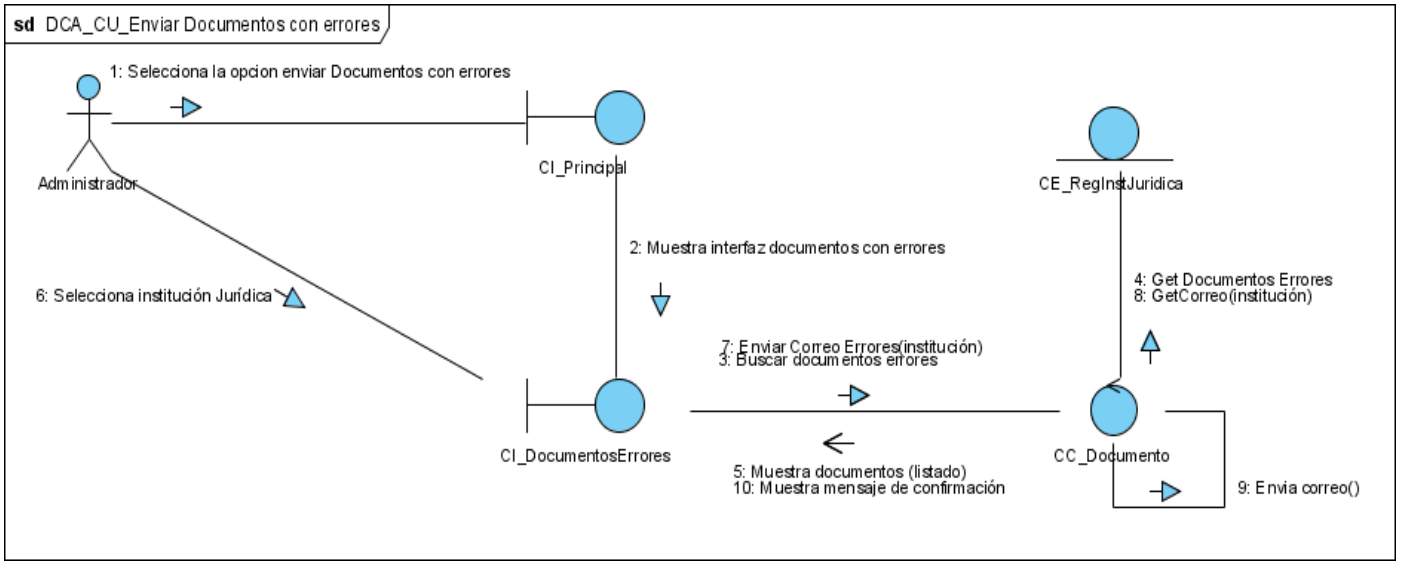

Figura 40 CU: Enviar Documentos con errores

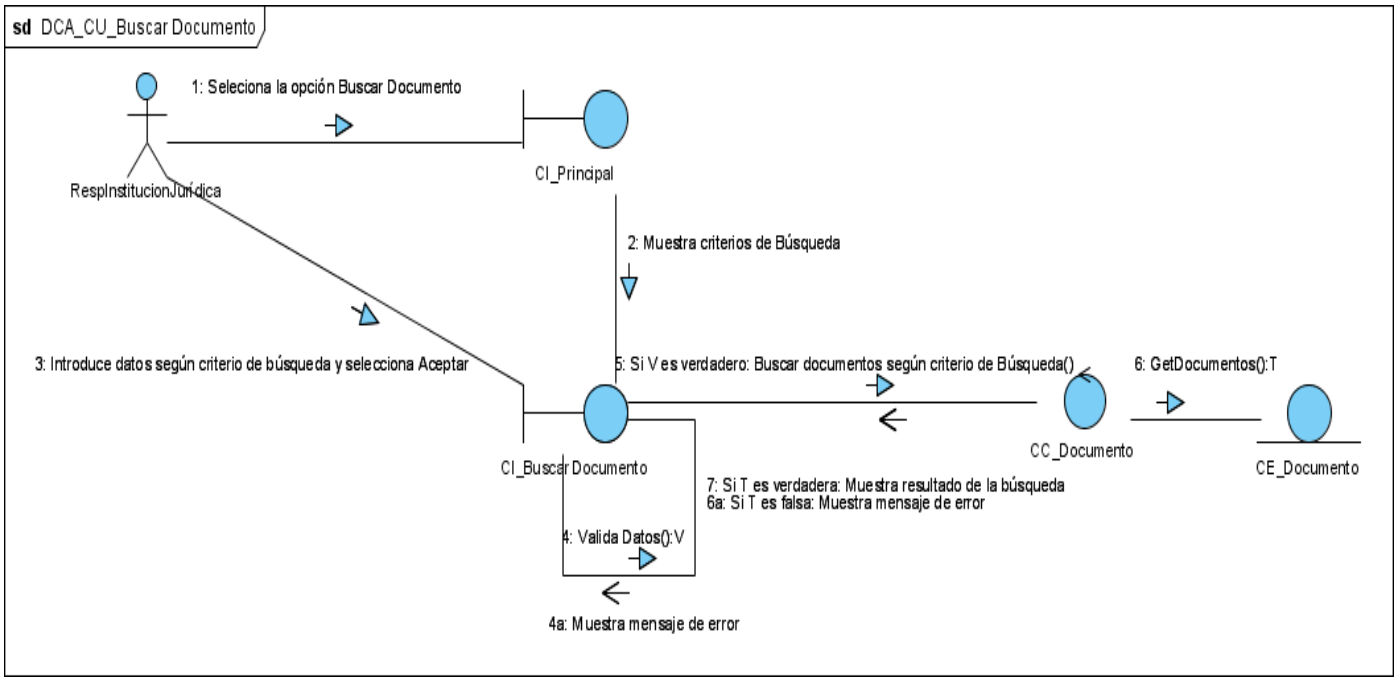

Figura 41 CU: Buscar Documento fichero

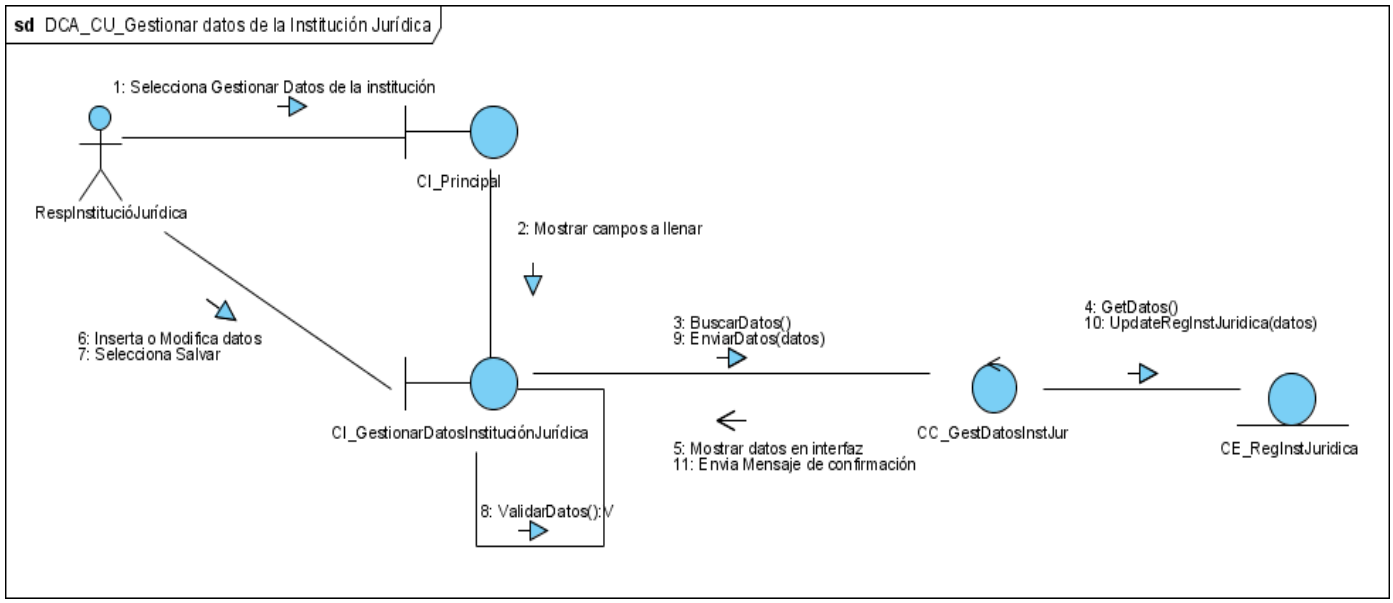

Figura 42 CU: Gestionar datos de la Institución Jurídica

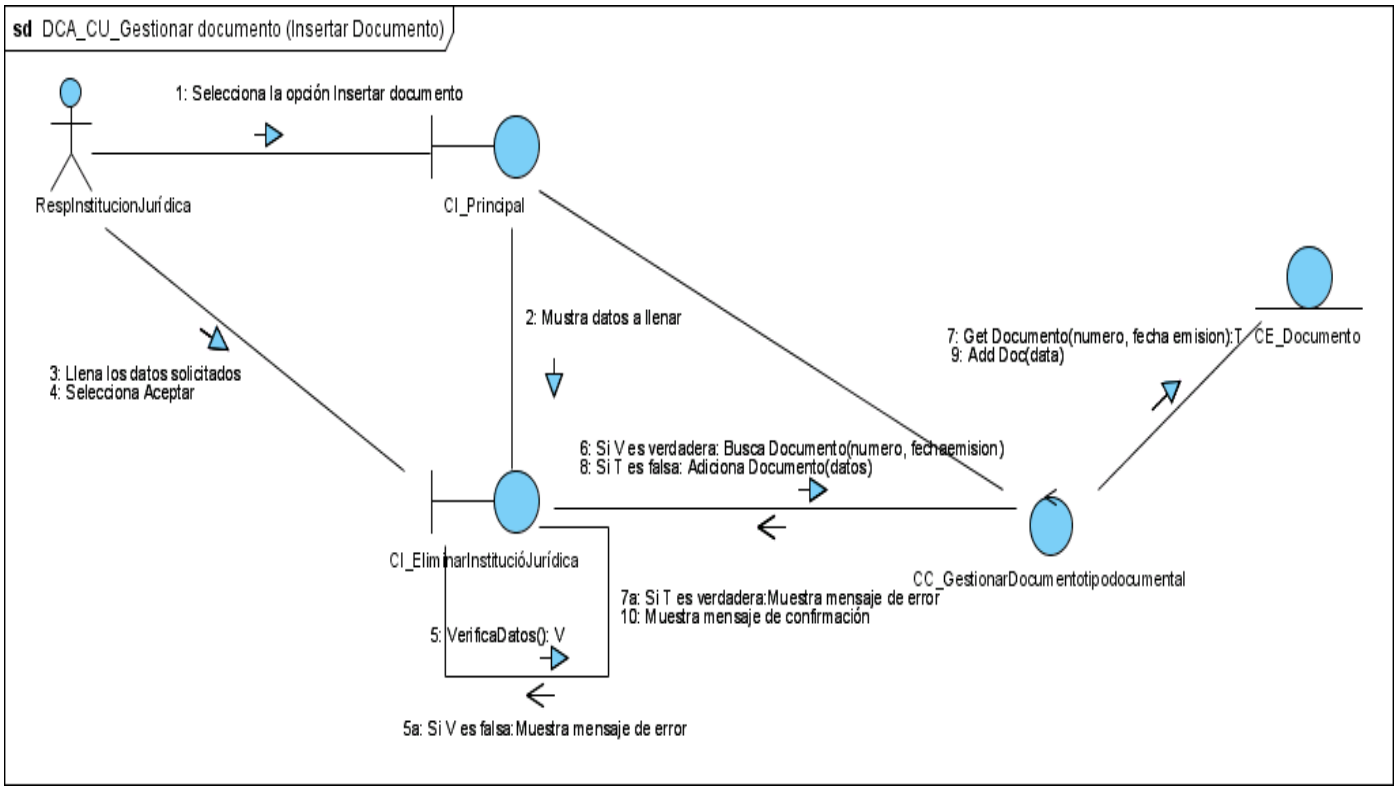

Figura 43 CU: Gestionar documento (Sección Insertar Documento)

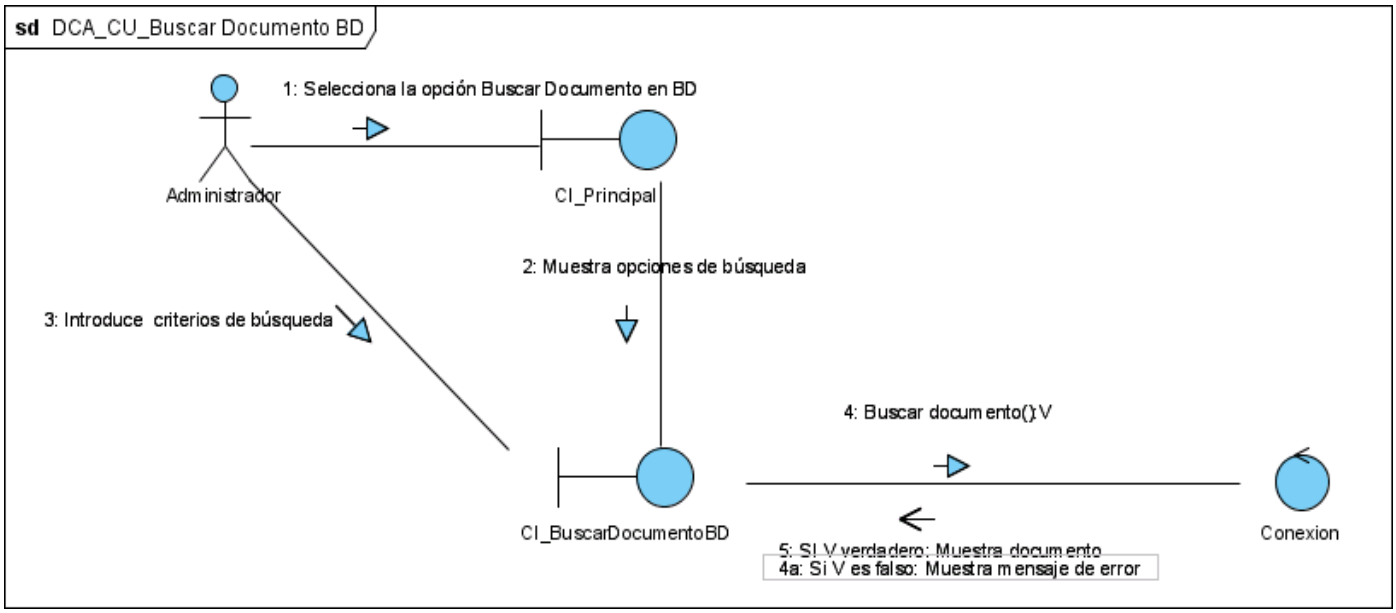

Figura 44 CU: Buscar Documento BD

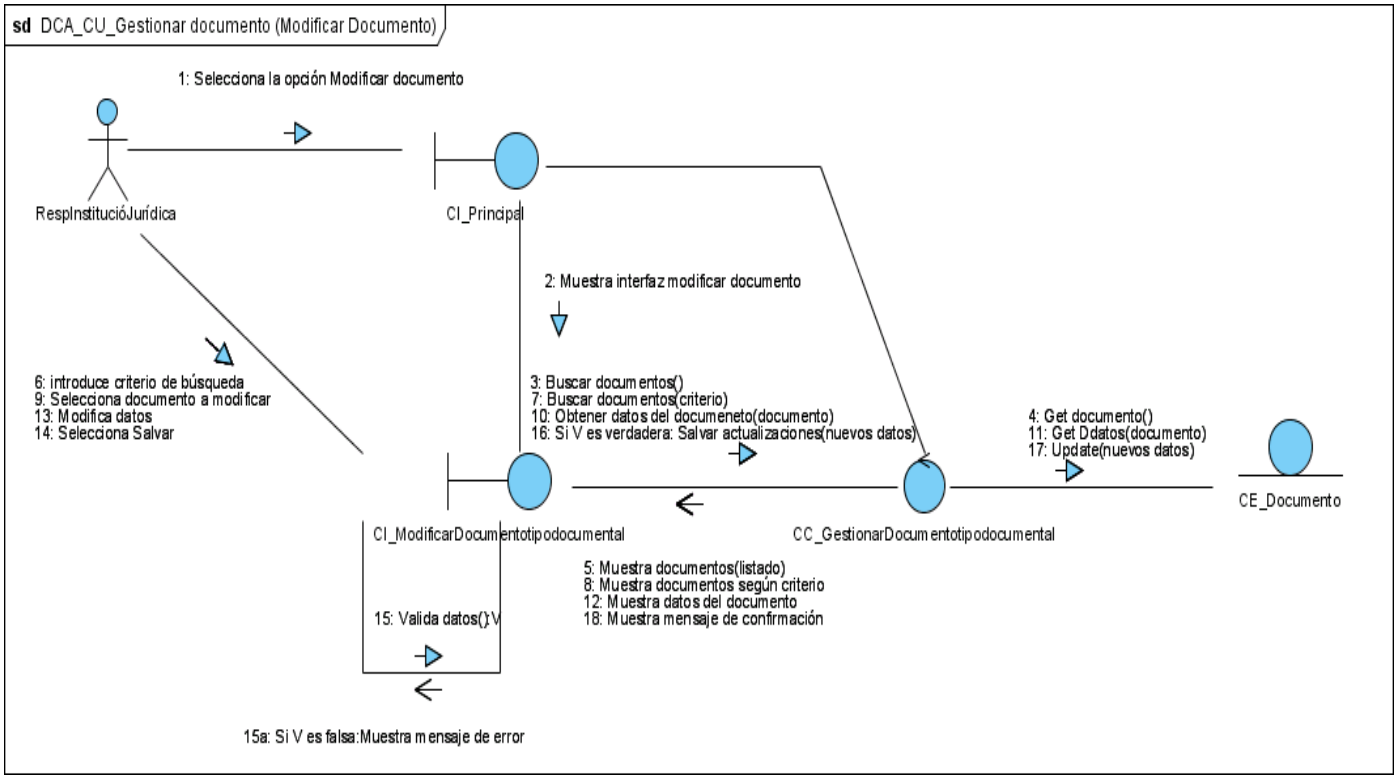

Figura 45 CU: Gestionar documento (Sección Modificar Documento)

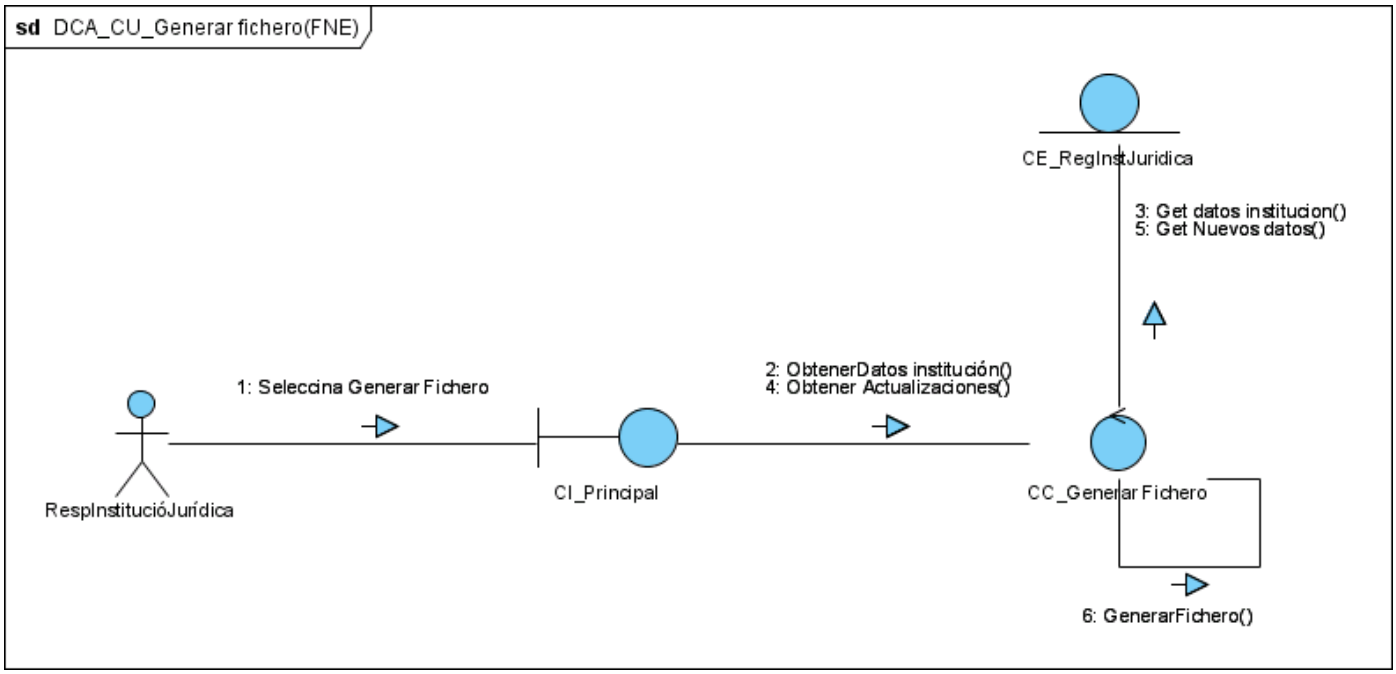

Figura 46 CU: Generar fichero (Flujo Normal de Eventos)

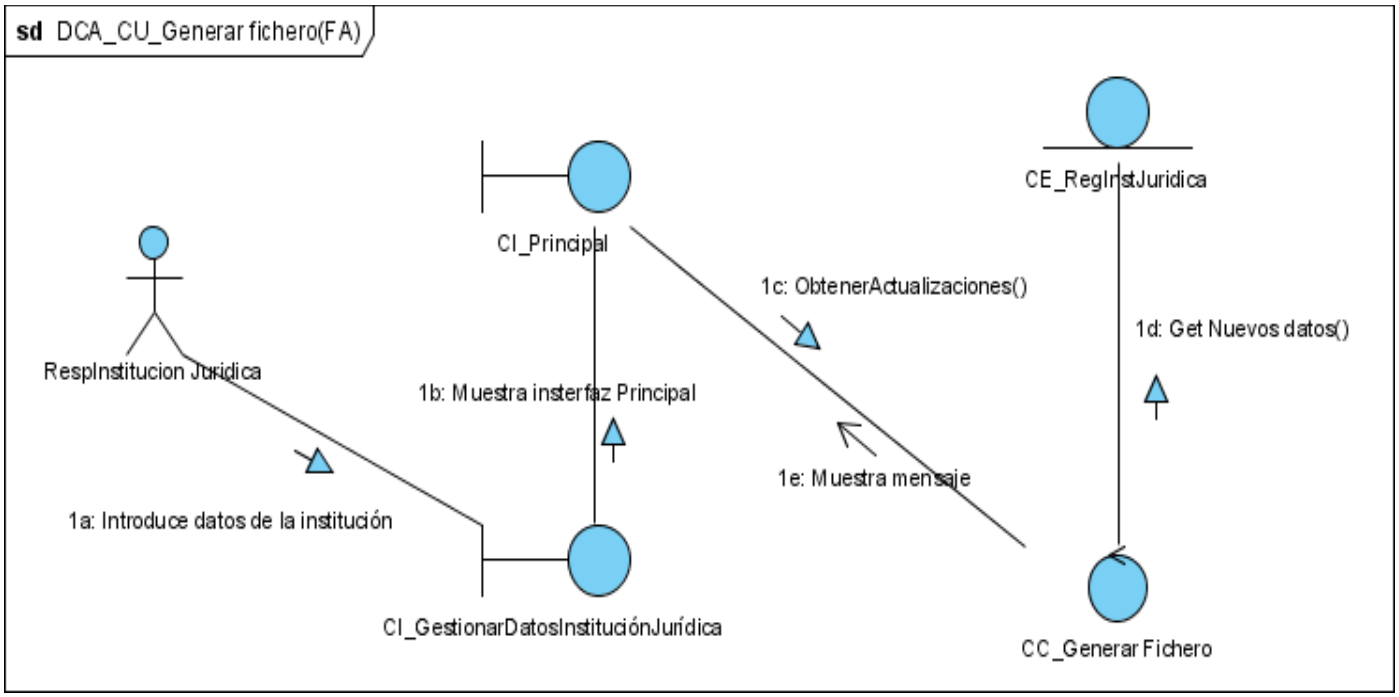

Figura 47 CU: Generar fichero (Flujos alternos)

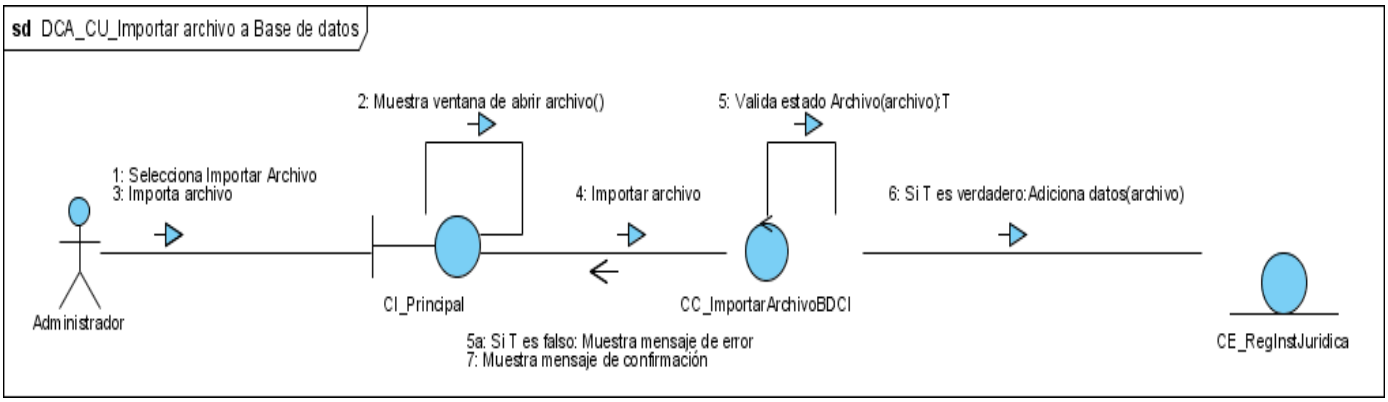

Figura 48 CU: Importar Archivo

Diagramas de Secuencia del Diseño

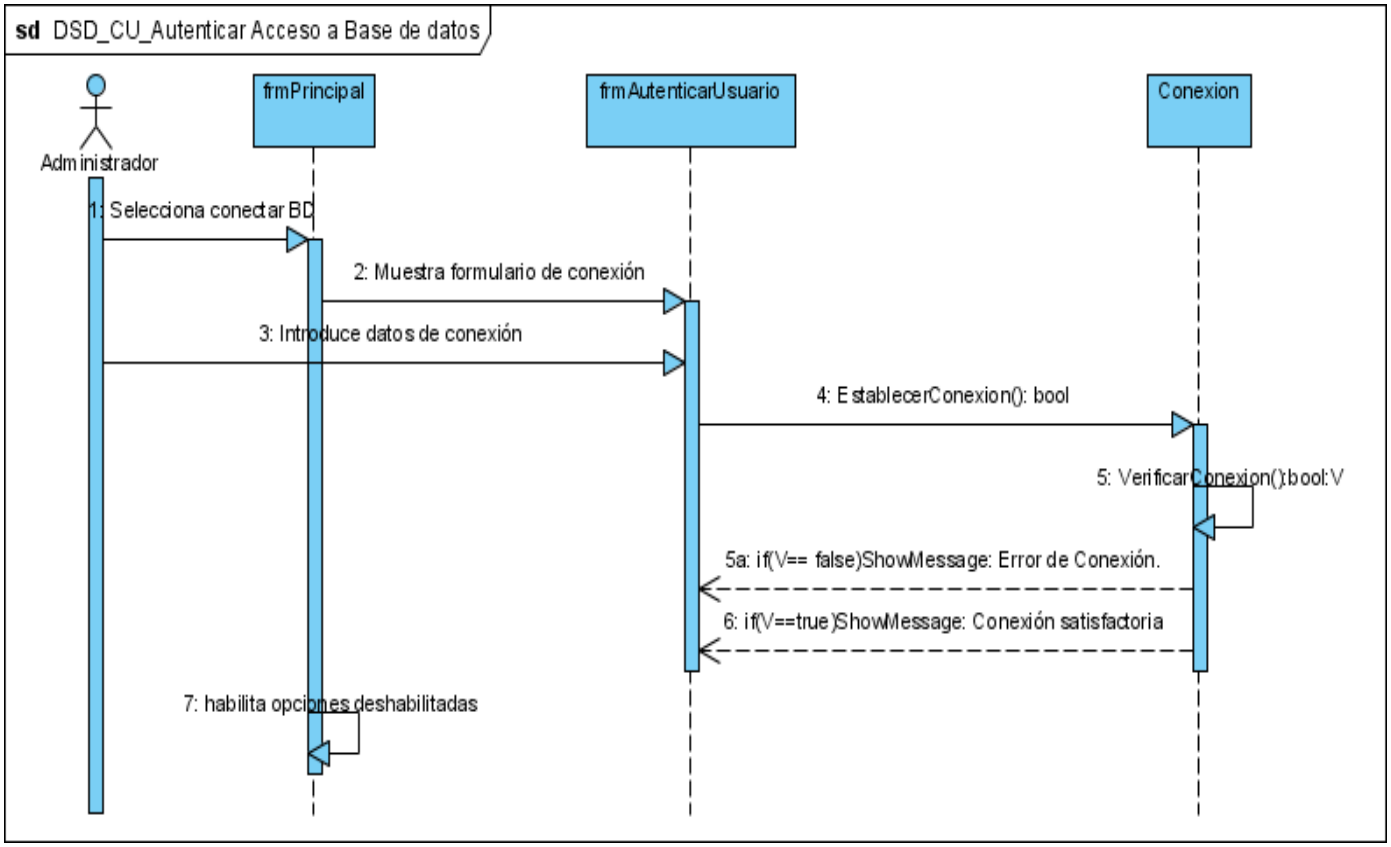

Figura 49 CU: Autenticar Acceso a Base de datos

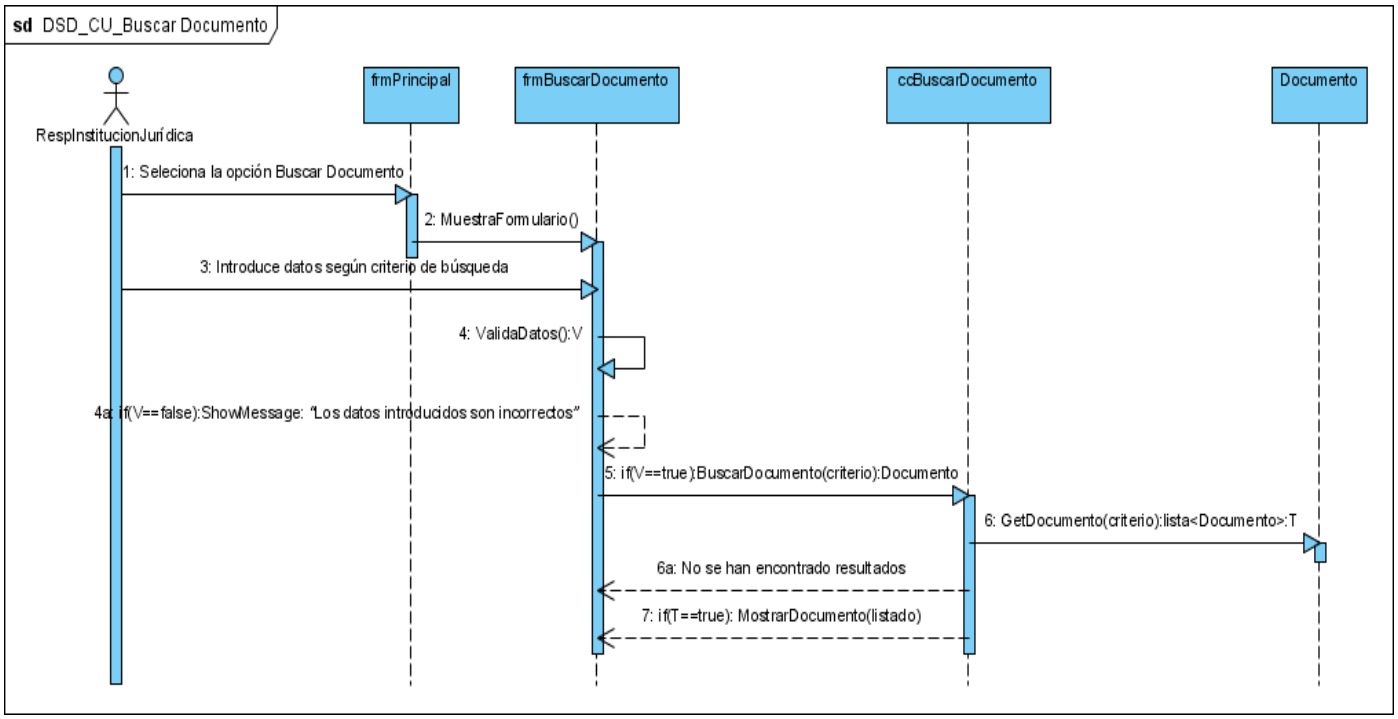

Figura 50 CU: Buscar Documento fichero

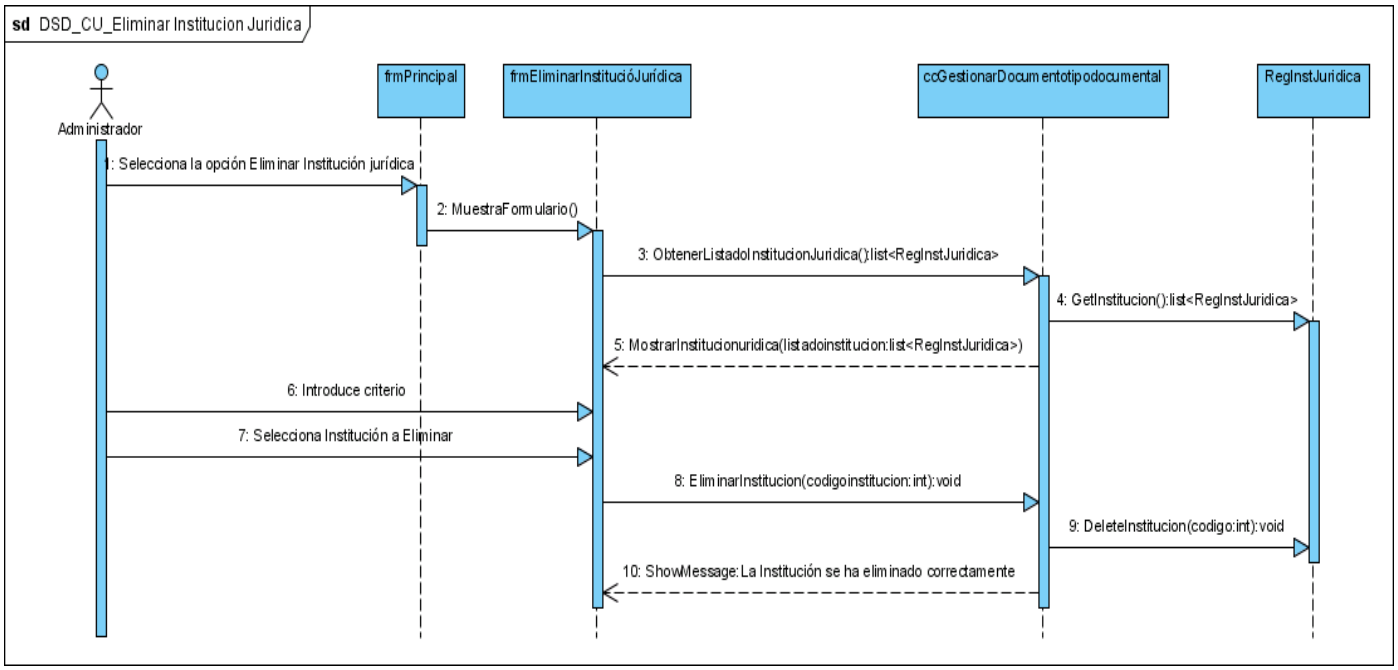

Figura 51 CU: Eliminar Institución Jurídica

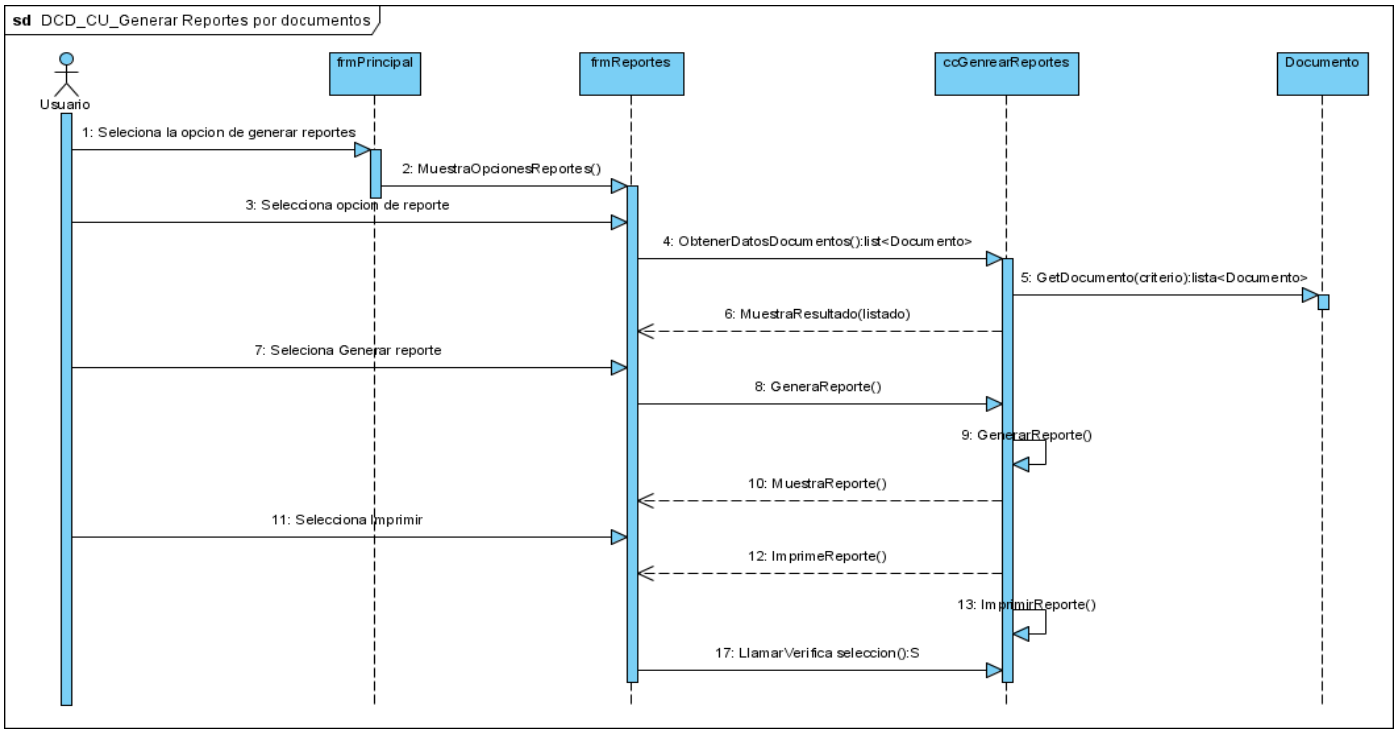

Figura 52 CU: Generar Reportes por documentos

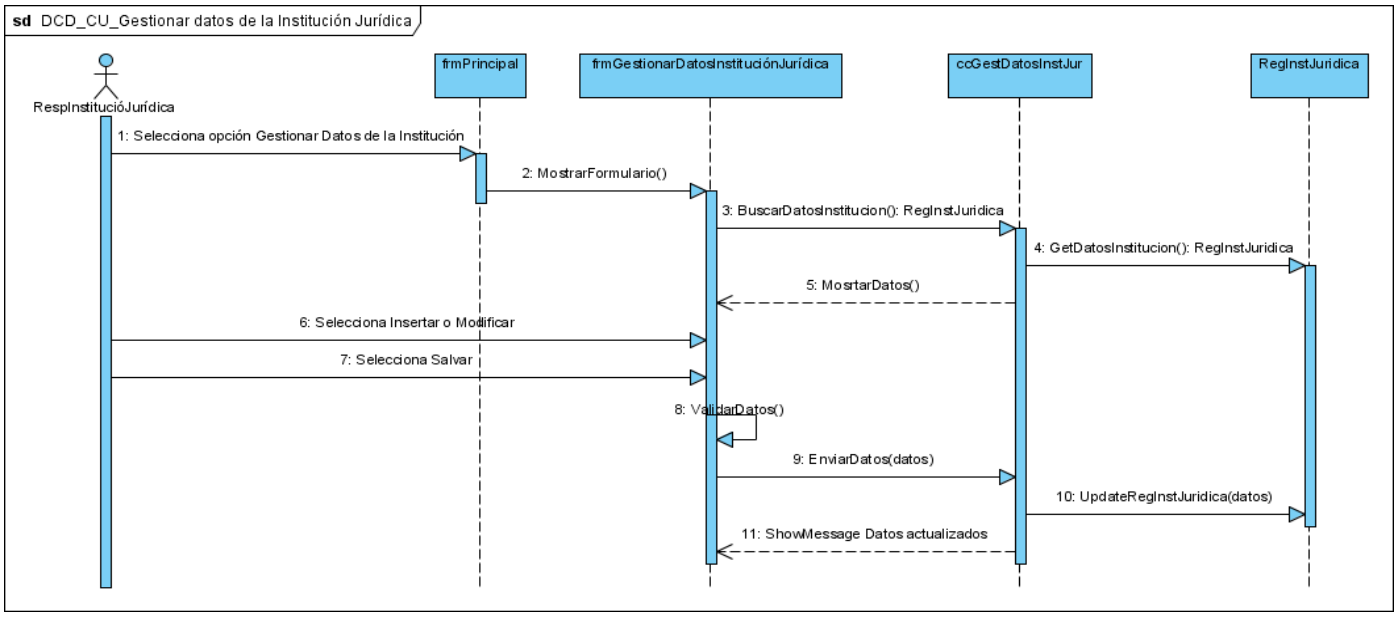

Figura 53 CU: Gestionar datos de la institución jurídica

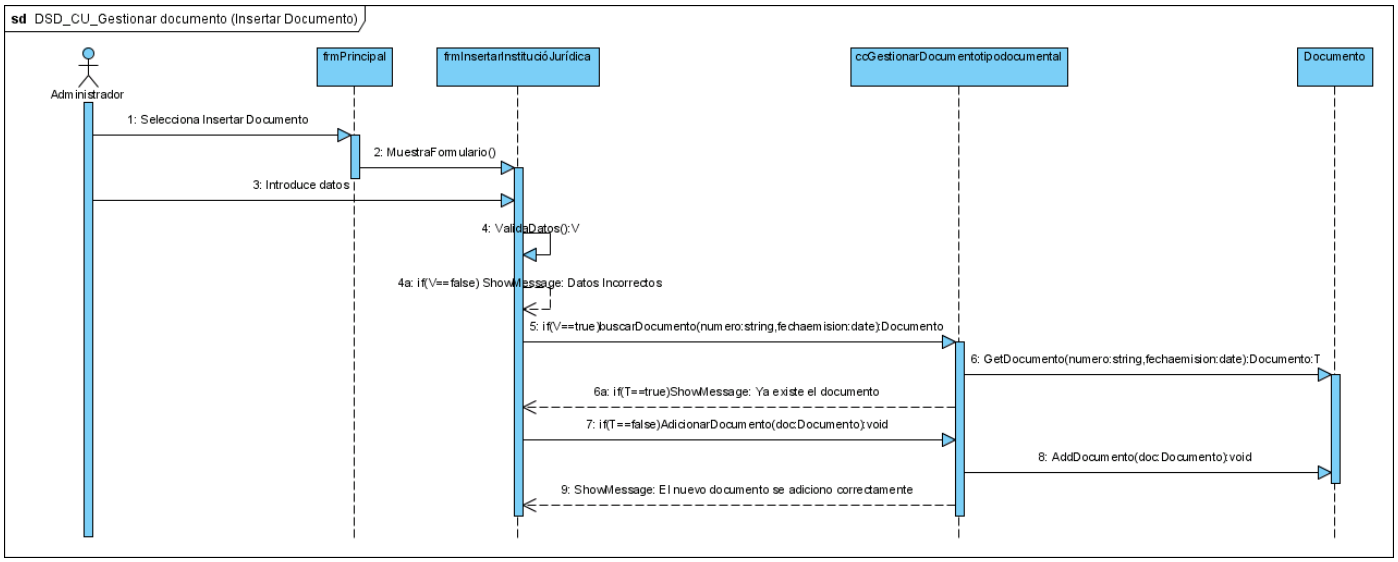

Figura 54 CU: Gestionar documento (Sección Insertar Documento)

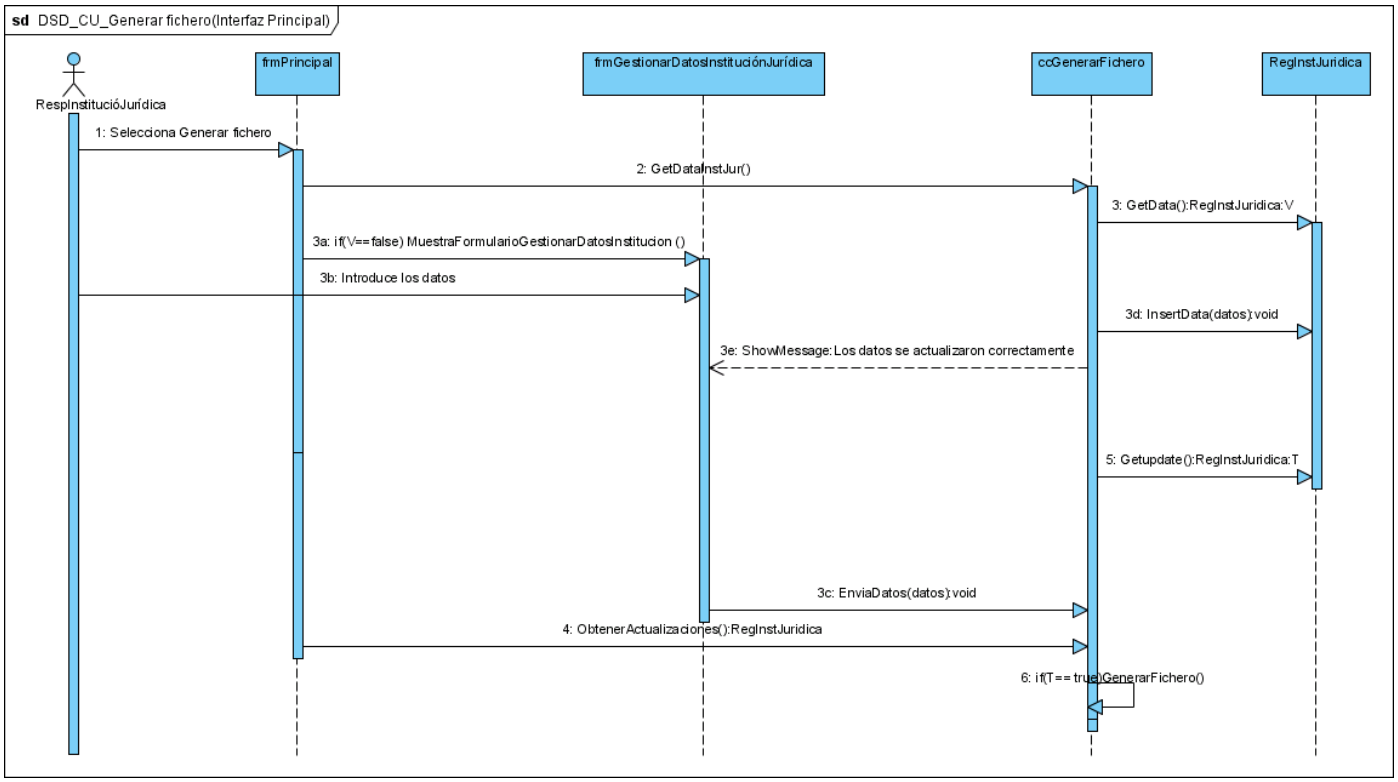

Figura 55 CU: Generar fichero

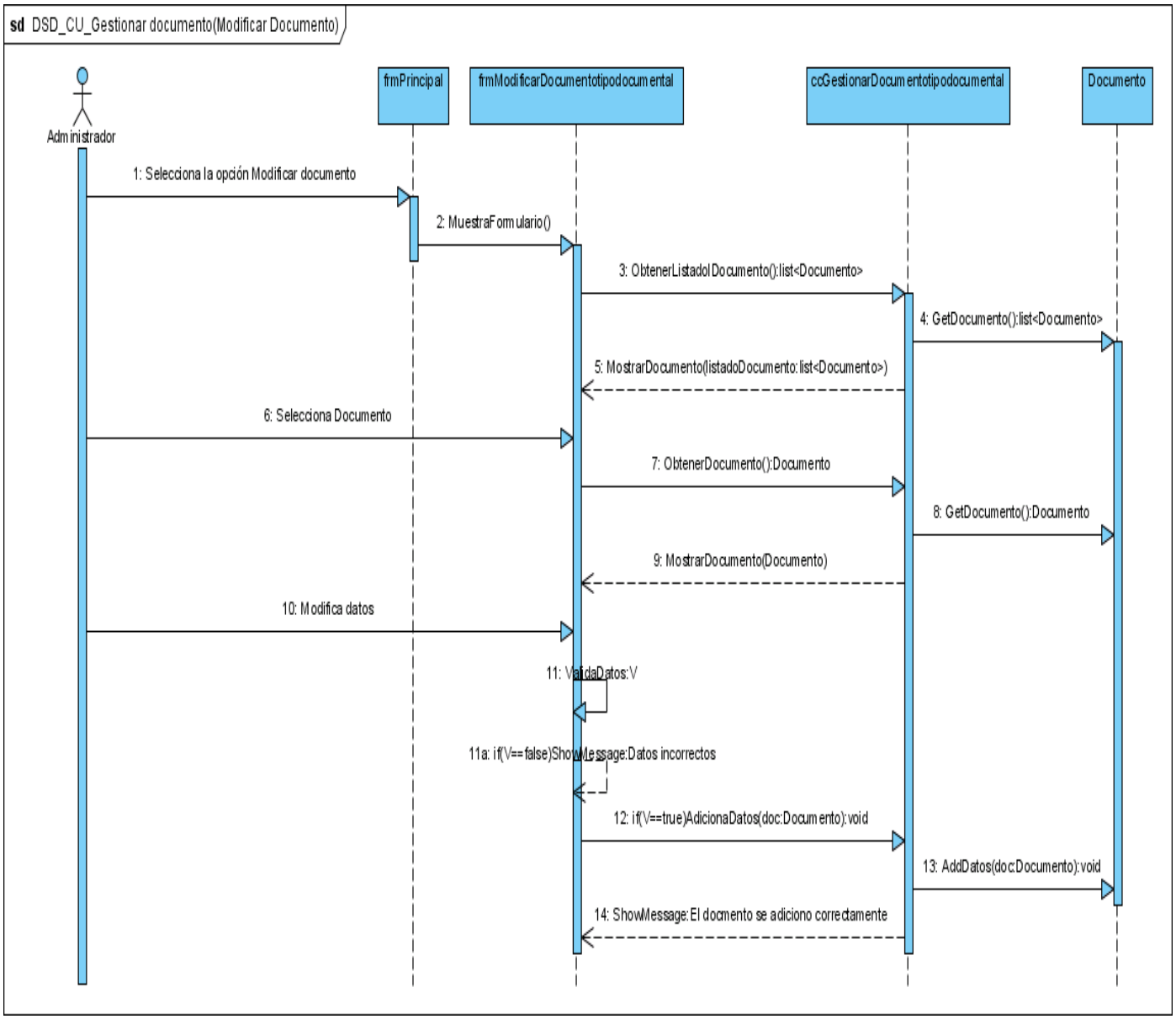

Figura 56 CU: Gestionar documento (Sección Modificar Documento)

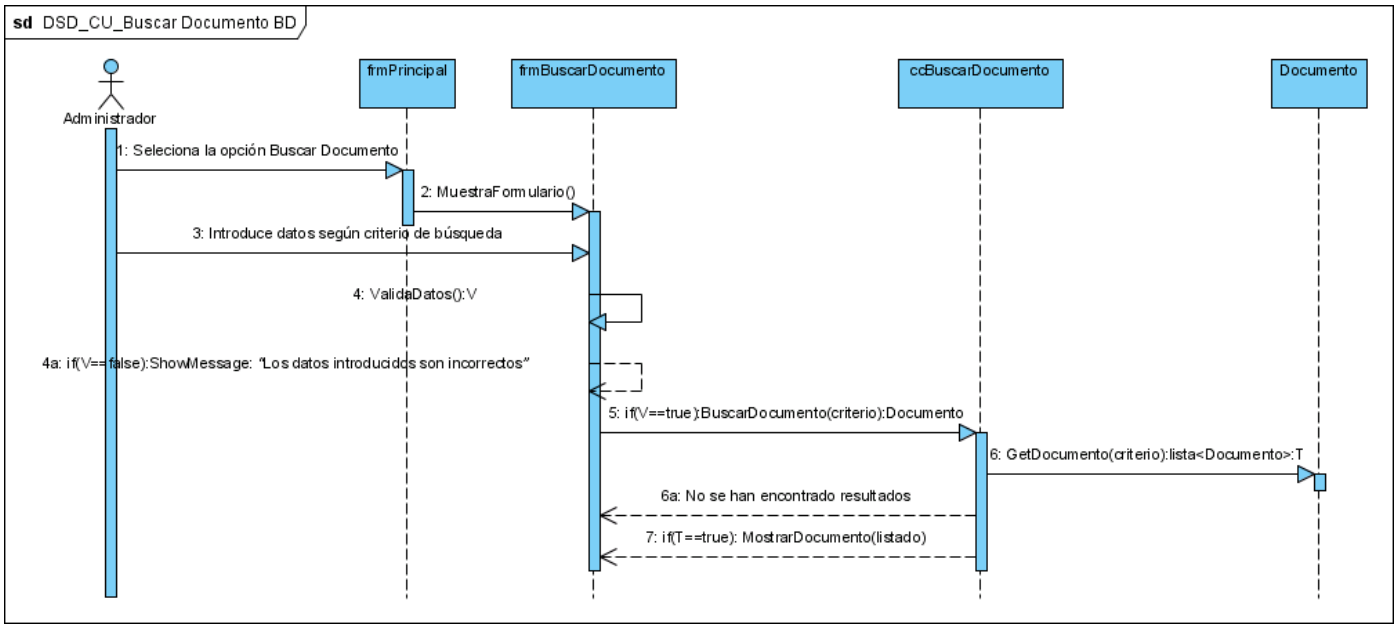

Figura 57 CU: Buscar Documento BD

Interfaz de Usuario

Figura 58 Conexion a BD

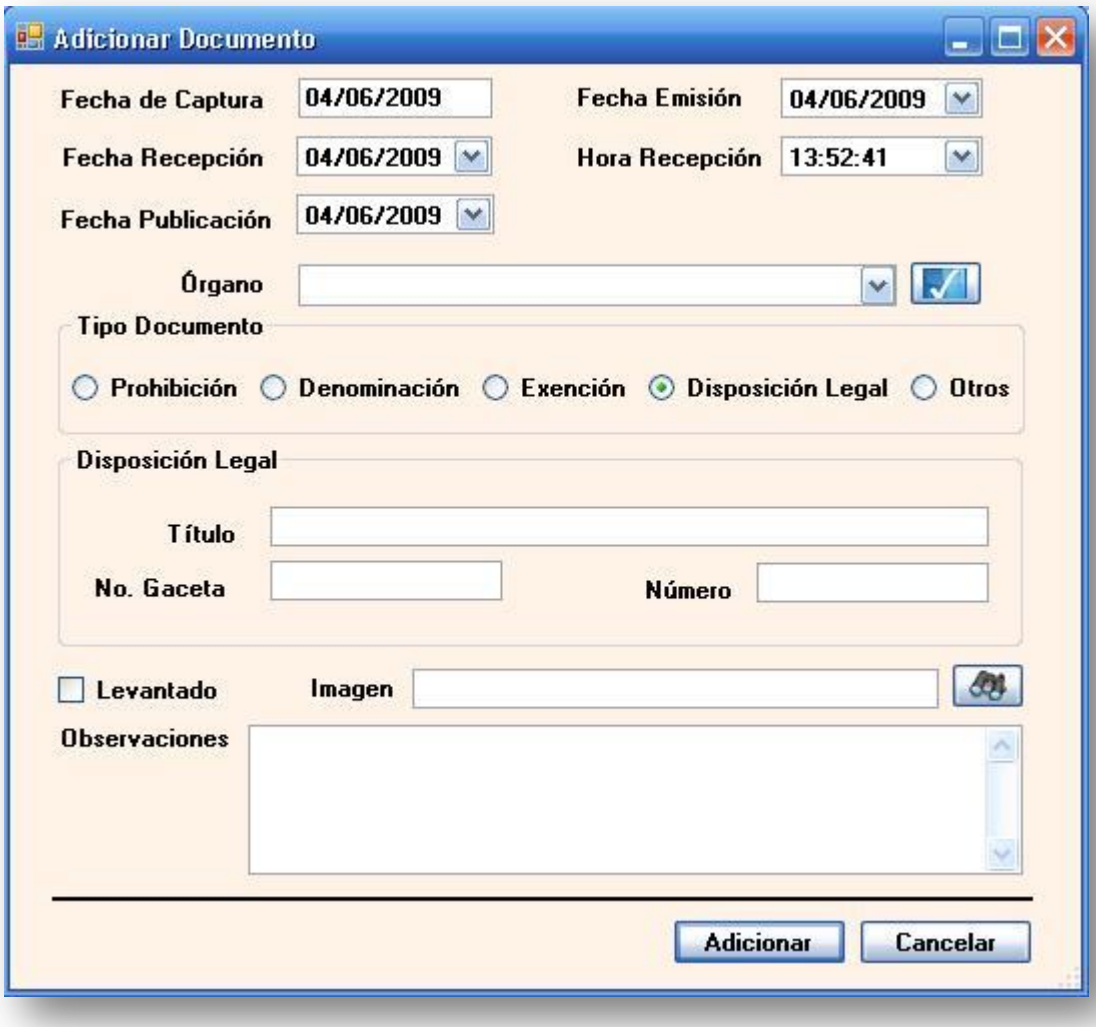

Figura 59 Adicionar Documento Disposición Legal

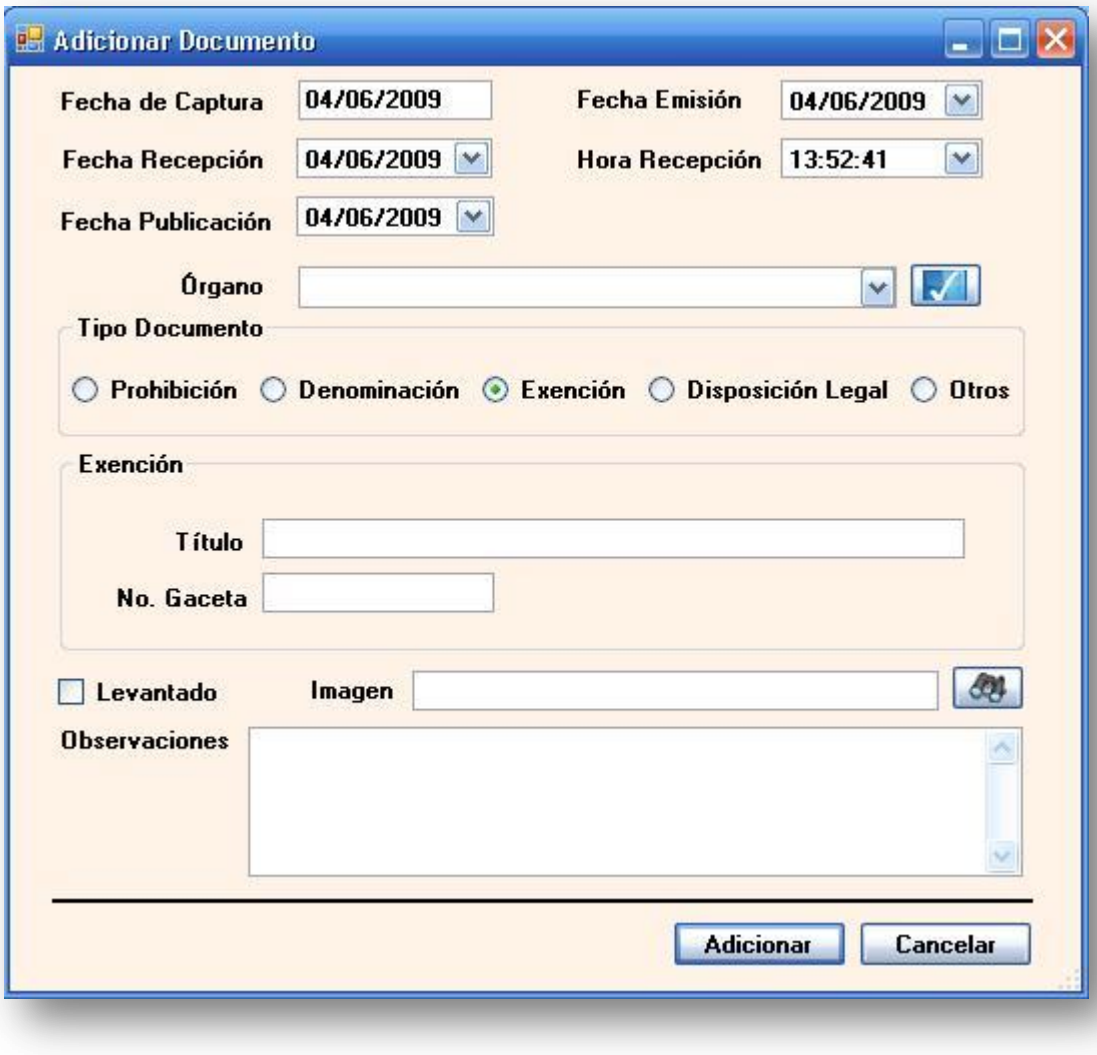

Figura 60 Adicionar Docuemnto Exención

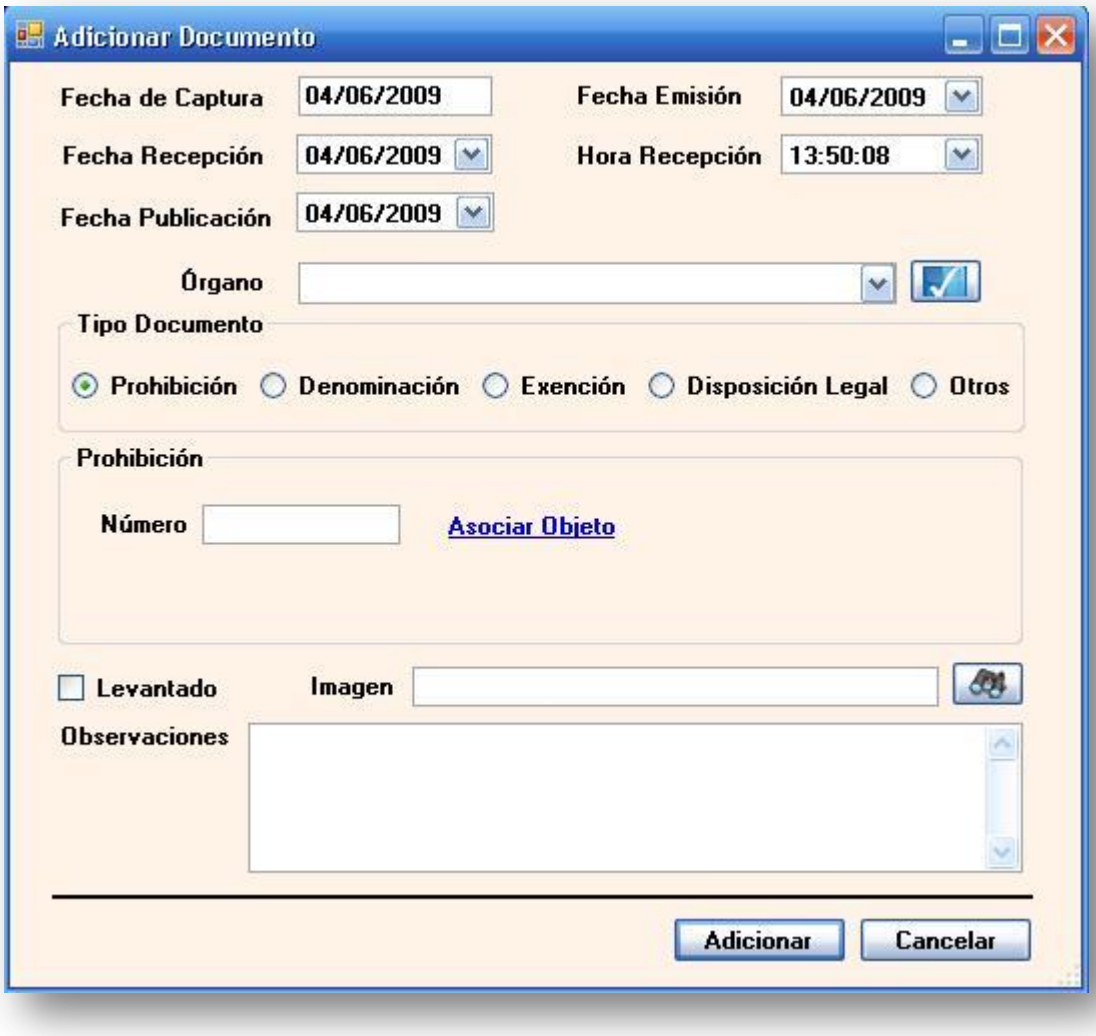

Figura 61 Adicionar Documento Prohibición

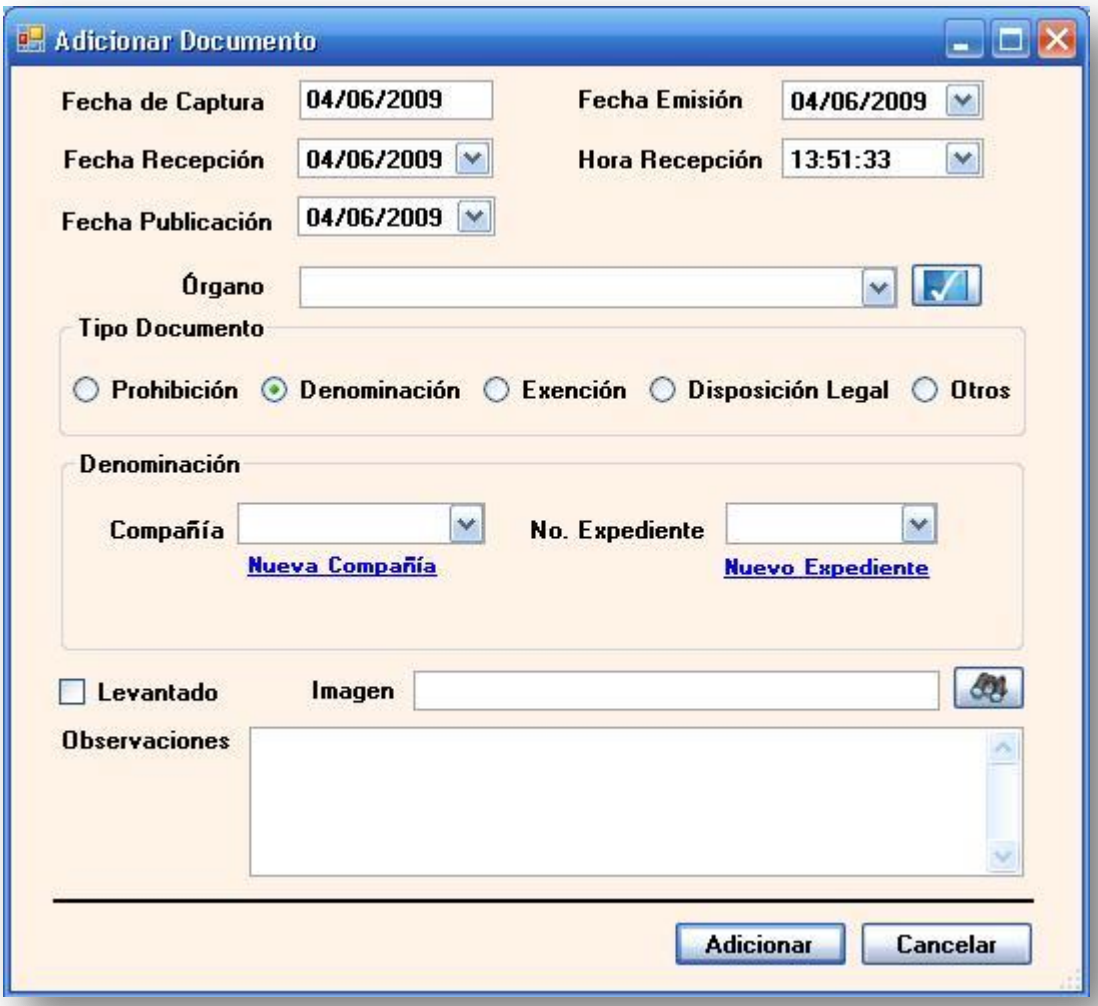

Figura 62 Adicionar Documento Denominación

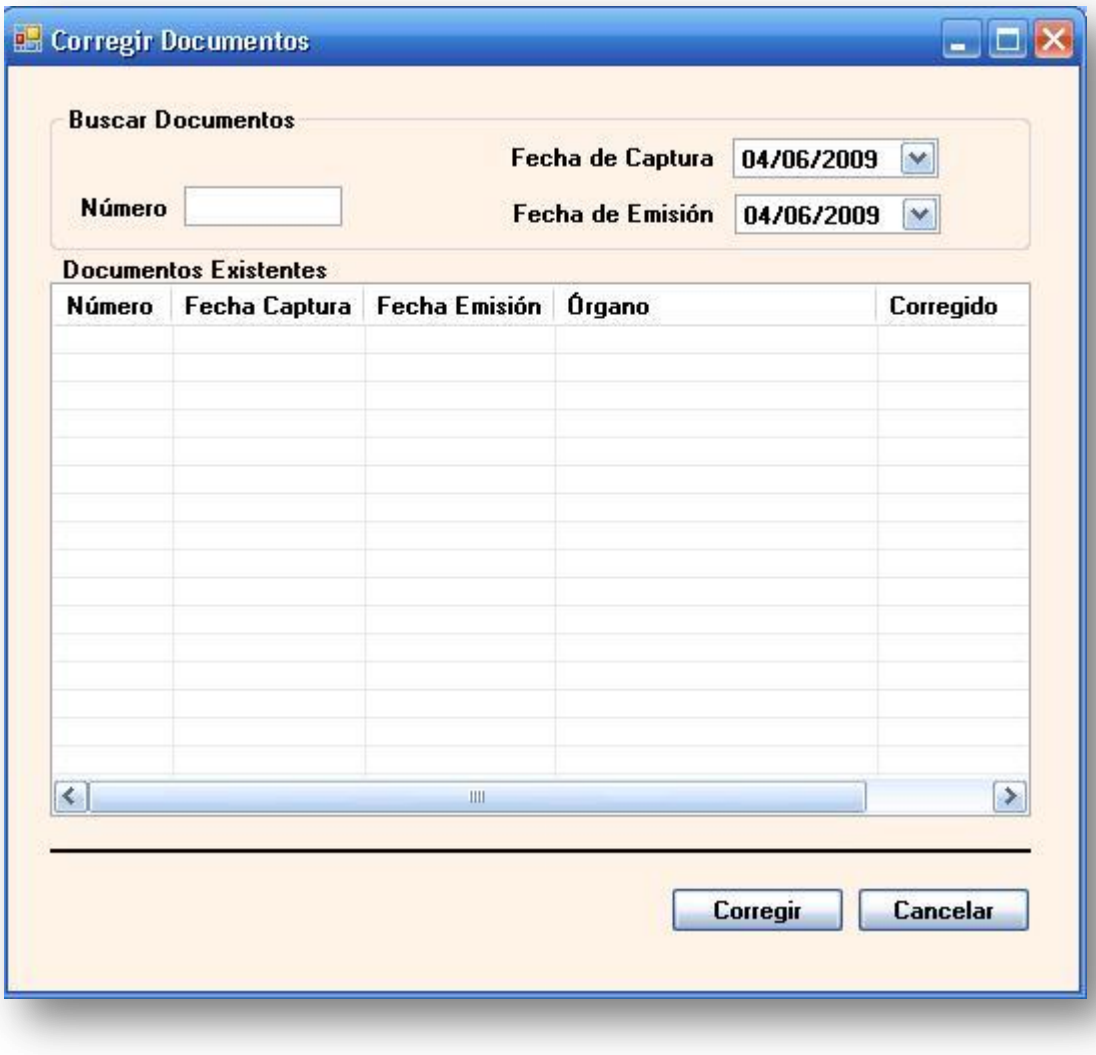

Figura 63 Modificar Documentos

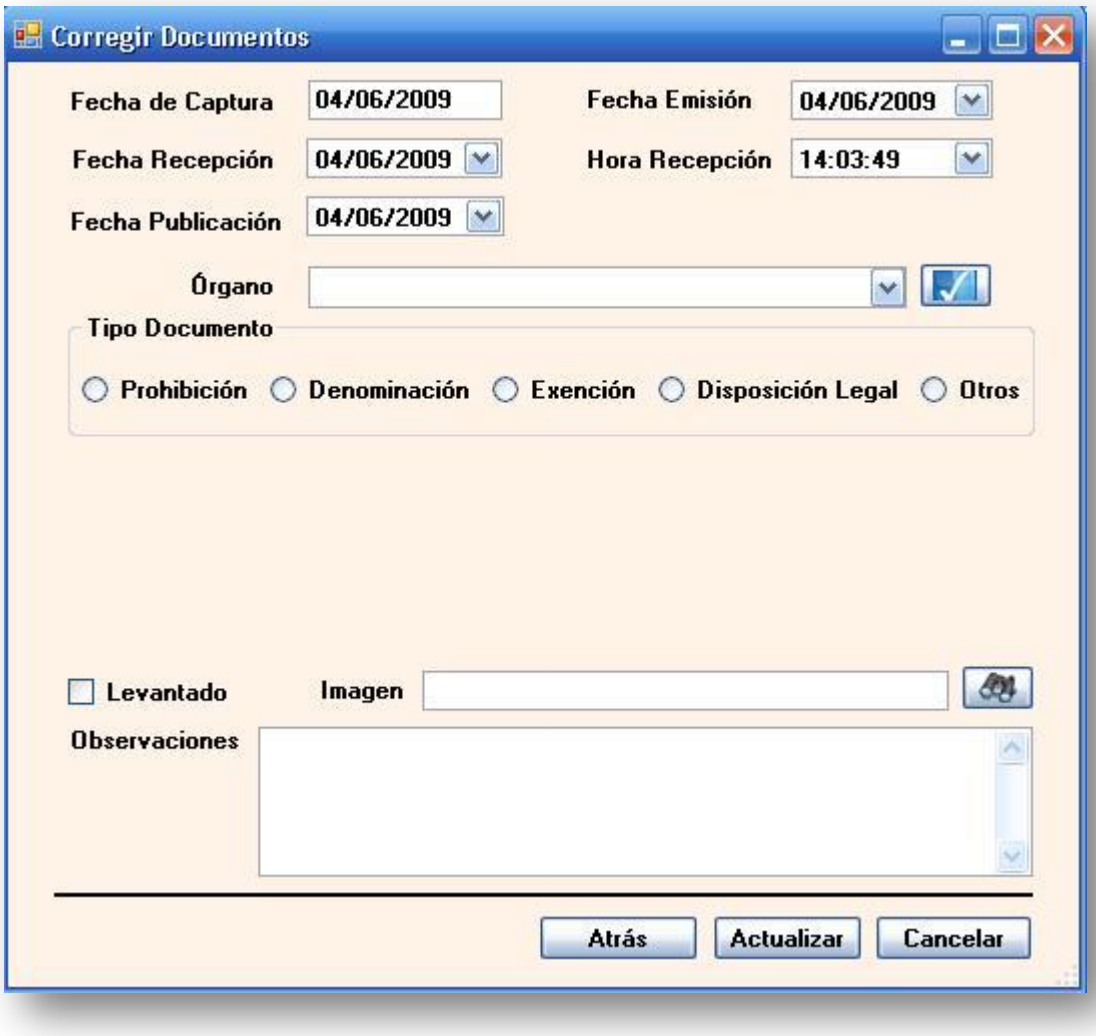

Figura 64 Modificar Documento