Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e implementación del módulo “Inscripción” del Sistema para la Gestión de Antecedentes Penales (SIGESAP).

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

Autor (es):
Dayana Sanz Campos
Jorge Alberto Rueda Flores

Tutor: Ing. Eliober Cleger Despaigne

Declaración de autoría

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

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

_____ _____

Firma del autor

Dayana Sanz Campos

Firma del autor

Jorge Alberto Rueda Flores

_____

Firma del tutor

Eliober Cleger Despaigne

A mis padres por ser lo más maravilloso que me ha dado la vida.

A mis abuelos por sus enseñanzas, su preocupación y su amor infinito.

Dayana

Dedico el presente trabajo de diploma a mis padres por ser lo más valioso que tengo en mi vida. A ellos que me han educado y enseñado mucho de lo que sé.

A mi abuela que ocupa un lugar especial en mi vida.

Jorge

En la actualidad la cantidad de información que se manipula diariamente se incrementa de modo considerable, por lo cual el uso de los sistemas informáticos se hace imprescindible para el tratamiento de la misma. Todas las naciones tienen el deber y la responsabilidad de inscribir según la constitución los antecedentes penales de sus ciudadanos. La División de Antecedentes Penales (DAP) es una institución única de su tipo dentro del territorio venezolano y es la encargada de realizar dicho proceso. Debido a los volúmenes de información que se manipulan y a las irregularidades existentes en dicha institución se decide la realización del proyecto Solución Tecnológica Integral para la Automatización y Modernización de la DAP de la República Bolivariana de Venezuela.

El proceso de inscripción es uno de los fundamentales que se realizan en la DAP. Entre los principales problemas detectados en el mismo se encuentran la duplicidad de documentos y dificultades en el área de revisión. Para darle solución a esta problemática se realiza el presente trabajo de diploma con el objetivo de lograr el diseño e implementación del módulo Inscripción. Para su desarrollo se realizó un análisis sobre la metodología a emplear, así como las herramientas y tecnologías. Finalmente se evaluó la calidad de la solución propuesta mediante la realización de pruebas de caja negra, integración y aceptación. El presente trabajo expone los resultados obtenidos durante cada una de estas actividades.

Palabras claves: diseño, implementación, inscripción.

ÍNDICE

INTRODUCCIÓN

El acelerado desarrollo de la informática en conjunto con Internet ha marcado una pauta en el mundo actual. A partir de los años 90 se comenzó a utilizar el término “era de la información”, debido a que las redes permitían transmitir la información de un lugar a otro sin realizar desplazamientos físicos de personas u objetos, convirtiéndose en un recurso decisivo para el desarrollo mundial y las comunicaciones. La cantidad de la información que se manipula diariamente se incrementa de modo considerable, por lo cual el uso de los sistemas informáticos se hace imprescindible para el tratamiento de la misma.

En medio de este desarrollo el presidente de Venezuela, Hugo Chávez Frías, y el entonces presidente de Cuba, Fidel Castro Ruz, llevaron a cabo el 30 de octubre del año 2000 la firma de un acuerdo de cooperación integral para la elaboración de programas con el fin de fortalecer los lazos de amistad y fomentar el progreso de sus economías. Para la aprobación de los proyectos a desarrollar, las partes establecieron una Comisión Mixta integrada por representantes de ambos países.

En la VII Comisión Mixta desarrollada en La Habana se propuso el proyecto Solución Tecnológica Integral para la Automatización y Modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela conocido por las siglas RAP (Registro de Antecedentes Penales), pero no fue hasta diciembre del 2009, cuando se celebró la X Comisión Mixta, que se aprobó la realización del mismo.

La Universidad de las Ciencias Informáticas (UCI) cuenta con varios centros que tienen como objetivo la creación de productos, servicios y soluciones informáticas de alta calidad. El Centro de Gobierno Electrónico (CEGEL) es uno de ellos, dedicado específicamente a la informática jurídica y es el encargado de la realización de dicho proyecto.

La División de Antecedentes Penales (DAP) adscrita a la Dirección General de Justicia, Instituciones Religiosas y Cultos, perteneciente al Vice Ministerio de Política Interior y Seguridad Jurídica, tiene como responsabilidad el registro de los antecedentes penales de los ciudadanos venezolanos y la emisión de certificaciones de antecedentes penales a los distintos entes, tanto nacionales como internacionales, que se encuentren autorizados por la legislación venezolana vigente.

La inscripción y certificación de antecedentes penales constituyen los procesos fundamentales para la gestión de antecedentes penales en la República Bolivariana de Venezuela. Por su importancia y las irregularidades detectadas en ambos, el primero constituye el centro de atención del presente trabajo de diploma.

Para cumplir con las actividades previstas en la inscripción de tipos documentales, la DAP se encuentra organizada en cinco áreas de trabajo: presentación (taquilla), procesamiento de datos (operadores), revisión legal (abogados), otorgamiento (director), archivo (archivólogo). La inscripción de unidades documentales es un proceso complejo y fragmentado, cuyo éxito depende en gran medida de la ética profesional de los funcionarios que en él laboran y de las condiciones tecnológicas para su ejecución. En cada área se incorpora un elemento necesario para cumplir con la meta propuesta. La taquilla debe aportar los metadatos generales de cada unidad documental, en el área de procesamiento de datos se deben incorporar los metadatos de actualización o condena para cada privado de libertad incluido en la unidad documental. Los abogados, quienes juegan un papel fundamental, deben dar fe legal de la calidad y transparencia del proceso, emitiendo la respuesta correspondiente a la inscripción en curso. La autoridad máxima de la división debe otorgar los trámites revisados en el área anterior, dando así su autorización para que la unidad documental finalmente sea archivada en el fondo documental de la DAP.

Los pasos antes mencionados fueron el resultado de un proceso de transformación organizacional realizado por el equipo de especialistas del proyecto. El mismo perseguía el objetivo de introducir mejoras que contribuyeran a la gestión eficiente de los antecedentes penales de los ciudadanos venezolanos. Para obtener la información necesaria se realizó un diagnóstico sobre la situación existente en la DAP, sus resultados para el proceso de inscripción se exponen a continuación:

- El sistema en explotación solo se encarga de la inserción y recuperación de información, no incluye la búsqueda de coincidencias para los documentos a inscribir, permitiendo la duplicación de los mismos.
- Todos los documentos se inscriben como Sentencias Definitivamente Firmes, no se permite la inscripción de ningún otro tipo de documento asociado a un proceso penal.
- Las áreas de presentación y otorgamiento no intervienen en el proceso informatizado.

- La digitalización de unidades documentales no está contemplada dentro del proceso de asociación de metadatos.
- La revisión legal presenta dificultades, debido en gran medida a la ausencia de dicha funcionalidad en el sistema y a la carencia de personal para el área. (1)

Se pudo concluir, mediante la observación del proceso de inscripción, que las limitantes antes mencionadas provocan un aumento excesivo del volumen de los expedientes por duplicación de unidades documentales. Además, existen unidades documentales archivadas con errores y documentos correspondientes a un mismo proceso penal fuera del expediente al que pertenecen.

La problemática anteriormente descrita afecta el correcto funcionamiento de la DAP. Por ello durante el análisis realizado al proceso de inscripción por los especialistas del proyecto, se identificaron los requisitos funcionales y no funcionales que deben ser incorporados al nuevo sistema para la mejora del proceso de inscripción, generando el siguiente **problema**: ¿Cómo satisfacer los requisitos acordados con los clientes de manera que se mejore la gestión documental del proceso de inscripción que se lleva a cabo en la División de Antecedentes Penales de la República Bolivariana de Venezuela?

El **objeto de estudio** de la investigación es el proceso de desarrollo de software para sistemas de gestión gubernamental. Para dar solución al problema anteriormente planteado, se define como **objetivo general:** realizar el diseño y la implementación del módulo Inscripción del Sistema para la Gestión de Antecedentes Penales (SIGESAP). El **campo de acción** en el cual estará enmarcada la presente investigación es el diseño e implementación de sistemas registrales.

Como guía para la investigación se sostiene la siguiente **idea a defender**: la realización del diseño e implementación del módulo Inscripción de SIGESAP, mejorará la gestión documental que se lleva a cabo durante el proceso de inscripción.

Variable independiente: diseño e implementación del módulo Inscripción.

Variable dependiente: gestión documental mejorada (disminución del volumen excesivo de los expedientes por duplicación de unidades documentales, disminución de documentos fuera del expediente al que pertenecen y disminución de errores en las unidades documentales archivadas).

Para lograr el objetivo general se plantean los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Realizar el diseño del módulo Inscripción.
- Realizar la implementación del módulo Inscripción.
- Validar la solución propuesta.

Se definen como **tareas de la investigación** para dar cumplimiento a los objetivos antes mencionados:

1. Estudio de los sistemas registrales para la inscripción de antecedentes penales.
2. Fundamentación de las tecnologías y herramientas a utilizar para el desarrollo de la solución propuesta.
3. Análisis de los artefactos especificación de requisitos de software y modelo del sistema como base para el diseño y la implementación a realizar.
4. Desarrollo de los artefactos realización de casos de uso del diseño, modelo de datos y modelo de despliegue.
5. Desarrollo del diagrama de componentes para la implementación del módulo.
6. Implementación de los casos de uso e integración con los casos de uso reutilizables.
7. Diseño y aplicación del modelo de pruebas de integración y caja negra para el sistema.

Los **métodos de investigación** que sustentan el presente trabajo son:

- Analítico - sintético: se empleó este método para estudiar toda la bibliografía seleccionada sobre el tema, procesarla y arribar a conclusiones.
- Histórico - lógico: se utilizó este método para realizar un estudio del arte, analizando el desarrollo de los sistemas registrales existentes.
- Hipotético - deductivo: se empleó este método para la confección de la idea a defender.

- Modelación: se utilizó para la creación de modelos y diagramas de diseño del módulo Inscripción.
- Observación: se empleó este método para analizar el flujo de actividades que se realiza durante el proceso de inscripción.
- Experimento: se utilizó este método para validar la solución mediante pruebas realizadas al sistema implementado.

El contenido a desarrollar en el presente trabajo está estructurado en tres capítulos de la siguiente forma:

Capítulo 1: Fundamentación teórica**.** En este capítulo se realiza un análisis del estado del arte y se exponen los principales conceptos relacionados con el mismo. Además se fundamenta el uso del Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) como metodología de desarrollo de software y de las principales herramientas y tecnologías utilizadas.

Capítulo 2: Propuesta de Solución**.** En este capítulo se exponen los elementos de la etapa de requisitos, así como la realización de casos de uso del diseño. Además se fundamenta la arquitectura utilizada y se presenta la vista de implementación a través del diagrama de componentes.

Capítulo 3: Validación de la solución. En este capítulo se valida la solución propuesta mediante la realización de pruebas que aseguren el adecuado funcionamiento del sistema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1. Introducción

Este capítulo abarca los principales elementos teóricos a tener en cuenta para una mejor comprensión de la solución que se propone. Aborda ejemplos de sistemas registrales para gestionar los antecedentes penales. Se exponen los conceptos fundamentales relacionados con la inscripción de unidades documentales con valor jurídico. Además, se explica el funcionamiento de este proceso dentro de la DAP de Venezuela. Se plantean las características de la metodología a emplear y por último se analizan las principales herramientas y tecnologías propuestas para el desarrollo del módulo Inscripción.

2. Ejemplos de sistemas registrales para la gestión de antecedentes penales

En la actualidad numerosos países han decidido cambiar el modo en que realizaban la gestión de los antecedentes penales, creando sistemas registrales informáticos que facilitaran el trabajo manual en las oficinas dedicadas a dicha tarea. Entre los países que emplean dichos sistemas se encuentran Chile, Perú y Venezuela. A continuación se exponen las características de los mismos:

El Registro Civil e Identificación de Chile ha creado una aplicación web que ofrece numerosos servicios relacionados con el registro civil con el objetivo de descongestionar sus oficinas territoriales y mejorar la atención a la población. Esta aplicación brinda a los ciudadanos chilenos la posibilidad de obtener toda la información que requieran para realizar sus trámites, así como los documentos necesarios para efectuar los mismos. Entre sus múltiples servicios se destaca el registro de bancos de datos personales, discapacidades, vehículos, nacimientos, defunciones, salvoconductos, matrimonios y testamentos.

Entre sus funcionalidades se encuentran además la emisión de algunos tipos de certificados y la inscripción de antecedentes penales, lo cual posibilita dejar constancia de la información remitida por los tribunales de justicia. Para disfrutar de estos servicios el usuario deberá poseer una clave especial. La implantación de este sistema no solucionaría los problemas existentes en la DAP venezolana, debido a que no es posible realizar la inscripción de los tipos documentales asociados a un proceso penal. Para cancelar los antecedentes penales de una persona es necesario presentarse personalmente ante un

notario o un organismo competente que valide la veracidad de los documentos necesarios. En el caso del registro general de condenas se inscriben las anotaciones penales que afectan a un ciudadano, abriendo un prontuario penal (documento público que da fe de la identidad de un individuo y de sus anotaciones judiciales) a las personas declaradas privados de libertad por crímenes. La realización de este último trámite no está comprendida entre los servicios brindados por la aplicación, solo se realiza una descripción general del trámite y se aclaran los documentos que deben presentarse a las oficinas competentes.

En Perú, el Poder Judicial puso en funcionamiento un sistema similar al descrito anteriormente denominado Solicitud de Certificado de Antecedentes Penales (SCAP), cuya principal funcionalidad es la emisión de un documento oficial que certifica si la persona posee o no antecedentes penales. Este sistema posibilita a los usuarios obtener su certificado de antecedentes penales y consultar sus expedientes judiciales. Para disfrutar de estos servicios el ciudadano debe presentar su documento de identidad y el comprobante de pago de la tasa correspondiente al trámite que debió abonar previamente en el Banco de la Nación, el cual se validará en el módulo web correspondiente. El sistema brinda la posibilidad al usuario de seleccionar la oficina en la que desea recoger los documentos generados en el trámite realizado. Este sistema no comprende la inscripción de documentos por lo que no resultaría conveniente su uso en la DAP venezolana.

Actualmente en la DAP se emplea un sistema informático denominado Sistema de Registro y Control de Antecedentes Penales (SIRCAP). Esta aplicación no cumple con los requisitos necesarios para abarcar todo el trabajo que se realiza en la entidad. Específicamente en el proceso de inscripción no incluye todas las áreas, por lo que el trabajo en ellas se debe realizar de forma manual. Además se limita solamente a recuperar e insertar información sin hacer una búsqueda para verificar que no se dupliquen los documentos. Estas y otras irregularidades provocan que exista en la institución la necesidad de realizar un cambio hacia otra aplicación informática que cumpla con los requisitos acordados inicialmente con el cliente para el correcto funcionamiento de la DAP.

3. Conceptos asociados a la inscripción de unidades documentales con valor jurídico

Para una mejor comprensión de las terminologías empleadas en la presente investigación, se exponen a continuación elementos conceptuales relacionados con el acto registral.

3.1. Derecho registral

Es el conjunto de sistemas, principios y normas que tienen por objeto regular la estructura orgánica de los entes estatales encargadas de registrar personas, hechos, actos, contratos derechos y obligaciones; así como la forma y modo de practicarse tales inscripciones, sus efectos y consecuencias jurídicas que se derivan de estas, orientado a darles fe y publicidad, otorgando seguridad jurídica al acto de inscripción. Es el estudio de las instituciones de carácter registral, en su procedimiento, doctrina, sistemas, instancias e instituciones que lo constituyen como una materia autónoma. (2)

3.2. Sistema registral

Sistema registral es el conjunto de normas que en un determinado país regulan las formas de publicidad de los derechos reales sobre los bienes inmuebles a través del Registro de la Propiedad, así como el régimen y organización de esta institución. Es el conjunto de normas reguladoras de la institución del Registro de la Propiedad, tanto desde un punto de vista sustantivo (valor de los asientos como forma de constitución), como desde el punto de vista formal (la organización y el régimen de registro). (3)

Los sistemas registrales se rigen por principios que regulan la organización registral y varían de un país a otro, por lo que cada territorio tiene su propio sistema con características distintivas. Venezuela posee un sistema registral que permite asegurar la veracidad de lo inscrito y su integridad, así como la legitimidad de su titular. Está basado en la Constitución de la República Bolivariana de Venezuela, la Ley de registro de antecedentes penales, el Código orgánico procesal penal y la Ley orgánica de la administración pública.

3.3. Sistema registral constitutivo

Son aquellos sistemas que no admiten la existencia de un acto si este no se ha inscrito. La inscripción resulta un acto de validez del acto jurídico, es decir, cuando los derechos de las partes nacen a partir de la inscripción y solo a partir de ello nacen los derechos y obligaciones. (2)

El acto registral que se realiza en los registros de antecedentes penales responde a personas naturales y no jurídicas. El proceso tiene particularidades que no permiten su ajuste específico a ninguno de los conceptos antes mencionados. Sin embargo la filosofía de este acto registral guarda estrecha relación con lo anteriormente expuesto y de igual forma se rige por un conjunto de leyes y normas que cada nación implanta.

3.4. Inscripción de documentos en la División de Antecedentes Penales

El proceso de inscripción es la anotación de documentos en registros públicos. El contenido de dicha anotación es considerado como cierto y produce todos sus efectos ante terceros, hasta que no se modifique o se declare de forma legal su invalidez.

En la DAP la inscripción es el proceso que permite que se registren las unidades documentales que arriban a la oficina de antecedentes penales, las cuales son enviadas por los entes judiciales del país a la entidad. Posibilita que se asienten en la división las Sentencias Definitivamente Firmes privativas de la libertad emitidas por tribunales nacionales y tipos documentales asociados, definidos por la ley, que permitan certificar y dar fe pública de los antecedentes penales de aquellos ciudadanos que cometan actos delictivos dentro del territorio nacional.

Actualmente la DAP se encuentra estructurada en cinco áreas. A continuación se exponen las actividades que se realizan en cada una de ellas y la reestructuración de las acciones realizadas durante el proceso de inscripción que se propone para mejorar el funcionamiento de dichas áreas.

4. Estructura de la DAP para la gestión de antecedentes penales.

La DAP se encuentra estructurada actualmente en cinco áreas:

Presentación: en esta área se reciben y procesan de forma manual los recaudos de los trámites de inscripción y de certificación; además se le asignan las claves a las Sentencias Definitivamente Firmes.

Procesamiento de datos: es el área en la que se asocian los metadatos de las unidades documentales mediante el Sistema de Registro y Control de Antecedentes Penales (SIRCAP).

Revisión legal: en esta área los abogados revisan las unidades documentales en busca de errores que hayan podido ocurrir durante la inscripción. Se emplea para ello el sistema informático existente, aunque no cuenta con esa funcionalidad específicamente. El sistema se utiliza para comprobar los nombres de los ciudadanos, los antecedentes que tengan registrados y de esa forma conocer que el acto de inscripción fue realizado.

Otorgamiento: es el área donde el director debe firmar los trámites de certificación realizados, dando fe legal de que la copia certificada es válida. El sistema en explotación no permite la firma de los tipos documentales inscritos.

Archivo: en esta área se asientan las unidades documentales y los recaudos de los trámites.

Para que las áreas antes mencionadas funcionen adecuadamente se hizo necesaria la reestructuración de las acciones que se realizan en cada una de ellas, quedando como se muestra a continuación:

Presentación**:** se informatiza completamente el proceso, permitiendo la recepción de documentos y asociación de metadatos generales, así como la emisión de respuestas a entes judiciales, entes penitenciarios y personas naturales. Se pueden insertar los datos iniciales de los siguientes trámites de inscripción:

- Inscripción de Sentencia Definitivamente Firme.
- Inscripción de actualización de datos del privado de libertad.
- Rectificación de Sentencia Definitivamente Firme.
- Cancelación de Sometimiento a Juicio.

Procesamiento de datos: se incorporan tres nuevos tipos de trámites. Se culmina la asociación de metadatos a cada tipo de trámite. Se añade la funcionalidad de la búsqueda de coincidencias, la cual evita que existan duplicados en los expedientes. Se digitalizan las unidades documentales y recaudos. Se añaden mejoras al proceso con la incorporación de nuevas funcionalidades que facilitan el trabajo de los funcionarios del área: ver detalles de expediente, mostrar detalles de unidad documental, ver notas, cambiar estado del trámite, gestionar delitos, gestionar persona de unidad documental, asociar unidad documental a expediente.

Revisión legal: se informatiza este proceso, el cual no estaba incluido en el sistema informático empleado. Permite que se revisen los metadatos insertados previamente los operadores. Además posibilita la revisión de las unidades documentales coincidentes, así como de los documentos a rectificar o cancelar

en formato digital. Incorpora la emisión de varias respuestas: resumen de ficha de antecedentes penales, oficios para trámites con error en datos y notas para trámites rechazados.

Otorgamiento: se informatiza este proceso, el cual no se incluía en el sistema informatizado que se empleaba. Incorpora la firma digital como elemento para proveer valor legal a los documentos digitalizados.

Archivo: se incorporan nuevas funcionalidades que constituyen tema para otro trabajo de diploma.

Para realizar el sistema informático que permita la reestructuración antes mencionada en la DAP se realizó la selección de una metodología de desarrollo de software que guiara todo el ciclo de desarrollo. A continuación se exponen los aspectos fundamentales de dicha selección.

5. Metodología de desarrollo de Software

5.1. ¿Qué es una metodología de desarrollo de software?

Para lograr obtener un producto final es necesario realizar un conjunto de tareas que requieren la definición de las herramientas, procesos y técnicas que se emplearán y los roles que participarán. Para ello surgen las metodologías de desarrollo de software, las cuales sirven de guía a los desarrolladores, indicándoles cómo organizar el trabajo para obtener los productos parciales y finales, garantizando el cumplimiento de los requisitos del cliente. (4). Las características de cada software son disímiles es por ello que ha surgido una gran variedad de metodologías, las cuales se dividen en dos grupos fundamentales:

Metodologías tradicionales o pesadas: empleadas fundamentalmente en grandes proyectos, se centran esencialmente en los procesos y tareas a realizar, generando gran cantidad de documentación. Entre las metodologías pertenecientes a este grupo se destaca el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés).

Metodologías ágiles o ligeras: se basan en la interacción con el cliente. Antepone el desarrollo del producto a la realización de documentación. Lo más importante es la capacidad de reaccionar ante los cambios que puedan ocurrir aunque esto implique incumplir el plan. Entre las metodologías pertenecientes

a este grupo se destacan Programación Extrema (XP, por sus siglas en inglés), Scrum y Proceso Ágil Unificado (AUP, por sus siglas en inglés).

La metodología seleccionada para el desarrollo fue RUP, a continuación se exponen sus características y se fundamenta su utilización en el proyecto.

5.2. Proceso Unificado de Desarrollo

RUP es una metodología de software tradicional. Un elemento clave de esta metodología es el empleo del Lenguaje Unificado de Modelado para la realización de diagramas. Posee tres características fundamentales:

- Dirigido por casos de uso: Los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductual. Los casos de uso se emplean como guía para el proceso de desarrollo y se representan en el modelo de casos de uso.

- Centrado en la arquitectura: la arquitectura sirve como base para el proceso de desarrollo. Esta debe ser reutilizable, de fácil comprensión y flexible ante los cambios que puedan ocurrir. Al comienzo del desarrollo se crea una arquitectura, la cual va evolucionando hasta que pueda ser considerada como una arquitectura estable.

- Iterativo e incremental: los grandes proyectos de desarrollo deben ser separados en pequeñas partes para lograr una implementación más sencilla. Cada una de estas partes constituye una iteración y el crecimiento del producto se refiere a la característica incremental de esta metodología. (5)

RUP identifica cuatro fases que definen el ciclo de vida del proceso de desarrollo: inicio, elaboración, construcción y transición; cada una de ellas culmina con el cumplimiento de un hito. Sobre estas fases se desarrollan nueve flujos de trabajo: seis conocidos como de procesos (modelado del negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue) y tres de soporte (gestión de configuración y cambios, gestión de proyectos y entorno).

A continuación se exponen los objetivos de cada fase:

- Inicio: tiene como principales objetivos definir el alcance del proyecto y encontrar los casos de uso críticos del sistema y los escenarios básicos que definen la funcionalidad. Además permite estimar el coste del proyecto en cuanto a recursos y tiempo, así como identificar los riesgos y las fuentes de incertidumbre.
- Elaboración: se especifican en detalle la mayoría de los casos de uso del producto. Además, se diseña la arquitectura del sistema, se desarrolla el plan del proyecto y se eliminan los mayores riesgos.
- Construcción: su finalidad es crear el producto de forma incremental a través de las sucesivas iteraciones, donde la arquitectura crece hasta convertirse en el sistema completo.
- Transición: cubre el periodo durante el cual el producto se convierte en versión beta. Su finalidad es poner el producto en manos de los usuarios finales, para lo que típicamente se requerirá desarrollar nuevas versiones actualizadas del producto. (6)

Descripción de los flujos de trabajo conocidos como de procesos:

- Modelación del negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requisitos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requisitos), por lo que indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán, la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba: busca los defectos a lo largo del ciclo de vida.
- Despliegue: produce liberaciones del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales. (7)

Se decidió emplear RUP para el diseño e implementación del módulo Inscripción de SIGESAP en común acuerdo con el equipo del proyecto RAP, debido a que es una metodología flexible, lo cual posibilita su ajuste a las características propias del proyecto. Durante el análisis, el diseño y la implementación no existió comunicación directa con el cliente, debido a que el desarrollo del software se llevó a cabo en Cuba. Por este motivo fue importante contar con una metodología robusta que permitiera generar gran cantidad de documentación que facilitara la comprensión del trabajo realizado en cada una de las fases. Otro elemento a tener en cuenta fue el hecho de que RUP está pensada para proyectos de larga duración, que incluyan la transferencia de conocimientos y artefactos generados durante la ejecución de los proyectos, lo cual se ajusta a las necesidades del proyecto RAP. Por último los desarrolladores del proyecto tienen experiencia con el uso de esta metodología y poseen una amplia documentación sobre la misma.

Una vez definida la metodología a emplear se procedió a la selección de las herramientas y tecnologías a utilizar durante el desarrollo de la solución. A continuación se muestra la fundamentación del uso de las mismas.

6. Fundamentación del uso de las principales herramientas y tecnologías.

El uso de herramientas y tecnologías se hace imprescindible para el desarrollo de un sistema informático. A continuación se especifica el lenguaje de modelado seleccionado así como la herramienta CASE. Además se expone el lenguaje de programación, el entorno de desarrollo integrado, el sistema de gestión de bases de datos y el servidor de aplicaciones a emplear. De igual forma se plantean las principales tecnologías a utilizar. De cada herramienta y tecnología seleccionada de acuerdo a las necesidades del proyecto RAP se muestran sus características y la fundamentación de su uso.

6.1. Lenguaje de modelado

Se seleccionó para modelar los artefactos propuestos por la metodología el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), el cual es un lenguaje estándar resultado del trabajo de Ivar Jacobson, Grady Booch y James Rumbaugh. Se emplea para modelar sistemas mediante diagramas sencillos y de fácil comprensión. Estos diagramas permiten obtener diferentes puntos de vista del producto que se desea implementar, facilitando así la comunicación entre los desarrolladores, los clientes y cualquier otra persona involucrada con el proceso de desarrollo.

UML ayuda a describir qué es lo que deberá hacer el sistema pero no plantea cómo se debe implementar, se podría definir como el lenguaje en el que está descrito el modelo.

Las principales funciones de UML se pueden resumir en:

- Visualizar: permite expresar de forma gráfica un sistema para facilitar su comprensión.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado, sirviendo para su futura revisión. (8)

Una de las ventajas de su utilización en el proyecto es que permite captar la información sobre la estructura estática y el comportamiento dinámico del sistema (9). Se seleccionó UML como lenguaje de modelado para el proyecto RAP debido a que facilita la comprensión de los artefactos por los diversos elementos gráficos que posee.

6.2. Herramienta CASE: Visual Paradigm 3.4

Uno de los aspectos fundamentales dentro del desarrollo de un proyecto es la correcta selección de la herramienta CASE (Ingeniería de Software Asistida por Computadora) a utilizar para su modelación, la cual permite aumentar la productividad en el desarrollo de software reduciendo el costo de la misma en términos de tiempo y de dinero.

Visual Paradigm es una herramienta CASE que brinda un entorno para la realización de diagramas con UML y soporta todo el ciclo de desarrollo. Posibilita una mejor comunicación entre los miembros del equipo ya que establece un lenguaje estándar, común para todos.

Entre sus principales características se encuentran:

- Logra la integración con diversos entornos de desarrollo como el Netbeans.
- Posibilita la realización de ingeniería inversa para Java.

- Brinda facilidades para exportar diagramas en forma de imágenes jpg, png y svg.
- Es multiplataforma.
- Permite representar todos los tipos de diagramas de clases.
- Ofrece facilidades para generar documentación en diversos formatos (.pdf,. html, .doc)
- Se encuentran disponibles múltiples versiones, cada una enfocada a necesidades específicas.
- Posibilita la generación de código (diagrama a código).
- Permite la realización de ingeniería inversa de bases de datos (desde sistemas gestores de bases de datos existentes a diagramas de entidad relación).
- Permite el modelado colaborativo.
- Generación de beans para el desarrollo y despliegue de aplicaciones.
- Permite la generación de bases de datos, transformando los diagramas de entidad relación en tablas de base de datos.
- Distribución automática de diagramas: reorganización de las figuras y conectores de los diagramas UML. (10)

Por todas las características anteriormente planteadas Visual Paradigm fue la herramienta CASE que se seleccionó para la modelación de los artefactos propuestos por la metodología. Cumple y satisface las necesidades del proyecto debido a que soporta el ciclo de vida completo del desarrollo de software, posee recursos que garantizan un alto nivel de integración con el entorno de desarrollo del proyecto y con el sistema gestor de base de datos, brinda facilidades para el trabajo colaborativo y permite el modelado de todos los artefactos propuestos por la metodología. Da soporte a varios lenguajes de modelado como UML. Es importante mencionar que al inicio del desarrollo del proyecto, en la UCI, se contaba con gran experiencia en el trabajo con dicha herramienta en su versión 3.4 y se poseían las licencias necesarias para su uso, dado que Visual Paradigm es una herramienta privativa.

6.3. Lenguaje de programación: Java

Un lenguaje de programación posee una cierta estructura sintáctica y semántica. Además, los lenguajes de programación le especifican a la computadora con qué datos debe operar, cómo debe almacenarlos o transmitirlos y qué acciones debe ejecutar ante distintas circunstancias.

De acuerdo con su nivel de abstracción los lenguajes se clasifican en: lenguaje de máquina (secuencias binarias que pueden ser entendidas directamente por la computadora), de bajo nivel (se acerca al funcionamiento de la computadora) y de alto nivel (está formado por elementos del lenguaje humano). (11)

De acuerdo a su forma de ejecución se clasifican en: compilados (requiere dos etapas: traducir el programa a código máquina y ejecutar y procesar los datos) e interpretados (simula una máquina virtual, donde el lenguaje de máquina es similar al lenguaje fuente). (12)

Java es un lenguaje de programación que posee las siguientes características:

- Simple: elimina muchas de las características desventajosas de otros lenguajes como C++ y añade otras muy útiles como el reciclado de memoria dinámica, el cual reduce los errores más comunes de programación con lenguajes como C y C++.
- Orientado a objetos: posee un conjunto de clases que se encuentran ordenadas en paquetes. (13)
- Seguro: elimina el uso de punteros con el objetivo de no usar indebidamente los recursos en memoria. La Máquina Virtual de Java (JVM, por sus siglas en inglés) verifica que el software no posea fragmentos de código ilegales (código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto) antes de ejecutarlo. Considera la seguridad como parte de su diseño.
- Interpretado y compilado: es compilado debido a que su código fuente se transforma en una especie de código máquina (bytecodes). A su vez es interpretado pues este código se ejecuta sobre una máquina virtual en la cual se convierte en código de máquina para la arquitectura donde se implemente. (14)
- Portable: un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.

- Robusto: maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria. Estas características reducen el tiempo de desarrollo de aplicaciones en Java. Estos procedimientos son realizados sin necesidad de que se le sean indicados.

Para el desarrollo de la aplicación se seleccionó Java como lenguaje de programación por las características planteadas anteriormente y por la experiencia de los desarrolladores del proyecto en dicho lenguaje. Satisface las necesidades del proyecto debido a que posee una arquitectura neutral: el software desarrollado con este lenguaje es completamente independiente de la arquitectura de la computadora donde se emplee. El intérprete de Java es el que se encarga de tomar el código desarrollado por la JVM y lo convierte en código de máquina para la arquitectura donde se ha implementado.

Para la creación de programas en Java se hace necesario el uso del Entorno de Desarrollo Java (JDK, por sus siglas en inglés), el cual provee herramientas para el desarrollo tales como compiladores, depuradores y contiene además el Entorno de Ejecución Java (JRE, por sus siglas en inglés) que constituye lo mínimo que debe contener un sistema para poder ejecutar una aplicación Java. Para el desarrollo de la aplicación se empleó la JDK en su versión 1.6.

6.4. Entorno de desarrollo integrado: NetBeans 7.0.1

Dentro de las principales herramientas para el desarrollo de un producto de software, como es el caso de la aplicación SIGESAP, se encuentra el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés). Se podría definir como un programa informático compuesto por un conjunto de herramientas de programación o un entorno de programación que ha sido empaquetado. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

NetBeans es un entorno de desarrollo integrado de código abierto y gratuito para uso tanto comercial como no comercial. Fue creado por la Sun MicroSystems en el año 2000. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Posee una plataforma de aplicaciones que posibilita a los desarrolladores crear productos rápidamente. Es un software conformado por módulos y está basado en estándares. Es compatible con numerosas tecnologías como Java Enterprise Edition y el servidor de aplicaciones Glassfish. (15)

NetBeans es un entorno de desarrollo poderoso que está apoyado por una gran comunidad en constante crecimiento; lo cual facilita la obtención de documentación y de capacitación, posibilitando que los problemas que puedan surgir durante su utilización generalmente encuentren rápida solución, siendo de gran utilidad en el proyecto. Esta comunidad ha realizado numerosos aportes como por ejemplo la creación de traducciones al IDE.

Este IDE se destaca por su rapidez, sencillez y una pre-configuración que ofrece grandes ventajas para su uso. Tiene un depurador excelente y un editor de interfaz gráfica de usuario muy avanzado que proporciona los diversos componentes Swing (conjunto de librerías con las que cuenta Java para crear y mostrar una interfaz gráfica). Netbeans es multiplataforma, lo que permite su utilización en los sistemas operativos que son empleados para el desarrollo de la aplicación SIGESAP. Soporta el desarrollo de todos los tipos de aplicaciones Java (J2SE, web, EJB y aplicaciones móviles) y su instalación es sencilla. El sistema que se desarrolla es una aplicación de escritorio para lo cual Netbeans no requiere plugins adicionales, como ocurre con otros entornos de desarrollo como Eclipse.

6.5. Sistema de gestión de bases de datos: PostgreSQL 8.4

Los Sistemas de Gestión de Bases de Datos (SGBD) son el conjunto de programas o aplicaciones que permiten la creación y el mantenimiento de la base de datos, con características primordiales como la integridad, confidencialidad y seguridad de los datos. Definen la base de datos mediante el lenguaje de definición de datos, el cual permite especificar la estructura, tipo de datos y las restricciones sobre los mismos, almacenándolo todo en la base de datos. Además permiten la inserción, eliminación, actualización y consulta de los datos mediante el Lenguaje de Manejo de Datos de forma sencilla.

PostgreSQL es un gestor de base de datos relacional de código abierto. Emplea el modelo cliente/servidor y usa multiprocesos, garantizando así la estabilidad del sistema (si falla uno de sus procesos el resto no se verá afectado y el sistema podrá continuar con su funcionamiento). Ofrece control de concurrencia de múltiples versiones, soportando la mayoría de las sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario). (16)

Entre sus características o prestaciones más destacadas se encuentran:

- Su estabilidad y la integridad que ofrece a los datos lo convierten en un gestor confiable.

- Es escalable en cuanto a la cantidad de información que se logra manejar y al número de usuarios de forma concurrente que puede alojar.
- Optimiza y planifica consultas.
- Permite realizar relaciones de herencias entre sus tablas.
- Ejecuta procedimientos en más de una docena de lenguajes como Java, C++, Python, entre otros.
- Es fácil de administrar, basada en usuarios y privilegios.
- Posee numerosos tipos de datos y brinda la posibilidad al desarrollador de definir nuevos tipos.
- El tamaño de base de datos es ilimitado (en dependencia de su sistema de almacenamiento).
- Sus opciones de conectividad abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC.
- Puede extenderse con librerías externas para soportar encriptación, búsquedas por similitud fonética (soundex), entre otras.
- Control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados y subconsultas.
- Implementación de algunas extensiones de orientación a objetos. En PostgreSQL es posible definir un nuevo tipo de tabla a partir de otra previamente definida. (17)
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Actualización integrada (pg_upgrade).
- Completa documentación. (18)

PostgreSQL es extensible, posibilitando agregar nuevos tipos de datos y funciones al servidor que se comporten como los ya incorporados. El cliente solicitó que la herramienta empleada para gestionar la base de datos fuera de código abierto, multiplataforma y sin restricciones de uso. Los requisitos antes mencionados se satisfacen con el empleo de PostgreSQL, lo cual concuerda además con la legislación venezolana vigente que plantea como política la utilización de tecnologías libres en los sistemas informáticos que se apliquen en el país.

Para administrar la base de datos se empleó pgAdmin III, versión 1.10.0. Esta herramienta de código abierto posibilita la administración de bases de datos PostgreSQL. Fue creada para trabajar en el desarrollo de grandes bases de datos y soporta todas las características del gestor seleccionado, posibilitando así una administración simple. (14)

6.6. Servidor de aplicaciones: Oracle Glassfish Open Source Server 3.1

Un servidor de aplicaciones proporciona servicios que permiten soportar la ejecución y disponibilidad de las aplicaciones desplegadas. Un servidor de aplicaciones (que cumple con los estándares mencionados en la especificación J2EE) debe tener los siguientes componentes: contenedor de aplicaciones cliente (donde se ejecutan las aplicaciones clientes), contenedor web (conocido como servidor web) y contenedor EJB (donde se ejecuta la lógica de negocio). Los servidores de aplicaciones proporcionan además facilidades para el desarrollo y despliegue de aplicaciones: seguridad, escalabilidad, gestión de acceso concurrente a los componentes y alta disponibilidad. (14)

Generalmente gestiona la totalidad o una buena parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación, permitiendo la centralización y la disminución de la complejidad. Un servidor de aplicaciones incluye normalmente los siguientes servicios:

- Agrupación de recursos (agrupación de conexiones de base de datos y agrupación de objetos).
- Administración de transacciones distribuida.
- Comunicación asincrónica de programa, normalmente a través de colas de mensajes.
- Un modelo de activación de objetos oportuno.
- Interfaces de servicios web XML automáticas para tener acceso a objetos de empresa.

- Servicios de detección de errores y estado de las aplicaciones.
- Seguridad integrada. (19)

Oracle Glassfish Open Source Server es un servidor de aplicaciones de software, que usa código de Sun Java Application Server y de la arquitectura TopLink de Oracle. Implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Su licencia es de libre uso, por lo que es gratuito y de código abierto. (20)

Para la implementación del sistema se ha seleccionado este servidor debido a su velocidad, su alta escalabilidad, el manejo centralizado de clústeres e instancias, el bajo consumo de memoria y una excelente interfaz guiada para establecer las configuraciones. Además posee una consola de administración web, la cual permite administrar el servidor de manera remota, simplificando así la tarea de administración. Proporciona pre-configuraciones para el entorno de desarrollo seleccionado siendo una excelente opción para el desarrollo del proyecto. Posee una madura implementación para la gestión de la alta disponibilidad, tolerancia a fallos, concurrencia, seguridad, y conexiones a sistemas de información externos. La implementación de Oracle Glassfish Open Source Server está soportado por una fuerte comunidad de desarrollo que le va siguiendo los pasos en cuanto a funcionalidades al Oracle Glassfish Server. (14)

6.7. Tecnologías

Para acelerar la implementación de la solución y proporcionarle robustez y escalabilidad al sistema se emplea un conjunto de tecnologías que se mencionan a continuación:

Enterprise Java Beans (EJB)

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

- Java Standard Edition (JSE): dentro de esta categoría se aglomeran todas las tecnologías existentes en Java para sistemas simples o estándares.
- Java Enterprise Edition (JEE): esta categoría agrupa todas las tecnologías que proporciona Java para la implementación de aplicaciones empresariales.

- Java Micro Edition (JME): esta categoría contiene las tecnologías que proporciona Java para aplicaciones móviles. (14)

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

Un EJB es un componente del lado del servidor que se debe desplegar sobre un contenedor de EJBs. Está conformado por diversos archivos (interfaces y clases Java). El contenedor provee a los EJBs servicios y controla su ciclo de vida. (21)

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

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

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

- Manejo de transacciones: controla las transacciones asociadas a las llamadas a los métodos del bean.
- Seguridad: controla el acceso a los métodos del bean.
- Concurrencia: llamada simultánea a un mismo bean desde múltiples clientes.
- Servicios de red: gestiona la comunicación entre el cliente y el bean en máquinas distintas.
- Gestión de recursos: gestión automática de múltiples recursos.

- Persistencia: sincronización entre los datos del bean y tablas de una base de datos.
- Gestión de mensajes: manejo de Java Message Service (JMS).
- Escalabilidad: posibilidad de constituir clúster de servidores de aplicaciones con múltiples hosts para poder dar respuesta a aumentos repentinos de carga de la aplicación con solo añadir hosts adicionales.
- Adaptación en tiempo de despliegue: posibilidad de modificación de todas estas características en el momento de despliegue del bean. (14)

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

API de persistencia de Java (JPA)

JPA es la Interfaz de Programación de Aplicaciones (API) de persistencia desarrollada para la plataforma Java Enterprise Edition. Brinda un modelo de persistencia basado en objetos comunes conocidos como POJO´s (Plain Old Java Object) para mapear las bases de datos en Java. El mapeo o relación entre entidades Java y tablas de la base de datos se realiza mediante anotaciones en las propias clases de entidad. (22)

El objetivo de esta API es unificar la forma en la que funcionan las utilidades que proveen un mapeo objeto-relacional. Su diseño persigue no perder de vista las ventajas de la orientación a objetos al interactuar con la base de datos. Se decidió emplear esta API por la integración que posee con la tecnología EJB. Existen varias implementaciones de JPA como: EclipseLink, TopLink e Hibernate. Se eligió EclipseLink debido a que es un marco de trabajo estable y muy recomendable para soluciones desarrolladas en Java y desplegadas en un servidor de aplicaciones Glassfish. (14)

7. Conclusiones del capítulo

A partir de la realización del presente capítulo “Fundamentación teórica” se puede concluir que:

- El estudio del arte realizado, mediante la exposición de conceptos relacionados a la materia registral y al proceso de inscripción, permitió un mayor entendimiento de las bases teóricas y el funcionamiento de dicho proceso. Aportó elementos para demostrar la necesidad de la construcción de un módulo Inscripción como el que propone el presente trabajo de diploma.

- El análisis de las características principales de la metodología, las herramientas y las tecnologías a emplear, permitió obtener una mayor claridad en la definición de cuáles se utilizarían en el desarrollo de la solución. Por las características propias del proyecto RAP se necesitaba una metodología robusta, siendo RUP la metodología de desarrollo de software utilizada, la cual facilitó la generación de gran cantidad de documentación para asegurar el nivel de partida y el control de cambios; unido a la elección de UML como lenguaje de modelado, que brindó la posibilidad de obtener el plano del sistema, además permitió realizar la verificación y validación del modelo realizado y una mayor rigurosidad en la especificación. La herramienta CASE seleccionada para el modelado es Visual Paradigm en su versión 3.4, la cual propició el trabajo en diversas plataformas y posibilitó el modelado con el lenguaje UML.

- La elección de Netbeans 6.9.1 como entorno de desarrollo integrado posibilitó la integración con el lenguaje de programación Java y con la biblioteca gráfica Swing. Además, el sistema implementado es una aplicación de escritorio para lo cual Netbeans no requiere plugins adicionales. La selección de PostgreSQL en su versión 8.4 como sistema gestor de base de datos facilitó el manejo de grandes volúmenes de datos y garantizó la estabilidad del sistema. La elección del servidor de aplicaciones Oracle Glassfish Open Source Server en la versión 3.1, el cual implementa las tecnologías definidas en la plataforma Java EE, permitió ejecutar aplicaciones que siguen esta especificación y brindó una alta escalabilidad y velocidad. En la selección de estas herramientas se tuvo en cuenta que la aplicación debía ser multiplataforma y se dio cumplimiento a los acuerdos firmados con el cliente referente a la utilización de herramientas libres.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

1. Introducción

Este capítulo expone una descripción general de la propuesta del sistema para lograr una mejor comprensión de su funcionamiento mediante la explicación de los procesos a informatizar. Enuncia los requisitos funcionales y no funcionales identificados durante la fase de Requisitos. Resume los casos de uso del módulo y muestra una descripción de la arquitectura del sistema. Analiza los patrones empleados para el diseño. Presenta el diagrama general de paquetes, los diagramas de clases del diseño, la realización de casos de uso del diseño, el modelo de datos y el modelo de despliegue. Además muestra una vista de la implementación a través del diagrama de componentes.

2. Propuesta del sistema

Teniendo en cuenta el análisis del diagnóstico realizado en la DAP y los requisitos acordados con el cliente para el establecimiento de un nuevo sistema se realizó la propuesta de organización de los diferentes procesos para su posterior implementación. Para una mejor comprensión del flujo de trabajo y de la propuesta de solución se han representado las actividades que se realizan en cada una de las áreas de la DAP en el proceso de “Inscribir Sentencia Definitivamente Firme” (Fig. 1). A continuación se describe dicho proceso:

Descripción del proceso “Inscribir Sentencia Definitivamente Firme”.

El proceso de inscripción de Sentencia Definitivamente Firme (SDF) comienza en el área de Presentación. En dicha área se recibe la SDF y se registran los primeros metadatos en el sistema. El proceso continúa en el área de Procesamiento de datos donde se realiza la búsqueda de coincidencias para evitar la duplicación del documento a inscribir, la inscripción de las personas relacionadas en la SDF así como sus datos procesales correspondientes, la digitalización del documento y la asociación de los metadatos digitalizados a un expediente en caso de existir la causa. A continuación en el área de Revisión legal el funcionario abogado es el encargado de comprobar que no existan errores en el proceso de inscripción de SDF. Para poder realizar su labor el abogado podrá observar los metadatos de la SDF insertados

anteriormente, consultar la SDF digitalizada, verificar las coincidencias documentales, las personas y sus datos procesales correspondientes y la asociación a expediente. Seguidamente se emite una respuesta acorde con los resultados de las verificaciones mencionadas anteriormente. En el área de Otorgamiento el funcionario director puede realizar una revisión de todo el proceso si lo desea. Su principal función es otorgar el trámite y firmar las imágenes digitalizadas. Finalmente se asienta en el área de Archivo la Sentencia Definitivamente Firme.

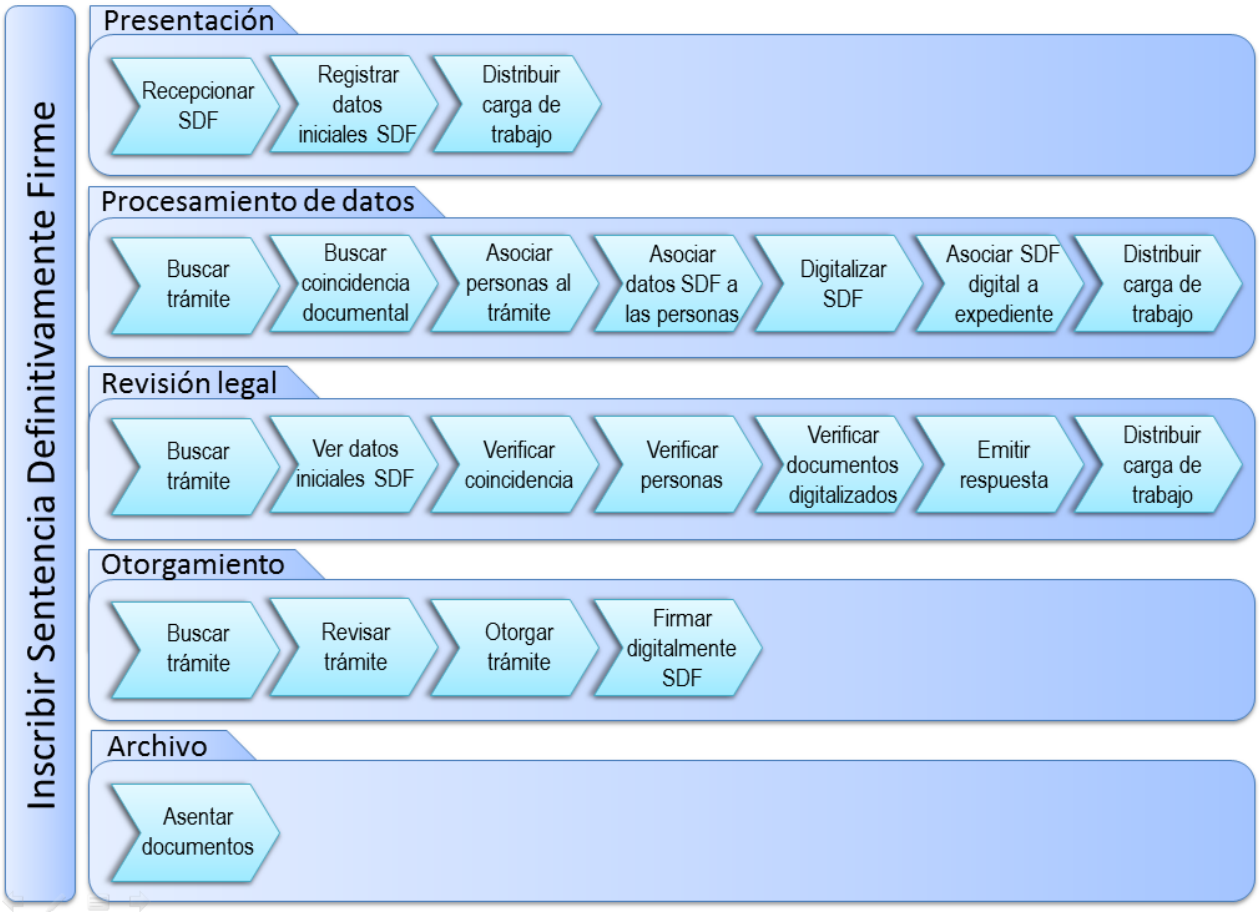

Fig. 1 Representación del proceso "Inscribir Sentencia Definitivamente Firme".

Con el objetivo de mejorar los procesos que se realizan durante la inscripción de documentos en la DAP los especialistas del proyecto identificaron los requisitos funcionales y no funcionales que deben ser incorporados al nuevo sistema informático. Dichos requisitos constituyen la base para el posterior diseño e implementación de la solución a desarrollar y se muestran a continuación.

3. Requisitos de software

Un requisito de software es un atributo necesario en un sistema, una instrucción que identifica una capacidad, característica o cualidad de un sistema con el fin de que tenga valor y utilidad para el cliente y solucione un problema particular. Posee gran importancia debido a que proporciona la base para el desarrollo. (23)

Los requisitos deben poseer las siguientes características: posibles, necesarios, priorizados, concretos y verificables (24). A continuación se especifican los requisitos funcionales y no funcionales del sistema a implementar.

Requisitos funcionales: son condiciones o capacidades que el sistema debe alcanzar para solucionar el problema planteado por el usuario. Describen lo que el sistema debe cumplir para satisfacer el contrato u otro documento formal que plantee lo acordado con el cliente (25).

A continuación se enuncian los 22 requisitos funcionales identificados:

RF_I.01. Insertar datos iniciales de la Sentencia Definitivamente Firme (SDF).

RF_I.02. Generar número de trámite de inscripción.

RF_I.03. Distribuir carga de trabajo por áreas.

RF_I.04. Mostrar resumen de trámite de inscripción.

RF_I.05. Insertar datos iniciales para actualizar datos del privado de libertad.

RF_I.06. Insertar datos iniciales para cancelar Sometimiento a Juicio.

RF_I.07. Insertar datos iniciales para rectificar Sentencia Definitivamente Firme.

RF_I.08. Mostrar listado de trámites de inscripción.

RF_I.09. Buscar trámites de inscripción.

RF_I.10. Modificar datos iniciales de la Sentencia Definitivamente Firme.

RF_I.11. Adicionar antecedentes penales.

RF_I.12. Mostrar vista previa antecedentes penales agregados.

RF_I.13. Actualizar datos del privado de libertad.

RF_I.14. Mostrar detalles del trámite de actualización de datos del privado de libertad.

RF_I.15. Modificar antecedentes penales agregados.

RF_I.16. Mostrar datos generales de la unidad documental.

RF_I.17. Mostrar datos procesales del privado de libertad.

RF_I.18. Emitir oficio del trámite de inscripción en curso.

RF_I.19. Anexar documento a expediente.

RF_I.20. Mostrar documentos probatorios digitalizados.

RF_I.21. Otorgar trámites de inscripción.

RF_I.22. Digitalizar tipos documentales asociados a trámites de inscripción. (26)

Requisitos no funcionales: son propiedades o cualidades que el sistema debe tener. Definen propiedades o restricciones del sistema. Pueden ser clasificados en: requisitos de funcionamiento, de mantenimiento, de seguridad, de confiabilidad, de usabilidad, de fiabilidad, de eficiencia, de interfaz (25).

A continuación se enuncian los 16 requisitos no funcionales identificados:

RNF_I.01. Cumplir con las pautas de diseño de las interfaces.

RNF_I.02. Agrupar vínculos y botones por grupos funcionales.

RNF_I.03. Mostrar los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema en idioma español.

RNF_I.04. Utilizar campos de selección en la interfaz en los casos que sea posible.

RNF_I.05. Usar combinaciones de teclas para agilizar la interacción del usuario con el sistema.

RNF_I.06. Prever contingencias para caída del sistema.

RNF_I.07. Responder en un tiempo inferior a los 3 segundos las peticiones que se realicen en el sistema.

RNF_I.08. Garantizar acceso concurrente para todos los usuarios del sistema.

RNF_I.09. Desarrollar el sistema para un entorno de escritorio.

RNF_I.10. Diseñar el sistema siguiendo las pautas definidas por el Ministerio del Poder Popular para las Telecomunicaciones y la Informática.

RNF_I.11. Integrar ayuda al sistema.

RNF_I.12. Restringir el acceso a la información contenida en los servidores de bases de datos desde las estaciones de trabajo.

RNF_I.13. Almacenar acciones de los usuarios sobre el sistema.

RNF_I.14. Instalar en las estaciones de trabajo el software necesario para el correcto funcionamiento del sistema.

RNF_I.15. Proporcionar características mínimas de hardware a las estaciones de trabajo.

RNF_I.16. Proporcionar características mínimas de hardware a los servidores. (27)

Para darle cumplimiento a los requisitos antes expuestos se definió un conjunto de casos de uso, los cuales se muestran a continuación.

4. Definición de los casos de uso

El módulo a implementar garantizará la informatización de todos los procesos de inscripción propuestos para la gestión de los diferentes tipos documentales. La solución está basada en la realización de 10 casos de uso, así como en la utilización de otros 10 casos de uso del módulo de Integración. A

continuación se muestra el diagrama de actores, la descripción de las funcionalidades por cada actor, los diagramas de casos de uso y la explicación general de los casos de uso del módulo Inscripción.

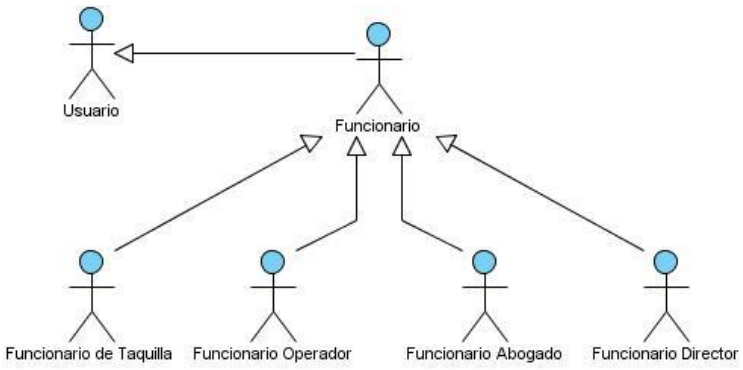

Fig. 2 Diagrama de actores del módulo Inscripción.

- Usuario: usuario que accede al sistema para realizar las funcionalidades a las que tiene permiso. Debe autenticarse en el sistema y configurar su perfil de usuario.
- Funcionario: actor genérico que incluye a los funcionarios de la DAP que pueden solicitar expedientes en el Archivo.
- Funcionario de Taquilla: funcionario que se encarga de la recepción de los tipos documentales que se almacenan en la DAP y la devolución de las respuestas emitidas por la entidad.
- Funcionario Operador: funcionario que se encarga del proceso de asociación de metadatos y digitalización de los trámites de inscripción de tipos documentales.
- Funcionario Abogado: funcionario que encarga de la revisión legal de los diferentes trámites que se llevan a cabo en la DAP.
- Funcionario Director: funcionario jefe de la DAP que se encarga del otorgamiento de los trámites que se llevan a cabo en la misma así como la firma digital de los tipos documentales almacenados en el sistema como objetos digitales.

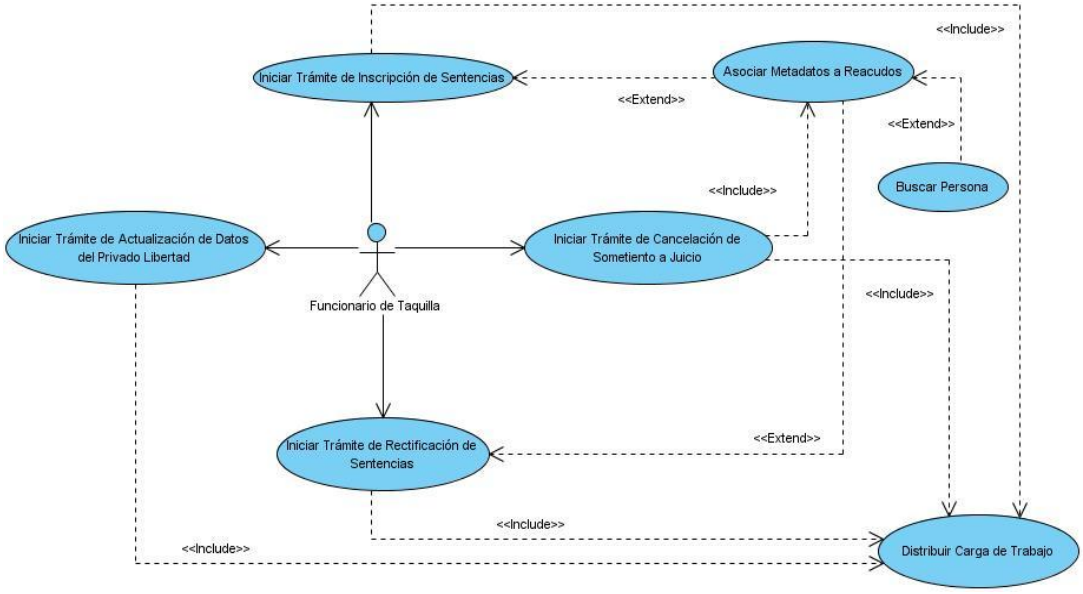

Fig. 3 Diagrama de casos de uso (actor Funcionario de Taquilla).

Iniciar trámites de inscripción de Sentencia Definitivamente Firme:

El caso de uso inicia cuando el funcionario de taquilla selecciona la opción “Iniciar trámites de inscripción de Sentencia Definitivamente Firme”, a continuación introduce los datos requeridos para iniciar la inscripción de la Sentencia Definitivamente Firme. El funcionario puede además asociar un trámite de certificación a la inscripción de sentencia en curso. El sistema le asigna un número de trámite y muestra el resumen de los datos insertados, terminando de esta forma el caso de uso.

Iniciar trámites de rectificación de Sentencia Definitivamente Firme:

El caso de uso inicia cuando el funcionario de taquilla necesita introducir en el sistema los datos iniciales para realizar un trámite de rectificación de Sentencia Definitivamente Firme. La funcionalidad permite recoger los datos generales del tipo documental a rectificar y muestra además el resumen de los datos insertados.

Iniciar trámites de actualización de datos del privado libertad:

El caso de uso se inicia cuando llega a la DAP una unidad documental de actualización de datos del privado de libertad proveniente de un ente penitenciario. El funcionario de taquilla introduce los datos

generales del trámite, el sistema le asigna un número de trámite y muestra el resumen de los datos insertados.

Iniciar trámites de cancelación de Sometimiento a Juicio:

El caso de uso consiste en iniciar el trámite de cancelación con motivo de anular la información existente en la DAP con respecto a un Sometimiento a Juicio sin respuesta definitivamente firme por parte del tribunal, por lo cual se invalida al ciudadano de la obtención de una certificación de no poseer antecedentes penales. El caso de uso concluye al recopilarse toda la información general del trámite más los documentos absolutorios o condenatorios suministrados por el ciudadano, y con el estado del trámite establecido como iniciado.

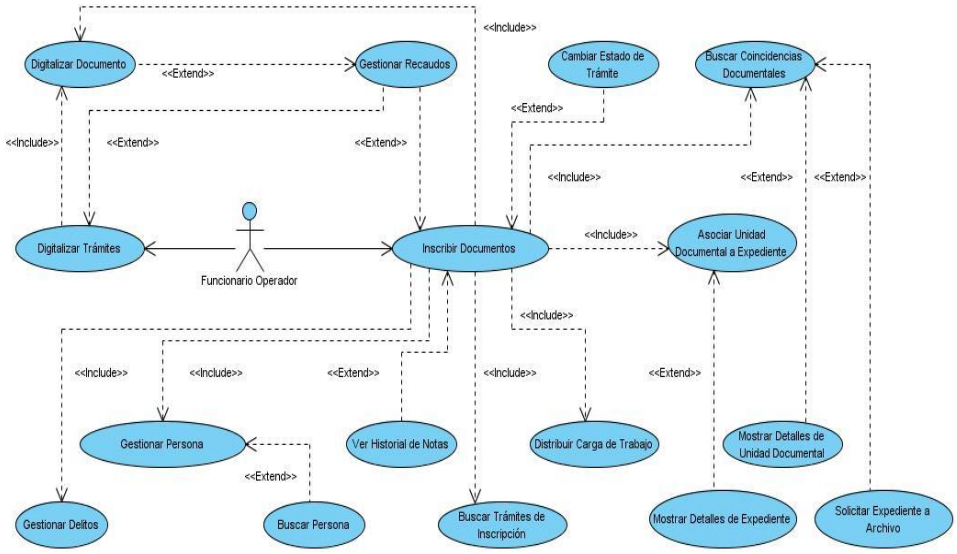

Fig. 4 Diagrama de casos de uso (actor Funcionario Operador).

Inscribir documentos:

El caso de uso inicia cuando el funcionario operador selecciona la opción “Inscribir documentos”, luego de mostrados los trámites de inscripción en curso debe seleccionar uno, seguidamente puede buscar coincidencia documental para no incurrir en la duplicación de los documentos a inscribir. En dependencia del tipo de trámite seleccionado podrá además ver los datos generales del mismo, así como las personas involucradas, sobre las cuales se ejecutarán las acciones antes mencionadas, además de poder ver sus

datos personales y procesales. Luego de terminar de registrar los datos del trámite en curso se procede a la digitalización de los documentos involucrados. El caso de uso termina con la asociación del tipo documental que se está inscribiendo al expediente correspondiente.

Digitalizar unidad documental del trámite:

El caso de uso inicia cuando el funcionario operador selecciona la opción “Digitalizar unidad documental del trámite”, busca mediante una serie de criterios los trámites a digitalizar. Una vez seleccionado el trámite se procede a digitalizar la unidad documental de dicho trámite.

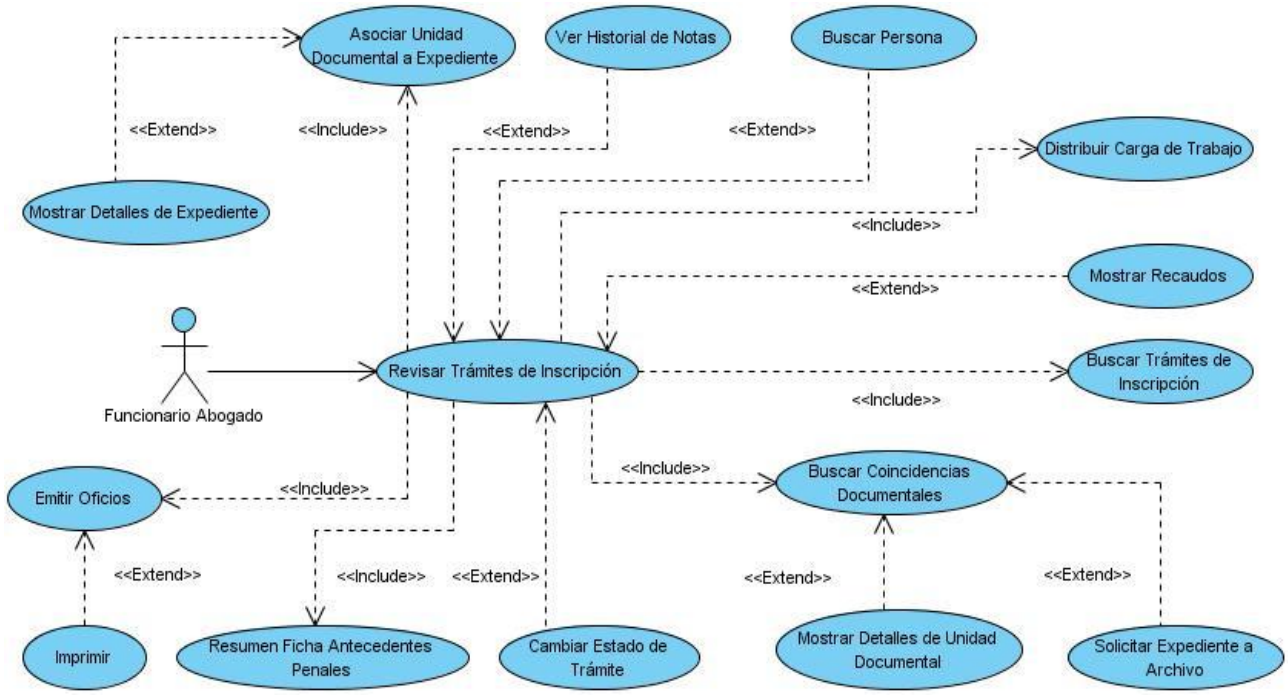

Fig. 5 Diagrama de casos de uso (actor Funcionario Abogado).

Revisar trámites de inscripción:

El caso de uso inicia cuando el funcionario abogado selecciona la opción “Revisar de Trámites de Inscripción”, luego de mostrados los trámites de inscripción en curso debe seleccionar uno, seguidamente puede buscar coincidencias documentales para evitar la duplicación de los documentos a inscribir. En dependencia del trámite seleccionado podrá observar los datos generales del mismo, así como las

personas involucradas que podrán ser verificadas, además de poder ver sus datos personales, procesales y la ficha resumen. El funcionario abogado podrá seguidamente observar las unidades documentales asociadas al trámite en formato digital, corregir la asociación a expedientes realizadas por el funcionario operador, para ello podrá ver además el expediente digital. El caso de uso termina con la emisión e impresión del oficio correspondiente al trámite.

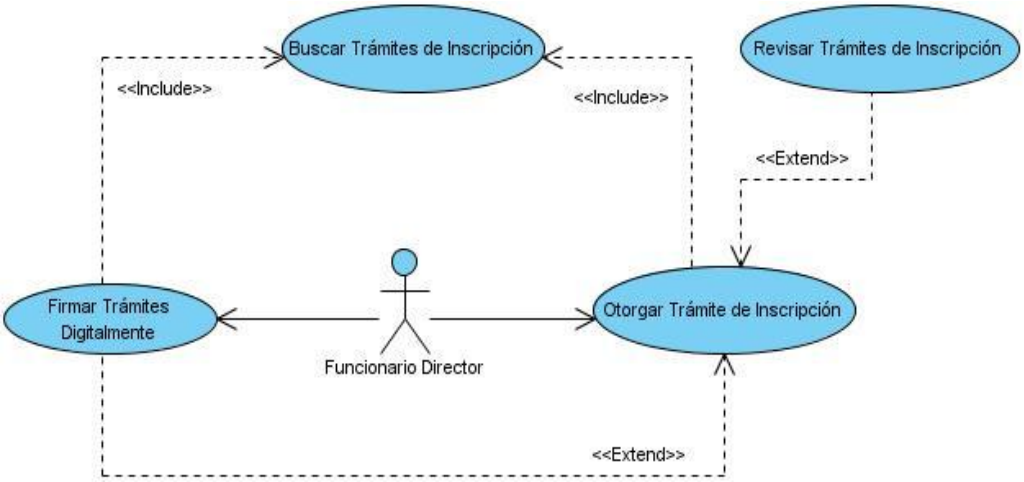

Fig. 6 Diagrama de casos de uso (actor Funcionario Director).

Otorgar trámites de inscripción:

El caso de uso inicia cuando el funcionario director selecciona la opción “Otorgar trámites de inscripción”, seguidamente introduce una serie de filtros de búsqueda en el sistema y se muestran los trámites que coinciden con los datos proporcionados que están listos para ser otorgados, donde el funcionario director puede revisar el trámite seleccionado u otorgarlo. Además este funcionario puede firmar el trámite digitalmente una vez otorgado.

Firmar trámites de inscripción:

El caso de uso se inicia cuando el actor funcionario director necesita firmar digitalmente los documentos escaneados asociados a los trámites de inscripción de documentos que se almacenarán en la DAP luego de un proceso de revisión legal y de haber sido otorgados por el propio funcionario director. Este funcionario introduce una serie de filtros de búsqueda en el sistema y se muestran los trámites que

coinciden con los datos proporcionados y que están listos para ser firmados digitalmente. El caso de uso termina con la aplicación de la firma digital de los documentos seleccionados.

Buscar trámites de inscripción:

El caso de uso inicia al ser invocado por uno de los siguientes casos de uso: “Inscribir documentos”, “Revisar trámites de inscripción”, “Otorgar trámites de inscripción”, “Firmar trámites de inscripción” o “Digitalizar unidad documental del trámite”, debido a la necesidad del actor de buscar los trámites para realizar las distintas funciones que él desempeña. El sistema mostrará los trámites asignados al funcionario, el cual podrá filtrar por diferentes criterios de búsqueda. El caso de uso finaliza cuando el funcionario selecciona el trámite de inscripción a realizar.

5. Arquitectura de software

Para el desarrollo de la solución se hace necesario definir una arquitectura de software que permita describir los diferentes componentes de la misma y la forma en la que estos interactúan. Además la arquitectura contribuye a elaborar detalladamente las actividades del diseño. Producto a la gran magnitud de la solución y a modo de mejorar la escalabilidad y soporte del sistema se definen 3 tipos de estilos arquitectónicos:

Arquitectura cliente – servidor: este estilo arquitectónico se basa en el fraccionamiento entre dos entidades denominadas cliente y servidor. Generalmente la mayor parte del trabajo se realiza en el servidor, mientras que el cliente solo se encarga de interactuar con el usuario, permitiendo así separar las responsabilidades de cada entidad lo cual facilita el entendimiento del diseño del sistema. Permite incrementar la flexibilidad, la escalabilidad y la interoperación de los sistemas.

Arquitectura en n capas: Las capas se ocupan de la división lógica de componentes y funcionalidades. Ayudan a diferenciar entre los distintos tipos de tareas que deben realizar los componentes, ofreciendo un diseño que maximiza la reutilización y el mantenimiento. Aplica el principio de separación de responsabilidades dentro de una arquitectura (28). Una de las características de este estilo arquitectónico es que cada capa se comunica con su capa inferior y devuelve los resultados a su superior, además las capas son independientes entre sí. Se separa la capa de presentación (es la capa con la que interactúa el usuario), de la lógica del negocio (recibe solicitudes de la capa de presentación y presenta los resultados

que obtiene de la capa de acceso a datos,) y del acceso a datos (ayuda a enlazar la lógica del negocio con el sistema gestor de base de datos); facilitando así la realización de cambios en la arquitectura: cuando se modifique una capa en específico el resto podrá permanecer sin alteraciones.

Arquitectura basada en componentes: Un componente es una unidad de composición de aplicaciones software que posee un conjunto de interfaces y de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (29). Este estilo arquitectónico descompone el diseño en componentes funcionales o lógicos, lo cual proporciona un mayor nivel de abstracción del principio orientado a objetos.

Por lo anteriormente expuesto se muestra a continuación una representación de la arquitectura de la aplicación SIGESAP:

Fig. 7 Representación de la arquitectura. Entidad cliente.

Se emplea una arquitectura cliente servidor, distribuida en n capas debido a la existencia de una aplicación cliente que contendrá las capas de presentación y de negocio local. La capa de presentación está destinada a la interacción con el usuario y se encuentra conformada por las acciones y formularios. La capa del negocio local es la encargada de desarrollar la lógica que responde al cliente y realiza las llamadas a la lógica de negocio servidora.

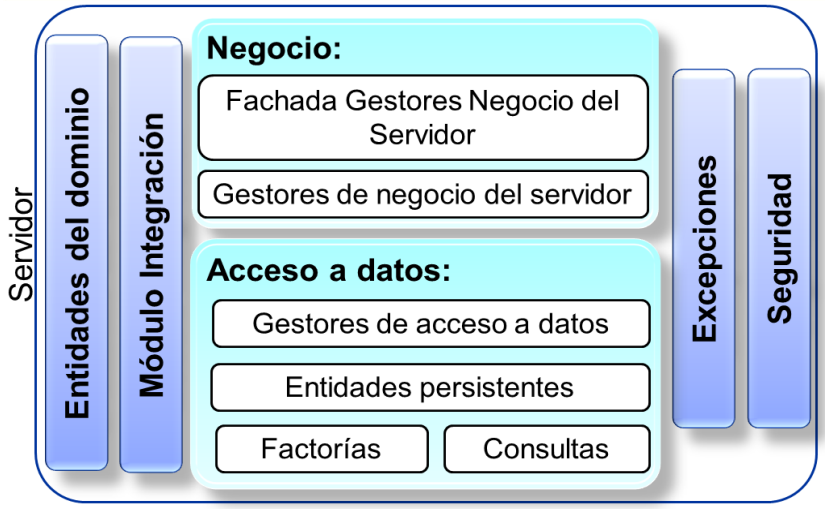

Fig. 8 Representación de la arquitectura. Entidad servidor.

La parte del servidor estará compuesta por la capa de lógica de negocio, desarrollada mediante los EJB de sesión que contienen los gestores del servidor y sus fachadas remotas. Además se encuentra la capa de acceso a datos desarrollada mediante los EJB de entidad. Dicha capa está conformada por sus gestores, las entidades persistentes, factorías y consultas, los cuales son los encargados de realizar las peticiones de datos al servidor de base de datos.

Verticalmente existen diferentes elementos interactuando con cada una de las capas, tanto en el cliente como en el servidor. Estos elementos son las entidades del dominio, el módulo Integración, las excepciones y la seguridad.

En la DAP funcionará el subsistema Centro de Digitalización (DIGIDAP), encargado gestionar la informatización del fondo documental que se encuentra archivado desde los inicios de la institución hasta la fecha en que se ponga en marcha el subsistema SIGESAP. El módulo Integración engloba las funcionalidades comunes entre los dos subsistemas que son desarrollados para la DAP, permitiendo que las soluciones desarrolladas para las actividades que cumplen con las mismas reglas del negocio sean reutilizables.

Los EJB's son componentes del lado del servidor. Debido a su utilización para la implementación de la lógica de negocio del servidor, se plantea que la arquitectura está basada en el desarrollo de

componentes. Los gestores de negocio del servidor, las factorías y los gestores de acceso a datos son componentes EJB's, creados para encapsular la lógica del negocio que se repite; permitiendo utilizar las funcionalidades implementadas mediantes las interfaces que cada uno de ellos expone.

Después de analizar los requisitos, realizar la definición de los casos de uso y la elección de la arquitectura del sistema, se cuenta con las condiciones necesarias para proceder a la realización de los distintos artefactos del diseño propuestos por la metodología.

6. Diseño del sistema

La mayoría de los sistemas, incluso los más pequeños, se deben diseñar antes de su implementación a fin de evitar revisiones costosas debido a errores de diseño. Los modelos visuales simplifican la comunicación del diseño y son utilizados como entrada esencial para las actividades en implementación y pruebas.

6.1. Estándares de diseño

Los estándares de diseño ayudan a facilitar un entendimiento común entre los involucrados en el desarrollo del software y garantizan una mejor organización. En el proyecto SIGESAP se adoptaron los siguientes estándares de diseño:

- El esquema de nombres es una de las ayudas más importantes para entender el flujo lógico de una aplicación. Un nombre debe más bien expresar el "qué" y no el "cómo".
- La nomenclatura se debe definir en español.
- Los paquetes deberán comenzar con el nombre del sistema (SIGESAP) seguido por el subsistema, luego el módulo y por último la capa lógica a la que pertenece, quedando de la siguiente forma Sigesap.Subsistema.Módulo.Capa. Para el caso de los paquetes se usará la misma nomenclatura definida para los métodos de clases.
- Las clases persistentes comenzarán con las siglas definidas en la base de datos para cada tabla.
- Los formularios de tipo popup comienzan con las siglas Frm, y los de tipo panel con las siglas Pnl.
- Los gestores de negocio del cliente comienzan con las siglas GestorNC.

- Las clases de la lógica de negocio de acceso a datos comienzan con las siglas GestorAD.
- Las clases de la lógica de negocio del servidor comienzan con las siglas GestorNS.
- Las interfaces publicadas en el servidor terminan con el sufijo Remotos.
- Las clases interceptoras de validación terminan con el prefijo Interceptor.
- Las excepciones terminan con el sufijo Exception.
- Las relaciones entre clases emplean el estereotipo <<use>>.
- Las relaciones entre clases e interfaces emplean el estereotipo <<realize>>.
- Las relaciones entre clases y paquetes emplean el estereotipo <<access>>.

Para los componentes de los formularios se empleará la siguiente nomenclatura:

- Los botones (JButton) comienzan con las siglas btn.
- Los campos de texto (JTextField) comienzan con las siglas tfd.
- Los campos fecha (Date) comienzan con las siglas fch.
- Para el componente JDateChooser se emplean las siglas dtc.
- Para el componente JDayChooser se emplean las siglas dyc.
- Los componentes de tipo Jscrool usan las siglas sc.
- Las cajas de texto (JComboBox) comienzan con las siglas cmb.
- Las áreas de texto (JTextArea) comienzan con las siglas txt.
- Los labels (JLabel) comienzan con las siglas lbl.
- Los cuadros de chequeo (JCheckBox) comienzan con las siglas chb.
- Los radiobutton (JRadioButton) comienzan con las siglas rbt.

- Las tablas (JTable) comienzan con las siglas tbl. (30)

6.2. Patrones para el desarrollo de software

Para el diseño del módulo Inscripción se hace necesario el uso de patrones que describan la solución a problemas comunes durante esta fase.

Christopher Alexander da la siguiente definición de patrón: “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”. Un patrón de diseño es la descripción de objetos que se comunican y clases que son adaptadas para solucionar un problema general de diseño en un contexto particular. El patrón de diseño identifica las clases y las instancias que participan, sus roles, la colaboración y la distribución de responsabilidades. Cada patrón se centra en un problema de diseño en particular. Entre los elementos esenciales que debe poseer un patrón se encuentran: el nombre, el problema que describe cuándo aplicar el patrón, la solución y las consecuencias o resultados de aplicar el patrón. Debido al gran número de patrones de diseño existentes, estos se han clasificado en tres grupos fundamentales: patrones de creación, estructurales y de comportamiento. (31)

Existen diferentes categorías de patrones para el desarrollo de software, a continuación se caracterizan los utilizados en la aplicación:

- Patrones de diseño: aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Normalmente son referenciados como la Grupo de los Cuatro o simplemente GoF por sus siglas en inglés (Gang of Four).

- Patrones Generales de Software para Asignación de Responsabilidades (GRASP): aquellos que expresan los principios fundamentales de la asignación de responsabilidades a objetos durante la programación orientada a objetos.

6.2.1. Patrones de diseño de software

A continuación se describe los patrones GoF implementados en la solución:

Patrones de creación: muestran una guía de cómo crear objetos cuando su creación depende de tomar decisiones. Plantean cómo se deben estructurar y encapsular estas decisiones, las cuales normalmente se resolverán de forma dinámica, decidiendo qué clases se deben instanciar o sobre qué objetos se delegarán responsabilidades. Abstraen el proceso de instanciación, oculta como los objetos son creados. (32)

- Instancia única o singleton: utilizado para asegurar que cada clase solo sea instanciada una vez, además proporciona un único punto de acceso global a la misma. El empleo de este patrón se puede observar en los gestores de negocio y en la clase de internacionalización de idioma.

Patrones estructurales: describen cómo las clases y objetos se combinan para formar estructuras más complejas y realizar nuevas funcionalidades. (31)

- Fachada o facade: ofrece una interfaz unificada sencilla, empleada para simplificar el acceso a objetos que se encuentren relacionados, proporcionando un único objeto con el que los elementos externos pueden interactuar. Es utilizado para la comunicación de los gestores del cliente y el servidor mediante el uso del gestor “FachadaGestoresRemoto”. Promueve un acoplamiento débil entre el subsistema y sus clientes al reducir las dependencias.

Patrones de comportamiento: se emplean para organizar, manejar y combinar comportamientos. Estos patrones describen la asignación de responsabilidades entre los objetos. No solo muestran los patrones de objetos y clases, incluyen además los patrones de comunicación entre ellos. (31)

- Iterador o iterator: utilizado en la capa de acceso a datos para recorrer de forma secuencial los elementos de un objeto, sin necesidad de conocer su estructura interna.

6.2.2. Patrones Generales de Software para la Asignación de Responsabilidades

El uso de estos patrones es recomendable en el diseño de software. Se aplican durante la elaboración de los diagramas de interacción, al asignar responsabilidades a los objetos y al diseñar la colaboración entre ellos. (33). La arquitectura definida propicia el empleo de los GRASP, mediante las clases gestoras del cliente, del servidor y del acceso a datos. Dichos patrones se enuncian a continuación:

- Experto: este patrón posibilita asignar la responsabilidad a la clase que cuenta con la información necesaria para cumplirla; facilitando la comprensión del sistema, su mantenimiento y la reutilización de componentes. Es utilizado en los gestores del cliente, del servidor y de acceso a datos, los cuales trabajan con la información de las entidades de dominio que representan. De este modo se obtendrá un diseño con mayor cohesión y la información se mantendrá encapsulada (disminución del acoplamiento).

- Creador: este patrón asigna responsabilidades que se relacionan con la creación de objetos. El objetivo de este patrón es encontrar al creador que se debe relacionar con el objeto que se produce. Su utilización contribuye al bajo acoplamiento, facilitando mantenimiento y ofreciendo oportunidades para la reutilización. Este patrón es utilizado en las factorías.

- Bajo acoplamiento: este patrón logra una dependencia escasa entre las clases. El acoplamiento es una medida de la fuerza con que una clase se relaciona con otras. Su utilización facilita el diseño de clases más independientes, lo cual posibilita que se adapten mejor a los cambios. Los estilos arquitectónicos n capas y basado en el desarrollo de componentes favorece el bajo acoplamiento, facilitando el entendimiento de las clases por separado y la reutilización. Es importante mencionar que el grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión.

- Alta cohesión: este patrón se encarga de mantener las clases relacionadas, facilitando que ninguna realice un trabajo enorme. La cohesión es la medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase. La alta cohesión consiste en garantizar que una clase colabore con otras para compartir el trabajo si la tarea es grande. Su utilización permite comprender fácilmente el diseño, simplifica el mantenimiento y aumenta la capacidad de reutilización. Hace de los componentes del sistema un conjunto bien acoplado.

- Controlador: posibilita la asignación de responsabilidades para controlar el flujo de eventos del sistema a clases específicas. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario enviándolo a las distintas clases según el método llamado.

6.3. Diagrama general de paquetes.

El diagrama de paquetes del diseño refleja cómo el sistema está dividido en agrupaciones lógicas y las relaciones de dependencias entre estas. Los paquetes están normalmente organizados para maximizar la coherencia interna y minimizar el acoplamiento externo entre los mismos. A continuación se muestra el diagrama general de paquetes y una breve explicación de los paquetes que por su relevancia en el módulo Inscripción constituyen los más importantes.

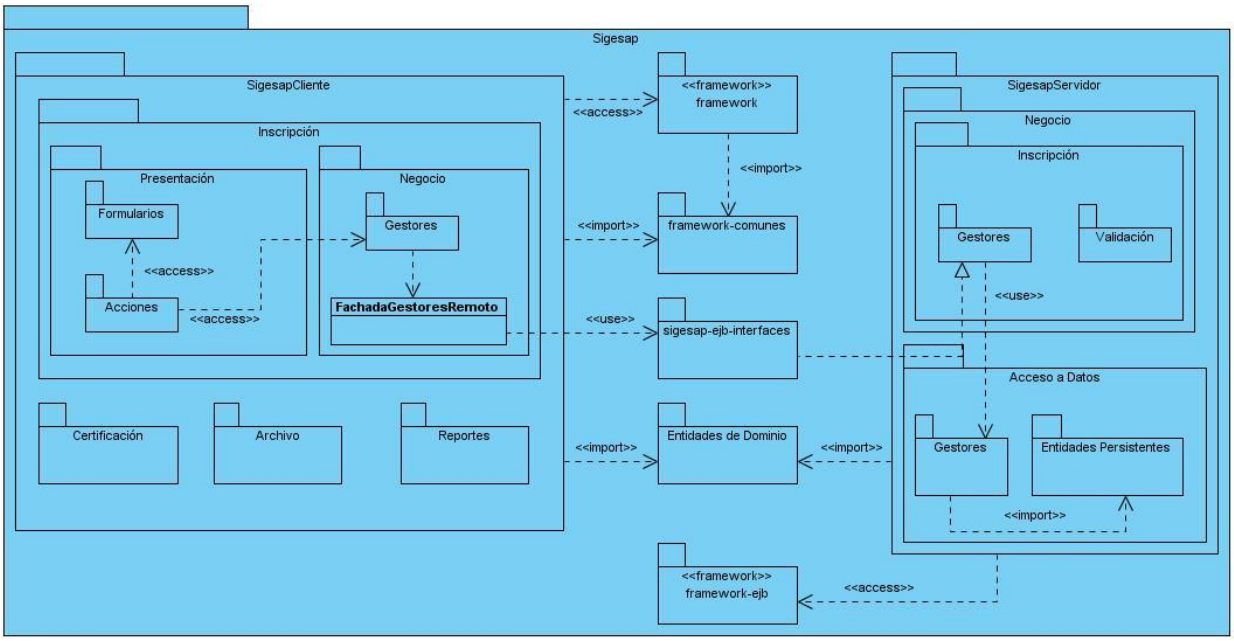

Fig. 9 Diagrama general de paquetes.

Paquetes del subsistema SigesapCliente:

- SigesapCliente: se encuentran los diferentes módulos que conforman la solución.
- Inscripción: se encuentran los paquetes de presentación y negocios del módulo Inscripción.
- Presentación: se almacenan los paquetes que contienen los formularios y las acciones que son utilizados en la presentación. El primero contiene las interfaces de usuarios y el segundo la implementación de las acciones del módulo.

- Negocio: se encuentran los gestores de apoyo a la implementación del negocio cliente que son las encargadas de comunicarse con la clase FachadaGestoresRemoto, para así comunicarse con la lógica del servidor.

Paquetes del subsistema SigesapServidor:

- SigesapServidor: contiene los paquetes que conforman el servidor de la aplicación.
- Negocio: se realiza la lógica de negocio del servidor del módulo Inscripción.
- Inscripción: se encuentran los gestores del negocio del servidor y los archivos remotos de estos que publican los métodos a los que se acceden desde el cliente.
- Acceso a datos: contiene todos los gestores y entidades persistentes que permite conectarse a la base de datos y realizar transacciones a la misma.

6.4. Diagramas de clases del diseño.

El diagrama de clases del diseño describe de forma gráfica las especificaciones de las clases de software y de las interfaces en una aplicación. Entre los elementos que lo componen se encuentran: clases, asociaciones, atributos, interfaces con sus operaciones, métodos e información sobre los tipos de atributos. (34)

A continuación se muestra el diagrama de clases del diseño del caso de uso “Inscribir documentos”. Por el tamaño del diagrama y a modo de mejorar su visualización se divide en dos partes lógicas cliente y servidor; la primera va a contener todas las acciones, paneles, gestores del cliente y la clase FachadaGestoresRemotos; la segunda incluye también la clase FachadaGestoresRemotos, los gestores del servidor, los gestores de acceso a datos y el paquete de entidades persistentes. La FachadaGestoresRemotos se representa en ambas imágenes por ser el punto de referencia común entre ellas. Los restantes diagramas pertenecientes al módulo Inscripción se encuentran en el Expediente del proyecto.

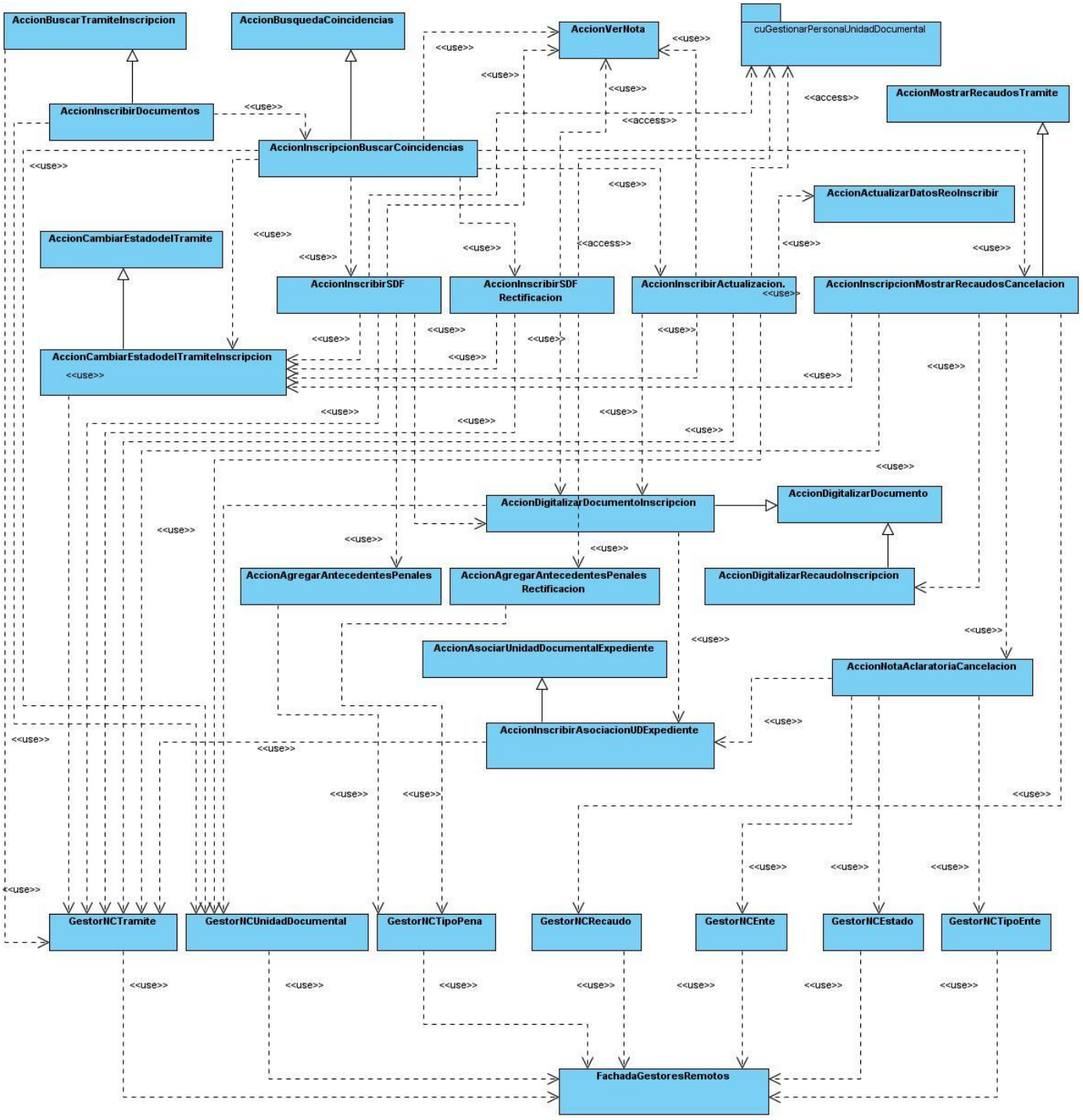

Fig. 10 Diagrama de clases del diseño del caso de uso “Inscribir documentos” (cliente).

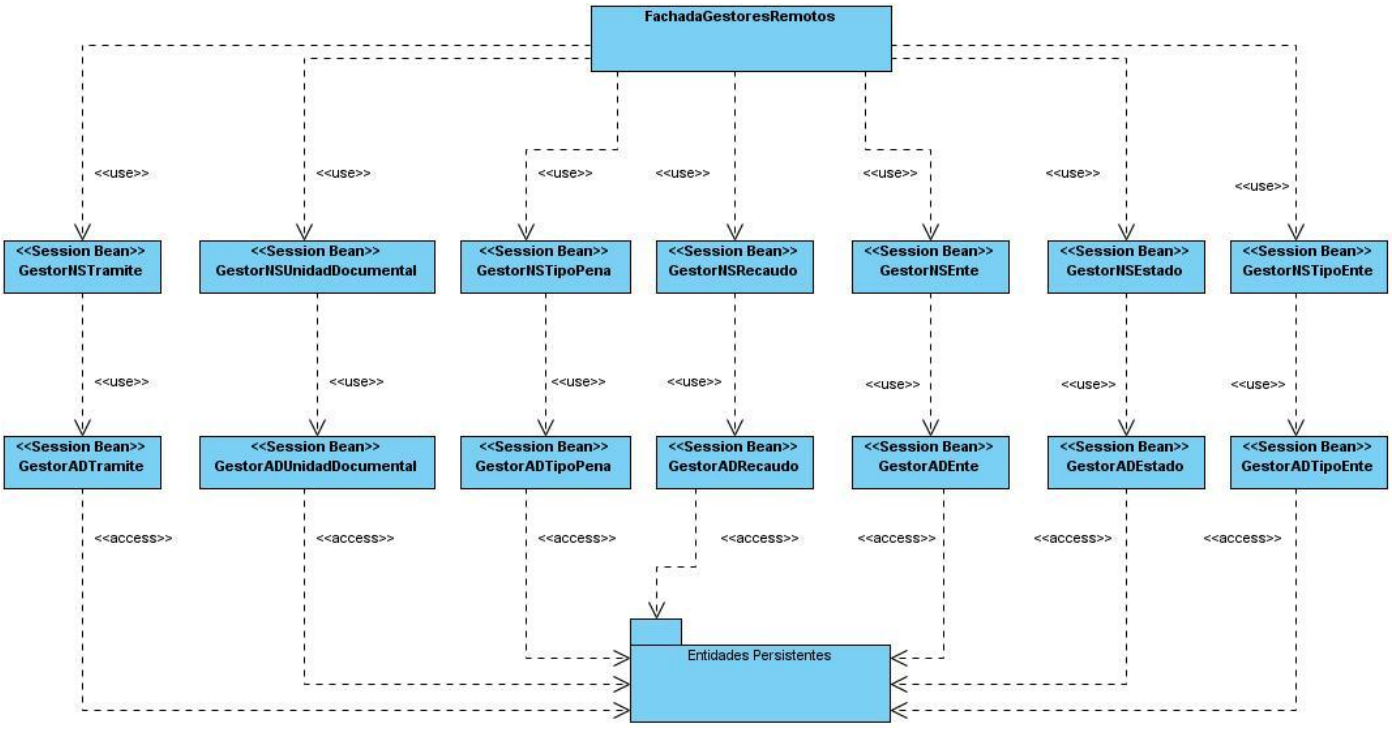

Fig. 11 Diagrama de clases del diseño del caso de uso "Inscribir documentos" (servidor).

A continuación se muestra la explicación de algunas de las clases que conforman los diagramas anteriormente expuestos:

AccionInscribirSDF: es la encargada de controlar las funcionalidades, mediante el panel pnlDatosGeneralesSDF, de los datos generales del tipo de trámite "inscripción de Sentencia Definitivamente Firme (SDF)" y gestionar los datos procesales de las personas involucradas en dicho trámite.

GestorNCTramite: es el gestor donde se realiza la lógica del negocio del cliente para la entidad Trámite. Realiza las llamadas correspondientes a la lógica de negocio del servidor mediante la clase FachadaGestoresRemoto.

FachadaGestoresRemotos: es la clase gestora encargada de la comunicación entre los gestores del cliente con los gestores del servidor, haciendo uso del patrón fachada.

GestorNSTramite: es el gestor donde se realiza la lógica de negocio del servidor. Es la encargada de usar las operaciones que se encuentran en la capa de acceso a datos y atender las peticiones realizadas desde el cliente.

GestorADTramite: es el gestor encargado de implementar la totalidad de las operaciones de persistencia y obtención de datos referente al trámite.

Mediante los diagramas de clases se pudo representar, de forma sencilla, la colaboración y las responsabilidades de las distintas clases que integran la solución. Además permitió entender, aclarar y transmitir información sobre el código. El diseño obtenido cumple con los patrones de Bajo acoplamiento y Alta cohesión permitiendo la colaboración entre los elementos de las clases, sin verse afectados la reutilización de los mismos y el entendimiento de estos cuando se encuentran aislados.

A cada una de las clases le fueron asignadas las tareas que podían realizar según la información que poseen, además de crear las instancias de otras clases en correspondencia con la responsabilidad dada, evidenciando la utilización de los patrones Experto y Creador. También se usó el patrón controlador para asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas.

6.5. Realización de casos de uso del diseño.

Una realización de casos de uso del diseño proporciona una realización física del caso de uso del análisis para el que es trazado. Los diagramas de interacción modelan los aspectos dinámicos de un sistema, debido a que muestran gráficamente cómo los objetos se comunican entre sí para cumplir con los requisitos. Estos diagramas pueden ser expresados en diagramas de colaboración o de secuencia. Estos últimos fueron los realizados en la etapa de diseño, permitiendo mostrar la distribución temporal de los mensajes y un mayor nivel de detalle en el diseño planteado. A continuación se muestra el diagrama de secuencia del caso de uso “Inscribir documentos”:

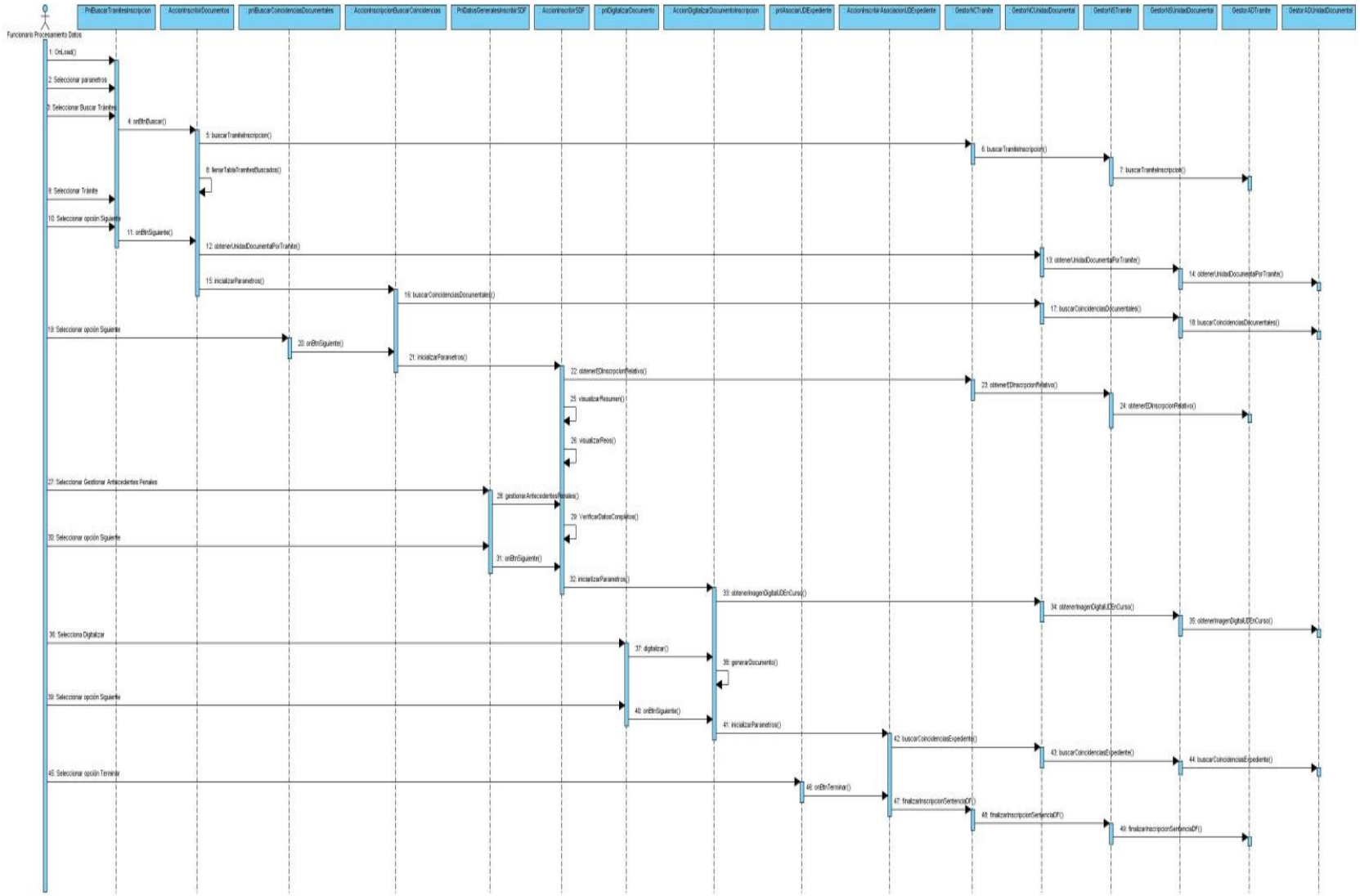

Fig. 12 Diagrama de secuencia del caso de uso “Inscribir documentos”.

En el diagrama de secuencia se puede apreciar el orden de los flujos de mensajes para el caso de uso “Inscribir documentos”. En este diagrama se reflejan las características de los estilos arquitectónicos y patrones del diseño utilizado. El punto de entrada del caso de uso es la clase AccionInscribirDocumento que es la encargada de inicializar el panel PnlBuscarTramiteInscripcion. El proceso continua avanzando hasta realizar la persistencia de los datos ingresados.

6.6. Modelo de datos.

Como parte del flujo de trabajo del diseño se crea el modelo de datos. Las bases de datos constituyen un conjunto de datos relacionados entre sí, una colección estructurada de información. (35). Tiene gran importancia en el ciclo de desarrollo de software, de manera particular para la fase de implementación, pues define formalmente las estructuras permitidas y las restricciones que se aplican con el fin de representar los datos del dominio de la aplicación. La base de datos de RAP se encuentra dividida en dos esquemas:

Framework: creada con el objetivo de facilitar la implementación de los módulos que componen la aplicación SIGESAP. Está compuesto por 14 tablas de persistencia y 4 nomencladores, para un total de 18 tablas.

Público: está conformado por las entidades persistentes necesarias a utilizar por los distintos módulos que forman la solución de software. Se encuentra constituido por 78 tablas de persistencia, 31 nomencladores y 8 entidades de configuración para un total de 117 tablas.

A continuación se muestran ejemplos de tablas del esquema Público, utilizadas en la implementación:

- Entidades persistentes: se almacenan los datos ingresados desde la aplicación.
    - t_tramite: almacena los trámites que se realizan en el sistema.
    - t_unidad_documental: almacena los datos generales de todas las unidades documentales.
    - t_inscripcion: almacena las inscripciones de SDF en el sistema.
    - t_expediente: almacena los expedientes creados por el sistema.
    - t_imagen_unidad_documental: almacena las imágenes de las unidades documentales.

- Nomencladores: se almacena la carga inicial de la aplicación.
    - n_delito: almacena los delitos cometidos, definidos en el sistema.
    - n_tipo_pena: almacena todos los tipos de pena existentes en el sistema.
- Entidades de configuración: se representan las posibles combinaciones.
    - c_estado_tramite: almacena los estados por los que pueden pasar los tipos de trámites.
    - c_flujo_tramite: representa los distintos cambios de estado de cada tipo de trámite, es decir, el flujo completo por el que pasarán los trámites de un tipo determinado.

Las principales tablas del esquema SIGESAP contienen mucha información y sobre ellas se realizan numerosas consultas, por lo que se hizo necesaria la utilización de índices para mejorar el rendimiento de las consultas y ganar tiempo a la hora de realizar las búsquedas. La base de datos estará dividida en dos servidores que garantizarán un balance de carga que permitirá distribuir las consultas de selección entre los mismos.

El modelo de datos general se encuentra en el diccionario de la base de datos en el Expediente del proyecto donde se representa toda la información persistente que se maneja en el proyecto.

6.7. Modelo de despliegue.

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una gran influencia en su diseño. En el modelo cada nodo representa un recurso, los nodos poseen relaciones que representan medios de comunicación entre ellos. (5). A continuación se muestra el diagrama de despliegue donde se representa una panorámica global de los elementos a establecer en la DAP y una descripción de cada uno de ellos.

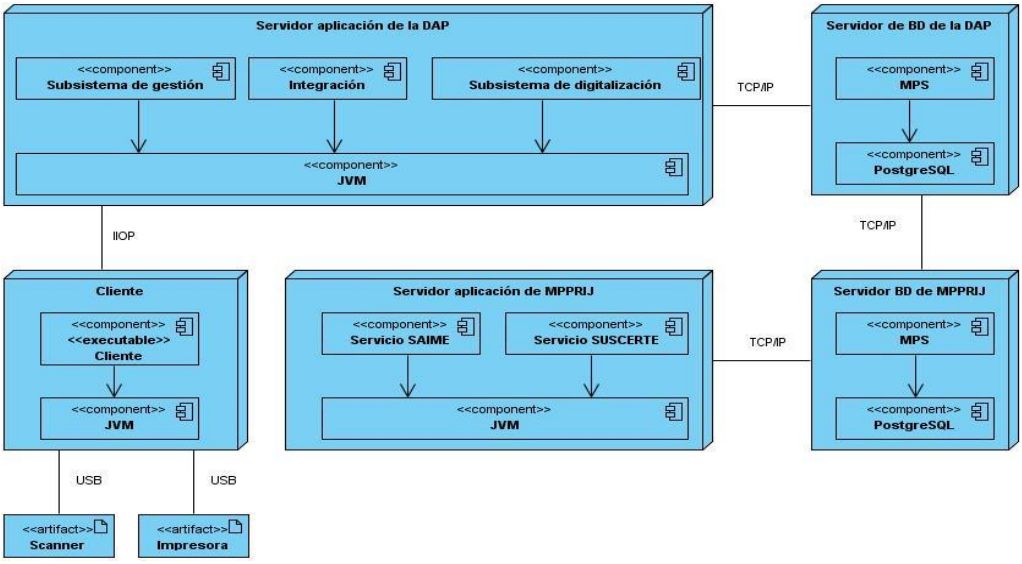

Fig. 13 Diagrama de despliegue.

- Servidor de aplicación de la DAP: nodo que representa el servidor de aplicaciones de la DAP, el cual estará montado sobre el sistema operativo Red Hat Enterprise Linux versión 6.2. En el mismo se encuentran instalados los subsistemas SIGESAP, Integración y DIGIDAP. También se encuentra la JVM en su versión 1.6.

- Servidor de base de datos de la DAP: es el servidor de base de datos que se encuentra en la DAP, montado sobre el sistema operativo Red Hat Enterprise Linux versión 6.2. Se utiliza PostgreSQL 8.4 para su administración y el motor de procesamiento de sentencia (MPS) para la realización de las réplicas hacia el Ministerio del Poder Popular y Relaciones de Interior y Justicia (MPPRIJ).

- PC clientes: nodo correspondiente a las máquinas donde se encuentra instalado el cliente de la aplicación y la JVM en su versión 1.6, montados sobre Ubuntu 11.04, y Windows 7 en español para las estaciones de trabajo que tienen interacción con los dispositivos externos: impresora y escáner.

- Impresora: dispositivo que garantiza la funcionalidad de impresión de documentos y reportes obtenidos como salidas del sistema. Se utilizaran los modelos HP LaserJet P1102 y HP LaserJet M4345 MFP.

- Escáner: dispositivo que se utiliza en la DAP para garantizar la digitalización de los recaudos. El modelo utilizado es Kodak i4000.
- Servidor de base de datos del MPPRIJ: es el servidor de base de datos que se encuentra en dicho ministerio. Utiliza de igual forma PostgreSQL 8.4 para su administración y el motor de procesamiento de sentencia (MPS) para las réplicas.
- Servidor de aplicación del MPPRIJ: nodo donde se encuentra el servidor de aplicaciones del ministerio, el cual cuenta con el Servicio Administrativo de Identificación, Migración y Extranjería (SAIME) y la Superintendencia de Servicios de Certificación Electrónica (SUSCERTE) que es el organismo encargado de validar la firma digital.

7. Implementación del sistema

Al concluir el diseño del sistema se poseen los elementos necesarios para comenzar con la implementación del mismo. Inicialmente se definirán los estándares de codificación empleados para lograr un entendimiento común entre los desarrolladores y posteriormente se realizará el modelo de implementación.

7.1. Estándares de codificación

Los estándares de codificación son un conjunto de reglas que deben cumplir los desarrolladores de un software con el fin de establecer una estructura y un formato que sea común para todo el equipo de trabajo. El cumplimiento de los mismos permite que todos los involucrados en el desarrollo de la solución logren un entendimiento del código generado de forma rápida (36). Los estándares fueron recogidos en el documento del proyecto: “Estándares de codificación”. A continuación se enuncian algunos de ellos:

Ficheros fuente Java:

Cada fichero fuente Java contiene una única clase o interfaz pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, pueden ponerse en el mismo fichero que la clase pública. La clase o interfaz pública debe ser la primera clase o interfaz del fichero.

Los ficheros fuentes Java tienen la siguiente ordenación:

- ✓ Comentarios de comienzo.
- ✓ Sentencias package e import.
- ✓ Declaraciones de clases e interfaces.

Declaraciones de clases e interfaces:

La siguiente lista describe las partes de la declaración de una clase o interfaz, en el orden en que deberían aparecer.

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

Sentencias:

- ✓ Cada línea debe contener como mucho una sentencia.
- ✓ Las sentencias encerradas deben indentarse un nivel más que la sentencia compuesta.

- ✓ La llave de apertura se debe poner al final de la línea que comienza la sentencia compuesta y la llave de cierre debe empezar una nueva línea y ser indentada al mismo nivel que el principio de la sentencia compuesta.
- ✓ Las llaves se usan en todas las sentencias, incluso las simples, cuando forman parte de una estructura de control, como en las sentencias if-else o for. Esto hace más sencillo añadir sentencias sin incluir bugs accidentalmente por olvidar las llaves.
- ✓ Una sentencia if-else debe tener la siguiente forma:

```
if (condicion) {
    //sentencias
} else {
    //sentencias
}
```

- ✓ Una sentencia for debe tener la siguiente forma:

```
for (inicialización; condición; actualización ) {
    //sentencia
}
```

- ✓ Una sentencia try-catch debe tener la siguiente forma:

```
try {
    //sentencias
} catch (Exception e) {
    //sentencias
}
```

Clases

Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases descriptivos y

simples. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo).

Métodos:

Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forman en mayúscula.

7.2. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Este modelo refiere también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modulación disponibles en el entorno de implementación y en el lenguaje de programación empleado y cómo dependen los componentes unos de otros. (5) Uno de los elementos fundamentales de este modelo es el diagrama de componentes, el cual se expone seguidamente.

7.2.1. Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases del diseño. Los componentes tienen relaciones de traza con los elementos de modelo que implementan. Es normal que un componente implemente varios elementos, sin embargo, la forma exacta en que se crea esta traza depende de cómo van a ser estructurados los ficheros de código fuente dado el lenguaje de programación que se esté usando. (5)

Dadas las características de la arquitectura y con el fin de propiciar un mejor entendimiento se realiza una división en tres capas del diagrama de componentes. Esta división lógica estará conformada por una capa de presentación que es la encargada de la interacción con el usuario, una de negocio donde se encuentran tanto la lógica del cliente como la del servidor y una de acceso a datos encargada de la búsqueda de los datos solicitados.

Los principales componentes que se encuentran en la capa de presentación son las acciones, paneles y los casos de uso utilizados del módulo Integración. A continuación se muestra el diagrama de componentes del caso de uso “Inscribir documentos” de la capa antes mencionada:

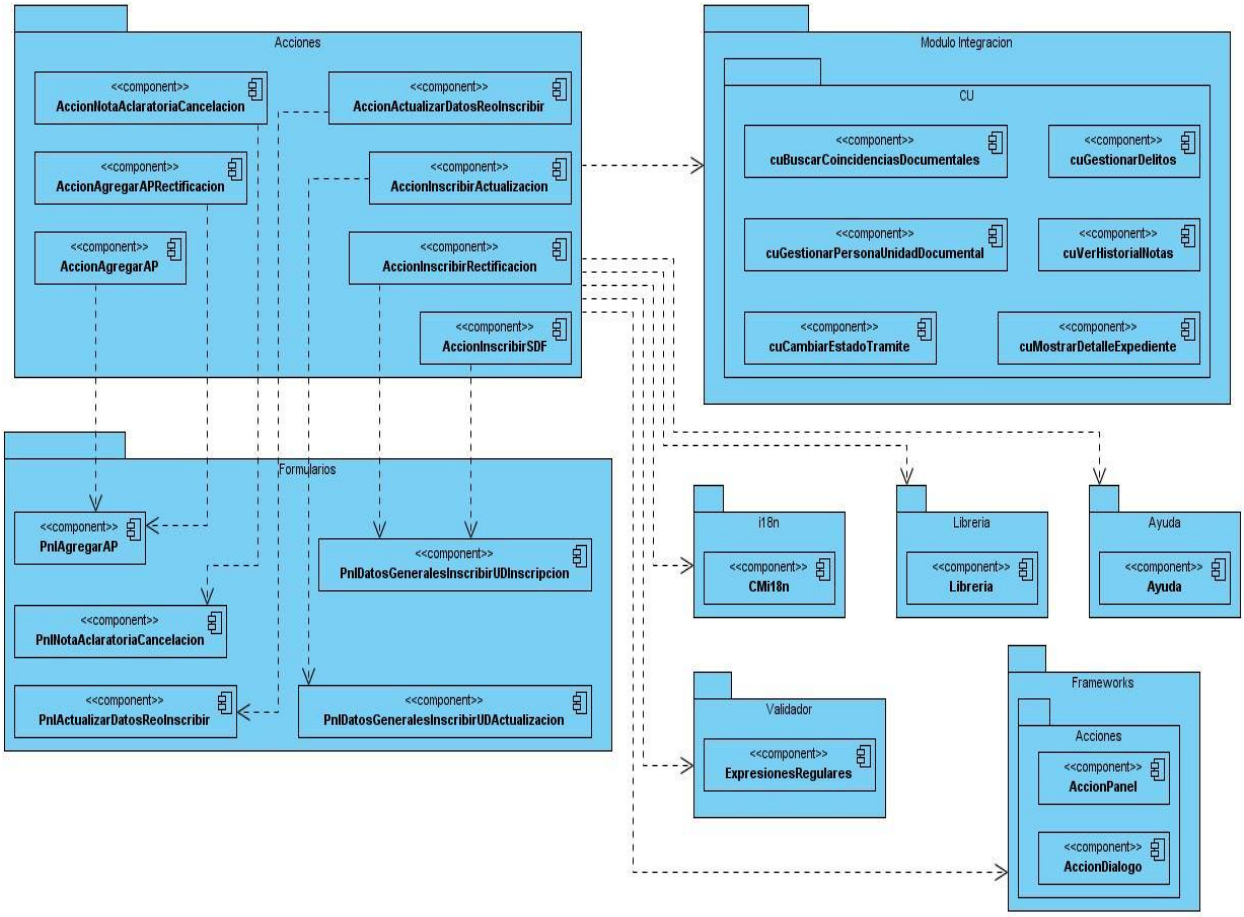

Fig. 14 Diagrama de componentes de la capa de Presentación del caso de uso "Inscribir documentos".

En el negocio se encuentran los diferentes gestores del cliente que atenderán las solicitudes realizadas por el usuario, los cuales harán uso de la interfaz FachadaGestoresRemoto que es la encargada de realizar la comunicación con los gestores del servidor para cumplir las peticiones especificadas. Los gestores del servidor efectuarán de igual forma los pedidos de información a los gestores pertenecientes al acceso a datos. A continuación se muestran los componentes de la capa de negocio del caso de uso "Inscribir documentos":

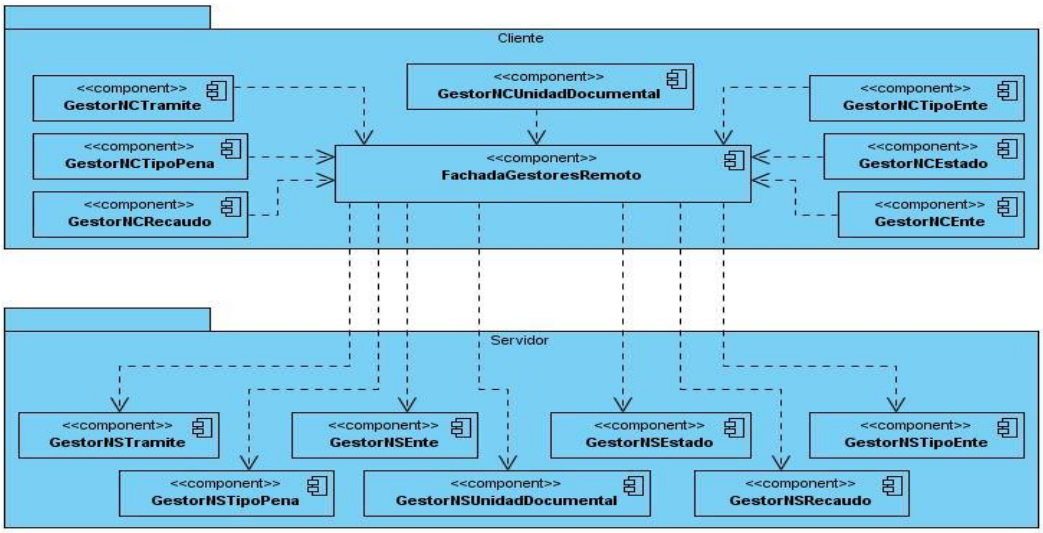

Fig. 15 Diagrama de componentes de la capa de Negocio del caso de uso "Inscribir documentos".

La capa de acceso a datos estará conformada por sus propios gestores, entidades persistentes y los nomencladores. Para poder dar respuesta a las diferentes solicitudes realizadas, los gestores hacen uso del paquete ClasesPersistentes. Una vez obtenidos los datos necesarios, estos se entregan a su capa superior. A continuación se muestran los componentes de la capa de acceso a datos del caso de uso "Inscribir documentos":

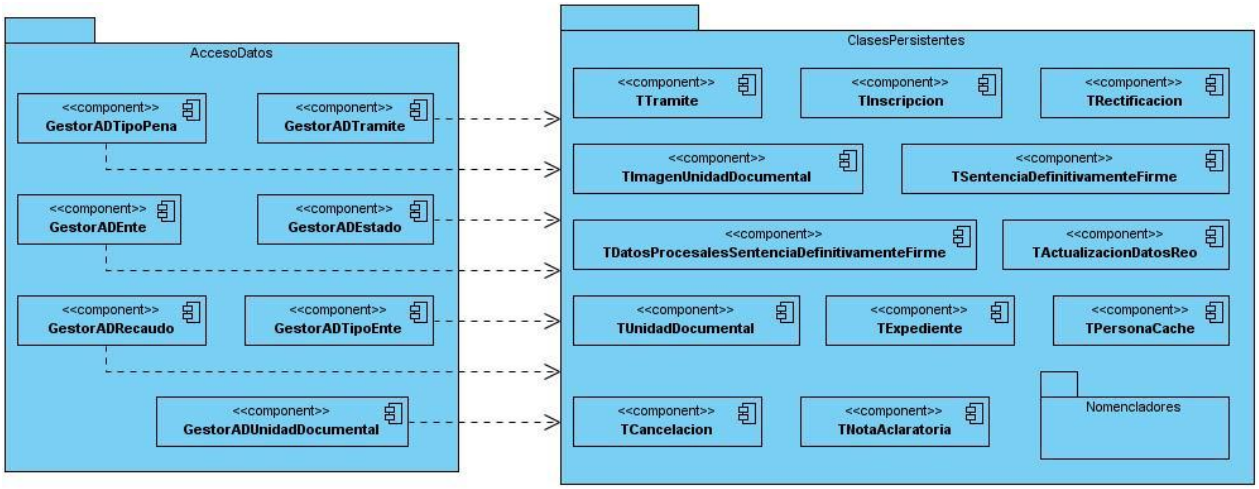

Fig. 16 Diagrama de componentes de la capa de Acceso a datos del caso de uso "Inscribir documentos".

8. Conclusiones del capítulo

A partir de la realización del diseño y la implementación de la solución propuesta para el módulo Inscripción en el presente capítulo, se puede concluir que:

- El análisis de los requisitos funcionales y no funcionales, así como la definición de los casos de uso del sistema, proporcionó una base para desarrollar la solución del sistema propuesto. El estudio de las características del mismo propició que se definiera una arquitectura cliente - servidor, distribuida en n capas y orientada al desarrollo de componentes de software.

- La definición de estándares de diseño y de codificación favoreció un entendimiento común entre los integrantes del equipo de desarrollo. El empleo de los patrones de diseño: fachada, instancia única e iterador, así como la utilización de los GRASP contribuyó a aplicar soluciones ya probadas a diferentes situaciones presentadas.

- La obtención de diferentes artefactos del diseño como el diagrama general de paquetes, los diagramas de clases del diseño, la realización de casos de uso del diseño, el modelo de datos y el modelo de despliegue facilitó la posterior implementación del sistema. La elaboración del modelo de implementación permitió la descripción de cómo los elementos del diseño se implementan en términos de componentes, dejando el sistema listo para someterse a las pruebas necesarias.

CAPÍTULO 3. VALIDACIÓN Y PRUEBAS.

1. Introducción.

Este capítulo muestra la validación de la solución propuesta mediante la realización de pruebas con el objetivo de asegurar el correcto funcionamiento del sistema. Se exponen los resultados obtenidos al aplicar las pruebas de caja negra al módulo Inscripción. Se incluye un resumen sobre las pruebas de integración realizadas al sistema y se exponen las pruebas de aceptación efectuadas.

2. Pruebas de software

El proceso de pruebas es un instrumento que se emplea para determinar el estado de la calidad de un producto de software. En este proceso se realizan diferentes pruebas dirigidas a los componentes independientes del software o al sistema en su totalidad, con el objetivo de medir en qué grado la aplicación cumple con los requisitos iniciales y verificar el resultado de la implementación. (37)

Existen dos métodos de prueba fundamentales:

- Pruebas de caja negra: se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

- Pruebas de caja blanca: se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. (38)

3. Pruebas de caja negra

Se emplea el método de caja negra para probar el sistema propuesto. Las pruebas de caja negra se realizan sobre la interfaz del software; también son conocidas como pruebas de comportamiento debido a

que se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una caja negra cuyo comportamiento solo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas.

Las pruebas de caja negra examinan algunos aspectos del sistema sin hacer énfasis en la estructura interna del software. Se centran principalmente en los requisitos funcionales del mismo. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Además, posibilitan encontrar funciones incorrectas o ausentes, errores de interfaz, en estructuras de datos o en accesos a las bases de datos, errores de rendimiento, de inicialización y de terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre las que se encuentran:

- Técnica de la partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de grafos de causa-efecto: permite validar complejos conjuntos de acciones y condiciones. (39)

Para la validación del software se elaboraron 10 casos de pruebas, aplicando la técnica de partición de equivalencia. A continuación se muestran el caso de prueba para el caso de uso “Inscribir documentos”, específicamente del proceso “Inscribir Sentencia Definitivamente Firme”.

Secciones a probar: la tabla muestra las diferentes secciones en las se divide el caso de uso. Además se detallan los posibles escenarios de cada sección y su correspondiente descripción de la funcionalidad.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Buscar trámites de inscripción.	EC 1.1: Mostrar correctamente los trámites según los filtros seleccionados por el usuario. Ver diseño de caso de prueba “Buscar trámites de inscripción”.	El sistema muestra todos los trámites existentes según los criterios seleccionados.
	EC 1.2: No mostrar los trámites.	No se muestran los trámites de inscripción si no coinciden los criterios con la información existente.
SC 2: Buscar coincidencias documentales.	EC 2.1: Trámite de inscripción de SDF. No se encuentran coincidencias. Ver diseño de caso de prueba “Buscar coincidencias documentales”.	El sistema permite encontrar las coincidencias con unidades documentales existentes, evitando el duplicado de documentos.
	EC 2.1.1: Se encuentran coincidencias.	Aparecen una o varias unidades documentales que coinciden con la que se está inscribiendo. El sistema permite rechazar el trámite en curso evitando así que se duplique información. Ver caso de prueba “Cambiar estado del trámite”.
SC 3: Datos generales.	EC 3.1: Datos generales de unidad documental SDF a inscribir y a rectificar.	El sistema muestra los datos generales que han sido registrados hasta el momento para dicha unidad documental.
	EC 3.2: Agregar personas (privados de libertad) a la unidad documental. Ver diseño de caso de prueba “Gestionar persona de unidad documental”.	El sistema permite incorporar a la unidad documental las personas referidas en el documento en formato duro.
	EC 3.3: Agregar datos procesales para SDF a inscribir y a rectificar correctamente. Ver diseño de caso de prueba “Gestionar delitos”.	El sistema permite agregar los datos procesales presentes en el documento en formato duro a las personas incorporadas a la unidad documental. Permitiendo continuar con el flujo.

	EC 3.3.1: Agregar datos procesales para SDF a inscribir y a rectificar dejando campos vacíos.	El sistema no permite continuar con el flujo.
SC 4: Digitalizar documento	EC 4.1 Documento digitalizado.	El sistema digitaliza la unidad documental, la guarda y permite continuar con el flujo.
	EC 4.2 Documento no digitalizado.	El sistema no digitaliza la unidad documental y advierte la no digitalización, permitiendo continuar con el flujo.
SC 5: Asociar unidad documental a expediente.	EC 5.1 Trámite de inscripción de SDF. No muestra expediente a asociar para los estados iniciado y rectificado. Ver diseño caso de prueba "Asociar unidad documental a expediente".	El sistema permite asociar la unidad documental a un expediente dado o crear un nuevo expediente.

Tabla 1. Secciones a probar.

Descripción de variable: la tabla hace referencia a las descripciones de las variables utilizadas en el caso de uso. De cada una de ella se detalla su nombre, su clasificación, su valor nulo y su descripción.

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Tipo de pena	Campo de selección	No	Representa el campo en el que se selecciona qué tipo de condena tiene el privado de libertad.
2	Años	Campo numérico.	Si	Cantidad de años. Al menos uno de los campos numéricos debe ser completado para poder continuar con el flujo.
3	Meses	Campo numérico.	Si	Cantidad de meses
4	Días	Campo numérico.	Si	Cantidad de días
5	Horas	Campo numérico.	Si	Cantidad de horas

6	Minutos	Campo numérico.	Si	Cantidad de minutos
7	Segundos	Campo numérico.	Si	Cantidad de segundos

Tabla 2. Descripción de variable.

Matriz de datos: tabla donde se encuentran registrados un conjunto de datos de entrada de la aplicación que representan estados válidos y no válidos, para comprobar que la respuesta sea la esperada en ambos casos. A continuación se muestra la matriz de datos del escenario 3.3.

ID del escenario	Escenario	Tipo de pena	Campos de selección numérica.	Respuesta del sistema	Resultado de la prueba
EC 3.3	Agregar datos procesales para la SDF a inscribir.	V Penal	V Años hasta 30, meses hasta 12, días hasta 31, horas hasta 24, minutos hasta 60, segundos hasta 60.	El sistema permite la inserción de los datos procesales regresando a la pantalla de datos generales.	Satisfactorio.
EC 3.3.1	Agregar datos procesales para SDF a inscribir dejando campos vacíos.	N/A	N/A	El sistema muestra un mensaje de aviso informando que deben ser llenados todos los campos.	Satisfactorio
		Penal	Todos los campos de selección numérica vacíos.	El sistema muestra un mensaje de aviso informando que deben ser llenados todos los campos.	Satisfactorio

Tabla 3. Matriz de datos.

Las secciones, descripciones de variables y matrices de datos de los restantes casos de uso se encuentran en el Expediente del proyecto.

Las pruebas de caja negra fueron realizadas por un equipo de especialistas de Calisoft UCI, el cual llevó a cabo tres iteraciones y un proceso de regresión. Se detectaron 40 no conformidades en todo el proceso de

pruebas, el cual se caracterizó por la disminución paulatina de las no conformidades hasta llegar a cero. En la primera iteración realizada se obtuvieron 17 no conformidades. Posteriormente se le dio solución a las mismas y en la segunda iteración se detectaron 14 nuevas no conformidades, representando una disminución del 17,7 %. En la tercera iteración realizada se obtuvieron 9 no conformidades, lo cual representó una disminución del 42,1 % con respecto a las detectadas inicialmente. En el proceso de regresión no se obtuvo ninguna no conformidad y se comprobó la solución del 100% de las no conformidades detectadas en iteraciones anteriores. Quedando el módulo Inscripción aprobado y liberado por Calisoft.

A continuación se muestra una gráfica que representa el número de no conformidades detectadas en cada una de las iteraciones:

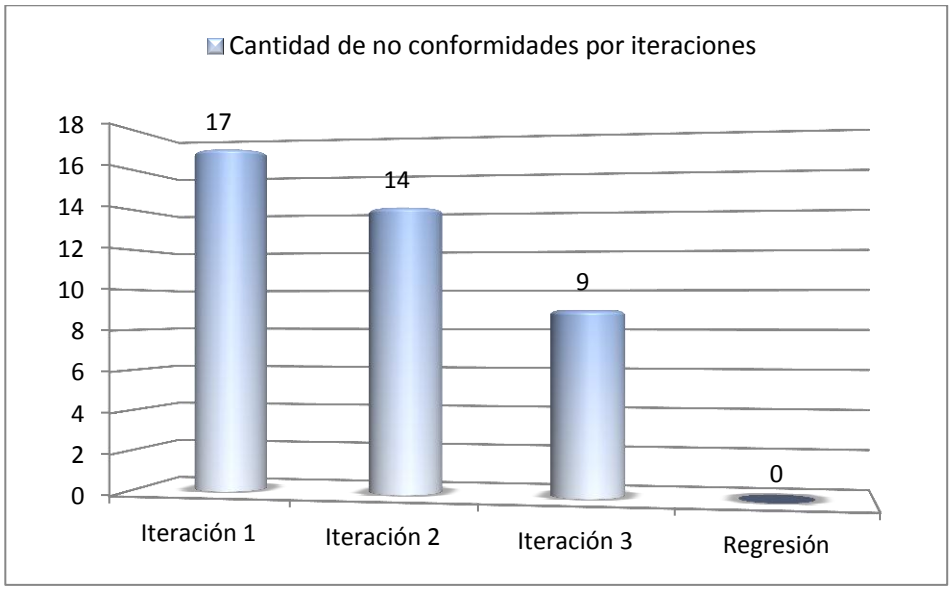

Fig. 17 Gráfica de las pruebas de caja negra.

4. Niveles de pruebas

Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas:

- Prueba de desarrollador: es la prueba diseñada e implementada por el equipo de desarrollo.

- Prueba independiente: es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores.
- Prueba de unidad: es la prueba enfocada a los elementos probables más pequeños del software.
- Prueba de integración: es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso.
- Prueba de sistema: son las pruebas que se hacen cuando el software está funcionando como un todo.
- Prueba de aceptación: es la prueba final antes del despliegue del sistema.

Con la utilización del método de prueba caja negra se puede validar la solución en cuanto a lo especificado en los casos de uso, pero no se demuestra completamente la ausencia de errores, ni se garantiza el correcto funcionamiento de la solución con el sistema. Por estas razones se deciden realizar diferentes niveles de pruebas a la solución.

4.1. Pruebas de integración

El proceso de la integración del sistema implica construir éste a partir de sus componentes y probar el sistema resultante para encontrar problemas que pueden surgir debido a la integración de los componentes. Las pruebas de integración comprueban que estos componentes realmente funcionan juntos, son llamados correctamente y transfieren los datos correctos en el tiempo preciso a través de sus interfaces.

Estas pruebas se llevaron a cabo durante la construcción del sistema y terminaron probando el sistema como un todo. Dichas pruebas son similares a las pruebas de caja negra, destinadas a encontrar fallos o errores en el módulo de inscripción cuando sus operaciones dependen de los servicios prestados por el módulo de integración, debido que un software suele fallar cuando trabaja de forma conjunta.

Existen dos formas de integración:

- Integración descendiente: Se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto.

- Integración ascendente: El programa se construye y se prueba en pequeños segmentos. (40)

Durante la realización de estas pruebas se detectaron problemas con el componente de digitalización, ficha digital, grid y fecha. Estas dificultades presentadas fueron resueltas durante el proceso de regresión de las pruebas de caja negra.

4.2. Pruebas de aceptación

Las pruebas de aceptación comparan el producto final con las necesidades de los clientes. Generalmente se llevan a cabo por estos, los cuales analizan si el programa satisface los requisitos aprobados al comienzo del desarrollo. En el caso de que los cumpla, el software es aceptado. (41)

Para la realización de las pruebas antes mencionadas se redactó el Plan de pruebas de aceptación, el cual contiene las estrategias y los recursos necesarios para ejecutar una metodología de pruebas de aceptación. Inicialmente se identificaron los involucrados y los responsables de efectuar las actividades propuestas en el plan de pruebas, integrados por representantes de ALBET, Calisoft y la DAP. Estas pruebas se realizaron en una única iteración, con el objetivo de analizar las funcionalidades que dieron respuesta a los requisitos funcionales acordados inicialmente con el cliente. Fueron aplicados dos tipos de pruebas: de funcionalidad (para comprobar que las funcionalidades presentes en la aplicación responden a los requisitos acordados inicialmente con el cliente) y de seguridad (para verificar que los datos y la aplicación solo son accedidos por los actores definidos según los niveles de acceso).

En la realización de ambos tipos de pruebas participaron 26 personas (dos expertos técnicos, un especialista de Calisoft, 12 expertos funcionales y 11 integrantes del equipo de ALBET). Además se emplearon como recursos técnicos: cuatro estaciones de trabajo donde se ejecutó un cliente del sistema y tres servidores (dos servidores de aplicaciones y un servidor de base de datos). Las pruebas de aceptación se realizaron de forma manual, dividiendo el trabajo en cuatro fases:

- Primera fase: se organizó el escenario de las pruebas y se capacitó al equipo que participó en las mismas.
- Segunda fase: se realizaron las pruebas de funcionalidad mediante las validaciones de la lógica del flujo básico del sistema.
- Tercera fase: se realizaron pruebas de seguridad.

- Cuarta fase: se aprobaron y firmaron de los documentos entregables de las pruebas.

Durante la realización de las pruebas fueron generados los siguientes documentos los cuales fueron firmados y aprobados por los principales involucrados (Calisoft, ALBET y el personal de la DAP):

- Plan de pruebas de aceptación.
- Informe final de no conformidades.
- Informe final de peticiones de cambio.
- Acta de aceptación.
- Informe resumen de las pruebas de aceptación.
- Informes diario de no conformidades.
- Informes diarios de peticiones de cambios.

Durante la realización de las pruebas de integración los involucrados interactuaron con el sistema con el objetivo de detectar fallos en el mismo. Mediante la aplicación de estas evaluaciones se logró comprobar una correspondencia entre los requisitos funcionales documentados y los implementados, se resolvieron todas las no conformidades y las peticiones de cambios fueron gestionadas. A continuación se muestran las no conformidades que fueron detectadas:

No.	Descripción de la no conformidad	Ubicación en el artefacto	Nivel de importancia de la no conformidad	Respuesta del equipo de desarrollo
1	Resaltar la funcionalidad revisar trámite con el formato de un link.	Trámites por otorgar	Media	Resuelta
2	Estandarizar los vínculos e íconos en todo el sistema.	Todo el sistema.	Media	Resuelta
3	Estandarizar los márgenes en los resúmenes de las fichas.	Revisión de otorgamiento.	Media	Resuelta

4	Estandarizar los links en negrita y subrayado en la ayuda.	Otorgar trámites de inscripción.	Media	Resuelta
5	No se puede buscar el sometimiento a cancelar.	Revisión legal.	Alta	Resuelta
6	Se debe dar la posibilidad de moverse por los componentes tanto por el teclado como por el mouse.	Procesamiento de datos.	Media	Resuelta
7	El sistema colapsa al intentar mostrar la imagen digital del documento a rectificar.	Procesamientos de datos.	Alta	Resuelta

Tabla 4. Registro de las no conformidades detectadas.

A continuación se muestran las peticiones de cambios realizadas por los clientes:

No.	Descripción de la petición de cambios.	Ubicación del artefacto	Necesidad o mejora.	Procede/Solución	Estado
1	Cambiar la palabra reo por privado de libertad.	Toda la aplicación.	Necesidad	Procede. Modificación a requisito.	Resuelto y aprobado.
2	Centrar los números (cantidad de folios).	Buscar trámites.	Mejora	Procede. Modificación a requisito.	Resuelto y aprobado.
3	Dar la posibilidad de insertar el mismo dato procesal a todos los privados de libertad.	Procesamiento de datos	Mejora	Procede. Nuevo requisito.	Resuelto y aprobado.

Tabla 5. Registro de peticiones de cambios.

Una vez concluidas las pruebas de aceptación se identificaron siete no conformidades las cuales fueron resueltas y aprobadas en su totalidad. Además se realizaron tres solicitudes de cambios, de las cuales dos se consideraron mejoras al sistema y una constituyó una necesidad para el correcto funcionamiento de la aplicación. Todas las peticiones de cambios se resolvieron y fueron aprobadas.

Como constancia de que la implementación del producto se realizó según las especificaciones del cliente se elaboró el Acta de Aceptación la cual fue firmada por los principales representantes de las partes involucradas. De esta forma quedó aceptado el software por los clientes finales.

Las actas de aceptación del cliente se encuentran en el Expediente del proyecto.

5. Conclusiones del capítulo

A partir de la elaboración del presente capítulo, se puede concluir que:

- La realización de pruebas de caja negra al módulo Inscripción permitió la obtención y resolución de las no conformidades detectadas.
- Mediante la aplicación de las pruebas de integración se comprobó que los componentes que conforman la solución funcionan correctamente en conjunto.
- La firma por parte de los clientes de las actas de aceptación del producto permitió comprobar que el software elaborado cumple con los requisitos aprobados por los mismos y satisface sus necesidades.

CONCLUSIONES

Al finalizar el presente trabajo de diploma se arriban a las siguientes conclusiones:

- El análisis de las principales características de la metodología, las herramientas y las tecnologías a emplear contribuyó a desarrollar el sistema propuesto.
- Se obtuvieron los artefactos necesarios para el diseño de acuerdo a la metodología de desarrollo seleccionada en el proyecto.
- Los artefactos elaborados durante el diseño sirvieron como base para la implementación del módulo.
- Mediante la aplicación de las pruebas de caja negra, de integración y de aceptación se comprobó que los componentes que conforman la solución funcionan correctamente, logrando satisfacer los requisitos acordados con los clientes.

De esta forma se da cumplimiento al objetivo general del trabajo de diploma: Diseño e implementación del módulo Inscripción del Sistema para la Gestión de Antecedentes Penales (SIGESAP).

BIBLIOGRAFÍA

1. **Cleger Despaigne, Eliober , y otros, y otros.** *Transformación de los procesos registrales en los registros de antecedentes penales.* Ciudad de La Habana : s.n., 2011.

2. **Walter Galloso, Mariños.** WG abogado. [En línea] 15 de febrero de 2011. [Citado el: 10 de diciembre de 2011.] http://derechoregistral.informaccion.com/.

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

4. **Mendez Virrueta, Alejandra.** Metodologías de desarrollo de software. *Investigación documental.* Apatzingán, México : s.n., 2010.

5. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley, 2000.

6. **Martínez, Alejandro y Martínez, Raúl .** *Guía a Rational Unified Process.*

7. Ecured. [En línea] [Citado el: 15 de diciembre de 2011.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo.

8. Cientec. [En línea] 19 de enero de 2012. http://www.cientec.com/analisis/ana-uml.html.

9. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley.

10. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* 2001. 8448132149.

11. Definición.de. [En línea] [Citado el: 22 de Febrero de 2012.] http://definicion.de/lenguaje-de-programacion/.

12. Lenguajes de programación. [En línea] [Citado el: 22 de Febrero de 2012.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

13. **Flanaga, David.** *Java en pocas palabras.* México : McGraw-Hill, 1998. 970-10-2070-7.

14. **Ing. Prieto, Julio César y Ing. Fadraga Artiles, Lissuan.** *Documento de la Arquitectura.* La Habana : s.n., 2011.

15. NetBeans. [En línea] [Citado el: 20 de diciembre de 2011.] http://netbeans.org/index_es.html.

16. **Lockhart, Thomas .** *Tutorial de PostgreSQL.*

17. **Gibert Ginestà, Marc y Pérez Mora, Oscar.** *Bases de datos en PostgreSQL.* P06/M2109/02152.

18. **Martinez, Rafael.** Sobre PostgreSQL | www.postgresql.org.es. [En línea] 2 de Octubre de 2010. [Citado el: 24 de Noviembre de 2011.] http://www.postgresql.org.es/sobre_postgresql.

19. Microsoft | TechNet. [En línea] 10 de enero de 2012. http://technet.microsoft.com/es-es/library/cc728199%28WS.10%29.aspx.

20. GlassFish. [En línea] 18 de abril de 2011. [Citado el: 6 de diciembre de 2011.] http://glassfish.java.net/es/public/users.html.

21. **Romina Ledesma, Claudia y Alí, Claudia Alejandra.** *Trabajo de investigación: J2EE.* Argentina : s.n.

22. **Rondón Grados, Luis.** Java J2EE. [En línea] 3 de Abril de 2009. [Citado el: 22 de Febrero de 2012.] http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html.

23. **Young, Ralph R.** *The requirements engineering handbook.* Boston : Artech House, 2004. 1-58053-266-7.

24. **Quintero, Juan Bernardo.** Ingenian. [En línea] [Citado el: 14 de Marzo de 2012.] http://trac.ingenian.com/proyectos/notificaciones/raw-attachment/wiki/RecursosRelacionadosNotificaciones/Requisitos.pdf.

25. **Abran, Alain , y otros, y otros.** *Software Engineering Body of Knowledge.* Estados Unidos : IEEE Computer Society, 2004. 0-7695-2330-7.

26. **Ruz Pérez, Yelaine y Medina Delgado, Ernesto .** *Especificación de requisitos de software v1.1. Módulo de Inscripción.* 2010.

27. **Ruz Pérez, Yelaine y Medina Delgado, Ernesto.** *Especificación de Requisitos de Software v1.0. Módulos Administración y Configuración Local.* 2010.

28. **de la Torre Llorente, César, y otros, y otros.** *Guía de arquitectura en N-Capas orientadas al Dominio con .NET 4.0.* España : Krasis Press, 2010. 978-84-936696-3-8.

29. **Szyperski, C.** *Component Software. Beyond Object-Oriented Programming.* s.l. : Addison-Wesley, 1998.

30. **Medina Delgado, Ernesto y Ruz Pérez, Yelaine.** *Pautas para la realización del Diseño.* La Habana : s.n., 2010.

31. **Gamma, Erich , y otros, y otros.** *Design Patterns. Elements of Reusable Object-Oriented Software.*

32. **Martínez Juan, Francisco Javier.** Entorno Virtual de Aprendizaje. [En línea] [Citado el: 7 de Marzo de 2012.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Diseno_de_software/Guia_de_Construccion_de_Software_en_Java_con_patrones_de_diseno.pdf.

33. **Larman, Craig.** *UML y patrones.* s.l. : Pearson.

34. Departamento de Informática Universidad Técnica Federico Santa María. [En línea] [Citado el: 2012 de Marzo de 16.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

35. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 2 de Marzo de 2012.] http://eva.uci.cu/file.php/160/Curso_2010-2011/Semana_6/Conferencia_5/Materiales_Complementarios/Teoria_y_Diseno_de_Base_de_Datos.ppt.

36. **Proyecto RAP.** *Estándares de codificación.* La Habana : s.n.

37. Pruebas de Software. [En línea] [Citado el: 7 de Mayo de 2012.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

38. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 26 de Abril de 2012.] http://eva.uci.cu/file.php/160/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Documentacion_sobre_Pruebas.pdf.

39. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 27 de Abril de 2012.] http://eva.uci.cu/file.php/160/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf.

40. **Sommerville, Ian.** *Ingeniería del software.* Madrid : Addison Wesley, 2005. ISBN 84-7829-074-5.

41. **Myers, Glenford J.** *The Art of Software Testing.* Segunda. New Jersey : John Wiley & Sons, 2004. ISBN 0-471-46912-2.