Diseño e Implementación del módulo Archivo de la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales.

FACULTAD 3

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

Autor: Leyandry García González

Tutora: Ing. Yanelis Vega García

Co - Tutor: Ing. Dusniel Horta Centeno

Ciudad de La Habana

2012

Declaración de Autoría

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

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

_____ _____

Leyandry García González Ing. Yanelis Vega García

Dedicatoria

Quiero dedicar el esfuerzo que hizo posible la realización de este trabajo de diploma a mis padres que son mi razón de ser, siempre han confiado en mí y me han apoyado en todo momento, que son y serán siempre mi ejemplo a seguir.

Agradecimientos

Quiero agradecer a mi mamá (Nancy) por ser la persona que siempre me ha impulsado a cumplir mis metas, que me ha dado ánimos para seguir adelante y por todo el amor y cariño que nunca me ha faltado.

A mi papá (José Enrique) por su sacrificio y preocupación, por los consejos y el cariño que siempre me ha dado.

A mi hermano (Leandry) que lo quiero con la vida.

A mi familia en general que siempre ha estado presente en mi vida y en especial en estos 5 años.

A mis abuelos (Aidee y Juan) por su apoyo incondicional, que han sido otros padres para mí.

A todos mis ti@s y prim@s que ocupan una parte importante de mi vida.

A mi tutora (Yanelis) por su ayuda y por ser parte de este trabajo.

A todas mis amistades de primer año Alejandro, Luis Ángel, Jorge, Dayana, Yanetsi, Yosbel, Yanet, El chiki y Arlettys, que hemos logrado llegar hasta el final.

A mis compañeros de aula por los momentos compartidos durante la carrera, a Yudier, Sojo, el Yhony, Bichot, Edgar, Amílcar, Raúl, Nilier, Yudith, Lenis, Dayami, Yiliam, el Cuadra, en fin a todos los que de una forma u otra contribuyeron a la realización de este trabajo.

Resumen

Producto del convenio de cooperación establecido entre Cuba y Venezuela, en la novena mixta es firmado el contrato para llevar a cabo la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales (DAP) de la República Bolivariana de Venezuela. La DAP tiene como objetivo salvaguardar los antecedentes penales de los ciudadanos venezolanos. La institución comprende varias áreas entre las que se encuentra Archivo que constituye el destino final de dichos antecedentes y cuya función principal es la de controlar la entrada y salida de los documentos físicos, tanto para la consulta por parte de los funcionarios del área de Revisión Legal como para el Centro de Digitalización. La División cuenta con la ayuda del Sistema de Registro y Control de Antecedentes Penales que sólo almacena los Expedientes de los últimos 10 años de trabajo, impidiendo la gestión completa del Fondo Documental que se remonta a más de 100 años. La solución que se propone con el módulo Archivo contribuirá a mejorar la gestión y organización del Fondo Documental dotándole de funcionalidades pensadas para esta área. Durante el desarrollo de la investigación y del sistema propiamente, se muestran los artefactos generados durante esta etapa a partir de los requisitos definidos por los analistas y utilizando las herramientas y tecnologías previamente definidas por el equipo de desarrollo. Además de validar los resultados de la investigación a través de la aplicación de distintos tipos de pruebas.

Índice

Introducción

Uno de los proyectos llevados a cabo por la Universidad de las Ciencias Informáticas es la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales de Venezuela, firmado en la novena mixta del Convenio Integral de Cooperación entre Cuba y la República Bolivariana de Venezuela establecido hace más de 10 años en el marco de la Alianza Bolivariana para las Américas (ALBA). La División de Antecedentes Penales (DAP) perteneciente al Ministerio del Poder Popular para Relaciones Interiores y Justicia, despacho del Viceministro de Política Interior y Seguridad Jurídica adscrita a la Dirección General de Justicia, Instituciones Religiosas y Cultos es la institución que se encarga de registrar como lo establece la constitución de la República Bolivariana de Venezuela y específicamente la Ley de Registro de Antecedentes Penales, los antecedentes penales de los ciudadanos venezolanos, emitir certificaciones de antecedentes penales a los distintos entes tanto nacionales como internacionales autorizados por la legislación venezolana vigente. La DAP es única y centralizada, esto trae consigo que los antecedentes penales enviados desde los Tribunales en cada uno de los estados venezolanos tengan como único destino el Archivo de la División. De esta forma la oficina cuenta con un extenso Fondo Documental que atesora Expedientes desde hace más de 100 años, salvaguardando así el patrimonio documental en cuanto a Antecedentes Penales se refiere.

En la DAP se llevan a cabo dos procesos fundamentales: Inscripción de unidades documentales que constituyen antecedentes penales y la Certificación de antecedentes penales. Cada uno de estos procesos se nutre del trabajo y el funcionamiento de las diferentes áreas que comprende la institución: Presentación, Procesamiento y asociación de metadatos, Revisión Legal, Otorgamiento y Archivo que es donde finalmente reposan en Expedientes los documentos producto del proceso de Inscripción. Los documentos más importantes que se reciben son Sentencias Definitivamente Firmes, Sometimientos a Juicio, Libertad Condicional de la Pena, Hoja de Reo, Nota Aclaratoria y Documento Complementario. Estos documentos llegan a la División como unidades documentales y cuando se le asigna un número mediante el proceso de Inscripción pasan a ser Expedientes, y se van agrupando según la causa a la que pertenecen, de esta forma aumenta la dificultad a la hora de identificar todos los Expedientes en los que está involucrado un penado.

Por otra parte, existe una vinculación parcial entre lo real existente en el Archivo y lo registrado en el Sistema de Registro y Control de Antecedentes Penales (SIRCAP), sistema informático usado hasta el momento que sólo tiene inscritos los antecedentes que corresponden a los últimos 10 años de trabajo, lo que dificulta la localización de muchos documentos y causa deficiencias en la gestión

de préstamos de los Expedientes. SIRCAP tampoco cuenta con funcionalidades pensadas para el área de Archivo lo que trae consigo que los funcionarios no puedan verificar ni controlar si el documento que llega listo para ser archivado es realmente el que fue inscrito.

El deterioro físico de los documentos aumenta con el tiempo que llevan archivados debido a las condiciones del local y la constante manipulación de los mismos. Algunos ejemplos que evidencian el deterioro son: papel rasgado, tinta corrida y escritura borrosa. Todo esto atenta contra la correcta conservación del Fondo Documental.

Teniendo en cuenta esta problemática se identificó el siguiente problema a resolver:

¿Cómo satisfacer los requisitos acordados con el cliente de manera tal que contribuya a mejorar el proceso de gestión de Archivo en la División de Antecedentes Penales de la República Bolivariana de Venezuela?

Para dar respuesta al problema planteado se definió como **objetivo general:** Desarrollar el módulo Archivo de la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales, enfocado en el Proceso de Desarrollo de Software como **objeto de estudio** y cuyo **campo de acción** estará enmarcado en el Desarrollo de Software de Gestión.

Como guía de la investigación se plantea siguiente idea a defender:

Con el desarrollo del módulo Archivo se contribuirá a mejorar el proceso de gestión del Fondo Documental de la División de Antecedentes Penales.

Para dar cumplimiento al objetivo general, se ha desglosado en los siguientes objetivos específicos:

1. Elaborar el marco teórico de la investigación.
2. Obtener el Modelo de Diseño.
3. Realizar la implementación del módulo Archivo.
4. Aplicar pruebas para validar los resultados obtenidos.

Tareas de investigación a realizar.

- ✓ Elaboración del marco teórico de la investigación.
- ✓ Estudio de sistemas informáticos similares.
- ✓ Estudio de la plataforma de desarrollo y de las herramientas utilizadas para diseñar,

implementar y probar el sistema.

- ✓ Estudio de los paradigmas de programación.
- ✓ Estudio y selección de los patrones de diseño más factibles para la solución propuesta.
- ✓ Análisis de los artefactos Especificación de requisitos de software y Modelo del Sistema que sirven de base para el diseño y la implementación que se pretende realizar.
- ✓ Realización de los Diagramas de Clases del Diseño.
- ✓ Realización de los Diagramas de Componentes.
- ✓ Realización del Diagrama de Despliegue.
- ✓ Realización del Modelo de Datos.
- ✓ Implementación de los componentes.
- ✓ Diseño de Casos de prueba.

Métodos de Investigación

Se considera método científico de investigación a una serie de pasos sistemáticos e instrumentos que nos llevan a un conocimiento científico. Estos pasos nos permiten llevar a cabo una investigación.[1]

A continuación se muestran los métodos a utilizar en el desarrollo de la investigación.

Métodos teóricos

Histórico - Lógico: Permitió realizar un estudio de los sistemas informáticos de gestión documental similares a la solución propuesta implementados.

Hipotético - Deductivo: Posibilitó la determinación de las herramientas factibles a utilizar en el desarrollo de la aplicación de acuerdo a las necesidades del cliente.

Modelación: Ayudó a la creación del modelo de diseño de la solución propuesta, para un mejor entendimiento a la hora de desarrollar el sistema.

Estructura del trabajo de diploma

El presente trabajo de diploma cuenta con la siguiente estructura de capítulos.

Capítulo 1: Fundamentación teórica

Se abordarán los elementos teóricos que sustentan el problema a resolver y los objetivos del trabajo, además de un análisis de las diferentes herramientas, metodologías y tecnologías que serán usadas para el desarrollo.

Capítulo 2: Propuesta de solución

Se realiza el diseño e implementación de la solución a partir de la especificación de los requisitos de software y los diagramas de casos de uso del sistema. Se exponen los patrones de diseño a utilizar y se realizará una breve descripción de la arquitectura para lograr el entendimiento del diseño. Se muestran como resultados los diagramas de componentes y de clases del diseño con la descripción de cada una por casos de uso, además de las pautas de diseño y estándares de codificación establecidos por el equipo de desarrollo.

Capítulo 3: Validación de los resultados

En este capítulo se evalúa la calidad y el correcto funcionamiento del módulo Archivo a través de distintos tipos de pruebas. Se realizará un análisis estadístico de las no conformidades detectadas durante las distintas etapas de pruebas. Se muestran los casos de pruebas diseñados para guiar el proceso de pruebas a través las técnicas seleccionadas en el capítulo 1. Finalmente se mostrará el Acta de Aceptación firmada por el cliente, evidenciando que quedaron satisfechos todos los requisitos establecidos por este así como la conformidad con el sistema implementado.

Al concluir el trabajo de diploma se espera como resultados:

El diagrama de clases para cada caso de uso implementado.

El diagrama de paquetes.

El diagrama de componentes.

El diagrama de despliegue.

El modelo de datos.

Obtener una versión funcional y probada del módulo Archivo que contribuya a mejorar la gestión de Fondo Documental de la División de Antecedentes Penales de Venezuela.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se abordarán temas relacionados con los principales elementos teóricos a tener en cuenta para dar solución al problema anteriormente descrito. Se tratarán algunos conceptos básicos referentes a la gestión documental necesarios para entender el funcionamiento del Archivo. Además, se realizará una descripción de la metodología, herramientas y tecnologías previamente escogidas por el equipo de desarrollo a utilizar durante el diseño, la implementación y la etapa de pruebas de la solución propuesta.

1.1. Conceptos básicos

Documento: Un documento es una carta, diploma o escrito que ilustra acerca de un hecho, situación o circunstancia. También se trata del escrito que presenta datos susceptibles de ser utilizados para comprobar algo.[2]

Archivo: Institución o una parte estructural de ella que realiza la recepción, organización y conservación de documentos para su utilización; conjunto orgánico de documentos producidos y/o acumulados por una persona natural o jurídica.[3]

1.2. Sistemas de gestión de Archivo similares

Dentro de los sistemas existentes en el mundo para la gestión de Archivo en la actualidad podemos encontrar:

OdiloA3W (España)

Es una aplicación Web diseñada y programada siguiendo el modelo líder de desarrollo Java 2 Enterprise Edition (J2EE), empleando para ello el lenguaje de programación Java y las tecnologías Enterprise Java Bean (EJB), Java Server Page (JSP), Servlets, XML, Apache Structs, Jasper Report y otros que la convierten en una aplicación multiplataforma, apropiada tanto para sistemas libres como propietarios.

Dado el revolucionario diseño modular de la aplicación, la estandarización de los componentes del sistema y la alta configuración del modelo de datos, OdiloA3W se adapta sorprendentemente a cada cliente, posee multilingüismo (permite pares de idiomas euskera/castellano, por ejemplo),

importación de datos desde cualquier sistema productor de documentación, transmisión de contenidos mediante Protocolo para la transmisión de contenido en internet, en inglés Open Archives Initiative-Protocol Metadata Harvesting (OAI-PMH), configuración de Web Services para comunicación fluida con aplicaciones, disponibilidad SaaS mediante Cloud Computing.[4]

Fig.1: Sistema de gestión de Archivo OdiloA3W.

Los principales clientes de este software son Universidades españolas entre las que se encuentran, Málaga, Granada y la Universidad Pública de Navarra entre otros entes gubernamentales.

OdiloA3W se basa fundamentalmente en la digitalización de documentos y requiere total conexión con internet. Este sistema no se adecua a las necesidades de la División teniendo en cuenta que la propuesta de solución debe ser de escritorio y su única comunicación será con el Ministerio del Poder Popular para Relaciones Interiores y Justicia.

DC-dot

Es una herramienta gratuita que extrae y crea metadatos con la estructura del sistema Dublin Core, se utiliza mayormente para documentos que varían su estructura y está estrechamente relacionada con la web. Esta herramienta es un producto acabado, con características propias que no tienen relación con todos los elementos que intervienen en el proceso de Gestión de Archivo en la DAP.

El Archivo brinda y consume información de otras áreas de la DAP, por lo que debe integrarse totalmente con otros componentes de la Solución Tecnológica Integral.

1.3. Gestión de préstamos

Durante un día normal de trabajo de la División, en el Archivo se atienden solicitudes de préstamos de Expedientes que pueden llegar a través del sistema o de forma personal con el funcionario de Archivo. Los Expedientes son prestados con motivo de ser trasportados al Centro de Digitalización o para ser consultado por algún funcionario del área de Revisión Legal. Siendo la gestión de estos préstamos uno de los procesos que tiene lugar en el Archivo. *Ver anexo 5*

1.4. Metodología de desarrollo

La metodología escogida para guiar el proceso de desarrollo es Rational Unified Process (RUP) o Proceso Unificado de Desarrollo.

El contrato firmado entre Cuba y Venezuela para llevar a cabo la solución de software no incluye soporte y mantenimiento del sistema, además el cliente no estará presente durante el desarrollo, por esta razón es necesario generar los artefactos pactados que formarán parte de los entregables que recibirá el mismo una vez terminado el proyecto. Por esta razón se selecciona RUP como una metodología que permite generar gran cantidad de documentación, detallando los elementos necesarios para facilitar al cliente la comprensión del software.

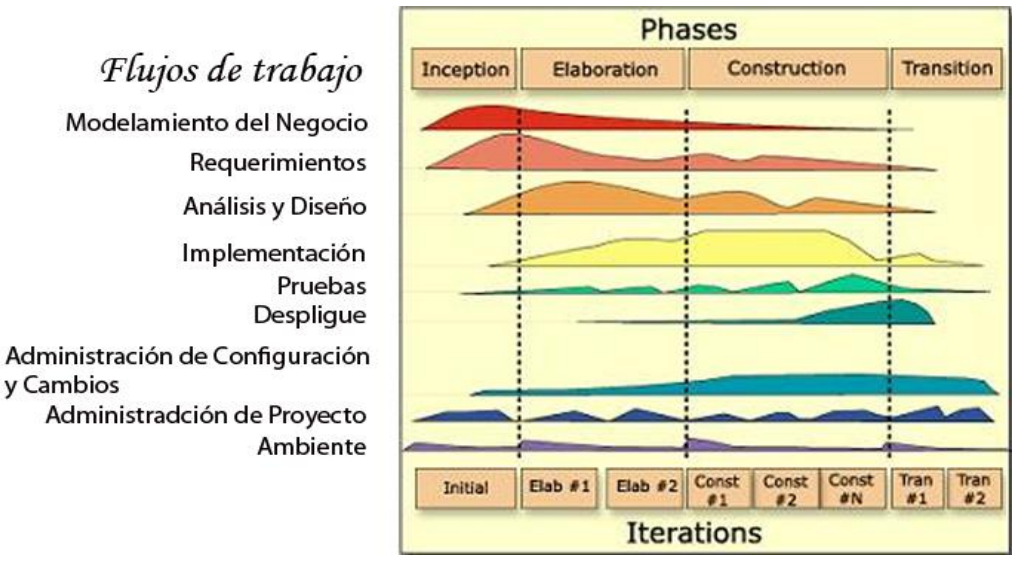


Fig.2: Proceso Unificado de Desarrollo

1.5. Herramientas de modelado

Existe una gran variedad de herramientas que permiten el modelado de soluciones informáticas, entre las más notables se encuentran RationalRose, Enterprise Architect y Visual Paradigm. Teniendo en cuenta las características de estas y las necesidades de la solución se decidió usar Visual Paradigm v3.4 ya que era la versión más estable y utilizada al inicio del proyecto. Además brinda facilidades en el trabajo colaborativo con Subversion para el control de versiones, integración con modelos relacionales de Base de Datos, el modelado de todos los artefactos propuestos por la metodología a través de UML y la generación de código Java a partir de diagramas usando ingeniería inversa todo en una sola herramienta.

1.6. Lenguaje de modelado

Como lenguaje de modelado se seleccionó el Lenguaje Unificado de Modelado v2.0, (Unified Modeling Language UML por sus siglas en inglés), ya que es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. [5]
Permite el modelado de todos los artefactos propuestos por la metodología escogida y es el lenguaje de modelado utilizado por Visual Paradigm.

1.7. ¿Qué es un patrón de diseño?

En programación orientada a objetos se entiende por patrón una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparecen repetidamente en el desarrollo de sistemas software.[6]

Los patrones de diseño pretenden:

- ✓ Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- ✓ Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- ✓ Formalizar un vocabulario común entre diseñadores.
- ✓ Estandarizar el modo en que se realiza el diseño.
- ✓ Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando

conocimiento ya existente.

Patrones GoF (Gang of Four)

Creacionales: Solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.

- ✓ **Fábrica abstracta:** proporciona una interfaz para crear familias de objetos que dependen entre sí, sin especificar sus clases concretas.
- ✓ **Constructor**: separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- ✓ **Método fábrica:** define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- ✓ **Prototipo:** especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- ✓ **Instancia única:** garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Estructurales: Solucionan problemas de composición (agregación) de clases y objetos.

- ✓ **Adaptador:** convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- ✓ **Puente:** desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
- ✓ **Compuesto:** combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
- ✓ **Decorador:** añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- ✓ **Fachada:** proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. Es importante tener en cuenta que las fachadas no implementan lógica alguna, solo constituyen una piel fina de los objetos remotos.

- ✓ **Peso ligero:** reduce la redundancia cuando gran cantidad de objetos poseen información idéntica.
- ✓ **Proxy:** proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

De comportamiento: Solucionan problemas respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

- ✓ **Cadena de responsabilidad:** evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- ✓ **Orden:** encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer las operaciones.
- ✓ **Intérprete:** dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- ✓ **Iterador:** proporciona un modo de acceder secuencialmente a los elementos de una colección independientemente del tipo de colección subyacente.
- ✓ **Mediador:** define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- ✓ **Recuerdo:** representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.
- ✓ **Observador:** define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- ✓ **Estado:** permite que un objeto modifique su comportamiento cada vez que cambia su estado interno.
- ✓ **Estrategia:** define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
- ✓ **Método plantilla:** define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
- ✓ **Visitador:** representa una operación sobre los elementos de una estructura de objetos.

Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

Patrones Generales de Software para la Asignación de Responsabilidades

(General Responsability Assignment Software Patterns GRASP por sus siglas en inglés). Este tipo de patrones describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades expresados como patrones, aunque muchos autores consideran que más que un patrón es el conjunto de buenas prácticas para el diseño de software. [7]

- ✓ **Experto:** es el principio básico de asignación de responsabilidades, lo cual nos indica que la responsabilidad de la creación de un objeto o la implementación de un método recae sobre la clase que conoce toda la información necesaria para crearlo.
- ✓ **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.
- ✓ **Alta cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas evitando que hagan más de lo que le corresponde. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo.
- ✓ **Bajo acoplamiento:** es un principio que se debe tener siempre en cuenta durante las decisiones de diseño, el acoplamiento es la medida en que una clase está conectada a otras clases. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad.
- ✓ **Controlador:** es un intermediario entre una interfaz y el algoritmo de tal manera que el usuario pueda ingresar o recibir datos dependiendo del método llamado.
- ✓ **Polimorfismo:** define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- ✓ **Fabricación pura:** este patrón desarrolla clases que se encargan de construir los objetos adecuados en cada momento. Estas son clases artificiales que no representan ningún concepto del dominio del problema, tienen un conjunto de responsabilidades altamente cohesivo y tienen un bajo acoplamiento, con lo que se consigue un diseño limpio y puro.
- ✓ **Indirección:** este patrón se basa en la creación de clases intermedias para desacoplar componentes o servicios, o asigna la responsabilidad a un objeto intermedio que medie

entre dos componentes o servicios de manera que no se acoplen directamente.

Teniendo en cuenta las características de la solución que se quiere implementar se seleccionaron los siguientes patrones:

- ✓ Instancia única
- ✓ Fachada
- ✓ Iterador
- ✓ Experto
- ✓ Creador
- ✓ Alta cohesión
- ✓ Bajo acoplamiento
- ✓ Controlador

1.8. Lenguaje de programación

Como lenguaje de programación se seleccionó Java v1.6 teniendo en cuenta que la experiencia de los desarrolladores con este lenguaje era superior con respecto a otros. Además de las características que posee y que influyeron en su selección, como son:

- ✓ **Seguro:** se elimina el uso de los punteros con el objetivo de evitar el uso innecesario de recursos en memoria. Además la máquina virtual se encarga de verificar que el software no posea fragmentos de código ilegal (código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto) antes de ejecutarlo.
- ✓ **Portable:** la compilación de las aplicaciones hechos en java son completamente independientes de la arquitectura de la estación de trabajo donde se ejecute, sólo depende de la Máquina Virtual de Java y las librerías fundamentales que sean usadas.
- ✓ **Distribuido:** Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.[8]

1.9. Entorno de desarrollo integrado

Los entornos de desarrollo integrados (IDE por sus siglas en inglés) consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Estos suelen ser una de las herramientas más importantes para desarrollar una aplicación informática.

Dentro de los IDE que se pueden encontrar para implementar aplicaciones en Java se pueden mencionar NetBeans, Eclipse, IntelliJ y Visual Studio. Para la codificación de la solución en

cuestión se decidió utilizar NetBeans v7.0.1 por las siguientes razones:

- ✓ NetBeans es un proyecto de código abierto con una comunidad en constante crecimiento en todo el mundo, además, es un producto libre y gratuito, sin restricciones de uso.
- ✓ Integra un conjunto de tecnologías muy populares en el desarrollo de aplicaciones en Java, que además son de código abierto. Proporciona interfaces gráficas de forma tal que el desarrollador pueda configurar esta integración de forma sencilla.
- ✓ Por otra parte, NetBeans v7.0.1 presenta más opciones de seguridad que facilitan el uso de roles y se integra mucho mejor con el servidor de aplicaciones Glassfish.[9]

1.10. Máquina Virtual de Java

La Máquina Virtual de Java (Java Virtual Machine (JVM) por sus siglas en inglés) constituye un elemento imprescindible para la ejecución de un software desarrollado en esta tecnología. Un programa realizado en Java tiene primeramente un proceso de compilación generando finalmente un fichero de bytecodes. Posteriormente, dicho programa compilado ya sin errores será interpretado por la JVM. La JVM no es un hardware sino un software que se instala en la estación de trabajo o donde se quiere ejecutar una aplicación implementada en este lenguaje.[9]

Para la compilación y ejecución del sistema se usó la JDK v1.0.6_26.

1.11. Servidor de aplicaciones

Un servidor de aplicaciones es un dispositivo de software que proporciona servicios a las aplicaciones clientes y generalmente gestiona la totalidad de la lógica de negocio y de acceso a los datos. Un servidor de aplicaciones debe tener los siguientes componentes:

- ✓ **Contenedor de aplicaciones cliente:** contenedor donde se ejecutan las aplicaciones clientes, ya sean applets, códigos htlm, u otros.
- ✓ **Contenedor web:** también denominado contenedor de Servlet /Java Server Page (JSP), maneja la ejecución de los servlets y páginas JSP. Es lo que comúnmente se conoce por servidor web.
- ✓ **Contenedor EJB:** es donde se ejecuta la lógica de negocio, basada en la tecnología Enterprise Java Bean (EJB).

Dentro de los servidores de aplicaciones más utilizados hoy en día se encuentran: WebLogic, IBM WebSphere, Borland AppServer, Oracle Glassfish Server, Oracle Glassfish Open Source Server,

JBoss. Muchas de estas soluciones son muy caras en el mercado y no es factible por el costo de sus licencias, excepto el Oracle Glassfish Open Source Server y JBoss que sus licencias son libres de uso.

Para implementar la solución se decidió usar Oracle Glassfish Open Source Server v3.1, ya que posee una buena implementación para la gestión de la alta disponibilidad, tolerancia a fallos, concurrencia, seguridad, transaccionalidad, y conexiones a sistemas de información externos. Posee una consola de administración web la cual permite administrar el servidor de manera remota y está soportado por una fuerte comunidad de desarrollo. Además, el contenedor EJB de Oracle Glassfish Open Source Server implementa un conjunto de servicios que serán de gran ayuda para la solución de software. A continuación se exponen algunos de los servicios implementados por el contenedor EJB:

- ✓ **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 estaciones de trabajo distintas.
- ✓ **Gestión de recursos:** gestión automática de múltiples recursos, como colas de mensajes, bases de datos o fuentes de datos en aplicaciones heredadas que no han sido traducidas a nuevos lenguajes/entornos y siguen usándose en la empresa.
- ✓ **Persistencia:** sincronización entre los datos del bean y tablas de una base de datos.
- ✓ **Gestión de mensajes:** Servicios de mensajes de Java (JMS).
- ✓ **Escalabilidad:** posibilita el uso de clúster de servidores de aplicaciones con múltiples hosts para poder dar respuesta a aumentos repentinos de carga de la aplicación con sólo añadir hosts adicionales.
- ✓ **Adaptación en tiempo de despliegue:** posibilidad de modificación de todas estas características en el momento del despliegue del bean.

1.12. Tecnologías para el desarrollo

Durante el desarrollo del sistema se utilizaron un conjunto de tecnologías que proporcionan robustez al sistema y le brindan un alto nivel de escalabilidad y mantenibilidad.

1.12.1. Enterprise Java Beans

El diseño de la solución estará centrado en el uso de la plataforma JEE, especialmente en la tecnología Enterprise Java Beans (EJB). EJB es una tecnología que permite crear una solución que se compone por un conjunto de componentes (módulos EJB). Con la tecnología EJB es posible desarrollar componentes que se pueden reutilizar y ensamblar en distintas aplicaciones dentro y fuera de la empresa.

Los componentes EJB son manejados por un contenedor EJB que se comporta como una aplicación que los gestiona. El contenedor EJB proporciona dos formas de interacción entre los componentes, una es a través de Interfaces Remotas a Métodos (RMI, por sus siglas en inglés) y la otra es a través de Servicios Web (WS, por sus siglas en inglés).

Bean de sesión: es un componente que se despliega en el lado del servidor encapsulando algún tipo de lógica de negocio y requiere ser integrado con otros componentes para que éste sea funcional. De esta forma se consigue aislar la lógica de la aplicación del diseño y aspecto, permitiendo que ambas fases puedan ser desarrolladas de forma simultánea e independiente para luego ser integradas. Otra de las características fundamentales es que siendo un componente que se ejecuta del lado del servidor delega en éste varias funcionalidades que en tiempo de desarrollo es costoso ocuparse de ellas como son la transaccionalidad, la seguridad, la concurrencia y otras más. [9]

A continuación se describen algunos de los beans de sesión:

- ✓ **Con estado (stateful):** las instancias del bean conservan la información obtenida durante la conexión con el cliente. Este estado conversacional que se establece sólo se modifica una vez que el cliente va realizando llamadas a los métodos de negocio del bean. El estado conversacional no se guarda cuando el cliente termina la sesión.
- ✓ **Sin estado (stateless):** los beans de sesión sin estado son objetos distribuidos que carecen de estado asociado permitiendo por tanto que se les acceda concurrentemente. No se garantiza que los contenidos de las variables de instancia se conserven entre llamadas al método.

1.12.2. Servicio de Mensajes de Java

Java Message Service, también conocida por sus siglas JMS, es la solución creada por Sun Microsystems para el uso de colas de mensajes. Este es un estándar de mensajería que permite a

los componentes de aplicaciones basados en la plataforma JEE, crear, enviar, recibir y leer mensajes. También hace posible la comunicación de manera síncrona y asíncrona en ambos casos. Existen dos modelos de la JMS, los cuales son:

- ✓ **Modelo Punto a Punto:** En este modelo participan solamente dos clientes, uno que envía el mensaje y otro que lo recibe. JMS asegura la llegada del mensaje ya que si el receptor no está disponible de cualquier forma se le envía el mensaje y este se encola para luego ser recibido según haya entrado.
- ✓ **Modelo Publicador/Suscriptor:** En este modelo no interesan los clientes que participan, unos que publican temas (tópicos), y los que ven estos tópicos, a diferencia del modelo anterior este modelo tiende a tener más de un consumidor. [9]

Este servicio será de gran utilidad, facilitando la comunicación entre el Funcionario de Archivo y aquellos funcionarios con permisos para realizar solicitudes de préstamos, manteniéndolos informados sobre el estado de las solicitudes a través del envío de mensajes manejados por el sistema.

1.12.3. Java Persistence API

Java Persistence API (JPA por sus siglas en inglés), es la Interfaz de Programación de Aplicaciones (API) para la persistencia y el mapeo objeto-relacional desarrollada para la plataforma Java EE. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos y permitir usar objetos comunes conocidos como POJOs (PlainOld Java Object), es decir, objetos simples que no heredan ni implementan otras clases. El principal factor de decisión de esta API es la integración que tiene con la tecnología EJB, que se utilizará para el desarrollo de la solución.

Existen varias implementaciones del API y que hoy son muy usadas, como son EclipseLink, TopLink e Hibernate. Para el desarrollo de la solución informática se decidió utilizar EclipseLink debido a que es un marco de trabajo estable por el que apuesta Oracle, desarrollador de Toplink, y que es recomendable para las soluciones que se desarrollen en java y sean desplegadas en un servidor Glassfish. [9]

1.12.4. Swing

Es un conjunto de clases de Java que simplifica la construcción de aplicaciones de escritorio. Permite tener un sistema de ventanas y componentes gráficos, independiente del sistema operativo

y la librería de dibujo que se tengan disponibles en las estaciones de trabajo clientes. Se utiliza para el diseño y construcción de los formularios de la aplicación debido a las ventajas que ofrece para el desarrollo de componentes que permiten la visualización y registro de los datos.

1.13. Sistema Gestor de Bases de Datos

La necesidad de persistir grandes volúmenes de información referente a las unidades documentales y los procesos que se ejecutan en la División de Antecedentes Penales requiere el uso de un gestor de base de datos que permita gestionar de manera ágil la información necesaria. Por esta razón se decide escoger PostgreSQL v8.4 que es un gestor de base de datos de código abierto muy útil cuando se trata de la gestión de grandes volúmenes de datos. Además, es usado en varias soluciones informáticas dentro del Ministerio del Poder Popular para Relaciones Interiores y Justicia (MPPRIJ), lo que facilita la transferencia y la mantenibilidad por la parte venezolana.

Otras de las razones que consolidan la selección, además de ser una solución libre, son las siguientes:

- ✓ Las transacciones en PostgreSQL protegen los datos de la concurrencia múltiple a través del procesamiento completo de transacciones (full transaction processing). El modelo de la transacción usado por PostgreSQL es basado en el Control Multiversión de Concurrencia (MVCC). Es una tecnología que PostgreSQL usa para evitar bloqueos innecesarios a los usuarios.
- ✓ PostgreSQL implementa integridad referencial apoyando las relaciones de llave primaria y foránea así como los triggers o disparadores.
- ✓ Permite un gran número de conexiones concurrentes. Por defecto permite 100 pero esta puede ser configurable, solo depende de la RAM del servidor.
- ✓ Recorridos de Mapas de Bits: los índices son convertidos a mapas de bits en memoria cuando es apropiado, otorgando hasta veinte veces más rendimiento en consultas complejas para tablas muy grandes.
- ✓ Particionamiento de Tablas: el optimizador de consultas es capaz de evitar recorrer secciones completas de tablas grandes, a través de una técnica conocida como Exclusión por Restricciones. Esta característica mejora tanto el rendimiento como la gestión de datos para tablas de varios gigabytes.
- ✓ Bloqueos Compartidos de Registros: el modelo de bloqueos a través de la adición de candados compartidos a nivel de registro para llaves foráneas. Estos candados compartidos mejoran el rendimiento de inserción y actualización para muchas aplicaciones.

Para la administración de la base de datos se usó pgAdmin III, ya que es una herramienta de código abierto que permite administrar bases de datos PostgreSQL. Está diseñado para desarrollar grandes bases de datos desde una herramienta simple que soporta todas las características del gestor, haciendo más simple la administración. [9]

1.14. Herramienta de pruebas

Como herramienta de prueba se seleccionó JUnit v4.5 que es un marco de trabajo de código abierto para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos Software. El marco de trabajo provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.[10]

Este marco de trabajo es uno de los más usados en la universidad para realizar pruebas al código en proyectos implementados en lenguaje Java.

1.15. Estilos arquitectónicos

Cliente-servidor

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores en el cual múltiples clientes distribuidos geográficamente, realizan peticiones a uno o más servidores.[11]

Ventajas

Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la actualización de los datos.

Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

En n capas

Estilo arquitectónico donde cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás. Una capa no tiene dependencias con la capa superior, cada capa depende solamente de la fachada que le permite comunicarse con la capa inmediata inferior, asegurando que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total.[12]

Ventajas

- ✓ Desarrollos paralelos en cada capa.
- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ Mantenimiento y soporte más sencillo.
- ✓ Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- ✓ Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Basado en componentes

La complejidad de los sistemas computacionales actuales nos ha llevado a buscar la reutilización del software existente. El desarrollo de software basado en componentes permite reutilizar piezas de código pre elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.[13]

Componente: Un componente es una parte física de un sistema (módulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes. La programación basada en componentes posibilita las áreas comunes a gran escala y la reutilización de componentes entre uno y otro proyecto.

Ventajas

- ✓ **Reutilización del software:** nos lleva a alcanzar un mayor nivel de reutilización de software.
- ✓ **Simplifica las pruebas:** permite que las pruebas sean ejecutadas probando cada uno de los

componentes antes de probar el conjunto completo de componentes ensamblados.

- ✓ **Simplifica el mantenimiento del sistema:** cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- ✓ **Mayor calidad:** dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

1.16. Pruebas de software

Pruebas de software: Consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo.[14]

1.16.1. Pruebas de Caja Blanca

La prueba de Caja Blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la Caja Blanca el ingeniero del software puede obtener casos de prueba que garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.[15]

Es por ello que se considera a la prueba de Caja Blanca como una prueba importante que se le aplica al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes y por ende una mayor calidad.[16]

Camino Básico

La prueba del camino básico es una técnica de prueba de la Caja Blanca que permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico de casos de prueba.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.
5. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Grafo de Flujo

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo. Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo. Para construir el grafo se debe tener en cuenta la notación para las instrucciones. Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

Nodo

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

Aristas

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

Regiones

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran. La cantidad de regiones es

equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. Cuando en un diseño se encuentran condiciones compuestas (uno o más operadores AND, NAND, NOR lógicos en una sentencia condicional), la generación del grafo de flujo se hace un poco más complicada.

Complejidad ciclomática

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. En términos del grafo de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino.

¿Cómo se calcula la complejidad ciclomática?

Se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como

V (G)=A-N+2

donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3**.** La complejidad ciclomática, V (G), de un grafo de flujo G también se define como

V (G) =P+ 1

donde P es el número de nodos predicado contenidos en el grafo de flujo G.

1.17.2. Pruebas de Caja Negra

Se refieren a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Una prueba de Caja Negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.[14]

Partición Equivalente

La partición equivalente es una técnica de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.[16]

Esta técnica es la más usada en la universidad por los diferentes equipos de calidad, ya que permiten evaluar el funcionamiento de los sistemas de software implementados desde la perspectiva del cliente.

¿Qué es un caso de prueba?

En la ingeniería del software, los casos de prueba o Test Case son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.[17]

1.17.3. Niveles de pruebas

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

- ✓ **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:** El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.
- ✓ Los niveles de pruebas que serán aplicados al sistema son: Pruebas de Aceptación y Pruebas de Integración.

1.17. Conclusiones del capítulo

En este capítulo se mencionaron algunos conceptos básicos relacionados con el tema de investigación, se realizó una breve descripción del proceso de Gestión de Préstamos que tiene lugar en el Archivo. Se realizó un análisis de las principales metodologías, herramientas y tecnologías a utilizar en el desarrollo donde las seleccionadas fueron:

- ✓ Metodología de desarrollo: Proceso Unificado de Desarrollo (RUP)
- ✓ Herramienta de modelado: Visual Paradigm 3.4
- ✓ Lenguaje de modelado: Proceso Unificado de Modelado (UML) 2.0
- ✓ Lenguaje de programación: Java 1.6
- ✓ Entorno de desarrollo integrado: NetBeans 7.0.1
- ✓ Servidor de aplicaciones: Open Source Glassfish Server 3.1
- ✓ Sistema gestor de base de datos: PostgreSQL 8.4
- ✓ Herramienta de pruebas: JUnit 4.5

Capítulo 2: Propuesta de Solución

Introducción

En el presente capítulo se realiza el diseño de la solución a partir de la especificación de los requisitos de software y los diagramas de casos de uso del sistema. Se exponen los patrones de diseño utilizados y se realiza una breve descripción de la arquitectura para lograr el entendimiento del diseño. Se muestran como resultado del modelo de diseño, los diagramas de clases del diseño, los diagramas de paquetes y de componentes, además las normas de diseño, estándares de codificación y el modelo de datos utilizado en el sistema.

2.1. Especificación de requisitos

Durante la fase inicial del proyecto el equipo de analistas realizó la captura y especificación de los requisitos que sirven de entrada a los siguientes flujos de trabajo que propone RUP.

En el expediente de proyecto se encuentra el documento donde quedaron recogidos los requisitos funcionales y no funcionales pactados con el cliente con los que el sistema debe cumplir.[18]

2.1.1. Requisitos funcionales de Archivo

RF_A.001	Buscar unidad documental.
RF_A.002	Mostrar unidades documentales.
RF_A.003	Dar entrada a tipo documental.
RF_A.004	Mostrar solicitudes de préstamo de expedientes.
RF_A.005	Notificar atención a solicitud.
RF_A.006	Buscar expedientes para préstamo.
RF_A.007	Prestar expediente.
RF_A.008	Devolver expediente en préstamo.
RF_A.009	Actualizar inventario.
RF_A.010	Consultar inventario.
RF_A.011	Cambiar categoría de expediente en archivo.
RF_A.012	Ver registro de entrada.

2.2. Sistema para la Gestión del proceso de preparación y organización del Fondo Documental.

Debido a la situación actual del Archivo físico de la División, el mismo ha sido sometido a un proceso de limpieza y organización que consiste en quitar las grapas que unen las páginas, imprimir pegatinas con código de barras para registrar los Expedientes, ubicar los documentos que pertenecen a un mismo proceso penal dentro del mismo Expediente, asociarle los metadatos y la cantidad de folios correspondientes, además de controlar la entrada y salida de los documentos al Archivo durante el proceso. Todo esto con el objetivo de preparar el fondo documental para su posterior digitalización, para ello fue necesario realizar un pequeño sistema **(Fondig)** que se encargara de gestionar todo el proceso mientras se desarrollaba la Solución Tecnológica Integral. Una vez culminado la preparación y organización de todo el fondo documental el sistema terminaría su vida útil.

2.2.1. Requisitos funcionales del sistema Fondig.

RF_I.001 Insertar datos del expediente.

RF_I.001 Consultar datos del expediente.

RF_I.003 Ver detalles del expediente.

RF_I.004 Adicionar tipo documental.

RF_I.005 Editar tipo documental.

RF_I.006 Imprimir etiquetas.

RF_I.007 Prestar expediente.

RF_I.008 Realizar reporte de producción diaria.

2.2.2. Modelo de datos del sistema Fondig

El sistema Fondig cuenta con una base de datos sencilla que está formada por 5 tablas, 3 de ellas para el almacenamiento de datos y 2 nomencladores, debido al carácter temporal que tiene, el volumen de datos se limitará a los expedientes que han sido registrados después de pasar por el proceso de limpieza y preparación, una cantidad equivalente a los documentos físicos almacenados en archivo. *Ver figura 3*

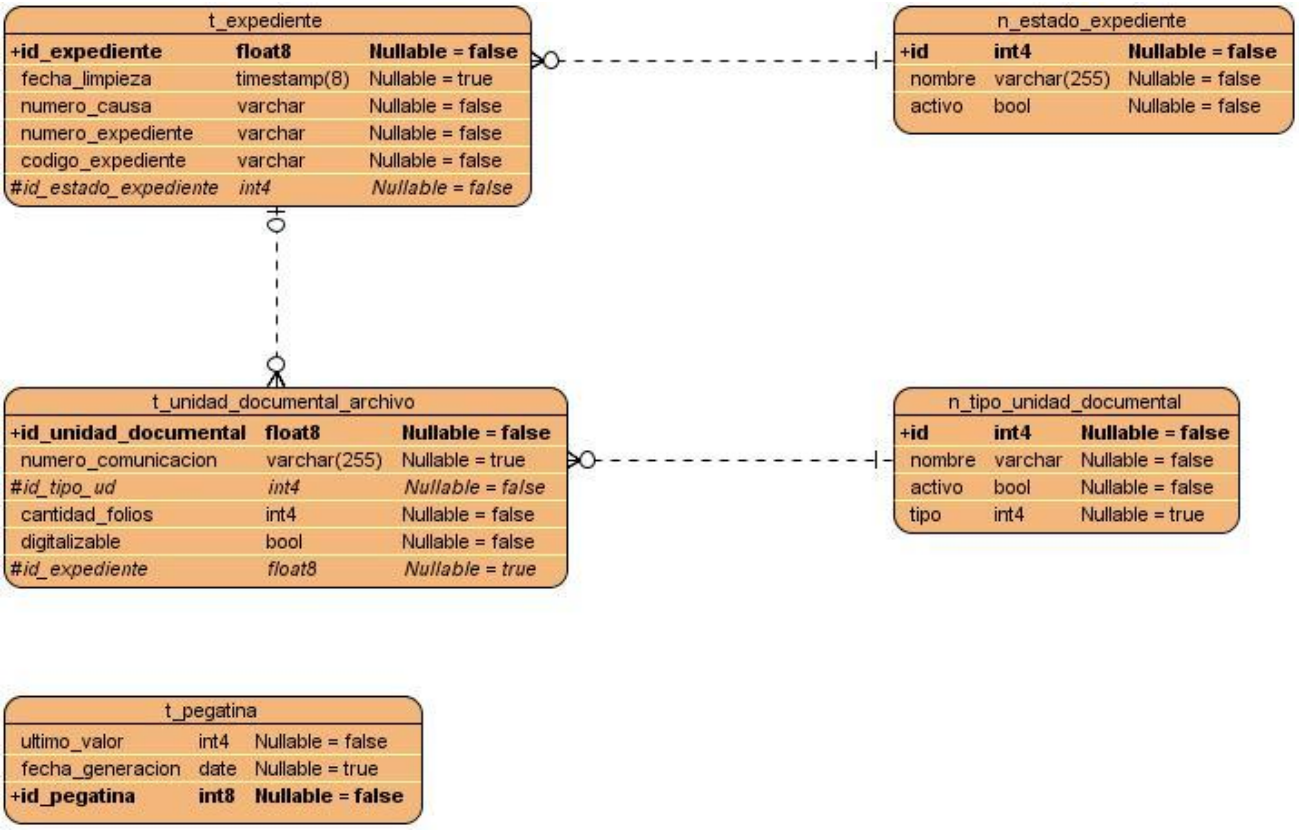


Fig. 3: Modelo de datos del sistema Fondig

2.3. Diagrama de clases del diseño

El modelo de diseño proporciona detalles acerca de la estructura de datos, las arquitecturas, las interfaces y los componentes de software que son necesarios para implementar el sistema.[19]

Como parte del modelo de diseño se realizaron los diagramas y descripciones de clases por caso de uso donde se obtuvieron un total de 11 diagramas de clases y 59 descripciones, conformado de la siguiente manera: [20]

- Clases Acción 18: que serán las encargadas de controlar los formularios con los que interactúa el usuario.
- Gestores de Negocio del Cliente (GestoresNC) 8: que serán invocados por las acciones para enviar los datos procesados del usuario hacia el servidor.
- Gestores de Negocio del Servidor (GestoresNS) 8: que se comunican con los GestoresAD a través de una fachada enviando los datos hacia el acceso a datos.

- Gestores de Acceso a Datos (GestoresAD) 8: que contienen la lógica de acceso a los datos, se encargarán de persistir, eliminar o recuperar los datos.
- Entidades Persistentes 10: son objetos simples que no implementan lógica y que contienen los datos que serán almacenados o recuperados, con el objetivo de no perder el paradigma de la orientación a objetos al interactuar con la base de datos.
- Nomencladores 1: es una tabla de valores definidos por el usuario que no pueden ser modificados.

A continuación se muestran el diagrama de clases del diseño y la interfaz correspondiente al caso de uso Gestionar Préstamos por ser el más complejo y abarcador de los implementados, además es uno de los procesos más importantes que tiene lugar en el Archivo.

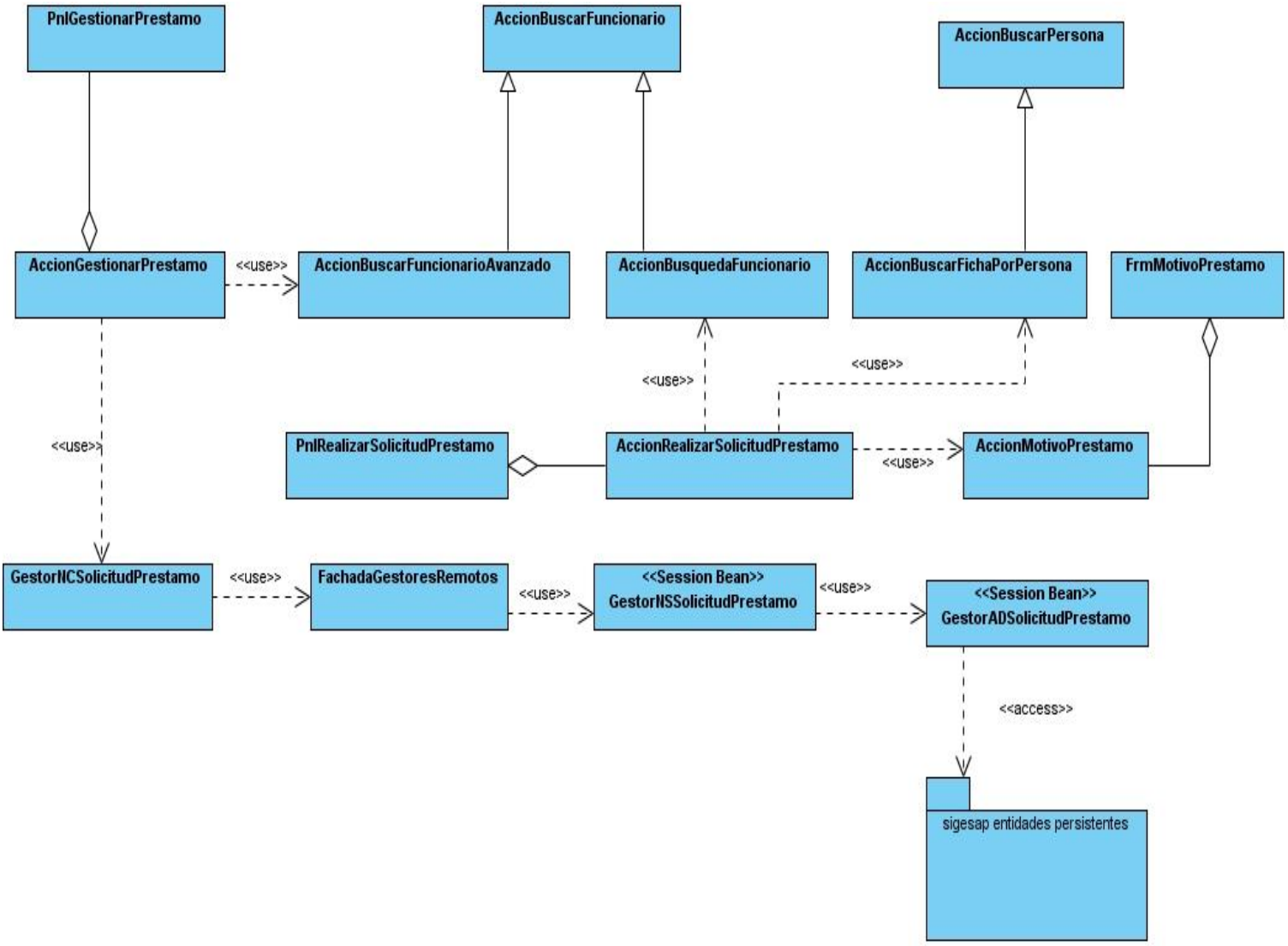


Fig. 4: Diagrama de clase del caso de uso Gestionar Préstamos.

Descripción de la clase AccionGestionarPrestamos**.**

La clase implementa los métodos necesarios para controlar el formulario PnlGestionarPrestamos.

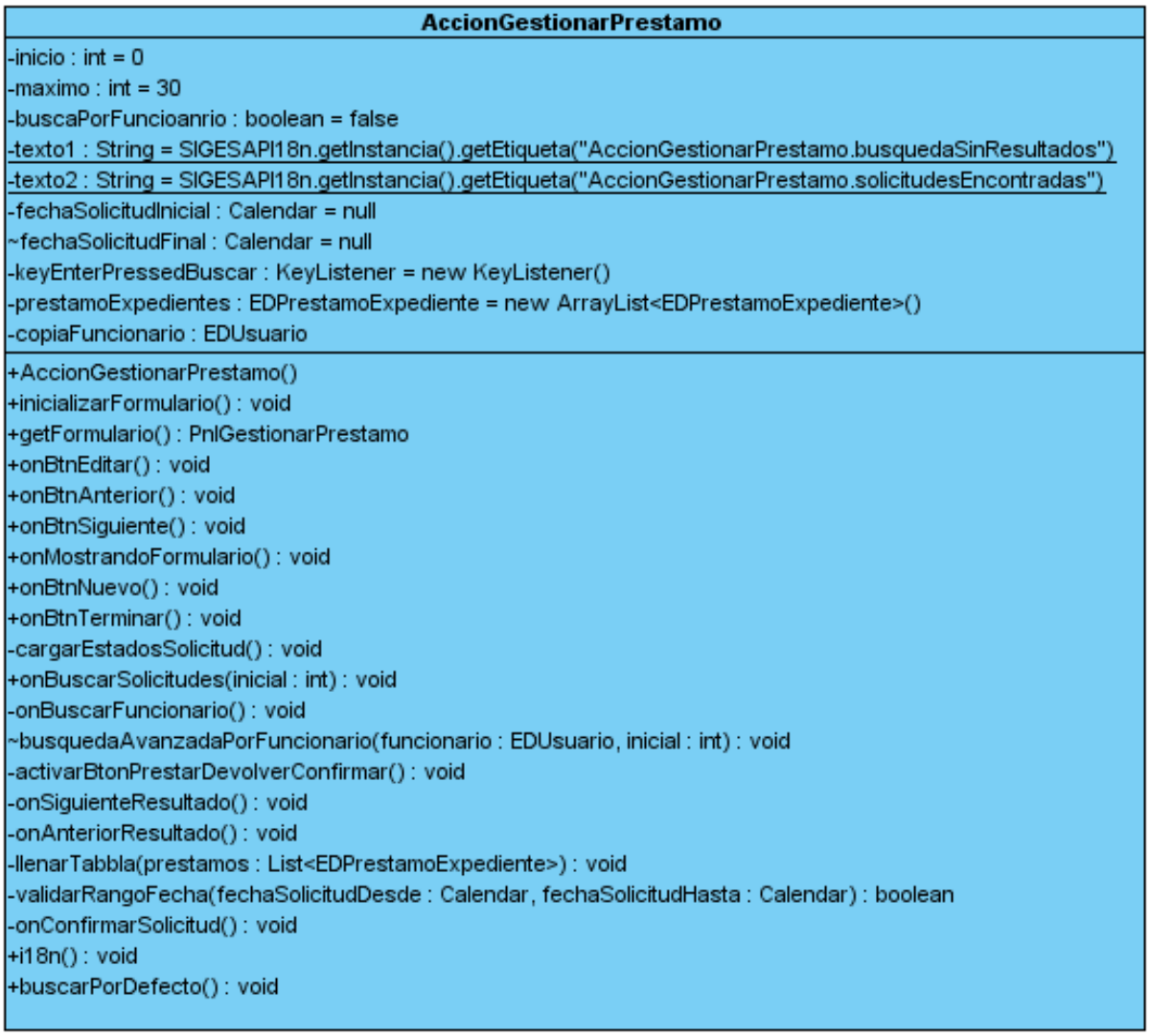


Fig. 5: Clase AccionGestionarPrestamos

La figura muestra la interfaz del caso de uso Gestionar Préstamos.

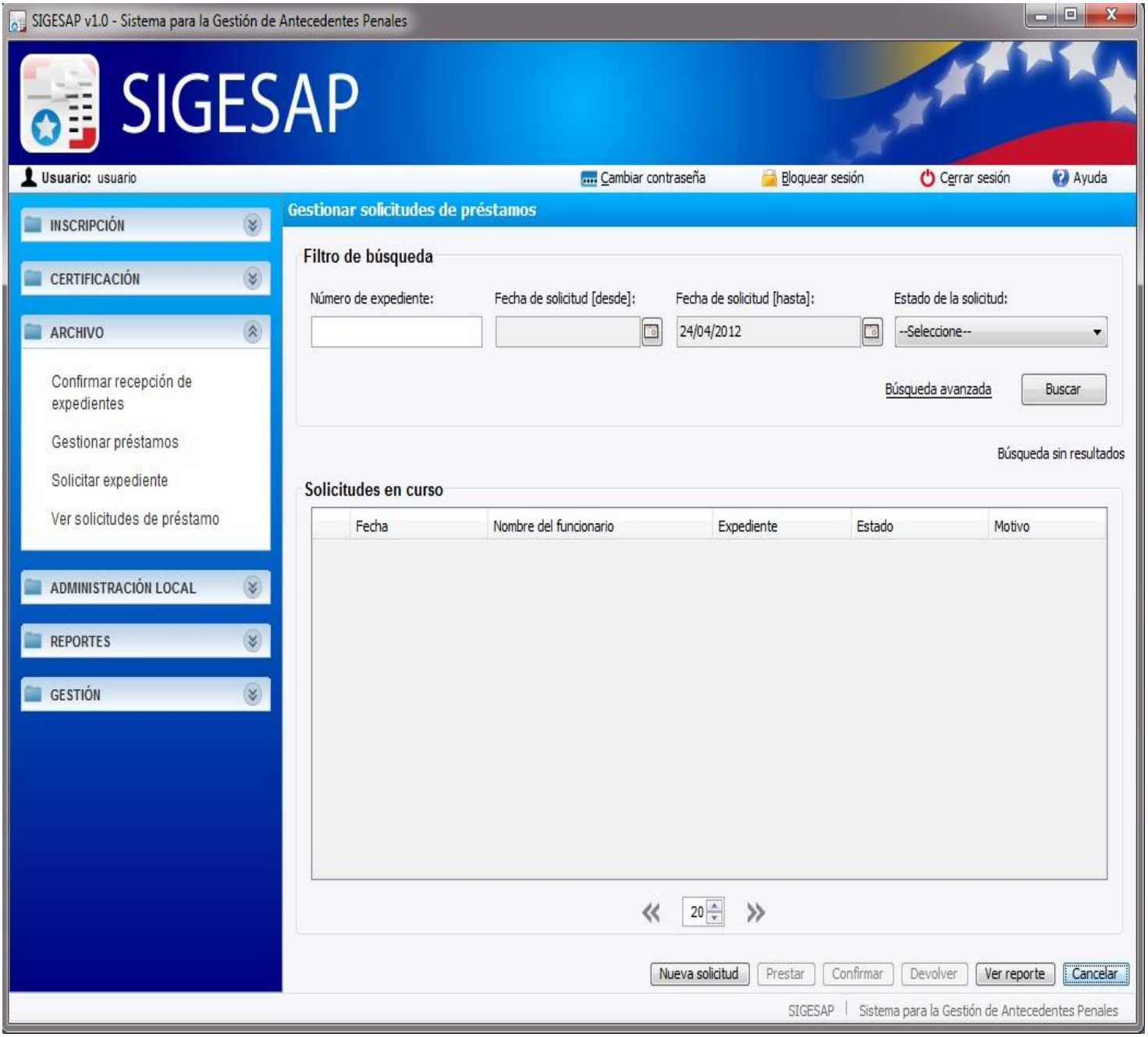


Fig. 6: Interfaz principal del caso de uso Gestionar Préstamos

2.4. Patrones de diseño utilizados

Los patrones de diseño utilizados en el desarrollo de la solución son:

Instancia única: Se usa en la clase FachadaGestoresRemotos que permite la comunicación entre el cliente y el servidor. *Ver figura 31 en el anexo 6*
Además se usa en la clase SIGESAPI18n, esta clase permite el desarrollo de sistemas de software ajustables a cualquier idioma y permite utilizar un estándar en los nombres de los componentes del formulario que permitirá modificaciones sin necesidad de hacer cambios en la ingeniería ni el código fuente. *Ver figura 32 en el anexo 6*

Fachada: Para el diseño que aquí se realiza, el uso de este patrón permitirá contar con la clase FachadaGestoresRemotos que servirá de puerta de enlace entre la capa de negocio del cliente y la capa de negocio del servidor comunicándose solamente con las interfaces remotas de los gestores de negocio del servidor y no con las clases que implementan código. Las interfaces remotas al igual que las locales son fachadas también. *Ver figura 33 en el anexo 6*

Iterador: Se utiliza en los gestores de acceso a datos para acceder a las colecciones de entidades persistentes que se obtienen a través de consultas a la base de datos. *Ver figura 34 en el anexo 6*

Experto: Se utiliza en los gestores ya que estos manejan una información específica referente a la responsabilidad que le corresponde. Por ejemplo el GestorNSSolicitudPrestamo que solo maneja información de las Solicitudes de Préstamos no se le debe pedir datos de los Expedientes porque no es su responsabilidad. *Ver figura 35 en el anexo 6*

Creador: En el proyecto las clases que tienen la responsabilidad de crear objetos son las clases Factorías que se encargan de convertir de una entidad de dominio a una entidad persistente, creando una instancia de la última para ser almacenada en la base de datos a la cual le pasa la información de la entidad de dominio, lo mismo sucede cuando se recuperan datos de la base datos. *Ver figura 36 en el anexo 6*

Alta cohesión y bajo acoplamiento: Entre estos patrones se trata de mantener un balance de manera que se diseñen clases que implementen pocas funcionalidades altamente relacionadas y que eviten la multiherencia. En el diseño de la solución se trató de lograr este equilibrio creando los gestores de negocio del servidor que son clases que cuentan con pocas funcionalidades,

implementadas de acuerdo a la información que manejan. Además comparten responsabilidades con otras clases como los gestores de acceso a datos con los que se comunica a través de interfaces para la persistencia y recuperación de la información de la base de datos y las factorías para la conversión de entidades de dominio y entidades persistentes. *Ver figura 35, 36 y 37 en el anexo 6*

Controlador: Este patrón se usa en las Acciones que sirven de intermediarios entre las interfaces y los gestores de negocio del cliente controlando las acciones que realiza el usuario según el método llamado. *Ver figura 38 en el anexo 6*

2.5. Estándares de codificación

Los estándares de codificación son un estilo de programación homogénea en un proyecto. Estos son de vital importancia durante la etapa de construcción del software, ya que son un conjunto de reglas a seguir por los desarrolladores con el objetivo de establecer un orden y un formato común en el código fuente del software en desarrollo, a continuación se muestran ejemplos de estos estándares aplicados en el proyecto.[21]

En los casos que los nomencladores sean compuestos por varias palabras se usará una notación de “Camello”. La notación “Camello” consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúscula: EndOfFile. Se llama notación “Camello” porque los identificadores recuerdan las jorobas de un camello. Existen dos variantes:

1. UpperCamelCase: en esta variante la primera letra también es mayúscula.
2. lowerCamelCase: la primera letra es minúscula.

En el caso del subsistema SIGESAP se utilizará la segunda variante para definir los atributos y variables privadas.

Ejemplos:

- ✓ private Calendar fechaSolicitudInicial
- ✓ private boolean buscaPorFuncioanrio

Los paquetes (package) 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.Modulo.Capa.* Para el caso de los paquetes se usará la misma

nomenclatura definida para los métodos y clases.

- ✓ Camello se utilizará en los nombres de las Entidades Persistentes, Entidades de Dominio, Propiedades y Funcionalidades del sistema.

  Ejemplos:

  - Entidades Persistentes: TPrestamoExpediente
  - Entidades de Dominio: EDPrestamoExpediente
  - Propiedades: getEstadoPrestamo
  - Funcionalidades: realizarSolicitud

- ✓ Las clases persistentes comenzarán con las siglas definidas en la BD para cada tabla.

  Ejemplos:

  - Tablas: t_prestamo_expediente
  - Nomencladores: n_estado_prestamo

- ✓ Los formularios de tipo JDialog comienzan con las siglas Frm, y los de tipo panel con las siglas Pnl.

  Ejemplos:

  - PnlGestionarPrestamos
  - FrmMotivoprestamo

  ✓ Los gestores de negocio del cliente comienzan con el prefijo GestorNC.

  Ejemplos:

  - GestorNCSolicitudPrestamo
  - GestorNCExpediente

  ✓ Los gestores de negocio del servidor comienzan con el prefijo GestorNS.

  Ejemplos:

  - GestorNSSolicitudPrestamo

- GestorNsExpediente

✓ Los gestores de acceso a datos comienzan con el prefijo GestorAD.

Ejemplos:

- GestorADSolicitudPrestamo
- GestorADExpediente

✓ Las interfaces publicadas en el servidor para los gestores de negocio terminan con el sufijo Remoto.

Ejemplos:

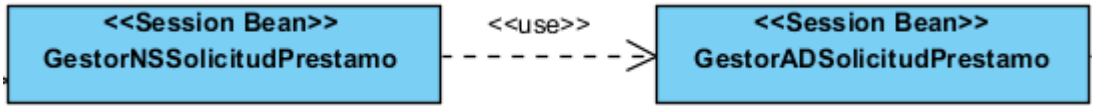
- GestorNSSolicitudPrestamoRemoto
- GestorNSExpedienteRemoto

✓ Las interfaces publicadas en el servidor para los gestores de acceso a datos terminan con el sufijo Local.

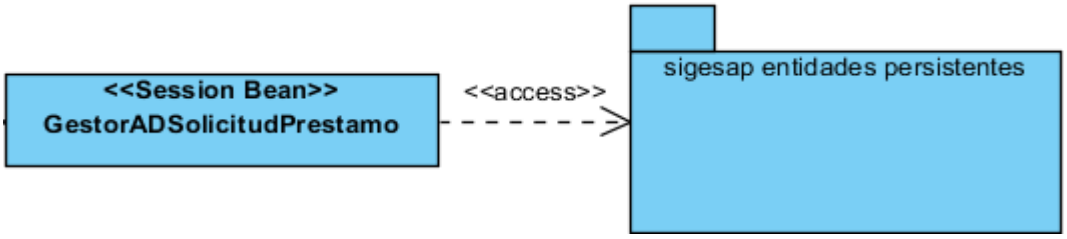
Ejemplos:

- GestorNSSolicitudPrestamoLocal
- GestorNSExpedienteLocal

✓ Las relaciones entre clases usarán el estereotipo <<use>>.



✓ Las relaciones entre clases y paquetes usarán el estereotipo <<access>>.



2.6. Pautas para el diseño de interfaces

A continuación se muestran las pautas utilizadas para el diseño de las interfaces del sistema.[22]

- ✓ La separación de los componentes respecto a los bordes del área de trabajo (margen izquierdo, derecho, superior e inferior) debe ser de 10 píxeles como mínimo.
- ✓ En un formulario, los datos asociados a una misma cosa se agruparán utilizando paneles con título. El título del panel se escribe en formato de oración, en negrita y sin dos puntos al final.
- ✓ Cuando el formulario tiene varios componentes. El nombre de las etiquetas se escribe encima de los componentes. El texto en las etiquetas se escribe en formato de oración, sin utilizar negrita y terminado en dos puntos. Los componentes irán debajo a una distancia de 8 píxeles. Cada componente de entrada de datos debe tener una altura de 23 píxeles. La separación horizontal entre componentes debe ser de 20 píxeles y vertical de 15 píxeles. Todo alineado a la izquierda, y en caso de que aparezcan unos debajo de los otros se debe tratar de que todos los componentes tengan un mismo tamaño (el del mayor componente).
- ✓ En caso de que el formulario tenga pocos componentes, se pondrá primero la etiqueta y al lado el componente, tanto los componentes como las etiquetas con las mismas pautas anteriores. Además se alinean las etiquetas a la izquierda y los componentes también a la izquierda.
- ✓ El texto de las etiquetas, títulos de columnas de tablas y botones debe ser siempre en formato de oración.
- ✓ Las listas desplegables o combobox deben mostrar como texto inicial –Seleccione--.
- ✓ Cuando haya que introducir una fecha debe ponerse un componente para seleccionar la misma, evitar las entradas manuales del usuario.
- ✓ Los botones siempre estarán situados en la parte inferior derecha de los formularios**,** con una altura de 25 píxeles y el ancho depende de la propiedad Caption (título), siempre

dejando un espacio de 8 píxeles delante y detrás de la palabra o el ícono. Sólo en el caso de los mensajes de confirmación, avisos y errores los botones irán en el centro. El texto que contienen debe estar en formato de oración y sin utilizar negrita. Cuando aparezcan varios botones uno al lado del otro, de ser posible todos deben tener el mismo ancho y la separación entre ellos debe ser de 10 píxeles. El ancho mínimo de los botones debe ser de 75 píxeles.

- ✓ Los botones no llevan íconos, el Anterior y Siguiente llevan los caracteres <,> respectivamente.

La *figura 6* muestra las pautas de diseño utilizadas en el diseño de las interfaces del módulo Archivo.

2.7. Arquitectura del sistema

En la construcción del sistema convergen tres estilos arquitectónicos comenzando por el cliente - servidor seleccionado a partir de las ventajas que proporciona, ya que permite separar las funciones y responsabilidades de la aplicación cliente de la aplicación servidora en ordenadores diferentes reduciendo el impacto de los cambios que puedan surgir en cualquiera de ellos. Junto a este estilo están contempladas las diferentes capas lógicas y los componentes utilizados en cada una que responden a una arquitectura en n capas y basada en componentes.

A continuación se representa la arquitectura del sistema en la parte del cliente.

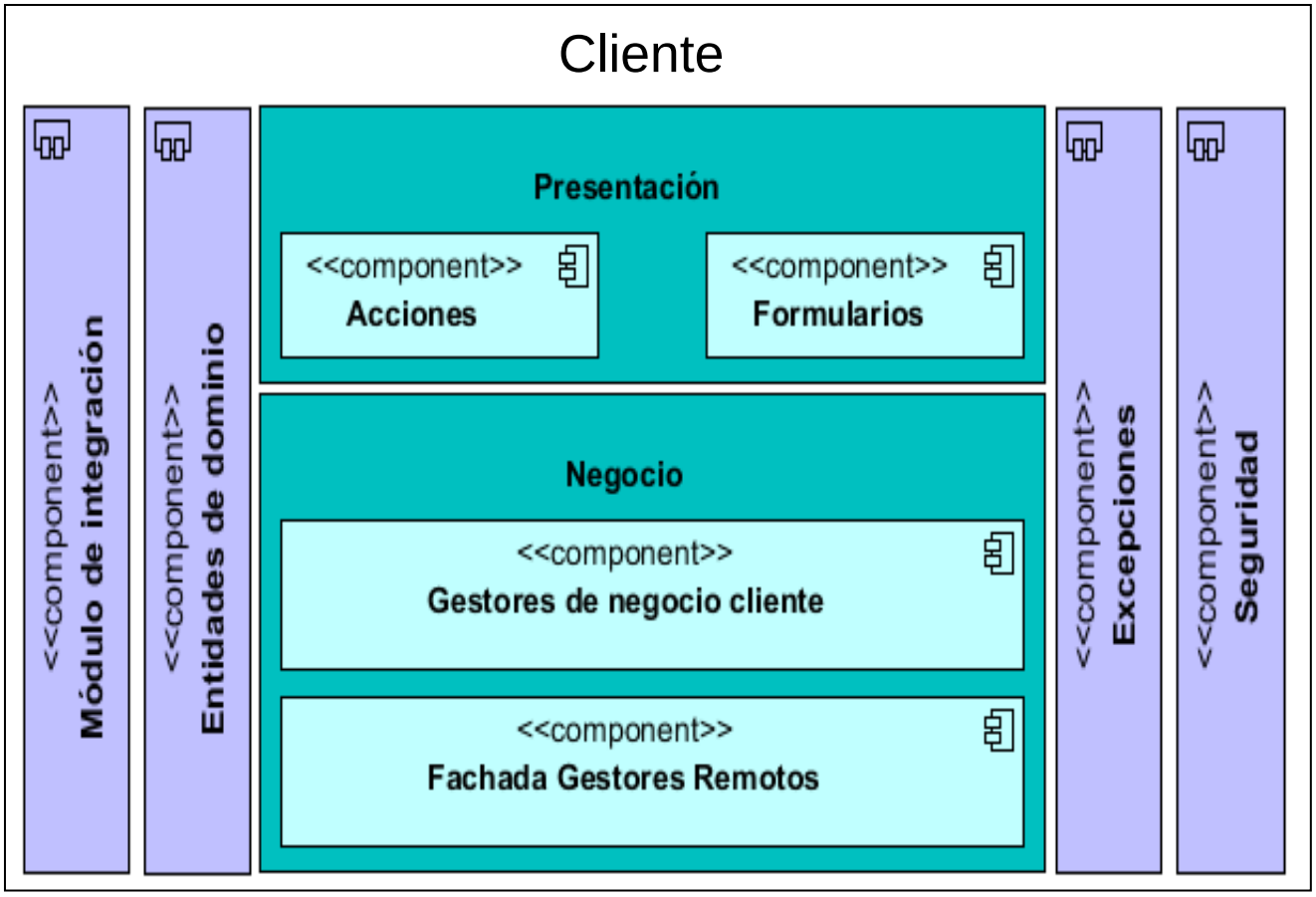


Fig.7: Representación de la arquitectura del sistema en el cliente

En el cliente se encuentran capa de Presentación que contiene las acciones que controlan los formularios con los que interactúan los usuarios finales y la capa de Negocio donde se procesan las peticiones de los usuarios recibidas de la capa de Presentación y enviadas hacia el servidor. Contiene los gestores de negocio del cliente y la FachadaGestoresRemotos que sirve de puerta de enlace con el servidor.

A continuación se representa la arquitectura del sistema en la parte del servidor.

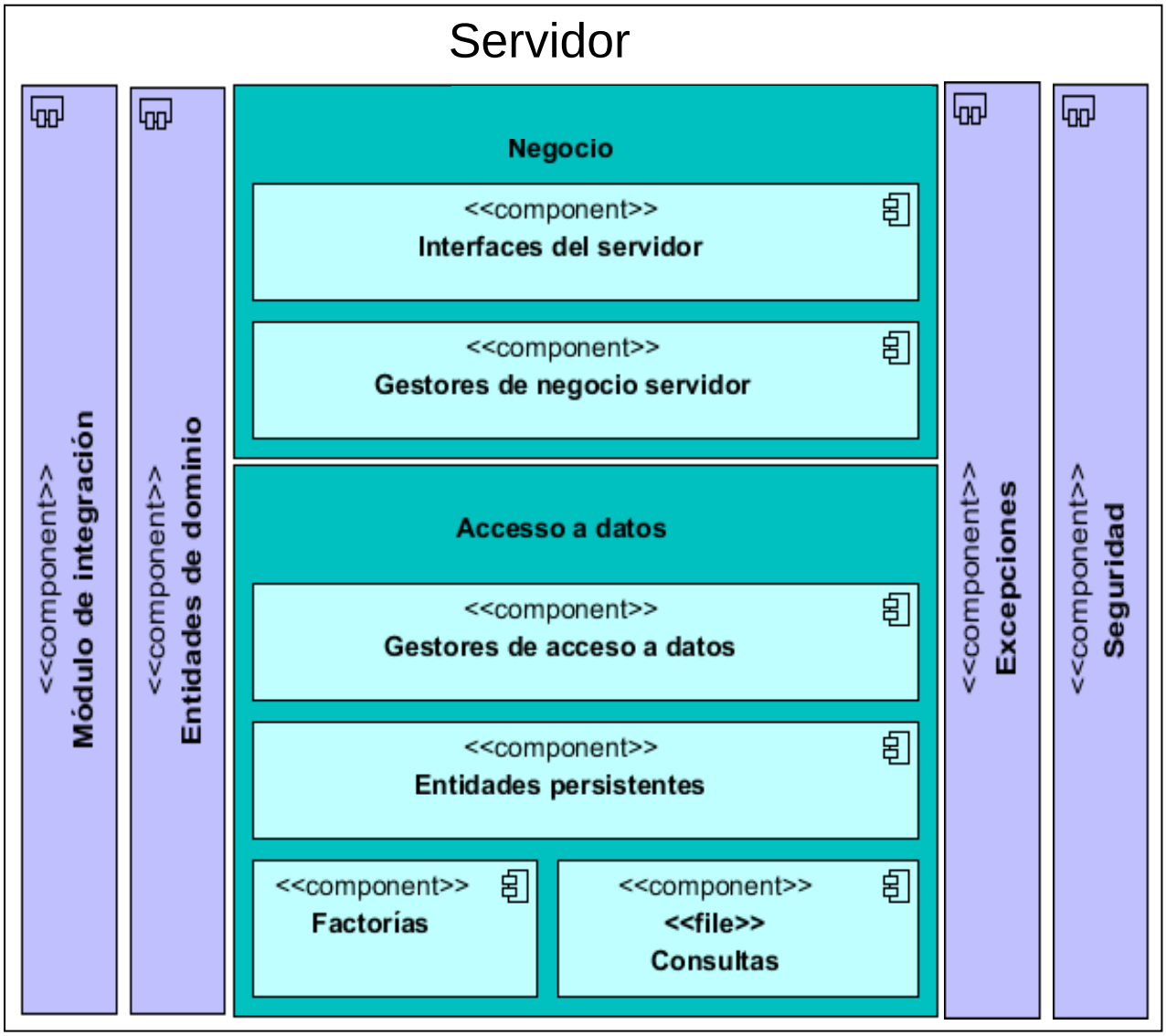


Fig.8: Representación de la arquitectura del sistema en el servidor

En el servidor se encuentra la **capa de Negocio** que se encarga de enviar los datos procesados por el cliente hacia la capa inmediata inferior a través de interfaces, contiene los gestores de negocio del servidor. También está presente la **capa de Acceso a Datos** que se encarga del almacenamiento y recuperación de los datos, contiene los gestores de acceso a datos que conjuntamente con los gestores del servidor son **beans de sesión sin estados**, las Entidades Persistentes que constituyen **beans de entidad**, obtenidas del mapeo objeto-relacional de las tablas de la base de datos, las Factorías para la conversión de dichas entidades a entidades de dominio del negocio y los ficheros de consultas que son mapeados a través de JPA.

Por otra parte, representados de forma vertical a cada una de las capas se encuentran **el módulo de Integración** que es un componente completamente reutilizable donde se encuentran implementadas funcionalidades comunes que serán usadas por los demás módulos del proyecto, las **entidades de dominio** del negocio, las **Excepciones** que serán capturadas y tratadas durante la implementación de la lógica de negocio tanto en el cliente como en el servidor y el **componente de seguridad** que permite la autenticación de los usuarios en el sistema, así como la asignación de permisos y privilegios según el rol que ocupan y el bloqueo de sesiones por inactividad.

La capa de Datos que no se representa en el esquema ya que está ubicada en el servidor de base de datos y que contiene los datos gestionados por la aplicación, las secuencias, funciones y los disparadores que permiten realizar el acceso tanto para inserción, actualización, eliminación (en algunos casos) y consulta de datos de manera correcta.

2.8. Modelo de datos.

La base de datos del subsistema SIGESAP se encargará de almacenar los documentos que serán registrados a través del proceso de Inscripción además de los Expedientes digitalizados procedentes de DIGIDAP que recibirá a través de réplica, lo que implica un crecimiento del volumen de información con el paso de los años.

La nomenclatura utilizada en el modelo es la siguiente:

Para el nombre de las tablas se inicia con la letra (t) seguidamente el nombre de la tabla separado por un guión bajo (_).

Ejemplos: t_prestamo_expediente

t_expediente

Para el nombre de los nomencladores se inicia con la letra (n) seguidamente el nombre de la tabla separado por un guión bajo (_).

Ejemplo: n_estado_prestamo

A continuación se muestra el modelo de datos del módulo Archivo que cuenta con 10 tablas de persistencia y 1 nomenclador.

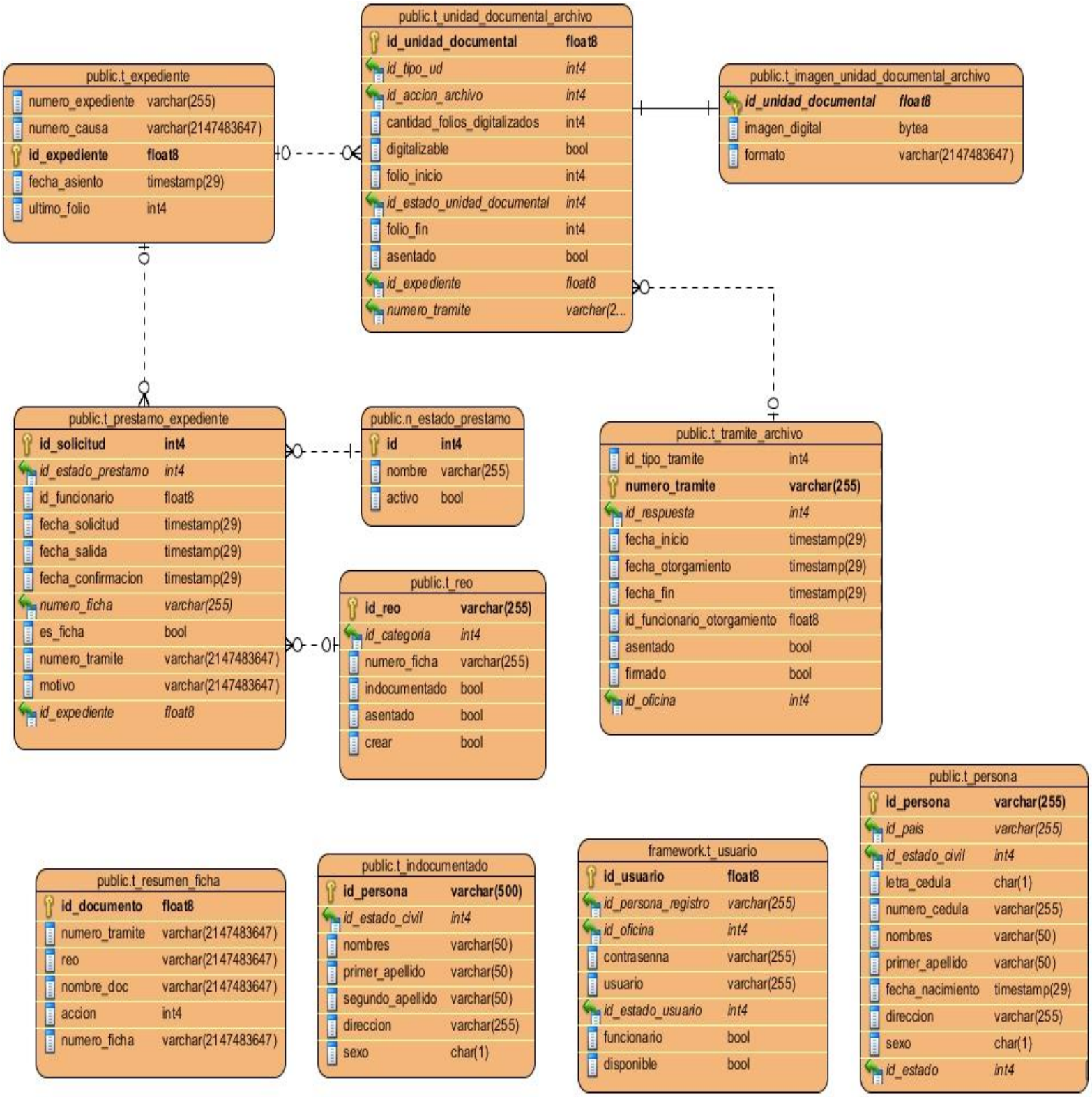


Fig.9: Modelo de datos del módulo Archivo

2.9. Diagrama de paquetes del sistema

Los diagramas de paquetes se usan para reflejar la organización de paquetes y sus elementos. Cuando se usan para representaciones, los diagramas de paquete de los elementos de clase se usan para proveer una visualización de los espacios de nombres. Los usos más comunes para los diagramas de paquete son para organizar diagramas de casos de uso y diagramas de clase, a pesar de que el uso de los diagramas de paquete no es limitado a estos elementos UML.[23]

La figura muestra la estructura de paquetes que tiene el módulo Archivo y su relación con los paquetes externos que se utilizaron.

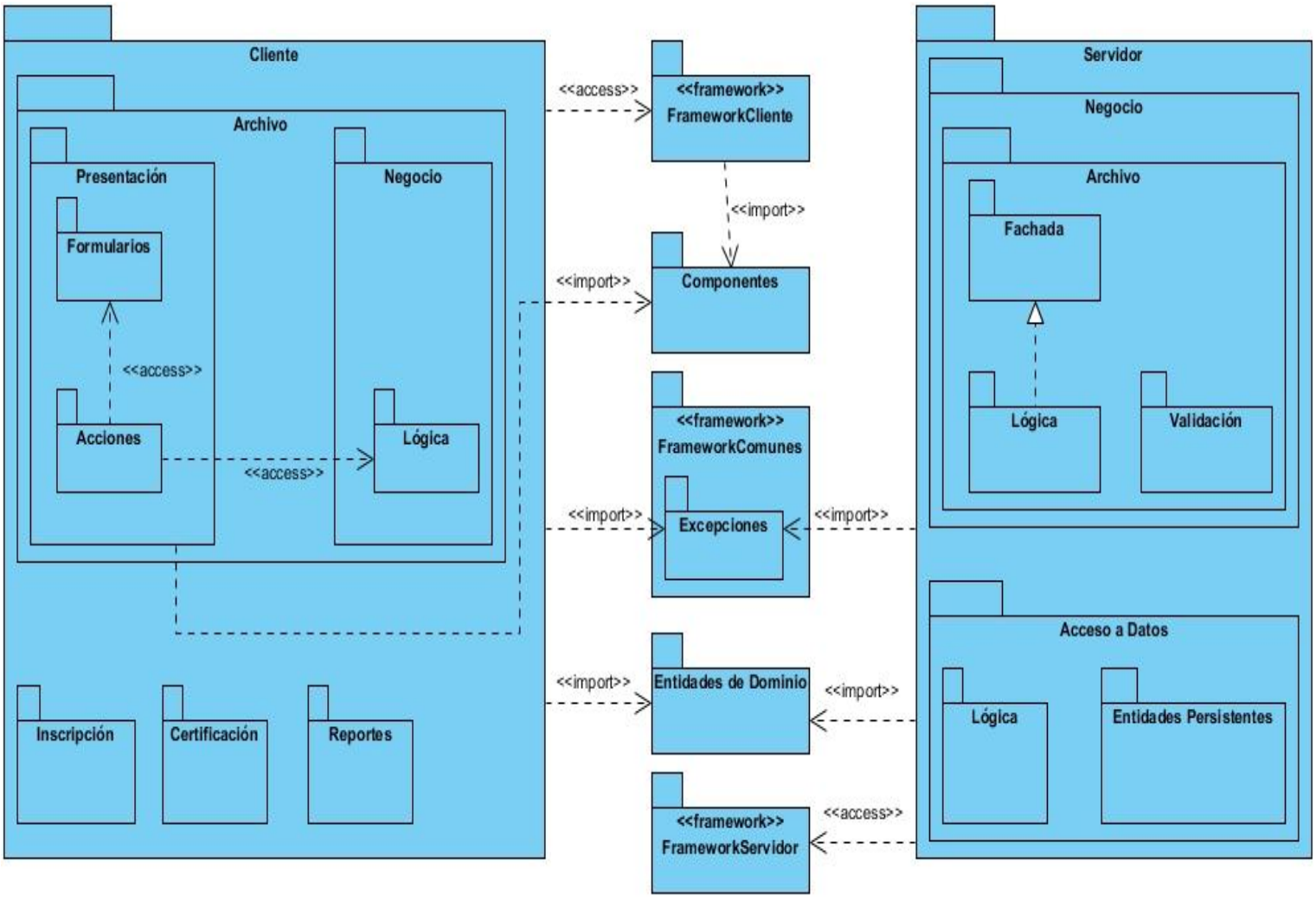


Fig.10: Diagrama de paquetes del módulo Archivo

2.9.1. Descripción de paquetes

La propuesta de solución se encuentra ubicada dentro del subsistema SIGESAP.

Los paquetes que se encuentran en el cliente son:

Imágenes: Guarda las imágenes e íconos que serán utilizadas en el subsistema propiciando organización y rapidez para cualquier necesidad de cambio.

Ayuda: Contiene los ficheros y paquetes necesarios para mostrar la ayuda de SIGESAP dividida por módulos.

I18n: Permite diseñar el software de manera que pueda adaptarse a diferentes idiomas, regiones y permite realizar cambios a nombres de botones, tablas, paneles y otros componentes en las interfaces sin la necesidad de realizar cambios de ingeniería ni en el código.

Negocio: En este paquete se encuentran las clases de apoyo a la implementación del negocio cliente que son los gestores encargados de comunicarse con la lógica del servidor a través de la fachada.

Presentación: Contiene los paquetes “acciones” y “formularios”, el primero contiene las acciones por cada caso de uso que implementan las funcionalidades que controlan los formularios y el segundo contiene las interfaces diseñadas para interactuar con el usuario.

Fig.11: Estructura de paquetes en NetBeans del subsistema SIGESAP cliente.

Los paquetes externos usados por el módulo Archivo son:

FrameworkCliente: Posee la implementación de la lógica de negocio del cliente para la gestión del marco de trabajo utilizado durante el desarrollo.

FrameworkServidor: Posee la implementación de la lógica de negocio del servidor para la gestión del marco de trabajo utilizado durante el desarrollo.

FrameworkComunes: Contiene el paquete Excepciones en el cual se encuentra las clases de excepciones que serían manejadas en la implementación y el paquete Útiles que contiene un conjuntos de clase de apoyo a la implementación. El paquete de forma general, fue un elemento común utilizado en todo el proyecto. Para ver la estructura de paquetes en Netbean ver *figura 39 en el anexo 7.*

Componentes: Contiene un conjunto de las librerías utilizadas en desarrollo.

Entidades de dominio: Contiene las entidades que fueron utilizadas para dominio del negocio durante la implementación.

Los paquetes que se encuentran en el servidor son:

Negocio: En este paquete se encuentran las clases de apoyo a la implementación del negocio servidor que son los gestores encargados de comunicarse con la lógica de acceso a datos a través de las interfaces.

Acceso a datos: En este paquete se encuentran las clases de apoyo a la implementación de la lógica de acceso a datos que son los gestores encargados de la persistencia y recuperación de la información y las entidades persistentes que se obtuvieron del mapeo de las tablas de la base de datos. *Ver figura 12*

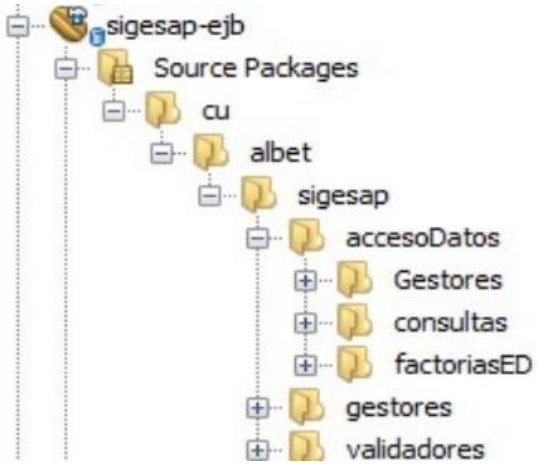


Fig.12: Estructura de paquetes en NetBeans del subsistema SIGESAP servidor.

2.10. Diagrama de componentes del sistema.

Durante la etapa de implementación se obtuvo como artefacto el diagrama de componentes del sistema en el cual se muestra como están relacionados cada uno de las componentes en las distintas vistas del sistema.

La vista de Presentación muestra la relación entre las acciones y los formularios utilizados para procesar los datos del usuario, así como su interacción con componentes externos necesarios para su funcionamiento, los principales son: El módulo de Integración del cual se usa la clase AccionBuscarFuncionario y el formulario PnlBuscarFuncionario, las entidades de dominio del Negocio, el módulo FrameworkComunes donde se encuentran las Excepciones que serán manejadas por el sistema y el componente de Seguridad que permite la autenticación de los usuarios y el bloqueo de secciones por inactividad. *Ver figura 13*

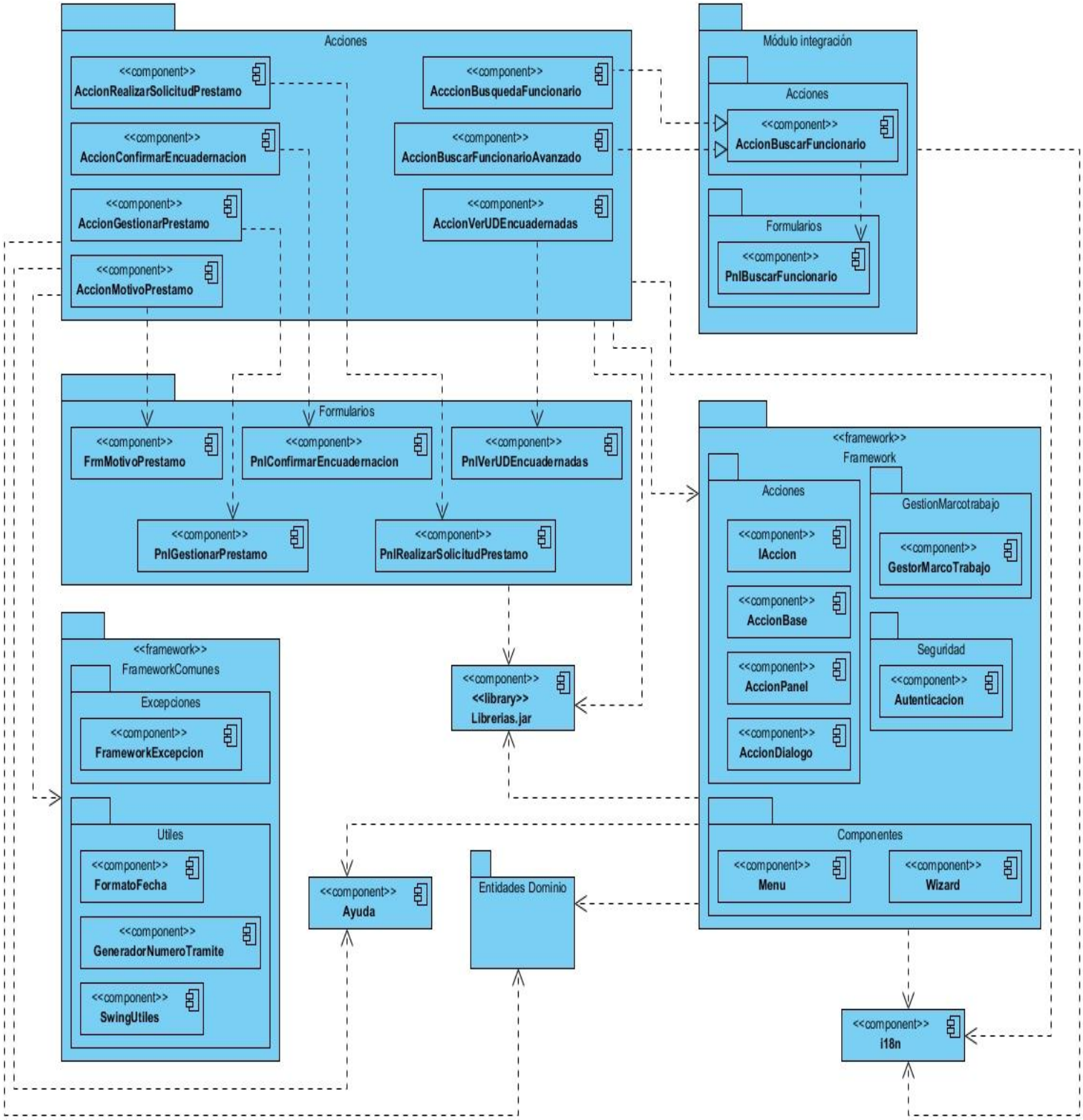


Fig.13: Vista de presentación

En la vista de Negocio se puede apreciar la comunicación entre los gestores de negocio del cliente y los gestores de negocio del servidor a través de la FachadaGestoresRemotos haciendo uso del patrón de diseño Fachada, así como su relación con las entidades de dominio y las interfaces que implementan los gestores del servidor con el servidor que estos implementan. *Ver figura 14*

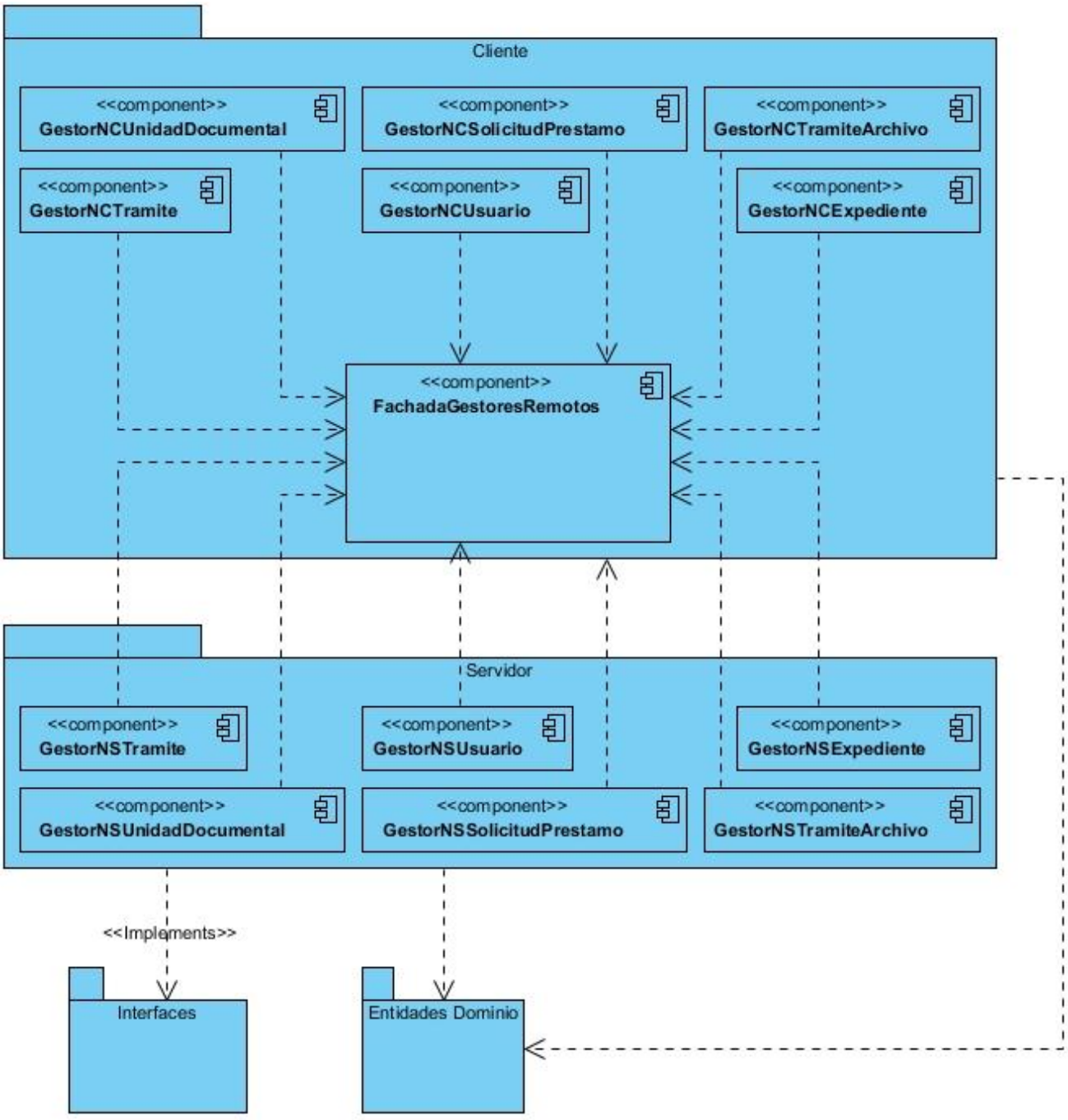


Fig.14: Vista de negocio

La vista de Acceso a Datos muestra el uso de las entidades de dominio, las entidades persistentes y el fichero de consultas por parte de los gestores de Acceso a Datos, además de las interfaces de comunicación con el servidor que estos implementan. *Ver figura 15*

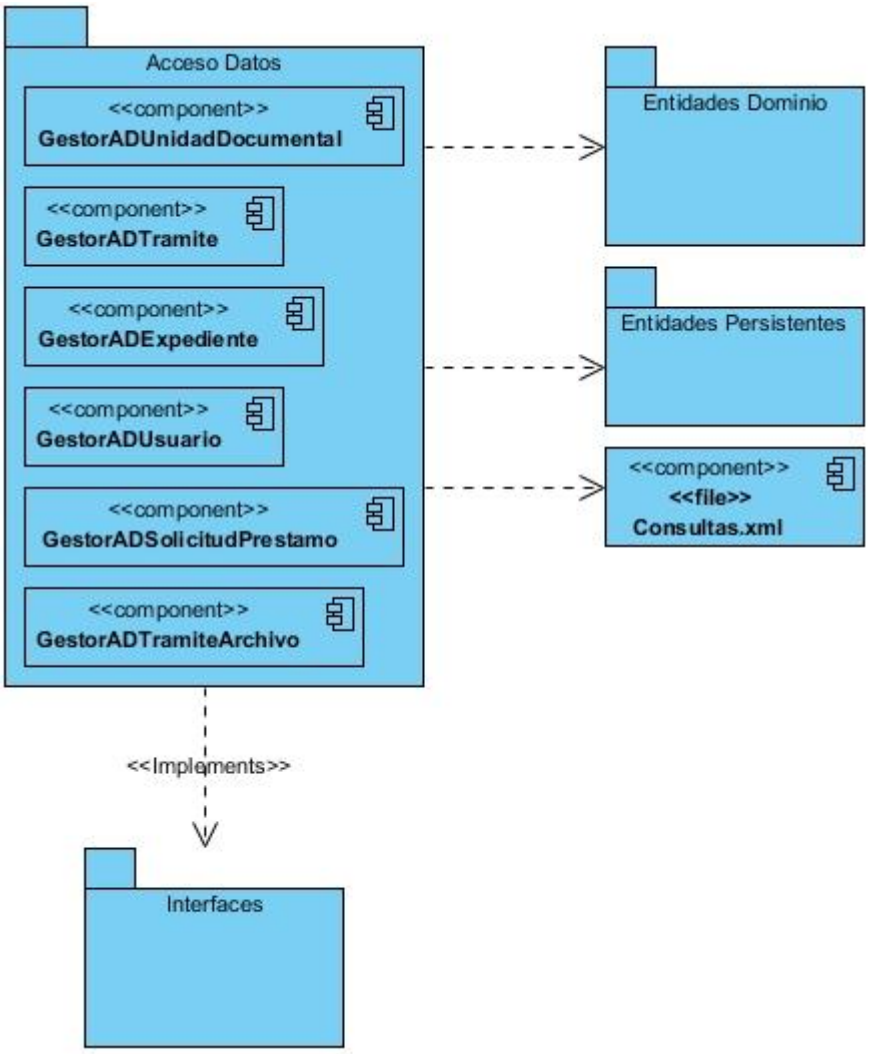


Fig.15: Vista de acceso a datos

2.11. Diagrama de despliegue del sistema

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.[24]

En la División de Antecedentes Penales que constituye el sitio tecnológico donde estará instalado el sistema, habrá un clúster de servidores de aplicaciones y un clúster de servidores de base de datos. En los servidores de aplicaciones estarán alojados los servicios que encapsulan la lógica de negocio de la solución de software y en los servidores de base de datos estará una instancia del subsistema SIGESAP. Además, el Archivo contará con una estación de trabajo correspondiente al funcionario del área donde se estará ejecutando SIGESAP. El propio funcionario de Archivo tendrá la posibilidad de imprimir el registro de entradas y realizar la búsqueda de Expedientes con la ayuda de una impresora y un lector de código de barras respectivamente que tendrá a su disposición. *Ver figura 16*

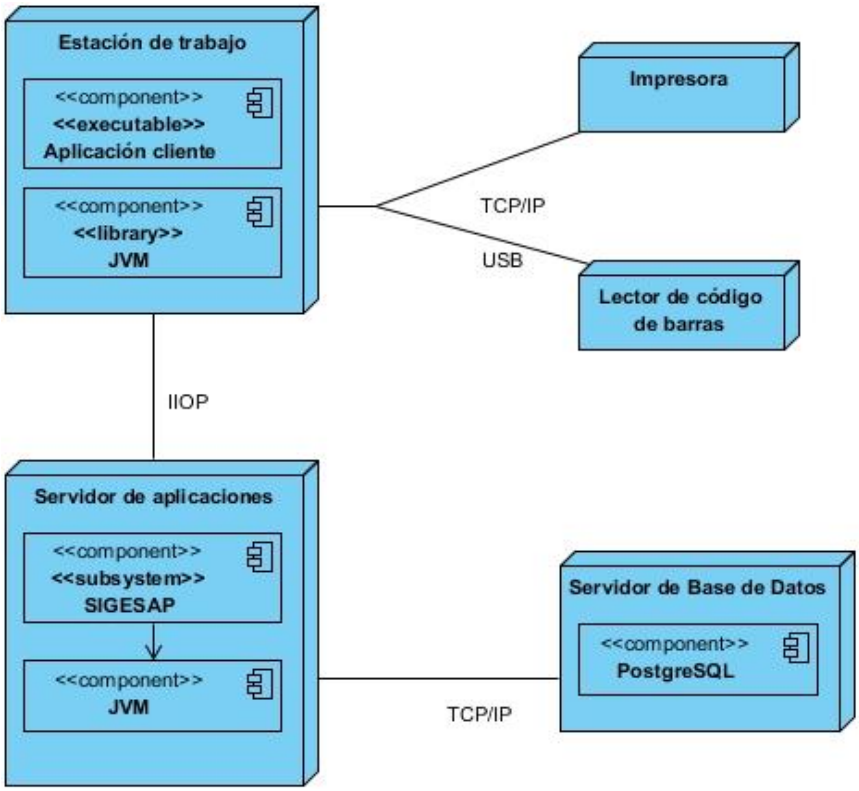


Fig.16: Diagrama de despliegue del módulo Archivo

A continuación se describen cada uno de los nodos que componen el diagrama de despliegue.

Dispositivos.

Impresora: dispositivo que permitirá la impresión de todas las salidas generadas por el sistema durante la ejecución de los procesos requeridos. El dispositivo será compartido a las estaciones de trabajo a través de la red local de la División de Antecedentes Penales.

Lector de código de barras: dispositivo que permitirá leer los códigos de barras que identifica a cada expediente existente en el Archivo. El mismo se comunicará con las estaciones de trabajo a través de un puerto USB de una estación de trabajo destinada a tal fin.

Estación de trabajo: procesador con el que interactuarán los usuarios finales de la solución informática, permite la ejecución de la aplicación cliente. Para que la aplicación cliente se ejecute correctamente las estaciones de trabajo deberán contener además la Máquina Virtual de Java (JVM).

Servidor de aplicaciones: el servidor de aplicaciones se encargará de contener las aplicaciones del lado del servidor del subsistema SIGESAP donde se encuentra el módulo Archivo. Es necesario tener instalado en el mismo el servidor de aplicaciones Glassfish en su versión 3.1 y la JVM.

Servidor de base de datos: el servidor de base de datos contendrá la base de datos donde se almacenará la información que será gestionada por el subsistema SIGESAP.

Requisitos de software y hardware para despliegue de la solución

Para que la solución tenga un resultado favorable es necesario contar con los siguientes requisitos de software y de hardware en los servidores y estaciones de trabajo.

Estaciones de trabajo.

Las configuraciones de software de las estaciones de trabajo deben contar con los siguientes elementos:

- Sistema Operativo: Ubuntu 10.10 o Windows 7.
- Máquina Virtual de Java 1.6, actualización 26.
- Visor de Documentos PDF: Si el sistema operativo es Windows se deberá instalar Acrobat Reader, en caso contrario instalar Evince.
- Herramientas Ofimáticas: OpenOffice 3.0

Las características técnicas de hardware son las siguientes:

- Procesador de doble núcleo a 2.0 GHz.
- 2 GB de RAM, DDR2 667 MHz.
- 160 GB de disco duro, SATA 5400 RPM.
- Adaptador de red Ethernet 1 Gbps.
- Monitor LCD 17''.
- Sistema de Energía Ininterrumpida (UPS) 500 Va

Servidores de la Solución de Software

Los servidores de la solución se dividen en dos grupos, servidores de aplicación y servidores de Bases de Datos. Estos deberán contar con Sistema Operativo Red Hat, para mantener uniformidad con el resto de los sistemas existentes en el MPPRIJ. Ante una alternativa se deberá instalar Debian. Además estos deberán contar con las siguientes aplicaciones que dependen del grupo al que pertenecen.

- Servidores de Aplicación.
  - ✓ Máquina Virtual de Java 1.6, actualización 26.
  - ✓ Servidor de Aplicación: Oracle Glassfish Open Source Server 3.1
  - ✓ IpTable.
- Servidores de Bases de Datos.
  - ✓ Gestor de Bases de Datos: PostgreSQL 8.4.

Las características técnicas mínimas de hardware requeridas son:

- Procesador de doble núcleo de 64 bits a 2.0 GHz. Cuatro procesadores en los servidores de aplicación y dos en los servidores de bases de datos.
- 2GB de RAM por cada instancia de java que correrá en el servidor de aplicaciones.
- 5 TB de espacio para el almacenamiento de la información de la solución particionado en discos duros configurados en RAID.

2.12. Conclusiones del capítulo

Durante este capítulo se expusieron los patrones de diseño, estándares decodificación y pautas de diseños utilizadas, y se realizó una descripción de la arquitectura escogida. Se obtuvo como resultados del diseño y la implementación del sistema los diagramas de clases del diseño, el diagrama de paquetes, el diagrama de componentes, el diagrama de despliegue y el modelo de datos. Finalmente se obtuvo una versión funcional del sistema listo para comenzar la etapa de pruebas.

Capítulo 3: Validación de los Resultados

Introducción

Para validar si el sistema funciona correctamente y cumple con todos los requisitos pactados con el cliente, es necesario realizar una serie de pruebas que lo verifiquen. En el presente capítulo se hace un resumen de las pruebas realizadas para medir la calidad del software, se realizaron las pruebas de Caja Blanca empleando la técnica del Camino Básico y las pruebas de Caja Negra utilizando la técnica de Partición Equivalente a través de los casos de prueba que permitieron su aplicación. Además se muestran los resultados de las pruebas de aceptación realizadas por el cliente.

3.1. Pruebas de Caja Blanca

Una de las ventajas de la metodología RUP es que te permite realizar pruebas durante casi todo el ciclo de desarrollo, teniendo mayor peso en la fase de Transición. Esta es una etapa importante debido a que al software se le realizan distintos tipos de pruebas que van disminuyendo el número de errores detectados y aumentando su calidad. Una de las pruebas realizadas fue Caja Blanca donde se usó la técnica del Camino Básico apoyándose en el marco de trabajo JUnit 4.5 para probar el código implementado.

A continuación se muestra el código probado del método BuscarSolicitudesPorCriterios. Este método se encarga de buscar en la base de datos las solicitudes de préstamos que cumplan con los criterios de entrada, tales como idFuncionario que indica el funcionario que realizó la solicitud, el idEstado que indica el estado (solicitado, confirmado, prestado) en que se encuentra la solicitud de préstamo, el número del Expediente solicitado y el rango de fecha en que fue realizada la solicitud. *Ver figura 17*

```
@Test
public void testBuscarSolicitudesPorCriterios() throws Exception {
    System.out.println("buscarSolicitudesPorCriterios");
    int idEstado = 2;
    String numeroExpedienteSentencia = "AP.01.23.84256";
    Calendar fechaSalidaDesde = Calendar.getInstance();
    Calendar fechaSalidaHasta = Calendar.getInstance();
    double idFuncionario = 142.14;
    int inicial = 1;
    int maximo = 50;
    EJBContainer container = javax.ejb.embeddable.EJBContainer.createEJBContainer();
    GestorADSolicitudPrestamoLocal instance = (GestorADSolicitudPrestamoLocal)
            container.getContext().lookup("java:global/classes/GestorADSolicitudPrestamo");
    List expResult = instance.buscarSolicitudesPorCriterios(idEstado,
            numeroExpedienteSentencia, fechaSalidaDesde, fechaSalidaHasta,
            idFuncionario, inicial, maximo);
    List result = instance.buscarSolicitudesPorCriterios(idEstado,
            numeroExpedienteSentencia, fechaSalidaDesde, fechaSalidaHasta,
            idFuncionario, inicial, maximo);
    assertEquals(expResult, result);
    container.close();
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

Fig.17: Código de prueba del método BuscarSolicitudesPorCriterios de la clase GestorADSolicitudPrestamo

Grafo de flujo

El grafo de flujo es utilizado por la técnica del Camino Básico para determinar la complejidad ciclomática y de esta forma saber la cantidad de casos de pruebas necesarios para realizar la prueba. A continuación se muestra el grafo de flujo correspondiente al código del método BuscarSolicitudesPorCriterios.

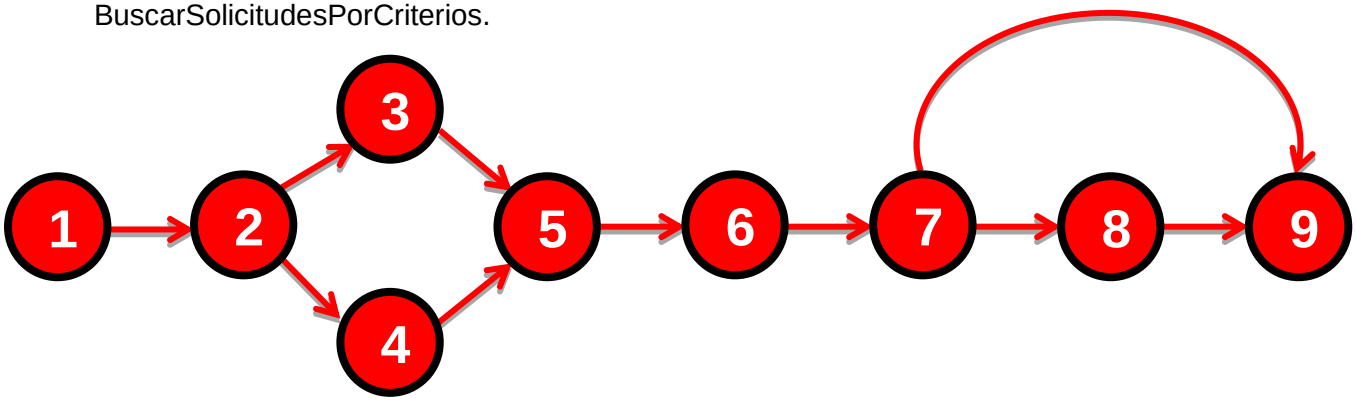
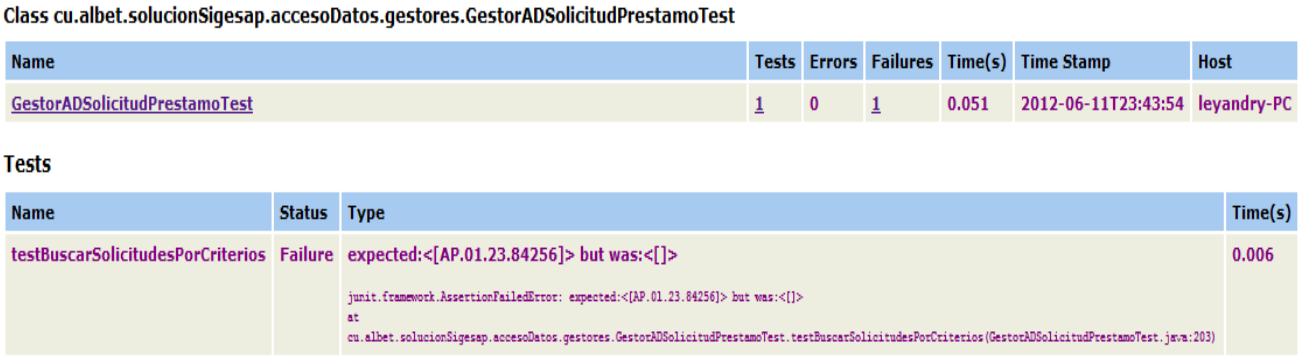


Fig.18: Grafo de flujo del método BuscarSolicitudesPorCriterios

Complejidad ciclomática

V (G) =A – N + 2
V (G) =10 – 9 + 2
V (G) =1 + 2
V (G) =3

Teniendo en cuenta que la complejidad ciclomática dio como resultado que el grafo tiene tres caminos independientes se deben aplicar tres casos de pruebas. En la prueba ejecutada para el primer conjunto de valores se detectó un fallo en el algoritmo, lo cual trajo consigo que el resultado del método BuscarSolicitudesPorCriterios no fuese el esperado como se puede apreciar en el reporte generado para esta prueba que se muestra en la *figura.19*



Class cu.albet.solucionSigesap.accesoDatos.gestores.GestorADSolicitudPrestamoTest

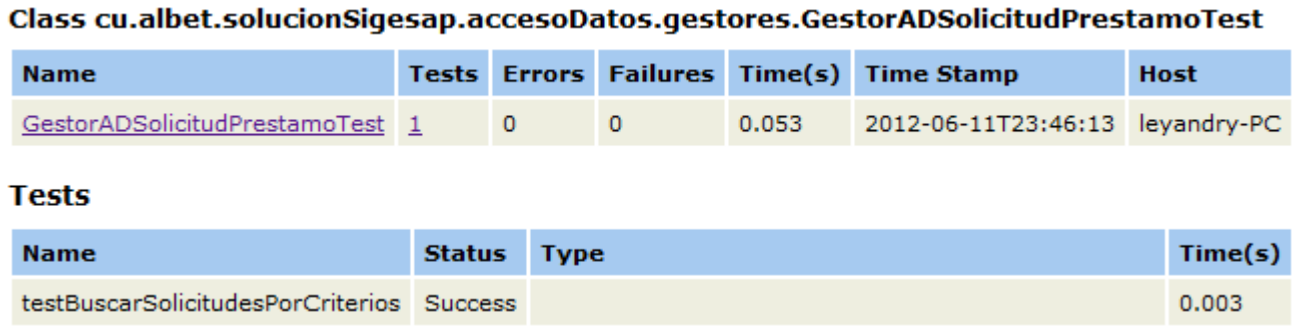
Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
GestorADSolicitudPrestamoTest	1	0	1	0.051	2012-06-11T23:43:54	leyandry-PC

Tests

Name	Status	Type	Time(s)
testBuscarSolicitudesPorCriterios	Failure	expected:<[AP.01.23.84256]> but was:<[]> junit.framework.AssertionFailedError: expected:<[AP.01.23.84256]> but was:<[]> at cu.albet.solucionSigesap.accesoDatos.gestores.GestorADSolicitudPrestamoTest.testBuscarSolicitudesPorCriterios(GestorADSolicitudPrestamoTest.java:203)	0.006

Fig.19: Reporte generado para el primer caso de prueba aplicado al método BuscarSolicitudesPorCriterios

Se resolvió el problema detectado y se realizó nuevamente la prueba para los mismos valores arrojando un resultado satisfactorio como se puede apreciar en el nuevo reporte generado.



Class cu.albet.solucionSigesap.accesoDatos.gestores.GestorADSolicitudPrestamoTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
GestorADSolicitudPrestamoTest	1	0	0	0.053	2012-06-11T23:46:13	leyandry-PC

Tests

Name	Status	Type	Time(s)
testBuscarSolicitudesPorCriterios	Success		0.003

Fig.20: Reporte generado para el primer caso de prueba aplicado al método BuscarSolicitudesPorCriterios

Para ver el resto de los reportes generados consultar el *anexo 7.*

Las pruebas fueron realizadas a las funcionalidades de mayor complejidad ciclomática y significativas para el cumplimiento de los requisitos del módulo Archivo, donde se obtuvieron los siguientes resultados:

Iteraciones	Código correcto (%)	Código fallido (%)
1	85	15
2	100	0

Tabla 1: Resultados de las pruebas de Caja Blanca

3.2 Pruebas de Caja Negra

Con el objetivo de probar las interfaces del sistema se realizaron pruebas de Caja Negra utilizando la técnica de Partición Equivalente aplicada a través de los casos de pruebas diseñados para demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Estas pruebas fueron realizadas por un equipo del Centro Nacional de Calidad del Software (Calisoft). A continuación se muestra el caso de prueba del caso de uso Gestionar Préstamos.

Descripción general

El caso de uso se inicia cuando el funcionario de Archivo desea gestionar las solicitudes de préstamo de Expedientes.

Condiciones de ejecución

El sistema debe estar instalado y ejecutándose correctamente.

El usuario debe estar autenticado con los permisos necesarios.

Secciones

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Buscar	EC 1.1: Introducir los datos correspondientes a la solicitud de préstamo de forma correcta.	Se realiza la búsqueda de solicitudes correctamente.
	EC 1.2: Dejar todos los campos vacíos.	Se muestra un mensaje indicando que no puede dejar todos los campos vacíos.

	EC 1.3: Introducir los datos correspondientes a la solicitud de préstamo de forma incorrecta.	No se realiza la búsqueda de solicitudes correctamente.
	EC 1.4: Introducir datos incompletos.	No se realiza la búsqueda de solicitudes correctamente.
SC 2: Prestar	EC 2.1: Prestar un expediente.	Al seleccionar una solicitud de la lista, en estado de "Solicitado" o "Confirmado", permite cambiar su estado a "Prestado".
SC 3: Devolver	EC 3.1: Devolver un expediente.	Al seleccionar una solicitud de la lista, en estado de "Prestado", permite devolver el expediente involucrado eliminando dicha solicitud.
SC 4: Confirmar	EC 4.1: Confirmar una solicitud.	Al seleccionar una solicitud en estado de "Solicitada", permite cambiar su estado a "Confirmada"
SC 5: Cancelar	EC 5.1 Cancelar el flujo.	El sistema finaliza el flujo que estaba iniciado en ese momento.
SC 6: Búsqueda avanzada	EC 6.1: Búsqueda avanzada por funcionario.	Permite buscar un funcionario por varios criterios de búsqueda. **Consultar el caso de prueba Buscar Funcionario**.
	EC 6.2: Cancelar.	Regresa a la interfaz Gestionar préstamos.
	EC 6.3: Anterior	El sistema regresa a interfaz Gestionar préstamos.
	EC 6.4: Buscar solicitudes	Permite buscar las solicitudes realizadas por el funcionario seleccionado.
SC 7: Nueva solicitud	EC 7.1: Buscar funcionario solicitante.	Permite buscar un funcionario por varios criterios de búsqueda. **Consultar el caso de prueba Buscar Funcionario.**
	EC 7.2: Cancelar	Regresa a la interfaz Gestionar préstamos.
	EC 7.3: Anterior	Regresa a la interfaz Gestionar préstamos.
	EC 7.4: Siguiente	Se muestra la interfaz "Registrar solitudes de préstamos de expedientes". **Consultar caso de prueba Realizar solicitudes.**
SC 8: Ver reporte	EC 8.1: Ver reporte	Se muestra un visor de pdf con la información tabulada brindando la opción de imprimir.

Descripción de variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Número de expediente	Campo de texto	Si	Cadena de caracteres.
2	Fecha de solicitud [desde]	Calendario	Si	Se selecciona la fecha de inicio de la solicitud del calendario.
3	Fecha de solicitud [hasta]	Calendario	Si	Se selecciona la fecha de fin de la solicitud del calendario.
4	Estado de la solicitud	Lista desplegable	Si	Se selecciona de la lista el estado deseado.

Matriz de datos

SC 1: Buscar

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 1.1	Introducir los datos correspondientes a la solicitud de préstamo de forma correcta.	V AP.01.01.382	V 14/05/1012	V 15/05/1012	V Prestado	El sistema realiza la búsqueda correctamente.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Clic en la opción “Buscar”.

EC 1.2	Dejar todos los campos vacíos.	(vacío)	(vacío)	(vacío)	(vacío)	El sistema muestra un mensaje de aviso: "Debe introducir al menos un criterio para efectuar la búsqueda. Por favor verifique".	1. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar Préstamos". 3. Clic en la opción "Buscar".
EC 1.3	Introducir los datos correspondientes a la solicitud de préstamo de forma incorrecta.	I AX.01.01.382	V 14/15/1012	V 15/15/1012	V Prestado	El sistema no muestra ningún resultado en la búsqueda.	1. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar Préstamos". 3. Clic en la opción "Buscar".
		V AP.01.01.382	I 14/05/2012	V 15/04/2012	V Prestado	El sistema muestra un mensaje de aviso: "El rango de fechas para la búsqueda no es válido".	1. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar Préstamos". 3. Clic en la opción "Buscar".
		V AP.01.01.382	V 14/05/2012	I 15/04/2011	V Prestado	El sistema muestra un mensaje de aviso: "El rango de fechas para la búsqueda no es válido".	1. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar Préstamos". 3. Clic en la opción "Buscar".

EC 1.4:	Introducir los datos incompletos .	I AP.01.01	V 14/15/1012	V 15/15/1012	V Solicitado	El sistema no muestra ningún resultado en la búsqueda.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Clic en la opción “Buscar”.

SC 2: Prestar expediente

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 2.1	Prestar un expediente	N.A	N.A	N.A	N.A	El sistema muestra el mensaje de información: “Préstamo realizado satisfactoriamente” y cambia el estado de la solicitud a “Prestado”.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Introducir los criterios de búsqueda. 4. Seleccionar la solicitud deseada. 5. Clic en la opción “Prestar”.

SC 3: Devolver expediente

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 3.1	Devolver un expediente	N.A	N.A	N.A	N.A	Muestra el mensaje de confirmación: ¿Está seguro que desea devolver el expediente? Luego muestra el mensaje de información: El expediente ha sido devuelto satisfactoriamente.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Introducir los criterios de búsqueda. 4. Seleccionar la solicitud deseada. 5. Clic en la opción “Devolver”.

SC 4: Confirmar solicitud

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 4.1	Confirmar una solicitud	N.A	N.A	N.A	N.A	El sistema muestra el mensaje de aviso: "¿Está seguro que desea confirmar la solicitud en curso?" Luego cambia el estado de la solicitud a Confirmado.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Introducir los criterios de búsqueda. 4. Seleccionar la solicitud deseada. 5. Clic en la opción “Confirmar”.

SC 5: Búsqueda avanzada

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 5.1	Búsqueda avanzada	N.A	N.A	N.A	N.A	El sistema muestra la interfaz “Buscar funcionario” y permite realizar la búsqueda por criterios.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar Préstamos”. 3. Clic en la opción “Búsqueda avanzada”.

EC 5.2	Terminar	N.A	N.A	N.A	N.A	El sistema regresa a la interfaz “Gestionar préstamos”.	1. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar préstamos”. 3. Clic en la opción “Búsqueda avanzada”. 4. Se muestra la interfaz “Buscar funcionario” con los criterios para realizar la búsqueda. 5. Clic en la opción “Terminar”.
EC 5.3	Anterior	N.A	N.A	N.A	N.A	El sistema regresa a la interfaz “Gestionar préstamos”.	. Iniciar sección de “Archivo”. 2. Clic en la opción del menú “Gestionar préstamos”. 3. Clic en la opción “Búsqueda avanzada”. 4. Se muestra la interfaz “Buscar funcionario. 5. Clic en la opción “Anterior”.

EC 5.4	Buscar solicitudes	N.A	N.A	N.A	N.A	El sistema realiza la búsqueda de las solicitudes asociadas al funcionario seleccionado y muestra los resultados en la interfaz "Gestionar préstamos".	. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar préstamos". 3. Clic en la opción "Búsqueda avanzada". 4. Se selecciona él funcionario. 5. Clic en la opción "Buscar solicitudes".

SC 6: Ver reporte

ID del escenario	Escenario	Número de Expediente	Fecha desde	Fecha hasta	Estado de la solicitud	Respuesta del sistema	Flujo Central
EC 7.1	Ver reporte	N.A	N.A	N.A	N.A	El sistema genera un reporte en visor de pdf de las solicitudes mostradas en la interfaz.	1. Iniciar sección de "Archivo". 2. Clic en la opción del menú "Gestionar Préstamos". 3. Clic en la opción "Ver reporte".

Listado de funcionalidades probadas del módulo Archivo.

Caso de uso	Funcionalidades probadas
Gestionar préstamos.	Buscar solicitudes de préstamos
	Búsqueda avanzada por funcionario
	Realizar nueva solicitud

	Prestar expediente
	Confirmar solicitud
	Devolver expediente
Confirmar recepción de expedientes	Buscar trámites
	Confirmar recepción de expedientes
Solicitar expediente en tramite	Buscar expedientes de sentencia
	Eliminar expediente de la lista.
	Registrar solicitud
Ver solicitudes de funcionario	Actualizar lista solicitudes
	Cancelar solicitud
Realizar solicitud de préstamo manual	Eliminar expediente de la lista.
	Motivo del préstamo.
	Búsqueda avanzada.
	Registrar solicitud

Primera iteración

Se probaron todos los casos de uso del módulo verificando que el sistema respondía a cada evento de la manera esperada. Fueron detectadas las siguientes no conformidades:

- ✓ No permitir la entrada de caracteres extraños en el campo “Motivo” para los préstamos de expedientes.
- ✓ Permitir el ingreso de más de 140 caracteres en el campo “Motivo” para los préstamos de Expedientes.
- ✓ Mostrar el motivo del préstamo en la tabla de Solicitudes de Préstamos.
- ✓ Error ortográfico en el mensaje de confirmación de Entrada a los Expedientes.
- ✓ No concuerda la descripción de la ayuda con la información que muestra la interfaz del caso

de uso Confirmar Encuadernación, en la pantalla de ver unidades documentales.

- ✓ Al tratar de almacenar un Expediente sin fecha de asiento o sin número antiguo (no son obligatorios) el sistema da un error y muestra una excepción no controlada.
- ✓ El linkbutton Motivo de préstamos permanece activo aun cuando no se ha seleccionado ninguna solicitud.
- ✓ El visor de archivos en formato pdf muestra una plantilla vacía aun cuando no se ha seleccionado ninguna unidad documental.
- ✓ Error ortográfico en nombre de la tabla de solicitudes en curso.
- ✓ El filtro de búsqueda de trámites permite la entrada de un rango de fechas inválido.

Segunda iteración

En la segunda iteración se detectaron 4 no conformidades la cuales fueron resueltas debidamente.

Prueba de regresión

El equipo de Calisoft realizó la prueba de regresión donde evidenciaron que todas las no conformidades habían sido resueltas y de esta forma queda liberado el módulo Archivo, como constancia se tiene el acta de liberación que fue anexada al Expediente del proyecto.

La siguiente tabla muestra los resultados de las pruebas.

Iteraciones	No conformidades detectadas (%)	Errores de interfaces (%)	Errores de excepciones no controladas (%)	Errores de acceso a datos (%)
1	10	50	20	30
2	4	50	0	50
Regresión	0	0	0	0

Tabla 2: Resultados de las pruebas de Caja Negra

3.3. Niveles de pruebas aplicados

Con la utilización del método de prueba Caja Negra se puede lograr la validez de la solución en cuanto a lo especificado en los casos de uso pero no demuestran completamente la ausencia de errores, ni garantiza el correcto funcionamiento de la solución con el sistema. Por estas razones se decide realizar diferentes niveles de pruebas a la solución.

3.3.1. Pruebas de Integración

Las pruebas de integración como su nombre lo indica tienen el objetivo de engranar todos los componentes del sistema y corroborar que ellos no solo funcionan de manera independiente.

El módulo Archivo tiene una estrecha relación con el módulo Inscripción del que recibe los trámites listos para darle entrada, así como los que vienen del Centro de Digitalización por medio de réplica de datos. Se ha dado entrada a los trámites provenientes de ambas áreas de manera satisfactoria. Otros elementos que se tuvieron en cuenta fueron la interacción con los dispositivos de impresión y el componente que permite visualizar los documentos escaneados. Además la salida más importante que tienen los Expedientes asentados en el archivo son los antecedentes penales registrados y a diario se emiten certificaciones a entes autorizados que han sido generadas de manera completa, con toda la información que contienen los datos procesales.

Por otra parte, el módulo de Integración como se explica en el epígrafe 2.6 es un módulo que agrupa las funcionalidades comunes que fueron necesarias implementar durante el desarrollo. El módulo Archivo cuenta con dos de estas funcionalidades: Solicitar Expediente a Archivo y Ver Solicitudes de Funcionario, las cuales permiten a los funcionarios autorizados realizar la solicitud de un Expediente desde un trámite de Inscripción y ver el estado de la misma satisfactoriamente. A su vez el módulo Integración presenta la funcionalidad Buscar Funcionario la cual permite al funcionario de Archivo realizar la búsqueda del funcionario solicitante de forma correcta.

Las pruebas se van realizando de manera incremental durante el desarrollo pero hay elementos que solo se podían probar en el ambiente real como es el caso de la réplica de trámites desde Centro de Digitalización.

3.3.2. Pruebas de Aceptación

Las pruebas de Aceptación se desarrollaron en un escenario controlado verificándose los elementos solicitados por los funcionarios en las diferentes etapas.

El proceso de pruebas se desarrolló en un ambiente de trabajo colaborativo evaluando las respuestas del sistema desde la perspectiva del cliente. Intervinieron en el proceso 14 funcionarios del Ministerio del Poder Popular para Relaciones Interiores y Justicia, siendo de estos 4 asesores del ministerio y 10 de la División de Antecedentes Penales de dicho Ministerio.[25]

Las pruebas del módulo Archivo fueron realizadas en presencia de 3 de estos funcionarios de la DAP, en la primera iteración se logró probar todos los casos de uso del módulo donde no se detectaron no conformidades, solo se realizaron las siguientes peticiones de cambios:

- Cambiar en toda la aplicación el nombre Reo por Privado de libertad (debido a una comunicación realizada por el presidente Chávez).
- Cambiar el nombre de la funcionalidad Confirmar encuadernación por Confirmar recepción de Expedientes.
- Permitir que el funcionario de Archivo tenga acceso al reporte trazas de trámite (para que en caso de error el funcionario de Archivo pueda regresarle el expediente al funcionario que trabajó con él en el área anterior o en caso de dudas saber a quién dirigirse).
- Permitir ver las solicitudes de préstamos sin tener que oprimir el botón Buscar la primera vez (al seleccionar en el menú la opción Ver Solicitudes el sistema muestre las solicitudes en curso, luego si el funcionario desea filtra según las posibilidades que brinda la interfaz).
- Permitir la impresión del reporte de Entrada.

Fueron implementadas las peticiones de cambios, las cuales se probaron en la segunda iteración y prueba de regresión realizadas en una misma sesión de trabajo. Al final de la etapa el cliente firmó el Acta de Aceptación. [26]

3.4. Conclusiones del capítulo

En este capítulo se realizó la validación de la propuesta de solución a través de distintos tipos de pruebas, entre las que se encuentran las pruebas de Caja Blanca usando la técnica de Camino Básico, Caja Negra usando la Partición Equivalente y las pruebas de Aceptación, evidenciando la aprobación y la satisfacción por parte del cliente con la firma del Acta de Aceptación. Además, se mostraron resultados que demuestran la productividad que ha tenido el Archivo en la prueba piloto que se lleva a cabo en la DAP y como está contribuyendo a mejorar su funcionamiento.

Conclusiones

Al finalizar el presente trabajo de diploma se logró dar cumplimiento a los objetivos planteados al inicio del mismo y de esta forma se arriba a las siguientes conclusiones:

1. Se realizó el diseño metodológico de la investigación que permitió identificar el problema, su campo de acción, los objetivos y las tareas necesarias para darle solución.
2. Se realizó el marco teórico de la investigación, lo que permitió estudiar y seleccionar las herramientas y tecnologías utilizadas durante el desarrollo. Las herramientas y tecnologías seleccionadas fueron:
   - ✓ Metodología de desarrollo: Proceso Unificado de Desarrollo (RUP)
   - ✓ Herramienta de modelado: Visual Paradigm 3.4
   - ✓ Lenguaje de modelado: Proceso Unificado de Modelado (UML) 2.0
   - ✓ Lenguaje de programación: Java 1.6
   - ✓ Entorno de desarrollo integrado: NetBeans 7.0.1
   - ✓ Servidor de aplicaciones: Open Source Glassfish Server 3.1
   - ✓ Sistema gestor de base de datos: PostgreSQL 8.4
   - ✓ Herramienta de pruebas: JUnit 4.5
3. Se obtuvo el Modelo de Diseño de la solución a partir de la especificación de requisitos y con la ayuda del Proceso Unificado de Desarrollo (RUP).
4. Se obtuvieron los diagramas de componentes y despliegue del sistema, correspondientes a la etapa de Implementación.
5. Se realizaron pruebas de Caja Blanca, de Caja Negra y de Aceptación para validar la propuesta de solución con resultados satisfactorios demostrando así el cumplimiento de los requisitos pactados con el cliente.
6. Se obtuvo un producto probado y listo para ejercer su función en la División de Antecedentes Penales de Venezuela.
7. El cliente firmó el Acta de Aceptación como muestra del funcionamiento apropiado del subsistema SIGESAP.

Recomendaciones

Teniendo en cuenta los resultados obtenidos con la realización del trabajo de diploma se recomienda:

1. Aplicar las experiencias y conocimientos obtenidos en el desarrollo de aplicaciones similares.
2. Implementar los nuevos reportes que sean requeridos por los funcionarios de Archivo.
3. Utilizar esta propuesta de arquitectura, probada y con resultados positivos en otros proyectos en la universidad con características similares.

Referencias Bibliográficas

[1] **María del Pilar García, Frida Ortiz**, «Métodos Científicos de Investigación - EcuRed», 2005. [En línea]. Disponible: http://www.ecured.cu/index.php/M%C3%A9todos_Cient%C3%ADficos_de_Investigaci%C3%B3n#Etapas_del_m.C3.A9todo_cient.C3.ADfico_de_investigaci.C3.B3n. [Accedido: 08-jun-2012].

[2] **«Documento - EcuRed»**. [En línea]. Disponible: http://www.ecured.cu/index.php/Documento. [Accedido: 09-jun-2012].

[3] **Irima Campillo Torres**, «Sistema de Gestión Integral de archivo para empresas de la construcción del territorio de Camagüey», Tesis Doctoral, Universidad de Granada, Universidad de la Habana, Granda, 2010.

[4] **«OdiloTID. OdiloA3W»**. [En línea]. Disponible: http://www.odilotid.es/general/odiloa3w.htm. [Accedido: 12-jun-2012].

[5] **«Uml»**, 2011. [En línea]. Disponible: http://EcuRed.cu. [Accedido: 28-may-2012].

[6] **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides**, *GoF Design Patterns with examples using Java and UML2*. Autoedición, 2008.

[7] **«Presentación: Patrones GRASP»**. [En línea]. Disponible: http://presentacion-ingenieria-sistemas.blogspot.com/p/patrones-grasp.html. [Accedido: 10-jun-2012].

[8] **«Características de Java»**, *Características de Java*, sep-2009. [En línea]. Disponible: http://sheyla88.blogspot.es/. [Accedido: 20-feb-2012].

[9] **«Arquitectura de Software v 1.0»**. AP-SW-DE-012, 07-dic-2011.

[10] **Rubén Gómez López**, «Pruebas de software con Junit». 20-ene-2009.

[11] **«Cliente - Servidor»**. [En línea]. Disponible: http://www.ultrasist.com.mx/index.php?option=com_content&view=article&id=21%3Acliente-servidor&catid=15%3Aarquitecturas&Itemid=13. [Accedido: 13-abr-2012].

[12] **E. R. Expósito Álvarez, A Y**., «Diseño e implementación de las capas de Negocio y Acceso a datos de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP.» 2008.

[13] **Julio Casal Terrero**, «Desarrollo de Software basado en Componentes». [En línea]. Disponible: http://msdn.microsoft.com/es-es/library/bb972268.aspx. [Accedido: 13-abr-2012].

[14] **«Prueba de RUP - EcuRed»**. [En línea]. Disponible: http://www.ecured.cu/index.php/Prueba_de_RUP. [Accedido: 10-jun-2012].

[15] **«Pruebas de caja blanca - EcuRed»**. [En línea]. Disponible: http://www.ecured.cu/index.php/Pruebas_de_caja_blanca. [Accedido: 10-jun-2012].

[16] **Roger S. Pressman**, *Ingeniería del Software. Un enfoque práctico.*, Quinta ed. 1997.

[17] **José Luis Aristegui O**., «Los casos de prueba en la prueba de software». 12-abr-2010.

[18] **«Especificación de Requisitos de Software v 1.0»**.AP-SW-DE-001, ALBET, 06-ago-2010.

[19] **Roger S. Pressman,** *Ingeniería del Software. Un enfoque práctico.*, Sexta ed. 2005.

[20] **«Modelo de Diseño Archivo v1.0»**. AP-SW-DE-007, ALBET, 05-dic-2012.

[21] **«Estándares de Codificación v1.0»**. 27-nov-2011.

[22] **«Pautas Generales para el Diseño de Interfaces v1.0»**. 22-nov-2011.

[23] **«Sparx Systems - Tutorial UML 2 - Diagrama de Paquete»**, 2007. [En línea]. Disponible: http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html. [Accedido: 20-may-2012].

[24] **«Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue»**, 2007. [En línea]. Disponible: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html. [Accedido: 20-may-2012].

[25] **«Informe Resumen de las Pruebas de Aceptación»**. CAL_PA-AP_012**,** ALBET.

[26] **«Acta de Aceptación».** CAL_PA-AP_011**,** ALBET, 25-nov-2011.

Bibliografía

1. [En línea] http://netbeans.org/community/releases/71/.

2. [En línea] http://www.postgresql.org/about/.

3. [En línea] http://www-01.ibm.com/software/awdtools/rup/.

4. [En línea] http://glassfish.java.net/.

5. [En línea] http://www.visual-paradigm.com/.

6. [En línea] http://www.uml.org/.

7. [En línea] http://courseware.ikor.org/java.

8. [En línea] http://www.slideshare.net/maxmouse/java-persistence-api-jpa.

9. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Lenguaje Unificado de Modelado.* s.l. : Addison-Wesley, 1999. 84-7829-028-1.

10. **Keith, Mike y Schincariol, Merrick.** *Pro JPA2 Mastering the Java Persistence API.* 2009. ISBN:978-1-4302-1956-9.

Glosario de términos

1. **Digitalización de documentos**

   La digitalización de documentos permite su consulta on-line y facilita el acceso a la información a través de diferentes dispositivos electrónicos. Pero además, asegura la conservación de muchos de los fondos más valiosos y antiguos de las bibliotecas y archivos. El paso del tiempo o una mala conservación pueden deteriorar estos documentos, de modo que muchos centros documentales se ven obligados a restringir el acceso público a determinados ejemplares.

2. **Antecedentes Penales**

   Antecedentes penales son aquellas anotaciones que se realizan en un registro correspondiente, dependiente del Ministerio del Interior, como son los registros que lleva el poder judicial de una región en particular, de las condenas impuestas a los individuos como consecuencia de la comisión de algún delito.

3. **Implementación**

   Proceso por el cual se escribe (en un lenguaje de programación), se prueba, se depura y se mantiene el código fuente de un programa informático.

4. **Sistema Gestor de Base de Datos**

   Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos asegurando su integridad, confidencialidad y seguridad.

5. **Base de datos**

   Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

6. **Marco de trabajo**

   Es un concepto sumamente genérico, se refiere a un "ambiente de trabajo y ejecución" utilizado para desarrollar aplicaciones. En general, los marcos de trabajo son soluciones completas que contemplan herramientas y motores de ejecución de apoyo a la construcción de un sistema.

7. **Módulo**

   Es una parte del sistema que se instala y funciona por separado, entrelazándose con otros módulos con los que intercambia información.

8. **Nomencladores**

   Tabla de valores definidos por el usuario que luego no pueden ser modificados.

9. **Plataforma**

   Entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones.

10. **Software**

    Es el conjunto de programas informáticos, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

11. **Arquitectura de software**

    Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático.

12. **Objeto**

    Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

13. Clúster de servidores

Un clúster está formado por dos o más servidores independientes pero interconectados que funcionan como uno solo.

14. RAID (Redundant Array of Independent Disks)

En informática, el acrónimo RAID (del inglés, Conjunto Redundante de Discos Independientes) hace referencia a un sistema de almacenamiento que usa múltiples discos duros entre los que se distribuyen o replican los datos. Un solo disco duro contiene la información dentro de sí una sola vez. El uso de múltiples discos duros con información duplicada y/o distribuida posee ventajas en su seguridad, mayor tolerancia a fallos, mejor rendimiento y mayor capacidad. Estas ventajas dependen del tipo de configuración RAID que se utilice.

15. Banda de los Cuatros (Gang of Four)

Es el nombre con el que se conoce comúnmente a los autores del libro “Patrones de Diseño con ejemplos usando Java”.

16. Dublin Core

Es un sistema cuya función es describir a través de metadatos, objetos digitales como audio, texto, video e, incluso, imágenes que se encuentran en Internet. Así como una ficha bibliográfica es útil para la búsqueda de información en una biblioteca, los campos o metadatos resguardados en una base de datos describen al objeto digital depositado en algún medio de almacenamiento, lo que permite un acceso al objeto por medio de un campo particular, como es la dirección URL (Universal Resource Locator).

Anexos

Anexo 1: Diagramas de clases del diseño.

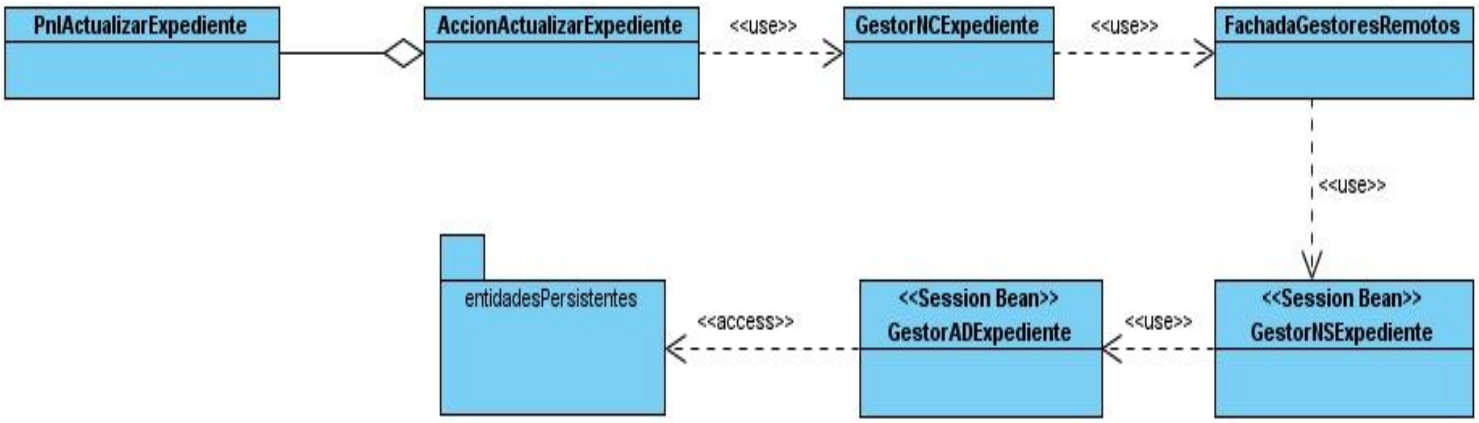


Fig.32: Diagrama de clase del caso de uso Actualizar Expediente

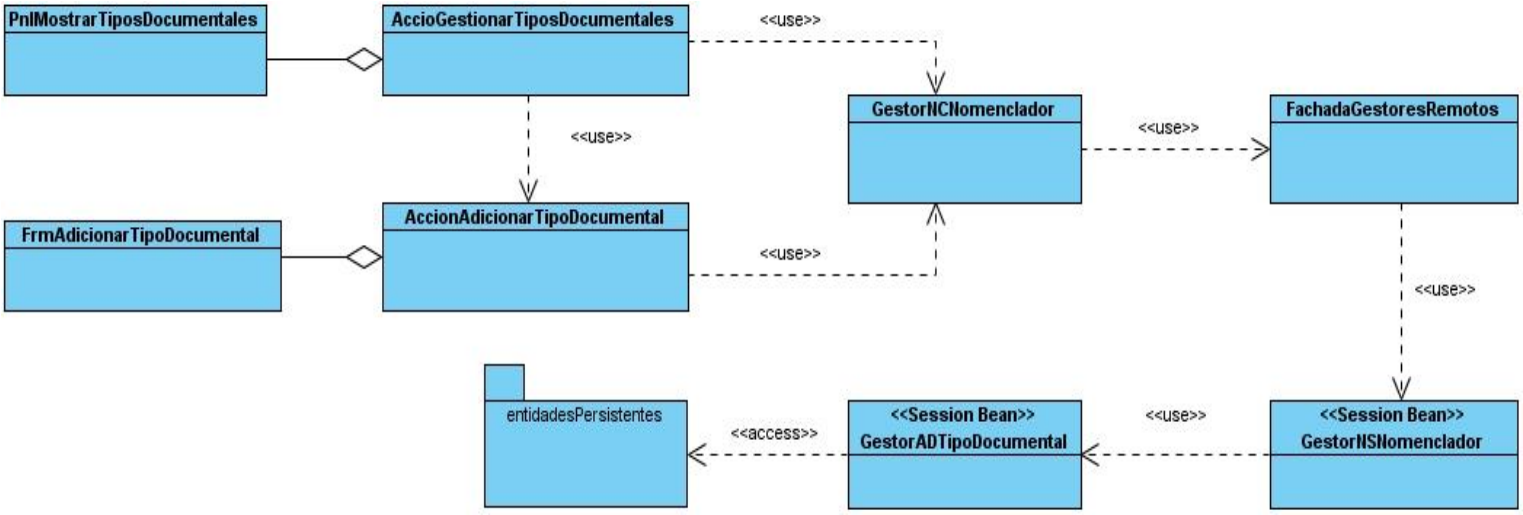


Fig.21: Diagrama de clase del caso de uso Gestionar Tipos Documentales

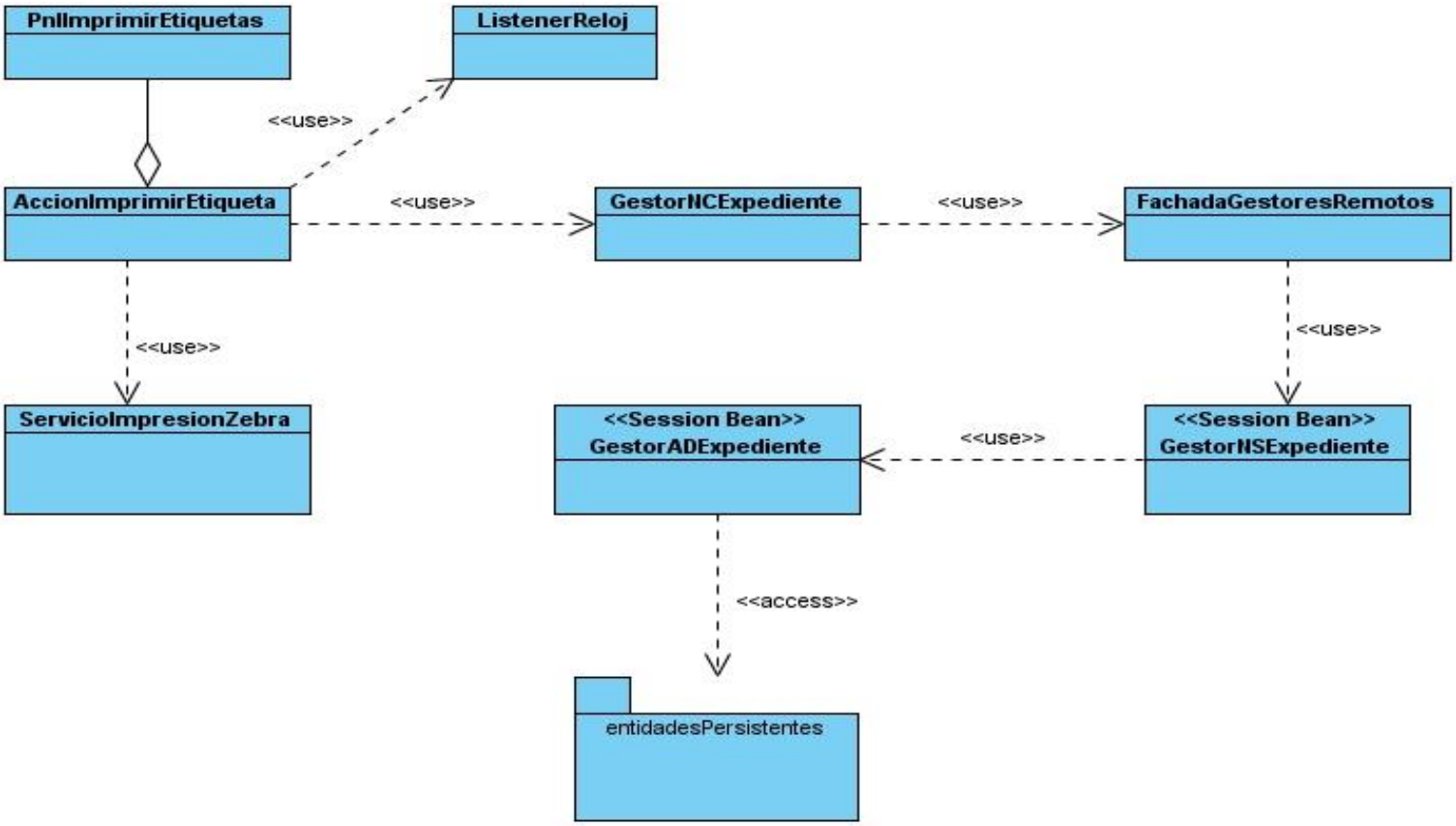


Fig.22: Diagrama de clase del caso de uso Imprimir Etiquetas

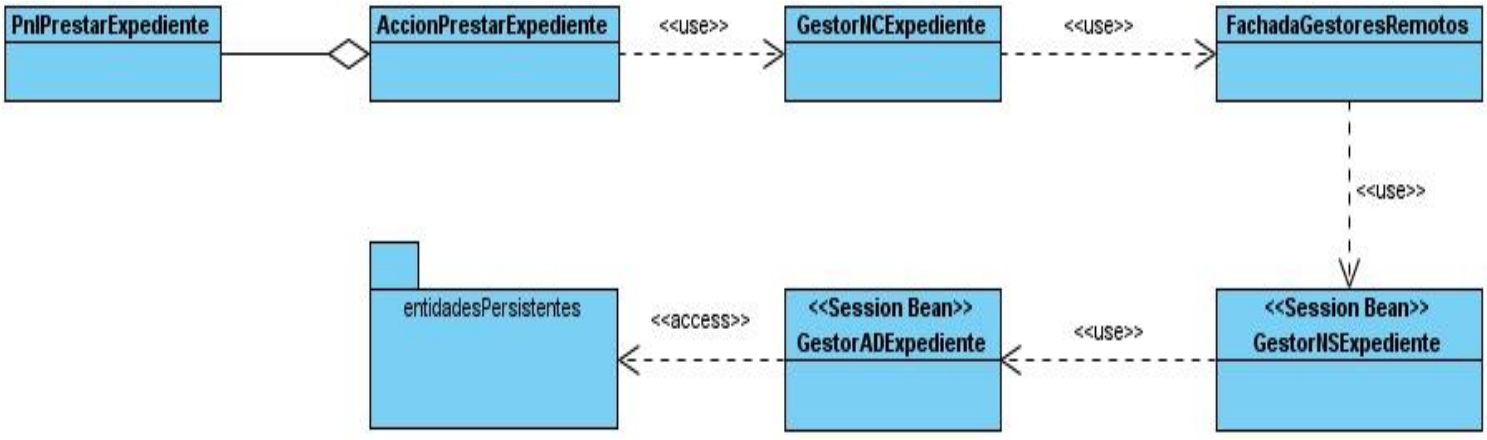


Fig.23: Diagrama de clase del caso de uso Prestar Expediente

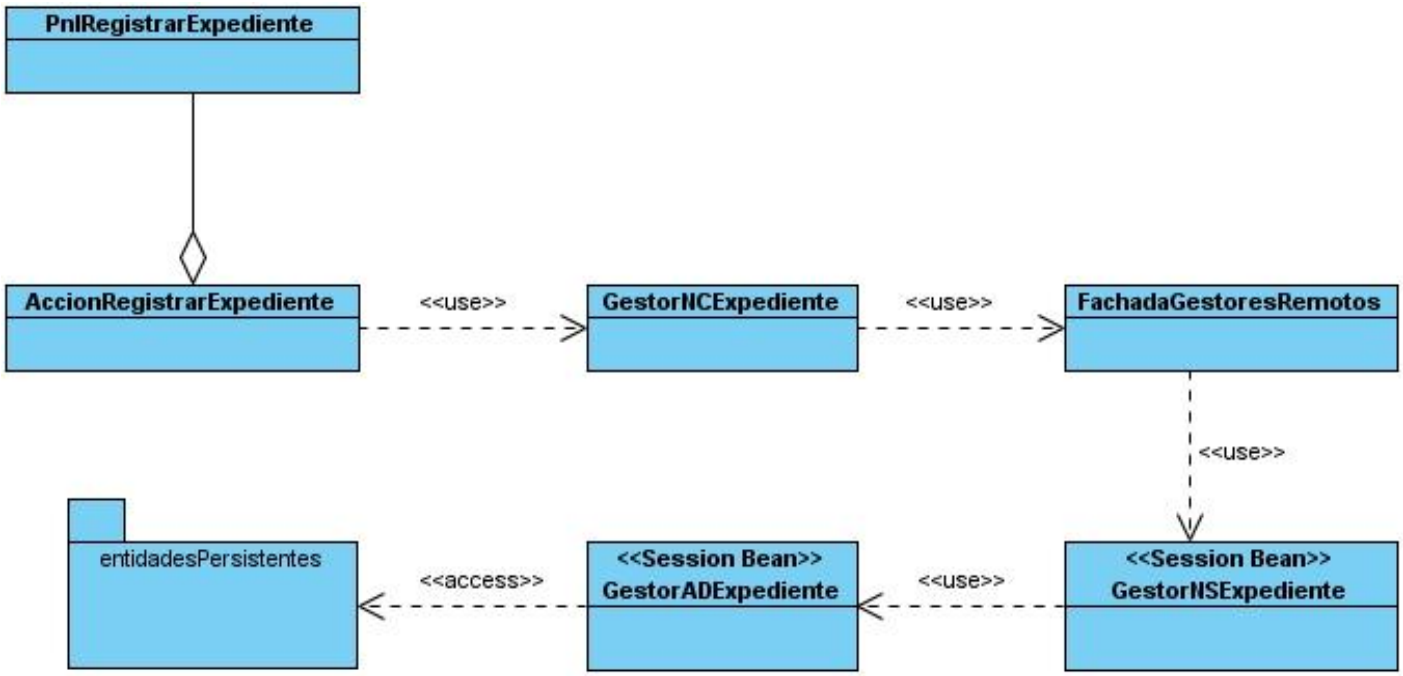


Fig.24: Diagrama de clase del caso de uso Registrar Expediente

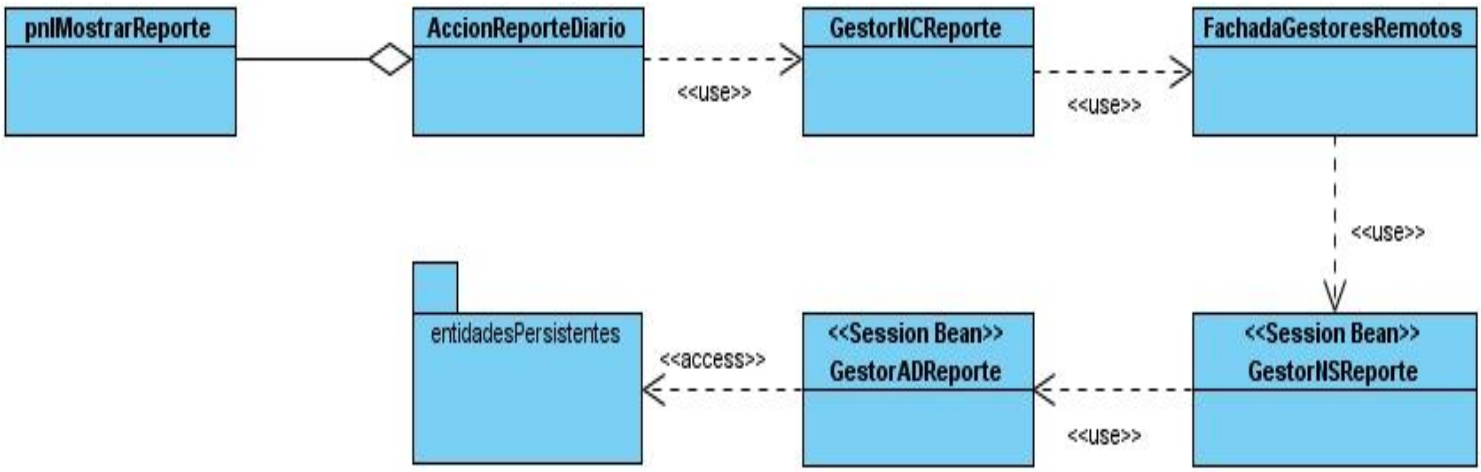


Fig.25: Diagrama de clase del caso de uso Reporte Diario

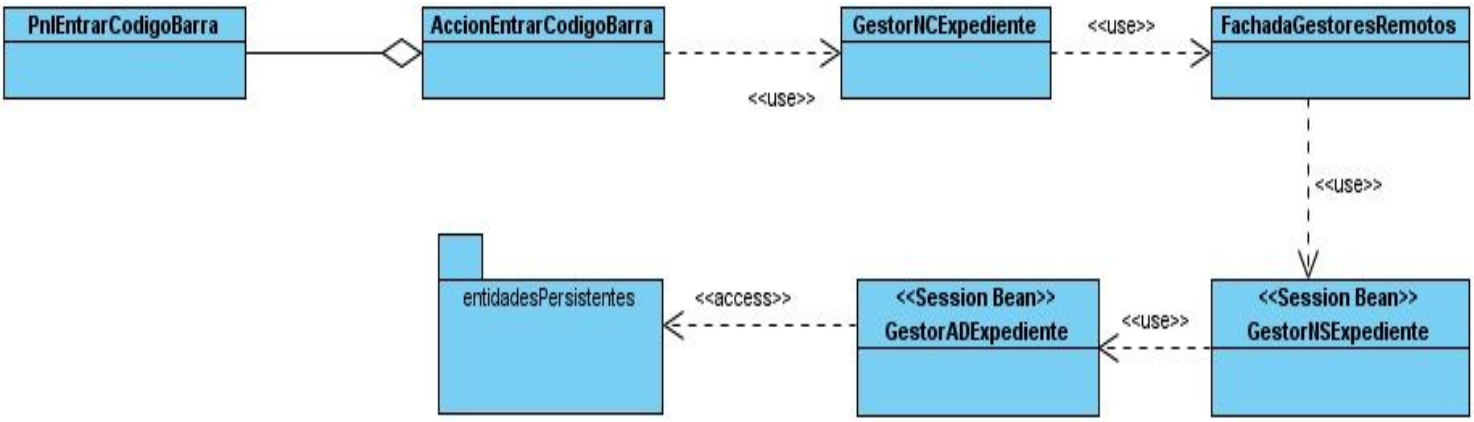


Fig.26: Diagrama de clase del caso de uso Verificar Código de Barras

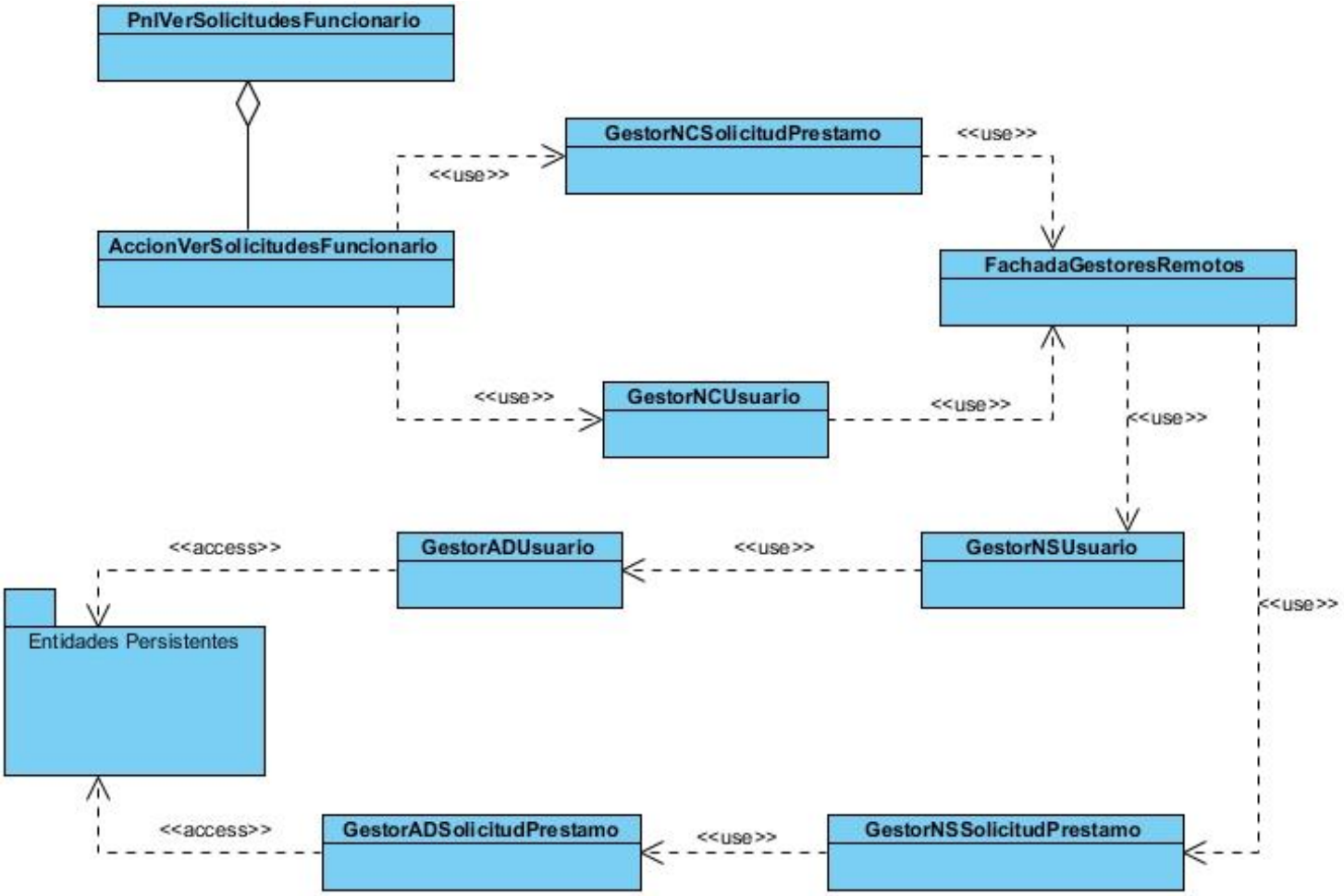


Fig. 27: Diagrama de clase del caso de uso Ver Solicitudes de Funcionario

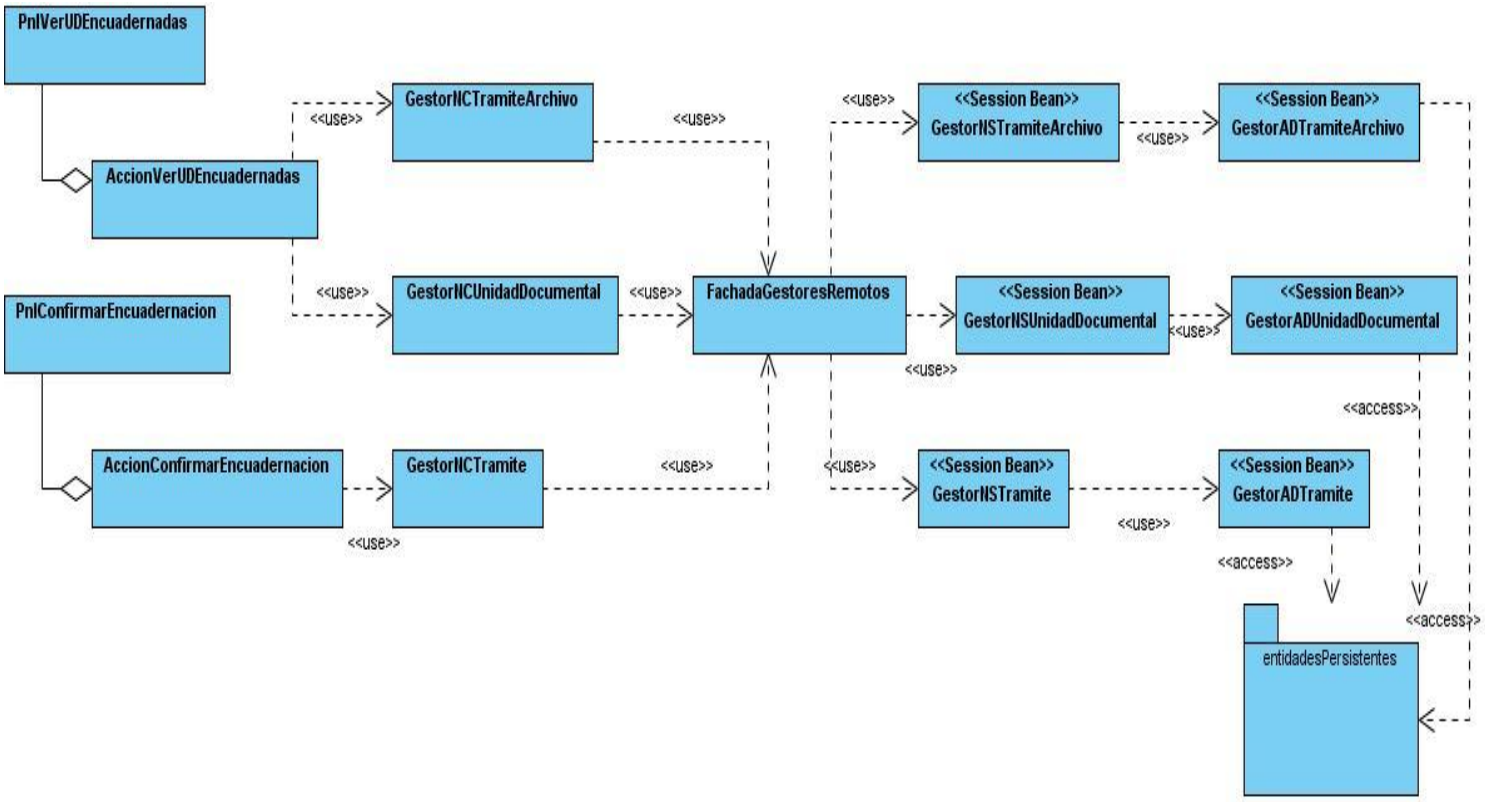


Fig.28: Diagrama de clase del caso de uso Confirmar Recepción de Expedientes

Anexo 2: Pautas generales para el diseño de interfaces

- Cuando la etiqueta no es para una entrada de datos, sino para mostrar los mismos, entonces irá en negrita.
  Ej.: **Nombre y apellidos:** Yolanda Acosta González.

- Las listas desplegables o combobox deben mostrar como texto inicial –Seleccione--.

Anexo 3: Estándares de codificación

Nomenclatura para los componentes de formularios:

- 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 ------ dtc.
- Para el componente JDayChooser ------ dyc.
- 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.
- Las tablas (JTable) comienzan con las siglas tbl.

Anexo 4: Diagrama de despliegue de la Solución Tecnológica General.

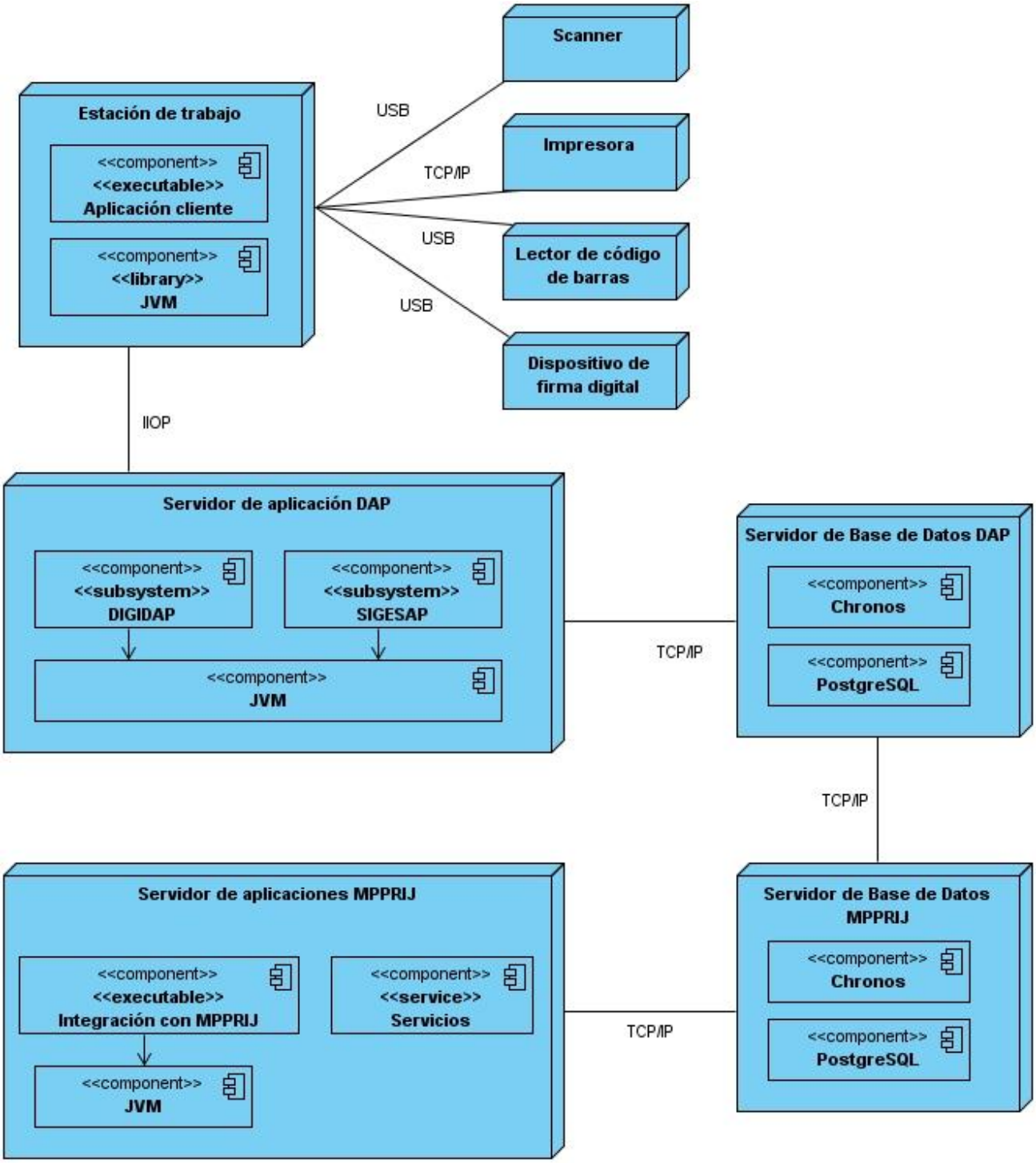


Fig.29: Diagrama general de despliegue de la Solución Tecnológica Integral

Anexo 5: Gestión de préstamos

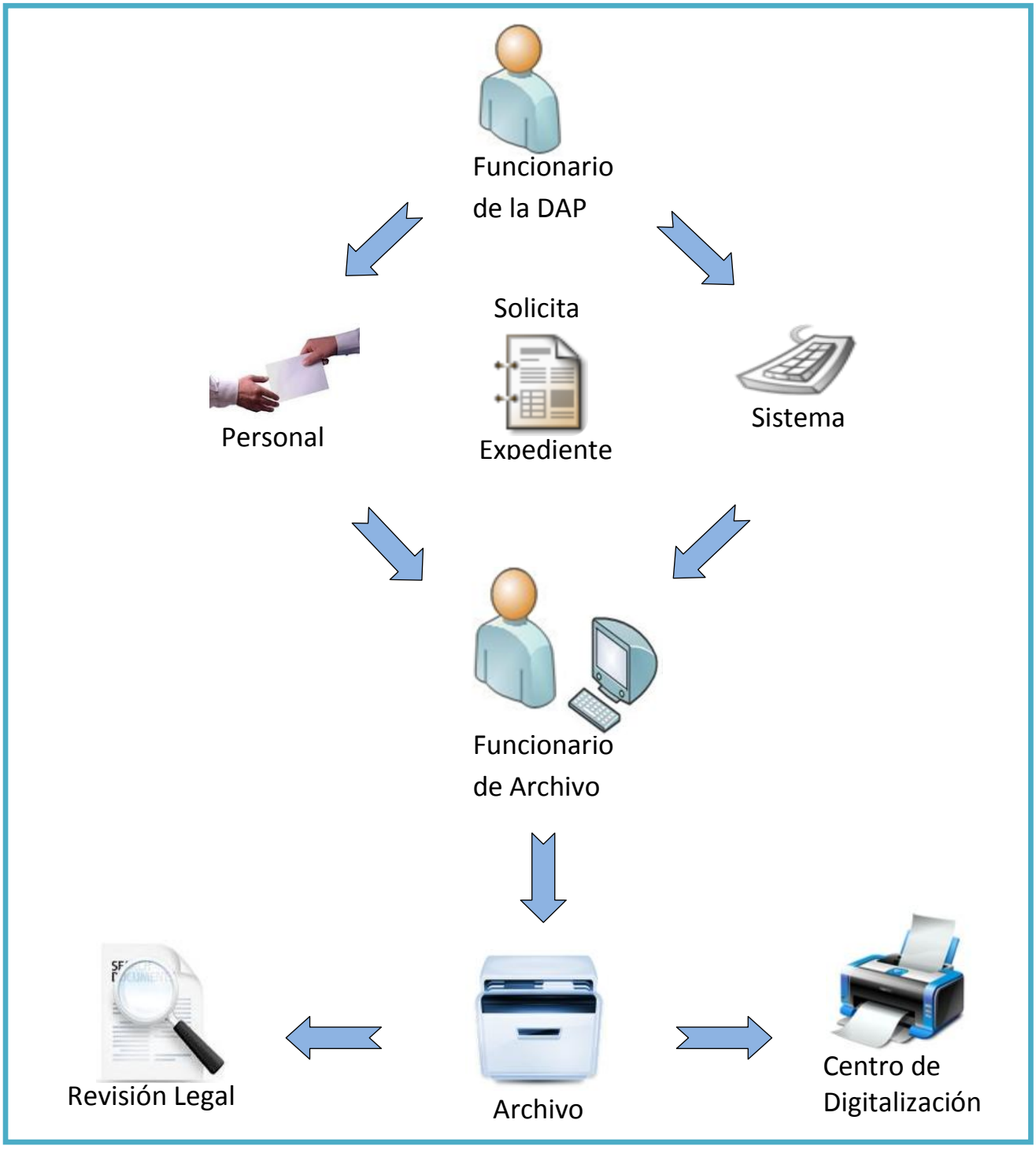


Fig. 30: Gestionar préstamos de expedientes.

Anexo 6: Patrones de diseño utilizados

```
public class FachadaGestoresRemotos {

    private static FachadaGestoresRemotos instancia;
    private InitialContext ctx = null;
    private String JNDI_PORTABLE = "java:global/sigesap-ea/modulo-integracion-ejb/";

    private FachadaGestoresRemotos() throws Exception {...}

    private InitialContext getCtx() {...}

    static public FachadaGestoresRemotos getInstancia() throws Exception {...}

    /**...*/
    private Object getGestorRemoto(String nombreBean) throws Exception {...}

    public GestorNSExpedienteRemoto getGestorExpediente() throws Exception {...}

    public GestorNSFichaRemoto getGestorFicha() throws Exception {...}
```

Fig.31: Fragmento de código de la clase FachadaGestoresRemotos
que muestra el uso del patrón instancia única

```
public final class SIGESAPI18n extends I18n {

    static private SIGESAPI18n instancia;

    public static SIGESAPI18n getInstancia() {
        if (instancia == null) {
            instancia = new SIGESAPI18n();
        }
        return instancia;
    }

    private SIGESAPI18n() {
        super(SIGESAPI18n.class.getPackage().getName().concat(".").concat("SIGESAPEtiquetas_es_ES"),
                SIGESAPI18n.class.getPackage().getName().concat(".").concat("SIGESAPMensajes_es_ES"),
                SIGESAPI18n.class.getPackage().getName().concat(".").concat("SIGESAPExcepciones_es_ES"),
                SIGESAP_RUTA_ICONOS.class.getPackage().getName().concat(".").concat("SIGESAPIconos"));
    }
}
```

Fig.32: Fragmento de código de la clase SIGESAPI18n que muestra el uso del patrón Instancia única

```
public class FachadaGestoresRemotos {

    private static FachadaGestoresRemotos instancia;
    private InitialContext ctx = null;
    private String JNDI_PORTABLE = "java:global/sigesap-ea/modulo-integracion-ejb/";

    private FachadaGestoresRemotos() throws Exception {...}

    private InitialContext getCtx() {...}

    static public FachadaGestoresRemotos getInstancia() throws Exception {...}

    /**...*/
    private Object getGestorRemoto(String nombreBean) throws Exception {...}

    public GestorNSExpedienteRemoto getGestorExpediente() throws Exception {...}

    public GestorNSFichaRemoto getGestorFicha() throws Exception {...}
```

Fig.33: Fragmento de código de la clase FachadaGestoresRemotos que muestra el uso del patrón fachada

```
public List<EDPrestamoMostrar> buscarPrestamosPorEstado(int idEstado) {
    List<TPrestamoExpediente> listaPrestamosEB = new ArrayList<TPrestamoExpediente>();
    List<EDPrestamoMostrar> listaMostrar = new ArrayList<EDPrestamoMostrar>();
    Query consulta = em.createNamedQuery("TPrestamoExpediente.buscarPrestamosPorEstado");
    consulta.setParameter("idEstado", idEstado);
    listaPrestamosEB = (List<TPrestamoExpediente>) consulta.getResultList();
    if (listaPrestamosEB.isEmpty()) {
        return listaMostrar;
    }
    for (Iterator<TPrestamoExpediente> it = listaPrestamosEB.
            iterator(); it.hasNext();) {
        TPrestamoExpediente tPrestamo = it.next();
        listaMostrar.add(factoriaEDSolicitudPrestamo.
                convertir(tPrestamo));
    }
    for (int i = 0; i < listaMostrar.size(); i++) {
        TUsuario funcionario = em.find(TUsuario.class,
                listaPrestamosEB.get(i).getIdFuncionario());
        listaMostrar.get(i).setNombreFuncionario(factoriaEDUsuario.convertir
                (funcionario).getNombreCompleto());
    }
    return listaMostrar;
}
```

Fig.34: Fragmento de código de la clase GestorADSolicitudPrestamo que muestra el uso del patrón Iterador

```
@Stateless
@RolesAllowed(RolBase.USUARIO_BASICO)
public class GestorNSSolicitudPrestamo implements GestorNSSolicitudPrestamoRemote {

    @EJB
    private GestorADSolicitudPrestamoLocal gestorADSolicitudPrestamo;

    public GestorNSSolicitudPrestamo() {
    }

    public List<EDPrestamoExpediente> obtenerSolicitudesFuncionario(double idFuncionario) {
        return gestorADSolicitudPrestamo.obtenerSolicitudesFuncionario(idFuncionario);
    }

    public void cancelarSolicitud(int idSolicitud) throws Exception {
        gestorADSolicitudPrestamo.cancelarSolicitud(idSolicitud);
    }
```

Fig.35: Fragmento de código de la clase GestorNSSolicitudPrestamo

```
@Stateless
public class factoriaEDSolicitudPrestamo implements factoriaEDSolicitudPrestamoLocal {

    @EJB
    private FactoriaEDEstadoPrestamoLocal factoriaEDEstadoPrestamo;
    @EJB
    private factoriaEDReoLocal factoriaEDReo;
    @EJB
    private factoriaEDExpedienteLocal factoriaEDExpediente;

    public TPrestamoExpediente convertirAT(EDPrestamoExpediente edPrestamo) {

        TPrestamoExpediente tPrestamo = new TPrestamoExpediente();
        NEstadoPrestamo nEstadoP = new NEstadoPrestamo();
        tPrestamo.setIdFuncionario(edPrestamo.getFuncionarioSolicitante().getIdUsuario());
        tPrestamo.setFechaSolicitud(edPrestamo.getFechaSolicitud().getTime());
        Calendar fechaSalida = Calendar.getInstance();
        tPrestamo.setFechaSalida(fechaSalida.getTime());
        nEstadoP = factoriaEDEstadoPrestamo.convertirANEstadoPrestamo(edPrestamo.getEstadoPrestamo());
        tPrestamo.setNEstadoPrestamo(nEstadoP);
        tPrestamo.setMotivo(edPrestamo.getMotivoPrestamo());
        //tPrestamo.setNumeroTramite(edPrestamo.getNumerotramite());
        TExpediente expediente = new TExpediente();
        expediente = factoriaEDExpediente.convertir(edPrestamo.getExpediente());
        tPrestamo.setTExpediente(expediente);
        tPrestamo.setEsFicha(true);
        return tPrestamo;
    }
```

Fig.36: Fragmento de código de la clase factoriaEDSolicitudPrestamo que muestra el uso del patrón Creador

```
@Local
public interface GestorADSolicitudPrestamoLocal {

    public List<EDPrestamoExpediente> obtenerSolicitudesFuncionario(double idFuncionario);

    public void cancelarSolicitud(int idSolicitud) throws Exception;

    public boolean estaExpedienteSentenciaPrestado(String numeroExpediente);

    public boolean estaFichaPrestada(String numeroFicha);

    public void RealizarSolicitud(List<EDPrestamoExpediente> prestamos);

    public void devolverExpediente(int idSolicitud) throws Exception;

    public void actualizarEstadoPrestamo(int idSolicitud) throws Exception;

    public List<EDPrestamoMostrar> buscarPrestamosPorEstado(int idEstado);
```

Fig.37: Fragmento de código de la interfaz GestorADSolicitudPrestamoLocal

```
public class AccionGestionarPrestamo extends AccionPanel implements IniciarFlujo
{
    // <editor-fold defaultstate="collapsed" desc="Atributos">
    private int inicio = 0;
    private int maximo = 30;
    private EDUsuario copiaFuncionario;
    private boolean buscaPorFuncioanrio = false;
    private static String texto1 = SIGESAPI18n.getInstancia().getEtiqueta
            ("AccionGestionarPrestamo.busquedaSinResultados");
    private static String texto2 = SIGESAPI18n.getInstancia().getEtiqueta
            ("AccionGestionarPrestamo.solicitudesEncontradas");
    private Calendar fechaSolicitudInicial = null;
    private Calendar fechaSolicitudFinal = null;
    private Calendar fechaServidor;
    private boolean buscandoPorDefecto = false;
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="Constructor">
    public AccionGestionarPrestamo() {...}
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="Inicializar formulario">
    public void inicializarFormulario() throws Exception {...}

    public PnlGestionarPrestamo getFormulario() throws Exception {...}//
    </editor-fold>

    Botones del asistente

    // <editor-fold defaultstate="collapsed" desc="Métodos">
    private void cargarEstadosSolicitud() throws Exception {...}
    private KeyListener keyEnterPressedBuscar = new KeyListener() {...};

    public void onBuscarSolicitudes(int inicial) {...}

    private void onBuscarFuncionario() {...}

    void busquedaAvanzadaPorFuncionario(EDUsuario funcionario, int inicial)
            throws Exception {...}

    private void activarBtonPrestarDevolverConfirmar() {...}

    private void onSiguienteResultado() {...}

    private void onAnteriorResultado() {...}

    private void llenarTabbla(List<EDPrestamoExpediente> prestamos) throws
            Exception {...}
```

Fig.38: Fragmento de código de la clases AccionGestionarPrestamos que muestra el uso del patrón Controlador

Anexo 7: Paquetes del módulo FrameworkComunes en Netbean

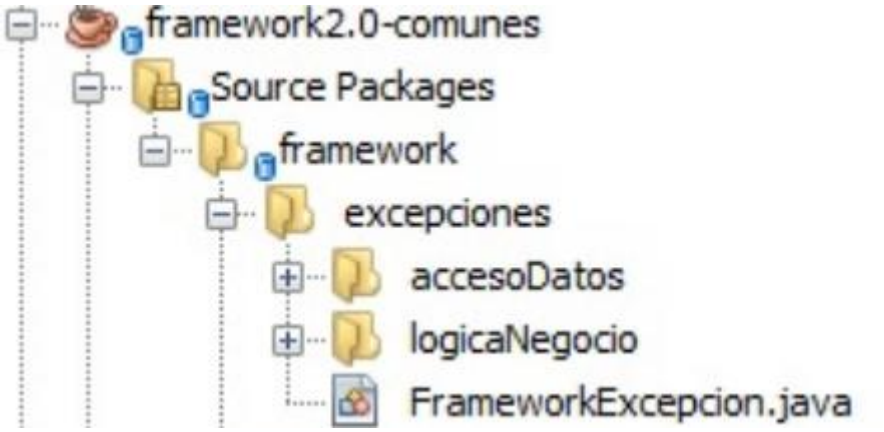


Fig.39: Estructura de paquetes en NetBeans el módulo FrameworkComunes

Anexo 8: Reportes de las pruebas de Caja Blanca Realizadas con JUnit

Class cu.albet.solucionSigesap.accesoDatos.gestores.GestorADSolicitudPrestamoTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
GestorADSolicitudPrestamoTest	1	0	0	0.101	2012-06-11T23:50:55	leyandry-PC

Tests

Name	Status	Type	Time(s)
testBuscarSolicitudesPorCriterios	Success		0.051

Fig.40: Reporte generado para el segundo caso de prueba aplicado al método BuscarSolicitudesPorCriterios

Class cu.albet.solucionSigesap.accesoDatos.gestores.GestorADSolicitudPrestamoTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
GestorADSolicitudPrestamoTest	1	0	0	0.103	2012-06-11T23:55:07	leyandry-PC

Tests

Name	Status	Type	Time(s)
testBuscarSolicitudesPorCriterios	Success		0.049

Fig.41: Reporte generado para el tercer caso de prueba aplicado al método BuscarSolicitudesPorCriterios