Universidad de las Ciencias Informáticas
Facultad 2

Título: "Desarrollo de los módulos Administración de Casos y Actividades de Atención para el régimen Extramuros del Sistema Penitenciario Venezolano".

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autores:

Zunielis C. Quintana Laguna
Dennis Molina Alfonso

Tutor:

Ing. Gueorgui Obregón Obregón

DECLARACIÓN DE AUTORÍA.

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

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

Zunielis C. Quintana Laguna

_____

Firma del Autor

Dennis Molina Alfonso

_____

Firma del Autor

Ing. Gueorgui Obregón Obregón

_____

Firma del Tutor

DATOS DE CONTACTO.

Ing. Gueorgui Obregón Obregón.

Graduado en el año 2008 de la carrera Ingeniería de la Ciencias Informáticas. Ha trabajado en la Universidad de Ciencias Informáticas desde septiembre de 2008 participando en eventos como el Fórum de Ciencia y Técnica 2008 a nivel de base. Se ha vinculado a los proyectos productivos en el rol de programador de Acceso a Datos y Diseñador, además ha impartido cursos de capacitación sobre el framework Hibernate a integrantes de proyectos productivos de la UCI.

Email: gobregon@.uci.cu

AGRADECIMIENTOS

Agradecer es una forma sublime por la cual una persona reconoce el valor de una palabra de apoyo. Es expresarles a las personas que nos dieron una palabra de aliento la gran ayuda que nos brindaron para que este día llegara.

A la Revolución por permitirnos la educación y formación profesional, por hacernos personas dignas de estos tiempos.

A la UCI por ser nuestro hogar, por darnos la posibilidad de crecernos como personas, por educarnos, por darnos tantos amigos.

A los profes por contribuir a nuestra formación profesional y revolucionaria.

A Karina por darnos el tema de la tesis y ser el líder del proyecto SIGEP que tanto ha significado para nuestra formación profesional.

A todos los del proyecto que nos hicieron sonreír y aportaron granitos de felicidad a nuestra estancia.

A nuestro tutor Gueorgui que estuvo con nosotros hasta el final, le agradecemos todas las horas de su valioso tiempo dedicadas a la realización de este trabajo de diploma.

A Roberto por habernos comprendido y haber estado junto a nosotros desde el principio hasta el final.

A todas las amistades que nos acompañaron en estos cinco años.

A todos los que nos ayudaron de una u otra forma, gracias.

DEDICATORIA.

A la razón de mi existir mí querida mamita por ser la persona que más me ama en esta vida y por todo el sacrificio que has hecho, espero que te sientas orgullosa de tu niña pues siempre anhelabas este triunfo, a ti que eres la persona que más quiero en esta vida. Te quiero tanto que el mundo se me hace pequeño y todo lo que pueda hacer por ti en mi vida me parecerá poco ante la dedicación y amor que siempre me brindaste.

A mi papá que aunque no está físicamente a mi lado ya que Dios me lo arrebato estando yo muy chiquita yo se que donde quiera que él esté me está cuidando y aunque la vida no le permitió verme graduarme yo se que él estaría muy orgulloso de mí.

A mí otro papito Pedri, gracias por haber sido para mí como un padre y ofrecerme tu cariño siempre incondicional.

A mí abuelita linda “Cuqui la Mora” gracias por tus consejos en los momentos más difíciles de mi vida y gracias por siempre confiar en mí.

A mí hermanito te quiero con la vida, sé que eres la persona con la que podré contar siempre, quien va a estar a mi lado en las buenas y malas, de quien siempre voy a recibir una ayuda y todo tu amor, para mí sería un orgullo que siguieras mi ejemplo.

A mí tía Migdy por mostrarme con su ejemplo como ser una mejor persona.

A mis tíos Josué y Fabri que me quieren tanto y yo los quiero mucho también.

A mis primitos Lorys, Carlos Daniel, Liz y Janielis que son una cosita muy importante en mi vida, los quiero mucho.

A mí familia completa, ustedes son parte de este trabajo.

A la Universidad por darme la oportunidad de formarme como profesional.

A mí compañero de tesis, Dennis, por apoyarme siempre que hizo falta.

A todas las personas del proyecto por ayudarme en cada duda que tenia y nunca decir que no.

A mis amigos y compañeros de aula que de una forma u otra siempre estuvieron presentes en cada momento de alegría y tristeza.

DEDICATORIA

A Toscano gracias por ayudarme tanto cuando me hizo falta y por todos tus consejos que me fueron de gran ayuda.

A mis amigas Arletis (La Farandulera), Yisel (Lalo) ustedes que siempre han sido tan divertidas, gracias por haber tenido la dicha de conocerlas y por lo importante que ha sido poder contar siempre con ustedes, nunca las olvidaré, las quiero mucho.

A Yanet, por ser la persona con la que he podido contar durante estos 5 años, el cariño que siento por ella nunca va a mermar.

A Yanet la de Guines y Yadelys por siempre ser tan buenas compañeras y excelentes amigas, nunca las olvidaré.

A Aymee por haberme entregado en tan poco tiempo su amistad, nunca olvidaré todos tus consejos.

A todos aquellos que también forman parte de mi selecto club de amigos de la UCI: Mis trigueños: El Puchy, Husse, Jorge Heriberto, Eudis, Maky, Sayli, Isabel, Maidelis, Yonaika, Yanet Quiala, Damian, Osmany, Daniela, Gretell, Mariem, Rita, Yeniel gracias por tenerlos cerca siempre que los he necesitado, con ustedes supe formar otra familia.

A mis grupos de danza “Los Sensuales de la Pista” y la rueda de casino, merengue y todo lo que baile por mi facultad por siempre dar lo mejor de sí sobre la tarima.

A todas aquellas personas que participaron y contribuyeron al desarrollo de este trabajo.

A todos, Muchas Gracias.

Zunielis.

A mí papá por apoyarme siempre y confiar en mí, por estar siempre a mi lado, por haberme ayudado y lograr lo mejor de mí, aun cuando haya condiciones adversas.

A mí hermano por ser mi apoyo en las situaciones difíciles, siempre poder contar contigo, por ser mi confidente y mejor amigo, por ser el que siempre está ahí cuando he caído.

A mí tía Pao que ha sido una madre para mi, por su amor incondicional y su fe en mí, por quererme tanto y entenderme, por defenderme y creer en mí cuando otros no lo han hecho.

A Reglita que mucho me ha ayudado y comprendido, por su apoyo y cariño in condicional.

A mí tío Juan Carlos que me ha apoyado y alentado, que en muchas ocasiones me ha guiado y dado soluciones.

A mis hermanos Yusniel, Yuniel, Yosmel , Javier, Yandy y Osiel que siempre creyeron en mí y me apoyaron, es mucho el camino que hemos recorrido juntos y el que nos queda por recorrer, por siempre poder contar con ellos, por ayudarme y apoyarme en todos los momentos, sobre todo por estar a mi lado en el día más difícil de mi vida.

A Jóse que ha sido un gran amigo, hermano, confidente, ha estado en cada situación conmigo, en momentos buenos y malos.

A todos los mariachis Agner, Pidio, Denier, Darien, Lester, Salmeron, Alexei, Yanniel, Carlos, Rogelio, Manuel, Yadier, Tito, Santo que más que compañeros han sido mis hermanos, con los cuales he compartido buenos momentos, y tenemos otros buenos recuerdos, de esos que no se olvidan nunca.

A Eriberto, Lester, Andrés, Adrian, Yasiel, Cartaya, Dino, Luis Felipe, Raiko, Jóse Miguel, Hussein, Ángel, Chao, Yeniel, Oscar, Lázaro que han sido a lo largo de estos cinco años un gran apoyo, muchos vinimos juntos desde el primer año que tanto significó, ya que ahí vimos quienes eran los de verdad, otros que he conocido en el recorrido pero siento que los conozco desde hace tiempo.

A mí compañera de tesis Zunielis que siempre me apoyo y tuvo paciencia conmigo, gracias por creer en mí.

A todos los del proyecto profesores y estudiantes que tanto me han ayudado y junto a ellos he pasado este último año.

A Adita que ha sido un gran apoyo y mi confidente en muchas ocasiones, que siempre ha estado a mi lado, que me entiende y sobre todo por su paciencia. Gracias por compartir conmigo este momento tan especial. Gracias por ayudarme tanto.

A todas las personas que de una forma u otra han formado parte de mi vida.

En especial a mí mamá que aunque no se encuentra en este mundo se que siempre estará conmigo, que me ayuda en los momentos difíciles y que nunca me dejará solo, gracias por todo, por hacerme el hombre que soy, para tí y por tí es todo mi esfuerzo, quiero ser tu orgullo.

Dennis.

RESUMEN

El Sistema de Gestión Penitenciaria de la República Bolivariana de Venezuela (SIGEP) fruto del proyecto de Humanización Penitenciaria, cuyo objetivo es realizar mejoras integrales de las condiciones de vida de los individuos en todos los centros de reclusión a lo largo y ancho del país, cuenta con el módulo Administración de Casos encargado de indicar cuáles de los funcionarios serán los encargados de llevar el expediente del penado y el módulo Actividades de Atención el cual permite al usuario registrar y consultar todas las actividades de atención desarrolladas en cada Centro de Residencia Supervisada.

El presente trabajo muestra la solución que dieron los autores a partir de las actividades realizadas por ellos como integrantes del SIGEP. En él se muestra el diseño e implementación de la solución brindada para los módulos Administración de Casos y Actividades de Atención, a partir de los requisitos definidos con el cliente y el estudio de la arquitectura definida para el proyecto. Se realiza una descripción de las herramientas, tecnologías y metodología utilizada para el diseño e implementación de los módulos, facilitando la comprensión del documento. Se analizan y describen los elementos más notables de la solución y los artefactos obtenidos como resultado del proceso.

Se llegó a la solución propuesta, con la obtención de los módulos Administración de Casos y Actividades de Atención. Como resultado de todas las actividades realizadas se integraron los módulos como componentes ejecutables al SIGEP 2.1.

ÍNDICE

INTRODUCCIÓN

Introducción

Con la llegada a la presidencia de la República Bolivariana de Venezuela del Comandante Hugo Rafael Chávez Frías en el año 1998, se comenzó a desarrollar un profundo programa de transformaciones sociales, pero para ello era necesario una nueva constitución que sirviera de base legal para todas las actividades que se querían llevar a cabo, por lo que a finales del año 1999 se aprueba la Constitución de la República Bolivariana de Venezuela y comienza una ola de reformas encaminadas a resolver los principales problemas que se habían heredado de las políticas neoliberales llevadas a cabo por los anteriores gobiernos. En este marco nace el proyecto de Humanización del Sistema Penitenciario que integra atención a la salud de los individuos, asesoría especializada y un sistema informático para gestionar y automatizar los procesos penitenciarios conocido por las siglas SIGEP. El objetivo es realizar mejoras integrales de las condiciones de vida de los individuos en todos los centros de reclusión a lo largo y ancho del país.

Las prisiones en todo el mundo han existido por siglos, con la finalidad de mantener en áreas restringidas a personas para sancionarlas por hechos delictivos que cometieron. Se ha demostrado que las personas que pasan por estos recintos pueden educarse o convertirse en personas más antisociales. Por ello es que varios gobiernos han tratado de que las prisiones ayuden a solucionar los males que la sociedad genera. Los establecimientos penitenciarios venezolanos se caracterizan por el hacinamiento, inadecuadas instalaciones físicas, graves deficiencias en materia de servicios públicos y asistenciales, imperio de la violencia, extorsión, corrupción, carencia de oportunidades y medios para la rehabilitación y reinserción de los privados(as) de libertad. En cuanto a los aspectos administrativos se aprecia lentitud de los procesos de administración y de gestión.

En el Sistema Penitenciario Venezolano existen dos tipos de regímenes para clasificar la ubicación de la población penal, el régimen intramuros, que agrupa a los reclusos que permanecen dentro de instalaciones penitenciarias y el régimen extramuros , que agrupa la población que se encuentra fuera de los recintos penitenciarios en formas alternativas de cumplimiento de pena como son: las Unidades Técnicas de Supervisión y Orientación (UTSO), no son más que aquellas personas que están bajo libertad condicional y los Centros de Residencia Supervisada (CRS), que conforman instituciones, de carácter especial, la cual está conformada por una edificación tipo residencia, proyectada para brindar alojamiento, alimentación, tratamiento integral, educación, recreación y asistencia médica básica a la población penal.

En estos centros se realiza la administración de casos con el propósito de indicar cuáles de los funcionarios serán los encargados de llevar el expediente del penado. También se realizan actividades de atención, las cuales le permiten al usuario registrar y consultar todas las actividades de atención desarrolladas en cada Centro de Residencia Supervisada.

La administración de casos y el control de las actividades de atención son procesos que todavía no se contemplan en la solución tecnológica SIGEP. En la actualidad la información relacionada con las afectaciones que se detectan en el penal se colectan a través de correos electrónicos, el teléfono o personalmente, esto trae consigo que la mayoría de la información sea incompleta y a su vez esto genere dificultad en la gestión de las actividades de atención y de la administración de casos. Además existe un inadecuado control de las Actividades de Atención pues estas se realizan de manera manual, la información que se recoge no posee formatos estándares porque no mantienen un flujo informativo uniforme. En la administración de los casos no se tiene un control preciso de los delegados de prueba y la cantidad de casos que estos tienen asignados.

Se identifica para este trabajo el siguiente problema a resolver:

El sistema que existe actualmente para procesar los casos y las actividades de atención en el régimen Extramuros es insuficiente para tener control de los procesos penales de los individuos.

Por tanto el objeto de estudio que guiará nuestra investigación se centrará en: el control de los procesos penales en el sistema penitenciario venezolano.

Con lo que se persigue el objetivo general siguiente:

Desarrollo de los módulos Administración de Casos y Actividades de Atención de los subsistemas Administración de la Sede y Atención, Supervisión y Orientación para garantizar el control de los procesos penales de los individuos en el régimen Extramuros del Sistema Penitenciario Venezolano.

El campo de acción lo constituye: el Control de la administración de casos y las actividades de atención en el régimen Extramuros del Sistema Penitenciario Venezolano.

Además, se trazaron los siguientes objetivos específicos:

- Elaborar el marco teórico de la investigación.

- Diseñar los módulos Administración de Casos y Actividades de Atención
- Implementar los módulos Administración de Casos y Actividades de Atención.
- Diseñar y realizar las pruebas al funcionamiento de los módulos implementados.

Posibles resultados:

- Modelo de diseño de los módulos Administración de Casos y Actividades de Atención para el régimen Extramuros.
- Modelo de Implementación de los módulos Administración de Casos y Actividades de Atención para el régimen Extramuros.
- Módulos Administración de Casos y Actividades de Atención para el régimen Extramuros funcional integrados al SIGEP.

El presente trabajo está dividido en los siguientes cuatro capítulos que recogen todo lo abordado en el trabajo investigativo.

Capítulo 1: Fundamentación Teórica. Se brinda un estado del arte sobre los elementos que forman parte del problema que se presenta, así como aquellos que son importantes a tener en cuenta para dar la solución que se requiere.

Capítulo 2: Características del Sistema. Describe la situación problémica y la propuesta de solución. Además se especifican las características que debe tener, se propone un flujo de actividades y se identifican las entidades que interactúan en dichas actividades.

En el Capítulo 3: Diseño e Implementación del Sistema. Construcción de la solución propuesta, aborda los detalles relacionados con la construcción del mecanismo y de cada uno de los elementos que lo conforman, además se fundamenta la elección de herramientas que agilizan la implementación del mismo.

En el Capítulo 4: Pruebas. En este se realizan las pruebas a las funcionalidades del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 **Introducción**

En este capítulo se ejemplifican sistemas para la gestión penitenciaria en Latinoamérica. Se abordarán conceptos que facilitarán la comprensión del documento y características de las herramientas, tecnologías y metodología previamente definidas en el proyecto SIGEP, utilizadas en el diseño e implementación de los módulos Administración de Casos y Actividades de Atención.

1.2 Concepto de Sistema Penitenciario.

Existen dos conceptos muy concatenados, el de Sistema y Régimen Penitenciario, según el libro Diccionario de Ciencias Jurídicas, Políticas y Sociales, en este se define régimen como: "conjunto de normas legislativas o administrativas encaminadas a determinar los diferentes sistemas adoptados para que los penados cumplan sus penas. Se encamina a obtener la mayor eficacia en la custodia o en la readaptación social de los delincuentes. Esos regímenes son múltiples, varían a través de los tiempos; y van desde el aislamiento absoluto y de tratamiento rígido hasta el sistema de puerta abierta con libertad vigilada "(1). De acuerdo a la situación existente en cada país se define una serie de normas que estandarizan los procesos penitenciarios.

Cada gobierno define la estructura de su Sistema Penitenciario de acuerdo a la legislación y las condiciones reales que posee. En el caso específico de la República Bolivariana de Venezuela, tal sistema está constituido por la legislación vigente, los métodos que se emplean para lograr su funcionamiento, las diferentes dependencias encargadas de su aplicación, los equipos de trabajo y la infraestructura carcelaria.

1.3 Sistemas Informáticos Penitenciarios.

Una pésima infraestructura y las dificultades en la estandarización de procesos de tratamiento penitenciario son características comunes en los Sistemas Penitenciarios de América Latina. En los últimos años con el objetivo de mejorar la situación existente en estos centros, varios países han venido desarrollando una campaña de modernización, donde entre otras metas, se encuentra la instalación de sistemas informáticos que ayuden al control y estandarización de los procesos y la información que de estos se genera.

Para desarrollar los módulos Administración de Casos y Actividades de Atención se realizó un estudio sobre varios sistemas informáticos similares al SIGEP, de los cuales

se obtuvo la información de sus respectivos sitios oficiales. Se proporcionan características de cada uno con el objetivo de mostrar sus fortalezas y debilidades.

Ejemplo de sistemas penitenciarios

1.3.1 Sistematización Integral del Sistema Penitenciario y Carcelario. (SISIPEC) de Colombia

Sitio web oficial: http://www.inpec.gov.co/portal/page/portal/Inpec

Es un proyecto de inversión desarrollado por el Instituto Nacional Penitenciario y Carcelario (INPEC) de Colombia en el 2004, creado para suplir la necesidad de sistematización de la información de los internos recluidos en los centros de reclusión. La finalidad del software es proporcionar un sistema de información integral y completo para el manejo de los detenidos y condenados a nivel nacional incluyendo establecimientos carcelarios y penitenciarios. Es un sistema centralizado que se enlaza con otros centros judiciales a través de sistemas de comunicaciones y computacionales de punta. Tiene instalado los módulos básicos Identificación: Sistema Progresivo, Ubicación, Estadía, Jurídico, Traslados, Beneficios, Disciplinario, Fomento, Social, además del módulo que gestiona el tratamiento penitenciario donde se controla y se vincula al interno al trabajo, se le da atención espiritual, social y psicológica así como asistencia médica especializada.

1.3.2 Gestión para Centros Penitenciarios.

Sitio web oficial http://www.neotec.cc/solutions/i3j/index_es.html

El proyecto i3j fue creado por una empresa de consultoría y desarrollo de software nombrada NEOTEC de Bolivia en el año 2001, es una solución de gestión de centros penitenciarios basado en la Web y tiene una base de datos centralizada. Está desarrollado en Java para la plataforma Java 2 Enterprise Edition (J2EE). Opera bajo los servidores de aplicaciones J2EE IBM Websphere, Sun Java System Application Server y el servidor open source Apache Tomcat y con las bases de datos IBM DB2, Oracle y la base de datos open source MySQL. La aplicación cuenta con un módulo Inter-Enterprise Integration (IEI) para compartir información selecta con las instituciones judiciales. La información de los prisioneros está disponible a estas instituciones a las cuales es posible otorgar acceso. Esta información incluye el registro de los reclusos, movimientos, visitas a reclusos, compañeros de celda, comportamiento y monitoreo de la libertad condicional. El sistema provee servicios para gestionar el tratamiento de los internos facilitando el control del estudio y el trabajo de los reclusos.

La evaluación de la conducta del recluso puede ser registrada. Esta función es especialmente útil para evaluar la conducta del recluso y establecer beneficios de condena. Sanciones y la historia médica del recluso también se registran. Informes de Seguimiento de Casos. Cada usuario tiene una visión completa de las actividades pendientes asignadas a él con un indicador de prioridad y la fecha de vencimiento, por ejemplo los reportes de evaluación del recluso. El supervisor tiene la lista de las actividades asignadas a sus dependientes así como las completadas y rechazadas. Con i3 es fácil para el supervisor dar seguimiento a casos y a al desempeño de su equipo.

1.3.3 Sistema Automatizado para el Control del Recluso (SACORE) Cuba.

Surge para dar cumplimiento a la Orden 43/99 del Vice Ministro Primero del Ministerio del Interior de Cuba y tiene las siguientes características:

• Garantiza respuestas inmediatas a las solicitudes de información de los diferentes órganos e instituciones del estado como son: Jefatura del MININT, Ministerio de Justicia, Tribunales, Fiscalías, MINED, INDER, FMC, MINFAR.

• Recoge prácticamente la totalidad de la información de los reclusos en todas las especialidades.

• Tiene más de 200 reportes impresos.

• Permite la recuperación dinámica a partir de una solicitud de búsqueda.

• Los partes que se emiten son obtenidos de forma automatizada.

• Permite el traslado automático de todos los datos del recluso al nivel nacional.

El SACORE cuenta con educadores guías que son los encargados de seguir la trayectoria de los prisioneros que le son asignados, El Educador Guía es el máximo responsable del funcionamiento de su Colectivo, organiza y ejecuta el tratamiento educativo, con la utilización de controles, métodos, medios, medidas y actividades, de acuerdo al objeto-sujeto de su trabajo. El SACORE no contempla procesos relacionados con los CRS, ni centros de Rehabilitación, lo más cercano son los Campamentos, donde los penados salen a trabajar con una rigurosa custodia.

1.4 Metodología, tecnologías y herramientas utilizadas en el desarrollo.

La selección de las herramientas correctas tiene una relación directa con el tiempo y la calidad de los artefactos asociados al ciclo de vida del software. Todas las herramientas y tecnologías utilizadas en la realización del SIGEP fueron definidas al inicio del proyecto por el Grupo de Arquitectura con el objetivo de garantizar la calidad del producto. A continuación se hace una breve descripción de las herramientas y tecnologías utilizadas.

1.4.1 Proceso Unificado de Desarrollo (RUP, Rational Unified Process).

Se decide utilizar Rational Unified Process (RUP) como metodología de referencia para el desarrollo del software, por ser un proceso:

- ✓ Iterativo e incremental: Se divide en 4 fases: Inicio, Elaboración, Construcción y Transición, y cada una de ellas se divide en iteraciones. En cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas. Las disciplinas propuestas por RUP son: Modelado del negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión de proyecto y Ambiente. Cada iteración añade funcionalidades al producto de software o mejora las existentes.

- ✓ Dirigido por casos de uso: el cual es uno de los métodos más utilizados y efectivos para reflejar los requisitos. Son los encargados de guiar el ciclo de vida del proyecto (JACOBSON, 1999).

- ✓ Centrado en la arquitectura: lo cual permite organizar o estructurar el sistema en sus partes más relevantes e ir refinando esta estructura progresivamente. RUP define “un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto”. (JACOBSON, 1999) Propone flujos de trabajo en los que se definen las secuencias de actividades, quienes las deben desarrollar y los artefactos a generar.

En la Figura 1.1 se muestran las fases y las disciplinas que define RUP.

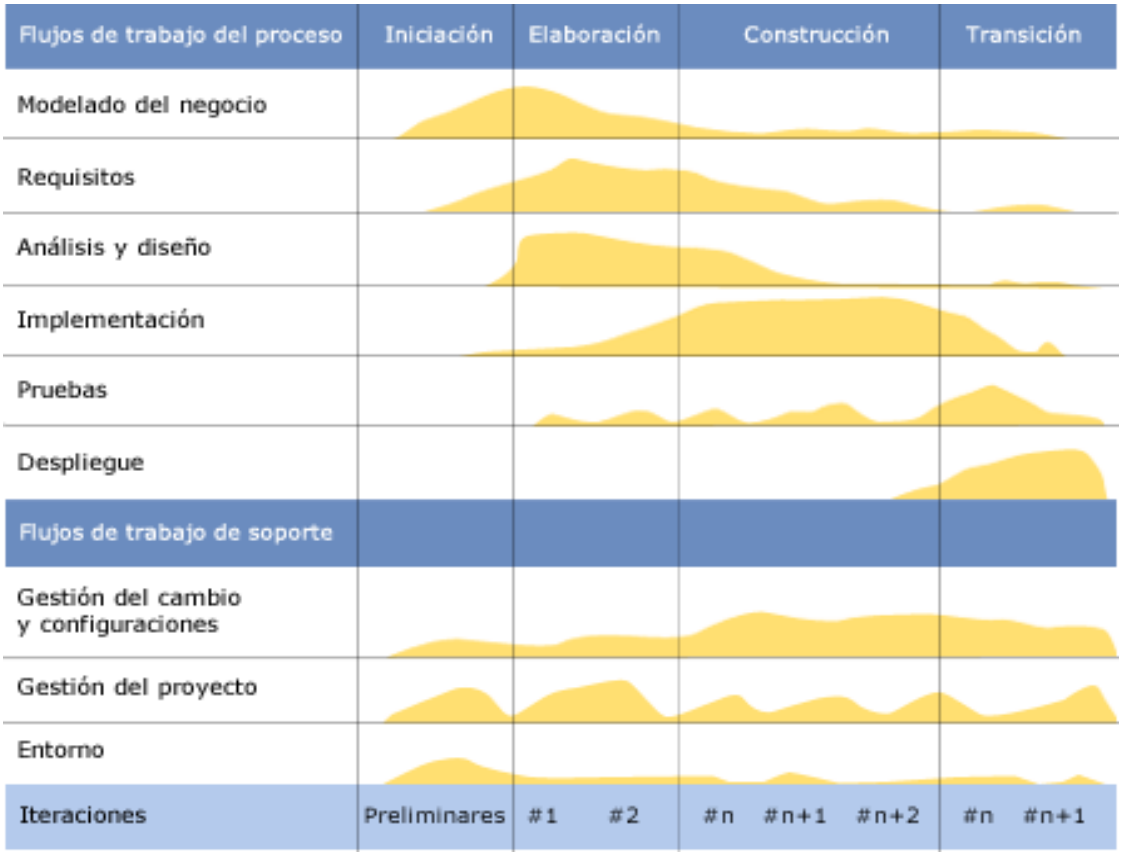


Figura 1.1 Fases y disciplinas de RUP.

En el proyecto SIGEP se decidió establecer RUP como una referencia para el desarrollo del proyecto, por ser adaptable al contexto y necesidades del mismo. RUP es frecuentemente utilizada en proyectos de gran envergadura como el SIGEP por ser exhaustiva en la generación de documentación basándose en UML. Es práctica para el trabajo con equipos de desarrollo grandes y por tanto difíciles de organizar ya que predefine una serie de roles con tareas y guías de trabajo que reflejan básicamente lo que se espera de ellos. Dentro de los principales roles que RUP define están: ingeniero de requisitos, diseñador, arquitecto, implementador, líder de proyecto y probador.

1.4.2 Lenguaje Unificado de Modelado (UML, Unified Modeling Language).

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Contiene construcciones para representar decisiones de implementación y para elementos de

tiempo de ejecución en componentes. (4) UML proporciona tres beneficios claves: la visualización, la gestión de la complejidad y la comunicación clara. En UML hay diferentes tipos de diagramas que en la Figura 1.2 aparecen estructurados para un mejor entendimiento.

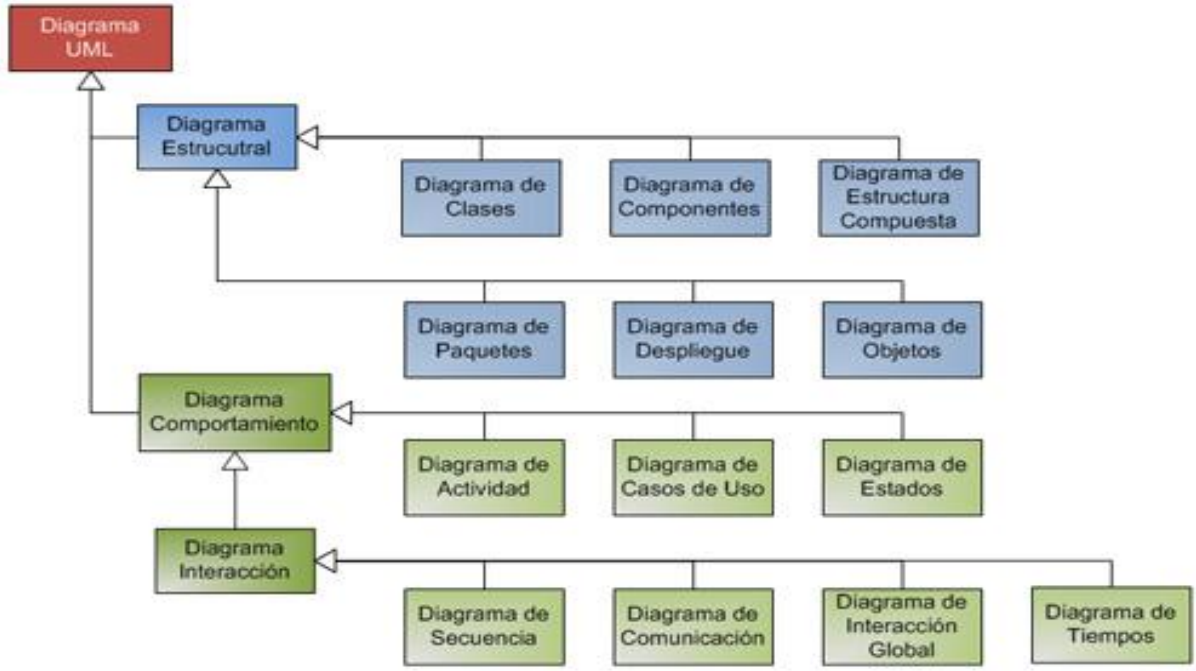


Figura 1.2: Estructura de Diagramas UML

1.4.3 Visual Paradigm 3.4 Enterprise Edition.

Es una herramienta de de código abierto, de modelado que soporta: UML, la captura de requisitos, diseño de base de datos, modelado de procesos de negocio para el analista de sistemas y desarrollador de software, además permite realizar el diseño y mantener aplicaciones de software de una forma disciplinada y colaborativa. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Visual Paradigm se integra con el IDE Eclipse usado en el proyecto SIGEP permitiendo la generación de código Java y documentación a partir de diagramas. (5)

1.4.4 Erwin Data Modeler.

Es una herramienta de diseño de bases de datos que te ayuda a generar, y mantener alta calidad y gran rendimiento en las aplicaciones de bases de datos. Desde un modelo lógico de los requerimientos de información y las reglas de negocio que

definen la base de datos al modelo físico optimizado por las características especificas de la base de datos. Presenta algunas características como son:

- Fácil acceso a cualquier base de datos relacional.
- Comparación comprensiva entre el modelo de datos y la base de datos
- Soporta la separación del modelo lógico y del físico.

1.4.5 Patrones de Diseño.

Los patrones de diseño se rigen por la recopilación del conocimiento de los expertos en desarrollo de software ya que se utilizan en situaciones frecuentes basándose en la experiencia acumulada en resolver problemas que se han convertido en reiterativos por lo que evita la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, favoreciendo la reutilización de código, formalizando un vocabulario común entre los diseñadores y facilitando el aprendizaje de las nuevas generaciones condensando conocimiento ya existente. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes. A partir de los lineamientos arquitectónicos definidos en el proyecto SIGEP, los patrones de diseño más utilizados son los siguientes:

Data Access Object (DAO).

Se encarga de manejar todo lo relacionado con la capa de Acceso a Datos de una aplicación. Se usa para aislar la aplicación de la tecnología de persistencia utilizada, en el caso de que se quiera cambiar o actualizar la tecnología de persistencia, no es necesaria que otras partes de la aplicación sean afectadas por dicho cambio. El patrón DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos y oculta completamente los detalles de implementación de la fuente a sus clientes. Su principal ventaja es que separa en una capa aparte todo el acceso a datos y abstrae la comunicación con la fuente de datos, esto hace que sea mucho más fácil una migración de fuente de datos en caso de ser necesaria.

En el SIGEP por cada objeto del dominio persistente se define un DAO que expone a través de una interfaz sus funcionalidades donde la implementación está relacionada directamente con la tecnología de persistencia.

Facade (Patrón Fachada).

Sirve para proveer una interfaz unificada sencilla llamada fachada que haga de intermediaria entre un cliente y un grupo de objetos más complejos. Reduce el número de objetos con los que tiene que interactuar un cliente proporcionando un menor acoplamiento y facilitando el cambio de componentes sin afectar a sus clientes.

Model-View-Controller (Patrón Modelo-Vista-Controlador, MVC).

Divide una aplicación en tres componentes, el modelo, las vistas y los controladores:

- ✓ **Modelo:** contiene los datos resultantes de la ejecución de la lógica de negocio, los cuales son mostrados en la respuesta.
- ✓ **Vistas:** son las encargadas de mostrar los datos del modelo que han sido suministrados por un controlador. Existen diferentes tipos de vistas para mostrar los modelos como JSP (Java Server Page), HTML (Hyper Text Markup Language), documentos PDF, documentos de Excel e imágenes.
- ✓ **Controlador:** es el responsable de procesar las entradas del usuario en forma de peticiones HTTP (Hyper Text Transfer Protocol), invocando las funcionalidades necesarias, expuestas por la capa de lógica de negocio. (6)

1.4.6 Plataforma de desarrollo.

1.4.6.1 Java 2 Enterprise Edition (J2EE).

La plataforma J2EE (Java 2 Enterprise Edition) es un estándar para el desarrollo de aplicaciones empresariales utilizando el lenguaje de programación Java. La tecnología Java 2 Enterprise Edition (J2EE) es un estándar de la industria para desarrollar aplicaciones empresariales portables, robustas, escalables y seguras. Define una arquitectura para desarrollar aplicaciones distribuidas complejas utilizando un modelo multicapas. Integra un conjunto de APIs, frameworks y patrones de programación. La lógica de aplicación está dividida en componentes de acuerdo a su función. Los componentes de la capa cliente se ejecutan en la máquina cliente, los componentes web y de la capa de negocio se ejecutan en el servidor de aplicaciones y otros componentes en sistemas legados o servidores de bases de datos.

1.5 Tecnologías de desarrollo.

1.5.1 Java-Servlets.

Permite extender las capacidades de un servidor de aplicaciones que es accesible a través del modelo petición-respuesta. Generalmente es utilizado en ambientes web. Los servlets proveen un mecanismo efectivo de interacción entre la lógica de negocio que se ejecuta en un servidor y las aplicaciones clientes. Los servlets viven en un contenedor de servlets que se ejecuta en un servidor web. Este contenedor gestiona su ciclo de vida y traduce las peticiones que un cliente web hace a través del protocolo HTTP, en objetos que las encapsulan. De igual forma, el contenedor traduce las respuestas de los servlets al protocolo web correspondiente.

1.5.2 Java Server Pages (JSP).

Como parte de la familia de la tecnología Java, la tecnología JSP permite el desarrollo rápido de aplicaciones basadas en Web que son independientes de la plataforma. La tecnología JSP separa la interfaz de usuario de la generación de contenidos, permitiendo a los diseñadores cambiar el diseño de la página en general, sin alterar el contenido dinámico subyacente.

1.6.1 Frameworks.

Es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se pueden añadir las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo, como el uso de patrones. Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

1.6.2 Spring framework v2.0.

Es un marco de trabajo de aplicación de código abierto que ayuda a hacer el desarrollo en JEE mucho más fácil. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes. Es el más popular y el más ambicioso de todos los framework de peso ligero. Este framework interviene en todas las capas arquitectónicas de una aplicación JEE. Además está diseñado para facilitar una flexibilidad arquitectónica. El mismo presenta varios módulos bien definidos como se muestra en la Figura 1.3.

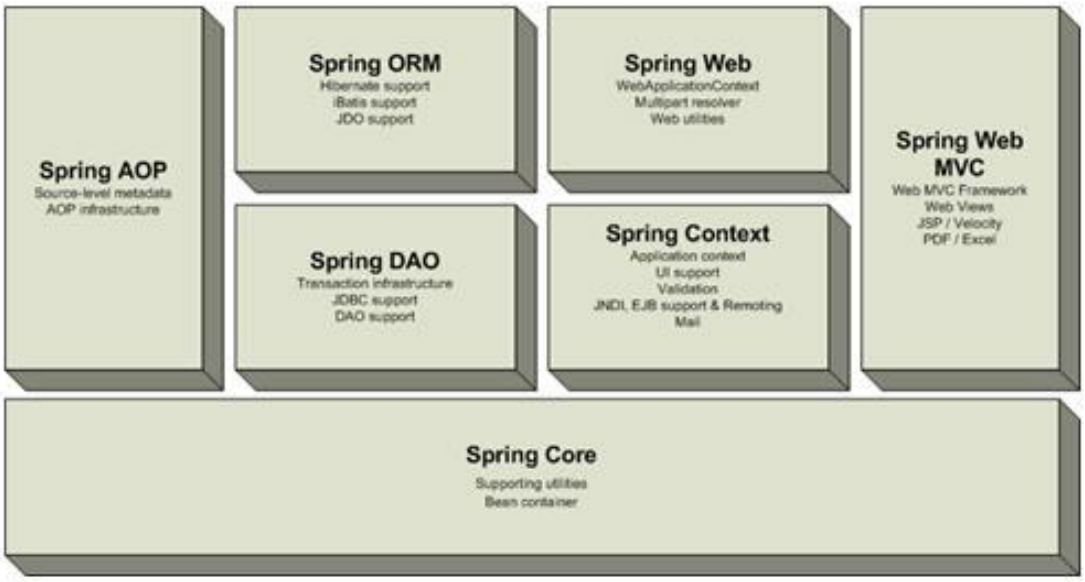


Figura 1.3 Módulos del Framework Spring.

Los módulos de Spring utilizados en proyecto fueron AOP, ORM, MVC, DAO, CORE.

1.6.2 Hibernate framework v3.2.0 cr4.

Hibernate es un potente framework de mapeo objeto/relacional y generador de consultas SQL para Java. Permite diseñar objetos persistentes que podrán incluir asociación, composición, polimorfismo, relaciones, colecciones y un gran número de tipos de datos. Ofrece su propio lenguaje de consulta de datos HQL (Hibernate Query Language). Además, permite generar sentencias de manera fácil y optimizada en cualquiera de los entornos que soporta como Oracle, MySQL, PostgreSQL, entre otros. Evita tener que realizar cualquier manipulación de la información que se obtiene a partir de la ejecución de las consultas. (7)

1.7 Entorno Integrado de Desarrollo (IDE) 2.0.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#.

1.7.1 Eclipse 3.4.

Eclipse es una plataforma de software de código abierto independiente de una plataforma para desarrollar. El ambiente de desarrollo integrado de Eclipse emplea módulos (plugins) para proporcionar toda su funcionalidad al frente de la plataforma de

cliente rico, a diferencia de otros entornos donde las funcionalidades están todas incluidas. Permite además desarrollar aplicaciones en varios lenguajes de programación como son Java, C++, PHP y otros. (8)

Plugins:

Un plugin o plug-in, es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible. Dentro de los más usados para el desarrollo del SIGEP encontramos:

Spring IDE 2.0:

Spring IDE es un plugin que sirve como interfaz de usuario gráfica para la configuración de los archivos usados por Spring Framework. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración.

Hibernate Tools:

Constituye un conjunto de herramientas para facilitar el uso del framework Hibernate. Las principales funcionalidades que brinda son: un editor de mapeos de los meta datos de la base de datos a las clases y una consola para la ejecución de consultas HQL. Permite realizar ingeniería inversa a partir de la base de datos de las clases y de los mapeos de las entidades.

Subclipse v1.0.1:

Subclipse es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion), permitiendo operaciones de sincronización, actualización, y dispone de una vista de comparación entre el recurso local y remoto en caso de que exista conflicto entre la versión del recurso local con el remoto. Es un plugin muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.

1.8 Gestor de Base Datos.

Oracle 10g.

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos

garantizan la seguridad e integridad de los datos; que las transacciones se ejecutan de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos, es estable, escalable y multiplataforma. (9)

1.9 Librerías.

Las librerías son un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables.

1.9.1 Dojo Toolkit 0.4.

Es una biblioteca JavaScript de código abierto la cuál brinda una variedad de clases y widgets para facilitar el desarrollo de aplicaciones web. La idea fundamental de la librería es abstraer al desarrollador de las dificultades de DHTML y de las diferencias entre navegadores, que hacen que el código JavaScript sea diferente para cada uno de ellos. La gran variedad de clases componentes y widgets, es un punto muy importante en la elaboración en la capa presentación de la aplicación, posibilita a los programadores de interfaz una serie de funcionalidades y elementos dinámicos que facilita la programación en el cliente. (10)

1.10 Servidor Web.

1.10.1 Apache Tomcat 5.5.17.

Apache Tomcat es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y Java Server Pages (JSP). Es desarrollado en un ambiente colaborativo y abierto. Apache Tomcat es usado en numerosas aplicaciones web de gran escala y críticas en diversas industrias y organizaciones que se referencian en su sitio oficial. (11)

1.11 Formatos de Texto

1.11.1 XML

XML (Extensible Markup Language [lenguaje de marcas extensible]) es un lenguaje de etiquetas desarrollado por World Wide Web Consortium10 (W3C). Es un lenguaje sencillo y de fácil interpretación por las personas por lo que es muy usado por los desarrolladores de software. Las especificaciones del mismo definen una manera estándar de añadir marcas a los documentos, permitiendo un intercambio de

información estructurada entre diferentes plataformas. Puede ser usado en bases de datos, editores de texto, hojas de cálculo, entre otros. Los archivos de configuración de la aplicación están en formato XML, facilitando la gestión de cambios en las configuraciones. Además permite adicionar nuevas etiquetas, lo que lo hace un lenguaje extensible, característico que lo distingue.

CONCLUSIONES

En el presente capítulo se realizó una breve descripción de los procesos de negocio relacionados con los módulos Administración de Casos y Actividades de Atención. Además se realizó una descripción de la metodología de desarrollo, las herramientas y los patrones a utilizar para la construcción del software. SIGEP es una aplicación web de gestión empresarial cuyo equipo de desarrollo es numeroso. Basado en este hecho se utilizó RUP como metodología de desarrollo de software. Los flujos de trabajo sobre los que se basa la realización del presente trabajo son Diseño e Implementación La aplicación en desarrollo está solicitada para que se ejecute sobre ambiente libre. De esta manera las herramientas y tecnologías utilizadas en su mayoría son libres y multiplataforma. Esto trae como consecuencia positiva una reducción notable del costo del sistema por concepto de licencias de software y soporte.

CAPÍTULO 2: ARQUITECTURA DEL SISTEMA

2.1 Introducción

En el capítulo siguiente se describen las características de la arquitectura del SIGEP, a partir de las cuales se desarrollan los módulos Administración de Casos y Actividades de Atención. Se caracterizan las capas por las que está compuesta y se brinda un breve análisis de estas. Además se describen brevemente las funcionalidades de los módulos Administración de Casos y Actividades de Atención.

2.2 Arquitectura del SIGEP

Para el desarrollo del SIGEP se definió e implementó como arquitectura base ArBaWeb, (Arquitectura Base sobre la Web). La arquitectura está organizada teniendo en cuenta dos características fundamentales, la definición de subsistemas y módulos, y el modelo de capas, basado en la arquitectura Cliente-Servidor y la arquitectura de tres capas lógicas bien definidas.

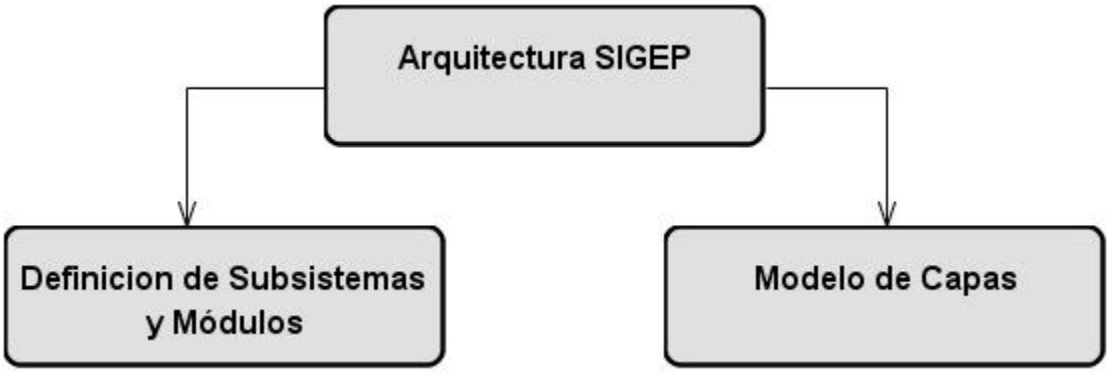


Figura 2.1 Características fundamentales de la arquitectura del SIGEP.

2.2.1 Modelo de capas

La arquitectura del SIGEP está regida por tres capas lógicas bien definidas, Capa de Presentación, Capa de Negocio y la Capa de Acceso a Datos. A continuación se muestra en la Figura 2.2 una imagen de las mismas.

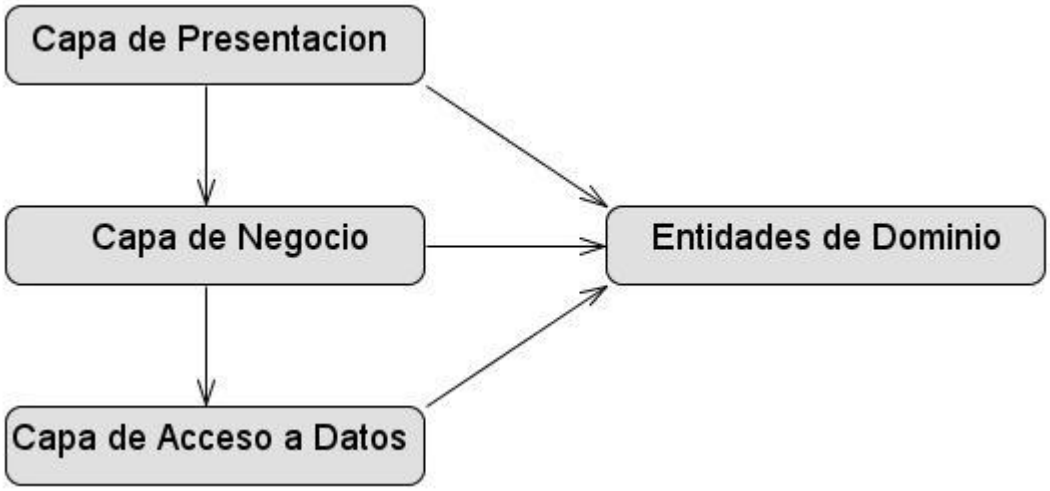


Figura 2.2 Modelo de capas

En esta arquitectura cada una de las capas puede ser modificada tanto como sea posible sin afectar en nada las otras capas definidas, ninguna de estas es consciente de lo que ocurre en las capas superiores, solo tienen relación con las capas inmediatamente inferiores. Esta dependencia con las capas inferiores es generalmente con las interfaces para asegurar un acople bajo.
A continuación se describen en detalle cada una de las capas que aparecen en la Figura 2.2, comenzando de abajo hacia arriba.

Capa de Acceso a Datos

Capa responsable de persistir y recuperar información hacia y desde la base de datos, se encarga también de la comunicación con el gestor de base de datos utilizando el framework de persistencia Hibernate. Esta capa contiene los objetos que encapsulan la lógica de acceso a datos (DAO) y brinda interfaces para que sea accedida desde la capa de negocio. Las implementaciones de los DAOs estarán disponibles para los objetos (típicamente para los objetos de negocio) haciendo uso de la inyección de dependencias con los objetos de negocio y las instancia de los DAOs, configurada en el contenedor de inversión de control de Spring Framework.

Capa de negocio

Capa responsable de definir e implementar las funcionalidades que responden directamente a los requisitos. En esta capa radican los objetos de negocio o Business Objects. Los objetos de negocio separan los datos y la lógica de negocio usando un modelo de objetos. Para cada módulo están definidas una o más fachadas en caso de que se requieran las que agrupan los métodos de negocio implementados en los manejadores o managers. La fachada de un módulo está basada en el patrón Facade

para permitir una clara división entre las capas arquitectónicas. Los managers son las clases que se especializan en un conjunto de funcionalidades que representan el negocio sobre una o varias entidades. Estas son las únicas clases en la aplicación que presentan lógica de negocio mientras que las fachadas se limitan solamente a agrupar las funcionalidades de los managers para ser expuestas a capas superiores.

Capa de Presentación

En esta capa se utiliza Spring MVC como framework Model View Controller (MVC) para manejar el flujo de datos entre el cliente y las capas inferiores.
Es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado. Está compuesta por tres tipos de objetos:

Controladores: Estos objetos son responsables de procesar las entradas del usuario en forma de peticiones HTTP, invocando las funcionalidades necesarias expuestas por la capa de servicios de negocio y devolviendo un modelo requerido para ser mostrado.

Modelo: Estos objetos (command) contienen los datos resultantes de la ejecución de la lógica de negocio los cuales son mostrados en la respuesta.

Vistas: Son responsables de mostrar el modelo resultante en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP, HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los datos del modelo que han sido suministrados por un controlador.

El cliente interactúa con esta capa directamente utilizando el navegador Web a través de peticiones URL. En el cliente se forma dinámicamente una pequeña capa de presentación que está formada por código HTML y JavaScript. Esta capa interactúa mediante peticiones basadas en la tecnología Ajax con los componentes necesarios del servidor creados bajo la tecnología de Spring MVC.

2.2.2 Definición de Subsistemas y Módulos.

Módulo

Un módulo encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño.

Subsistema

Un subsistema se refiere a un conjunto de módulos que por razones de similitud o de perseguir objetivos comunes son agrupados.

2.2.3 Estructura de un módulo en el SIGEP.

En un módulo existen todas las capas lógicas que se definieron en la arquitectura. A continuación se muestra esta estructura.

Figura 2.3 Estructura de paquetes de un módulo.

Cada módulo está compuesto por los siguientes paquetes:

businessvalidator: Contiene las interfaces de las validaciones del negocio. businessvalidator.impl: Este paquete contiene todas las implementaciones de las validaciones del negocio.

configuration: Contienen los archivos que tienen que ver con la configuración del módulo, los “Application-Context” de Spring Framework, los archivos utilizados para la internacionalización, ficheros de propiedades “.properties” y cualquier otro destinado a estos fines.

dao: Se encuentran las interfaces DAOs.

dao.impl: Se encuentran las implementaciones de las interfaces DAOs.

dao.map: Se encuentran los ficheros de mapeo utilizados por Hibernate.

dao.test: Se encuentran las clases de prueba utilizadas para realizar las pruebas de unidad a las implementaciones de los DAO.

domain: Se encuentran todas las entidades persistentes o no del dominio pertenecientes al módulo.
manager: Se encuentran las interfaces de los managers de negocio.
manager.impl: Se encuentran las implementaciones de los managers de negocio.
facade: Se encuentran las interfaces de las fachadas de negocio.
facade.impl: Se encuentran las implementaciones de las fachadas de negocio.
error: Se encuentran las excepciones y las clases de utilidad para las validaciones.
web: Se encuentran los controladores de la capa de presentación.
web.command: Se encuentran las clases utilizadas para guardar datos procedentes de las peticiones web (Command).
web.editor: Se encuentran los PropertyEditors.
web.validator: Se encuentran las clases utilizadas para validar datos (Validadores).

Estructura de un Subsistema.

Un subsistema contiene varios módulos que tienen similitudes. En la Figura 2.4 se expone esta estructura:

Figura 2.4 Estructura de paquetes de un Subsistema.

Para cada subsistema existe un módulo común donde se gestionan las funcionalidades comunes nombrado “common” ver Figura 2.5. Los demás paquetes encapsulan los módulos pertenecientes al subsistema.

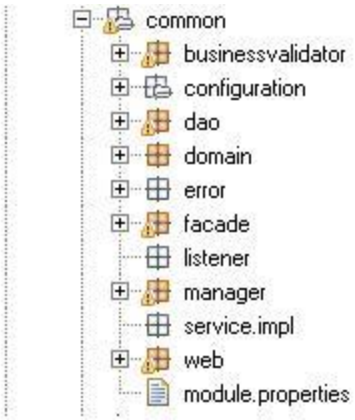
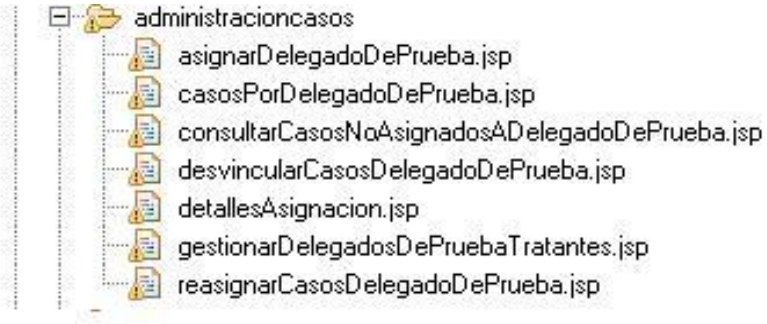


Figura 2.5 Representación del módulo "common".

En las siguientes figuras se muestran las estructuras de las “Vistas” de los módulos Administración de Casos y Actividades de Atención, ubicadas en la capa de presentación.



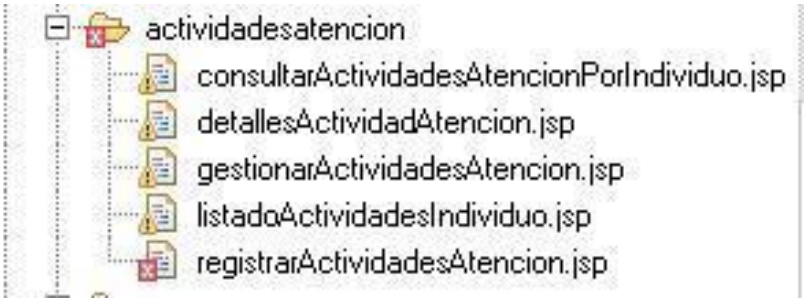


Figura 2.6 Ubicación de las paginas JSP (Java Server Pages).

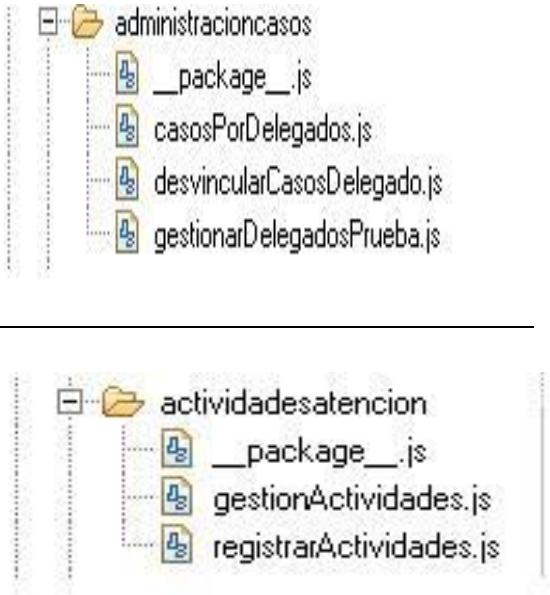


Figura 2.7 Ubicación de los archivos contenedores del código Java Script 2.3

Descripción de las funcionalidades

A continuación se listan las funcionalidades definidas para los módulos Administración de Casos y Actividades de Atención de los subsistemas Administración de la Sede y Atención, Orientación y Supervisión. De cada una se especifica el nombre y una breve descripción.

Módulo Actividades de Atención

Nombre: Gestionar actividades de atención.

Breve Descripción: En esta funcionalidad se muestran todas las actividades de atención registradas en el Centro de Residencia Supervisada. Permite registrar o modificar actividades, consultar sus detalles y eliminar alguna de ellas.

Nombre: Registrar-modificar actividad de atención.

Breve Descripción: Esta funcionalidad permite registrar los datos de una actividad de atención, los participantes y los responsables. Para modificar los detalles de una actividad registrada debe seleccionarse la misma.

Nombre: Consultar detalles de actividad de atención.

Breve Descripción: El objetivo de esta funcionalidad es mostrar los detalles de la actividad de atención seleccionada previamente.

Módulo Administración de Casos

Nombre: Consultar casos no asignados a Delegado de Prueba.

Breve Descripción: Se mostraran todos los casos que no tienen asignado un Delegado de Prueba. Los casos aparecerán ordenados ascendentemente por la fecha de ingreso de estos en la sede .

Nombre: Asignar nuevos casos a Delegado de Prueba.
Breve Descripción: Se registra el Delegado de Prueba para el caso o los casos seleccionados previamente. De la asignación se registrará además la fecha y la observación asociada, si es de interés del usuario.

Nombre: Consultar casos por Delegado de Prueba.
Breve Descripción: Se mostrarán todos los Delegados de Prueba y el total de casos que atiende en el momento.

Nombre: Reasignar casos de Delegado de Prueba.
Breve Descripción: Se le reasigna un caso al delegado de prueba, se registrará la fecha en la que uno o varios casos de un Delegado de Prueba son reasignados a otro, incluyendo el motivo por el cual se reasignan.

Nombre: Desvincular casos de Delegado de Prueba.
Breve Descripción: Se registra la desvinculación de uno o varios casos del Delegado de Prueba que lo(s) atiende hasta el momento.

Nombre: Gestionar Delegados de Prueba.
Breve Descripción: Se podrá registrar cuáles funcionarios del centro harán funciones de Delegado de Prueba para la atención y supervisión de los casos.

Conclusiones

En el capítulo se realizó una descripción de la arquitectura del SIGEP. Se representaron las diferentes capas que la componen y la relación entre ellas. Se describieron las funcionalidades definidas para los módulos Administración de Casos y Actividades de Atención, lo que posibilita una mejor comprensión de la solución dada al problema durante el diseño y la implementación.

CAPÍTULO 3: DISEÑO E IMPLEMENTACION DEL SISTEMA.

INTRODUCCIÓN

En este capítulo se muestra el diseño y la implementación de los módulos Administración de Casos y Actividades de Atención a partir de las funcionalidades definidas por el equipo de analistas del proyecto. Se representa una muestra de los diagramas de clases del diseño e implementación de los métodos más importantes de estos módulos teniendo en cuenta la secuencia de actividades y la metodología definida para el desarrollo de estos módulos del SIGEP.

3.1 Modelado del Dominio.

El diseño del dominio constituye la entrada principal a las restantes actividades de diseño. En este punto se identifican las entidades que serán gestionadas por la capa de negocio, persistidas o recuperadas por la capa de acceso a datos y mostradas por la capa de presentación. Se identifican los nomencladores. La definición del dominio, en específico de las entidades persistentes sirve como una primera aproximación al diseño definitivo del modelo de datos. En la Figura 3.1 y 3.2 se muestra el diagrama de clases persistentes de los módulos Actividades de Atención y Administración de Casos respectivamente.

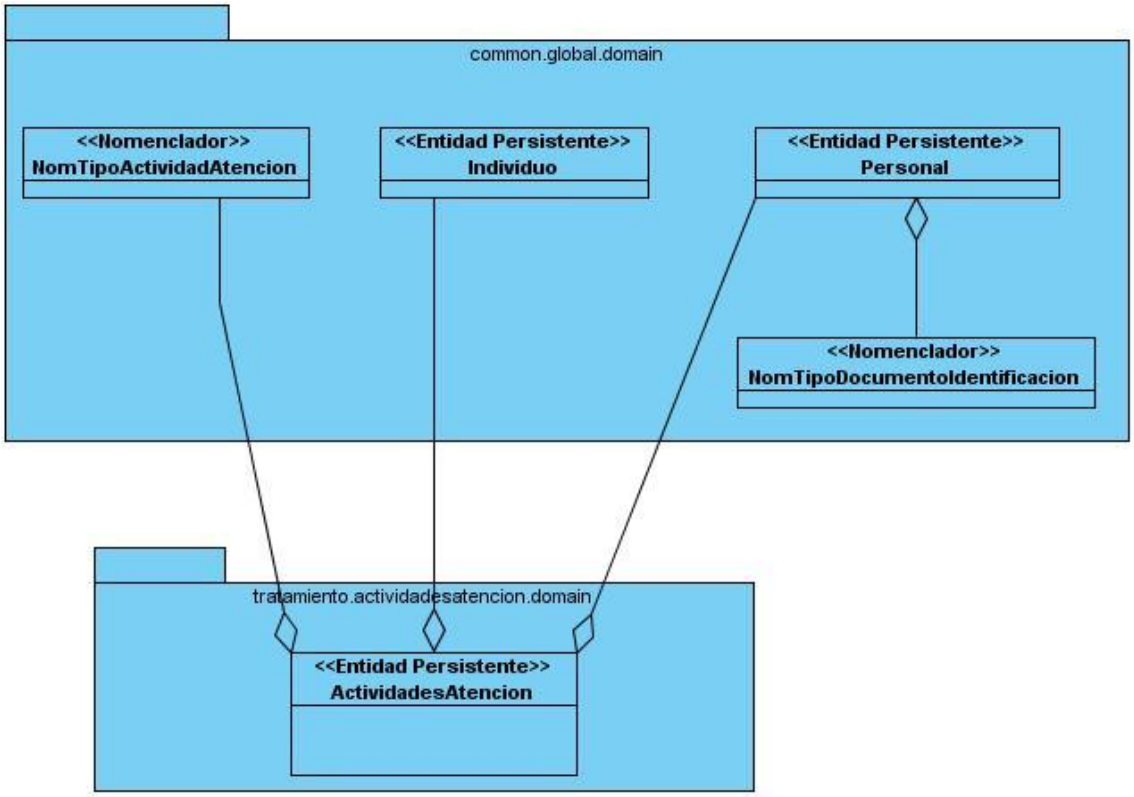


Figura 3.1 Diagrama de dominio del módulo Actividades de Atención.

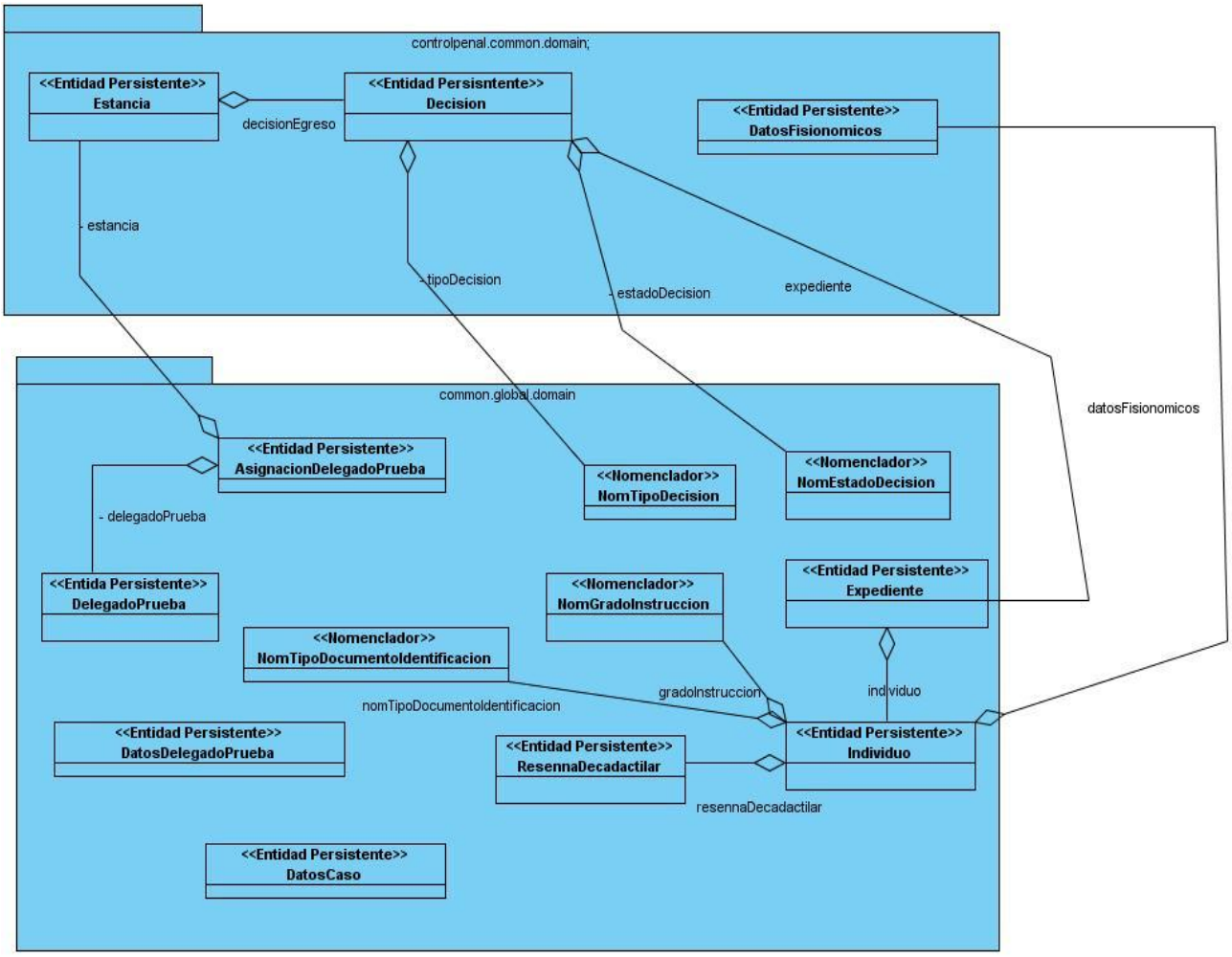


Figura 3.2 Diagrama de dominio del módulo Administración de Casos.

3.2 Diseño del modelo de datos.

Al definir las entidades que van a persistir en la aplicación, se lleva a cabo la realización del modelo de datos y las relaciones entre las entidades lo cual sirve como punto de partida para el diseño posterior de la base de datos. El modelo de datos puede presentar transformaciones durante el avance de las actividades de diseño ya que pueden surgir nuevas clases que sean necesarias a persistir.

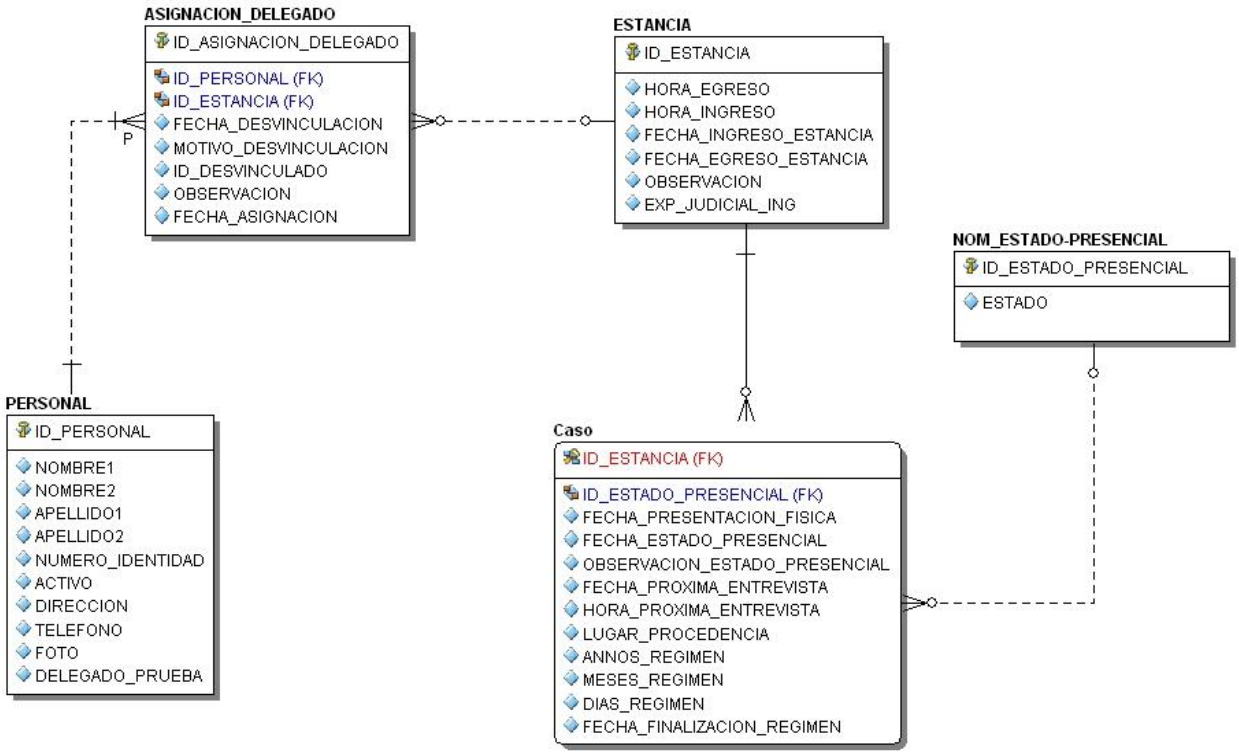


Figura 3.3 Modelo entidad relación del módulo Administración de casos.

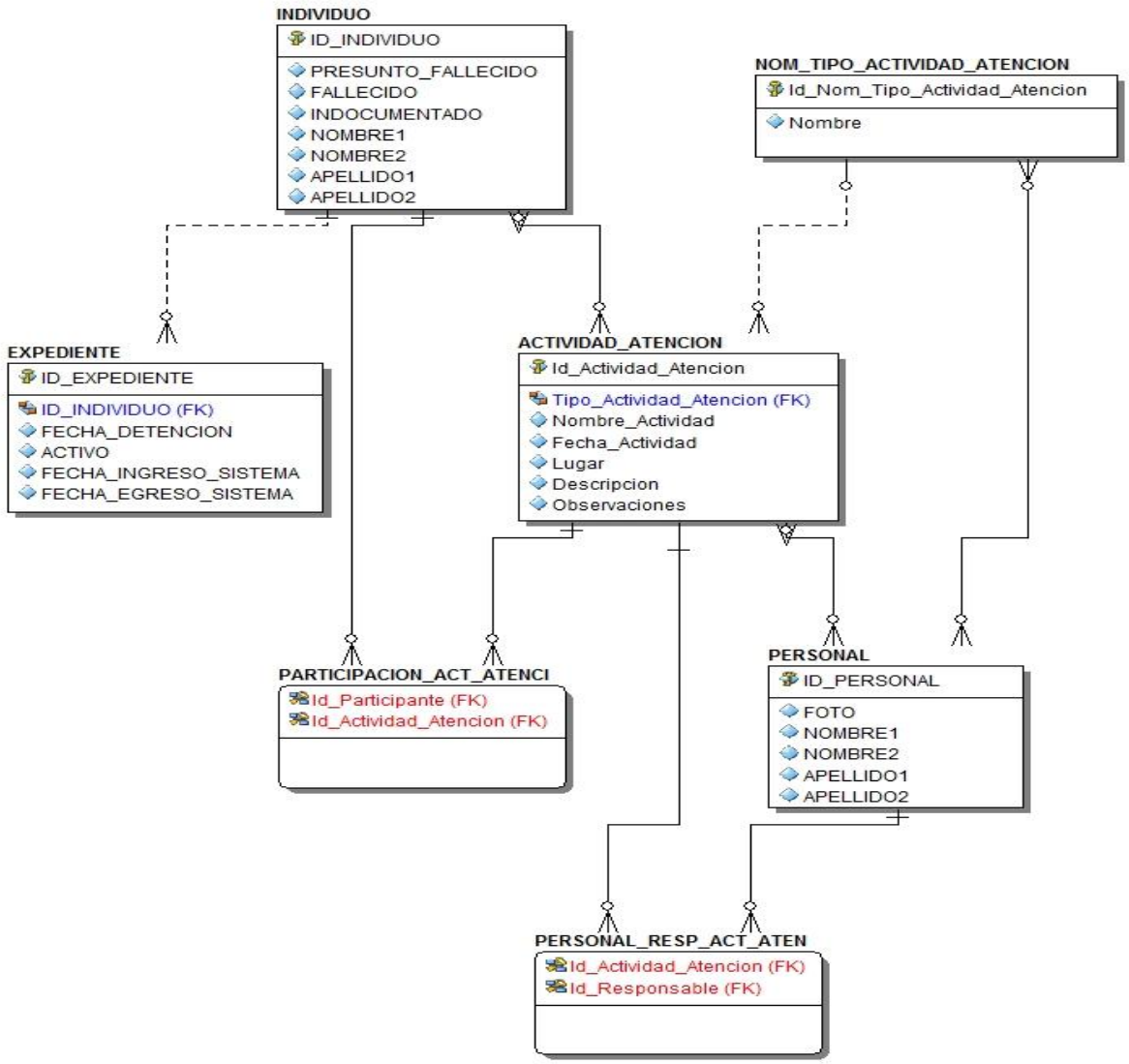


Figura 3.4 Modelo entidad relación del módulo Actividades de Atención.

3.3 Diseño de los módulos.

A continuación por cada funcionalidad definida se describen los elementos más significativos del diseño de la misma, así como los diagramas de clases donde se expresan las relaciones entre las entidades.

- **Módulo Administración de casos.**

Consultar casos no asignados a Delegado de Prueba.

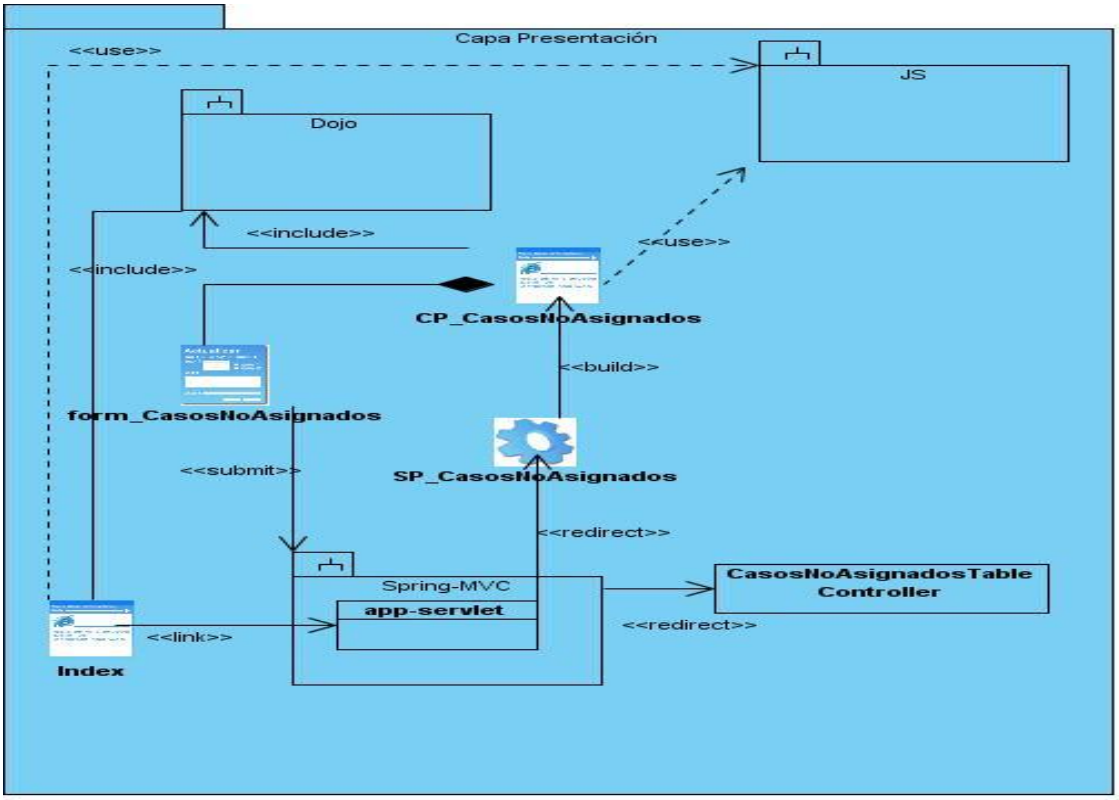


Figura 3.5: CU: Consultar casos no asignados a Delegado de Prueba (Capa de Presentación).

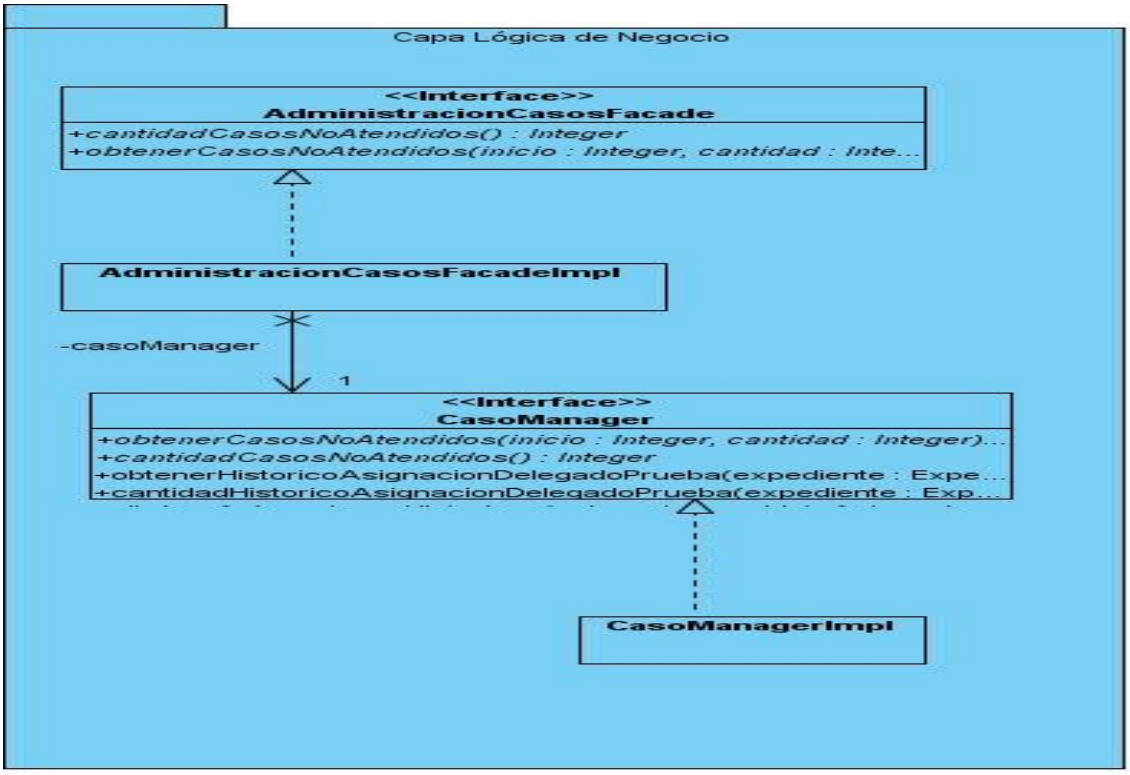


Figura 3.6: CU: Consultar casos no asignados a Delegado de Prueba (Capa Lógica del Negocio).

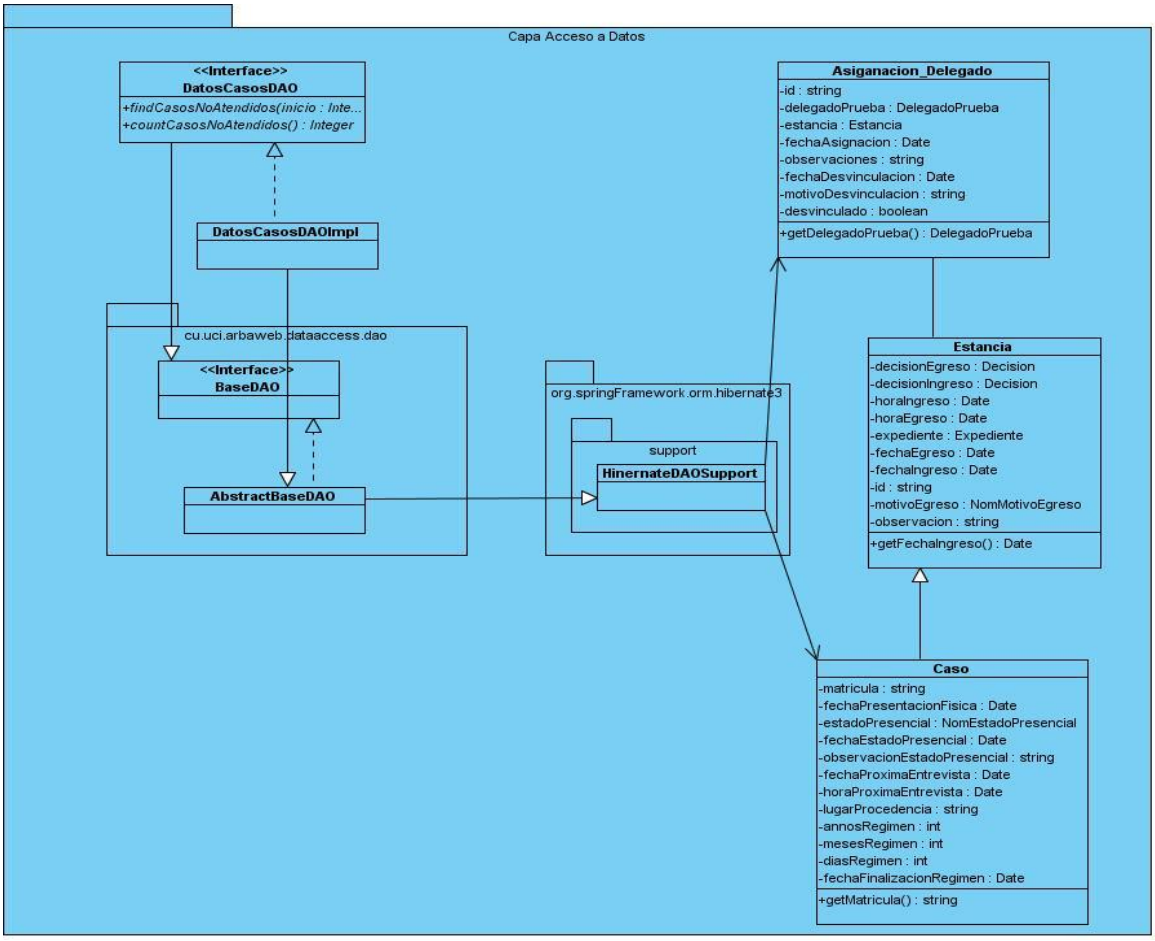


Figura 3.7: CU: Consultar casos no asignados a Delegado de Prueba (Capa de Acceso a Datos).

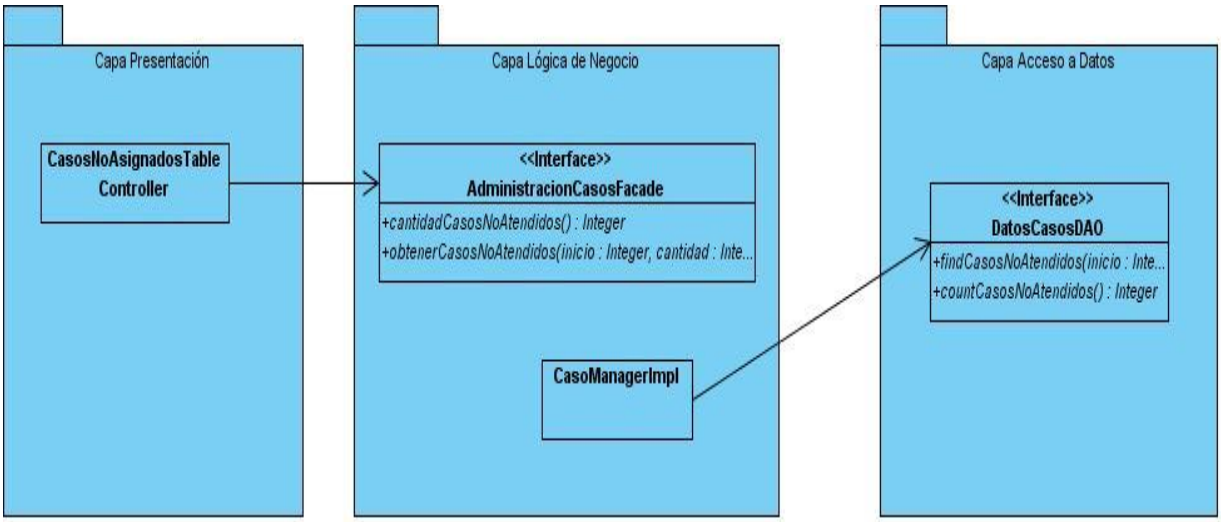


Figura 3.8: CU: Consultar casos no asignados a Delegado de Prueba (Relaciones entre capas).

- **Módulo Actividades de Atención.**

  Consultar detalles de actividad de atención.

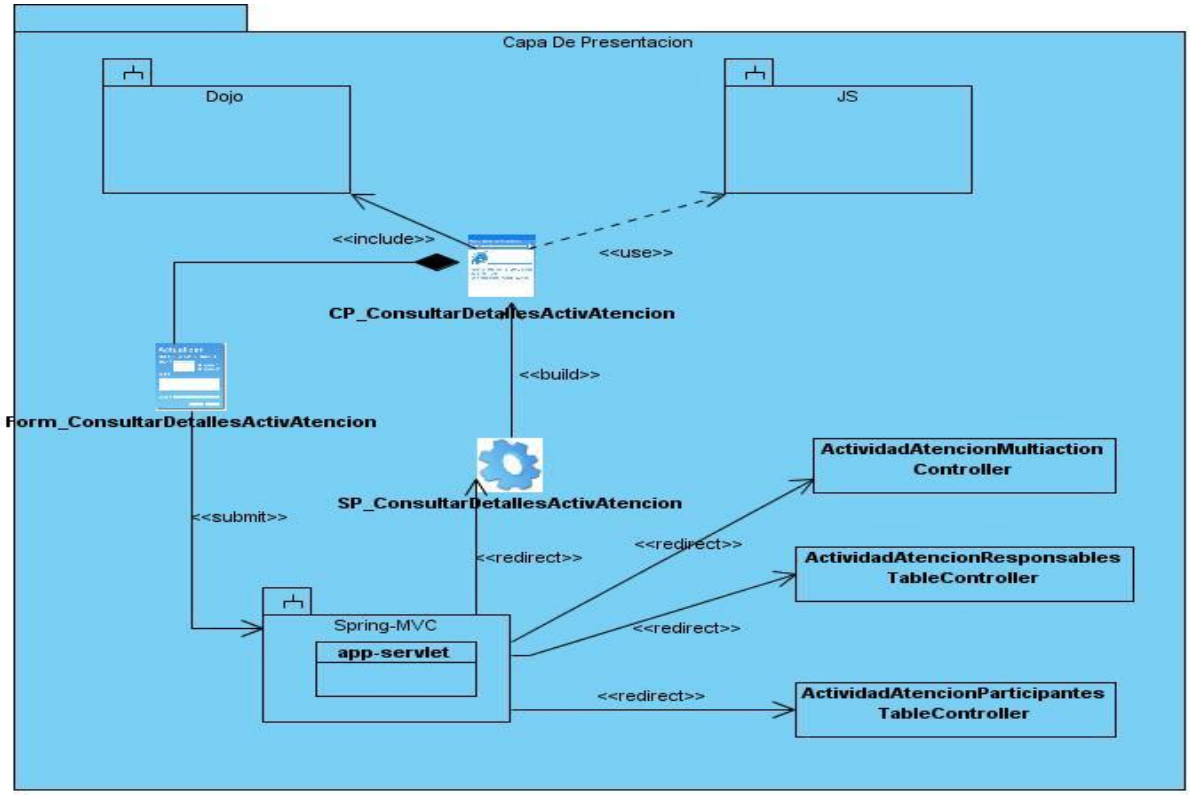


Figura 3.9: CU: Consultar detalles de actividad de atención (Capa de Presentación).

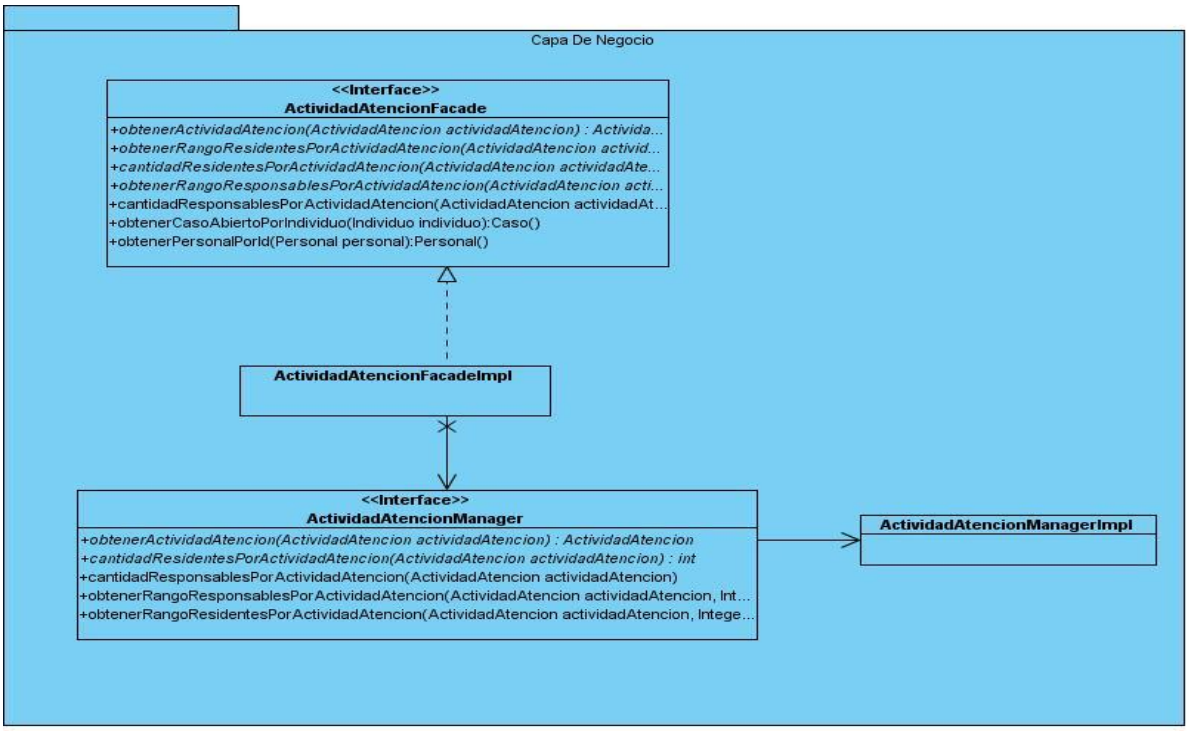


Figura 3.10: CU: Consultar detalles de actividad de atención (Capa Lógica del Negocio).

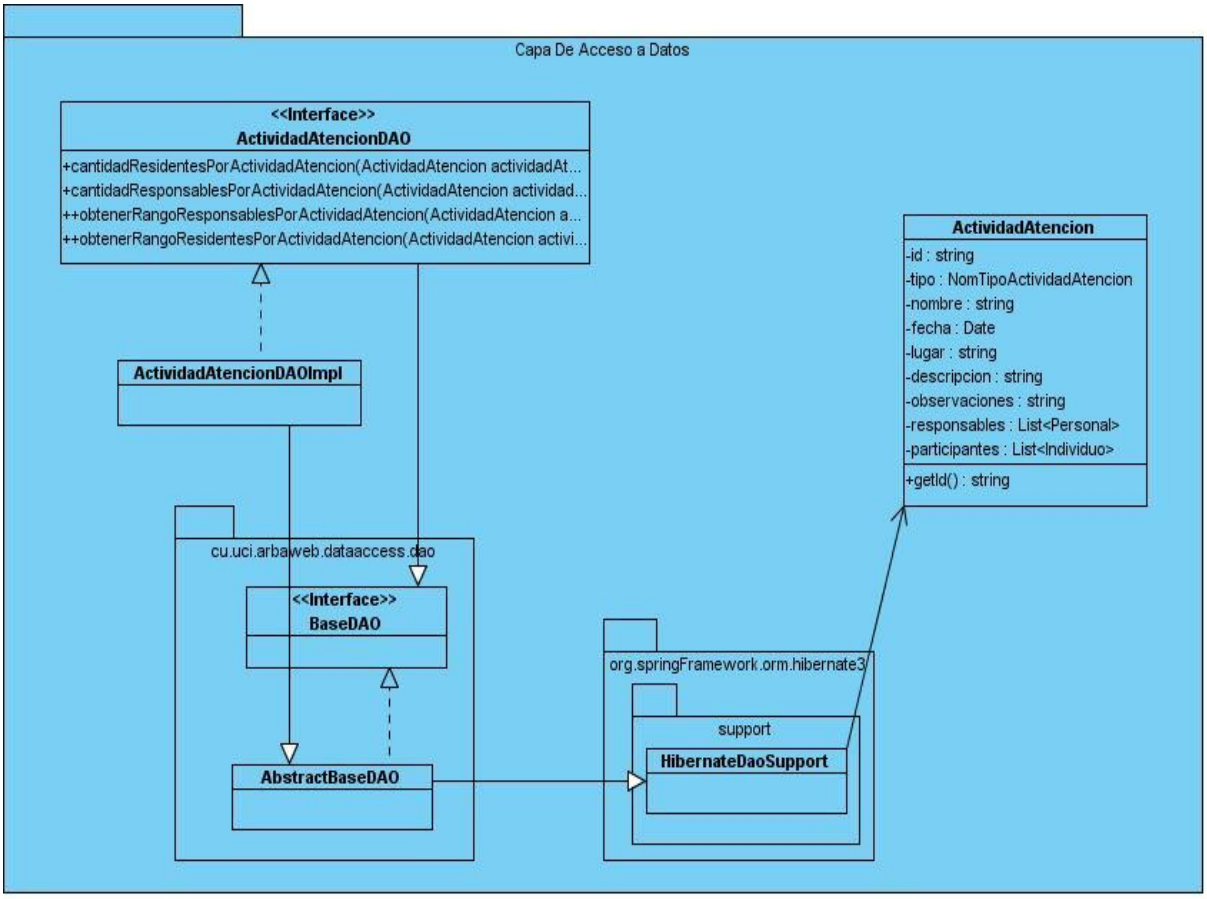


Figura 3.11: CU: Consultar detalles de actividad de atención (Capa de Acceso a Datos).

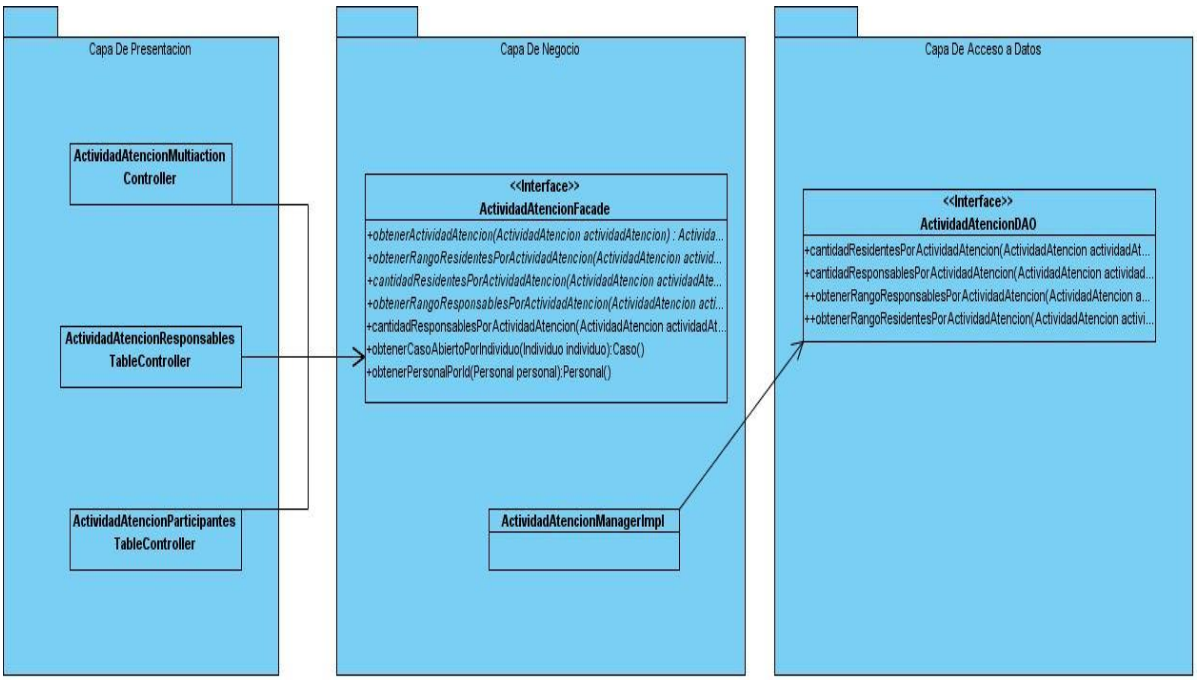


Figura 3.12: CU: Consultar detalles de actividad de atención (Relaciones entre capas).

3.4 Implementación.

3.4.1 Modelo de Implementación.

El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe como los elementos de diseño se implementan en componentes. Dicho modelo se considera el artefacto más significativo del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes. [23]

3.4.2 Diagrama de Componente.

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros.

A continuación los diagramas de componentes de los módulos Administración de casos y Actividades de Atención, además de fragmentos de código fuente, teniendo en cuenta las actividades de implementación definidas para el SIGEP.

- ✓ **Módulo Administración de Casos.**

Consultar casos no asignados a Delegado de Prueba.

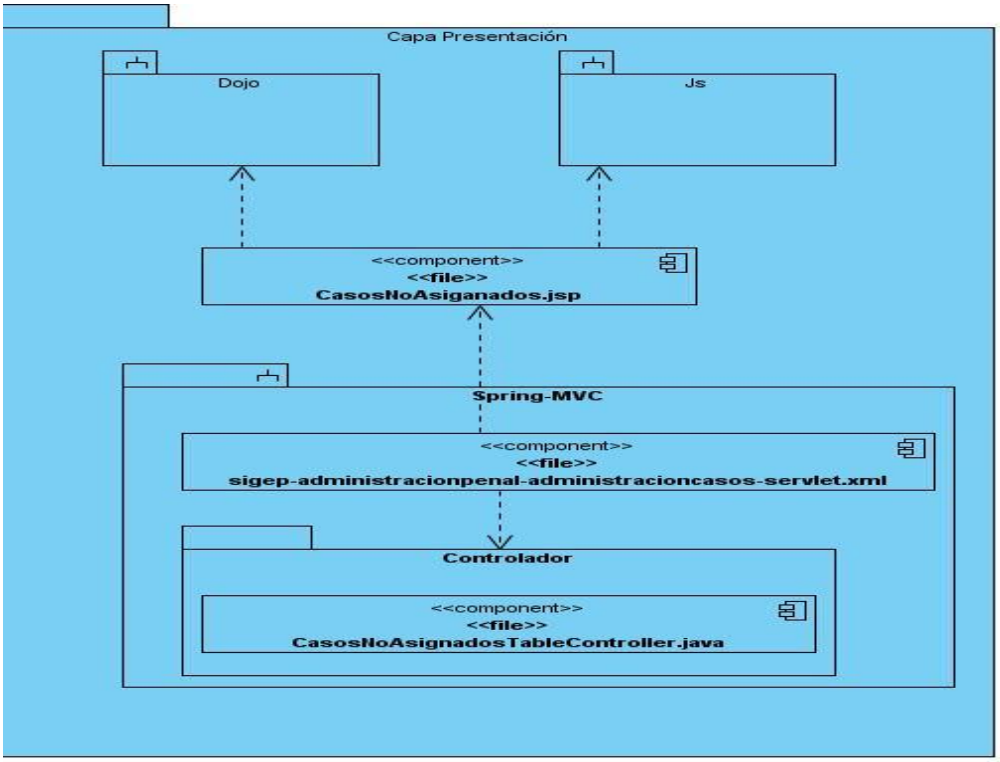


Figura 3.13: CU: Consultar casos no asignados a Delegado de Prueba (Capa de Presentación).

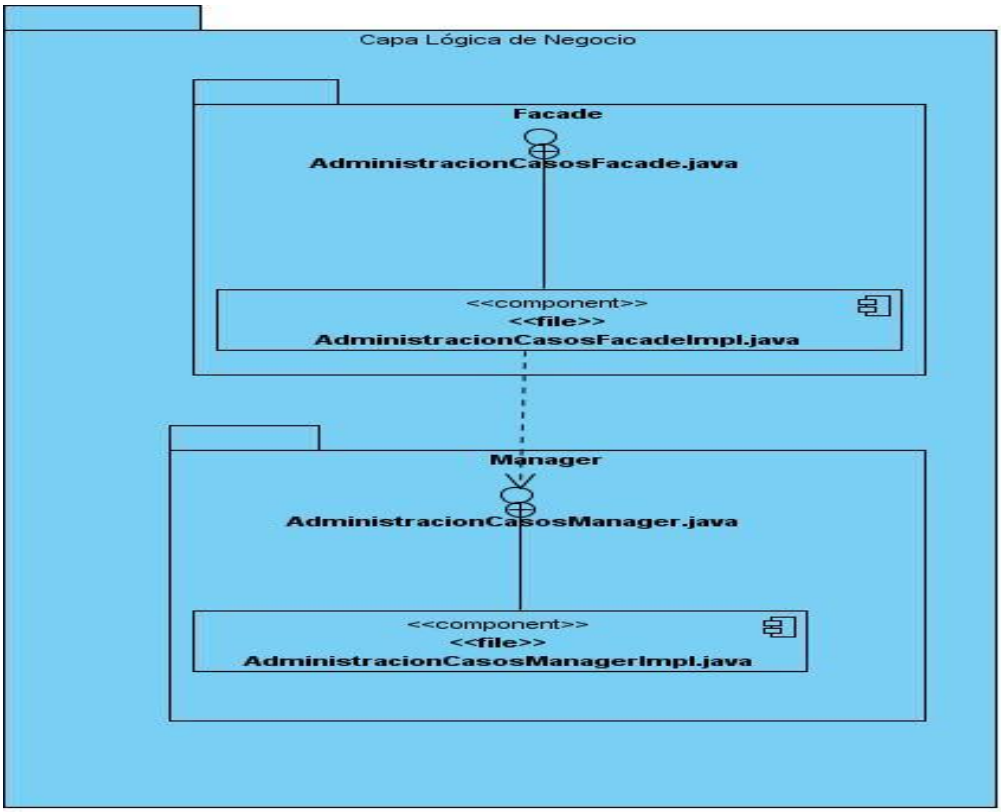


Figura 3.14: CU: Consultar casos no asignados a Delegado de Prueba (Capa Lógica del Negocio).

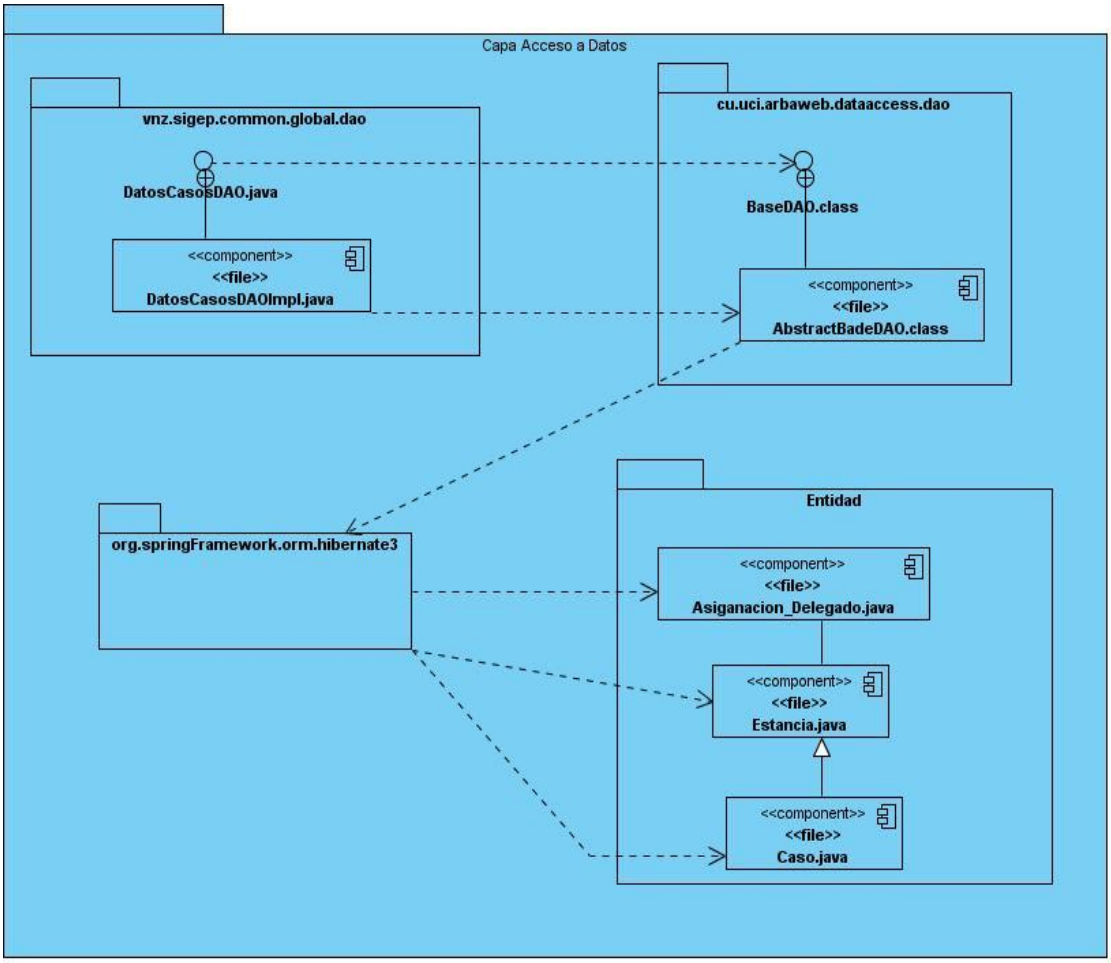


Figura 3.15: CU: Consultar casos no asignados a Delegado de Prueba (Capa de Acceso a Datos).

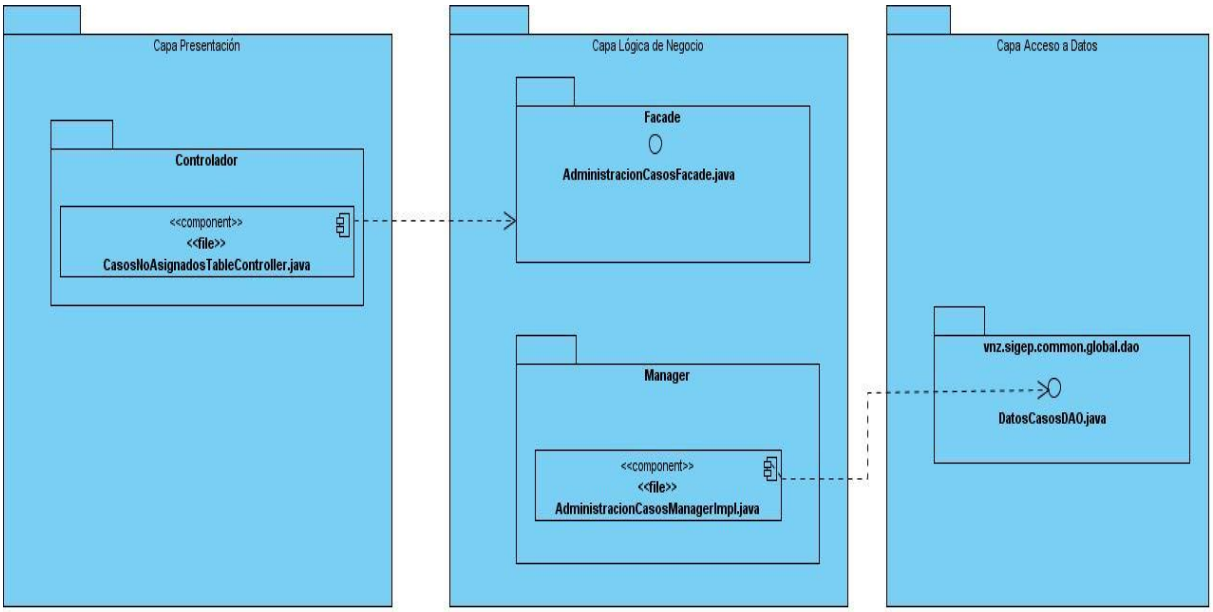


Figura 3.16: CU: Consultar casos no asignados a Delegado de Prueba (Relaciones entre las capas).

- ✓ **Módulo Actividades de Atención.**

Consultar detalles de actividad de atención.

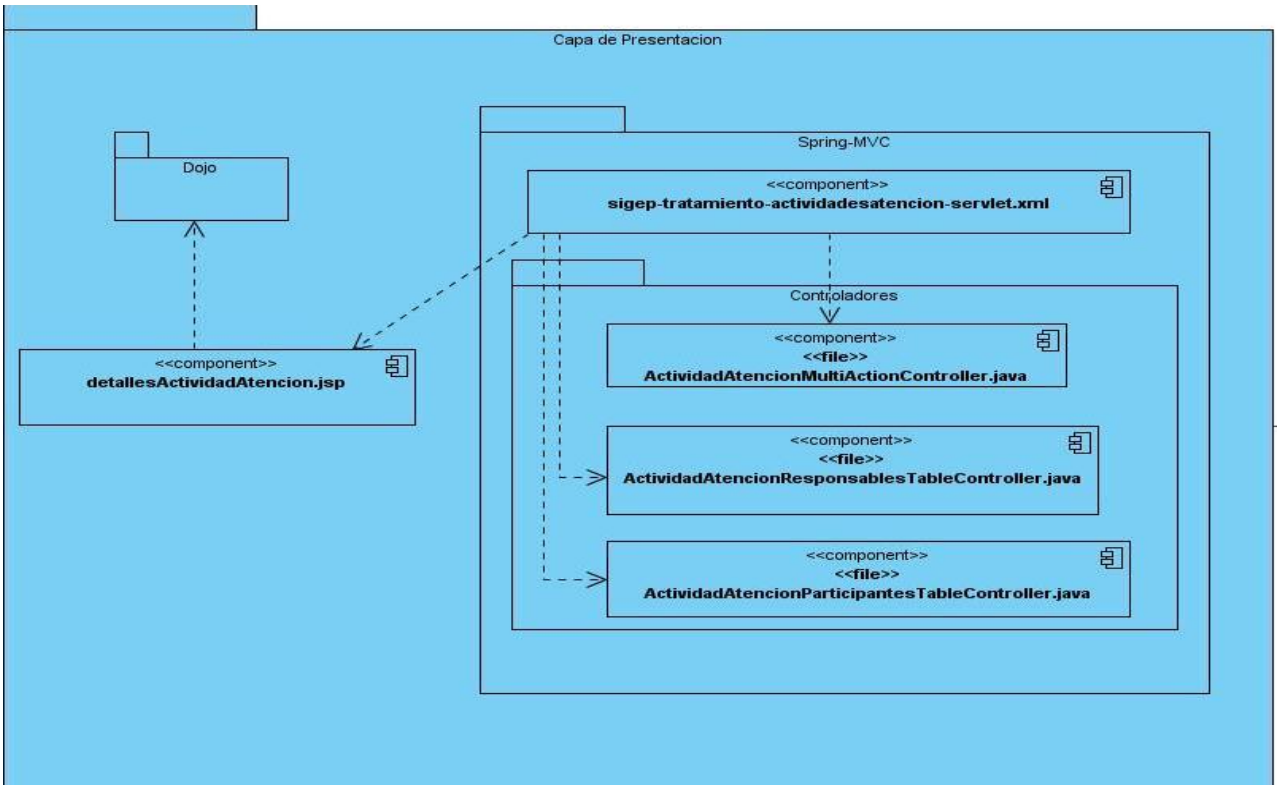


Figura 3.17: CU: Consultar detalles de actividad de atención (Capa de Presentación).

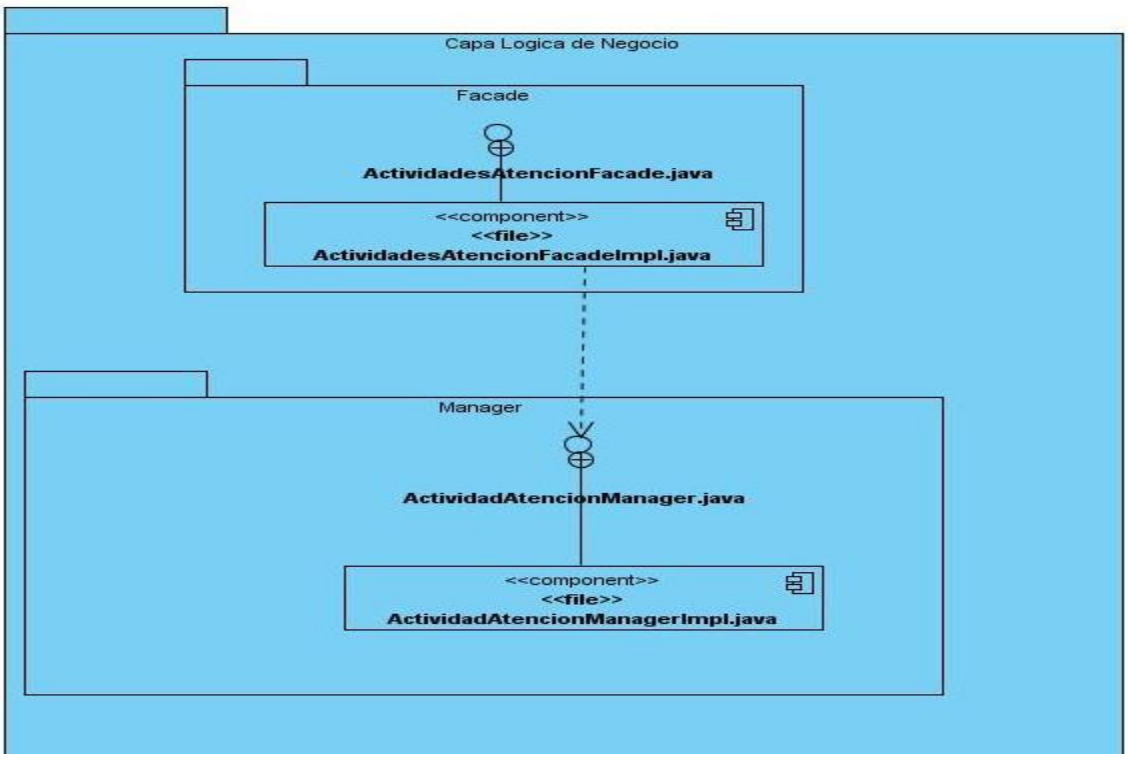


Figura 3.18: CU: Consultar detalles de actividad de atención (Capa Lógica de Negocio).

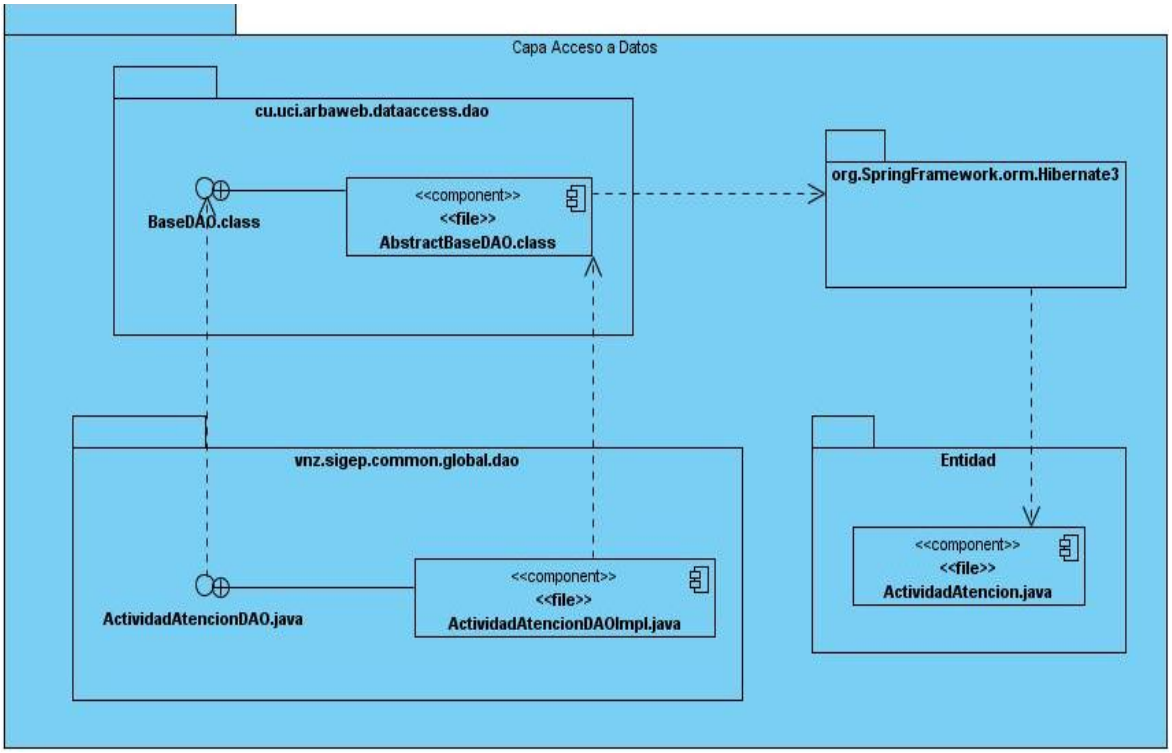


Figura 3.19: CU: Consultar detalles de actividad de atención (Capa de Acceso a Datos).

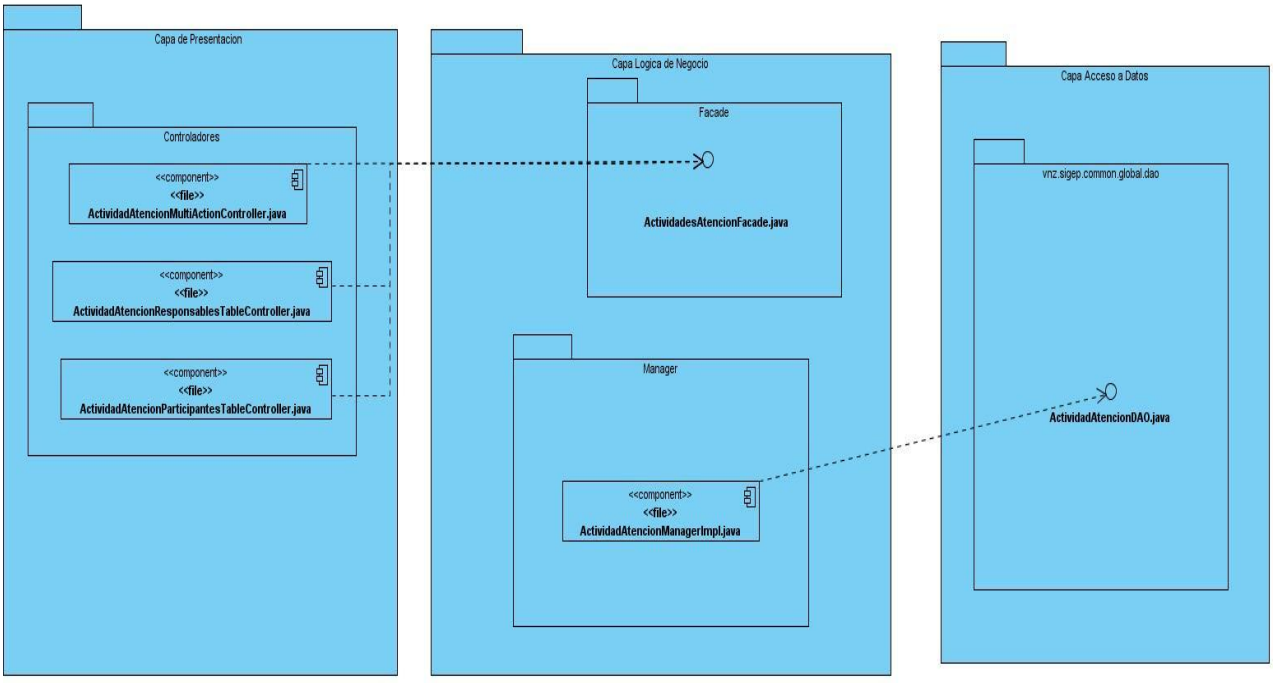


Figura 3.20: CU: Consultar detalles de actividad de atención (Relaciones entre capas).

3.5 Aspectos relevantes en el desarrollo del módulo Administración de Casos.

El módulo Administración de Casos tiene varias funcionalidades que son comunes con otros módulos, como solución se crearon en el paquete common del SIGEP un grupo de clases restringidas y abstractas. Estas clases encapsulan las operaciones comunes, lo que hace más flexible y reutilizable el código en las clases del dominio, de acceso a datos y de negocio.

3.6 Transacciones a Nivel de Negocio.

Las transacciones son un elemento importante en las aplicaciones de gestión empresarial, evitando inconsistencias en los recursos y datos almacenados. Con este fin se aprovechó las facilidades que brinda el framework Spring para la gestión de transacciones. Las declaraciones de las transacciones se realizan en ficheros XML, de forma tal que no es necesario escribir ningún código asociado al manejo de transacciones en los métodos que contienen la lógica de negocio. El framework Spring define un conjunto de etiquetas para manejar las transacciones mediante aspectos entre las que se encuentra:

• <aop:advisor> Contiene dos propiedades fundamentales *pointcut* (una expresión regular que indica el punto de corte donde se aplicará el aspecto, generalmente uno o varios métodos) y *advice-ref* (se declara cómo se va a aplicar la transacción). *(12)*

A continuación se muestra un ejemplo donde se aplica lo antes explicado:

```
<aop:config>
<aop:advisorpointcut="execution(* *..*ActividadAtencionManagerImpl.*(..))"
        advice-ref="defaulMgrAdvice" />
</aop:config>
        <tx:advice id="defaulMgrAdvice">
                <tx:attributes>
                        <tx:method name="*" rollback-for="java.lang.Exception" />
                </tx:attributes>
        </tx:advice>
```

La expresión *execution* significa: cuando el método sea ejecutado. La expresión entre paréntesis representa los métodos a los que se aplicará la transacción, en este caso:

* *..* **ActividadAtencionManagerImpl**.*(..)*. El primer asterisco significa cualquier tipo de retorno; la expresión *..* **ActividadAtencionManagerImpl** significa cualquier clase cuyo

nombre termine en **ActividadAtencionManagerImpl** y la expresión *.*(..)* significa cualquier método con cualesquiera parámetros.

Con el atributo name se indica el nombre del método al cual se le aplicará rollback al ser lanzada una determinada excepción y con el atributo rollback-for se definen las excepciones que, en caso de ser lanzadas, se desea que fallen las transacciones, en este caso específico a cualquier excepción lanzada fallará la transacción.

3.7 Jerarquía de Clases.

Como se explicó en el capítulo anterior, en el diseño del dominio del módulo se definió una jerarquía de clases, donde la clase Caso extiende de Estancia. El siguiente fragmento de código muestra la clase padre (*Estancia*) con los atributos comunes.

```
public class Estancia extends BaseObject {
 protected Centro centro;
      // centro penitenciario
protected Decision decisionEgreso;
     // Entidad que contiene los datos de una decisión emitida por
cualquier  instancia del poder judicial
protected Decision decisionIngreso;
     // Entidad que contiene los datos de una decisión emitida por
cualquier * instancia del poder judicial
protected Date horaIngreso;
    // hora de ingreso
protected Date horaEgreso;
    // hora de egreso
protected Expediente expediente;
   //Entidad de dominio que define los datos de un expediente de
un individuo en el sistema penitenciario
protected Date fechaEgreso;
   // fecha de egreso
protected Date fechaIngreso;
  // fecha de ingreso
protected String id;
  //identificador
protected NomMotivoEgreso motivoEgreso;
 //Nomenclador de los motivos por los cuales puede egresar un
individuo de un establecimiento penitenciario
```

```
protected String observacion;
    //Detalles que pueda destacar }
```

La herencia en lenguaje Java se evidencia mediante la palabra extends más el nombre de la clase padre.

A continuación se muestra como se declaró la herencia en la clase Caso:

```
Public class Caso extends Estancia {…}
```

3.8 Mapeo de Objetos

El framework Hibernate permite asociar una tabla de la base de datos a un objeto Java, relacionando cada campo de la tabla con un atributo del objeto. Esta configuración se hace en ficheros XML; a continuación se muestra un ejemplo:

```
<hibernate-mapping
package="vnz.sigep.tratamiento.actividadesatencion.domain">
<class name="ActividadAtencion" table="ACTIVIDAD_ATENCION">
<id name="id" type="string">
 <column name="ID_ACTIVIDAD_ATENCION" length="20" />
 <generator
class="vnz.sigep.common.global.util.dataaccess.id.PrefixPersistentGenerator">
 <param name="sequence">SEQ_ACTIVIDAD_ATENCION</param>
 <param name="maxlo">1000</param>
 </generator>
</id>
                <property name="descripcion" type="string">
                        <column name="DESCRIPCION"></column>
                </property>
                <property name="fecha" type="date">
                        <column name="FECHA_ACTIVIDAD"></column>
                </property>
                <property name="lugar" type="string">
                        <column name="LUGAR"></column>
                </property>
                <property name="nombre" type="string">
                        <column name="NOMBRE_ACTIVIDAD"></column>
                </property>
```

```
                <property name="observaciones" type="string">
                        <column name="OBSERVACIONES"></column>
                </property>
<many-to-one name="tipo"
class="vnz.sigep.common.global.domain.NomTipoActividadAtencion"
column="TIPO_ACTIVIDAD_ATENCION" not-null="true" />
<bag name="participantes" table="PARTICIPACION_ACT_ATENCION">
<key>
<column name="ID_ACTIVIDAD_ATENCION" length="20" not-null="true" />
</key>
<many-to-many                 class="vnz.sigep.common.global.domain.Individuo"
column="ID_PARTICIPANTE"/>
</bag>
<bag name="responsables" table="PERSONAL_RESP_ACT_ATENCION">
<key>
<column name="ID_ACTIVIDAD_ATENCION" length="20" not-null="true" />
</key>
<many-to-many                 class="vnz.sigep.common.global.domain.Personal"
column="ID_RESPONSABLE"/>
</bag>
</class>
</hibernate-mapping>
```

En el código anterior se observan etiquetas y algunos de los atributos que estas contienen. El atributo package nos dice cuál es el paquete donde se encuentra la clase que va a ser mapeada. En la etiqueta class se especifican los nombres de la clase y de la tabla de la base de datos. Dentro de la etiqueta id se encuentra el nombre del atributo de la clase que va a almacenar el identificador, el tipo de dato, la columna de la tabla con la cual se hace corresponder y el generador del identificador (generator), en este caso, es la clase PrefixPersistentGenerator. En la etiqueta property se configura un atributo de la clase declarando el nombre del mismo, el tipo de dato y la columna de la tabla con la cual va a ser mapeado. Por cada atributo hay que generar una etiqueta property. La etiqueta many-to-one significa que hay una relación de uno a muchos, donde se especifica el nombre del atributo, la clase de este atributo (el tipo de dato) y la columna de la tabla con la cual se mapea.

3.9 Descripción de las entidades de dominio significativas.

ActividadAtencion

Descripción	Clase ubicada en el paquete domain del subsistema, presenta las funcionalidades y atributos comunes para crear una Actividad de Atención.	
Atributos	**Visibilidad**	**Tipo**
id	private	String
tipo	private	NomTipoActividadAtencion
nombre	private	String
fecha	private	Date
lugar	private	String
descripcion	private	String
observaciones	private	String
responsables	private	List<Personal>
participantes	private	List<Individuo>

Individuo

Descripción	Entidad que refleja los participantes de la Actividad de Atención, clase ubicada en el paquete domain del common global, ya que es utilizada en otros módulos.	
Atributos	**Visibilidad**	**Tipo**
Nombre1	private	String
Nombre2	private	String
apellido1	private	String
Apellido2	private	String
numeroIdentidad	private	String
Alias	private	String

Apodo	private	String
banda	private	String
datosFisionomicos	private	DatosFisionomicos
direcciones	private	List<IndividuoDireccion>
expedientes	private	List<Expediente>
fechaNacimiento	private	Date
id	private	String

3.10 Descripción de las clases significativas de la capa de negocio.

ActividadesAtencionFacade

Descripción	Interfaz ubicada en el paquete facade del subsistema Actividad de Atención, facade que tiene todas las funcionalidades de este módulo.		
Nombre	**Visibilidad**	**Tipo**	**Parámetros**
persistirActividadAtencion	public	void	List<Individuo> listaIndividuosAdicionados, List<Individuo> listaIndividuosEliminados, List<Personal> listaPersonalAdicionados, List<Personal> listaPersonalEliminados
obtenerRangoActividadesAtencion	public	List<ActividadAtencion>	Integer inicio, Integer cantidadMaxima, List<PropertyFilter> filtros, List<OrderCriteria> ordenar
obtenerActividadAtencion	public	ActividadAtencion	ActividadAtencion actividadAtencion
eliminarActividadAtencion	public	void	List<ActividadAtencion> actividadesAtencion
obtenerRangoResponsablesPorActividadAtencion	public	List<Personal>	ActividadAtencion actividadAtencion, Integer

			inicio,Integer cantidadMaxima, List<String> idsExcluidos
cantidadResponsablesPorActividadAtencion	public	Integer	ActividadAtencion actividadAtencion, List<String> idsExcluidos
obtenerCantidadActividadesAtencionPorIndividuo	public	Integer	Individuo individuo, List<PropertyFilter> filtros
obtenerCasoAbiertoPorIndividuo	public	Caso	Individuo individuo
obtenerPersonalPorId	public	Personal	Personal personal

ActividadAtencionManager

Descripción	Interfaz del manager que gestiona los procesos de negocio del módulo Actividad de Atención.		
Nombre	**Visibilidad**	**Tipo**	**Parámetros**
obtenerActividadAtencion	public	ActividadAtencion	ActividadAtencion actividadAtencion
eliminarActividadAtencion	public	void	List<ActividadAtencion> actividadesAtencion
cantidadResidentesPorActividadAtencion	public	Integer	ActividadAtencion actividadAtencion, List<String> idsExcluidos
cantidadResponsablesPorActividadAtencion	public	Integer	ActividadAtencion actividadAtencion, List<String> idsExcluidos
persistirActividadAtencion	public	void	List<Individuo> listaIndividuosAdicionados, List<Individuo>

			listaIndividuosEliminados, List<Personal> listaPersonalAdicionados, List<Personal> listaPersonalEliminados
obtenerCantidadActividadesAtencionPorIndividuo	public	Integer	Individuo individuo, List<PropertyFilter> filtros
obtenerRangoActividadesAtencionPorIndividuo	public	List<ActividadAtencion>	Individuo individuo, Integer inicio, Integer cantidadMaxima, List<PropertyFilter> filtros, List<OrderCriteria> ordenar

3.11 Descripción de las clases significativas de la capa de acceso a datos.

ActividadAtencionDAO

Descripción	Interfaz DAO ubicada en el paquete dao del subsistema, presenta las funcionalidades para una actividad de Atención.
Padre	BaseDAO

3.12 Implementación de las entidades del dominio

Las entidades del dominio suelen tener muy poco comportamiento. Los métodos equals(), hasCode() y toString() deben ser implementados en la mayoría de los casos, ya que son frecuentemente utilizados. En este paso se debe refinar la definición de todos los atributos de las entidades de manera que queden listas en un buen por ciento para implementaciones reusables (11). A continuación se muestra la implementación de los métodos mencionados anteriormente en la clase del dominio.

```
Public class ActividadAtencion extends BaseObject {
 public boolean equals(Object o) {
```

```
            if (!(o instanceof ActividadAtencion))
                return false;

            ActividadAtencion aa = (ActividadAtencion) o;
            return new EqualsBuilder().append(getId(), aa.getId()).isEquals();
        }

        @Override
        public int hashCode() {
            return new HashCodeBuilder(7, 11).append(id).append(nombre)
                    .toHashCode();
        }
        @Override
        public String toString() {
            return new ToStringBuilder(this,
ToStringStyle.SIMPLE_STYLE).append(
                    "ID", id).append("Tipo", tipo).append("Nombre",
nombre).append(
                    "Fecha", fecha).toString();
}
```

Conclusiones:

En este capítulo se presentaron los elementos más significativos en la implementación del módulo Administración de Casos y Actividades de Atención. Se siguieron los lineamientos de la arquitectura del sistema y el diseño definido para el cumplimiento de las funcionalidades pactadas con el cliente. Además se mostraron el diseño y la implementación de los módulos a través de diagramas de clase y fragmentos de código fuente, teniendo en cuenta las actividades de diseño e implementación definidas para el SIGEP.

Capítulo 4: PRUEBAS.

4.1 Introducción.

Después de terminadas las tareas asociadas al diseño e implementación de los módulos Administración de Casos y Actividades de Atención para comprobar el correcto funcionamiento de los módulos implementados se realizó un conjunto de pruebas a los mismos. El objetivo de estas pruebas es encontrar defectos en el software y estas tienen éxito si se descubre un defecto.

4.2 Pruebas de software.

Las pruebas de software son procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad. Algunas definiciones de esta son:

- La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado.

- La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

4.3 Niveles de Prueba.

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Cuando se evalúa dinámicamente un sistema de software se debe comenzar por los componentes más pequeños y más simples y avanzar progresivamente hasta probar todo el software en su totalidad. Los niveles que se pueden distinguir son:

Pruebas al Sistema:

Son similares a las pruebas de caja negra, solo que éstas buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. A éste tipo de pruebas donde se

estudia el producto completo se les llama Prueba de Sistema. También se realizan estas pruebas a la documentación del sistema.

Pruebas de Desarrollador:

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para las pruebas de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

Pruebas de unidad:

Se enfocan en un programa o un componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Es la prueba enfocada a los elementos testeables más pequeño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. El diseño de casos de prueba de una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente. (13)

4.4. Técnicas de Pruebas.

Cada nivel de prueba engloba una técnica de prueba específica según los atributos de calidad que se deseen verificar con las pruebas al software.

Pruebas de Funcionalidad.

Algunos ejemplos de prueba de funcionalidad son:

Pruebas Funcionales:

Objetivo: Este tipo de pruebas examina si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones de diseño. En ellas se debe verificar si el sistema lleva a cabo correctamente todas las funciones requeridas, se debe verificar la validación de los datos. Estas pruebas deben estar enfocadas a tareas, a límites del sistema, a condiciones planeadas de error y de exploración. Para estas pruebas usamos los esquemas de pruebas de caja negra ya que nos interesa saber si funciona o no, independientemente de la forma en que lo haga.

Caso de prueba: Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previo a la realización de las pruebas funcionales de la aplicación. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, para hacer más fructífera la ejecución de las pruebas.
Los casos de prueba pertenecientes a los módulos Administración de Casos y Actividades de Atención dentro de los subsistemas Administración de la Sede y Atención, Supervisión y Orientación respectivamente se pueden encontrar en el expediente del proyecto del SIGEP en su versión 2.1 de la Fase IV. Entre los cuales se encuentra el diseño del caso de prueba del CU Gestionar Delegado de prueba perteneciente al módulo Administración de Casos.

Método de Caja Negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Se centran principalmente en los requisitos funcionales del software. (14)

No Conformidades: La plantilla de No Conformidades recoge los errores que son detectados durante la revisión de la documentación del sistema. Se elabora un documento por cada revisión que se haga y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba. Además de estar inmerso en la planilla de diseño de casos de prueba, estas No Conformidades se van registrando en un documento aparte para luego enviarlo al equipo de desarrolladores.
En la siguiente tabla un resumen de los resultados de las No Conformidades de los módulos: Administración de Casos y Actividades de Atención, distinguiendo de ellas cuáles proceden, cuáles no proceden y las resueltas.

Módulo Administración de Casos.

Descripción	Tipo de petición	Estado
Cuando no existen casos para asignar el sistema no muestra el mensaje estipulado. “No existen Casos para realizar la	No conformidad.	No resuelta

Asignación".		
Cuando se reasigna un caso el sistema debe mostrar el mensaje "Reasignación registrada correctamente"	Solicitud de cambio.	No resuelta
El submenú dice Gestionar Delegado de prueba tratante, cuando debe decir Gestionar Delegado de Prueba.	Solicitud de cambio	No resuelta
Cuando no existen casos para desvincular el sistema no muestra el mensaje estipulado. "No existen Casos para realizar la Reasignación".	No conformidad.	Resuelta

Tabla 4.1 Descripción, tipo de observación y estado de las no conformidades.

Módulo Actividades de Atención.

Descripción	Tipo de petición	Estado
Cuando no se selecciona ningún participante el sistema debe mostrar el mensaje "Debe seleccionar al menos un participante".	No conformidad.	No resuelta
Cuando no se selecciona ningún responsable el sistema debe mostrar el mensaje "Debe seleccionar al menos un responsable".	No conformidad.	No resuelta

Tabla 4.2 Descripción, tipo de observación y estado de las no conformidades.

Pruebas de seguridad:

Se realizan para asegurar que los datos o el sistema solamente son accedidos por los actores deseados.

Pruebas de Confiabilidad:

La confiabilidad del software se refiere a la exactitud con la que una aplicación suministra, los servicios que se establecieron en las especificaciones originales sin que estas tengan errores. Se realizan para evaluar el rendimiento y está relacionado también con la obtención de resultados correctos y con el control de la detección de errores y de la recuperación para evitar que se produzcan errores. (15)

- **Prueba de estrés:**

Es una prueba de carga y performance basada en la funcionalidad del sistema bajo cargas pesadas, un gran número de repeticiones, manejo de grandes datos. Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible). Donde se somete el sistema a casos extremos, no esperados en situación normal, exagerando la cantidad de datos o la frecuencia de transacciones.

Pruebas de Rendimiento:

Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Se evalúa el número de transacciones por unidad de tiempo o el tiempo de respuesta a determinadas funcionalidades.

- **Prueba de carga:**

Está basada en las aplicaciones bajo cargas pesadas, generalmente usadas en sitios web y en servidores con gran cantidad de datos donde se determina en cuales puntos existen degradaciones del sistema.

Las pruebas de carga se realizaron mediante la herramienta JMeter, simulando varias peticiones de un número variable de usuarios.

Simulación de 10 usuarios con un periodo de subida de 30 segundos

La siguiente figura muestra que la prueba realizada tiene un margen de error. Esto se deduce de la columna representativa del tanto por ciento de errores para cada una de las peticiones asociadas a cada conjunto de muestras. El rendimiento muestra que para una simulación de 10 usuarios junto a un periodo de subida de 30 segundos el servidor es capaz de aceptar una media de 7.8 peticiones por segundo. Tiene un margen de error de 2.69%.

Label	# Sampl...	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
/sigep.crs/security/login.htm	10	42352	1959	92765	40690.59	50.00%	5.1/min	0.23	2822.8
/sigep.crs/security/captcha.jpeg	10	10811	818	95259	28153.38	40.00%	5.1/min	0.19	2254.6
/sigep.crs/common/global/servertime.htm	20	625	108	2012	721.59	20.00%	10.0/min	0.08	480.5
/sigep.crs/common/global/timezonepopup.htm	20	5332	132	96894	21014.85	15.00%	10.2/min	0.12	737.4
/sigep.crs/security/j_acegi_security_check	10	891	147	2011	760.97	30.00%	5.4/min	0.27	3121.9
/sigep.crs/	10	843	274	2015	766.35	30.00%	5.4/min	0.28	3118.4
/sigep.crs/common/global/welcome.htm	20	5186	65	95833	20802.83	10.00%	9.5/min	0.53	3428.8
/sigep.crs/common/global/registerClientOffset.htm	20	384	29	2012	575.20	10.00%	9.6/min	0.04	223.9
/sigep.crs/common/global/subSystemMenuAccess.htm	20	5047	30	96263	20930.15	5.00%	9.8/min	0.02	123.4
/sigep.crs/administracionpenal/index.htm	10	561	169	1910	557.90	10.00%	5.8/min	0.33	3428.8
/sigep.crs/common/global/rpc/jsonrpc.htm	30	3471	135	94968	16994.11	3.33%	17.7/min	0.03	90.0
/sigep.crs/common/global/isUsableResource.htm	60	177	30	696	94.92	0.00%	3.0/sec	0.07	23.0
/sigep.crs/administracion/seguridad/jsonrpc/jsonrpc.htm	10	258	89	497	96.73	0.00%	2.9/sec	0.06	23.0
/sigep.crs/seguridadcustodia/jsonrpc.htm	10	187	66	323	74.23	0.00%	3.2/sec	0.07	23.0
/sigep.crs/common/busqueda/criteriosRpc/jsonrpc.htm	10	282	73	625	171.82	0.00%	2.9/sec	0.07	23.0
/sigep.crs/tratamiento/jsonrpc.htm	10	308	148	688	195.13	0.00%	2.9/sec	0.06	23.0
/sigep.crs/administracionpenal/capacidades/jsonrpc/js...	10	260	90	523	111.90	0.00%	2.9/sec	0.07	23.0
/sigep.crs/administracionpenal/welcome.htm	20	221	80	619	141.60	0.00%	1.2/sec	0.03	23.0
/sigep.crs/common/global/hasAccessToResource.htm	40	173	90	427	53.77	0.00%	3.4/sec	0.08	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	168	78	333	49.72	0.00%	4.2/sec	0.09	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	100	180	70	489	63.57	0.00%	20.7/sec	0.47	23.0
/sigep.crs/security/isSessionExpired.htm	10	213	127	682	157.72	0.00%	3.9/sec	0.09	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	186	144	304	43.37	0.00%	4.7/sec	0.11	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	169	136	242	21.30	0.00%	2.4/sec	0.05	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	40	176	103	348	44.98	0.00%	3.1/sec	0.07	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	182	137	232	33.15	0.00%	4.7/sec	0.10	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	50	161	45	260	36.82	0.00%	7.2/sec	0.16	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	153	94	222	28.23	0.00%	3.4/sec	0.08	23.0
/sigep.crs/common/global/fichaCaso/verFichaCaso.htm	30	178	82	278	48.52	0.00%	3.7/sec	0.08	23.0
/sigep.crs/common/global/fichaCaso/datosDeIdentifica...	60	174	45	266	45.02	0.00%	7.1/sec	0.16	23.0
/sigep.crs/common/global/fichaCaso/datosLegales.htm	30	163	91	317	45.01	0.00%	3.6/sec	0.08	23.0
/sigep.crs/common/global/fichaCaso/datosDeLaMedid...	30	178	105	219	25.55	0.00%	3.6/sec	0.08	23.0
/sigep.crs/common/busqueda/ejecuciones.htm?idExpe...	20	161	83	222	38.81	0.00%	2.4/sec	0.05	23.0
/sigep.crs/common/busqueda/delitosProbados.htm?	30	179	87	254	41.54	0.00%	3.6/sec	0.08	23.0
/sigep.crs/common/busqueda/procesosPenales.htm?i...	20	180	118	261	41.93	0.00%	2.4/sec	0.05	23.0
/sigep.crs/common/busqueda/presuntosDelitos.htm?	30	161	107	233	25.26	0.00%	3.6/sec	0.08	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	171	126	250	29.21	0.00%	4.6/sec	0.10	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	173	125	235	32.48	0.00%	4.5/sec	0.10	23.0
/sigep.crs/common/busqueda/ejecuciones.htm?idExpe...	10	162	133	234	26.90	0.00%	5.0/sec	0.11	23.0
/sigep.crs/common/busqueda/procesosPenales.htm?i...	10	145	85	207	36.38	0.00%	4.9/sec	0.11	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	40	172	113	254	30.52	0.00%	7.1/sec	0.16	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	193	139	293	47.99	0.00%	4.4/sec	0.10	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	186	68	243	51.32	0.00%	4.4/sec	0.10	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	178	143	240	36.06	0.00%	4.6/sec	0.10	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	20	162	110	302	46.86	0.00%	5.6/sec	0.13	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	191	100	256	45.31	0.00%	4.6/sec	0.10	23.0
/sigep.crs/js/sigep/widget/templates/images/down.gif	10	182	82	257	48.18	0.00%	4.6/sec	0.42	93.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	165	133	224	34.61	0.00%	4.9/sec	0.11	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	172	133	210	27.62	0.00%	4.8/sec	0.11	23.0
/sigep.crs/administracionpenal/administracioncasos/g...	10	154	64	215	40.65	0.00%	4.9/sec	0.11	23.0
TOTAL	1080	1066	29	96894	8571.39	2.69%	7.8/sec	1.91	251.3

Figura 4.1 Simulación de 10 usuarios con un periodo de subida de 30 segundos.

Por cada una de estas filas se muestra la siguiente información:

- **Label**: El nombre de la muestra (conjunto de muestras).
- **# Muestras**: El número de muestras para cada URL.
- **Media**: El tiempo medio transcurrido para un conjunto de resultados.
- **Mín**: El mínimo tiempo transcurrido para las muestras de la URL dada.
- **Máx**: El máximo tiempo transcurrido para las muestras de la URL dada.
- **Error %:** Porcentaje de las peticiones con errores.
- **Rendimiento**: Rendimiento medido en base a peticiones por segundo/minuto/hora.

- **Kb/sec**: Rendimiento medido en Kilobytes por segundo.
- **Avg. Bytes**: Tamaño medio de la respuesta de la muestra medido en bytes.

Simulación de 30 usuarios con un periodo de subida de 60 segundos

Realizando la simulación con 30 usuarios considerando un periodo de subida de 60 segundos (de nuevo 3 segundos entre el lanzamiento de cada hilo) los resultados serán los siguientes, teniendo en cuenta que dichos resultando se irán solapando a los ya obtenidos en la simulación anterior.

Label	# Samp...	Average	Min	Max	Std. Dev.	Error %	Through...	KB/sec	Avg. Bytes
/sigep.crs/security/login.htm	44	10283	86	92765	26060.41	11.36%	3.4/min	0.19	3411.0
/sigep.crs/security/captcha.jpeg	44	2831	31	95259	14105.57	9.09%	3.4/min	0.13	2284.7
/sigep.crs/common/global/servertime.htm	84	587	36	2012	450.52	4.76%	5.9/min	0.02	181.5
/sigep.crs/common/global/timezonepopup.htm	82	1731	41	96894	10581.29	3.66%	5.6/min	0.05	564.7
/sigep.crs/security/j_acegi_security_check	41	812	49	2684	552.38	7.32%	3.1/min	0.18	3475.1
/sigep.crs/	41	847	47	3593	669.92	7.32%	3.1/min	0.18	3470.4
/sigep.crs/common/global/welcome.htm	54	2419	37	95833	12843.36	3.70%	2.4/min	0.14	3526.5
/sigep.crs/common/global/registerClientOffset.htm	54	486	29	2012	458.45	3.70%	2.4/min	0.00	97.4
/sigep.crs/common/global/subSystemMenuAccess.htm	54	2272	30	96263	12917.95	1.85%	2.4/min	0.00	60.2
/sigep.crs/administracionpenal/index.htm	41	717	44	2822	577.85	2.44%	3.1/min	0.18	3546.1
/sigep.crs/common/global/rpc/jsonrpc.htm	123	1254	23	94968	8492.47	0.81%	9.2/min	0.01	39.3
/sigep.crs/common/global/isUsableResource.htm	179	665	22	12151	1096.43	0.00%	8.6/min	0.00	23.0
/sigep.crs/administracion/seguridad/jsonrpc/jsonrpc.h...	41	549	45	1768	343.37	0.00%	3.5/min	0.00	23.0
/sigep.crs/seguridadcustodia/jsonrpc.htm	41	477	28	2014	415.62	0.00%	3.4/min	0.00	23.0
/sigep.crs/common/busqueda/criteriosRpc/jsonrpc.htm	41	496	30	2476	449.68	0.00%	3.4/min	0.00	23.0
/sigep.crs/tratamiento/jsonrpc.htm	41	557	51	1201	345.10	0.00%	3.4/min	0.00	23.0
/sigep.crs/administracionpenal/capacidades/jsonrpc/j...	41	596	31	1205	343.89	0.00%	3.3/min	0.00	23.0
/sigep.crs/administracionpenal/welcome.htm	54	574	23	4008	625.90	0.00%	2.6/min	0.00	23.0
/sigep.crs/common/global/hasAccessToResource.htm	127	783	24	8487	1053.37	0.00%	6.3/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	80	732	23	5984	883.60	0.00%	4.8/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	399	660	23	5145	605.38	0.00%	23.3/min	0.01	23.0
/sigep.crs/security/isSessionExpired.htm	41	498	23	1177	284.72	0.00%	2.9/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	39	1162	24	5256	1119.15	0.00%	2.4/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	49	894	24	7715	1265.66	0.00%	2.4/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	75	571	30	2377	571.14	0.00%	3.6/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	32	1006	137	6089	1282.39	0.00%	1.7/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	97	907	24	10686	1451.59	0.00%	4.8/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	45	817	25	5110	1086.10	0.00%	2.3/min	0.00	23.0
/sigep.crs/common/global/fichaCaso/verFichaCaso.htm	59	614	82	3492	712.23	0.00%	2.9/min	0.00	23.0
/sigep.crs/common/global/fichaCaso/datosDeIdentific...	117	653	29	4746	809.94	0.00%	5.8/min	0.00	23.0
/sigep.crs/common/global/fichaCaso/datosLegales.htm	56	569	26	3076	623.45	0.00%	2.8/min	0.00	23.0
/sigep.crs/common/global/fichaCaso/datosDeLaMedi...	54	723	23	7876	1193.20	0.00%	2.7/min	0.00	23.0
/sigep.crs/common/busqueda/ejecuciones.htm?idExp...	38	652	83	3561	734.60	0.00%	1.9/min	0.00	23.0
/sigep.crs/common/busqueda/delitosProbados.htm?	52	864	87	6964	1280.39	0.00%	2.6/min	0.00	23.0
/sigep.crs/common/busqueda/procesosPenales.htm?...	36	1032	22	10164	1789.66	0.00%	1.8/min	0.00	23.0
/sigep.crs/common/busqueda/presuntosDelitos.htm?	50	814	107	6579	1232.76	0.00%	2.5/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	35	1280	24	16193	3039.44	0.00%	1.8/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	34	931	125	7357	1423.80	0.00%	1.7/min	0.00	23.0
/sigep.crs/common/busqueda/ejecuciones.htm?idExp...	16	719	133	3539	996.19	0.00%	49.0/hour	0.00	23.0
/sigep.crs/common/busqueda/procesosPenales.htm?...	16	452	85	1537	476.17	0.00%	48.8/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	56	492	113	5960	856.78	0.00%	2.7/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	15	476	139	1779	516.01	0.00%	44.9/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	14	320	26	1065	315.56	0.00%	41.6/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	14	429	24	1985	576.00	0.00%	41.6/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	26	322	110	1474	343.94	0.00%	1.3/min	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	13	328	100	1208	291.83	0.00%	38.5/hour	0.00	23.0
/sigep.crs/js/sigep/widget/templates/images/down.gif	13	264	82	610	157.32	0.00%	38.4/hour	0.00	93.0
/sigep.crs/administracionpenal/administracioncasos/...	13	361	133	1524	407.89	0.00%	38.4/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	13	427	133	2239	579.55	0.00%	38.3/hour	0.00	23.0
/sigep.crs/administracionpenal/administracioncasos/...	13	329	64	1141	344.58	0.00%	38.2/hour	0.00	23.0
TOTAL	2837	984	22	96894	5351.40	1.02%	2.1/sec	0.71	351.4

Figura 4.2 Simulación de 30 usuarios con un periodo de subida de 60 segundos.

El rendimiento muestra que para una simulación de 30 usuarios junto a un periodo de subida de 60 segundos el servidor es capaz de aceptar una media de 2.1 peticiones por minuto. Tiene un margen de error de 1.02%.

Pruebas Unitarias:

Facilitan el cambio de código para mejorar la estructura y así asegurarse de que los nuevos cambios no han introducido errores, simplifica la integración, separación de la interfaz y la implementación, se puede cambiar cualquiera de los dos sin afectar al otro y de esta forma los errores están más acotados y son fáciles de localizar.

Pruebas exploratorias:

Se ejecutan pruebas a ciegas y al azar a la aplicación con el objetivo de encontrar errores y provocar fallos al sistema. Para este tipo de pruebas no se sigue ningún manual de usuario ni documento de especificación de casos de uso o requisitos.

4.5. Herramientas para automatizar las pruebas.

4.5.1. JUnit.

En el presente trabajo, se realizaron las pruebas unitarias a nivel de desarrollador, teniendo en cuenta, que las mismas deben ser automatizables, completas, reutilizables e independientes. Para realizar las pruebas se utilizó **JUnit** integrado al IDE Eclipse en forma de plug-in, lo cual permite que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador, enfocarse en la prueba y el resultado esperado, dejando a la herramienta, la creación de las clases que permiten coordinar las pruebas.

4.5.2 JMeter.

Utilizar **JMeter** en aplicaciones web para la comprobación de los recursos del sistema, supone una mayor efectividad en el proceso y en la fiabilidad de los resultados. Como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios. Se verifica el rendimiento del sistema mediante las pruebas de Carga y Estrés. JMeter permite realizar pruebas de carga sobre cada una de las capas que conforman la aplicación a probar. De esta forma, es sencillo localizar el foco que está generando contención y está afectando los tiempos de respuesta de un caso de uso específico. [16]

4.6 Conclusiones:

Para lograr una mayor calidad y eficacia en el proceso de automatización se llevó a cabo la concepción de las pruebas con el régimen que dictaminan las pautas para su desarrollo y además un buen acondicionamiento del ambiente de las pruebas. Se describieron los principales tipos de prueba utilizadas. Se listaron las no conformidades

detectadas durante las pruebas de liberación realizadas a los módulos Administración de Casos y Actividades de Atención, las cuales fueron resueltas.

Conclusiones

Una vez culminado el trabajo es posible afirmar que se les dio cumplimiento a los objetivos trazados para el mismo:

- Se realizó el diseño de los módulos Administración de Casos y Actividades de Atención teniendo en cuenta los requisitos capturados durante el proceso de definición del sistema.
- El sistema fue implementado utilizando herramientas, lenguajes y tecnologías, en su mayoría distribuidas bajo licencias de software libre.
- Las actividades de diseño e implementación fueron ejemplificadas a través de diagramas de clase y fragmentos de código fuente.
- Se cumplió el objetivo general planteado contribuyendo al mejoramiento del Sistema Penitenciario Venezolano.

RECOMENDACIONES

Se recomienda:

Comprobar el rendimiento de los módulos frente a bases de datos de grandes volúmenes de información para demostrar que el diseño propuesto satisface tales condiciones.

Revisar y estudiar las funcionalidades de los módulos Administración de Casos y Actividades de Atención con el objetivo de agregar nuevas funcionalidades.

REFERENCIAS BIBLIOGRÁFICAS.

1. **OSSORIO, MANUEL.** *Diccionario de Ciencias Jurídicas, Políticas y Sociales.* Guatemala : Datascan, S.A.

2. INPEC. *Instituto Nacional Penitenciario y Carcelario.* [Online] http://www.inpec.gov.co/portal/page/portal/Inpec.

3. NEOTEC. [Online] http://www.neotec.cc/about_us/index_es.html.

4. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado.* Madrid : Pearson Educación, 2007.

5. Visual Paradigm. [Online] http://www.visual-paradigm.com/product/vpuml/.

6. **Juan Pavón Mestras.** *El patrón Modelo-Vista-Controlador.*

7. Hibernate . [Online] http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=Hibernate.

8. BEATON W. Eclipse Platform Technical Overview. [Online] http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html. .

9. Oracle. [Online] http://www.oracle.com/technetwork/database/enterprise-edition/documentation/index.html.

10. Framework Dojo. [Online] http://www.dojotoolkit.org/.

11. Apache Tomcat. [Online] http://tomcat.apache.org/tomcat-5.5-doc/index.html.

12. **Granda, Roberto.** *TRABAJO DE DIPLOMA Diseño e Implementación del módulo Requisas y Decomisos.* Ciudad de la Habana : s.n., 2010.

13. Conferencia # 7 Disciplina de Prueba. Ingeniería del Software II. [Online] http://eva.uci.cu .

14. **Michael, Ing. Aguilar Hernández Violena y Ing.González.** *Proceso de pruebas de caja negra basado en la descripción de casos de uso.*

15. Boucchechter, Carolina Zibert van Gricken Israel. [Online] http://carolina.terna.net/ingsw3/datos/Pruebas_de_Confiabilidad.pdf..

16. **Basco, Sociedad Informatica del Gobierno.** *JMeter. Manual de usuario.*

BIBLIOGRAFÍA

- Atención, Supervisión y Orientación de Casos en un CRS: proceso elemental de negocio Humanización Penitenciaria Fase IV SIGEP v2.1.

- Supervisión y Orientación de Casos en UTSO: proceso elemental de negocio Humanización Penitenciaria Fase IV SIGEP v2.1.

- MANUAL DE USUARIO DEL SUBSISTEMA ADMINISTRACIÓN DE LA SEDE SIGEP V 2.1.

- MANUAL DE USUARIO DEL SUBSISTEMA ATENCIÓN, SUPERVISIÓN Y ORIENTACIÓN SIGEP V 2.1.

- **WALLS, CRAIG y BREINDENBACH, RAYN.** *Spring in Action.* Greenwich : MANNING, 2008.

- **PIMENTEL GONZÁLEZ, LUIS ALBERTO.** *Documento de Arquitectura de Software.* Ciudad de La Habana : ALBET, 2007.

- **PÉREZ RIVERO, IÓSEV y PIMENTEL GONZALEZ, LUIS ALBERTO.** *ArBaWeB: Arquitectura Base sobre la web.* Ciudad de La Habana : UCI, 2007.

ANEXOS

Anexo 1 Diagramas de clase del diseño.

- **Módulo Administración de Casos.**

Figura 3.21 Asignar nuevos casos a Delegado de Prueba.

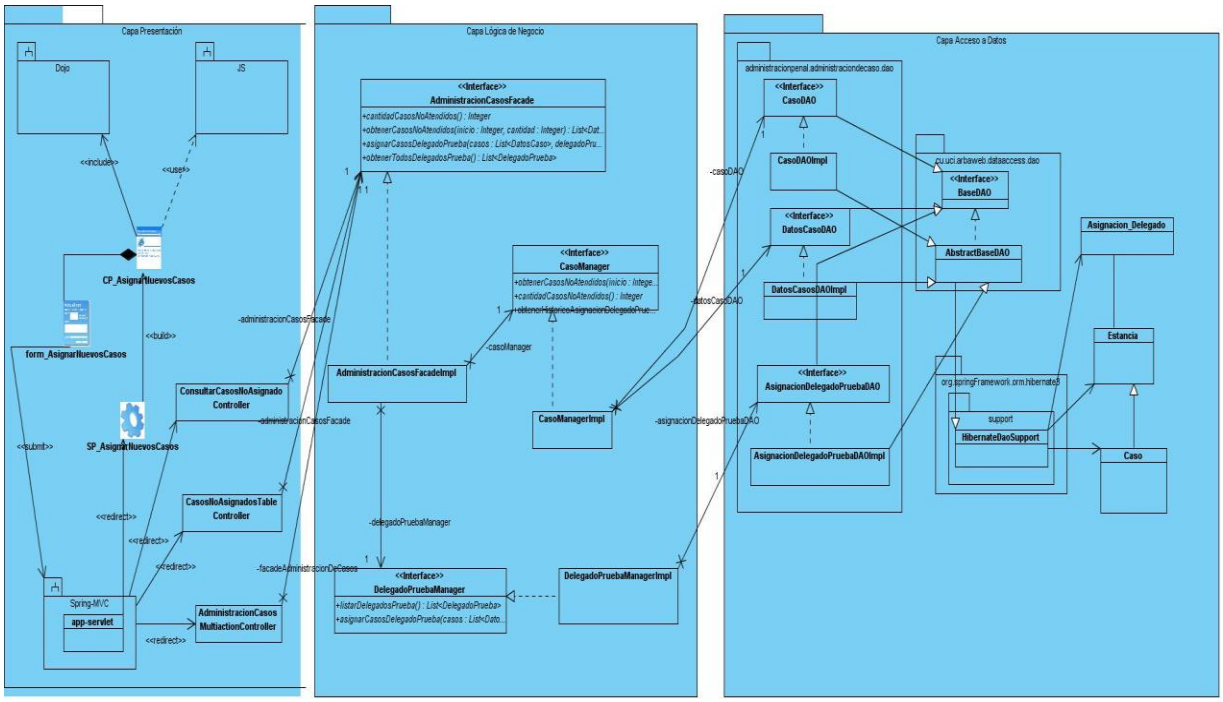


Figura 3.22 Consultar casos por Delegado de Prueba.

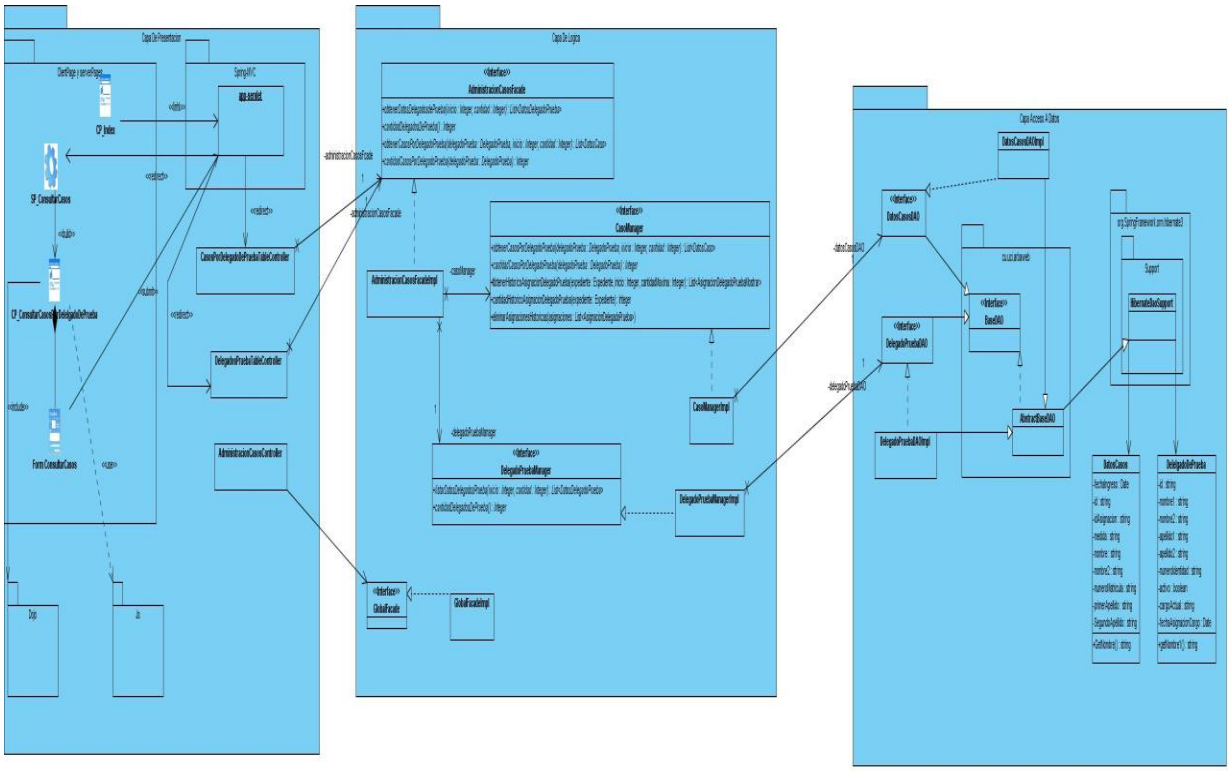


Figura 3.23 Reasignar casos de Delegado de Prueba.

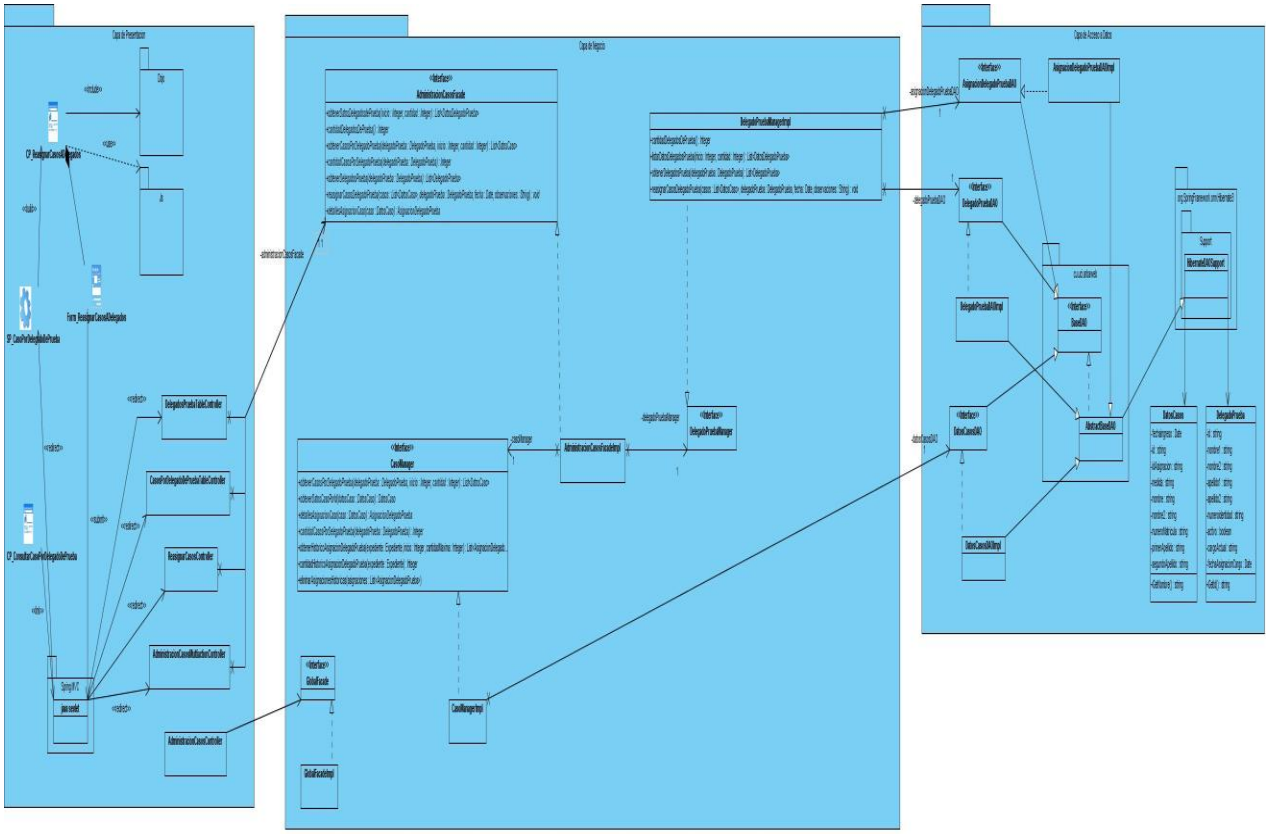


Figura 3.24 Desvincular casos de Delegado de Prueba.

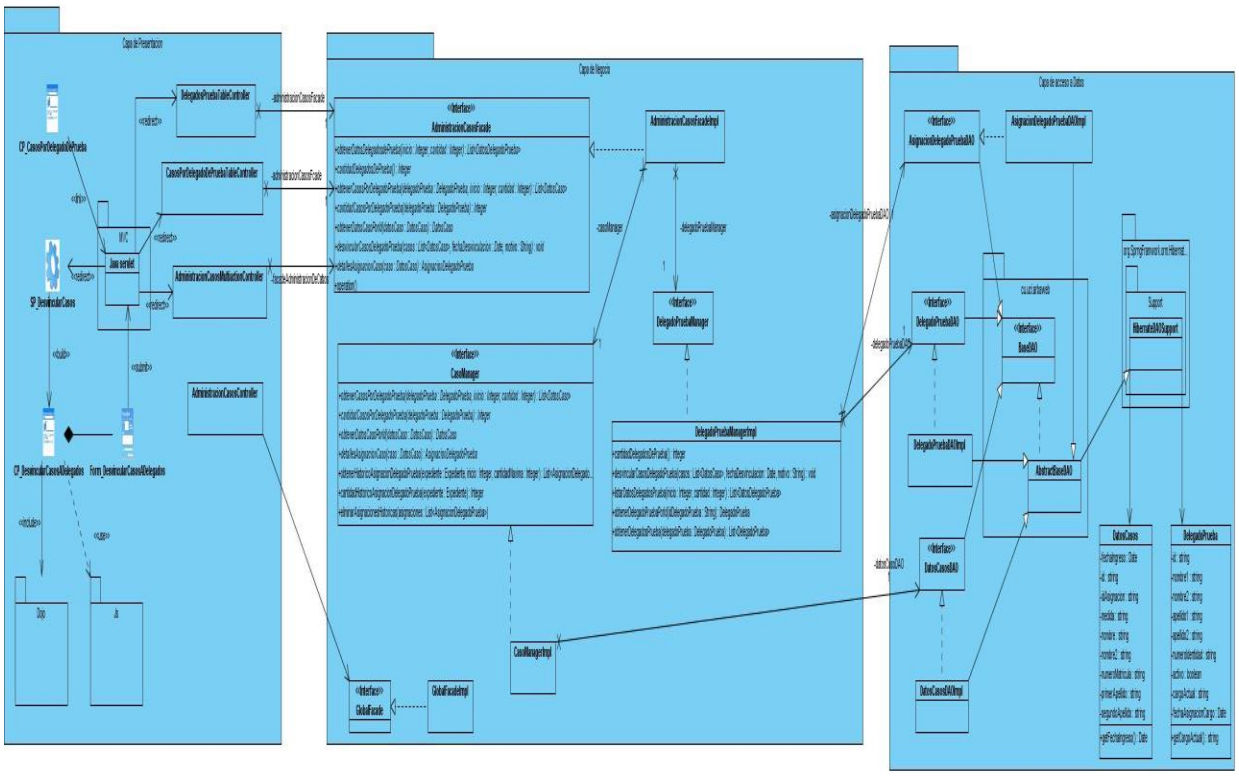
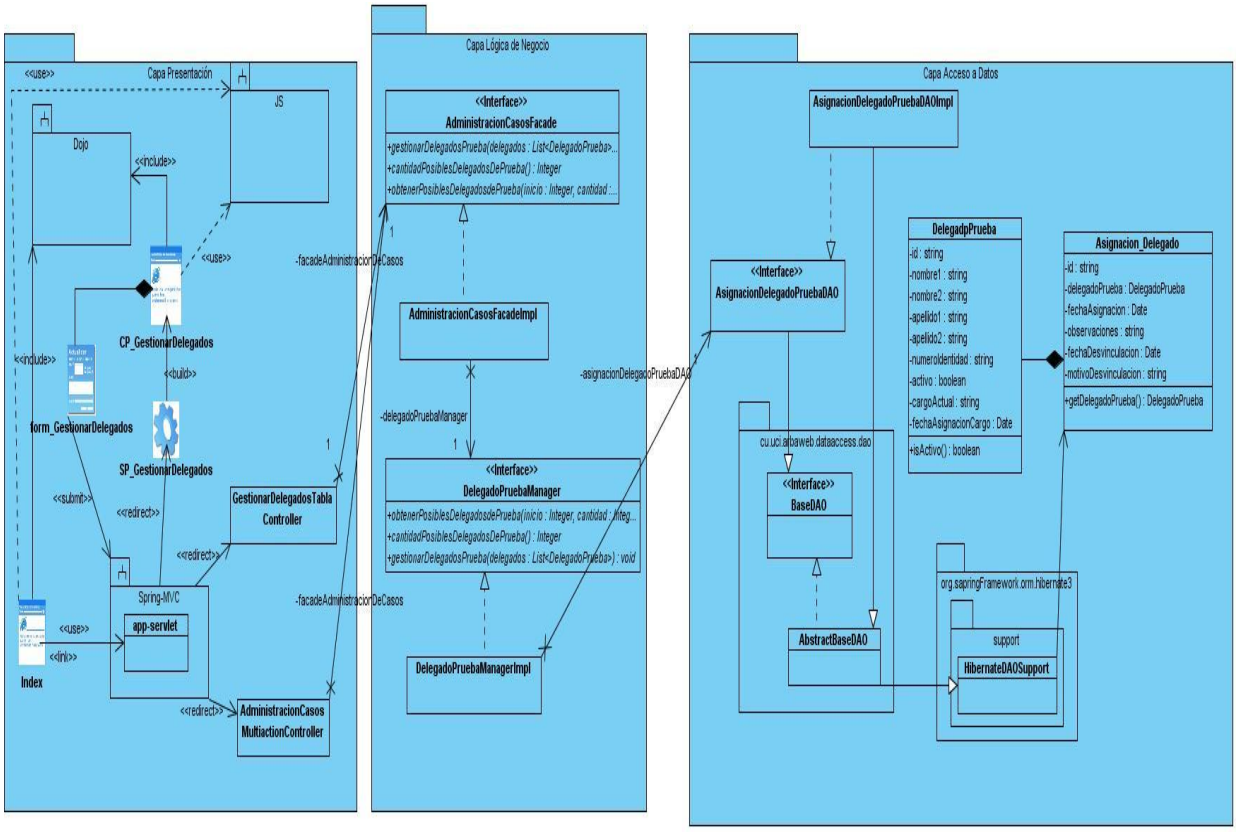


Figura 3.25 Gestionar Delegados de Prueba.



Módulo Actividades de Atención.

Figura 3.26 Registrar-modificar actividad de atención.

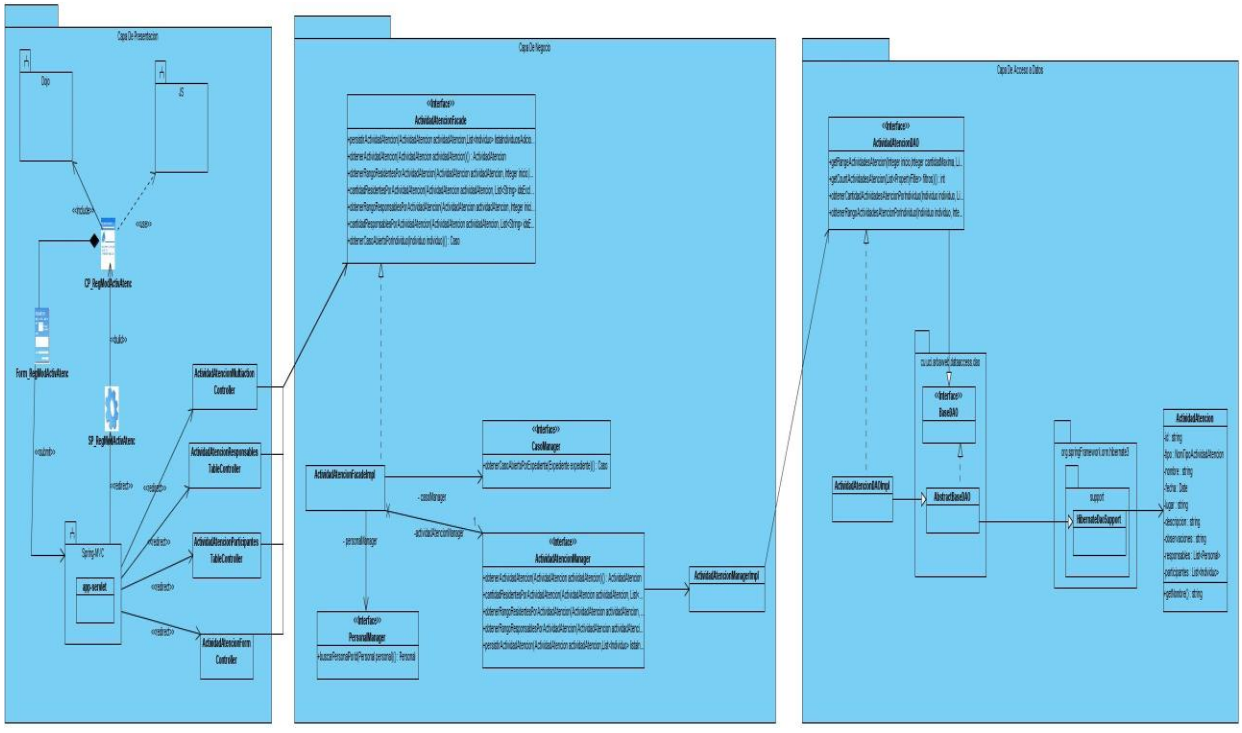
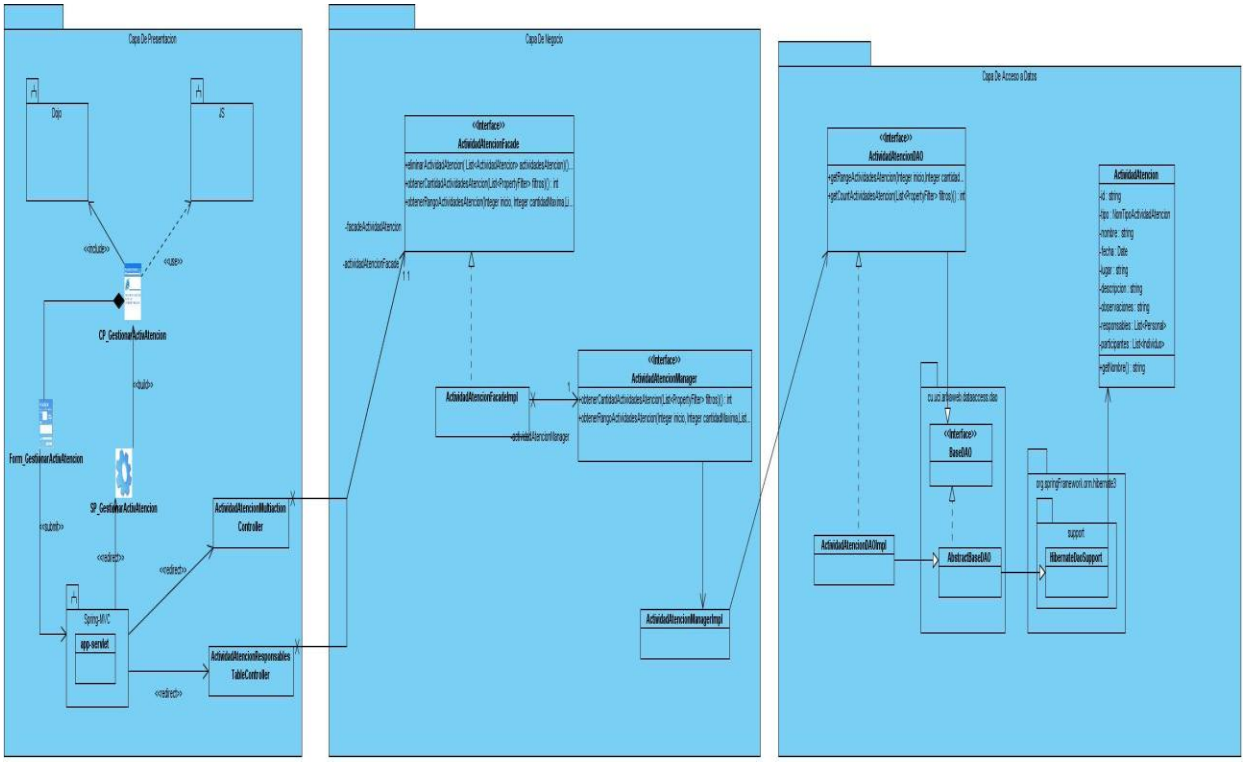


Figura 3.27 Gestionar actividades de atención.



Anexo 2 Diagramas de componentes.

- **Módulo Administración de Casos.**

Figura 3.28 Asignar nuevos casos a Delegado de Prueba.

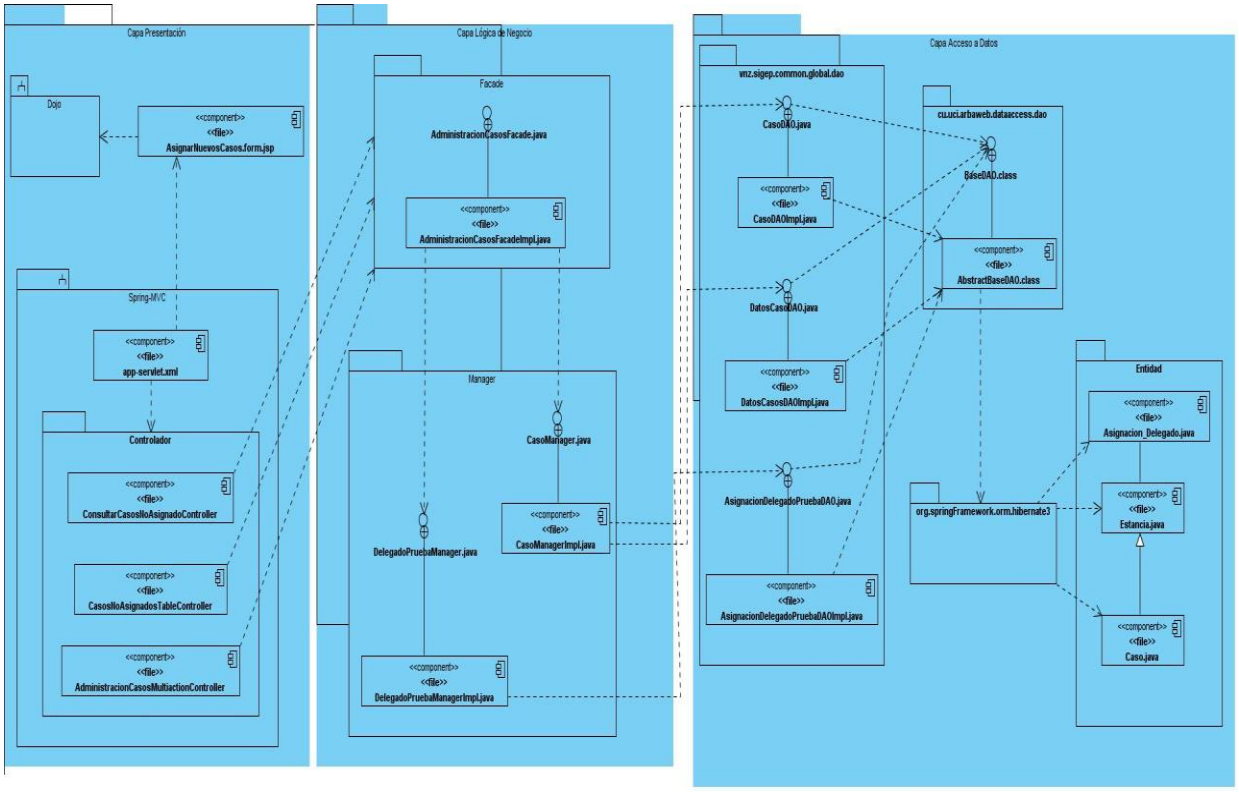


Figura 3.29 Consultar casos por Delegado de Prueba.

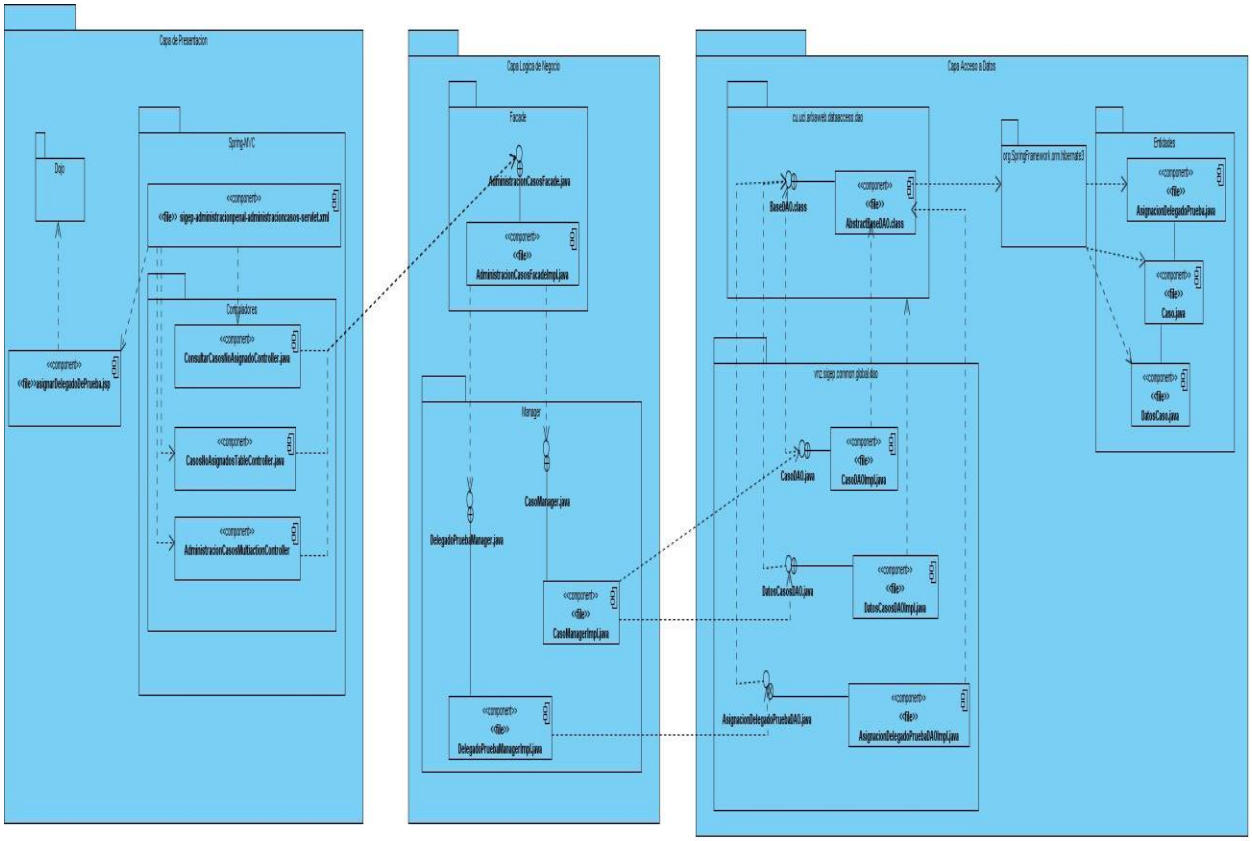


Figura 3.30 Reasignar casos de Delegado de Prueba.

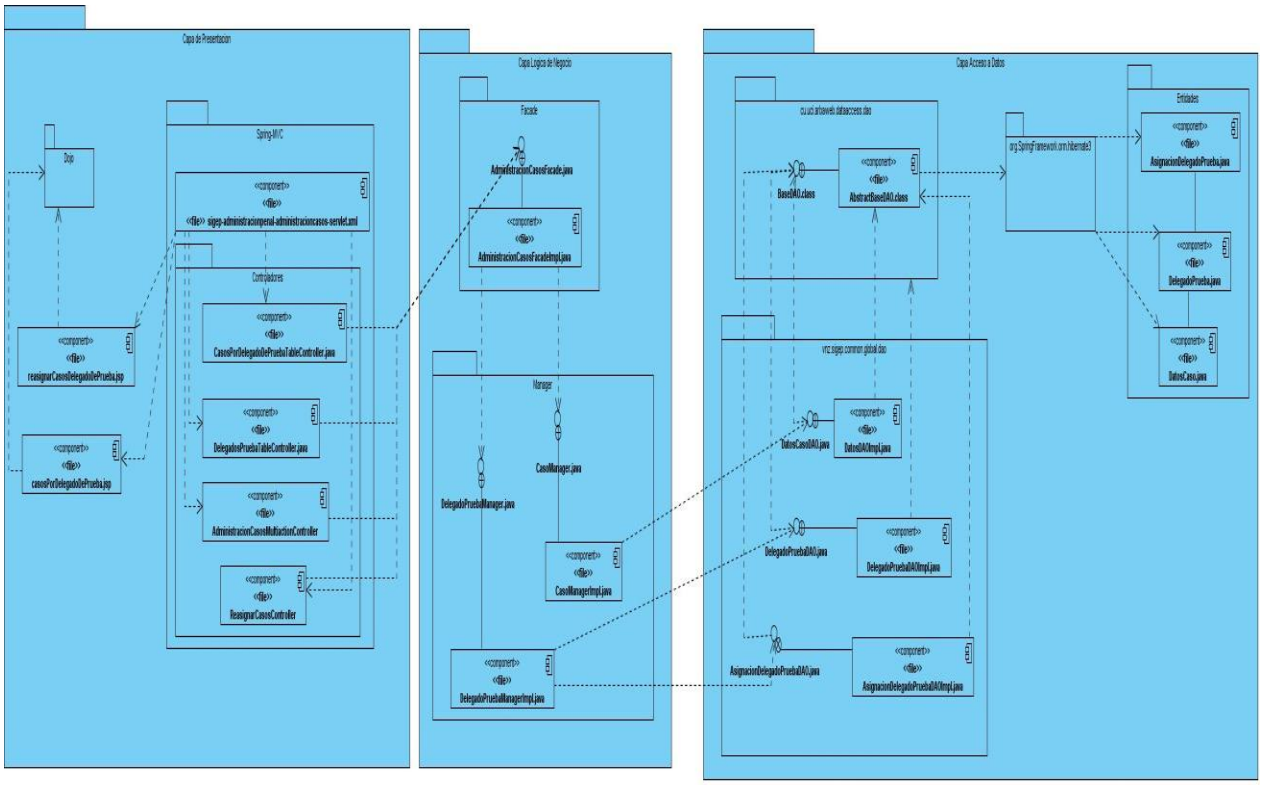


Figura 3.31 Desvincular casos de Delegado de Prueba.

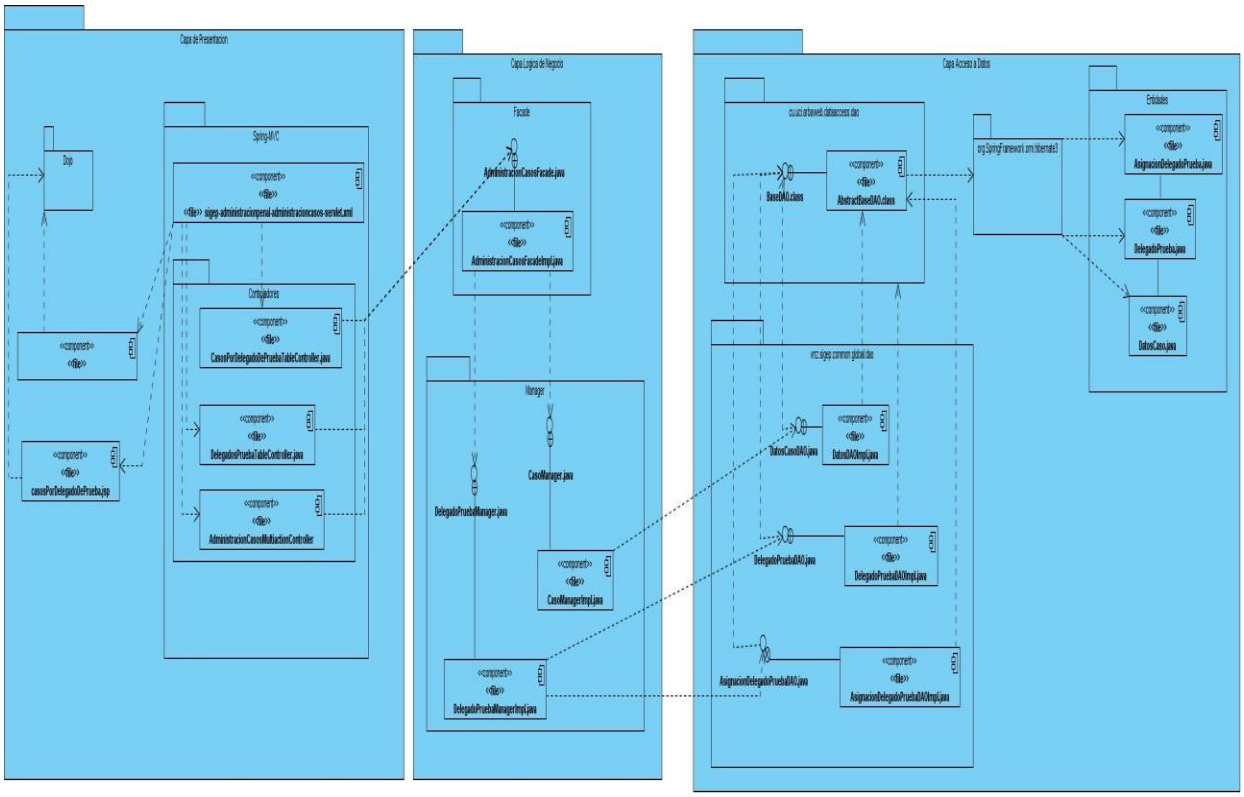
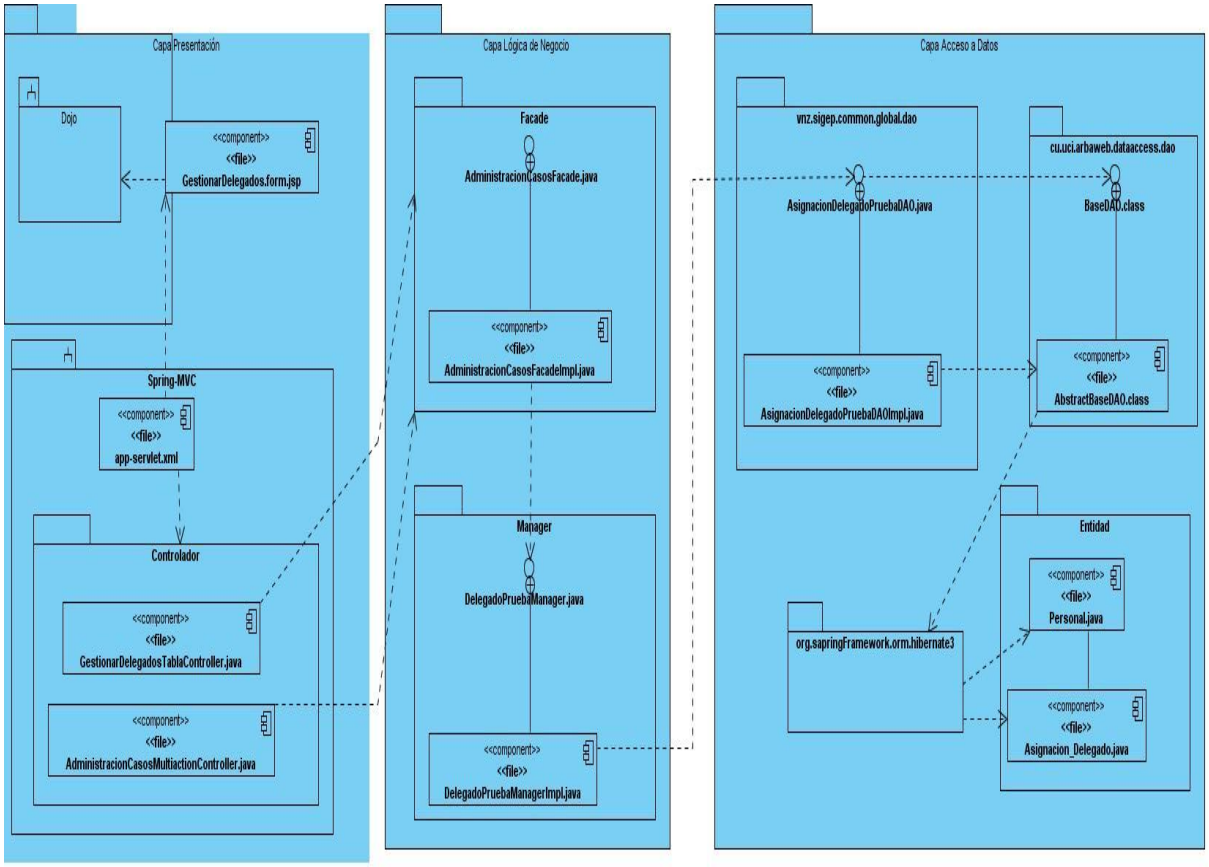


Figura 3.32 Gestionar Delegados de Prueba.



- **Módulo Actividades de Atención.**

Figura 3.33 Gestionar actividades de atención.

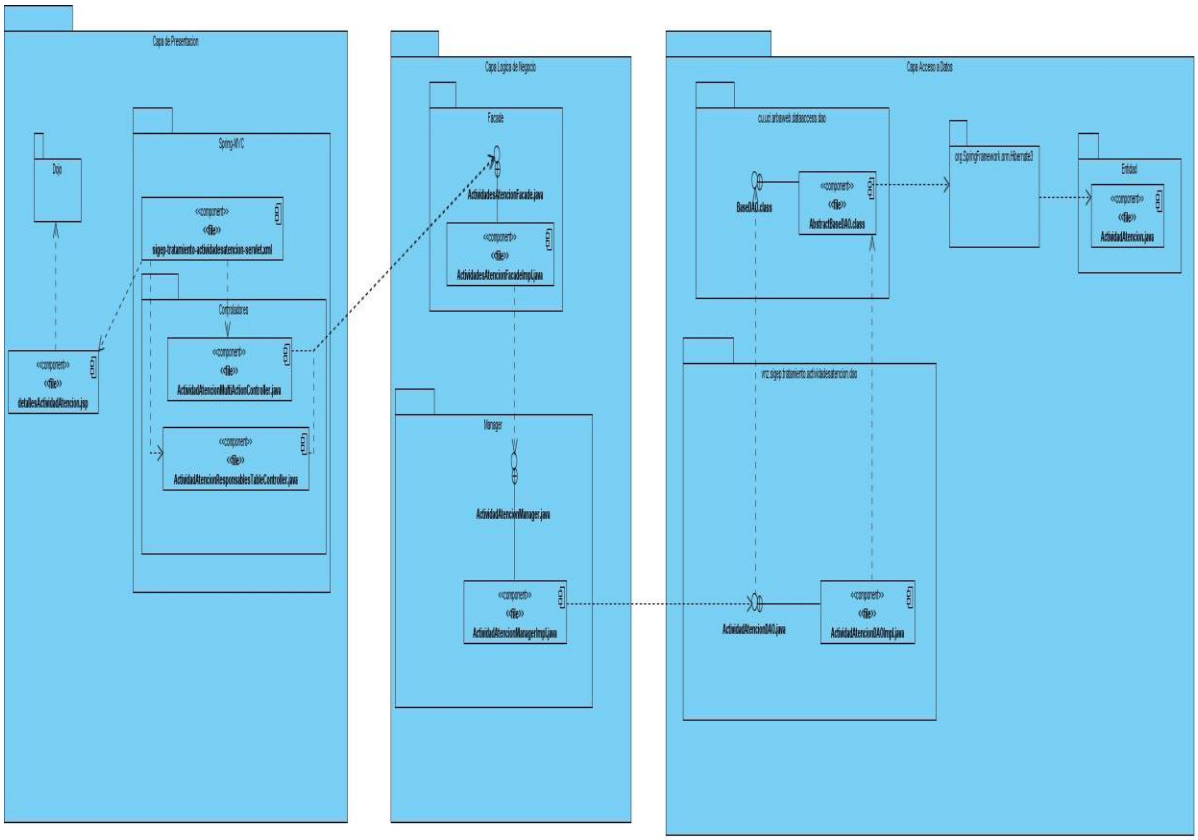
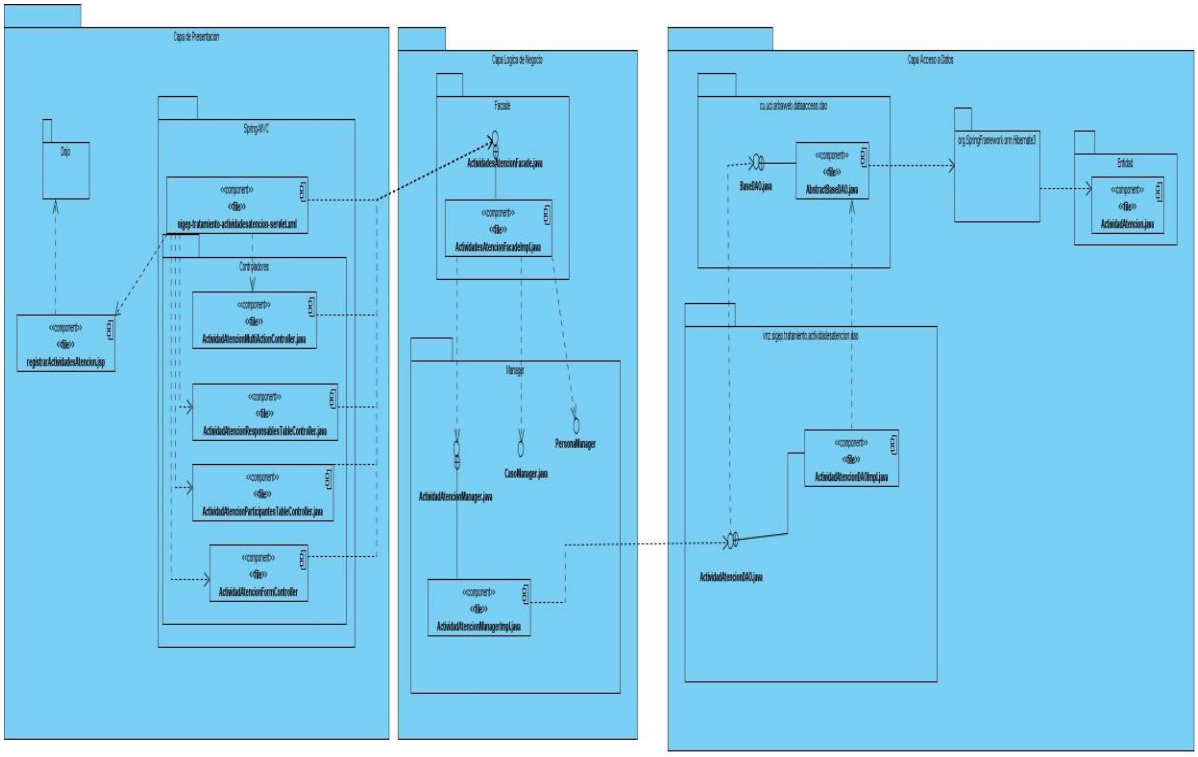


Figura 3.34 Registrar-modificar actividad de atención.



Anexo 3 Descripción de las entidades de dominio significativas.

NomTipoActividadAtencion

Descripción	Nomenclador que refleja el tipo de actividad.	
Atributos	**Visibilidad**	**Tipo**
id	private	String
nombre	private	String

Personal

Descripción	Entidad que refleja los responsables de la Actividad de Atención, clase ubicada en el paquete domain del common global, ya que es utilizada en otros módulos.	
Atributos	**Visibilidad**	**Tipo**
user	private	User
id	private	String
Nombre1	private	String
Nombre2	private	String

apellido1	private	String
Apellido2	private	String
numeroIdentidad	private	String
centro	private	Centro
activo	private	boolean
telefono	private	String
direccion	private	String
cargoFuncionarios	private	List<CargoFuncionario>
tipoDocumentoIdentificacion	private	NomTipoDocumentoIdentificacion

Anexo 4 Descripción de las clases significativas de la capa de negocio.

ActividadesAtencionFacadeImpl

Descripción	Implementación de la Interfaz ubicada en el paquete facade del subsistema Actividad de Atención, facade que tiene todas las funcionalidades de este módulo.
Interfaz	ActividadesAtencionFacade

Anexo 5 Descripción de las clases significativas de la capa de acceso a datos.

ActividadAtencionDAOImpl

Descripción	Clase DAO ubicada en el paquete dao/impl del subsistema, presenta las funcionalidades para una actividad de atención.
Padre	AbstractBaseDAO

GLOSARIO

SIGEP: Sistema de Gestión Penitenciaria.

CRS: Centros de Residencia Supervisada.

UTSO: Unidades Técnicas de Supervisión y Orientación.

Módulo: Encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño.

IDE: Integrated Development Enviroment (Ambiente Integrado de Desarrollo).

API's: (Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Subsistema: Un subsistema se refiere a un conjunto de módulos que por razones de similitud o de perseguir objetivos comunes, son agrupados**.**

CU: (Caso de Uso) es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software.

Código abierto u Open source: es el término con el que se conoce al software distribuido y desarrollado libremente.

Entidades: Objetos concretos o abstractos que presentan interés para el sistema.

HTML: Acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

JSP (Java Server Pages): es una tecnología orientada a crear páginas web con programación en Java.