UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Desarrollo del Subsistema Equipos Multidisciplinarios del Sistema Informativo de la Dirección de Establecimientos Penitenciarios.

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

Autor(es): Zuleyma Viera González.

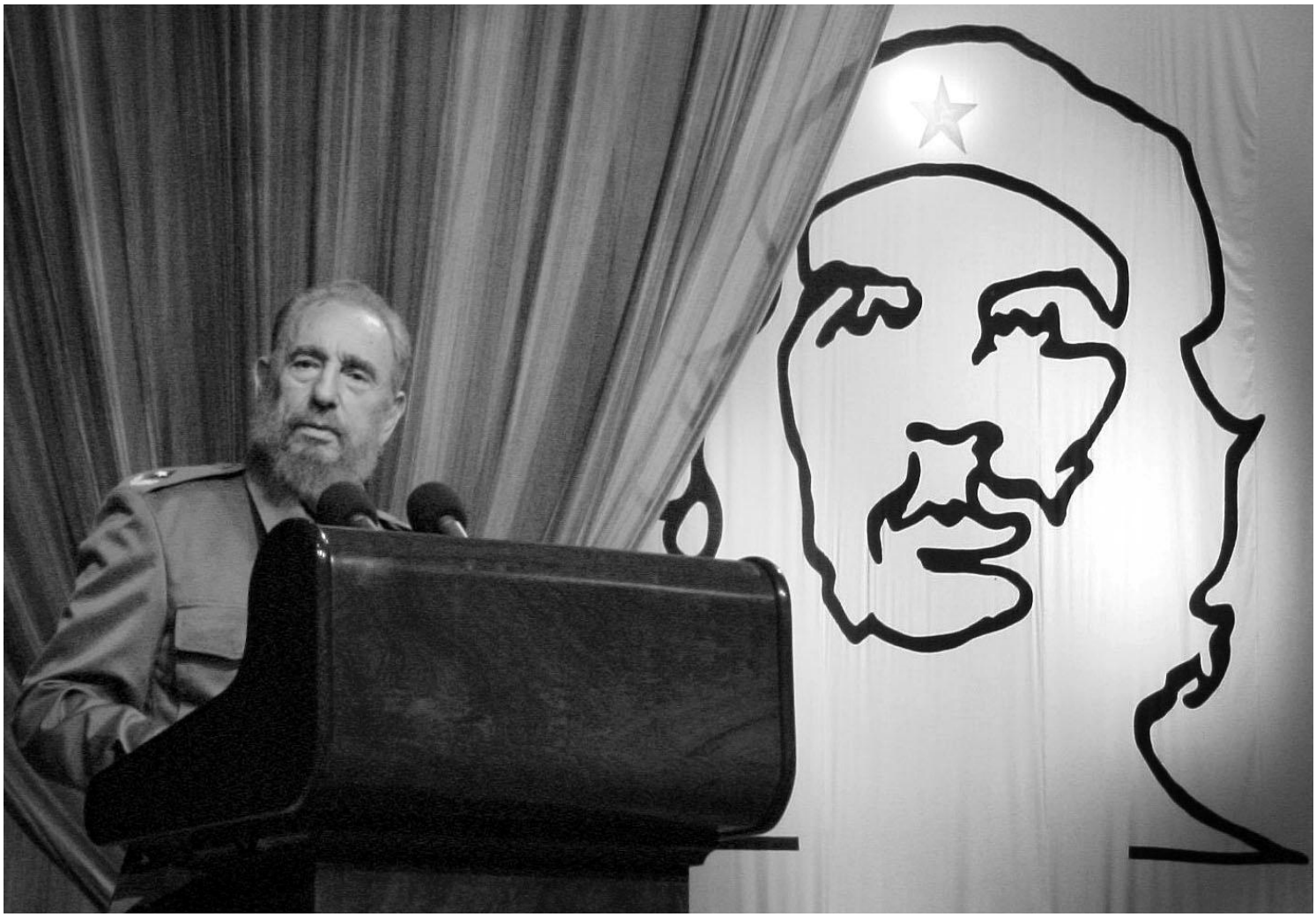
Daniel Martín Morejón.

Tutor: Ing. Guillermo E. Ferrás Pérez.

Co-Tutora: Ing. Anabel Betancourt de los Santos.

La Habana, junio de 2012

"Año 54 de la Revolución"



¡Adelante, compañeros, hacia las nuevas metas, a cumplir las nuevas promesas, a cumplir las nuevas tareas, a hacerse maestros, a hacerse técnicos, a hacerse médicos, a hacerse profesores, a hacerse ingenieros, a hacerse intelectuales revolucionarios!

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Por este medio declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de Ciencias Informáticas para que haga el uso que estime pertinente con el mismo.

Para que así conste firmamos la presente a los___ días del mes de___ del año_____.

Zuleyma Viera González

_____

Firma del Autor

Daniel Martín Morejón

_____

Firma del Autor

Ing. Guillermo E. Ferrás Pérez

_____

Firma del Tutor

Ing. Anabel Betancourt de los Santos

_____

Firma del Cotutor

DATOS DE CONTACTO

Tutor:

Ing. Guillermo Enrique Ferrás Pérez.

Graduado en la Universidad de las Ciencias Informáticas (UCI) como Ingeniero en Ciencias Informáticas.

Email: geferras@uci.cu

Cotutora:

Ing. Anabel Betancourt De los Santos

Graduado en la Universidad de las Ciencias Informáticas (UCI) como Ingeniero en Ciencias Informáticas.

Email: adelossantos@uci.cu

Agradecimientos de Zuleyma Viera González

Son muchas las personas especiales a las que me gustaría agradecer por su amistad, apoyo y compañía en las diferentes etapas de mi vida. Algunas están aquí conmigo y otras en mis recuerdos y en el corazón. Sin importar en donde estén o si alguna vez llegan a leer estas líneas, quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todo su amor.

A mis abuelitos maternos, mis ángeles guardianes, mis protectores incondicionales, que durante sus cortas vidas me apoyaron, malcriaron y sobre todo me dieron todo su amor y cariño, en este momento donde quiera que se encuentren, sé que se sienten orgullosos de mí.

A mi Madre, "mima" no me equivoco si digo que eres la mejor mamá del mundo, gracias por todo tu esfuerzo, apoyo ilimitado e incondicional y la confianza que depositaste en mí, por tener la fortaleza de salir adelante sin importar los obstáculos, por haberme transformado en una mujer de bien. Has sido madre, compañera, amiga y consejera. Gracias porque siempre, aunque lejos, has estado a mi lado. No existen palabras en este mundo para agradecerte lo que has hecho por mi hermana y por mí. Te quiero.

A mi hermana Zuleyka, por su nobleza y cariño, porque con su risa desmedida me ha hecho reír en los momentos más difíciles.

A toda mi familia, por contribuir en la realización de este gran sueño.

A María del Carmen, doy gracias por contar con el apoyo de alguien tan especial como tú en los momentos difíciles, conocerte es lo mejor que me ha podido suceder. Gracias por el cariño, por el afecto y por todo lo que has hecho por mí. Es bueno saber que en el mundo

hay alguien como tú, siempre dispuesto a dar una mano. Si tuviera que decir una palabra que te identifique esa sería "Madre".

A Carmen Leisa, gracias por no juzgar, por escuchar sin opinar, por hacerme saber que siempre estarás allí si te necesito, por hacerme saber que, aunque hago cosas que no comprendes, siempre me estarás esperando. Gracias por tu ayuda desmedida en la realización de este trabajo. Tu amistad ha sido de las mejores cosas que me ha pasado en la vida, por todo esto y mucho más, Gracias.

A Rivi, Dayi, Yona, Yei, Marvis, Pedro, Yulia y Yilén por su amistad y apoyo, por estar presente en las experiencias buenas y malas durante la etapa de la universidad.

A Pierre, gracias por tu apoyo y ayuda sin límites, sin ti muchas cosas me hubiesen sido difíciles y en algunos casos imposibles, pues siempre has estado ahí, en las buenas y malas, gracias.

A Daniel, mi compañero de tesis, por su compresión, paciencia y dedicación durante el desarrollo de nuestro trabajo.

A Guillermo, nuestro tutor, por su confianza y ayuda en todo lo que le fue posible.

A Yanay, gracias por toda el apoyo que me brindaste durante la realización de mi tesis.

A Leandro, por estar presente siempre que necesité de ti, si no fuera por ti no hubiese desarrollado este tema de tesis.

A todos los presentes, por apoyarme en este momento tan difícil, especial e inolvidable.

Agradecimientos de Daniel Martín Morejón

Agradezco a mis padres por haberme dado el apoyo necesario durante el transcurso de la carrera, por ayudarme en lo que estuvo a su alcance.

Agradezco a mis profesores durante toda la carrera por instruirme y poner a mi alcance los conocimientos durante los 5 años de la carrera.

Agradezco a mi líder de proyecto y al resto de los profesores que brindaron su ayuda durante el desarrollo de la solución.

Agradezco a mi tutor por orientarnos y guiarnos con el objetivo de que el trabajo de diploma se terminara en tiempo y con la calidad requerida.

Agradezco a mis compañeros de aula y de residencia durante todos estos años los cuales jugaron un importante papel en el compañerismo, solidaridad y amistad demostrada que influenció de manera positiva durante toda la carrera.

Agradezco a mi amigo Daniel Alejandro Linares Brooks por ser más que un amigo fiel que siempre está en los momentos buenos y también en los malos, aconsejándome como a un hermano y haciendo todo lo que está a su alcance para mi crecimiento personal y profesional. Le agradezco por ser esa mano firme que siempre pude estrechar sin importar las circunstancias.

Le doy gracias a Dios por permitirme llegar hasta donde he llegado.

Dedicatoria de Zuleyma Viera González

Dedico mi trabajo de diploma a mis abuelitos, los cuales se esforzaron al máximo para que cumpliera mis metas y sueños. A mi mamá, por su confianza, amor y dedicación, porque con solo escuchar su voz soy capaz de salir adelante por muy abatida que me encuentre. Te dedico este triunfo porque el orgullo y admiración que sientes por mí, es lo que me ha impulsado hasta el final. Porque admiro tu fortaleza y por lo que has hecho de mí. A mis familiares que me han dado su apoyo durante toda mi vida, para que hiciera realidad este sueño.

Dedicatoria de Daniel Martín Morejón

Les dedico mi trabajo de diploma a mis padres, demás familiares, amigos, en especial a Daniel Linares y personas que de una forma u otra brindaron su apoyo para poder llegar hasta el final.

RESUMEN

EL Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP) cuenta en la actualidad con el Sistema Automatizado para el Control del Recluso (SACORE) que permite gestionar la información relacionada con los internos. Sin embargo, en los centros penitenciarios se llevan a cabo otros procesos que el SACORE no incluye. Ejemplo de lo anterior es la labor que desempeña el colectivo de Equipos Multidisciplinarios, encargado de evaluar psicológica y criminológicamente a los reclusos, cuyo trabajo se ve afectado, por no contar con una herramienta que les permita automatizar sus funciones.

En el siguiente trabajo se desarrolla el Subsistema Equipos Multidisciplinarios del SIDEP, cumpliendo con los objetivos y tareas planteadas a partir del análisis de los requerimientos de software establecidos de acuerdo con el cliente. Además, la presente investigación hace uso de las tecnologías y herramientas definidas por el proyecto Prisiones Cuba, perteneciente a la Universidad de las Ciencias Informáticas (UCI), quien está responsabilizado con la automatización de todos los procesos que desarrolla el Sistema Penitenciario Cubano.

Los resultados esperados son los modelos de análisis, diseño e implementación y la validación del Subsistema, así como obtener una aplicación capaz de gestionar todos los procesos que desarrolla el colectivo de Equipos Multidisciplinarios del Sistema Penitenciario Cubano.

PALABRAS CLAVES

Equipos Multidisciplinarios (EMD), interno, vínculo[1], procesos, evaluación

[1] Familiar del interno.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Como parte de la expansión de las Tecnologías de la Informatización y las Comunicaciones (TIC) en toda la sociedad, en el año 1989, tras la orden 43/99 del Viceministro Primero del Interior, se inicia la automatización del Sistema Penitenciario Cubano "*con un sistema de control penal, que permitía la gestión de los principales datos del recluso y algunos aspectos del trabajo de la especialidad".* (1)

Comienza a desarrollarse de esta manera el Sistema Automatizado para el Control del Recluso (SACORE), culminado en el 2002, aunque no es hasta enero del 2003 que inicia su explotación. El mismo posee tres módulos principales: Control Penal, Reeducación Penal y Orden Interior, los cuales facilitan la automatización de los datos fundamentales del interno. A su vez, en los centros penitenciarios se cuenta con otras dos aplicaciones: el Sistema Automatizado de Incidencias del Departamento de Establecimientos Penitenciarios (SAIDEP) y el Sistema Automatizado de Capacidades del Departamento de Establecimientos Penitenciarios (SACDEP).

El Plan 20x50 del Ministerio del Interior (MININT) estableció la necesidad de llevar a cabo una modernización tecnológica dentro del Sistema Penitenciario Cubano, con el objetivo de contribuir al control de la legalidad y la seguridad de los internos en los Establecimientos Penitenciarios. Actualmente surge la idea de implementar un único sistema informático, que abarque todos los procesos que se llevan a cabo en las prisiones; agregándole las funcionalidades de los tres sistemas ya existentes y otras que aún no han sido automatizadas.

La Universidad de las Ciencias Informáticas (UCI), baluarte en la informatización del país, juega un rol protagónico en esta renovación, a través del proyecto Prisiones Cuba, el cual desde el año 2009 desarrolla la aplicación informática Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP); siguiendo al pie de la letra las disposiciones legales del MININT, los Órganos de Justicia y el Estado cubano (1).

Entre las funcionalidades a añadir al nuevo sistema se encuentran los procesos que realiza el Sistema de Equipos Multidisciplinarios (EMD), encargado de mantener evaluado psicológica y criminológicamente a los internos. Los procesos principales que forman parte del EMD son: Adiestramiento a Combatientes, Entrevista de Atención, Grupos de Riesgos, Investigaciones y Opinión Fundamentada.

Estos procesos se desarrollan de forma manual en la actualidad, lo cual ocasiona muchos inconvenientes para el EMD debido a que las entrevistas que les realizan a los internos o a sus familiares pudieran extraviarse y entonces estos no recibirían una respuesta necesaria, quedando inconformes con la gestión del EMD. Similar es la situación con las Opiniones Fundamentadas[2], las cuales en caso de pérdida por no estar almacenadas en un sistema, pudieran perjudicar al interno y este verse privado de un beneficio.

Así mismo a los miembros del Equipo Multidisciplinario se les dificulta llevar un estricto control sobre el plan de medidas preventivas para un interno o grupo de riesgo ya que los datos se encuentran dispersos. También al EMD le resulta complejo tener un registro completo de todas las investigaciones que se llevan a cabo y los datos de las mismas como quién las realizó, los objetivos y resultados finales.

Esta situación impide además a los integrantes del EMD realizar un seguimiento de los temas que se abordan y mantener el control de asistencias a las actividades que se realizan para la superación y el adiestramiento a los combatientes.

De manera general a los miembros del Equipo Multidisciplinario se les hace engorroso su trabajo, por desarrollarlo de manera manual y no contar con un sistema informático capaz de gestionar estos procesos. Además se corre el riesgo de archivar incorrectamente algún documento en el expediente de un interno, o que con el tiempo se deteriore una información y por estar manuscrita no pueda recuperarse y entonces haya que iniciar nuevamente las evaluaciones.

Dada la situación anterior se plantea como **problema a resolver**: ¿Cómo gestionar los procesos de evaluación psicológica y criminológica de los internos en el Sistema Penitenciario Cubano?

Se define como **objeto de estudio** la gestión de los procesos de evaluación psicológica y criminológica, planteándose como **objetivo general**: Desarrollar el Subsistema de Equipos Multidisciplinarios para la gestión de los procesos de evaluación psicológica y criminológica de los internos en el Sistema Penitenciario Cubano.

La presente investigación se enfoca en el **campo de acción** la gestión de los procesos de evaluación psicológica y criminológica de los internos en el Sistema Penitenciario Cubano.

[2] Es una opinión que redactan los miembros del Equipo Multidisciplinario cuando un interno solicita un beneficio.

A partir de la problemática planteada surge como **idea a defender:** El desarrollo del Subsistema de Equipos Multidisciplinarios permitirá la gestión de los procesos de evaluación psicológica y criminológica de los internos en el Sistema Penitenciario Cubano.

El objetivo general se divide en los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Realizar el modelo de análisis y diseño del Subsistema Equipos Multidisciplinarios.
- Realizar el modelo de implementación del Subsistema Equipos Multidisciplinarios.
- Realizar pruebas de calidad al Subsistema Equipos Multidisciplinarios.

Para dar cumplimiento a los objetivos específicos anteriormente planteados se definen las siguientes **tareas de investigación:**

1. Análisis de soluciones nacionales e internacionales con un mismo perfil.
2. Descripción de las herramientas y tecnologías para dar solución al problema.
3. Modelamiento del negocio de los procesos del Subsistema Equipos Multidisciplinarios.
4. Especificación de requisitos del Subsistema Equipos Multidisciplinarios.
5. Descripción de los casos de uso del Subsistema Equipos Multidisciplinarios.
6. Diseño de los diagramas de clases del análisis del Subsistema Equipos Multidisciplinarios.
7. Diseño de los diagramas de clases del diseño del Subsistema Equipos Multidisciplinarios.
8. Diseño de los diagramas de interacción del Subsistema Equipos Multidisciplinarios.
9. Diseño de la Base de Datos del Subsistema Equipos Multidisciplinarios.
10. Elaboración de los diagramas de componentes del Subsistema Equipos Multidisciplinarios.
11. Implementación del Subsistema Equipos Multidisciplinarios.
12. Diseño del diagrama de despliegue del Subsistema Equipos Multidisciplinarios.
13. Diseño de los casos de prueba para el Subsistema Equipos Multidisciplinarios.
14. Aplicación de las prueba al Subsistema Equipos Multidisciplinarios.

Los **métodos de investigación** a utilizar son:

Métodos Teóricos

- **Analítico – Sintético**: Se realiza una revisión de la documentación existente relacionada con los procesos del Equipo Multidisciplinario, de la misma se obtiene información que se considera necesaria para el desarrollo de la investigación. Además permite estudiar las herramientas y las tecnologías a utilizar en la solución, que aunque están previamente definidas por el equipo de desarrollo del proyecto, es importante conocer sus características y tenerlas en cuenta durante el trabajo.
- **Modelación:** Se utiliza en el diseño del Subsistema, permitiendo la creación de todos los modelos requeridos en la implementación.

Métodos Empíricos

- **Entrevista**: Se realiza con el objetivo de conocer elementos significativos del negocio que son necesarios identificar para el desarrollo del Subsistema.
- **Observación:** Permite a los autores, a medida que se va desarrollando el Subsistema, estar en contacto con todo el equipo de desarrollo e identificar el marco de trabajo del proyecto.

El presente documento cuenta de 4 capítulos:

Capítulo 1: **"Fundamentación Teórica".**

En este capítulo se analizan soluciones nacionales e internacionales, enfatizando en los procesos relacionados con los Equipos Multidisciplinarios. Además se describen las herramientas y metodologías a utilizar para el desarrollo del presente trabajo.

Capítulo 2: **"Características del sistema".**

En este capítulo se aborda todo lo relacionado con las características que presenta el Subsistema. Se realiza el modelamiento del negocio de los procesos del Subsistema de Equipos Multidisciplinarios, especificación de los requisitos y se realiza la descripción de los casos de uso.

Capítulo 3: **"Análisis y Diseño del sistema".**

En este capítulo se aborda todo lo relacionado con el análisis y diseño de la solución propuesta. Se realizan los diagramas de clases del análisis, se define la arquitectura del sistema, así como los patrones de diseño que se utilizan. Además se realiza el diseño lógico de los datos para la persistencia de la información.

Capítulo 4: **"Implementación y prueba de la solución"**.

En este capítulo se aborda todo lo relacionado con la implementación de la solución, acorde a lo propuesto en el capítulo anterior. Se realizan los diagramas de interacción y de componentes. Se diseñan los casos de pruebas para validar y verificar la calidad de la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo aborda un estudio sobre algunas soluciones nacionales e internacionales. Se describen además las herramientas y metodologías a utilizar para el desarrollo de los procesos relacionados con el diseño e implementación del Subsistema de Equipos Multidisciplinarios del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

1.2 Principales conceptos

1.2.1 Sistema Penitenciario

Se define como Sistema Penitenciario al órgano encargado de garantizar el proceso de ejecución de la sanción de privación de libertad, de la sanción de trabajo correccional con internamiento, la medida de seguridad reeducativa de internamiento y la medida cautelar de prisión provisional.

La Dirección de Establecimientos Penitenciarios del Ministerio del Interior rige al Sistema Penitenciario en nuestro país, sustentándose en la integración de principios, conceptos, procedimientos, fuerzas y medios que garantizan el funcionamiento de los centros destinados al internamiento y el tratamiento de los internos. (2)

1.2.2 Equipos Multidisciplinarios

"El Equipo Multidisciplinario en los establecimientos penitenciarios (EMD) está integrado por psicólogos, pedagogos, sociólogos, comunicadores sociales, licenciados en estudios socioculturales, juristas, trabajadores sociales y profesionales de otras ciencias sociales". (3)

El colectivo del EMD está encargado de evaluar psicológica y criminológicamente a los internos, así como realizarle diferentes Entrevistas de Atención y realizar el plan diferenciado para cada uno de los grupos de riesgos existentes. Además realizan investigaciones para determinar cualquier factor que incida en el comportamiento de los internos.

1.3 Sistemas informáticos en Sistemas Penitenciarios

Como parte del desarrollo tecnológico expandido a toda la sociedad, los Centros Penitenciarios también se han visto en la necesidad de modernizarse. Por tales motivos, varios países cuentan con sus propios

sistemas informáticos, desarrollados de conjunto con los diferentes organismos nacionales e internacionales especializados en el tema. Los mismos facilitan tener un mayor control de los datos de los internos, así como la gestión de los principales procesos que se efectúan en las prisiones.

Se estudiaron diferentes sistemas informáticos existentes en el Sistema Penitenciario de Cuba y otros países, para analizar según las funcionalidades que brindan, si podían ser parte de la presente solución.

1.3.1 Sistema Automatizado para el Control del Recluso (SACORE)

El Sistema Automatizado para el Control del Recluso (SACORE), puesto en práctica desde el año 2003 en el Sistema Penitenciario Cubano, abrió paso al desarrollo tecnológico impulsado por el MININT. Una vez instalado este sistema, se evidenció que no contemplaba otros procesos que en las prisiones se realizaban, por lo que fue necesario desarrollar otras dos aplicaciones, surgiendo de esta manera el Sistema Automatizado de Incidencias del Departamento de Establecimientos Penitenciarios (SAIDEP) y el Sistema Automatizado de Capacidades del Departamento de Establecimientos Penitenciarios (SACDEP).

El SACORE cuenta con tres módulos principales: Control Penal, Reeducación Penal y el Orden Interior, los cuales facilitan la automatización de los datos fundamentales del interno. Sin embargo, luego de 8 años de explotación, aún cuenta con requisitos incompletos o pendientes, destacándose entre ellos, los relacionados con los procesos que realiza el Equipo Multidisciplinario. (1)

1.3.2 Sistema de Gestión Penitenciaria de Venezuela (SIGEP)

El Sistema de Gestión Penitenciaria (SIGEP), desplegado en Venezuela y desarrollado en la Universidad de las Ciencias Informáticas (UCI), fue creado con el objetivo general de *"desarrollar e implantar un sistema informático que soporte las decisiones estratégicas del Ministerio del Interior y Justicia y de la Dirección General de Custodia y Rehabilitación del Recluso".* (4)

Entre los procesos que este sistema tiene en cuenta se encuentra el Control de las Entrevistas de Atención, el cual muestra el listado de las que se han realizado al interno, permite registrar cronológicamente si este se presentó o no a la misma; además de la fecha y la hora propuesta para el próximo encuentro.

Este sistema gestiona los vínculos de cada interno y almacena sus datos personales. EL SIGEP archiva además, los informes con el resultado de las evaluaciones realizadas por el Equipo Multidisciplinario de profesionales que laboran en los Centros Penitenciarios de Venezuela.

1.3.3 Sistema Penitenciario de Panamá

A inicios del año 1997 se creó en Panamá el Sistema Penitenciario, resultado de un proyecto financiado por las Naciones Unidas y el gobierno español, en coordinación con la Dirección General de Sistemas Penitenciarios de Panamá (DGSP). En esta aplicación se registran los datos de los internos detenidos en los centros penales a nivel nacional. (5)

En estos centros existe un Consejo Técnico de Establecimientos Penales y Correccionales, integrado por sociólogos, médicos, psiquiatras y especialistas. El objetivo principal de este consejo es orientar y organizar la clasificación de los penados, la enseñanza que debe impartírseles, así como la disciplina requerida para su readaptación social. Sin embargo, el sistema informático vigente, no contempla todas las actividades que este consejo realiza. (6)

1.3.4 Conclusiones parciales de las soluciones existentes en los Sistemas Penitenciarios

Luego de analizar los sistemas informáticos anteriores, se llega a la conclusión de que ninguna de esas soluciones satisfacen las necesidades requeridas respecto al trabajo que realiza el Equipo Multidisciplinario. El Sistema de Gestión Penitenciaria (SIGEP) realiza algunas funciones similares a las del EMD, pero no llega al nivel de profundización que se desea en el Sistema Penitenciario Cubano, el cual requiere poder automatizar los procesos que desarrollan profesores, psicólogos, médicos y especialistas en aras de mantener evaluado psicológica y criminológicamente a los internos.

1.4 Herramientas y tecnologías a utilizar

Para el desarrollo de la presente aplicación, se utilizaron diferentes herramientas y tecnologías. Las mismas han sido seleccionadas por parte del equipo de arquitectura del proyecto Prisiones Cuba.

1.4.1 Metodología de Desarrollo

La metodología de desarrollo a utilizar en la arquitectura del proyecto es RUP, ya que permite mayor agilidad y organización del trabajo. Además constituye la metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (7)

RUP es un proceso que se caracteriza por ser:

Dirigido por casos de uso: Los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo.

Centrado en la arquitectura: La arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto.

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, entre otras. Cada iteración mejora la anterior o le agrega nuevas funcionalidades al producto.

1.4.2 Herramientas de modelado

Las herramientas de modelado de software utilizadas para generar y representar los artefactos del Subsistema a realizar son:

Embarcadero ER/Estudio 8.0: Se utiliza ER\Studio para modelar las entidades del Subsistema Equipos Multidisciplinarios y sus relaciones. Es una herramienta para el modelado de los datos con soporte de múltiples niveles de análisis. Además se usa para el diseño y la construcción lógica y física de base de datos. Permite la transformación automática de un diseño lógico a un diseño físico para una plataforma específica de base de datos. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes. Otras de sus funcionalidades es que la documentación es basada en HTML y posee un repositorio para el modelado. (8)

Visual Paradigm 5.0: La herramienta CASE Visual Paradigm for UML Enterprise Edition 8.0 (VP Suite 5.0) emplea UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite la generación de código para Java y exportación como HTML, así como la compatibilidad con las versiones anteriores, además de la integración con herramientas de desarrollo (IDE). Una de sus características fundamentales es que suele ser multiplataforma debido a que puede ejecutarse sobre diferentes sistemas

operativos (Linux y Windows), además soporta BPMN (Modelo de Procesos de Negocio) y UML (Lenguaje Unificado de Modelado). (9)

Tiene la ventaja de presentar una interfaz de usuario de fácil uso, posibilita la realización de diagramas, la generación de los diferentes artefactos durante el desarrollo del software y realiza un control de las versiones durante todo el ciclo de trabajo.

Está diseñada para una amplia gama de usuarios, incluidos los ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquiera que esté interesado en la construcción de forma fiable a gran escala los sistemas de software con un enfoque orientado a objetos.

1.4.3 Herramientas de desarrollo

Las herramientas utilizadas para llevar a cabo el desarrollo de las funcionalidades son:

Springsource Tool Suite 2.5: Es un entorno de desarrollo integrado de código abierto, es multiplataforma el cual soporta una amplia gama de lenguajes de programación. Realiza la compilación en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS. Su principal característica para usarlo en el desarrollo del Subsistema Equipos Multidisciplinarios es su integración con el marco de trabajo de aplicaciones web definido por el proyecto Prisiones Cuba.

Contenedor Web (Apache Tomcat 6.0): Apache Tomcat es un contenedor Web escrito en Java, soporta diferentes sistemas operativos, por lo que se puede decir que es multiplataforma, siempre y cuando disponga de una máquina virtual de Java. Tomcat implementa las especificaciones de los *servlets* y de *JavaServer Pages* (JSP). (10) Se selecciona esta versión, debido a que viene embebido en el marco de trabajo de desarrollo web seleccionado por el proyecto Prisiones Cuba.

Subversion (VisualSVN Server): Software de sistema de control de versiones. Desarrollado sobre software libre bajo una licencia de tipo Apache/BSD, además se le conoce como *svn* por ser el nombre de la herramienta de línea de comandos. Se resalta como característica importante que los archivos versionados no tienen cada uno un número de revisión independiente, pero a diferencia de ellos, todo el repositorio tiene un único número de versión, que identifica un estado común de todos los archivos del repositorio en determinado punto del tiempo. El acceso al repositorio es a través de la red, lo que permite ser usado por personas que se encuentran en distintos ordenadores.

Su principal importancia radica en que varias personas pueden modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomentando así la colaboración. Permite además integrar varias interfaces a entornos de desarrollo, ejemplo de ello es el TortoiseSVN, el cual provee una integración con el explorador de Windows. Por estas facilidades se utilizará en el desarrollo del presente trabajo. (11)

1.4.4 Sistema Gestor de Base de Datos

Oracle Database 11g: Sistema Gestor de Bases de Datos con características objeto-relacionales, utilizando tecnología cliente/servidor, siendo uno de los más usados a nivel mundial. Se considera uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad y escalabilidad.

Oracle puede ejecutarse en todas las plataformas y cuenta con un lenguaje de diseño de base de datos muy completo (PL/SQL) que permite procedimientos almacenados, con una integridad referencial declarativa bastante potente. Otra considerable ventaja es que el software del servidor puede correr en multitud de sistemas operativos, además es la base de datos con más orientación hacia Internet. (12)

1.4.5 Tecnologías

Grails 1.3.7: Es un framework de aplicaciones web dinámicas construidas sobre Java y Groovy para la creación de propiedades y métodos dinámicos en los objetos de la aplicación, incluye Spring para los flujos de trabajo e inyección de dependencia, *Hibernate* para la persistencia, *Site Mesh* para la composición de la vista y *Ant* para la gestión del proceso de desarrollo. Grails sigue los principios *Don't Repeat Your Self* (No te repitas) y *Convention over Configuration* (Convención sobre configuración).

Convention over Configuration es un paradigma de programación de software que busca decrementar el número de decisiones que un desarrollador necesita hacer, ganando así en simplicidad pero no perdiendo flexibilidad por ello y DRY es una filosofía de definición de procesos que promueve la reducción de la duplicación. Ambos patrones en general proporcionan un entorno de desarrollo estandarizado y ocultan en gran parte detalles de configuración. (13)

Los *plugins* que utiliza el *framework* son:

1. dojo-release-1.3.0.

2. grails-hibernate-1.3.5.
3. grails-webflow-1.3.5.
4. grails-acegi-0.5.3.
5. grails-spring-security-core-1.0.1.
6. grails-dojo-1.3.0.
7. grails-tomcat-1.3.5.

Al crear una aplicación Grails, en ella se definen un conjunto de paquetes ordenados según su arquitectura y tipos de clases que vayan a almacenar para que al relacionarse entre sí generen la aplicación final.

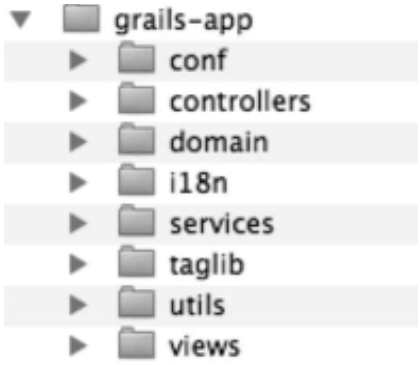


Figura 1: Estructura de paquetes de un proyecto Grails.

La carpeta grails-app contiene la mayor parte de la aplicación, donde:

- conf: contiene los archivos de configuración de la aplicación, así como las configuraciones personales de Spring e Hibernate si usted desea adicionar alguna.
- controllers: es donde se ubican todos los controladores del flujo web de la aplicación.
- domain: donde se almacenan las clases que Grails mapea como entidades en la BD.
- i18n: contiene los archivos de internacionalización de la aplicación.
- services: contiene los servicios de Grails.
- taglib: contiene la configuración de las etiquetas que el desarrollador crea para comodidad a la hora de trabajar en las vistas.

- utils: donde se almacenan las clases útiles para el desarrollo de la aplicación.
- views: se almacenarán todas las vistas de la aplicación.

Dojo 1.5: Dojo es un *framework* que contiene APIs y *widgets* (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and dropAPIs, widgetAPIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX.

Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas (*bookmarking*), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Proporciona una gama amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos.

Es de destacar que esta biblioteca es de código abierto. La licencia permite crear aplicaciones, utilizarlo en productos comerciales y modificarlo. Cuenta con el patrocinio de IBM, Google, AOL y Nexaweb.2. (14)

Es utilizado en las páginas clientes para realizar determinadas funciones en el manejo de los componentes de las vistas.

1.4.6. Lenguajes.

UML: Por sus siglas en inglés, *Unified Modeling Language*, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar. (15)

JavaScript: Lenguaje script utilizado para unir el conjunto de tecnologías usadas en la web y acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web, permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Su función principal es realizar operaciones en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML. (16)

Groovy: Lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Groovy usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde él se puede acceder directamente a todas las API existentes en Java. El *bytecode* generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java.

Todo lo anterior unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy, hacen que este lenguaje sea de muy fácil adopción para programadores Java. (17)

Java: Es un lenguaje simple, orientado a objetos, distribuido, interpretado, portable, de altas prestaciones, multitarea y dinámico. Es muy utilizado debido a su estrecha similitud con el lenguaje Groovy, siendo todo lo implementado en Java, válido en Groovy.

HTML: Utilizado debido a que es el código que posibilita la creación y edición de documentos web, basado en etiquetas, que tiene como virtud entre otras, la de poder ser implementado por código de otros lenguajes como JavaScript que amplían y mejora su capacidad original.

1.5 Conclusiones Parciales

En este capítulo se realizó un estudio detallado de algunos sistemas informáticos nacionales e internacionales, el cual evidenció que las soluciones existentes no responden a la necesidad del Sistema Penitenciario Cubano, de automatizar los procesos que desarrolla el Equipo Multidisciplinario. Se abordaron además las herramientas y tecnologías aprobadas por el proyecto Prisiones Cuba para el desarrollo de la presente aplicación.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se realiza una descripción de los principales procesos del Subsistema Equipos Multidisciplinarios, con los diagramas y la explicación del flujo básico correspondiente. Se definen además los requisitos funcionales del software y los casos de uso.

2.2 Descripción de los procesos del Subsistema Equipos Multidisciplinarios

El Subsistema Equipos Multidisciplinarios está compuesto por los módulos Entrevista de Atención, Investigaciones, Opinión Fundamentada, Grupos de Riesgo y Adiestramiento a Combatientes. Este Subsistema cuenta con 23 Casos de Uso (CU), de ellos: 10 son CU de complejidad alta, 9 de complejidad media y 4 con complejidad baja.

A continuación se realiza una descripción de los procesos de negocio Entrevista de Atención y Opinión Fundamentada del Subsistema de Equipos Multidisciplinarios con BPMN, el resto de los diagramas del Proceso del Modelo de Negocio (BPMN) se encuentra en los anexos.

2.2.1 Modelo de procesos de negocio para el módulo Entrevista de Atención

- **Descripción del proceso Entrevista de Atención**

Nombre:	Entrevista de Atención
Objetivos:	Gestionar una entrevista de atención ya sea para un interno o para un vínculo.
Evento(s) que lo generan:	Solicitud de la entrevista.
Precondiciones:	El interno debe estar en el Establecimiento Penitenciario.
Postcondiciones:	Se realizó la entrevista.
Reglas de Negocio:	Entrevistas familiares: sólo se le pueden realizar a los vínculos del interno.

Responsables:	Oficial del EMD.
Clientes internos:	Oficial del EMD. Interno.
Clientes externos:	Vínculo.
Entradas:	No aplica.
Salidas:	Documento de la Entrevista.
Actividades:	Solicitar entrevista. Definir el tipo de entrevista. Realizar la entrevista. Archivar entrevista. Registrar resultados.

Tabla 1: Descripción del proceso Entrevista de Atención.

- **Diagrama del proceso Entrevista de Atención**

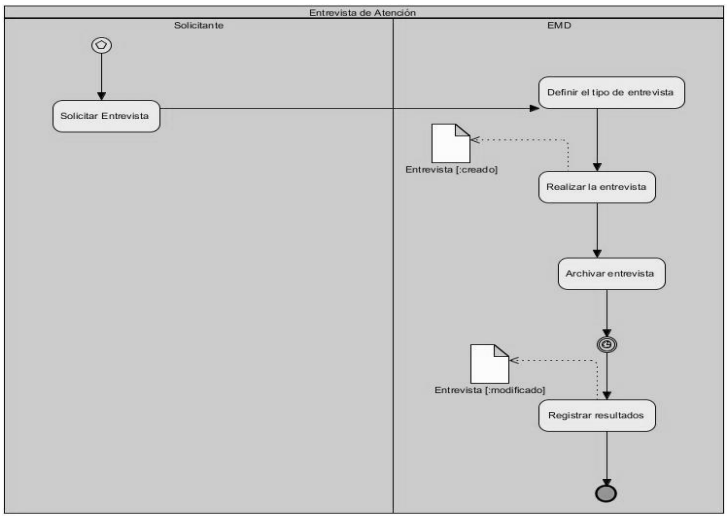


Figura 2: Diagrama del proceso Entrevista de Atención.

- **Descripción del flujo básico**

1. Solicitar entrevista: El interno, un vínculo o un Oficial de Evaluación y Diagnóstico solicita que se realice una entrevista.

Responsable: Solicitante.

Entradas: No aplicable.

Salidas: No aplicable.

2. Definir el tipo de entrevista: Seleccionar el tipo de entrevista que se va a realizar.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: No aplicable.

3. Realizar entrevista: El Oficial de EMD recoge todos los datos para la entrevista en curso.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: Documento de la Entrevista.

4. Archivar entrevista: El Oficial del EMD archiva la entrevista realizada.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: No aplicable.

5. Registrar resultados: se le registra los resultados a una entrevista en caso de que no tenga.

Responsable: Oficial del EMD.

Entradas: Documento de la Entrevista.

Salidas: Entrevista actualizada.

2.2.2 Modelo de procesos de negocio para el módulo Opinión Fundamentada

- **Descripción del proceso Opinión Fundamentada**

Nombre:	Opinión Fundamentada
Objetivos:	Realizar una Opinión Fundamentada a un interno que esté solicitando algún tipo de beneficio.
Evento(s) que lo generan:	Solicitud de la Opinión Fundamentada.
Precondiciones:	El interno debe estar en el Establecimiento Penitenciario.
Postcondiciones:	Queda registrada una Opinión Fundamentada.
Reglas de Negocio:	Internos que requieren Opinión Fundamentada: a) Delitos de Asesinato, Violación, Pederastia con Violencia de carácter múltiple o en que las víctimas resulten menores, o que el hecho cometido haya causado conmoción o adquirido notable repercusión. b) Delitos de Corrupción de Menores de similar repercusión a los anteriores. c) Delitos de Tráfico de Personas que no clasifiquen en el régimen de Mayor Severidad. d) Delitos de producción, venta, demanda, distribución, tráfico y tenencia ilícita de drogas, estupefacientes, sustancias psicotrópicas y otras de efectos similares. e) Delitos de Hurto y Sacrificio Ilegal de Ganado Mayor.
Responsables:	Oficial del EMD.
Clientes internos:	Oficial del EMD. Miembro del Consejo de Promoción.

Clientes externos:	No aplica.
Entradas:	No aplica.
Salidas:	Documento de la Opinión Fundamentada.
Actividades:	Solicitar Opinión Fundamentada. Realizar Opinión Fundamentada. Entregar al Consejo de Promoción. Archivar Opinión Fundamentada.

Tabla 2: Descripción del proceso Opinión Fundamentada.

- **Diagrama del proceso de Opinión Fundamentada**

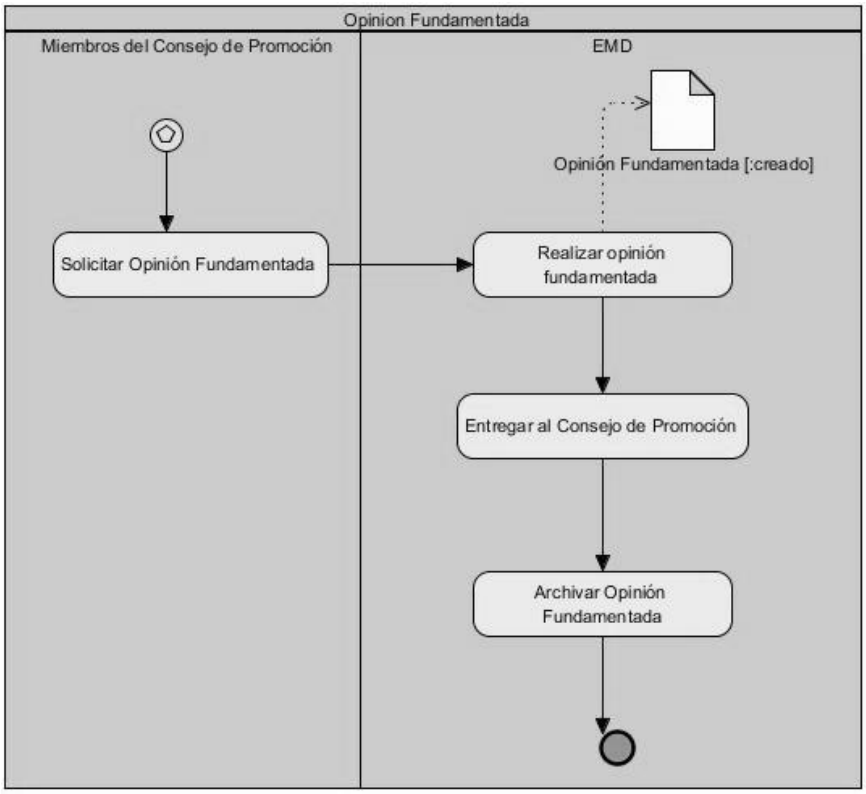


Figura 3: Diagrama del proceso Opinión Fundamentada.

- **Descripción del flujo básico**

1. Solicitar Opinión Fundamentada: Los miembros del Consejo de Promoción solicitan que se realice la opinión.

Responsable: Miembros del Consejo de Promoción.

Entradas: No aplicable.

Salidas: No aplicable.

2. Realizar Opinión Fundamentada: Se recogen los datos y se realiza la opinión.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: Opinión Fundamentada.

3. Entregar al Consejo de Promoción: Se le entrega a los Miembros del Consejo de Promoción la opinión realizada.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: Opinión Fundamentada.

4. Archivar Opinión Fundamentada.

Responsable: Oficial del EMD.

Entradas: No aplicable.

Salidas: No aplicable.

2.3 Requisitos funcionales del Subsistema

Nº	Funcionalidad	Descripción
Requisitos Funcionales Módulo Entrevista de Atención		
RF 1	Registrar entrevista de atención individual	Permite registrar una Entrevista de Atención Individual que se le realice a un interno (estas se

		clasifican en diferentes tipos).
RF 2	Registrar nuevo tipo de entrevista	Permite introducir un nuevo tipo de Entrevista de Atención que no se encuentre registrada en el sistema.
RF 3	Registrar entrevista familiar	Permite registrar la Entrevista Familiar que se le realiza a un vínculo del interno.
RF 4	Registrar nuevo vínculo	Permite registrar en el sistema un nuevo vínculo a un interno.
RF 5	Registrar resultado de entrevista de atención	Permite registrarle a una Entrevista de Atención que se encuentre en el sistema una respuesta.
RF 6	Modificar resultado de entrevista	Permite modificarle la respuesta de una entrevista registrada en el sistema.
RF 7	Consultar detalles de la entrevista	Permite consultar los detalles de una entrevista seleccionada.
RF 8	Consultar entrevistas de atención a internos	Permite consultar las Entrevistas de Atención que cumplan con determinados patrones de búsqueda.
Requisitos Funcionales Módulo Grupos de Riesgo		
RF 9	Consultar internos	Permite consultar los internos pertenecientes a un Grupo de Riesgo seleccionado.
RF 10	Registrar plan de medidas del grupo	Permite registrar un plan de medidas para un Grupo de Riesgo determinado.
RF 11	Actualizar plan de medidas del grupo	Permite actualizar el plan de medidas realizado para un Grupo de Riesgo.
RF12	Eliminar medidas del grupo	Permite eliminar una medida del listado de medidas del plan realizado para el grupo.
RF13	Consultar medidas del grupo	Permite consultar los detalles de una medida registrada para el grupo.
RF 14	Registrar plan de medidas individual	Permite registrar un plan de medidas individual para

		un interno determinado.
RF 15	Actualizar plan de medidas individual	Permite actualizar el plan de medidas realizado para un determinado interno.
RF16	Eliminar medidas individual	Permite eliminar una medida del listado de medidas registrada para un interno.
RF17	Consultar medidas individual	Permite consultar los detalles de una medida registrada para un interno.
Requisitos Funcionales Módulo Adiestramiento a Combatientes		
RF 18	Registrar Programa de Adiestramiento	Permite registrar el Programa de Adiestramiento anual.
RF 19	Actualizar Programa de Adiestramiento	Permite actualizar el Programa de Adiestramiento anual realizado.
RF 20	Adicionar tema	Permite adicionarle un tema al Programa de Adiestramiento realizado.
RF 21	Actualizar tema	Permite actualizar un tema adicionado al Programa de Adiestramiento.
RF 22	Eliminar tema	Permite eliminar un tema al Programa de Adiestramiento.
RF 23	Registrar actividad	Permite registrarle una actividad al tema.
RF 24	Actualizar actividad	Permite actualizar una actividad registrada en el sistema.
RF 25	Eliminar actividad	Permite eliminar una determinada actividad.
RF 26	Eliminar responsable	Permite eliminar el responsable de una actividad.
RF 27	Consultar Programa de Adiestramiento	Permite consultar el Programa de Adiestramiento por un determinado patrón de búsqueda.
RF 28	Registrar ejecución de actividad	Permite registrar la ejecución de una determinada actividad.

RF 29	Consultar actividades por funcionarios	Permite consultar las actividades en las que ha participado un determinado funcionario.
RF 30	Consultar funcionarios por tema	Permite consultar los funcionarios que han participado en un determinado tema.
Requisitos Funcionales Módulo Investigaciones		
RF 31	Registrar Investigaciones Científicas	Permite registrar Investigaciones Científicas en el sistema.
RF 32	Consultar Investigaciones Operativas	Permite consultar el listado de Investigaciones Operativas que cumplan con determinados patrones de búsquedas en caso de que haya.
RF 33	Consultar Investigaciones Científicas	Permite consultar el listado de Investigaciones Científicas que cumplan con determinados patrones de búsquedas en caso de que haya.
RF34	Registrar Investigaciones Operativas	Permite registrar Investigaciones Operativas realizadas.
RF 35	Modificar Investigaciones Operativas	Permite modificar los datos de una determinada Investigación Operativa.
RF 36	Registrar acciones	Permite registrarle acciones a una Investigación Operativa en caso de tener.
RF 37	Actualizar acciones	Permite actualizar una determinada acción ya registrada en el sistema.
RF 38	Eliminar acciones	Permite eliminar una determinada acción.
RF 39	Consultar acciones	Permite consultar los detalles de una determinada acción.
RF 40	Consultar Investigaciones Científicas	Permite consultar las investigaciones científicas realizadas.
RF 41	Registrar evento	Permite registrar un nuevo evento que no se encuentre en el sistema registrado.

RF 42	Eliminar evento	Permite eliminar un evento creado.
Requisitos Funcionales Módulo Opinión Fundamentada		
RF 43	Registrar Opinión Fundamentada	Permite registrar una Opinión Fundamentada en el sistema.
RF 44	Consultar Opinión Fundamentada	Permite consultar el listado de Opinión Fundamentada realizada a un determinado interno.

Tabla 3: Requisitos Funcionales del sistema.

2.4 Definición de los casos de uso

A continuación se presentan los diagramas de Casos de Uso de los módulos Entrevista de Atención y Opinión Fundamentada del Subsistema de Equipos Multidisciplinarios, así como la descripción detallada de los Casos de Usos Registrar Entrevista familiar y Registrar Opinión Fundamentada.

2.4.1 Diagrama de Casos de Uso del módulo Entrevista de Atención

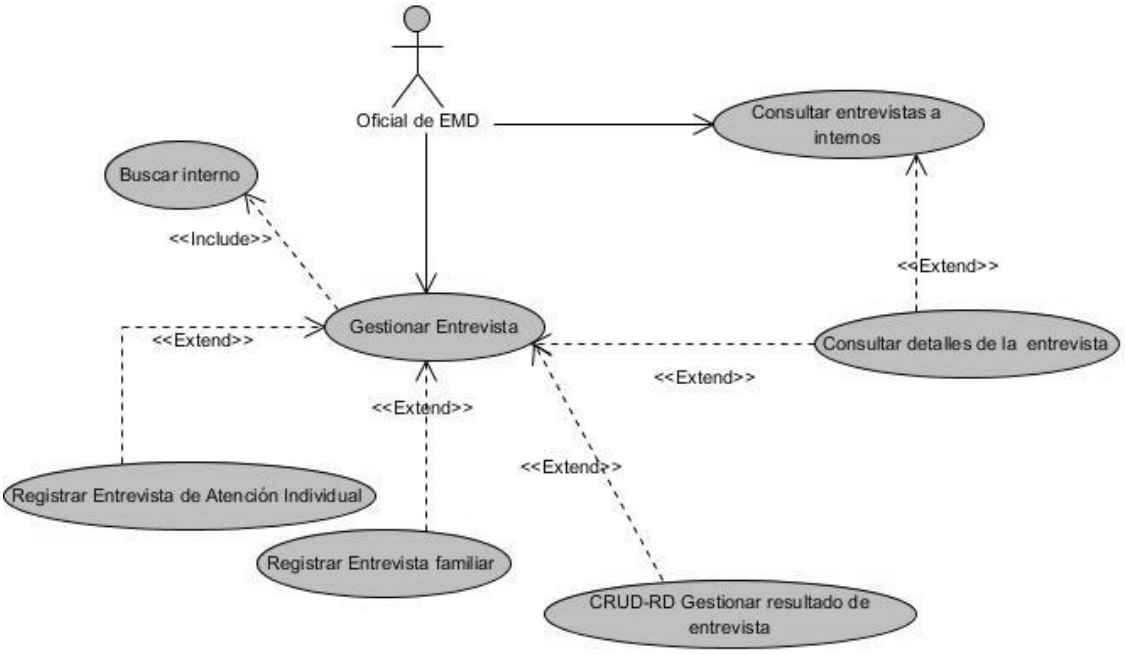


Figura 4: Diagrama de Caso de Uso de Entrevista de Atención.

2.4.2 Descripción del Caso de Uso Registrar Entrevista familiar.

Objetivo	Registrar una entrevista familiar
Actores	Oficial del EMD
Resumen	El Oficial del EMD introduce los datos de la entrevista familiar y selecciona el vínculo al que se le va a realizar la entrevista, en caso de que este vínculo no exista, el sistema brinda la opción de registrar un nuevo vínculo, el sistema guarda los datos y registra la entrevista y termina el CU.
Complejidad	Media
Prioridad	Crítico
Precondiciones	El Oficial del EMD debe estar autenticado en el sistema
Postcondiciones	Queda registrada la entrevista familiar
Flujo de eventos	
Flujo básico Registrar Entrevista familiar	

	Actor	**Sistema**
1.		Muestra los vínculos del interno que sean familiares. El listado contiene: • Primer nombre. • Segundo nombre. • Primer Apellido. • Segundo Apellido. • Parentesco. Y permite seleccionar uno o registrar un nuevo vínculo.
2.	Si oprime la pestaña “Nuevo vínculo”, ver el flujo alterno 2a. “Nuevo vínculo”. Si selecciona un vínculo y oprime el botón “Nuevo”.	
3.		Muestra los siguientes campos a llenar:

		• Motivo de la entrevista. • Resumen. • Recomendaciones.
4.	Introduce los datos y oprime el botón “Registrar”. Si oprime el botón “Cancelar” ver el flujo alterno 4a. “Cancelar registro de entrevista”.	
5.		Valida los datos introducidos y registra la entrevista familiar. Si falta alguno de los siguientes datos obligatorios: • Motivo de la entrevista. • Resumen. • Recomendaciones. ver el flujo alterno 3a. “Faltan datos obligatorios”.
6.		Asigna al campo estado de la entrevista el valor “No respondida”.
7.		Termina el CU.
Flujos alternos		
***Cancelar**		
	Actor	**Sistema**
1.	Oprime el botón “Cancelar”.	
2.		No almacena la información y regresa al paso 4 del flujo básico “Gestionar Entrevista de Atención”.
2a. Nuevo vínculo		
1.		Muestra los campos para registrar un

		nuevo vínculo: • CI. • Parentesco.
2.	Registra los datos y oprime el botón “Nuevo”. Si selecciona la opción “Es externo”, ver el flujo alterno 2a.2a “Registrar vínculo externo”. Si oprime el botón “Cancelar”, ver el flujo alterno 2a.2b “Cancelar registro del vínculo”.	
3.		Valida los datos, registra el nuevo vínculo y lo adiciona a la lista de vínculos. Regresa al paso 1 del flujo básico “Registrar Entrevista familiar”. Si falta alguno de los siguientes datos obligatorios: • CI. • Parentesco. ver flujo alterno 2.a. 3a.”Faltan datos obligatorios”. Si existen datos incorrectos ver el flujo alterno 2.a. 3b. “Datos incorrectos”.
2a.2a Registrar vínculo externo		
	Actor	**Sistema**
1.		Muestra los siguientes campos: • Primer nombre. • Segundo nombre. • Primer apellido. • Segundo apellido.
2.	Introduce los datos y oprime el botón “Nuevo”.	

3.		Valida los datos y registra al nuevo vínculo al sistema y lo adiciona a la lista de vínculos. Si falta algunos de los siguientes datos obligatorios: • Primer nombre. • Primer apellido. • Segundo apellido. ver flujo alterno 2.a. 3a.”Faltan datos obligatorios”.
2a.2b Cancelar registro del vínculo		
	Actor	**Sistema**
1.		No almacena la información y regresa al paso 1del flujo básico “Registrar Entrevista familiar”.
2a. 3a. Faltan datos obligatorios		
	Actor	**Sistema**
1.		Muestra el mensaje de error “Faltan datos obligatorios” y señala los campos obligatorios. Regresa al paso 1 del flujo alterno 2a. “Nuevo vínculo”.
2a. 3b. Datos incorrectos		
	Actor	**Sistema**
		Muestra el mensaje de error “Los dígitos del mes y el día son incorrecto”. Regresa al paso 1 del flujo alterno 2a. “Nuevo vínculo”.

4a. Cancelar registro de entrevista			
	Actor		**Sistema**
1.			No almacena la información y regresa al paso 1del flujo básico “Registrar Entrevista familiar”.
Relaciones	**CU Incluidos**	No aplicable.	
	CU Extendidos	No aplicable.	
Requisitos no funcionales	No aplicable		
Asuntos pendientes	No aplicable		

Tabla 4: Descripción del Caso de Uso Registrar Entrevista familiar.

2.4.3 Diagrama de Casos de Uso del módulo Opinión Fundamentada

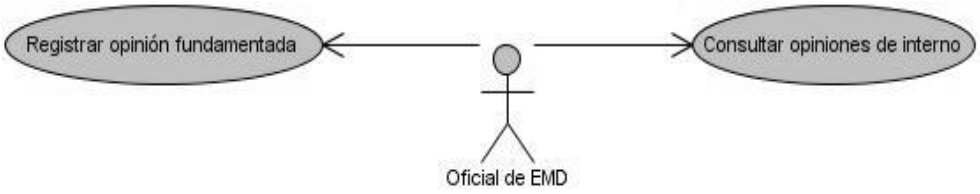


Figura 5: Diagrama de Caso de Uso de Opinión Fundamentada.

2.4.4 Descripción del Caso de Uso Registrar Opinión Fundamentada

Objetivo	Registrar la Opinión Fundamentada.
Actores	Oficial de EMD
Resumen	El caso de uso inicia cuando el actor selecciona la opción de registrar la Opinión Fundamentada con el objetivo de que quede almacenada la misma. El sistema registra la información y termina el CU.
Complejidad	Media

Prioridad	Crítico
Precondiciones	El Oficial de EMD debe estar autenticado y con los permisos para registrar la Opinión Fundamentada. El sistema determina los internos a los que se le van a realizar la Opinión Fundamentada.
Postcondiciones	Se registra la Opinión Fundamentada.

Flujo de eventos

Flujo básico Registrar Opinión Fundamentada

	Actor	**Sistema**
1.	El Oficial de EMD selecciona la opción en el menú “Equipo Multidisciplinario”,” Opinión Fundamentada”, “Registrar Opinión Fundamentada”.	
2.		El sistema muestra del módulo Gestión del Consejo de Promoción, el listado de los internos a los que se le van a realizar la Opinión Fundamentada. El listado contiene: Del interno: • Primer nombre. • Segundo nombre. • Primer apellido. • Segundo apellido. • Número de expediente. Del beneficio: • Tipo (es el beneficio que se le está analizando al interno).
3.	El Oficial de EMD selecciona la solicitud y oprime el botón “Registrar Opinión”.	
4.		El sistema muestra los siguientes campos: • Fecha.

		• Informe. Y el siguiente texto para completar el resultado: Somos de la opinión de: • Resultado. el/la “Tipo(es el beneficio que se le está analizando al interno)”.
5.	El Oficial de EMD introduce los datos solicitados y oprime el botón “Cerrar”.	
6.		El sistema valida los datos introducidos y registra los datos. Si hay datos incorrectos ver flujo alterno 6a. “Datos incorrectos”. Si alguno de los siguientes campos obligatorios está vacío: • Fecha. • Informe. • Resultado. ver el flujo alterno 6b “Datos obligatorios”.
7.		Termina el CU.

Flujos alternos		
*** Cancelar**		
	Actor	**Sistema**
1.	El Oficial de EMD oprime el botón “Cancelar”.	
2.		Regresa al paso 2 del flujo básico “Registrar Opinión Fundamentada”.
6a. Datos incorrectos		
	Actor	**Sistema**
1.		El sistema muestra un mensaje de error “Existen datos incorrectos”.

2.	El Oficial de EMD oprime el botón “Cerrar”.	
3.		Regresa al paso 2 del flujo básico “Registrar Opinión Fundamentada”.
6b. Datos obligatorios		
	Actor	**Sistema**
1.		El sistema muestra un mensaje de error “Introduzca los datos obligatorios” y señala los campos obligatorios que no fueron introducidos.
2.	El Oficial de EMD oprime el botón “Cerrar”.	
3.		Regresa al paso 2 del flujo básico “Registrar Opinión Fundamentada”.

Relaciones	**CU Incluidos**	No aplicable
	CU Extendidos	No aplicable
Requisitos no funcionales	No aplicable	
Asuntos pendientes	No aplicable	

Tabla 5: Descripción del Caso de Uso Registrar Opinión Fundamentada.

2.5 Conclusiones Parciales

En el presente capítulo se describieron los procesos del negocio de los módulos Entrevista de Atención y Opinión Fundamentada. Se detallaron los requisitos funcionales del Subsistema para cada uno de los módulos, así como la descripción detallada de los Casos de Uso Registrar Entrevista familiar y Registrar Opinión Fundamentada.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En el presente capítulo se realizan los diagramas de Clases del Análisis, Secuencia y Clases del Diseño para cada uno de los módulos del Subsistema Equipos Multidisciplinarios. Se fundamenta la arquitectura del sistema, así como todo el marco de trabajo de desarrollo de la aplicación. Se detallan los patrones de diseño utilizados en la presente solución. Se diseña la base de datos y se describen las tablas.

3.2 Análisis

A continuación se presentan los diagramas de Clases del Análisis de los Casos de Uso Registrar Entrevista familiar del módulo Entrevista de Atención y Registrar opinión fundamentada del módulo Opinión Fundamentada del Subsistema de Equipos Multidisciplinarios.

3.2.1 Diagramas de clases de análisis

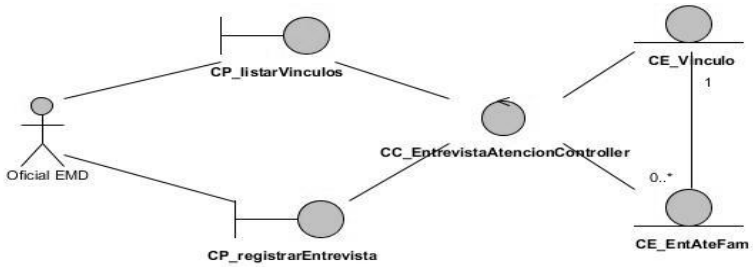


Figura 6: Diagrama de Clases del Análisis del Caso de Uso Registrar Entrevista familiar.

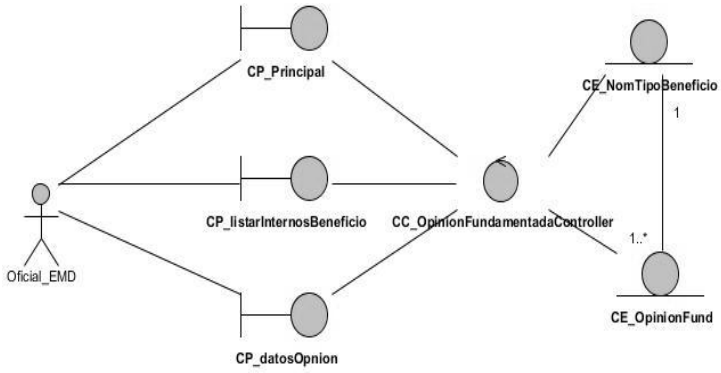


Figura 7: Diagrama de Clases del Análisis del Caso de Uso Registrar opinión fundamentada.

3.3 Arquitectura del sistema

Grails no solo es un marco de trabajo de desarrollo web de la plataforma Java. Como otros, también implementa el conocido patrón Modelo Vista Controlador (MVC) en su capa web (Figura 8). Su

arquitectura está compuesta por la Capa Web que contiene las vistas y los controladores, la Capa de Servicios que está conformada por los servicios y la Capa de Dominio que contiene las clases del dominio. Es considerado como una plataforma de trabajo donde convergen varias tecnologías, tales como Hibernate, SiteMesh y Spring.

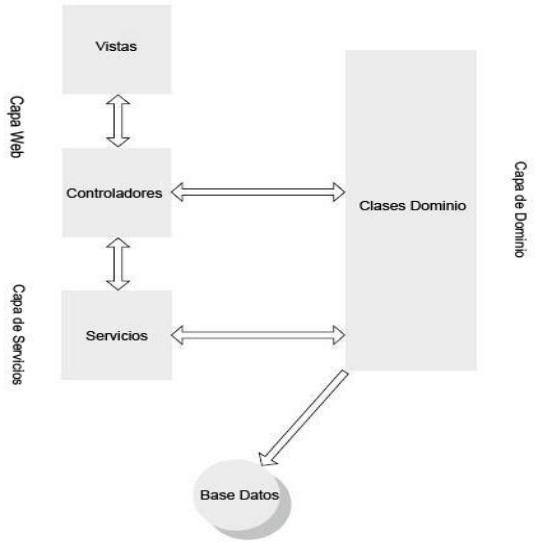


Figura 8: Arquitectura en capas de Grails.

El Subsistema EMD es un plugin del SIDEP y se ejecuta sobre la máquina virtual de Java (JVM) aprovechando la potencialidad de que esta permite implementar en lenguaje Groovy. Se usará Grails como framework para desarrollo de aplicaciones web. J2EE soporta el lenguaje Groovy por lo que esto posibilita el aprovechamiento de las ventajas de la flexibilidad de Groovy y la robustez de Java. Grails presenta dos principios de gran utilidad para el diseño como son: Don'tRepeatYourself (DRY) y ConventionOverConfiguration (CoC).

El framework Grails utiliza Spring para los flujos de trabajo e inyección de dependencias.

Hibernate para la persistencia, GORM (GrailsObjectReelationalMapping) para la abstracción de mapeo de objeto-relacional siendo esta una excelente herramienta y fácil de usar; en el proyecto se decidió usar para la interfaz visual.

El Subsistema EMD está compuesto por clases controladoras, vistas, entidades y servidoras como se muestra en la figura 7.

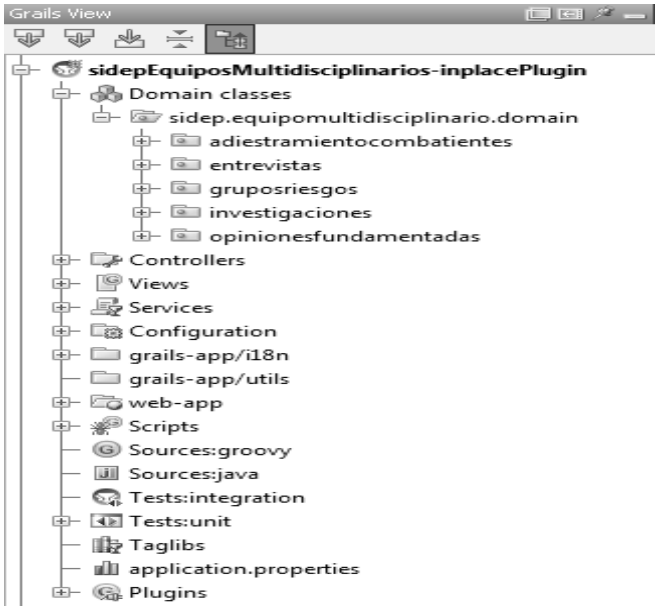


Figura 9: Arquitectura de la aplicación Subsistema Equipos Multidisciplinarios**.**

3.3.1 Entidades

Las entidades o clases del dominio componen esta capa, a través de Hibernate harán que persista la información que se introduce en la aplicación. En ellas persistirán los datos con los que el cliente interactúa y utiliza los servicios para realizar el intercambio de información con él. Grails utiliza GORM (GrailsObjectRelationalMapping) un gestor de persistencia escrito en Groovy para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos dinámicos que se crean en tiempo de ejecución para cada entidad y esto facilita enormemente la búsqueda.

GORM (GrailsObjectRelationalMapping) está construido sobre Hibernate, una herramienta de mapeo objeto-relacional que se encarga de relacionar las entidades con las tablas de la base de datos y las propiedades de las entidades con los campos de las tablas.

Una vez creado el modelo de datos Grails podemos solicitar una técnica llamada Scaffolding que genera las vistas y los controladores a partir del modelo de datos y permite realizar operaciones CRUD.

GORM nos permite restringir los valores que pueden asignarse a las propiedades de cada entidad, mediante una propiedad estática con el nombre constraints (restricciones).

3.3.2 Los controladores

En una aplicación MVC los controladores son los componentes responsables de gestionar la lógica del negocio y de actualizar la vista para mostrar al cliente como ha quedado el modelo de datos, también son

responsables de interceptar las peticiones HTTP del navegador y generar la respuesta correspondiente ya sea en el propio controlador o delegando el trabajo en otro o en una vista GSP.

3.3.3 Vistas: Groovy Server Pages

Las vistas son las encargadas de mostrar al usuario el estado actual del modelo de datos y de mostrarle al usuario las posibles acciones a realizar para que el elija lo que desea hacer a continuación.

Una vista en Grails es un archivo con extensión GSP (Groovy Server Pages). Mediante el método render podemos decidir durante la ejecución de la aplicación que vista queremos procesar y mostrar al cliente.

Utilizando código Groovy con el formato ${expresión} serán reemplazadas por su valor antes de ser enviadas al cliente.

3.3.4 Los servicios

La capa de los servicios es la encargada de implementar la lógica del negocio. Al utilizarlos para este propósito se reparten las responsabilidades, consiguiendo por un lado componentes más pequeños y fáciles de mantener y por otro lado la posibilidad de reutilizar el código en mayor medida.

Por defecto todos los servicios son Singleton (solo existe una instancia del mismo) y se inyecta en todos los artefactos que declaren la variable correspondiente.

3.4 Patrones de diseño

Los patrones de diseño pretenden:

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

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

En el diseño y la implementación realizada del Subsistema Equipos Multidisciplinarios se hizo uso de diferentes patrones de diseño.

3.4.1 Patrones GRASP

Entre los patrones GRASP (*Generales de Software para Asignación de Responsabilidades*) se utilizaron los patrones Alta Cohesión y Bajo Acoplamiento.

Estos patrones describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, es decir las obligaciones de un objeto en cuanto a su comportamiento. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (18)

- **Alta Cohesión**

Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

Ejemplos de una baja cohesión son las clases que hacen demasiadas cosas.

Ejemplos de buen diseño se producen cuando se crean los denominados "paquetes de servicio" o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia).

En Java, pensar en interfaces fuerza a que los sistemas sigan los principios de alta cohesión.

- **Bajo Acoplamiento**

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas se dificulta la reutilización de código. El bajo acoplamiento permite que si se realiza una modificación en una de las clases se afecten la menor cantidad de clases relacionadas con esta al haber poca dependencia.

3.4.2 Patrones GOF

Los patrones **GoF** (*Gang of Four*) son una serie de posibles soluciones a problemas que suelen ser comunes en el desarrollo del software. Estos se clasifican en 3 grandes categorías: **creacionales** (conciernen al proceso de creación de objetos), **estructurales** (tratan la composición de clases y objetos) y de **comportamiento** (caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos). (18)

Para el desarrollo del Subsistema se hizo uso de la categoría creacionales, utilizando el patrón Singleton.

- **Singleton**

Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Este criterio tiene como inconveniente que no se puede guardar información que sea “privada” de una petición en el servicio porque todos los controladores verían la misma instancia y el mismo valor. Este comportamiento se puede variar utilizando una variable scope con valor session, permitiendo crear una instancia nueva del servicio para cada sesión de usuario.

3.4.3 Inversión de control

El patrón Inversión de control (Inversion of Control en inglés, IoC) es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedurecalls) o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretos, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Definición: El flujo habitual se da cuando es el código del usuario quien invoca a un procedimiento de una biblioteca.

La inversión de control sucede cuando es la biblioteca la que invoca el código del usuario.

Típicamente sucede cuando la biblioteca es la que implementa las estructuras de alto nivel y es el código del usuario el que implementa las tareas de bajo nivel.

La utilización de interfaces y la aparición de los frameworks han popularizado este término. De hecho es el concepto central del Framework de Spring, ya que implementa un "Contenedor" que se encarga de gestionar las instancias (así como sus creaciones y destrucciones) de los objetos del usuario. Por tanto las aplicaciones que utilicen el framework de Spring (no Spring propiamente dicho) utilizarán Inversión de Control.

3.4.4 Patrón arquitectónico. Modelo – Vista – Controlador (MVC)

Respecto al diseño, Grails sigue el Patrón Modelo-Vista-Controlador (MVC), el cual provee un mecanismo para construir una capa de presentación, estableciendo que los componentes de una aplicación web deben organizarse en 3 capas distintas según su misión:

- **Modelo o capa de datos:** contiene los componentes que representan y gestionan los datos manejados por la aplicación. En el caso más típico, los objetos encargados de leer y escribir en la BD.
- **Vista o capa de presentación**: los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos y presentarle las distintas acciones disponibles.
- **Capa de control**: contiene los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica de negocio sobre el modelo de datos, y determinan que vista debe mostrarse a continuación.

Cuando se dice que los componentes de la capa de control "gestionan la aplicación de la lógica de negocio", se refiere a que son responsables de que ésta se aplique, lo cual no quiere decir que se implementará la lógica de los casos de uso en los controladores. Normalmente esta lógica estará implementada en la siguiente capa:

- **Capa de Servicios:** contiene los componentes encargados de implementar la lógica de negocio de la aplicación.

3.5 Diseño

3.5.1 Diagramas de clases del diseño

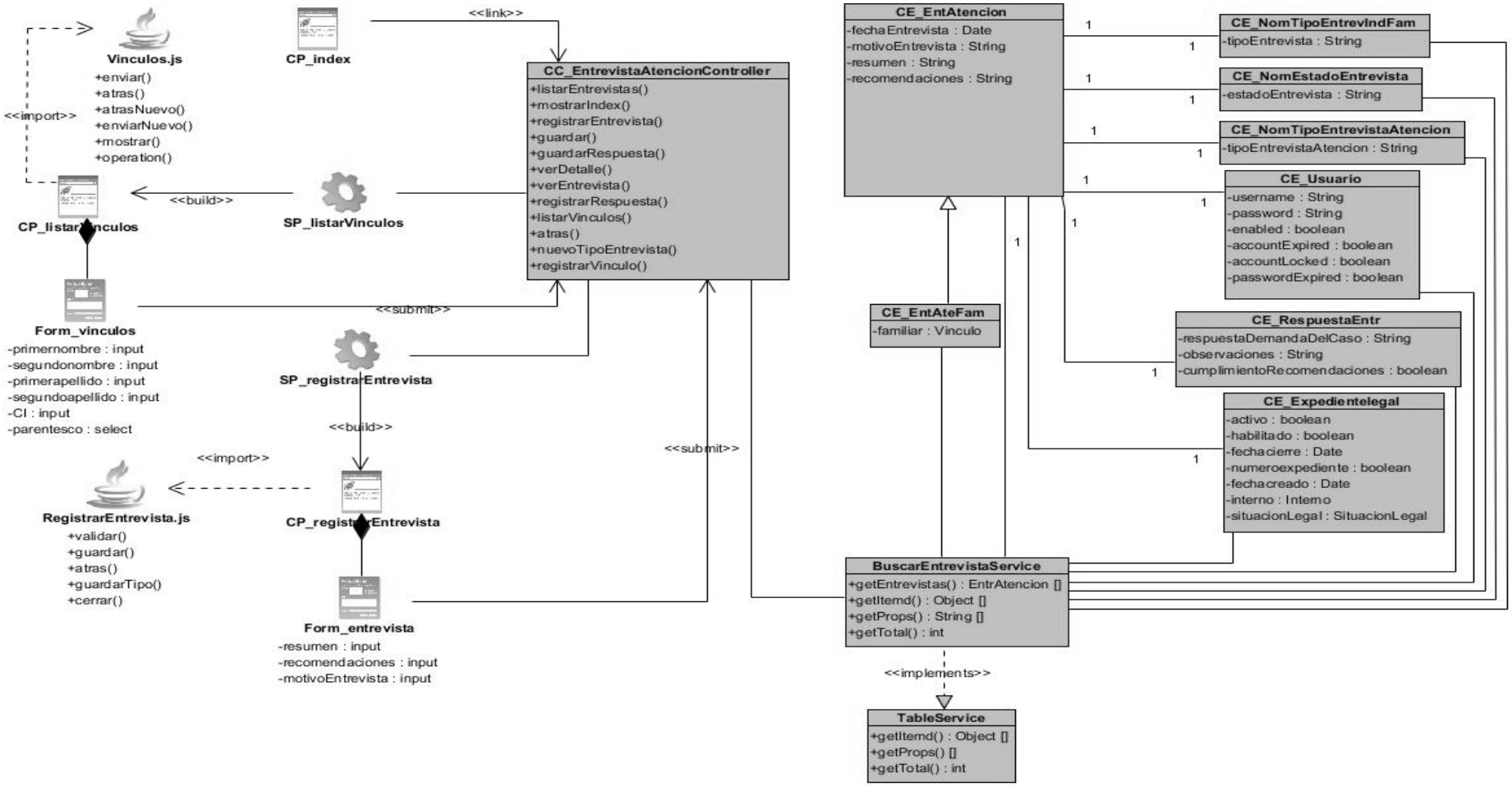


Figura 10: Diagrama de Clases del Diseño del Caso de Uso Registrar Entrevista Familiar.

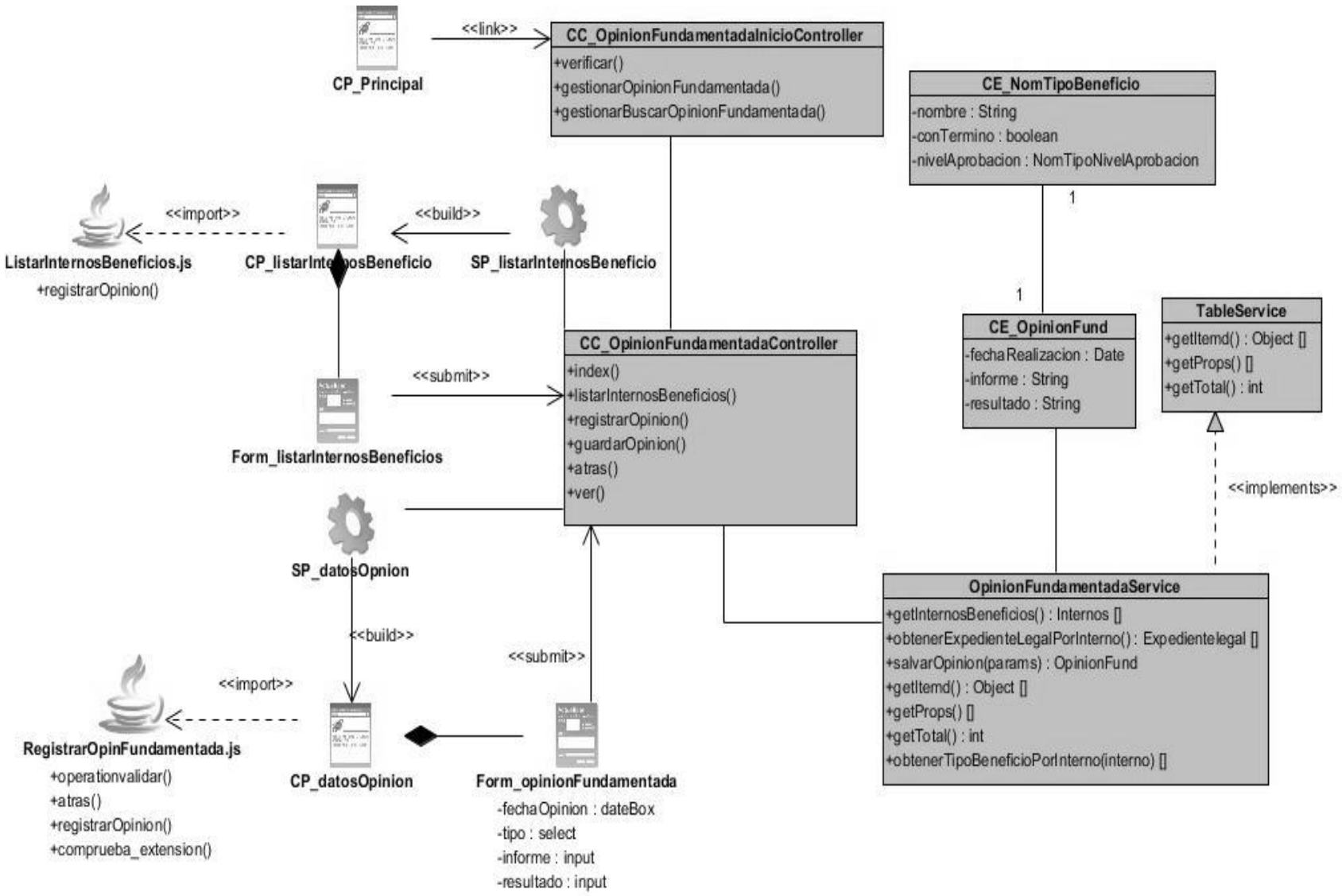


Figura 11: Diagrama de Clases del Diseño del Caso de Uso Registrar Opinión Fundamentada.

3.5.2 Diagramas de secuencia

Los diagramas de secuencia se diseñan con el objetivo de representar el flujo de actividades que se realiza en cada sección de los Casos de Uso.

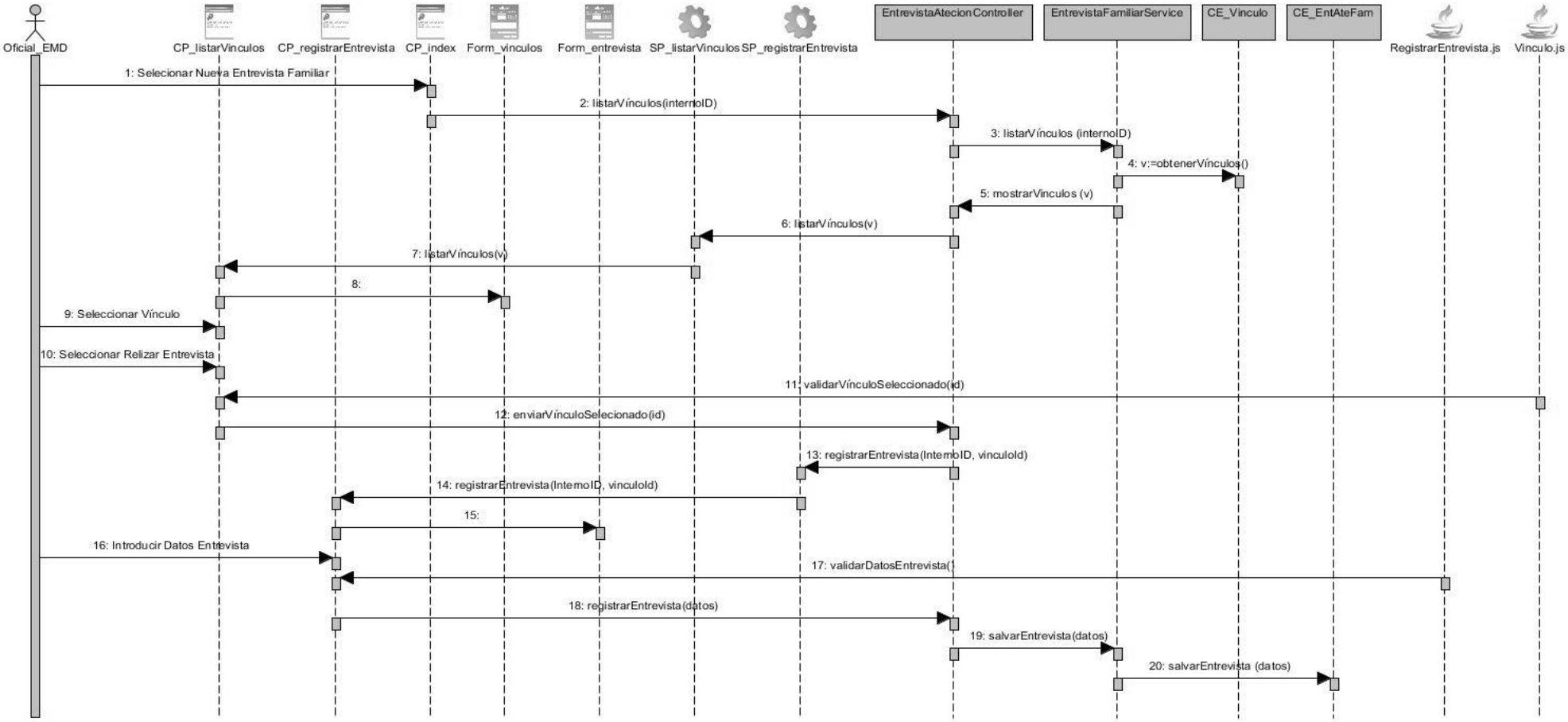


Figura 12: Diagrama de Secuencia del Caso de Uso Registrar Entrevista familiar.

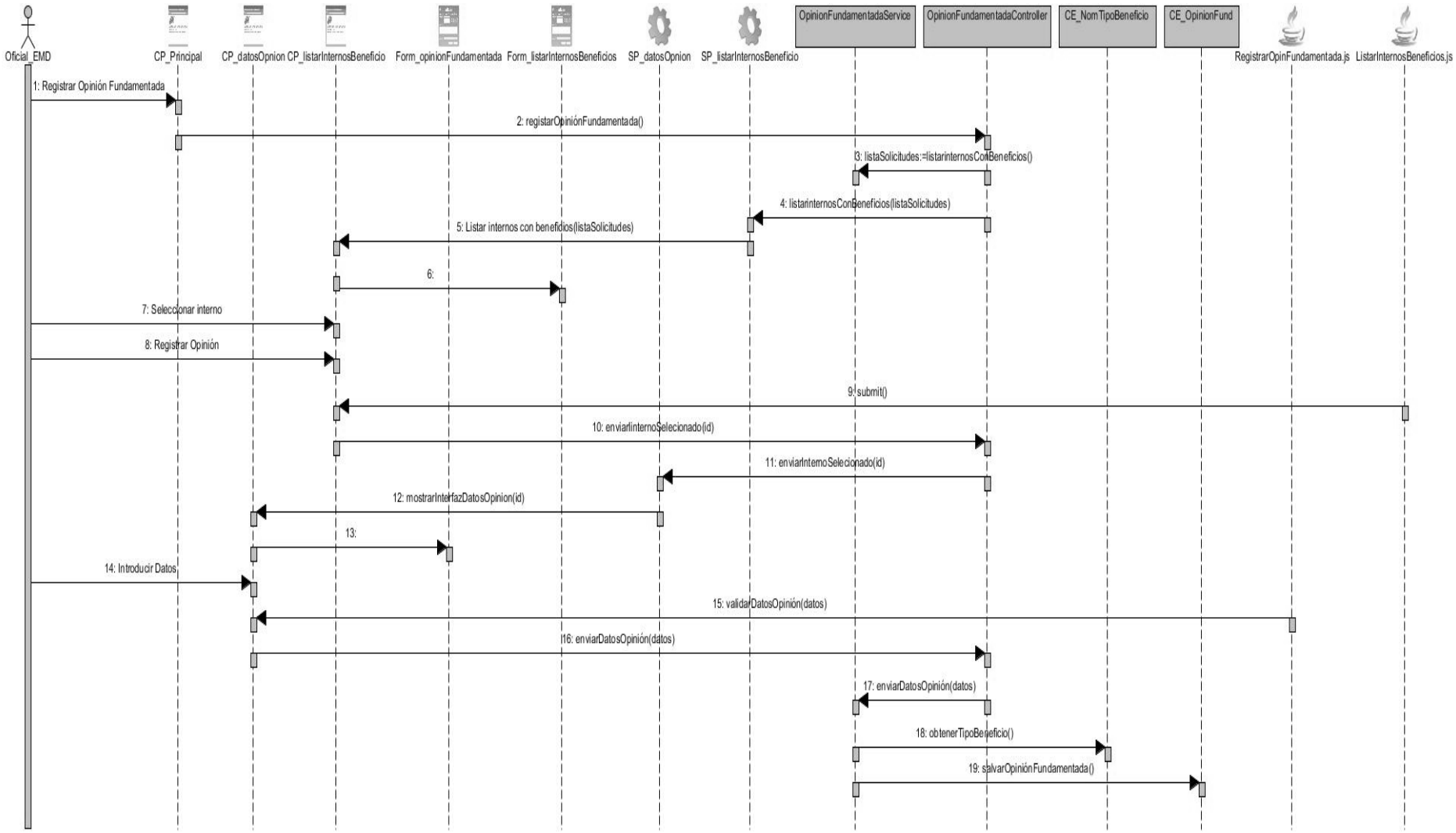


Figura 13: Diagrama de Secuencia del Caso de Uso Registrar Opinión Fundamentada.

3.5.3 Diseño de la Base de Datos

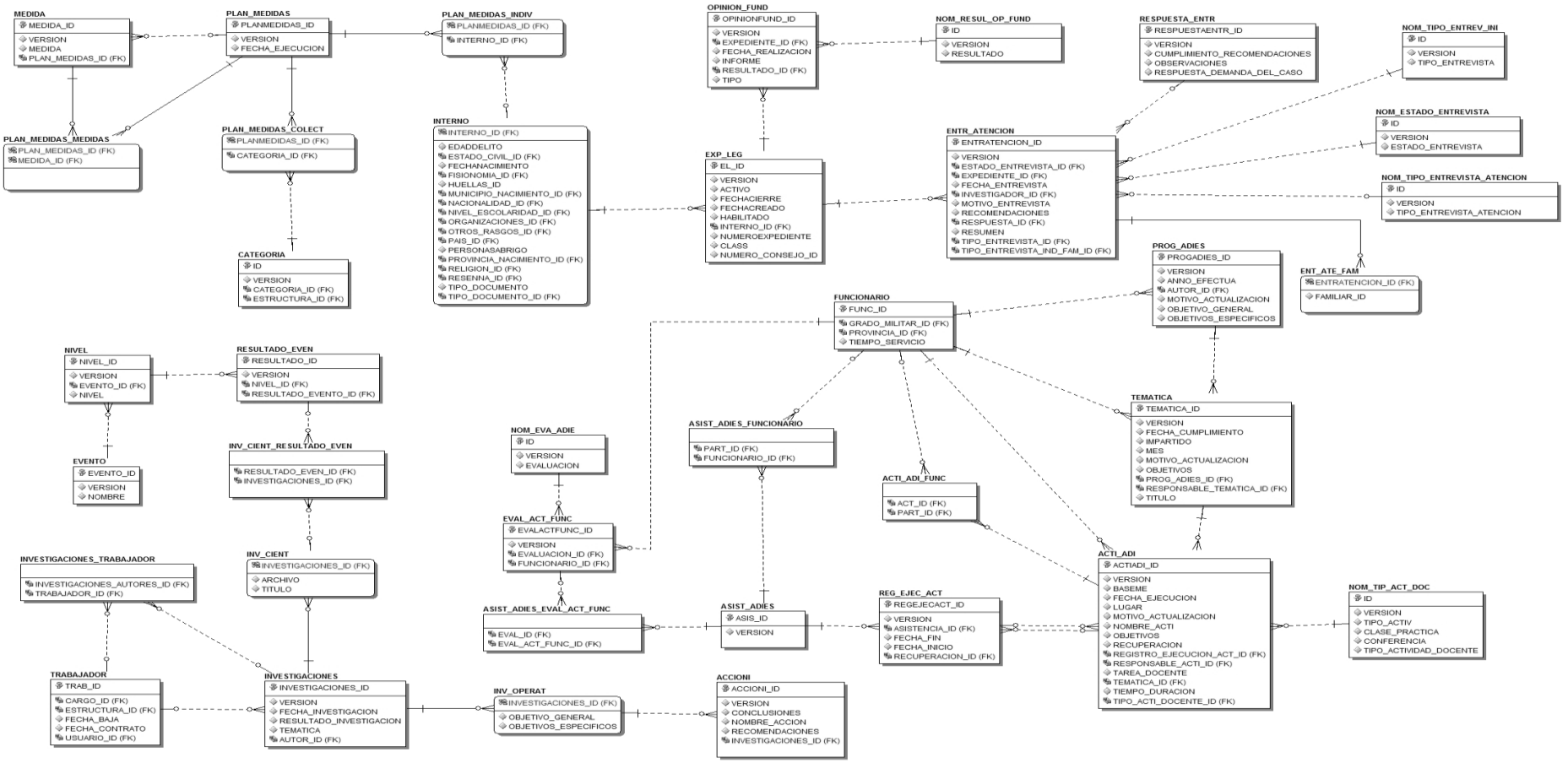


Figura 14: Diseño de la Base de Datos del Subsistema.

3.5.4 Descripción de las tablas de la Base de datos

Nombre	**ENT_ATE_FAM**	
Descripción	Almacena los datos de una Entrevista de Atención Familiar.	
Atributo	Tipo	Descripción
ENTATEFAM_ID	NUMBER(19)	Guarda el id de la entrevista de atención a un familiar, no puede ser nulo.
FAMILIAR_ID	NUMBER(19)	Guarda el id del familiar del interno, no puede ser nulo.

Tabla 6: Tabla de la entidad Entrevista Familiar.

Nombre	**OpinionFound**	
Descripción	Almacena los datos una Opinión Fundamentada.	
Atributo	**Tipo**	**Descripción**
OPINIONFUND_ID	NUMBER(19)	Guarda el id de la opinión fundamentada, no puede ser nulo.
EXPEDIENTE_ID	NUMBER(19)	Guarda el id del expediente legal del interno, no puede ser nulo.
FECHA_REALIZACION	DATE	Guarda la fecha en que se realizó la opinión, no puede ser nulo.
INFORME	VARCHAR2(10000)	Guarda un informe de la opinión, no puede ser nulo.
RESULTADO	VARCHAR2(10000)	Guarda los resultados de la opinión si es aceptado o denegado, no puede ser nulo.
TIPO	VARCHAR2(10000)	Guarda el tipo de beneficio del interno, no puede ser nulo.

Tabla 7: Tabla de la entidad Opinión Fundamentada.

La descripción de las tablas del resto de los módulos se encuentra en el Anexo 1.

3.6 Conclusiones parciales

En este capítulo se realizó el análisis y diseño del Subsistema Equipos Multidisciplinarios. Se detallaron los diagramas de Clases del Análisis, Secuencia y Clases del Diseño para cada uno de los módulos del Subsistema Equipos Multidisciplinarios. Se profundizó en la arquitectura del sistema y todos los elementos a tener en cuenta para la implementación de la solución, incluyendo la utilización de los patrones de diseño, el diseño de la base de datos, así como la descripción de las tablas, lo cual permite una mayor organización de los datos y las entidades asociadas a cada uno de los módulos.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En el presente capítulo se realiza la implementación del Subsistema Equipos Multidisciplinarios. Se define la organización de las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación. Se realizan los diagramas de componentes y despliegue. Además se incluyen resúmenes de los casos de prueba, especificando los niveles y tipos, y se muestran los resultados de las pruebas de caja negra.

4.2 Implementación

Esta disciplina de RUP describe cómo los elementos de diseño se implementan en componentes formando el modelo de implementación. Este modelo es considerado el artefacto más significativo dentro 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 y el diagrama de despliegue. El resultado final de esta fase es un sistema ejecutable. (14)

4.2.1 Diagrama 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 disponibles en el entorno de implementación y la relación entre cada uno de ellos. Es considerado un diagrama de clase a gran escala.

El Subsistema Equipos Multidisciplinarios utiliza varios componentes del Subsistema sidepCore tales como: la plantilla del SIDEP main.gsp, el componente main.css para el estilo CSS, el componente App.js que facilita funcionalidades comunes para todos los módulos del SIDEP y Dojo.js para la capa de presentación. El componente SidepAdminTagLib.gsp dentro de sidepAdmin brinda funcionalidades que utilizan los componentes en las vistas. Se relaciona además con los subsistemas sidepTratamientoEducativo y sidepRegistroLegal a través de las interfaces de estos.

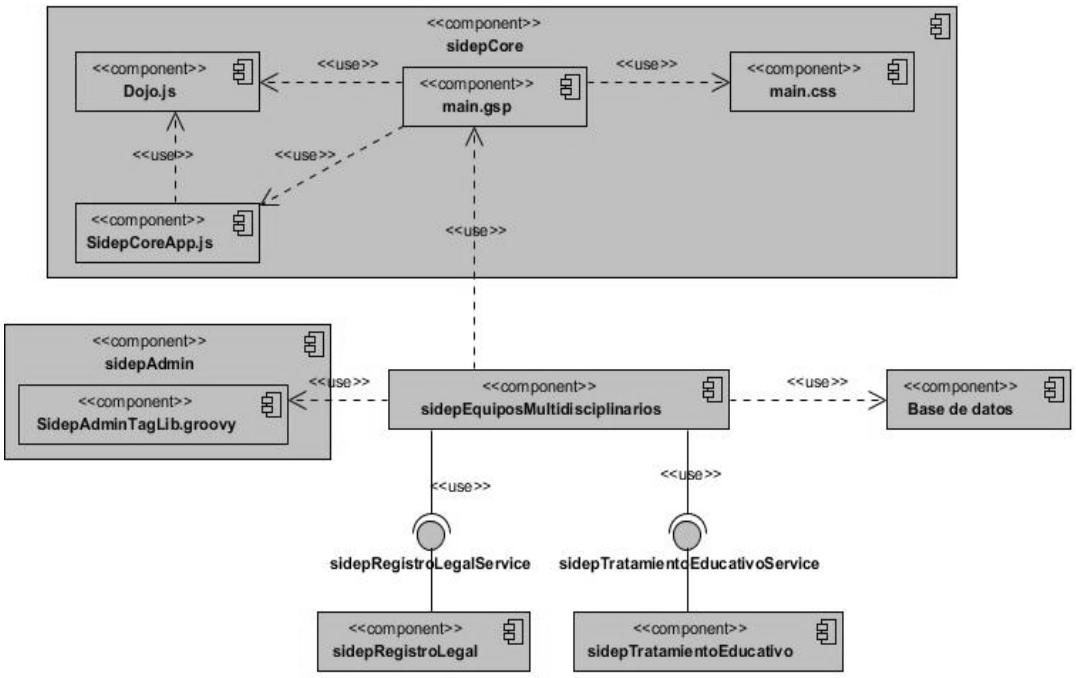


Figura 15: Relaciones del componente Equipos Multidisciplinarios con los componentes visuales y otros subsistemas.

El módulo Opinión Fundamentada tiene como controlador principal a OpinionFundamentadaInicioController el cual se apoya en los controladores BuscarOpinionFundamentadaController y OpinionFundamentadaController.

Estos dos últimos son los encargados de interactuar con las vistas, las cuales se relacionan con los servicios encargados de interactuar con la base de datos y proporcionarle a las vistas la información, recogiéndola y almacenándola en la base de datos. Estos servicios implementan la interfaz TableService.

De manera general el Subsistema Equipos Multidisciplinarios presenta una estructura similar a la de este módulo.

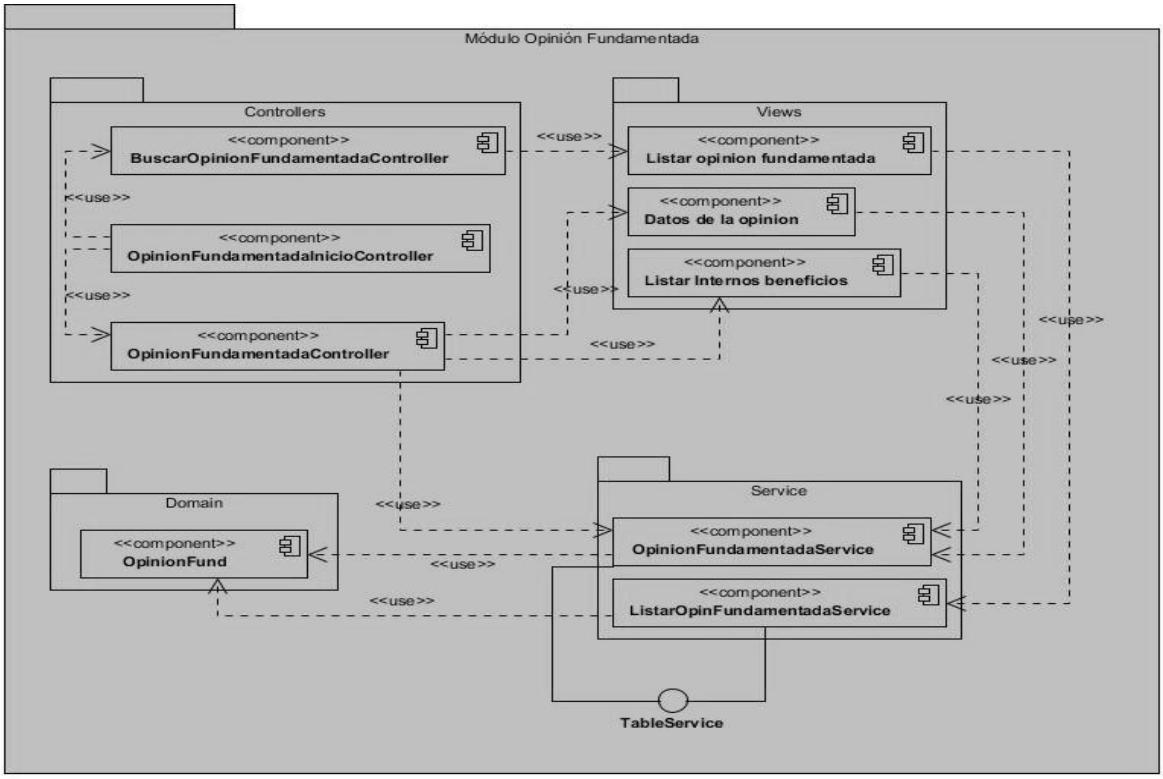


Figura 16: Relaciones del componente Opinión Fundamentada.

4.2.2 Implementación del Subsistema Equipos Multidisciplinarios

La clase *EntrevistaAtencionService* es un servicio que pertenece al módulo Entrevista de Atención, la misma es la encargada de la lógica del negocio y tiene el mayor peso dentro del módulo, ya que contiene las principales funcionalidades del mismo. Brinda además información de las entrevistas al controlador *EntrevistaAtencionController*.

Los principales atributos de la clase son:

- **expedientelegal:** es un identificador que corresponde al expediente legal de un interno.
- **idVinculo:** es un identificador que corresponde al vínculo de un interno.

Entre los métodos principales de esta clase está *salvarEntrevista()* donde si el atributo *idVinculo* tiene valor, se crea una entrevista del tipo *EntAteFam* que es la entrevista que se le realiza al familiar del interno y si el atributo idVinculo no tiene valor se crea una entrevista del tipo *EntrAtencion* que es la entrevista que se realiza al interno.

```
//salvar la entrevista
def salvarEntrevista(params) {
    def entrevista
    if (idVinculo) {
        entrevista = params.entrevista?.id ? EntAteFam.get(params.entrevista?.id.toLong()) : new EntAteFam()
    } else {
        entrevista = params.entrevista?.id ? EntrAtencion.get(params.entrevista?.id.toLong()) : new EntrAtencion()
    }
    entrevista.motivoEntrevista = params.entrevista.motivoEntrevista
    entrevista.resumen = params.entrevista.resumen
    entrevista.recomendaciones = params.entrevista.recomendaciones
    entrevista.fechaEntrevista = new Date()
    if (idVinculo) {
        entrevista.tipoEntrevistaIndFam = NomTipoEntrevIndFam.findByTipoEntrevista("Familiar")
        entrevista.familiar = getVinculo()
    } else {
        entrevista.tipoEntrevistaIndFam = NomTipoEntrevIndFam.findByTipoEntrevista("Individual")
        entrevista.tipoEntrevista = NomTipoEntrevistaAtencion.findById(params.cambiarSelect.toLong())
    }
    entrevista.estadoEntrevista = NomEstadoEntrevista.findByEstadoEntrevista("No Respondida")
    entrevista.investigador = usuarioOnline()
    entrevista.expediente = getExpediente()
    entrevista.save()
    idVinculo = null
    entrevista
}
```

Figura 17: Método para salvar entrevista**.**

El método `salvarVinculo()` crea un objeto de la clase `Vinculo()` y llama al método `guardarVinculoEnEntrevista()` de la clase `tratamientoEducativoService` quien es la encargada de salvar el vínculo en la base de datos.

```
//Aquí se valida el objeto de la clase InternoVinculo y se salva en la base de datos
def salvarVinculo(params) {
    def vin = new Vinculo()
        vin.numeroid = params.nuevoVinculo.numeroid
    boolean tipo = true;
    if (params.tipoVinculo == "false") {
        vin.primernombre = params.nuevoVinculo.primernombre
        vin.segundonombre = params.nuevoVinculo.segundonombre
        vin.primerapellido = params.nuevoVinculo.primerapellido
        vin.segundoapellido = params.nuevoVinculo.segundoapellido
        tipo = false
    }
    tratamientoEducativoService.guardarVinculoEnEntrevista(Interno.get(params.interno.toLong()),
            vin, NomParentesco.get(params.nuevoVinculo.parentesco.toLong()), tipo)
}
```

Figura 18: Método para salvar vínculo.

El módulo Opinión Fundamentada tiene como principal controlador a *OpinionFundamentadaController*, el mismo está relacionado con todos los servicios del módulo como son *opinionFundamentadaService* y *listarOpinFundamentadaService*, enviando los datos que vienen de las páginas cliente hacia el servicio y las respuestas del servicio hacia las páginas cliente. Entre las principales acciones de este controlador se encuentran:

registrarOpinion: en esta acción se obtienen los beneficios del interno y envía los datos del interno y sus beneficios hacia la página cliente.

```
def registrarOpinion = {
    def inter = Interno.findById(params.interno.toLong())
    def ben = opinionFundamentadaService.obtenerTipoBeneficioPorInterno(inter)
    render view: "${urlbase}/datosOpinion", model: [interno: inter, beneficioId: ben]
}
```

Figura 19: Método para registrar Opinión Fundamentada.

guardarOpinion: en esta acción el controlador le envía los datos que fueron enviados desde la página cliente hacia el servicio *opinionFundamentadaService* quien es el encargado de salvar en la base de datos a través del método *salvarOpinion()* y redirecciona hacia otra acción en correspondencia si la opinión fue salvada correctamente o no.

```
def guardarOpinion = {
    def opin = opinionFundamentadaService.salvarOpinion(params)
    if (!opin.validate()) {
        flash.error = "Los datos de la opinión son incorrectos"
        redirect action: "registrarOpinion", params: params
    } else
        redirect action: "listarInternosBeneficios"

}
```

Figura 20: Método para salvar Opinión Fundamentada.

4.2.3. Seguridad del Subsistema

- **Seguridad ante ataques:**

Grails cuenta con varios mecanismos de seguridad para evitar los ataques de inyección SQL y HTML. Todo SQL generado en GORM se escapa para evitar la inyección SQL. En el código implementado no se realizan consultas SQL evitando así la exposición a ataques de inyección SQL.

Las etiquetas que generan URLs (link, form, createLink y resouce) utilizan los mecanismos de escapado apropiados.

Escapar: en un lenguaje de programación o marcado, sustituir caracteres que tienen funcionalidad por secuencias que representan el caracter pero no la funcionalidad. P.ej. En HTML, sustituir '>' por '>'

Además la propia Máquina Virtual Java hace las labores de caja de arena para evitar problemas de desbordamiento de memoria y ejecución de código malicioso. (13)

- **Seguridad en el código:**

Las clases controladoras y acciones específicas, poseen anotaciones de seguridad para restringir el acceso a las diferentes funcionalidades a través del control de los roles, que van a solicitar las peticiones de los controladores. Cada anotación de seguridad tiene definido cuál o cuáles roles van a acceder a la acción que la contenga, donde cada anotación puede definirse para una o varias acciones dentro del controlador.

```
package sidep.equipomultidisciplinario.controllers.investigaciones

import sidep.core.controller.buscar.Commands
import sidep.admin.seguridad.Security

@Security("gestionarInvOperativas,registrarInvCientif")
class InvestigacionesInicioController {
    def listarAutoresService
```

Figura 21: Anotación de seguridad en el controlador.

```
@Security("gestionarGrupoRiesgo")
def actualizarListadoInternos = {
    listarInternosGrupoService.grupo = params.grupo.toString().toLong()
    def json = ["grupo": params.grupo]
    render json as JSON
}
@Security("consultarIntGrupoRiesgo")
def consultarInternosByGrupo = {
    listarInternosGrupoService.consultar = true
    render view: "${urlbase}/consultarInternosGrupo"
}

@Security("gestionarGrupoRiesgo")
def planMedidasColectivo = {
    planMedidasService.size = 0
    planMedidasService.inter = null
    planMedidasService.planAux = null
    planMedidasService.categoria = params.grupoId.toLong()
    planMedidasService.plan = PlanMedidasColect.findByCategoria(Categoria.findByIdAndEstructura(params.grupoIc
    render view: "${urlbase}/planMedidas", model: [tipo: "Colectivo", plan: PlanMedidasColect.findByCategoria
}
@Security("gestionarGrupoRiesgo")
def planMedidasIndividual = {
    planMedidasService.size = 0
    planMedidasService.categoria = null
    planMedidasService.planAux = null
    planMedidasService.inter = params.internoId.toLong()
    planMedidasService.plan = PlanMedidasIndiv.findByInterno(Interno.get(params.internoId.toLong()))
    render view: "${urlbase}/planMedidas", model: [tipo: "Individual", plan: PlanMedidasIndiv.findByInterno(Ir
}
@Security("gestionarGrupoRiesgo")
def salvarMedida = {
```

Figura 22: Anotaciones de seguridad en las acciones del controlador.

4.2.4. Tratamiento de errores

Todos los datos de entrada son validados en el cliente con JavaScript, pero como ningún sistema está totalmente libre de fallos, en caso de que estos ocurran en la validación del lado del cliente, se le da tratamiento en el servidor a los datos capturados, como se muestra en la figura 23.

```
def guardarInvCient = {
        def invCient = listarInvCientService.salvarInvCient(params)
        if (!invCient) {
            flash.error = "Los datos de la investigacion son incorrectos"
            redirect action: "registrarInvCient"
            return
        } else
            redirect action: "mostrarListadoInvCient", params: [id: listarInvCientService.autorInv]
    }
```

Figura 23: Tratamiento de errores en el servidor.

Además de esta validación existe la que se realiza en los *constraints* de las clase de dominio, los cuales son invocados antes de salvar a través del método *validate()*, con el objetivo de garantizar la integridad de los datos procesados.

```
def salvarInvCient(params) {
        def invcient = new InvCient()
        invcient.titulo = params?.tituloI
        invcient.tematica = params?.tematica
        invcient.resultadoInvestigacion = params?.resultadoInvestigacion
        invcient.archivo = !params.archivo ?: params.archivo.getBytes()
        invcient.fechaInvestigacion = new SimpleDateFormat("yyyy-MM-dd").parse(params?.fechaInvestigacion)
        def listaAutores = listarAutoresService.autores
        listaAutores.each {
            invcient.addToAutores(it)
        }
        if(invcient.validate()){
            invcient.save()
            return invcient
        }else{
            return null
        }
    }
```

Figura 24: Validación de objeto de clases del dominio.

El uso de '?' antes de llamar a una propiedad o valor que viene por parámetro, garantiza que no ocurra una excepción interrumpiendo la ejecución del programa en caso de que el valor de la propiedad sea nulo. De existir un valor nulo que no sea permitido para esa clase, se podrá detectar y a su vez darle tratamiento, mediante la llamada al método *validate()*.

```
    static constraints = {
        motivoEntrevista size: 1..10000, nullable: false, blank: false
        resumen size: 1..10000, nullable: false, blank: false
        recomendaciones size: 1..10000, nullable: false, blank: false
        respuesta nullable: true
        tipoEntrevista nullable: true
        fechaEntrevista nullable: false
        investigador nullable: false
        expediente nullable: false
        estadoEntrevista nullable: false
        tipoEntrevistaIndFam nullable: false
    }
```

Figura 25: Validación en el *constraints*.

En el proceso de gestión de las investigaciones científicas del módulo Investigaciones, puede ocurrir un error durante la ejecución de la función *delEvento()* de la clase *ListarEventoService*.

```
// elimina un evento dado un id
def delEvento(id) {
    if (puedeEliminarEvento(id)) {
        try{
            Evento.get(id).delete()
        }catch (RuntimeException ex){
            return false
        }
        return true
    }
   return false
}
```

Figura 26: Método para eliminar un evento.

Este método es invocado por otro desde el controlador, el cual es el responsable de notificar al cliente el error que ha ocurrido a través de un mensaje de error.

4.2.5. Diagrama de despliegue

La distribución de los componentes y los procesos son mostrados en un diagrama de despliegue. Estos definen la arquitectura física del sistema por medio de nodos interconectados, los cuales son elementos hardware que pueden ejecutar los elementos software.

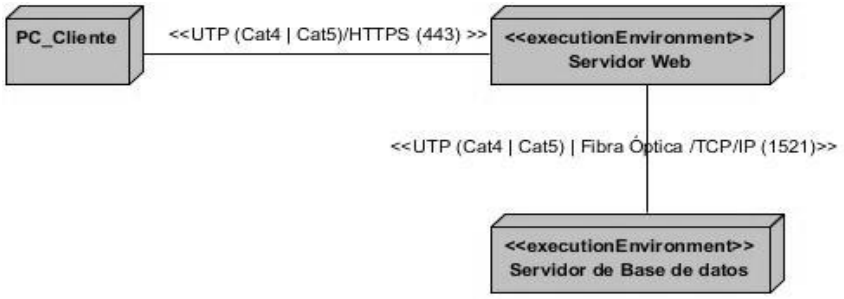


Figura 27: Diagrama de despliegue del Subsistema Equipos Multidisciplinarios.

Para el despliegue del módulo es necesario:

- Servidor de Bases de datos
    - JDK 1.6.
    - Oracle 11g.
- Servidor de la aplicación
    - JDK 1.6.

- Apache Tomcat 6.0.

- Puestos de trabajo (PC cliente)

- Navegador web Mozilla Firefox 3.6.*.

4.3 Pruebas

Para evaluar y determinar la calidad de un software es necesario realizarle diferentes tipos de pruebas. Estas se definen como "*el proceso de ejecución de un programa con la intención de descubrir errores previos a la entrega al usuario final".* (19)

Las pruebas se realizaron con el objetivo de comprobar que el Subsistema Equipos Multidisciplinarios cumpliera con los requisitos pactados con el cliente. Para ello se realizó la siguiente estrategia de prueba.

4.3.1 Métodos de Pruebas de Caja Negra

Las pruebas de Caja Negra son las 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.

4.3.1.1 Técnica de prueba: Diseño casos de prueba

- **Casos de pruebas**

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución, resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final.

Los casos de pruebas deben verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.

- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

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. (19)

4.3.2 Niveles de Pruebas

Las pruebas se aplican a diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Entre los niveles de pruebas existentes se encuentran las Pruebas de desarrollador, que fueron las realizadas al Subsistema de Equipos Multidisciplinarios.

- **Pruebas de desarrollador**

Estas pruebas indican los aspectos de diseño e implementación más adecuados a llevar a cabo por el equipo de desarrolladores. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas de desarrollador que las diseñó e implementó, aunque es recomendable que los desarrolladores creen las pruebas de forma que estén disponibles para que las ejecuten grupos de pruebas independientes. (19)

4.3.3 Tipos de Pruebas

Cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte.

- **Pruebas de funcionalidad**

Este tipo de prueba examina si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones de diseño. En ella se debe verificar si el sistema lleva a cabo correctamente todas las funciones requeridas, validando los datos y ejecutando pruebas de comportamiento ante distintos escenarios.

Estas pruebas deben de estar enfocadas a tareas y para ello se usan los esquemas de pruebas de caja negra, ya que el objetivo es saber si funciona o no, independientemente de la forma en que lo haga. (20)

Se realizaron las pruebas de función las cuales fijan su atención en las validaciones de las funciones, métodos, servicios y casos de uso.

A continuación se muestra una tabla con las No Conformidades (NC) encontradas al Subsistema Equipos Multidisciplinarios durante las pruebas realizadas, determinando la cantidad de NC por iteraciones.

Iteraciones	Entrevista de Atención	Opinión Fundamentada	Grupos de Riesgo	Investigaciones	Adiestramiento a Combatiente	Total
1	8	5	10	9	18	50
2	7	3	5	7	8	30
3	5	0	2	6	5	18
4	0	0	0	0	0	0
Total	20	8	17	22	31	98

Tabla 8: No conformidades detectadas en cada módulo por iteración.

Las principales NC detectadas durante las pruebas realizadas fueron por la poca correspondencia con otros artefactos (Casos de Uso y Casos de Prueba), errores de funcionalidad y ortografía.

4.4 Conclusiones Parciales

Se realizó la implementación del Subsistema Equipos Multidisciplinarios, quedando definido el diagrama de despliegue y los de componentes. Se definió el nivel de prueba y el tipo de prueba a realizar, en este caso las de Caja Negra, lo cual permitió comprobar que el Subsistema cumple con los requisitos acordados con el cliente, además de su correcto funcionamiento.

CONCLUSIONES GENERALES

Para el desarrollo del presente trabajo se realizó un estudio detallado de algunos sistemas informáticos nacionales e internacionales, el cual evidenció que las soluciones existentes no responden a la necesidad del Sistema Penitenciario Cubano, de informatizar los procesos que desarrolla el Equipo Multidisciplinario.

Se abordaron además las herramientas, tecnologías y metodología, aprobadas por el proyecto Prisiones Cuba, para el desarrollo del Subsistema Equipos Multidisciplinarios, destacando las características que las hacen factibles para la presente solución.

Se realizó el análisis, diseño y la implementación del Subsistema, obteniéndose el Subsistema Equipos Multidisciplinarios que permite gestionar los procesos de evaluación psicológica y criminológica de los internos. El mismo está compuesto por cinco módulos: Entrevista de Atención, Grupos de Riesgos, Opinión Fundamentada, Adiestramiento a Combatiente e Investigaciones.

Para garantizar la eficiencia de la aplicación y el cumplimiento de los requisitos definidos con el cliente, se realizaron pruebas de calidad de tipo función hasta la cuarta iteración, evidenciando que la presente solución no presenta no conformidades.

RECOMENDACIONES

Se recomienda:

- Realizar el Manual de Usuario a la solución para una mejor comprensión de la misma por parte de los clientes.
- Tener en cuenta las nuevas necesidades o funcionalidades que puede presentar el Sistema Informativo del Departamento de Establecimientos Penitenciarios.

REFERENCIAS BIBLIOGRÁFICAS

1. Del Risco Batista, Yanet. Sistema de Gestión Penitenciaria de la República de Cuba. Infraestrucutura Producitva, Universidad de las Ciencias Informáticas. Ciudad de La Habana, Cuba : s.n., 2009.
2. Zequeira Peña, Alfonso y Céspedes Quesada, Aimeé. Vocabulario Jurídico Penitenciario. MININT. Ciudad de La Habana : s.n., 2005.
3. Manual de Procedimientos. Equipos Multidisciplinarios. MININT. Ciudad de La Habana : s.n., 2005.
4. Martínez Bravet, René. "Repositorio de Datos de la Sala Situacional del SIGEP, desde el punto de vista del Control Penal.". Universidad de Ciencias Informáticas. Ciudad de la Habana, Cuba. : s.n., 2010. Tesis.
5. Guerrero, José Iván. Evolución histórica del Sistema Penitenciario. Panamá : s.n.
6. Bernal, Lisbeth. Panamá. Situación del Sistema Penitenciario. [Online] http://www.ilanud.or.cr/centro-de-documentacion/biblioteca-digital/181-sistemas-penitenciarios.html.
7. Rational Software Corporation. "Rational Unified Process". 2003.
8. Embarcadero Technologies Inc. Embarcadero. [Online] [Cited: 11 28, 2011.] http://www.embarcadero.com/products/er-studio.
9. Visual Paradigm. Visual Paradigm. [Online] http://www.visual-paradigm.com/product/vpuml/.
10. Apache Software Foundation. Apache Tomcat. [Online] 1999-2011. [Cited: 11 28, 2011.] http://tomcat.apache.org.
11. Subversion. Subversion. [Online] 2001-2009. [Cited: 11 28, 2011.] http://subversion.tigris.org/.
12. Oracle. [Online] http://www.oracle.com/index.html.
13. Brito, Nacho. Manual de Grails. [Online] http://www.manual-de-grails.es.
14. Dojo Toolkit. [Online] noviembre 28, 2011. http://dojotoolkit.org/.
15. Ivar Jacobson, Grady Booch James Rumbaugh. El Lenguaje Unificado de Modelado. Manual de Referencia .
16. Java Script. [Online] http://www.javascript.com/.
17. Subramaniam, V. Programming Groovy.

18. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objeto. México. : s.n., 2012.

19. EVA. [Online] http://eva.uci.cu/file.php/160/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf.

20. Pruebas del sistema. [Online] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo5.pdf.

BIBLIOGRAFÍA

1. Del Risco Batista, Yanet. *Sistema de Gestión Penitenciaria de la República de Cuba.* Infraestrucutura Producitva, Universidad de las Ciencias Informáticas. Ciudad de La Habana, Cuba : s.n., 2009.
2. Zequeira Peña, Alfonso y Céspedes Quesada, Aimeé. *Vocabulario Jurídico Penitenciario.* MININT. Ciudad de La Habana : s.n., 2005.
3. *Manual de Procedimientos. Equipos Multidisciplinarios.* MININT. Ciudad de La Habana : s.n., 2005.
4. Martínez Bravet, René. *"Repositorio de Datos de la Sala Situacional del SIGEP, desde el punto de vista del Control Penal.".* Universidad de Ciencias Informáticas. Ciudad de la Habana, Cuba. : s.n., 2010. Tesis.
5. Guerrero, José Iván. *Evolución histórica del Sistema Penitenciario.* Panamá : s.n.
6. Bernal, Lisbeth. Panamá. Situación del Sistema Penitenciario. [En línea] http://www.ilanud.or.cr/centro-de-documentacion/biblioteca-digital/181-sistemas-penitenciarios.html.
7. *Rational Software Corporation. "Rational Unified Process".* 2003.
8. Embarcadero Technologies Inc. Embarcadero. [En línea] [Citado el: 28 de 11 de 2011.] http://www.embarcadero.com/products/er-studio.
9. Visual Paradigm. *Visual Paradigm.* [En línea] http://www.visual-paradigm.com/product/vpuml/.
10. Apache Software Foundation. Apache Tomcat. [En línea] 1999-2011. [Citado el: 28 de 11 de 2011.] http://tomcat.apache.org.
11. Subversion. Subversion. [En línea] 2001-2009. [Citado el: 28 de 11 de 2011.] http://subversion.tigris.org/.
12. Oracle. [En línea] http://www.oracle.com/index.html..
13. Brito, Nacho. *Manual de Grails.* [En línea] http://www.manual-de-grails.es.
14. Dojo Toolkit. [Online] noviembre 28, 2011. http://dojotoolkit.org/.
15. Ivar Jacobson, Grady Booch James Rumbaugh. El Lenguaje Unificado de Modelado. Manual de Referencia .

16. Java Script. [Online] http://www.javascript.com/.

17. Subramaniam, V. *Programming Groovy.*

18. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* México. : s.n., 2012.

19. EVA. [Online] http://eva.uci.cu/file.php/160/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf.

20. Pruebas del sistema. [Online] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo5.pdf.

21. Smith, Glen y Ledbrook, Peter.*Grails in Action.* s.l. : Manning Publications Co., 2009. ISBN 978-1-933988-93-

22. Klein, Dave.*Grails. A Quick-Start Guide.* North Carolina, Dallas, Texas : s.n., 2009. ISBN-10: 1-934356-46-8, ISBN-13: 978-1-934356-46-3.

23. Judd, Christopher M., Faisal Nusairat, Joseph y Shingler, James .*Beginning Groovy and Grails From Novice to Professional.* New York : s.n., 2008. ISBN-13 (pbk): 978-1-4302-1045-0, ISBN-13 (electronic): 978-1-4302-1046-7.

24. Davis, Scott y Rudolph, Jason .*Getting Started with Grails Second Edition.* s.l. : C4Media, Publisher of InfoQ.com., 2010. ISBN: 978-0-557-18321-0.

25. Fischer, Robert.*Grails Persistence with GORM and GSQL.* 2009. ISBN-13 (electronic): 978-1-4302-1927-9, ISBN-13 (paperback): 978-1-4302-1926-2.

26. Dearle, Fergal.*Groovy for Domain-Specific Languages.* s.l. : Packt Publishing Ltd., 2010. ISBN 978-1-847196-90-3.