Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Diseño e Implementación de los módulos Vínculos y Trabajo del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

Autor: Leonel Arjona Pérez

Tutora: Ing. Liset Schery Sánchez

Co-Tutor: Msc. Julio Omar Prieto Entenza

La Habana, Junio 2012

“Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente.”

Ernesto Guevara.

DECLARACIÓN DE AUTORÍA.

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Leonel Arjona Pérez

_____

Firma del autor

Liset Schery Sánchez

_____

Firma de la tutora

Julio Omar Prieto Entenza

_____

Firma del co-tutor

DEDICATORIA

A mis padres en especial a mi mamá, a mis hermanos, a mi mujer y a mi querido hijo.

AGRADECIMIENTOS

Es una realidad que en muchas ocasiones de la vida nos reservamos el acto de agradecer a una persona por su forma de ser o por su ayuda brindada, y esto ocurre por varias razones, entre ellas cabe mencionar las siguientes:

- En ocasiones pasamos la vida creyendo que deberían agradecernos a nosotros.

- Creemos que la persona en cuestión tiene idea de lo mucho que la queremos y agradecemos.

Haciendo caso omiso de estas ideas, las cuales llegan a convertirse en frenos de sentimientos, hago llegar mi eterno agradecimiento a todas las personas que han contribuido con este logro.

Agradecimientos para:

Mi tutora Liset Schery Sánchez, la cual tuvo gran entrega demostrando madurez, paciencia y responsabilidad durante todo el tiempo de trabajo, gracias por siempre estar presente y tenderme la mano.

Roberto Granda Ruiz por su paciencia, consagración, por lograr aguantar todos estos meses de ajetreo sin fin, por siempre tener respuestas a diversas preguntas y por mantener en todo momento al equipo de desarrollo enfocado en cumplir con el trabajo correctamente.

Grupo de desarrollo del SIDEP, el cual llegó a convertirse en una gran familia donde todos trabajamos por un mismo objetivo fluyendo el trabajo en un ambiente cooperativo y consagrado.

2507: a todos los compañeros que conocí a partir de 3er año con los cuales he compartido diversas experiencias y emociones, siempre los recordaré.

A todas mis amistades de la facultad 4 con las cuales compartí mis primeros años de universidad y con los cuales fundé una amistad pura, más que una amistad una hermandad, cabe mencionar a Pablo Noriega, Lazara Doris, Alberto Batista, Yailyn Gonzales, Yosniel Montes de Oca, Raudel, entre otros.

A todos los cadetes insertados con los cuales he compartido durante todos estos años incluyendo la etapa de servicio militar, entre ellos Josué Gutiérrez, Ramsés Jiménez, Lázaro Valdés, Nairelys Padrón, Silvia, entre otros.

A mi familia:

A mi papá, Osvaldo Lorenzo Arjona Santos, a mi abuela Ramona Santos, a mi tía Cary,a mis tíos y primos por toda la confianza depositada en mí y los consejos brindados, por el empuje que siempre me han dado para seguir avanzando y por todo el cariño y aprecio que me han brindado.

A mis hermanos, Leosvany, Liudmila, Estebitan, Yaiselys, Yaidelys y Danger los cuales han sido un motor impulsor en momentos difíciles, ya que me han dado la fuerza para seguir y levantarme cuando he caído, mi anhelo es que me vean como un ser humano que puede equivocarse pero que sueña en convertirse en un ejemplo a seguir para ellos.

A Rafael, por toda la ayuda brindada y por apoyar a mi mamá durante toda esta etapa de camino recorrido.

A mi tía Milagros (nena), la cual se ha convertido en la patrona de la familia, es esa persona que siempre que hay una situación está dispuesta a darlo todo por resolver el problema.

A mi mujer Olimpia Albazzy, la cual me ha dado otra razón para esforzarme y entregarme desmedidamente, y esa razón tiene como nombre Layonel Jesús Arjona Pérez, mi pequeño gigante, el candil que alumbra mis pasos firmemente y da impulso a mi andar. Quiero agradecerte por tu entrega, por permitirme seguir con mi sueño, por mantener una postura responsable y sincera todo este tiempo.

A Esteban Ge Sánchez (papá 2), el ejemplo claro de patrón experto, pero en su caso no delega responsabilidades, sino que se encarga de cumplirlas él mismo, sin dudas un gigante, es el tipo de persona que entrega sin pensar en que recibir a cambio, ni en si se le agradece o no. Aprovechando este momento te agradezco por toda la ayuda brindada.

A mis abuelos maternos, los cuales se encuentran descansando en paz y pendientes de cada paso que doy desde el cielo, mi eterno agradecimiento por todo su amor, fraternidad, paciencia y entrega sin límites.

A mi mamá Lázara Pérez Baró el hada que al mover su varita ha hecho posible este sueño, el mérito es tuyo de punta a cabo, tu esfuerzo, consagración, constancia, entrega y amor me han dado fuerzas para encarar la vida, y poder regalarte este momento y este resultado.

RESUMEN.

Con el avance de las tecnologías a nivel mundial, Cuba se ha insertado en la informatización de un gran número de sectores, tanto económicos como sociales. Conociendo de las ventajas que propiciaría la creación de un sistema informático en cuanto a rapidez, control y eficiencia, el Ministerio del Interior (MININT), en conjunto con la Universidad de Ciencias Informáticas (UCI), se sumó a la tarea de desarrollar un sistema para la gestión de los centros penitenciarios en Cuba: Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

El presente trabajo de diploma tiene como objetivo el desarrollo de los módulos Vínculos y Trabajo, pertenecientes al SIDEP, para brindar información referente a las relaciones interpersonales que permita conocer posibles pensamientos negativos de los internos que conlleven a actos indebidos o violentos y de las actividades productivas a las que cada interno podrá integrarse de acuerdo a sus conocimientos en algún área laboral específica. Este objetivo toma como punto de partida el análisis de los requerimientos de software establecidos en acuerdo con el cliente y haciendo uso de las tecnologías y herramientas definidas en la arquitectura del proyecto.

Se pretende con la realización de los módulos ya mencionados brindar una aplicación que cumpla con todos los procesos referentes a la gestión de los vínculos y trabajo en los centros penitenciarios, siendo capaz de gestionarlos de manera eficaz.

Palabras claves: SIDEP, requerimientos.

TABLA DE CONTENIDO.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Antes del triunfo de la Revolución las prisiones de la tiranía se caracterizaban por realizar procedimientos abusivos e inhumanos contra los internos y la adopción de métodos de tortura era una vía aceptada por el sistema existente. Luego del 1ro de enero de 1959 el gobierno cubano, basándose en el respeto y en una gestión de control inspirada en la regulación de normas y leyes internacionales en cuanto al tratamiento a los reclusos, toma numerosas medidas con el objetivo de revertir muchas de las políticas y leyes que había dejado como legado la tiranía. Una de las medidas que se tomó, en aras de alcanzar un ordenamiento jurídico y crear nuevas formas de enfrentar el delito, fue la renovación del sistema penitenciario existente. En el transcurso de los años de revolución, el sistema penitenciario actual ha sufrido las incidencias de determinadas transformaciones que han permitido cumplir con los fines de la sanción penal, y han ayudado a situar el sistema de justicia cubano entre uno de los de mayor carácter humanista del mundo, por contar entre sus fortalezas con el perfeccionamiento de la legislación penitenciaria y de su base reglamentaria.

En el año 1989 comienza en Cuba la informatización de los centros penitenciarios, con la automatización de los datos principales del recluso y ciertos aspectos de control penal. Varios años después a raíz del cumplimiento de la orden 43/99 del Vice Ministro Primero del MININT la cual plantea la creación de un sistema capaz de gestionar la información referente a los centros penitenciarios, se crea el Sistema Automatizado para el Control del Recluso (SACORE), el cual comenzó a llevar el control y la gestión de la información de los centros penitenciarios. Luego de algunos años de explotación del sistema, a pesar de sus facilidades y ayuda brindada a dichos centros, aún contaba con requisitos incompletos lo que afectaba el proceso de toma de decisiones.

Luego de comprobar que el SACORE no lograba abarcar todos los procesos que se ejecutaban en las prisiones, la Dirección de Establecimientos Penitenciarios (DEP) se dio a la tarea de desarrollar otras dos aplicaciones. Por tales motivos surge así el Sistema Automatizado de Incidencias de la Dirección de Establecimientos Penitenciarios (SAIDEP), que gestiona el control y seguimiento de las incidencias o hechos extraordinarios que alteran el orden interior y afectan la seguridad de los establecimientos penitenciarios. También se desarrolla el Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios (SACDEP), que permite definir la estructura física del área de reclusión penal y gestiona el control sobre las capacidades penitenciarias, los niveles de ocupación, la disponibilidad y las afectaciones de las instalaciones penitenciarias cubanas.

Aunque la DEP cuenta con 3 aplicaciones para la gestión de la información de las instalaciones base, estas no ofrecen grandes facilidades para la gestión de la información ya que este proceso se realiza desorganizadamente y no se abarcan aún todos los procesos del sistema penitenciario nacional. Para lograr corregir estas deficiencias, la DEP autorizó el inicio del proyecto Prisiones Cuba, en colaboración con la UCI, con el objetivo de obtener un software para la centralización de toda la información referente a los establecimientos penitenciarios del país: SIDEP.

El SIDEP cuenta con diferentes subsistemas que están constituidos por módulos, los que representan los procesos que se desarrollan en el sistema penitenciario cubano. Entre los módulos que componen este sistema se incluyen Vínculos y Trabajo, los cuales gestionarán la información de los vínculos internos y externos de los reclusos para evitar hechos extraordinarios, además, controlar la incorporación de estos a brigadas de trabajo según su profesión, para de esta forma facilitarles la obtención de valor monetario y ayudar así a sus familiares.

A partir de la problemática antes planteada surge como **problema a resolver:**

¿Cómo gestionar la información referente a las relaciones interpersonales y a las actividades productivas de los internos en el SIDEP?

Definiendo como **objeto de estudio**:

Los procesos de tratamiento educativo en los sistemas penitenciarios.

Y quedando definido como **campo de acción**:

La informatización de los procesos de gestión de vínculos y trabajo en el SIDEP.

Teniendo en cuenta lo expuesto anteriormente se plantea como **objetivo general**:

Desarrollar los módulos Vínculos y Trabajo del SIDEP a partir del análisis de los requisitos de software establecidos con el cliente y haciendo uso de la arquitectura y las tecnologías definidas por el proyecto.

Teniendo como **idea a defender**:

Con el desarrollo de los módulos Vínculos y Trabajo del SIDEP se logrará gestionar la información referente a las relaciones interpersonales y a las actividades productivas de los internos en los centros penitenciarios de Cuba.

Como vías para darle solución a las distintas tareas se plantea como **objetivos específicos**:

1. Elaborar el marco teórico de la investigación.
2. Realizar el modelo de diseño de los requisitos planteados para los módulos Vínculos y Trabajo del SIDEP.
3. Implementar los módulos Vínculos y Trabajo del SIDEP.
4. Realizar las pruebas de calidad a los módulos Vínculos y Trabajo del SIDEP.

En aras de dar cumplimiento al objetivo planteado se traza el siguiente conjunto de **tareas de la investigación:**

1. Análisis de soluciones existentes en el mundo relacionadas con los sistemas penitenciarios.
2. Descripción de las herramientas y tecnologías para dar solución al problema propuesto.
3. Realización de los modelos de diseño de los módulos Vínculos y Trabajo.
4. Realización de los diagramas de componentes de los módulos Vínculos y Trabajo.
5. Realización del diseño de la base de datos de los módulo Vínculos y Trabajo.
6. Implementación de los módulos Vínculos y Trabajo.
7. Diseñar la estrategia de prueba para los módulos Vínculos y Trabajo.
8. Ejecución de las pruebas a los módulos Vínculos y Trabajo.

Para validar metodológicamente la investigación se utilizarán los siguientes **métodos científicos**:

Análisis documental: el análisis documental es una forma de investigación técnica, un conjunto de operaciones intelectuales, que buscan describir y representar los documentos de forma unificada y sistemática para facilitar su uso. Se emplea en el análisis de documentos para facilitar la creación del glosario de términos, diagramas de entidades, especificación de CUs, entre otros.

Analítico-Sintético: este método se basa en la investigación, análisis de teorías y documentos, por lo que ha sido de gran ayuda durante la confección del diseño teórico de la investigación. Es empleado para examinar los documentos consultados en la investigación y así conocer todo lo relacionado con los procesos de gestión de las relaciones interpersonales de los internos y la vinculación de un interno a una actividad productiva en el sistema penitenciario cubano.

Histórico-Lógico: se utiliza para el análisis de la trayectoria y evolución de la metodología de desarrollo de software y de las herramientas que se usarán durante el trabajo. Este método ha sido usado en el estudio de la línea temporal de los productos informáticos existentes en diferentes países que se relacionan con los sistemas penitenciarios.

Modelación: se utiliza para la elaboración de diagramas, figuras y otros artefactos importantes en las diferentes disciplinas a desarrollar. Se escoge este método ya que se pueden crear abstracciones con el propósito de explicar la realidad.

El presente documento consta de 3 capítulos. El primer capítulo, titulado **"**Fundamentación teórica", tiene como objetivo abordar conceptos relacionados con el sistema penitenciario cubano y se hace un análisis de los sistemas informáticos que se encuentran en uso en distintos centros penitenciarios del mundo. Se explicarán las principales herramientas y tecnologías a utilizar para el desarrollo de los procesos relacionados con el diseño e implementación de los módulos Vínculos y Trabajo del SIDEP. En el segundo capítulo, nombrado "Diseño del sistema", se relaciona lo referente al diseño propuesto para la solución del software, así como la descripción de la arquitectura a utilizar. Además, se especifica el conjunto de funcionalidades a desarrollar, dando un breve resumen de las mismas. Se describen los diferentes diagramas correspondientes a la disciplina de diseño, incluyendo también los patrones que se tuvieron en cuenta. En el tercer capítulo, titulado "Implementación y prueba", se detallan los elementos relacionados con la implementación de los módulos Vínculos y Trabajo, para dar cumplimiento a las funcionalidades propuestas en el capítulo anterior. Se detallará la estrategia de prueba a seguir para validar la calidad de los módulos Vínculos y Trabajo pertenecientes al SIDEP.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se abordan algunos conceptos relacionados con el sistema penitenciario cubano, así como ejemplos de soluciones informáticas que se encuentran en uso en distintos centros penitenciarios del mundo. Se realiza una breve descripción de la metodología, las herramientas, las tecnologías y los lenguajes utilizados para el desarrollo del proyecto SIDEP, específicamente de los procesos relacionados con el diseño e implementación de los módulos Vínculos y Trabajo.

1.2. Marco Conceptual

Vínculos: significa unión o atadura de una persona o cosa con otra. (1) En el sistema penitenciario cubano, los vínculos se definen como las relaciones interpersonales de los internos de carácter positivo o negativo que pueden ser internas en el centro penitenciario o fuera de la instalación.

Sistema penitenciario: este término se define como el 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. (2) Se puede concluir que el sistema penitenciario, es el conjunto de normas generales, establecidas y específicas referidas a las penas en sí, el modo de su cumplimiento y el tratamiento de los penados y procesados.

Cada estado estipula dichas organizaciones basándose en sus concepciones políticas, y adecuándose a las condiciones sociales y económicas de su desarrollo. En Cuba este sistema está dirigido por la DEP, perteneciente al MININT, y se sustenta en la integración de principios, conceptos, procedimientos, fuerzas y medios que garantizan el funcionamiento de los centros destinados al internamiento y el tratamiento a los internos. A su vez, los fundamentos de la política penitenciaria están determinados en la Constitución de la República, la Ley 62 del Código Penal la cual plantea que el régimen de sanciones previsto en el Código Penal por su coherencia, equilibrio y flexibilidad, debe responder a la gravedad de los diversos comportamientos delictivos, de manera que se garantice, al aplicar la sanción, una adecuada individualización de la misma. Además la organización y las condiciones de ejecución de las sanciones privativas de libertad se corresponden con lo establecido en las "Reglas Mínimas Clásicas para el Tratamiento a los Reclusos" aprobadas el 30 de agosto de 1955 por la Organización de las Naciones Unidas (ONU) y la Declaración Universal de los Derechos del Hombre del 10 de diciembre de 1948. (3)

1.3. Valoración del estado del arte

1.3.1. Sistemas nacionales e internacionales

Hoy en día el desarrollo de productos informáticos es un proceso constante y ascendente, por lo que conlleva a una evolución dinámica para adaptarse a las necesidades tecnológicas del momento histórico, por tal motivo los centros penitenciarios, como disímiles organizaciones, tienen la necesidad de realizar cambios en su forma de gestionar la información que manejan. Antiguas y rigurosas estrategias de trabajo, así como los arduos procesos de gestión existentes durante décadas en las prisiones, han sido sustituidas por eficientes soluciones informáticas capaces de gestionar dichos procesos con la máxima seguridad y mínimas dificultades.

A continuación se analizarán algunas soluciones informáticas existentes en los sistemas penitenciarios del mundo y de Cuba, los cuales se estudiarán para determinar si pueden constituir una posible solución o parte de la solución del presente trabajo, teniendo en cuenta las funcionalidades que brindan.

Sistema de Gestión Penitenciaria (SIGEP)

El Sistema de Gestión Penitenciaria (SIGEP), desarrollado en la UCI y desplegado en la República Bolivariana de Venezuela, es una aplicación informática que tiene como objetivo fundamental controlar la situación operativa de los establecimientos penitenciarios y constituye una base para la toma de decisiones tácticas y estratégicas de la Dirección General de Custodia y Rehabilitación del Recluso, teniendo en cuenta las legislaciones del sistema penitenciario de este país.

Este sistema se ha puesto en práctica en tres tipos de entidades dentro del sistema penitenciario venezolano: la sede central, los centros penales externos y los centros penales internos; (4) logrando mantener centralizada la información relacionada con los elementos más importantes de los internos.

Este sistema tiene implementado los módulos Vínculos y Trabajo pero de manera muy superficial pues no se contaba con la descripción de los requisitos para la implementación correcta de dichos módulos, por lo que se desarrollaron con funcionalidades básicas. En el caso de los vínculos se gestionan los vínculos internos y externos del recluso y solo se guarda de la clase que cuenta con la información de la relación entre el interno y el vínculo el tipo de vínculo que puede ser positivo o negativo, de ser negativo y ser un vínculo externo se le restringe la visita al interno. Para la gestión de los procesos de trabajo se implementa la funcionalidad gestionar unidad productiva a la cual se le incorporan los

internos, se muestra un reporte de desincorporados y se le pasa la asistencia a los interno registrando la hora de entrada y la hora de salida ya que en dependencia de las horas laboradas se le disminuye la condena al interno.

Sistema penitenciario del gobierno de Panamá

Este sistema fue creado a principios de año 1997, como resultado de un proyecto financiado por el Programa de las Naciones Unidas para el Desarrollo (PNUD) y el gobierno español en coordinación con la Dirección General de Sistemas Penitenciarios de Panamá (DGSP). La finalidad del software es mantener en una base de datos los registros de los internos que están detenidos en los centros penales a nivel nacional.

Este sistema tiene como objetivo planificar, organizar, administrar, coordinar, supervisar y dirigir el funcionamiento de los diferentes tipos de centros penitenciarios existentes y por crearse en la república de Panamá, dentro del marco del respeto a los derechos humanos, los principios de seguridad, rehabilitación y defensa social, y la aplicación de los avances científicos en su gestión. (5)

El sistema registra la información referente a los datos personales del interno, antecedentes médicos, datos socioeconómicos y jurídicos y algún otro dato de importancia, además muestra datos relacionados con los centros penitenciarios del país. La información capturada se registra automáticamente en una base de datos Oracle que radica en la sede central y la cual está disponible para los diferentes departamentos que tienen acceso al sistema. Esto garantiza que se refleje de forma inmediata los cambios en el expediente del interno tales como traspasos de autoridad, traslados de un centro a otro, libertades, diligencias médicas, jurídicas y la realización del cómputo automático de la sentencia.

Las limitaciones se evidencian en los centros que no poseen enlace aún con la dirección central lo que debe llevar a una transformación de la red externa, reestructuración de la red interna de la sede, la dotación de internet a la institución, la actualización de toda la información de los centros penitenciarios y actualización de los equipos.

Al igual que el SIGEP, este sistema se rige por las leyes jurídicas y penales del gobierno panameño, donde los procesos de los módulos Vínculos y Trabajo no constituyen aspectos relevantes en la gestión del sistema penitenciario, por tal razón estos procesos no se encuentran incluidos como funcionalidades del sistema.

Sistema Automatizado para el Control del Recluso (SACORE)

El Sistema Automatizado para el Control del Recluso (SACORE) fue desarrollado en Cuba en el año 2002 y se comenzó a utilizar a partir del 2003. Este sistema surge para dar cumplimiento a la Orden 43/99 del Vice-Ministro Primero del Ministerio del Interior de Cuba y con el objetivo de automatizar la gestión de los datos principales del recluso y algunos aspectos de control penal, este es el primer sistema informático que gestiona la información que generan las prisiones cubanas. (6)

El SACORE garantiza respuestas inmediatas a las solicitudes de información de los diferentes centros penitenciarios del país, siendo un sistema que recoge prácticamente la totalidad de la información de los reclusos en todas las especialidades. Entre las principales funcionalidades que brinda están contempladas las siguientes:

1. Respuestas inmediatas a las solicitudes de información de los diferentes órganos e instituciones del estado como son: Jefatura del MININT, Ministerio de Justicia, Tribunales, Fiscalías, entre otros.
2. Los partes que se emiten son obtenidos de forma automatizada.
3. Se recogen datos de medidas disciplinarias, visitas, ocupaciones de objetos no permitidos por lo establecido en el reglamento, requisas, entre otros. (6)

SACORE cuenta con 9 años de explotación y presenta determinados problemas que impide su uso exclusivo en la gestión de los procesos del sistema penitenciario cubano:

1. Gran cúmulo de información difícil de procesar.
2. Realización de consultas a la BD por personal no especializado.
3. Necesidad de generar gráficos para realizar los análisis estadísticos.

Estas dificultades provocan que el análisis estadístico se convierta en un proceso engorroso y en muchas ocasiones lento, propiciando la demora en la entrega a tiempo de los informes, afectando de esta forma la toma de decisiones. Para solucionar estos problemas se decidió implementar dos sistemas más en la DEP: SAIDEP y SACDEP, los cuales permitirían gestionar las incidencias y controlar las capacidades de los centros penitenciarios respectivamente.

Aún con la integración de estos 3 sistemas no se solucionó el problema de la lentitud de la gestión de la información y la toma de decisiones ya que muchos datos son manejados y conservados en formato duro. No se puede recurrir a ninguno de estos sistemas para dar solución a la problemática planteada debido a que no poseen las funcionalidades que dan cumplimiento a la gestión de los procesos de los módulos Vínculos y Trabajo.

Análisis General

Después de haber analizado algunos de las soluciones informáticas de los sistemas penitenciarios nacionales e internacionales, se puede concluir de que dichos sistemas no cumplen con las características que van acorde a las especificaciones del SIDEP. Además, algunos de estos sistemas no incluyen los procesos de vínculos y trabajo como parte de su solución y en otros casos no incluyen todas las funcionalidades que requiere el sistema penitenciario cubano. Por tal razón, se hace necesario implementar las funcionalidades relacionadas con dichos procesos como parte integradora del SIDEP.

1.4. Tecnologías y herramientas utilizadas en el desarrollo de la aplicación

La metodología fue escogida por el líder del proyecto, el equipo de arquitectos y los analistas, mientras las tecnologías, los lenguajes y las herramientas definidos para la construcción del software fue producto de un estudio realizado por el equipo de arquitectura del SIDEP y establecido como políticas del proyecto, por lo que su selección queda fuera del alcance del presente trabajo. Sólo se brindará una breve descripción de cada uno de estos aspectos a utilizar.

1.4.1. Metodología de Desarrollo

La metodología de desarrollo de software se encarga de elaborar estrategias; están centradas en las personas o los equipos y orientadas hacia la funcionalidad y la entrega. Su objetivo es elevar la calidad del software a través de un mayor control sobre el proceso. (7)

Proceso Unificado de Modelado

RUP[1] es una metodología que fue desarrollada por Rational (IBM) y constituye una enorme base de conocimiento de Ingeniería de Software que guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado, mientras se balancean los requisitos del negocio y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requisitos, detalla qué artefactos producir, cómo desarrollarlos y también provee patrones para la confección de estos. (8)

[1] *Rational Unified Process*, Proceso Unificado de Modelo

Lo que define a RUP es que está dirigido por CU, está centrado en la arquitectura y es iterativo e incremental. La metodología RUP divide en 4 fases el desarrollo del software:

1. Inicio, donde se define el alcance y los objetivos del proyecto.
2. Elaboración, aquí se define la arquitectura del sistema y se obtiene una aplicación ejecutable.
3. Construcción, donde se obtiene un producto listo para su utilización.
4. Transición, donde finalmente se instala el software en condiciones reales. (9)

En la figura 1 se muestran las fases y flujos de trabajo (disciplinas) que propone RUP como ciclo de vida de un proyecto.

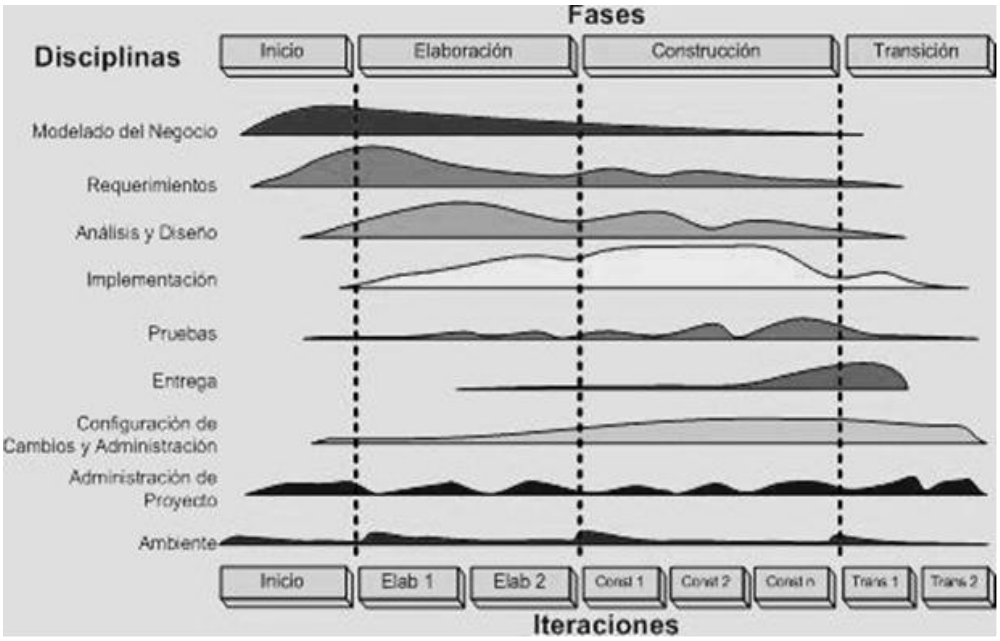

Figura 1 Fases y flujos de trabajo de RUP

SIDEP es una aplicación de gran envergadura para el país, es complejo y para su desarrollo necesita atravesar por todas las fases y los flujos de trabajo que establece RUP como parte del desarrollo de software. El objetivo principal de usar RUP es garantizar el éxito del desarrollo de los módulos haciendo uso del ciclo de vida de un proyecto que esta describe.

1.4.2. Herramientas de modelado

Visual Paradigm para UML 8.0

Visual Paradigm para UML[2] es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. UML es una herramienta CASE[3] que también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (10)

Para llevar a cabo las disciplinas del desarrollo de software incluidas en el presente trabajo, Diseño e Implementación, se hace uso de la herramienta Visual Paradigm, por las funcionalidades que brinda en la modelación de los diferentes diagramas, como por ejemplo: diagrama de clases del diseño, diagrama de colaboración, diagrama de componentes y modelo de despliegue. Además, esta herramienta, a pesar de ser privativa, puede ser utilizada en la universidad ya que se cuenta con la licencia para su uso. También es una herramienta fácil de instalación, actualización y uso.

Embarcadero ER/Studio 8.0

Es una poderosa herramienta CASE, para efectuar el modelado lógico y físico de bases de datos relacionales y multidimensionales. Entre otras características, ER/Studio permite hacer una separación entre los modelos lógicos y físicos, pero manteniendo una total integración entre ellos, así como con la base de datos. Es posible generar múltiples modelos físicos asociados a un mismo modelo lógico, para diferentes plataformas. (11)

Entre sus ventajas se tiene las siguientes:

1. Diseño físico y lógico separados.
2. Transformación automática de un diseño lógico a un diseño físico para una plataforma específica de BD.
3. Extensa validación de los modelos, lo que hace posible validar ampliamente la calidad e integridad, tanto del diseño lógico como del diseño físico del modelo.

Esta herramienta está enfocada a ayudar a tomar decisiones en cómo resolver embotellamientos de los datos y eliminar la redundancia.

[2] *Lenguaje Unificado de Modelado*, Unified Modeling Language
[3] *Computer Aided Software Engineering*, Herramientas de Ingeniería de Software Asistida por Computación

Se decide utilizar esta herramienta CASE para el modelado de las entidades de la BD de los módulos Vínculos y Trabajo, ya que es líder en el diseño y construcción de BD a nivel físico y lógico, además permite la transformación automática de un diseño lógico a un diseño físico para una plataforma específica de BD y permite comparar el diseño con la estructura real física de las tablas.

1.4.3. Herramienta de desarrollo

NetBeans 7.0

El IDE[4] NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas, (12) está escrito completamente en Java usando la plataforma NetBeans, pero puede servir para cualquier otro lenguaje de programación. Es una herramienta de código abierto. Cuenta con un compilador de Java interno y un modelo completo de los archivos fuente de Java; esto permite técnicas avanzadas de refactorización y análisis de código. Este IDE también hace uso de un espacio de trabajo, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente. Es un producto libre y gratuito sin restricciones de uso.

Este IDE es usado en el desarrollo del sistema debido a que se integra fácilmente con el lenguaje de programación Groovy y con el framework Grails que se incluyen como parte de la propuesta del sistema.

Apache Tomcat 6.0.*

Apache Tomcat es el servidor web más utilizado a la hora de trabajar con Java en entornos web, es una implementación completamente funcional de los estándares de JSP y Servlets, desarrollado en código abierto en lenguaje Java por lo que funciona en cualquier sistema operativo que disponga de una máquina virtual Java. Es desarrollado en un entorno abierto y participativo. Publicado bajo la licencia Apache versión 2, siendo una marca registrada de la Fundación de Software Apache. Teniendo además una aplicación de administración intuitiva basada en web. (13)

Se decide utilizar Apache Tomcat, además de los argumentos dados anteriormente, debido a que el cliente solicito su utilización por ser una herramienta de fácil uso para el personal de la institución en la que va a desplegarse el sistema.

[4] *Integrated Development Enterprise,* Entorno de Desarrollo Integrado

1.4.4. Tecnologías

Grails 1.3.7

Grails es un framework [5] web de código abierto compatible con Java. Está construido sobre cinco fuertes pilares:

- Groovy para la creación de propiedades y métodos dinámicos en los objetos de la aplicación.
- Spring para los flujos de trabajo e inyección de dependencia.
- Hibernate para la persistencia.
- SiteMesh para la composición de la vista.
- Ant para la gestión del proceso de desarrollo. (14)

Además, desde el punto de vista del diseño, combina dos principios fundamentales:

- Convención mejor que configuración: aunque en una aplicación Grails existen archivos de configuración es muy poco probable que tengan que ser editados manualmente ya que el sistema se basa en convenciones. (14)
- No te repitas (*Don't Repeat Yourself*, DRY por sus siglas en inglés): la participación de Spring Container en Grails permite la inyección de dependencia mediante IoC[6] (*Inversion of Control*), de forma que cada actor en la aplicación debe definirse una única vez, haciéndose visible a todos los demás de forma automática. (14)

Grails se puede ampliar a través de plugins. Por su dinamismo es capaz de acortar el ciclo de desarrollo, ahorrando tiempo de trabajo y agilizándolo. (14)

Se decide utilizar Grails como framework de desarrollo porque permite a los desarrolladores Java trabajar sobre un ambiente confiable, familiar y robusto, sin necesidad de utilizar extensos archivos de configuración y con la posibilidad de integrar sistemas existentes, logrando así un proceso de desarrollo ágil y productivo.

Dojo Toolkit 1.5

Dojo Toolkit es un framework Javascript que permite el desarrollo de aplicaciones web enriquecidas en el cliente y Ajax. Es popular porque está integrada en numerosos IDEs y otros frameworks para

[5] Marco de trabajo
[6] Inversión de Control

desarrollo de webs. Contiene un sistema de empaquetado inteligente, los efectos de UI[7], drag and drop[8] APIs[9], widget [10]APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX[11]. (15)

Es conocido como "la navaja suiza del ejército de las bibliotecas JavaScript" y proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos. Se define como marco de trabajo de presentación por sus estilos visuales y su implementación de AJAX.

Oracle Database 11g Release 2

Oracle es un Sistema de Gestión de Base de Datos Objeto-Relacional (Object-Relational Data Base Management System, ORDBMS por sus siglas en inglés), desarrollado por Oracle Corporation y basado en la tecnología cliente/servidor. Es un manejador de BD relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información. Está considerado como uno de los sistemas de BD más completos, destacando entre sus principales fortalezas: el soporte de transacciones, la estabilidad, escalabilidad y el soporte multiplataforma.

Se decide utilizar Oracle 11g R2 como sistema gestor para la BD de los módulos Vínculos y Trabajo ya que ayuda a administrar y almacenar grandes volúmenes de datos. También se puede afirmar que constituye una herramienta de administración gráfica cómoda de utilizar y que permite el análisis de la información. (16)

1.4.5. Lenguajes

HTML

HTML[12] es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados.

[7] *User Interface*, Interfaz de usuario
[8] Filosofía "Agarrar y soltar" aplicable a los componentes
[9] Interfaz de Programación de Aplicaciones (*Application Programming Interface* por sus siglas en inglés): es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.
[10] Identificador de componente
[11] *Javascript Asíncrono And XML*: es una técnica de desarrollo web para crear aplicaciones interactivas.

Una de las características esenciales de este lenguaje es la universalidad, y significa que prácticamente cualquier ordenador, independientemente del sistema operativo, puede leer o interpretar una página web. Sin embargo, el aspecto de dichas páginas depende del tipo de ordenador, del monitor, la velocidad de la conexión de Internet y, por último del navegador. Aunque no todos los navegadores muestran una misma página web de la misma forma.

Se decide utilizar porque es un 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.

JavaScript.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (17)

Es un lenguaje de programación ligero y orientado a objetos. El corazón del lenguaje se embebe en los navegadores para interpretar los códigos (scripts) que se escriben en las páginas. Con este lenguaje script se pueden generar páginas dinámicamente en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página. JavaScript es manejado por eventos, independiente de cualquier plataforma, permite un desarrollo rápido y es muy fácil de aprender. No incluye complejas reglas sintácticas.

JavaScript se usa para la validación del lado del cliente y también es utilizado para integrar un navegador web, permitiendo el desarrollo de interfaces de usuario mejoradas, páginas web dinámicas.

Groovy 1.7.8.

Groovy es un lenguaje de programación orientado a objetos, ágil, dinámica y de código abierto.

Groovy tiene la posibilidad de compilarse a bytecode y ejecutarse en el entorno de la Máquina Virtual de Java (Java Virtual Machine, JVM por sus siglas en inglés), con lo que es posible utilizar

[12] *HyperText Markup Language,* Lenguaje de Marcado de Hipertexto

prácticamente cualquier Interfaz de Programación de Aplicaciones (Application Programming Interface, API por sus siglas en inglés) en Java. El hecho de que se pueda utilizar el API de Java, permite a los desarrolladores Java aprender el lenguaje de forma rápida y la curva de aprendizaje se reduce bastante en comparación con otros lenguajes que se pueden ejecutar en la JVM. (18)

Las características que resaltan de Groovy son:

- Cuenta con la potente base de Java y algunas características de lenguajes de programación tan productivos como Python, Ruby o Smalltalk.
- El código generado es fácil de leer y mantener.
- Permite la creación de lenguajes DSL (*Domain Specific Language*).
- Puede mezclarse con aplicaciones Java al permitir la compilación en *bytecode.*
- Tipado estático y dinámico.
- Existencia de *closures* (básicamente es un trozo de código empaquetado como un objeto y definido entre llaves. Actúa como un método, al cual se le pueden pasar parámetros y pueden devolver valores). (18)

Se decide usar Groovy como lenguaje de programación ya que es el lenguaje que utiliza el marco de trabajo elegido para el desarrollo de los módulos Vínculos y Trabajo, Grails. Además, este lenguaje tiene una sintaxis muy parecida a Java; la mayor parte del código escrito en Java es totalmente válido en Groovy por lo que este lenguaje es de muy fácil adopción para programadores Java.

UML

UML[13] se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software". Es un sistema notacional (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. (19)

UML cuenta con varios tipos de modelos, los cuales muestran diferentes aspectos de las entidades representadas. Uno de los objetivos principales de la creación de UML fue posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. (20)

Se decide utilizar UML como lenguaje de modelado ya que facilita a los desarrolladores la visualización de los resultados del trabajo realizado en esquemas o diagramas estandarizados y trabaja

[13] *Unified Model Language,* Lenguaje Unificado de Modelado

correctamente con la mayoría de los procesos de desarrollo existentes. Este es el lenguaje que usa el Visual Paradigm, que es la herramienta seleccionada para la modelación del sistema.

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (21)

Se utiliza este lenguaje debido a su estrecha similitud con el lenguaje Groovy, gracias a que todo lo implementado en Java es válido en Groovy.

1.5. Conclusiones parciales

En este capítulo se realizó un estudio referente a algunos sistemas penitenciarios, tanto nacionales como internacionales, tales como el SIGEP, el Sistema penitenciario del gobierno de Panamá, el SACORE, el SAIDEP y el SACDEP, lo que permitió comprobar y analizar su forma de proceder y gestionar la información, llegando a la conclusión de que dichos sistemas no cumplen con las características que van acorde a las especificaciones del SIDEP. Además, se analizó la metodología de desarrollo utilizada, en este caso RUP; las herramientas, entre las que se encuentran Visual Paradigm para UML, Embarcadero ER/Studio, NetBeans y Apache Tomcat; mientras que las tecnologías propuestas son Grails, Dojo Toolkit y Oracle Database 11g R2; como lenguajes HTML, JavaScript, Groovy, UML y Java.

El análisis realizado en este capítulo constituye un elemento relevante para lograr dar una solución óptima al problema propuesto.

CAPÍTULO 2. DISEÑO DEL SISTEMA

2.1. Introducción

En el presente capítulo se mostrará una visión previa del sistema, teniendo en cuenta primeramente los elementos descriptivos del negocio y los requisitos correspondientes a los módulos Vínculos y Trabajo, dando paso a la explicación de las principales funcionalidades de dichos módulos. También se dará información sobre temas más enmarcados a cuestiones propias del sistema, como son la arquitectura, los patrones de diseño, los diagramas de clases, los diagramas de colaboración y el diseño de la BD. Además, se hará uso de métricas útiles para la validación del diseño de la solución de acuerdo al problema propuesto.

2.2. Reglas del negocio

Para lograr el funcionamiento eficiente de una organización deben existir determinadas reglas de negocio. Estas reglas permiten establecer un control en los procesos que se desarrollan, asegurando que las actividades que se llevan a cabo cumplan con las condiciones y políticas que impone la organización. A continuación se muestran las reglas del negocio correspondientes a los módulos Vínculos y Trabajo.

Módulo Vínculos:

No	Tipo	Nombre	Descripción
1	Textual	Vínculos automáticos	En los vínculos automáticos, cuando es una incidencia de agresión, intento de asesinato o asesinato calificar el tipo de vínculo como enemigo y la influencia como negativa.

Tabla 1 Reglas del negocio del módulo Vínculos

Módulo Trabajo:

No	Tipo	Nombre	Descripción
1	Textual	Organismos priorizados con los que se realizan contratos de Trabajo	➢ Ministerio de Economía y Planificación (MEP); ➢ Ministerio del Trabajo y Seguridad Social (MTSS); ➢ Consejos de la Administración Municipal y Provincial; ➢ Ministerio de Educación y Superior (MINED y MES); ➢ Asociación de Innovadores y Racionalizadores (ANIR); ➢ Ministerio de la Agricultura (MINAGRI); ➢ Ministerio de la Construcción (MICONS); ➢ Ministerio de la Industria Básica (MINBAS); ➢ Ministerio del Interior (Agropecuaria, Construcción, Inversiones, Transporte y Provari) ➢ Ministerio de las Fuerzas Armadas Revolucionarias (Agropecuaria, Construcciones y otros); ➢ Instituto Nacional del Deportes, Cultura Física y Recreación (INDER).
2		Prioridades para conceder el trabajo a los internos	➢ Oficio. ➢ Jóvenes. ➢ Responsabilidad civil. ➢ Ayuda económica por asistencia social. ➢ Los que tengan personas a su abrigo.
3		Los Jefes de	➢ Oficial activo.

		brigada pueden ser	➢ Jubilado. ➢ Agente de protección.

Tabla 2 Reglas del negocio del módulo Trabajo

2.3. Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen. (22) Se puede afirmar que los requisitos funcionales constituyen un elemento fundamental a la hora de desarrollar cualquier sistema.

A continuación se definen los requisitos funcionales correspondientes a los módulos Vínculos y Trabajo.

Módulo Vínculos:

Numeración	Requisitos funcionales
1	Registrar un vínculo externo
2	Modificar un vínculo externo
3	Consultar un vínculo externo
4	Eliminar un vínculo externo
5	Registrar un vínculo interno
6	Modificar un vínculo interno
7	Consultar un vínculo interno
8	Eliminar un vínculo interno
9	Asociar nuevo vínculo por intento de asesinato
10	Asociar nuevo vínculo por asesinato
11	Asociar nuevo vínculo por agresión

Tabla 3 Requisitos funcionales del módulo Vínculos.

Módulo Trabajo:

Numeración	Requisitos funcionales
1	Registrar unidad productiva
2	Modificar unidad productiva
3	Eliminar unidad productiva
4	Registrar contrato de trabajo
5	Modificar contrato de trabajo
6	Registrar brigada productiva
7	Modificar brigada productiva
8	Eliminar brigada productiva
9	Registrar incorporación
10	Registrar desincorporación
11	Consultar reporte de desincorporados
12	Registrar asistencia al trabajo

Tabla 4 Requisitos funcionales del módulo Trabajo

2.4. Diagrama de casos de uso del sistema

Los casos de uso (CU) constituyen una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de CU sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. (7)

En las siguientes figuras se muestran los diagramas de CU correspondientes a los módulos Vínculos y Trabajo, en el caso del primero se reflejan 2 CU y en el segundo 6 CU, cada uno de estos identificados a partir de los requisitos funcionales mencionados anteriormente.

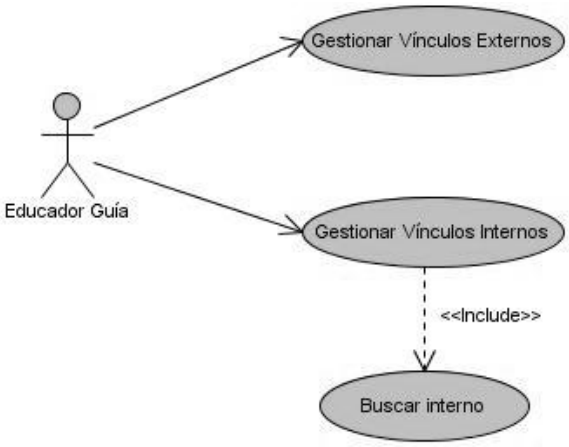

Figura 2 Diagrama de CU del sistema del módulo Vínculos

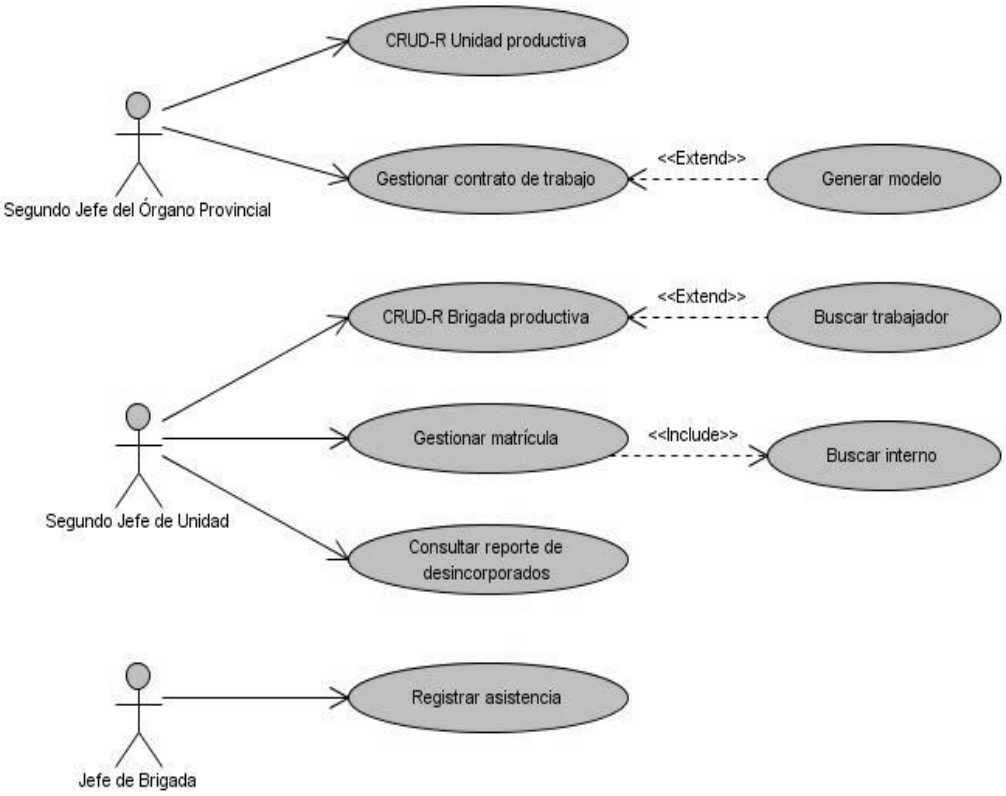

Figura 3 Diagrama de CU del sistema del módulo Trabajo

2.5. Descripción de las funcionalidades de los módulos Vínculos y Trabajo

En la siguiente tabla se muestran las funcionalidades con que cuentan los módulos Vínculos y Trabajo, así como la descripción de las acciones que realizan.

Módulo Vínculos:

CU	Nombre	Descripción
CU1	**Gestionar Vínculos Externos**	El caso de uso permite registrar, actualizar, consultar o eliminar un vínculo externo.
CU2	**Gestionar Vínculos Internos**	El caso de uso permite registrar, actualizar, consultar o eliminar un vínculo interno.
CU3	**Asociar nuevo vínculo por intento de asesinato**	El caso de uso permite asociar a un vínculo interno los vínculos negativos de acuerdo a una incidencia de intento de asesinato. Este caso de uso lo realiza el sistema automáticamente.
CU4	**Asociar nuevo vínculo por asesinato**	El caso de uso permite asociar a un vínculo interno los vínculos negativos de acuerdo a una incidencia de asesinato. Este caso de uso lo realiza el sistema automáticamente.
CU5	**Asociar nuevo vínculo por agresión**	El caso de uso permite asociar a un vínculo interno los vínculos negativos de acuerdo a una incidencia de agresión. Este caso de uso lo realiza el sistema automáticamente.

Tabla 5 Relación de CU del módulo Vínculos.

Módulo Trabajo:

CU	Nombre	Descripción
CU1	**CRUD-R Unidad productiva**	El caso de uso permite registrar, eliminar o actualizar los datos de la unidad productiva
CU2	**Gestionar contrato de trabajo**	El caso de uso permite registrar o actualizar

		los datos de un contrato de trabajo.
CU3	**CRUD-R Brigada productiva**	El caso de uso permite registrar, eliminar o actualizar los datos de la brigada productiva.
CU4	**Gestionar matrícula**	El caso de uso permite incorporar o desincorporar los internos de las brigadas productivas registradas.
CU5	**Consultar reporte de desincorporados**	El caso de uso permite consultar el listado de los internos que han sido desincorporados de una brigada productiva.
CU6	**Registrar asistencia**	El caso de uso permite registrar la asistencia de los internos al trabajo.

Tabla 6 Relación de CU del módulo Trabajo.

2.6. Marco de trabajo de desarrollo web

Grails es un marco de trabajo que se utiliza para el desarrollo de aplicaciones web sobre la plataforma Java Enterprise Edition. Pretende ser un marco de trabajo altamente productivo proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador. Está desarrollado sobre el lenguaje de programación Groovy y está basado en el patrón Modelo-Vista-Controlador (MVC), el cual propone una organización en tres capas, pero al ser una aplicación Grails este utiliza otra capa más; finalmente las capas quedan conformadas por el modelo o capa de datos, la vista o capa de presentación, el controlador o capa de control y la capa para los servicios. Grails proporciona un conjunto de adaptadores y código de interfaz que permite gestionar los diferentes componentes en cada capa del desarrollo, y asegurar que dichos adaptadores funcionan correctamente con tantas combinaciones de los componentes como sea posible. (14)

Sus principales características son:

1. **Alta productividad**: grails tiene tres características que intentan incrementar su productividad comparándolo con los Framework Java tradicionales:
   - ✓ Inexistencia de configuración XML.

- ✓ Entorno de desarrollo preparado para funcionar desde el primer momento.
- ✓ Funcionalidad disponible mediante métodos dinámicos.

2. **Scaffolding:** el scaffolding es una característica de determinados frameworks que permite la generación automática de código para las cuatro operaciones básicas de cualquier aplicación, que son: la creación, lectura, edición y borrado, lo que en inglés se conoce como CRUD (create, read, update and delete). El scaffolding en Grails se consigue escribiendo muy pocas líneas de código, con lo que se puede lograr la concentración en la especificación de las propiedades, comportamientos y restricciones de las clases de dominio.
3. **Integración con la plataforma Java:** la mejor característica que Grails ofrece en este ámbito es una integración transparente con clases mapeada mediante el Framework Hibernate. Esto significa que aplicaciones existentes que utilicen Hibernate pueden utilizar Grails sin recompilar el código o reconfigurar las clases, aprovechando los métodos de persistencia que se mencionan anteriormente. Una consecuencia es que se puede utilizar scaffolding con las clases Java mapeadas con Hibernate. Otra consecuencia es que las capacidades de Grails están totalmente disponibles para estas clases y las aplicaciones que las usan.
4. **Persistencia:** el modelo de datos en Grails se graba en la base de datos utilizando GORM (Grails Object Relational Mapping). Las clases de dominio se guardan en el directorio grails-app/domain.
5. **Plugins:** grails no siempre es la solución a cualquier problema que se pueda plantear en el desarrollo de aplicaciones web. Para ayudar, grails dispone de una arquitectura de plugins para seguridad, AJAX, testeo, búsqueda, informes y servicios web. Este sistema de plugins hace que añadir complicadas funcionalidades a la aplicación se convierta en algo muy sencillo. (14)

2.7. Arquitectura del sistema

La arquitectura del sistema se sustenta en una arquitectura n-capas estructurada por grails la cual utiliza Spring MVC como la infraestructura de aplicaciones web subyacente, en la cual se separa la presentación de la aplicación de la lógica del negocio permitiendo cambiar fácilmente el aspecto de la aplicación sin modificar su comportamiento. Entre las capas que utiliza grails se encuentran: Capa de Vistas, Capa de Controladores y Capa de Datos. La gestión de la lógica de negocio por parte de los controladores significa que son los responsables de que esta se aplique, lo cual no quiere decir que se deba aplicar dicha lógica en los controladores, normalmente esta lógica se deberá implementar en la capa de servicio por esta razón grails lo incorpora como parte de sus capas.

La manera de interactuar dichas capas se evidencia de la siguiente manera:

Los controladores se encargan de mostrar las vistas y controlar el flujo de datos entre las vistas y las entidades, recogen la información y la envían a los servicios, los cuales son los encargados de la lógica de negocio, para posteriormente realizar cambios en la base de datos.

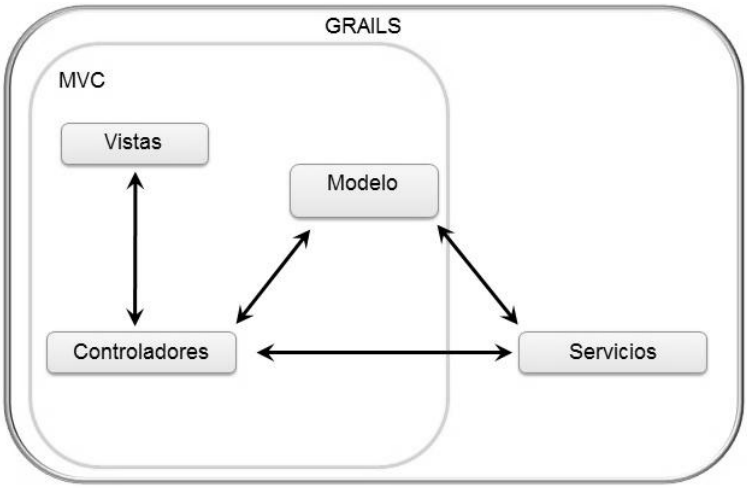

Figura 4 Arquitectura Grails MVC.

Vistas: grails utiliza para la interacción con el usuario la tecnología Página Servidora de Java (JSP), pero basada en una implementación mediante GSP[14], que es una extensión de JSP y puede incluir Groovy. Grails permite a los desarrolladores mezclar etiquetas de lenguajes de marcas tradicionales como HTML con código Java para producir vistas dinámicas. Las vistas son los recursos, que junto al modelo generado por los controladores, le permiten al cliente visualizar la información, estos pueden ser páginas HTML, documentos en formato PDF[15], hojas de cálculo, entre otras. Las mismas están representadas en el paquete de clase View and Layouts.

La capa de vistas agrupa los siguientes componentes:

1. gsp
2. taglib[16]
3. javascript
4. css

[14] GSP: Páginas de Servidor Groovy, por sus siglas en inglés (*Groovy* Servers Pages).
[15] *PDF* : Formato de documento portátil por sus siglas en inglés (*portable document format*).
[16] *Taglib: Librerias de etiquetas.*

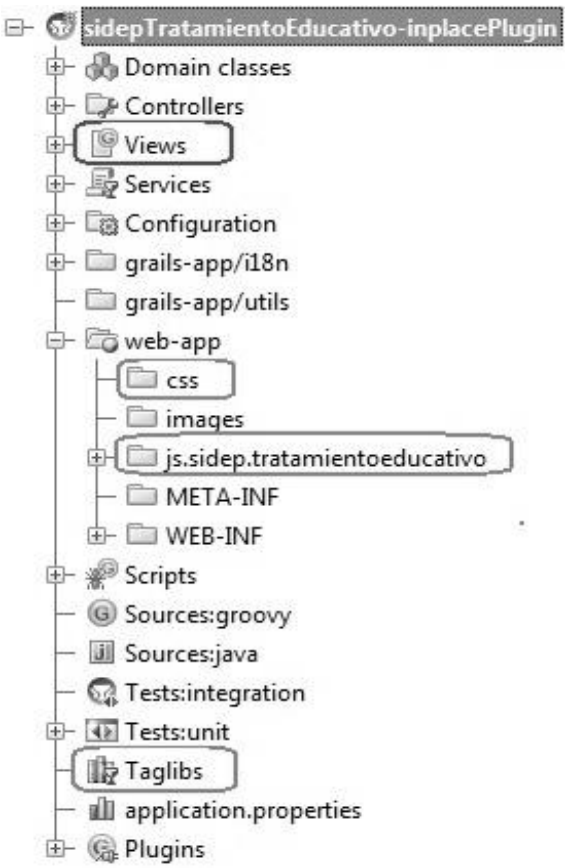

Figura 5 Componentes que intervienen en la capa de vistas.

Controladores: un controlador de Grails es una clase responsable del manejo de los pedidos provenientes de la aplicación. El controlador recibe la petición, realiza algún trabajo potencial con la misma y, finalmente, decide que sucederá a continuación, lo cual puede incluir algunos de los siguientes flujos:

1. Ejecutar otra función de controlador.
2. Mostrar una vista.
3. Mostrar información directamente con la respuesta de la petición.

Proveen la entrada principal para cualquier aplicación de Grails, coordinando los pedidos entrantes, delegando hacia los servicios o clases de dominio información para lograr el manejo de la lógica de negocio y mostrando las vistas. Las clases controladoras están enmarcadas en los paquetes Controllers.

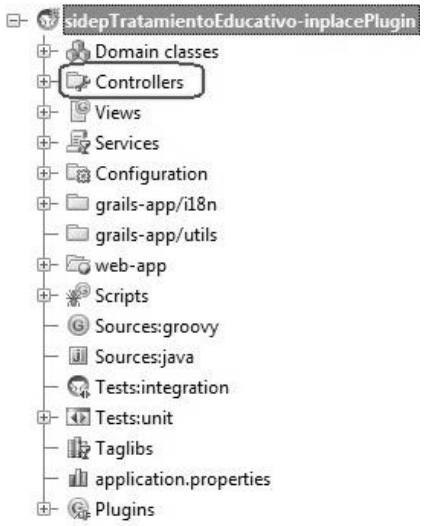

Figura 6 Paquete de controladores.

Datos: maneja los objetos de acceso a datos abstrayéndolos del mecanismo de persistencia usado; a través de interfaces que exponen las operaciones de persistencia. Grails para evitar trabajar directamente con un gestor de base de datos y sus tablas permite trabajar con objetos en su lugar, utilizando Hibernate 3 como herramienta de mapeo objeto-relacional (ORM); pero esta vez, dada la naturaleza dinámica de Grails y la adopción del convenio sobre la configuración, crea sobre una versión superior de una nueva implementación de Hibernate llamado Grails objeto mapeo relacional (*GORM*), que simplifica el trabajo con Hibernate y elimina cualquier configuración externa.

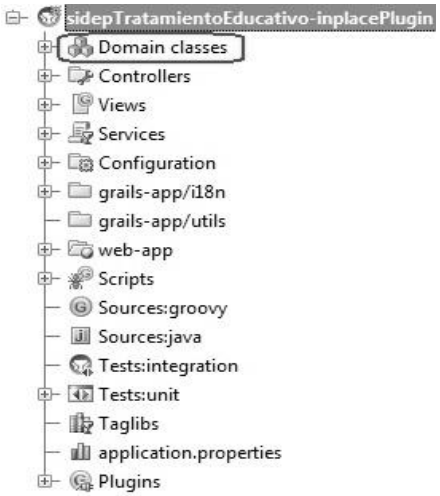

Figura 7 Paquete de clases del dominio.

Servicios: En esta capa se encapsula toda la lógica de la aplicación, las funcionalidades implementadas en estas clases son utilizadas por los controladores, además permiten exponen

algunos procesos de negocio a través de interfaces de servicios. Las clases incluidas en esta capa radicarán según la arquitectura propuesta por Grails en el paquete Services.

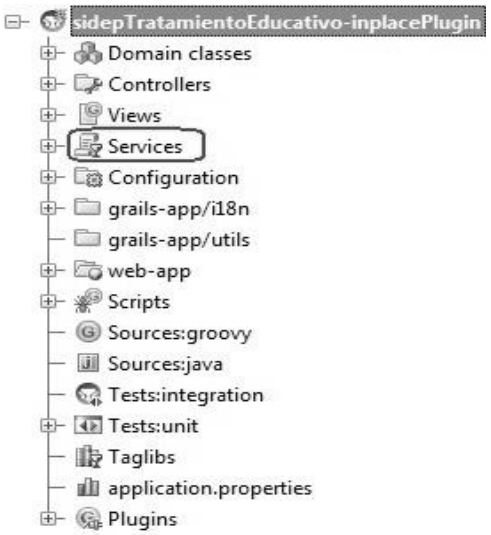

Figura 8 Paquete de servicios.

2.8. Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Estos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (23)

A continuación se explican los patrones de diseño utilizados en el desarrollo de la solución.

2.8.1. Patrones de diseño GRASP

Los GRASP[17] son patrones de diseño que se usan para asignar responsabilidades a una clase. A continuación se muestran los patrones utilizados:

Experto

Asigna una responsabilidad al más competente en información, donde la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos (OO). Es el patrón que más se usa para asignar responsabilidades.

[17] *General Responsibility Assignment Software Patterns*, Patrones Generales de Asignación de Responsabilidades

Este patrón se evidencia en la clase entidad “InternoVinculo” ya que contiene toda la información referente a la relación entre un interno y un vínculo. Por tal razón, siempre que se haga la gestión de la relación entre vínculos e internos es necesario consultar esta clase.

Creador

Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas OO[18]. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.
Este patrón se evidencia en el controlador listarVinculoExternoController ya que esta clase crea objetos de distintas clases del dominio tales como: Vinculo, VinculoInterno, Nacional, Extranjero y InternoVinculo.

Bajo Acoplamiento

Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad.

Este patrón está presente en el listarVinculoExternoController ya que el mismo tiene solo una instancia de un servicio en este caso listarVinculosExternosService, de tener un gran número de instancias el acoplamiento seria elevado.

Alta Cohesión

Asigna una responsabilidad de modo que la unión se mantenga a gran escala, es decir, asigna a las clases responsabilidades para que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.

La alta cohesión se manifiesta especialmente en el proceso de gestión de los vínculos, teniendo en cuenta que los mismos pueden ser internos, nacionales o extranjeros, por tal motivo las clases entidades, vínculo, vínculo interno, nacional y extranjero tienen información necesaria sobre los

[18] Orientado a Objeto

vínculos, para no sobrecargar una sola clase con toda esta información y para que trabajen todas en el proceso de la gestión de los vínculos en conjunto.

Controlador

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el que recibe los datos del usuario y los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

Los controladores representan este patrón claramente debido a que los mismos son los encargados del controlar los eventos y mensajes entre las vistas y los servicios.

2.8.2. Patrones de diseño GOF

Los patrones GOF[19] son patrones de diseño que definen una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

Estos patrones se dividen en tres categorías: los creacionales, los estructurales y los de comportamiento. Seguidamente se explicarán los patrones empleados en la solución del problema propuesto.

Singleton (instancia única)

Este patrón consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella.

Por defecto todos los servicios son *singleton*, ya que solo existe una instancia de la clase que se inyecta en todos los artefactos que declaran la variable correspondiente. 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

[19] *Gand of Four*, Banda de los Cuatro

puede variar utilizando una variable *scope*[20] con valor *session*[21] en la aplicación, permitiendo crear una instancia nueva del servicio para cada sesión de usuario.

2.8.3. Otros patrones de diseño utilizados

Modelo-Vista-Controlador

El patrón de diseño MVC se utiliza para separar la información, la salida y el procesamiento de los datos de la aplicación. La aplicación se divide en tres elementos: el modelo, la vista y el controlador; cada elemento gestiona una parte distinta del proceso.

1. Modelo: son los objetos de la aplicación, también conocida como lógica de negocio, o lógica de aplicación.
2. Vista: especifica la visualización de los datos, algunas veces conocida como lógica de presentación.
3. Controlador: es el coordinador entre estos dos últimos, es decir, define la forma en que la interfaz de usuario reacciona ante la entrada de usuario.

MVC desacopla el concepto de interfaz de usuario y lógica de negocio para aumentar la flexibilidad y modularidad del software, posiblemente permitiendo que el código pueda ser reutilizado.

Inversión de Control (IoC)

Inversión de Control (*Inversion of Control*, IoC según sus siglas en inglés) plantea que las dependencias de un componente no deben gestionarse desde el propio componente para que este solo contenga la lógica necesaria para hacer su trabajo. (19)

Este patrón está presente en el controlador listarVinculoExternoController ya que el mismo define una variable del servicio listarVinculosExternosService, luego de esto grails se encarga de crear una instancia de la clase servidora e inyectarla en el controlador.

[20] Scope: alcance, variable que se declara en un servicio para limitar hasta qué punto puede ser privado el mismo.
[21] Session: sesión, valor que puede tomar la variable scope para crear una instancia nueva del servicio para cada sesión de usuario.

2.9. Diagrama de clases

2.9.1 Descripción de las clases más significativas

En esta sección se muestra mediante tablas las clases más significativas que han sido creadas para dar desarrollo a los módulos Vínculos y Trabajo. Las tablas están representadas por clases de dominio, controladoras, servidoras, HTML y JavaScript. Se describen los datos que son utilizados, así como el funcionamiento que se realiza en la lógica del negocio.

Las mismas se encuentran representadas en los anexos (Anexo 1).

2.10. Diagramas de clase del diseño

Los diagramas de clase de diseño están formados por el conjunto de clases con responsabilidades específicas y sus respectivas relaciones. Estos muestran gráficamente las descripciones de las clases del sistema y sus interfaces. Normalmente contienen las siguientes informaciones:

1. Clases, asociaciones y atributos.
2. Interfaces, con sus operaciones y constantes.
3. Métodos.
4. Información sobre los tipos de los atributos.
5. Dependencias.

A continuación se muestran los diagramas de clases del diseño de los CU “Gestionar vínculo externo” y “Gestionar contrato de trabajo”. La gestión de los vínculos externos consiste en registrar, actualizar, mostrar los vínculos de un determinado interno y en la eliminación de los vínculos según sea el caso, mientras que la gestión de los contratos de trabajo se centran en registrar un contrato, asignándole al mismo una unidad productiva y su fuerza de trabajo, además de actualizarlo.

El resto de los diagramas de clases del diseño se encuentran incluidos en los anexos (Anexo 2).

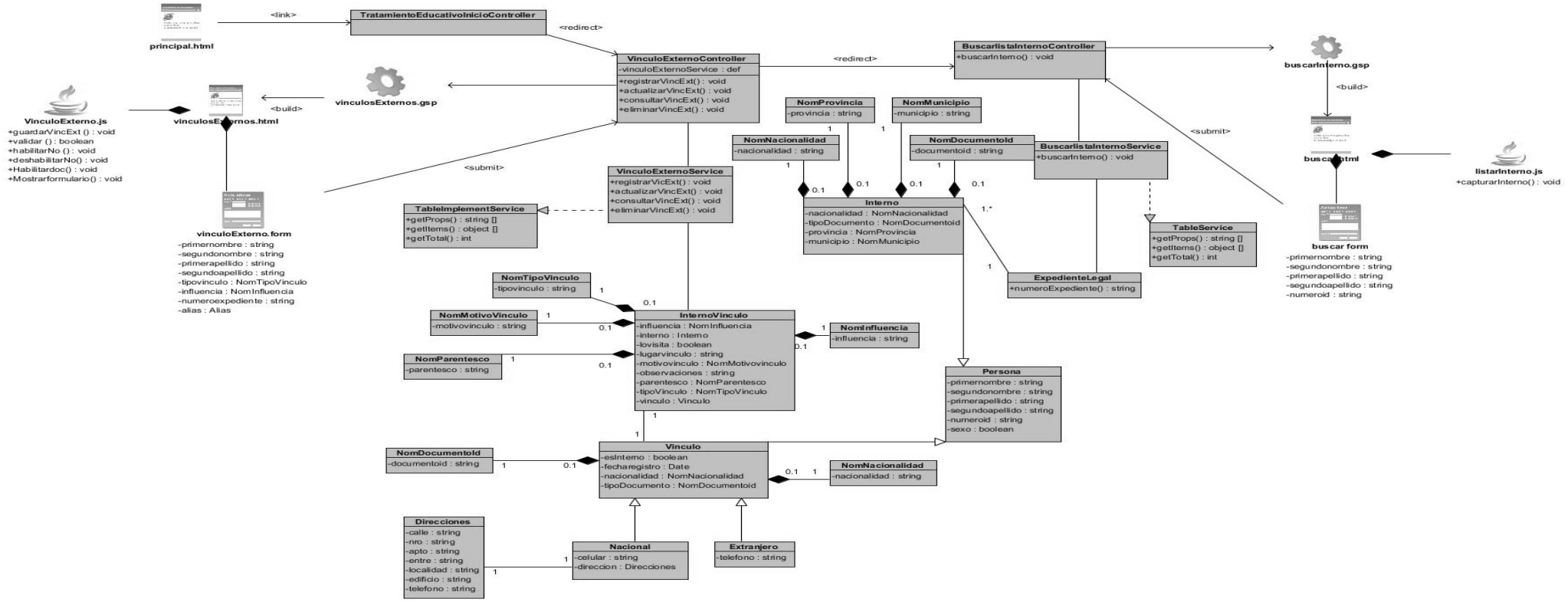

Figura 9 DCD[22] Gestionar Vínculo Externo.

[22] Diagrama de Clases del Diseño

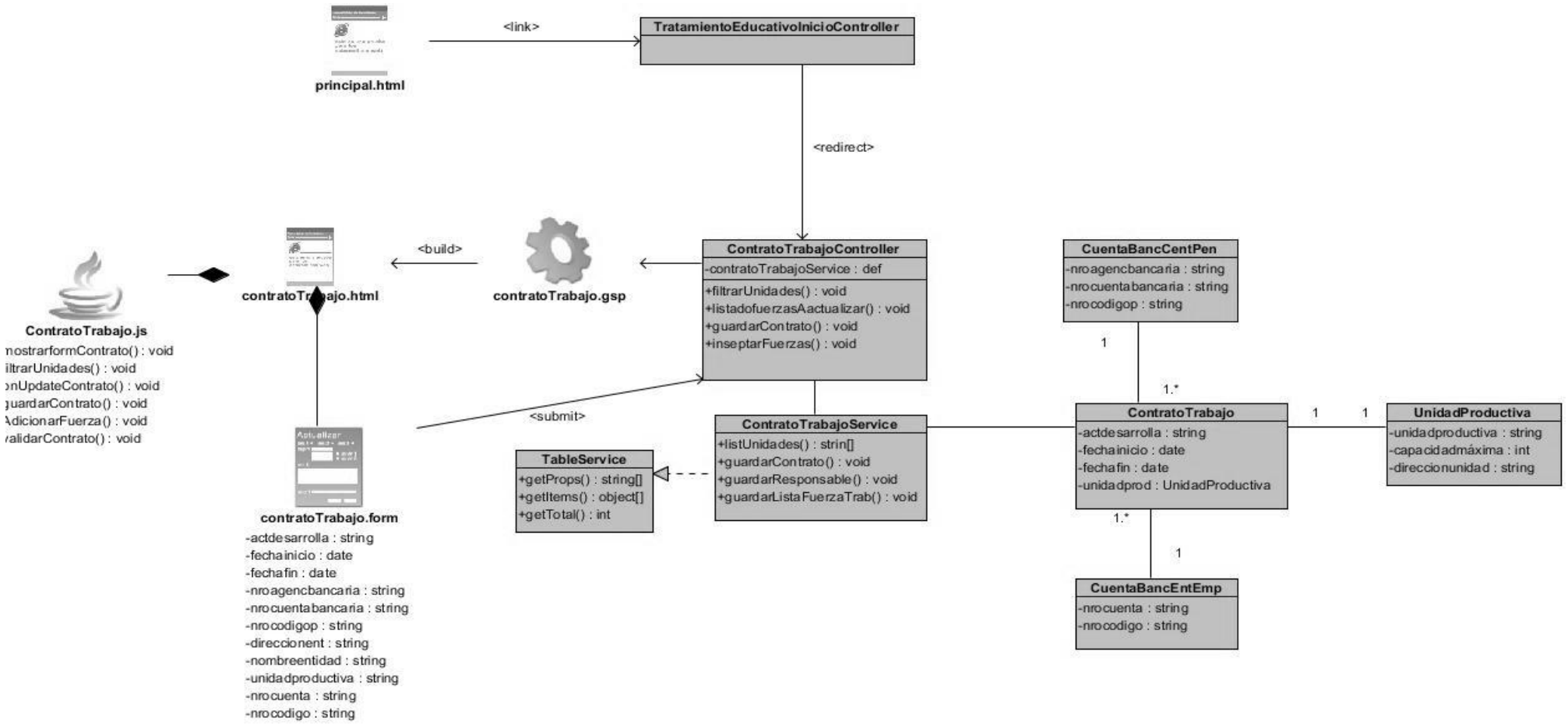

Figura 10 DCD Gestionar contrato de trabajo.

2.11. Diagramas de colaboración

Un diagrama de interacción contiene un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos, organizados lógica y secuencialmente. Dentro del grupo de los diagramas de interacción se encuentran los diagramas de colaboración, los cuales representan una combinación de información tomada desde el diagrama de clases, diagrama de secuencia y diagrama de CU, describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

Seguidamente se muestra el diagrama de colaboración de la funcionalidad “Registrar vínculo externo” correspondiente al caso de uso Gestionar vínculo externo del módulo Vínculos y el diagrama de colaboración de la funcionalidad “Registrar contrato de trabajo” correspondiente al caso de uso Gestionar contrato de trabajo del módulo Trabajo.

El resto de los diagramas de colaboración se encuentran incluidos en los anexos (Anexo 3).

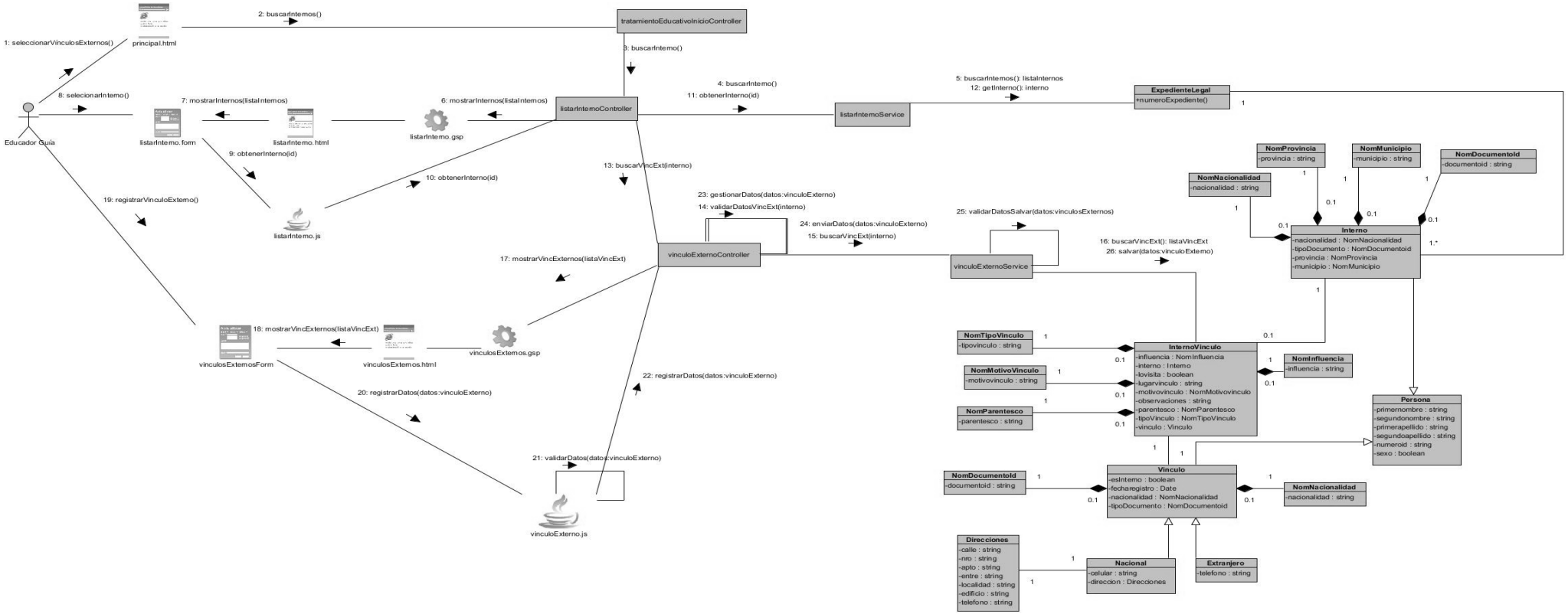

Figura 11 DC[23] Registrar Vínculo Externo.

[23] Diagrama de Colaboración

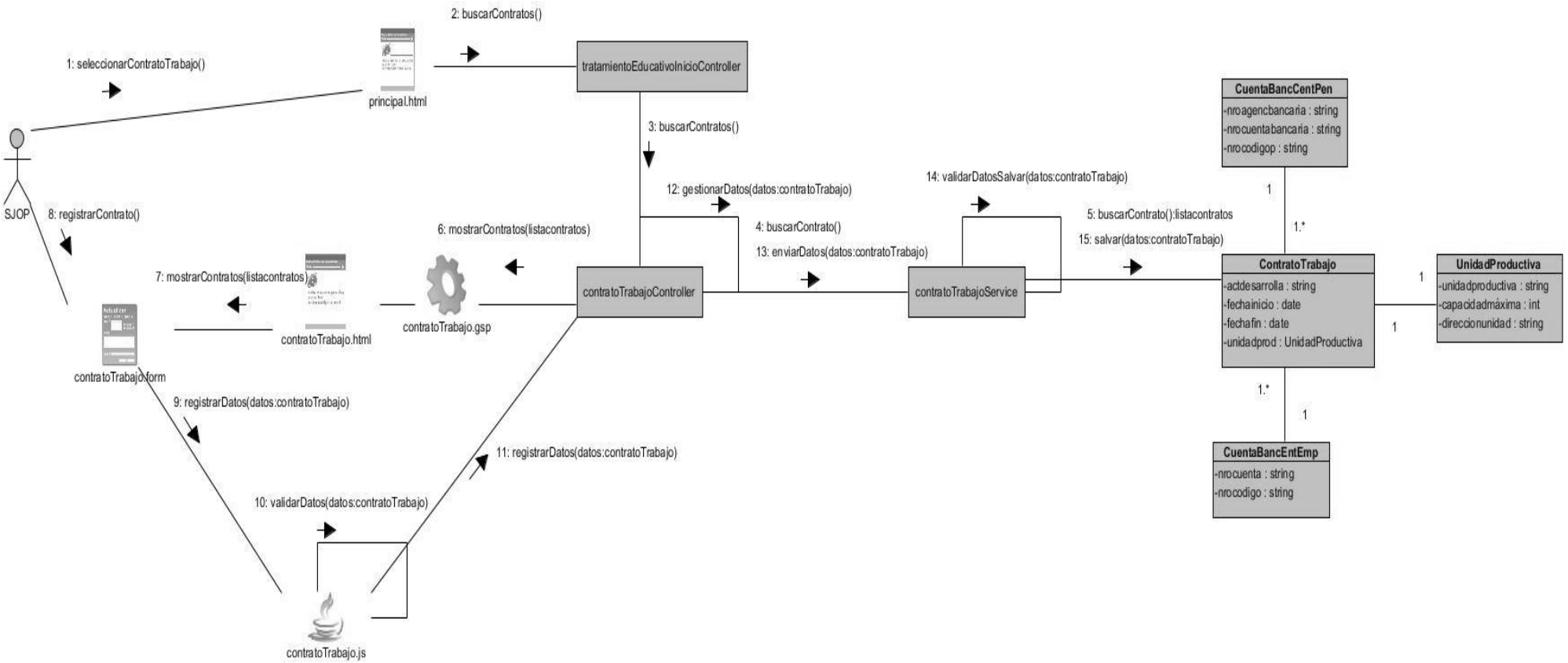

Figura 12 DC Registrar contrato de trabajo.

2.12. Diseño de la Base de Datos

Una Base de Datos (BD) correctamente diseñada permite obtener acceso a información exacta y actualizada. Puesto que un diseño correcto es esencial para lograr los objetivos fijados para la BD, parece lógico emplear el tiempo que sea necesario en aprender los principios de un buen diseño ya que, en ese caso, es mucho más probable que la BD termine adaptándose a sus necesidades y pueda modificarse fácilmente.

A continuación se muestran los modelos de BD correspondientes a los módulos Vínculos y Trabajo.

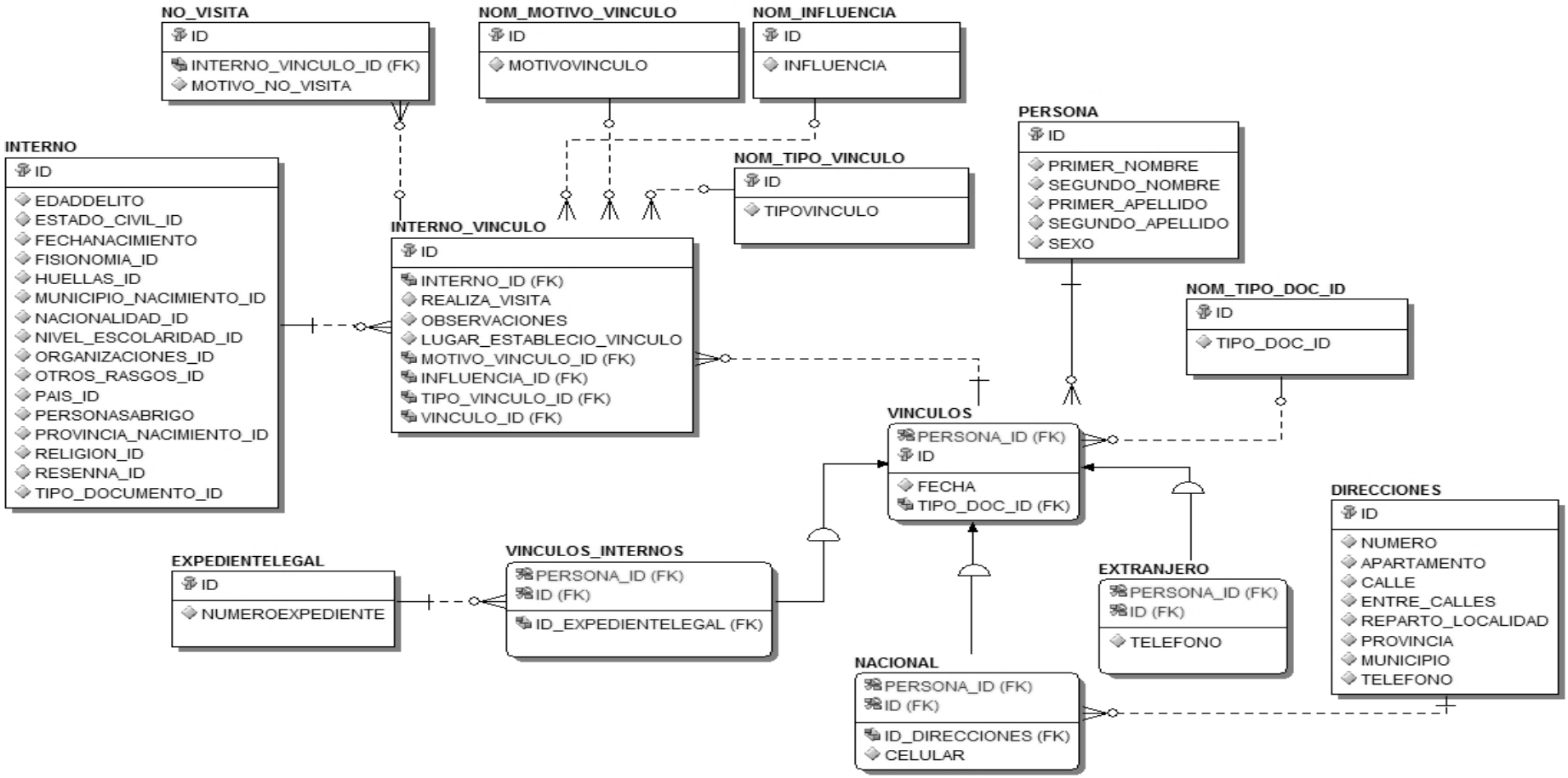

Figura 13 Modelo de datos del módulo Vínculos.

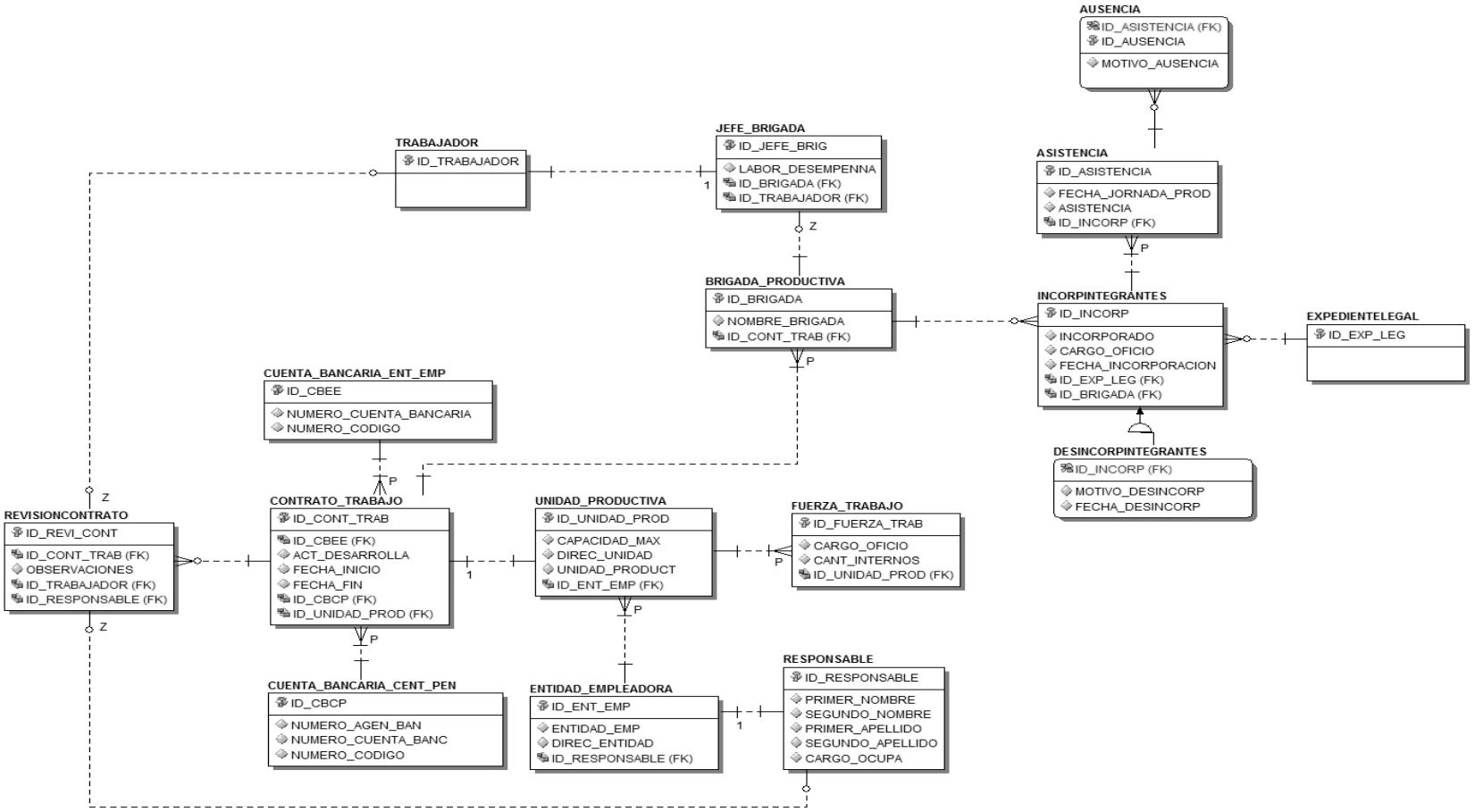

Figura 14 Modelo de datos del módulo Trabajo.

2.13. Descripción de las tablas de los módulos

Las tablas contenidas en la BD registran toda la información de los datos que serán manejados en la aplicación, por tal motivo es necesario conocer con exactitud el significado de cada uno de los elementos de estas tablas. Para mayor información remitirse a los anexos (Anexo 4).

2.14. Métricas para la validación del diseño

Se define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Según Pressman, las métricas ayudan a la evaluación de los modelos de análisis y de diseño, donde proporcionan una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudan en el diseño de pruebas más efectivas. Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento. (24)

Para medir la calidad del diseño se utilizarán métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objetos referenciadas por Pressman. Se utilizaron las siguientes métricas para la validación del diseño: tamaño operacional de clase, relaciones entre clases, niveles de árbol de herencia y número de hijos.

Tamaño operacional de clase (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Responsabilidad: un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

Complejidad de implementación: un aumento del TOC implica un aumento de la complejidad de implementación de la clase.

Reutilización: un aumento del TOC implica una disminución del grado de reutilización de la clase. (25)

A continuación se muestran en las siguientes tablas las clasificaciones de TOC que pueden tener las clases para los atributos de calidad descritos anteriormente al obtener un valor de umbral determinado. La variable N será el valor que tomará el umbral de acuerdo a las clases analizadas y la variable Prom será el valor del promedio de la cantidad de procedimientos de las clases.

Responsabilidad:

Categoría	Criterio
Baja	N <= Prom
Media	Prom > N <= 2 * Prom
Alta	N > 2 * Prom

Tabla 7Criterio de evaluación de responsabilidad

Complejidad de implementación:

Categoría	Criterio
Baja	N <= Prom
Media	Prom > N <= 2 * Prom
Alta	N > 2 * Prom

Tabla 8 Criterio de evaluación para la complejidad de implementación

Reutilización:

Categoría	Criterio
Baja	N > 2 * Prom
Media	Prom > N <= 2 * Prom
Alta	N <= Prom

Tabla 9 Criterio de evaluación para la reutilización

La siguiente tabla muestra el total de clases analizadas y el promedio de la cantidad de procedimientos.

Total de clases	Promedio de procedimientos
15	7,26

Tabla 10 Promedio de procedimientos según cantidad de clases

A continuación se muestran los resultados obtenidos tras la aplicación de la métrica TOC para las clases servidoras de los módulos Vínculos y Trabajo.

No	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	ListarInternopaInternoService	5	Baja	Baja	Alta
2	ListarInternoService	5	Baja	Baja	Alta
3	ListarVinculosExternosService	11	Baja	Baja	Alta
4	ListarVinculosInternosService	10	Baja	Baja	Alta
5	VinculoService	7	Baja	Baja	Alta
6	UnidadProductivaService	7	Baja	Baja	Alta
7	AsistenciaService	12	Baja	Baja	Alta
8	BrigadaProductivaService	11	Baja	Baja	Alta
9	BrigadaUnidadesService	4	Baja	Baja	Alta
10	ContratoTrabajoService	10	Baja	Baja	Alta
11	DesincIntegranteService	4	Baja	Baja	Alta
12	FuerzaTrabajoService	7	Baja	Baja	Alta
13	IncorporadoService	11	Baja	Baja	Alta
14	InternoPrioridadService	3	Baja	Baja	Alta
15	TrabajoService	2	Baja	Baja	Alta

Tabla 11 Resultados obtenidos utilizando la métrica TOC

A continuación se muestra la representación gráfica de los resultados obtenidos de acuerdo a la métrica TOC, teniendo en cuenta los criterios de responsabilidad, complejidad de implementación y reutilización.

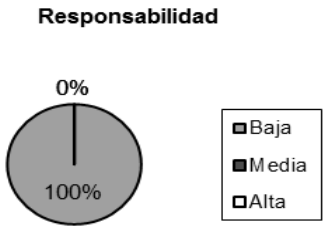

Figura 15 Resultados de la métrica TOC para el atributo de responsabilidad

Luego de haber aplicado esta métrica, se obtuvo como resultado que las 15 clases analizadas del diseño tienen una responsabilidad baja, por lo que se demuestra que las mismas no tienen grandes responsabilidades.

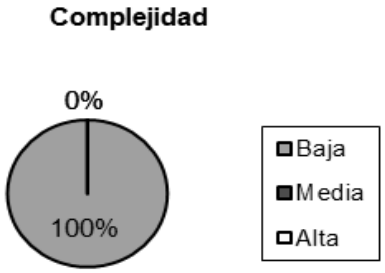

Figura 16 Resultados de la métrica TOC para el atributo de complejidad de implementación

Al realizar esta métrica da como resultado que las clases no tienen una alta complejidad de implementación.

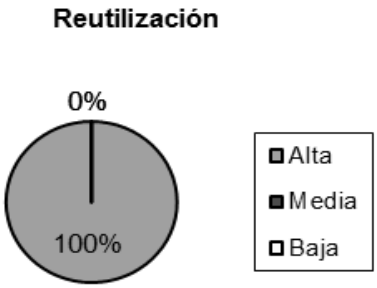

Figura 17 Resultados de la métrica TOC para el atributo de reutilización

Esta métrica arroja como resultado que es bajo el TOC, lo que implica un aumento del grado de reutilización de la clase.

De acuerdo a lo anteriormente planteado se puede afirmar que los resultados obtenidos por la métrica son positivos.

Relaciones entre clases (RC)

Esta métrica tiene en cuenta el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Acoplamiento: un aumento del RC implica un aumento del acoplamiento de la clase.

Complejidad de mantenimiento: un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

Reutilización: un aumento del RC implica una disminución en el grado de reutilización de la clase.

Cantidad de pruebas: un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase. (25)

A continuación se muestran en las siguientes tablas las clasificaciones de RC que pueden tener las clases para los atributos de calidad descritos anteriormente al obtener un valor de umbral determinado. La variable N será el valor que tomará el umbral de acuerdo a las clases analizadas y la variable Prom será el valor del promedio de la cantidad de relaciones de uso.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	N = 0
	Bajo	N = 1
	Medio	N = 2
	Alto	N >2
Complejidad de mantenimiento	Baja	N <= Prom
	Media	Prom > N <= 2 * Prom
	Alta	N > 2 * Prom
Reutilización	Baja	N > 2 * Prom
	Media	Prom > N <= 2 * Prom

	Alta	N <= Prom
Cantidad de pruebas	Baja	N <= Prom
	Media	Prom > N <= 2 * Prom
	Alta	N > 2 * Prom

Tabla 12 Criterios de evaluación para la métrica RC

A continuación se muestran los resultados obtenidos tras la aplicación de la métrica RC a las clases controladoras de los módulos Vínculos y Trabajo.

No	Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
1	ListarInternoController	2	Medio	Baja	Alta	Baja
2	ListarInternopaInternoController	2	Medio	Baja	Alta	Baja
3	ListarVinculoExternoController	1	Bajo	Baja	Alta	Baja
4	ListarVinculoInternoController	1	Bajo	Baja	Alta	Baja
5	AsistenciaController	2	Medio	Baja	Alta	Baja
6	BrigadaProductivaController	2	Medio	Baja	Alta	Baja
7	ContratoTrabajoController	3	Alto	Baja	Alta	Baja
8	DesincIntegranteController	0	Ninguno	Baja	Alta	Baja
9	UnidadProductivaController	1	Bajo	Baja	Alta	Baja

Tabla 13 Resultados obtenidos utilizando la métrica RC

A continuación se muestra la representación gráfica de los resultados obtenidos de acuerdo a la métrica RC, teniendo en cuenta los criterios de cantidad de relaciones de uso (dependencias), acoplamiento, complejidad de mantenimiento, reutilización, cantidad de pruebas y un promedio de cantidad de relaciones de uso de 1.55.

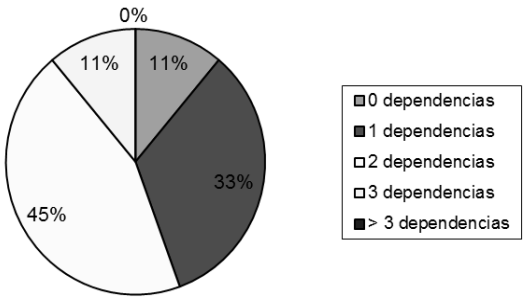

Figura 18 Comportamiento de los valores de dependencia

En la figura anterior se muestran las dependencias que tienen las clases controladoras respecto a los servicios, los cuales se usan para realizar la lógica del negocio.

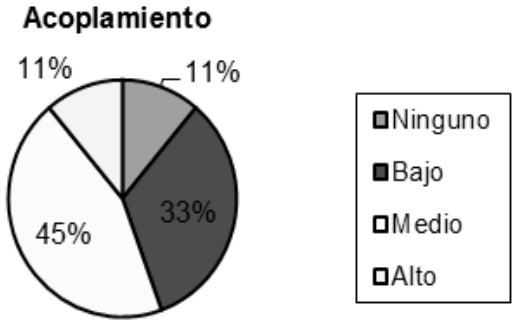

Figura 19 Resultados de la métrica RC para el atributo de acoplamiento

Luego de haber aplicado esta métrica, se obtuvo como resultado que las clases analizadas en su mayoría tienen un bajo y medio acoplamiento, por lo que se demuestra el mínimo acoplamiento entre las clases.

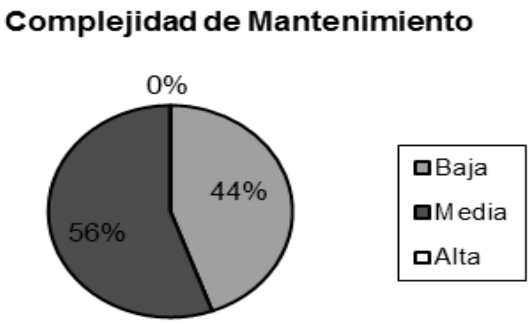

Figura 20 Resultados de la métrica RC para el atributo de complejidad de mantenimiento

Al realizar esta métrica se obtiene como resultado que las clases tienen un nivel medio de complejidad de mantenimiento debido a que hay un mayor porcentaje para dicha categoría.

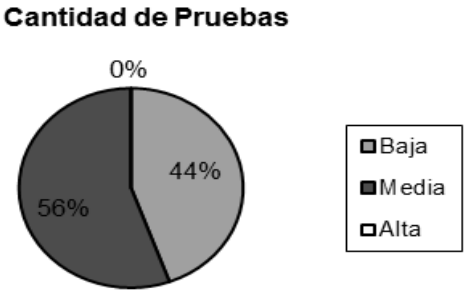

Figura 21 Resultados de la métrica RC para el atributo de cantidad de pruebas

Después de realizar esta prueba se obtiene como resultado que las clases necesitan un nivel medio de cantidad de pruebas de unidad para probar en cada una de ellas.

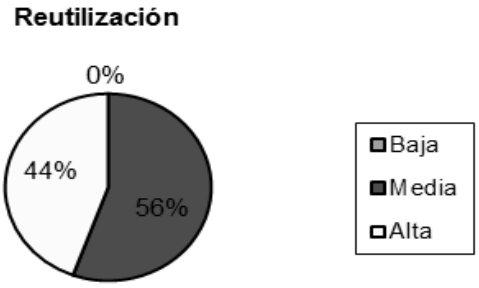

Figura 22 Resultados de la métrica RC para el atributo de reutilización

Esta métrica muestra que hay un nivel medio en el grado de reutilización de las clases.

Niveles de árbol de herencia (PH)

La métrica de árbol de herencia de una clase A es su profundidad en el árbol de herencia. Si A se encuentra en situación de herencia múltiple la longitud máxima hasta la raíz será la profundidad de su herencia. (26)

A continuación se muestran los resultados obtenidos tras hallar los niveles de árbol de herencia a las clases de dominio de los módulos Vínculos y Trabajo que presentan herencia.

No	Clase	Clase padre	Niveles del árbol de herencia
1	Vinculo	Persona	1
2	VinculoInterno	Vinculo	2
3	Extranjero	Vinculo	2
4	Nacional	Vinculo	2
5	DesincorpIntegrante	IncorpIntegrante	1

Tabla 14 Resultados obtenidos utilizando la métrica PH

A continuación se muestra la representación gráfica de los resultados obtenidos de acuerdo a la métrica PH.

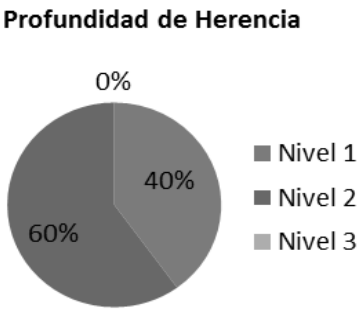

Figura 23 Resultados de la métrica de PH

El resultado de esta métrica muestra que el máximo nivel de herencia presente en el diseño es 3. Por lo que se concluye que el diseño evita la herencia de muchos métodos por parte de las clases inferiores, y se disminuye su complejidad.

Número de hijos

La métrica de número de hijos de una clase es el número de subclases inmediatamente subordinadas a una clase en la jerarquía. (26)

En las siguientes tablas se muestran los umbrales para la medición a utilizar en la métrica número de hijos.

Atributo	Categoría	Criterio
Reutilización	Bajo	N <= Prom
	Medio	Prom > N <= 2 * Prom
	Alto	N > 2 * Prom
Abstracción	Indefinida	N > 5
	Afectada	2 > N <= 5
	Definida	N <= 2
Cohesión	Baja	N > 5
	Media	2 > N <= 5
	Alta	N <= 2
Cantidad de pruebas	Baja	N <= 2
	Media	2 > N <= 5
	Alta	N > 5

Tabla 15 Criterio de evaluación

A continuación se muestran los resultados obtenidos tras hallar el número de hijos de las clases de dominio de los módulos Vínculos y Trabajo que tienen herencia.

No	Clase	Clase padre	Número de descendientes	Reutilización	Abstracción de la clase base	Cohesión de la JC
1	Vinculo	Persona	1	Baja	Definida	Alta
2	VinculoInterno	Vinculo	2	Media	Definida	Alta
3	Extranjero	Vinculo	2	Media	Definida	Alta
4	Nacional	Vinculo	2	Media	Definida	Alta

5	DesincorpIntegrante	IncorpIntegrante	1	Baja	Definida	Alta

Tabla 16 Resultados obtenidos utilizando la métrica números de hijos

A continuación se muestra la representación gráfica de los resultados obtenidos de acuerdo a la métrica número de hijos para un promedio de 1,6.

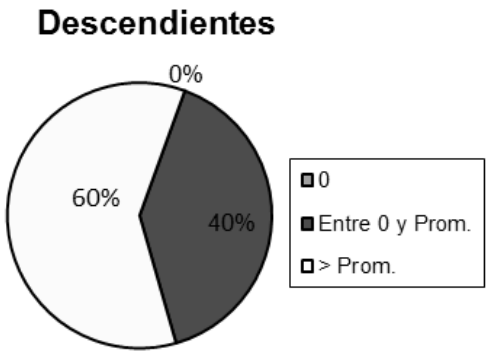

Figura 24 Resultados de la métrica número de hijos

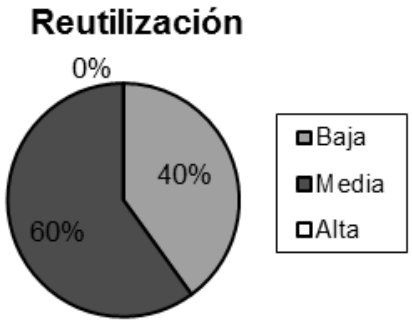

Figura 25 Resultados de la métrica RC para el atributo de reutilización

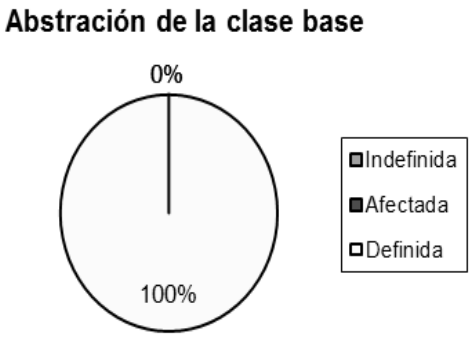

Figura 26 Resultados de la métrica RC para el atributo de abstracción de la clase base

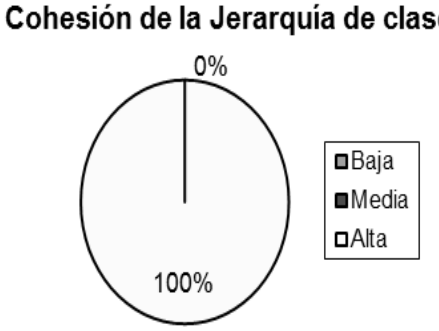

Figura 27 Resultados de la métrica RC para el atributo de cohesión de la jerarquía de clases

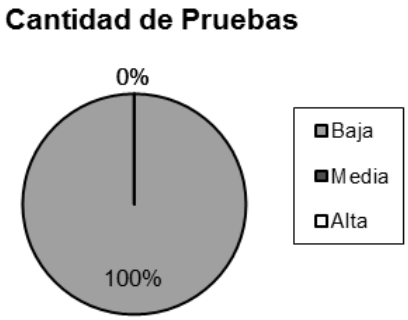

Figura 28 Resultados de la métrica RC para el atributo de cantidad de pruebas

Los resultados obtenidos con la aplicación de esta métrica muestran que el diseño realizado presenta abstracción entre las clases, la cohesión es alta, la cantidad de pruebas presenta un nivel bajo y la reutilización tiene un nivel medio. De manera general se puede afirmar que los resultados son satisfactorios.

2.15. Conclusiones parciales

Con el desarrollo de este capítulo se obtuvo el diseño del sistema, teniendo en cuenta los objetivos trazados en el mismo, tales como: el análisis de la arquitectura a utilizar, la definición de los patrones utilizados para realizar los diagramas de clases del diseño, conjuntamente con los diagramas de colaboración correspondientes a cada uno de los escenarios, así como la definición del diseño de la base de datos correspondiente a los módulos Vínculos y Trabajo. Con la aplicación de las métricas para la validación del diseño se determinó que el modelo de diseño que se propone puede considerarse aceptable de acuerdo a los resultados satisfactorios obtenidos.

Como resultado del diseño del sistema se garantizan las bases para la posterior implementación de los módulos Vínculos y Trabajo.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

En el presente capítulo, teniendo en cuenta el modelo de diseño obtenido en el capítulo anterior, se procederá a documentar todo el flujo de trabajo de implementación correspondiente a los módulos Vínculos y Trabajo. Como parte de dicho flujo se mostrarán los diagramas de despliegue y de componentes realizados para organizar las integraciones del sistema. Además, se definirá la estrategia de prueba a seguir para evaluar la calidad del producto que se propone.

3.2. Fase de Implementación

Esta fase tiene como objetivo la construcción de los componentes del sistema, y a partir de los resultados del diseño se inicia la conformación del sistema en términos de componentes. Los diagramas de despliegue y componentes en conjunto constituyen lo que se conoce como modelo de implementación, ya que describen los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue. Este define cómo se organizan 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. (27) En sentido general se puede afirmar que el propósito fundamental de la implementación es desarrollar la arquitectura y el sistema como un todo.

Como parte de la implementación de los módulos Vínculos y Trabajo, se realizan los diagramas de despliegue y componentes correspondientes a la solución que se propone.

3.3. Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse. (28)

A continuación se muestra el diagrama de despliegue referente a los módulos Vínculos y Trabajo, teniendo en cuenta cada uno de los componentes que intervienen en los distintos nodos físicos.

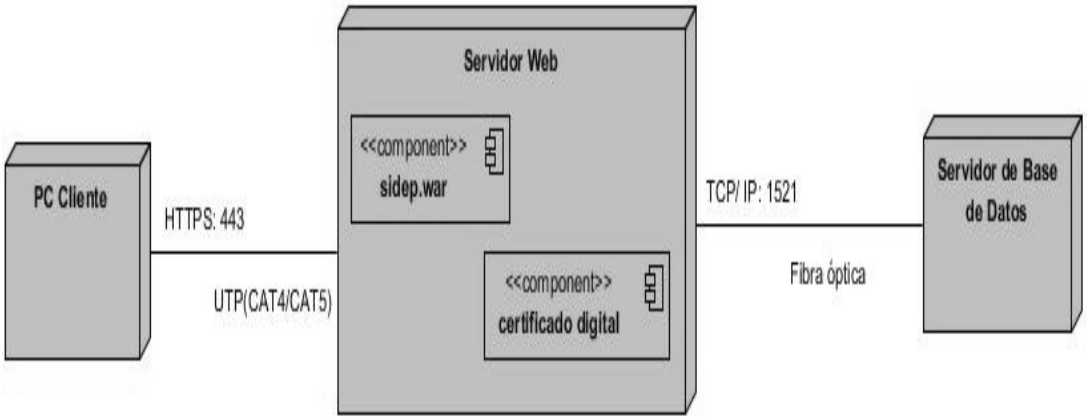

Figura 29 Modelo de despliegue.

Para el correcto uso de la información referente a los módulos Vínculos y Trabajo integrados con el SIDEP se necesita una estructura tecnológica consistente. La figura 29 muestra el modelo de despliegue generado estructurado por un conjunto de pc clientes ligeros que interactúan mediante el navegador Mozilla Firefox en su versión 3.6 o superior con el servidor web utilizando el protocolo HTTPS por el puerto 443. El servidor web que se utiliza es Apache Tomcat en su versión 6.0.*. En este se encuentra desplegado el ejecutable sidep.war; además dicho servidor usa el certificado digital para garantizar la veracidad del sistema e interactúa con el servidor de base de datos Oracle 11G R2 mediante el protocolo TCP/IP por el puerto 1521.

3.4. Diagrama de componentes

Un diagrama de componentes “muestra un conjunto de componentes y sus relaciones; los diagramas de componentes muestran los componentes de un sistema desde un punto de vista estático”. (20)

Una de las facilidades que brindan los diagramas de componentes es visualizan cómo los componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

A continuación se detallan la relación de componentes pertenecientes a los módulos Vínculos y Trabajo.

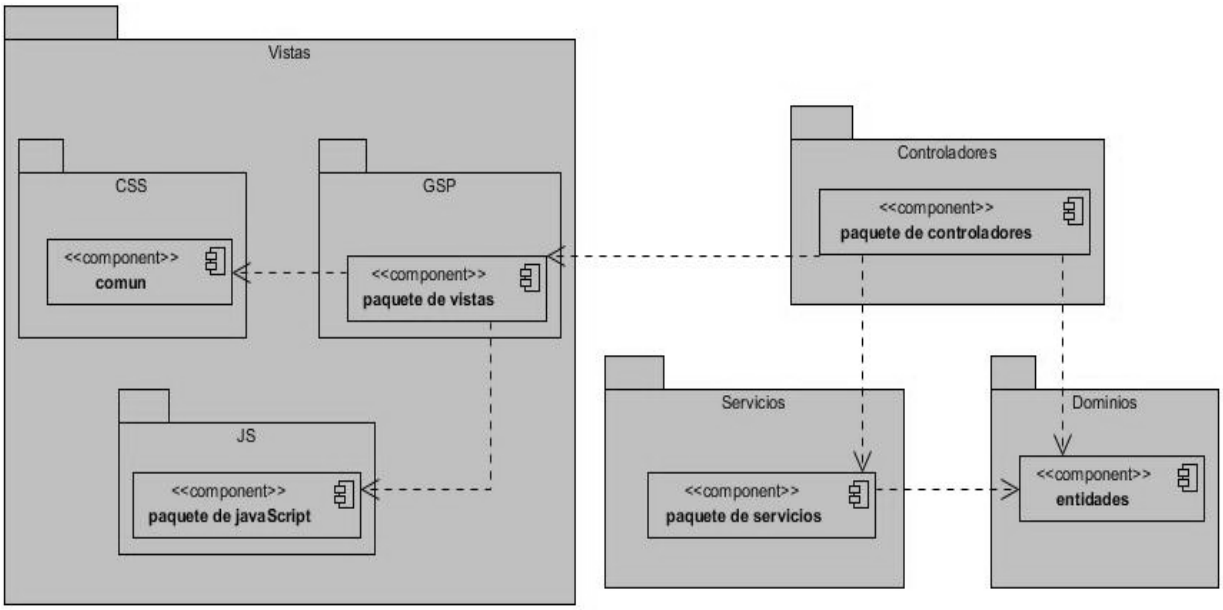

Figura 30 Componentes que interactúan en los módulos Vínculos y Trabajo.

En la figura 30 se representa la distribución de componentes utilizados así como sus relaciones, definiéndose la misma según la arquitectura propuesta para el desarrollo de los módulos.

A continuación se representan las dependencias que tienen los módulos Vínculos y Trabajo con otros componentes.

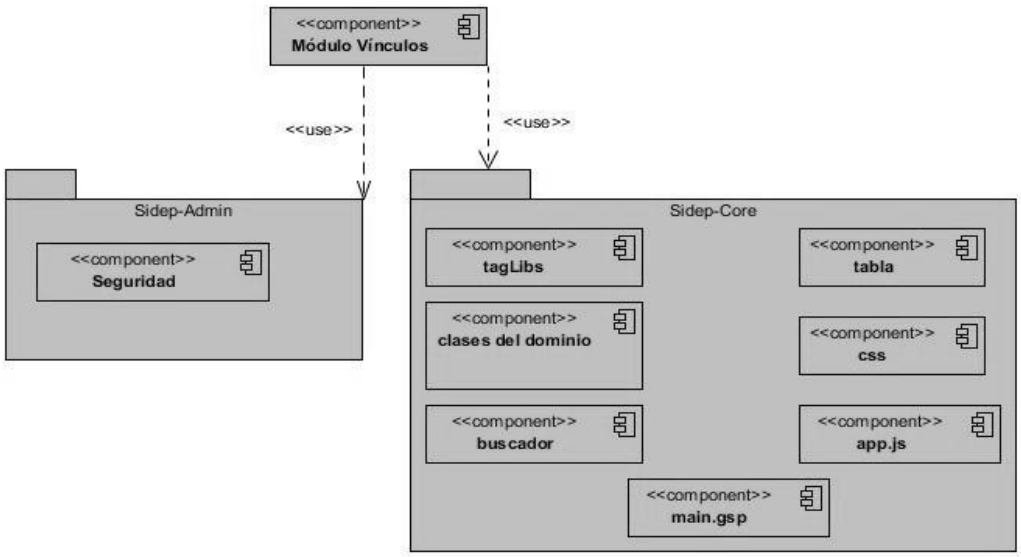

Figura 31 Diagrama de componentes del módulo Vínculos.

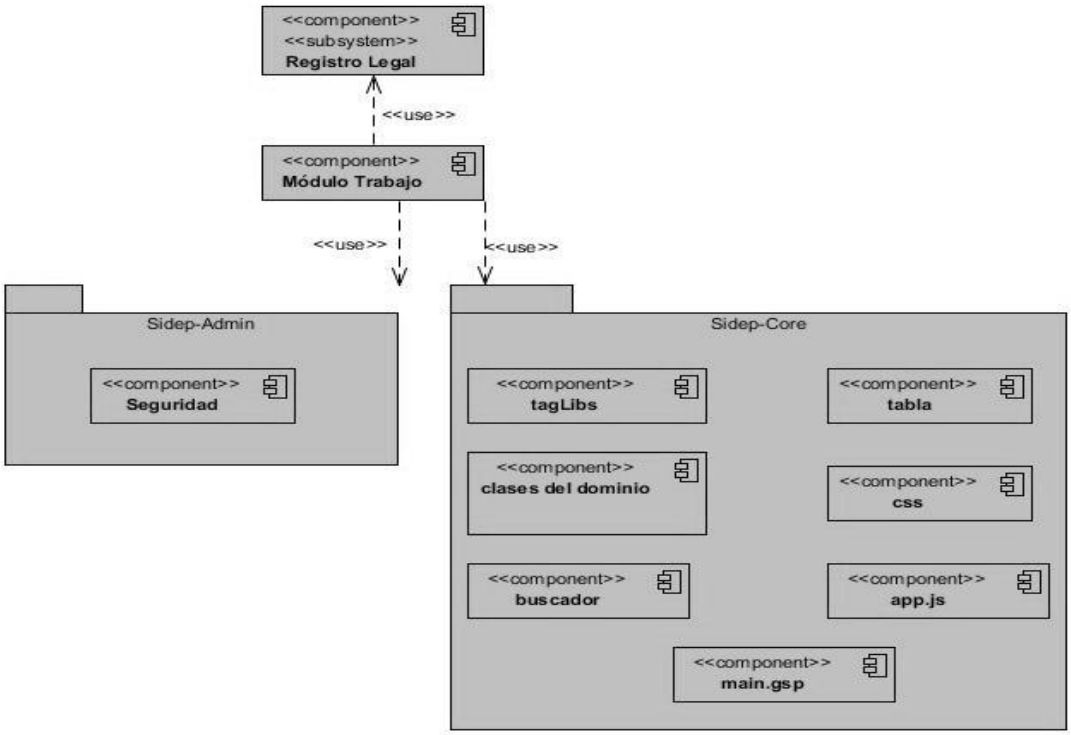

Figura 32 Diagrama de componente Trabajo.

La integración del módulo Vínculo con los subsistemas Sidep-Admin y Sidep-Core facilita el cumplimiento en gran medida de los procesos a desarrollar pues el admin se utiliza para el trabajo referente a la seguridad y el core para interactuar con clases comunes y de uso necesario. En el caso del módulo Trabajo también se sirve de las prestaciones que le brindan los subsistemas Sidep-Admin y el Sidep-Core, interactuando a su vez con el subsistema Registro Legal que cuenta con el módulo Situación legal para obtener información de la responsabilidad civil que puede llegar a tener un interno.

En los anexos se encuentran otros diagramas de componentes pertenecientes a los CU “Registrar vínculo externo” y “Registrar contrato de trabajo” los cuales forman parte de los módulos Vínculos y Trabajo. (Anexo 5)

3.5. Seguridad

La seguridad es un aspecto importante para proteger la integridad y privacidad de los datos y recursos en una aplicación web, por tal motivo es necesario designar estrategias que guíen su correcto uso. En los módulos Vínculos y Trabajo para proteger la aplicación de una serie de amenazas se utilizan mecanismos de autenticación y autorización mediante la integración con el subsistema Sidep-Admin el cual utiliza Spring Security.

Spring Security es un framework altamente adaptable y potente para la autentificación y el control de acceso. (29)

El proceso de autenticación se valida asignándole a cada usuario un rol determinado y una contraseña que le va a permitir tener acceso al sistema, en dependencia del rol del usuario estará dada la autorización de acceso a distintas funcionalidades dentro de la aplicación mediante notaciones de seguridad. En el menú de la aplicación usando el tag (etiqueta) hasAccess se le habilita al usuario registrado las funcionalidades a las cuales él tiene acceso, como se muestra en la figura siguiente.

```
<sa:hasAccess areas="vinculos">
    <div id="vinculos" class="micollapsed">
        <span>V&iacute;nculos</span>
        <sa:hasAccess areas="crud-d-VincInt,eliminarVincInt">
            <a href="${createLink(controller: 'tratamientoEducativoInicio', action: 'moduleVinculoInternoRedirectVin')}">Vínculos Int
        </sa:hasAccess>
        <sa:hasAccess areas="crud-d-VincExt,eliminarVincExt">
            <a href="${createLink(controller: 'tratamientoEducativoInicio', action: 'moduleVinculoExternoRedirectVin')}">Vínculos Ext
        </sa:hasAccess>
    </div>

</sa:hasAccess>
```

Figura 33 Seguridad en el menú de la aplicación.

Debido a que los controladores pueden tener definidas varias funcionalidades y que un determinado usuario puede tener o no autorización para acceder a todas, se crea una sentencia de seguridad *@Security("clave")* en el controlador como se plantea en la figura 34 para acceso pleno, y en caso de que el usuario no deba tener acceso a todas las funcionalidades se le incorpora a cada acción del controlador una clave dentro de la sentencia de seguridad para lograr limitar el acceso , como se muestra en la figura 35.

```
@Security("crud-d-VincExt")
class ListarVinculoExternoController {

    def listarVinculosExternosService

    def url = "/controlvinculos/listarinterno"

    def index = {
        render(view: url)
    }
```

Figura 34 Seguridad en el controlador.

```
@Security("eliminarVincExt")
  def eliminarVinculo = {
      boolean eliminoCorrectamente = false
      InternoVinculo iv = InternoVinculo.findById(params.idVinculo.toLong())
      // Verificando que exista en la base de datos ese interno vinculo
      if (iv) {
          eliminoCorrectamente = listarVinculosExternosService.eliminarVinculo(iv)
      }
      render([eliminoCorrectamente: eliminoCorrectamente] as JSON)
  }

}
```

Figura 35 Seguridad en una funcionalidad del controlador.

La clave que se escribe dentro de la sentencia de seguridad fue establecida por el equipo de seguridad del proyecto, la cual va a ser diferente para cada caso de uso del módulo y para cada acción que se realice, como se muestra en la figura siguiente.

Nombre	Clave	Tipo
Registrar, actualizar y consultar vínculo externo	crud-d-VincExt	
Registrar, actualizar y consultar vínculo interno	crud-d-VincInt	
Eliminar vínculo externo	eliminarVincExt	
Eliminar vínculo interno	eliminarVincInt	

Figura 36 Seguridad a nivel de funcionalidades del módulo Vínculos.

Nombre	Clave	Tipo
Administrar matrícula	adminMatricula	
Gestionar unidad productiva	gestionarUProductiva	
Consultar reporte de desincorporados	consultarDesincorp	
Gestionar brigada	gestionarBrigada	
Registrar asistencia al trabajo	reistrarAsisTrabajo	
Gestionar contrato de trabajo	gestionarContTrabajo	

Figura 37 Seguridad a nivel de funcionalidades del módulo Trabajo.

3.6. Pruebas

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un sistema informático. (30)

Con las pruebas se puede además observar hasta qué punto el software parece funcionar en concordancia con los requisitos funcionales descritos para el sistema; aunque no pueden asegurar la ausencia de defectos, sólo puede mostrar que existen.

3.6.1. Estrategia de prueba

La estrategia de prueba tiene como finalidad fundamental describir el enfoque y los objetivos generales de las actividades de prueba. Con el objetivo de guiar el proceso de pruebas y garantizar al máximo la eliminación de las no conformidades detectadas por el probador se hace necesario definir la estrategia a seguir, este proceso consiste en la integración de diferentes factores y pasos que dan como resultado una correcta construcción del software.

A continuación se especifica y explica la estrategia seleccionada para las pruebas de los módulos Vínculos y Trabajo:

1. Método de prueba: método de caja negra.
2. Tipo de prueba: funcionalidad.
3. Nivel de prueba: prueba de desarrollador.
4. Tipo de técnica de prueba: casos de prueba.

Método de prueba

Prueba de Caja Negra: se realizan pruebas que verifican que cada función es operativa y que la entrada y salida se producen de forma correcta. Estas pruebas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. (30)

Las pruebas de caja negra se centran principalmente en los requisitos funcionales del software y pretenden encontrar los siguientes tipos de errores:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en las salidas.
4. Errores en el acceso a los datos.
5. Errores de rendimiento.
6. Errores de inicialización y de terminación.

A su vez pretenden demostrar que:

1. Las funciones del software son operativas.
2. La entrada se acepta de forma correcta.
3. Se produce una salida correcta.
4. La integridad de la información externa se mantiene.

Tipo de prueba

Funcionalidad: estas pruebas fijan su atención en comprobar la funcionalidad del sistema validando las funciones, métodos, servicios y CU. (30)

La prueba de funcionalidad tiene como metas a seguir un conjunto de requisitos, los cuales tienen como objetivo validar que la aplicación:

5. Cumpla con los requisitos funcionales especificados en el diseño de la solución por medio de CU.
6. Cumpla con los requisitos no funcionales especificados en el diseño de la solución.
7. Cumpla con las restricciones de entrada y salida de la información especificada en el diccionario de datos.

8. Cumpla íntegramente con la estructura referencial especificada en el mapa de navegación.

Nivel de prueba

Prueba de desarrollador: indica los aspectos de diseño e implementación de la prueba más adecuada que debe llevar a cabo el equipo de desarrolladores. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas del desarrollador que la 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. (31)

Técnica de prueba

Casos de Prueba: está basada en el método de particiones equivalentes de RUP creando su diseño a partir de los CU y se verifican todas las funcionalidades probándose además todas las combinaciones posibles de clases validas e inválidas. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (31)

3.7. Conclusiones parciales

Con el desarrollo de este capítulo se detallaron los principales aspectos que intervienen en la implementación de los módulos Vínculos y Trabajo. Se generaron un conjunto de artefactos que contribuyen al mejor entendimiento del flujo de procesos que se desarrollan en dichos módulos, tales como los diagramas de componentes y el modelo de despliegue. También se describió cómo se implementa la seguridad en los módulos para garantizar la integridad de los datos. Además, a partir de la estrategia de prueba puesta en práctica, luego de implementada la solución, se garantizó el cumplimiento de las especificaciones del sistema y lograr que la solución propuesta tenga un mayor acierto.

.

CONCLUSIONES GENERALES

Teniendo en cuenta el cumplimiento de los objetivos propuestos, se arriba a las siguientes conclusiones:

1. Se detallaron los elementos necesarios para la búsqueda de la respuesta al problema propuesto, realizándose un estudio y análisis de la metodología, en este caso RUP, las herramientas, entre las que se encuentran Visual Paradigm para UML, Embarcadero ER/Studio, NetBeans y Apache Tomcat, las tecnologías propuestas, Grails, Dojo Toolkit y Oracle Database 11g R2, y como lenguajes HTML, JavaScript, Groovy, UML y Java, lo que permitió demostrar la elección de estos para el desarrollo de los módulos Vínculos y Trabajo.
2. Se generaron los artefactos necesarios para documentar el diseño del sistema, diagramas de clases, diagramas de colaboración y modelo de datos, permitiendo materializar con precisión los requisitos expuestos por el cliente mediante la implementación de los módulos Vínculos y Trabajo.
3. Se implementaron todas las funcionalidades definidas para los módulos Vínculos y Trabajo, lo que garantizó la satisfacción de las especificaciones dadas por el cliente para el SIDEP.
4. Se realizaron las pruebas de caja negra, donde las no conformidades encontradas fueron corregidas, demostrando que cada una de las funcionalidades responde apropiadamente a los requisitos funcionales definidos, incluyendo la navegación, la entrada de datos, el procesamiento y la obtención de resultados.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cesio, Sonia.** EnigmaPsi. [En línea] 2002-2008. [Citado el: 15 de Febrero de 2012.] http://www.enigmapsi.com.ar/configvinc.html.

2. **Zequeira Peña, Alfonso y Céspedes Quesada, Aimeé.** *Vocabulario Jurídico Penitenciario.* La Habana : MININT, 2005.

3. *Código Penal Cubano. Ley Nro. 62.* Cuba : s.n.

4. **Morciego Pérez, Irina.** *Diseño e Implementación del módulo Instrucción Escolar del SIGDATI.* La Habana : s.n., 2011.

5. **Dirección General del Sistema Penitenciario de Panamá.** Ministerio de Gobierno y Justicia de Panamá. [En línea] [Citado el: 15 de Febrero de 2012.] http://www.sistemapenitenciario.gob.pa/submenu.php?cod=25&id=1.

6. *Informe de factibilidad e ideas preliminares sobre la informatización del sistema penitenciario cubano.* La Habana : MININT.

7. **Hernández Estrada, Yelena.** *Análisis del Módulo Proceso Confiscatorio de Bienes del proyecto Sistema de Gestión .* La Habana : s.n., 2009.

8. **Fernández y Fernández, Carlos Alberto.** El Proceso Unificado de Rational para el Desarrollo de Software. [En línea] 20 de Octubre de 2000. [Citado el: 24 de Marzo de 2012.] http://www.utm.mx/~caff/doc/El%20Proceso%20Unificado%20Rational.pdf.

9. Rational Unified Process (RUP). [En línea] 2010. [Citado el: 25 de Marzo de 2012.] http://www.lsi.us.es/docencia/get.php?.

10. Free Download Manager. [En línea] 2004. [Citado el: 27 de Marzo de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p.

11. Revista digital SGuía. [En línea] 2009. [Citado el: 27 de Marzo de 2012.] http://sg.com.mx/guia/node/1454.

12. Oracle Corporation. Comunidad de código abierto del NetBeans. [En línea] 2012. [Citado el: 29 de Marzo de 2012.] http://netbeans.org/index_es.html.

13. **The Apache Software Foundation.** Apache Tomcat. [En línea] [Citado el: 2 de Abril de 2012.] http://www.tomcat.apache.org.

14. **Brito, Nacho.** *Manual de desarrollo web con Grails. JavaEE, como siempre debió haber sido.* s.l. : ISBN: 978-84-613-2651, 2009.

15. EcuRed. Dojo Toolkit. [En línea] 2011. [Citado el: 2 de Abril de 2012.] http://www.ecured.cu/index.php/Dojo_toolkit.

16. La Flecha. [En línea] Noviembre de 2003. [Citado el: 5 de Abril de 2012.] http://www.laflecha.net/canales/empresas/noticias/oracle-11g-la-nueva-base-de-datos-de-oracle.

17. Libros Web. Libro Introducción a Javascrpit. [En línea] [Citado el: 6 de Abril de 2012.] http://www.librosweb.es/javascript/capitulo1.html.

18. **García, Fran.** Informática, Internet y otras cosas del querer. [En línea] 10 de Junio de 2009. [Citado el: 8 de Abril de 2012.] http://www.frangarcia.net/txp/articles/319/caracteristicas-de-groovy.

19. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999.

20. **Jacobson, Ivor y Brooch, Gravy y Rum Baugh, James.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2000.

21. EcuRed. Lenguaje de programación Java. [En línea] 2011. [Citado el: 8 de Abril de 2012.] http://www.ecured.cu/index.php/Lenguaje_de_programaci%C3%B3n_Java.

22. EcuRed. Flujo de trabajo Requerimientos. [En línea] 14 de Diciembre de 2010. [Citado el: 10 de Abril de 2012.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_Requerimiento.

23. **Tedeschi, Nicolás.** MSDN. [En línea] 2012. [Citado el: 12 de Abril de 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

24. **Calero Muñoz, Coral.** Sistema de Información del Ministerio Fiscal. [En línea] [Citado el: 14 de Marzo de 2012.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Metricas4.pdf.

25. Ecured. Métrica de diseño. [En línea] [Citado el: 9 de Junio de 2012.] http://www.google.com.cu/url?sa=t&rct=j&q=Tama%F1o+operacional+de+clase&source=web&cd=1&ved=0CFYQFjAA&url=http%3A%2F%2Fwww.ecured.cu%2Findex.php%3Ftitle%3DEspecial%3APdfprint%26page%3DM%25C3%25A9trica_de_dise%25C3%25B1o&ei=TUfhT5SsJpOp0AHKpPWfAw&usg=AFQjC.

26. **Olmedilla Arregui, Juan José.** *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* 2005.

27. **Duran Rivas, Yaksel y Sanz Mantilla, Daylin.** *Diseño e implementación del módulo Importación de los Depósitos Temporales.* La Habana : s.n., 2011.

28. [En línea] [Citado el: 3 de Mayo de 2012.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

29. Spring-Security. [En línea] [Citado el: 13 de Junio de 2012.] http://static.springsource.org/spring-security/site.

30. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque Práctico 5ta edición.* s.l. : McGraw-Hill, 2005.

31. **Machado Peña., Yadira.** *Estrategia de Pruebas de función.*

GLOSARIO DE TÉRMINOS

API: Especificación de una librería o utilidad que documenta su interfaz y permite su uso sin conocimiento de su interior.

Arquitectura: responsable de definir la línea base de la arquitectura, define las pautas para el diseño y la codificación. Establece el marco de desarrollo que va a soportar el proyecto.

Caso de prueba: conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. Siempre es ejecutada como una unidad, desde el comienzo hasta el final.

CRUD: es un acrónimo para las operaciones básicas de Creación (*Create*), Lectura (*Read*), Actualización (*Update*) y Borrado (*Delete*).

HTTPS: *Hyper Text Transfer Protocol Secure* (Protocolo Seguro de Transferencia de Hipertexto). Combinación del Protocolo de Transferencia de Hipertexto (*Hyper Text Transfer Protocol,* HTTP por sus siglas en inglés) con el protocolo SSL / TLS para proporcionar comunicaciones encriptados y la identificación segura de un servidor web en la red.

Interno: se le denomina así a los sancionados, asegurados y acusados del sistema penitenciario.

CSS: Cascading Style Sheet (Hojas de estilo en cascada). Lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML.

MVC: es un patrón arquitectónico que permite organizar el código de una aplicación, facilitando modificar una de las capas sin necesidad de modificar las restante, así como la posterior reutilización del código.

SSL: *Secure Socket Layer* (Capa de Enchufle Seguro). Servidor de correo que proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico.

TCP/IP: *Transmission Control Protocol/Internet Protocol* (Protocolo de Control de Transmisión/ Protocolo de internet). Conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

TLS: *Transport Layer Security* (Seguridad de la Capa de Transporte). Es un protocolo mediante el cual se establece una conexión segura por medio de un canal cifrado entre el cliente y servidor.