Universidad de las Ciencias Informáticas

Facultad 2

Título: Informatización del Expediente Legal del Interno en el Sistema Penitenciario Cubano

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Leandro Roura Sixto

Juan Carlos Lugones Martinez

Tutor: Ing. Yanay Viera Lorenzo

La Habana, 2012

"Año 54 de la Revolución"

"Todo tiene que hacerse lo más simple posible,

pero ni un ápice más simple."

Albert Einstein

Declaración de Autoría

Declaramos ser los únicos autores de este trabajo y autorizamos al centro ISEC de la Universidad de las Ciencias Informáticas; así como a dicha universidad para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ___ días del mes de Junio del año 2012.

Leandro Roura Sixto
Nombre del autor

Juan Carlos Lugones Martinez
Nombre del autor

Ing.Yanay Viera Lorenzo
Nombre del tutor

Agradecimientos

Agradecemos a Yanay, Yanet, y Anabel, quienes nos escogieron para formar parte del equipo de desarrollo, en el cual, hemos crecido en la Universidad de las Ciencias Informáticas.

A mi compañero de tesis.

A Susana, quien nos obligó a estudiar.

A Maikel David y a Reinier por ayudar en la implementación del módulo.

A Mindpro, por ser el proyecto exploratorio que nos inició en el mundo de la programación web.

A la Revolución.

Dedicatoria

Dedicamos nuestro esfuerzo especialmente a nuestros padres, por la educación que nos han dado y por hacernos posible que estuviésemos hoy escribiendo estas líneas (Juan Carlos Lugones Torres y Marcia Martinez Gonzalez; José Felipe Roura Benítez y Josefa Sixto Oliva):

Juanky: A mi abuela Delia, a mi hermano Yoanky, a mis tíos Gustavo y Margarita, a mis angelitas Geimy y Geydis.

L3@Corporation: A mi hermano Lester, al cual nombré momentos antes de su nacimiento, a mi abuela, a mis tíos: Sonia, María y Dianco; Koky, Felito, Befo, Butico, Archie, Alina. A mis primos y familiares. A Yosdel y a Eddy.

Dedicamos nuestro trabajo a todos nuestros compañeros, especialmente a los que formamos parte del grupo 4107 y del proyecto.

A la tía Beba, a la psico Dagmara, a mis tías del comedor.

Resumen

Actualmente el Sistema Penitenciario Cubano cuenta con una solución informática que permite la creación de expedientes para controlar los procesos legales de los internos, el Sistema Automatizado de Control del Recluso. Este sistema no abarca todas las funcionalidades necesarias para gestionar las conversiones en los expedientes legales y su manejo tiende a ser complejo para los funcionarios del área responsable. A partir de la situación actual en el Sistema Penitenciario Cubano se decidió el desarrollo de un nuevo sistema, el Sistema Informativo de la Dirección de Establecimientos Penitenciarios. Entre los módulos definidos para este sistema se encuentra el módulo Expediente Legal que tiene por objetivo apoyar el control de los procesos y la generación automática de los trámites legales de los internos con los diferentes órganos de instrucción y de justicia penal.

El presente trabajo radicó en diseñar e implementar el módulo Expediente Legal del subsistema Registro Legal del Sistema Informativo de la Dirección de Establecimientos Penitenciarios a partir de los requisitos de software definidos para el módulo y haciendo uso de la arquitectura establecida para el sistema. Durante la investigación se realizó análisis de documentación para el estudio de soluciones nacionales e internacionales que reflejó la necesidad de implementar una nueva solución y para el estudio de las tecnologías, metodología, herramientas y lenguajes en los que se basa el desarrollo del módulo. Se utilizó, además, la modelación para la obtención de artefactos del diseño e implementación. Para comprobar el cumplimiento de los requisitos implementados se aplicaron pruebas de calidad mediante las cuales se eliminó la aparición de no conformidades. De esta forma se obtuvo el módulo Expediente Legal que permite gestionar las conversiones de la situación legal.

Palabras claves: estatus legal, expediente legal, ingreso, interno, proceso, reingreso, situación legal, trámite legal pendiente.

Índice

Introducción

En el año 1989 comienza en Cuba la informatización de los centros penitenciarios, con la recogida de los datos principales del recluso y ciertos aspectos de Control Penal; años más tarde se crea el Sistema Automatizado para el Control del Recluso (SACORE). Dicho sistema culminó su desarrollo a finales del 2002, poniéndose en marcha a principios del 2003. SACORE cuenta en la actualidad con tres módulos principales: Control Penal, Reeducación Penal y Orden Interior.

En los 9 años de explotación del software, aún cuenta con requisitos incompletos o pendientes, por no contar la institución con el tiempo para su análisis, diseño e implementación. (1) Uno de los requisitos incompletos es la informatización del expediente legal de los internos. Actualmente, el expediente legal es almacenado en formato duro posibilitando la pérdida o deterioro en algunas ocasiones de los documentos legales que lo componen.

En el 2009 comienza el desarrollo de un nuevo sistema: Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP) [1] con el objetivo de "informatizar el Sistema Penitenciario Nacional (…) para automatizar los procesos de trabajo que se ejecutan para el control, tratamiento y atención a los acusados, asegurados y sancionados en los centros penitenciarios…" (2) El SIDEP es una aplicación web compuesta por siete subsistemas dentro de los cuales se encuentra Registro Legal y éste a su vez se compone de siete módulos. El módulo Expediente Legal permite crear un nuevo expediente o actualizar un expediente existente a partir de un ingreso de un interno al centro penitenciario y almacena los datos que controlan la legalidad de dicho interno.

Entre las funcionalidades que implementa el SACORE, se encuentra el registro de la situación legal de los internos con los datos de los documentos, pero no recoge los tipos de documentos legales del interno para el ingreso o reingreso en el sistema o en la actualización de un proceso. Esto imposibilita que se pueda saber a qué documento pertenecen los datos del expediente legal ni qué órganos de instrucción o de justicia penal los remitió.

[1] Actualmente SIDEP, anteriormente Sistema de Gestión del Interno.

Cada interno trae consigo en el ingreso, una combinación determinada de documentos legales expedidos por los diferentes órganos. Esta combinación está determinada por el *motivo de ingreso* del interno al ingresar a prisión, le corresponde un estatus legal y con ella se registra un nuevo proceso para el interno. Estos procesos se actualizan con más documentos que pueden modificar el estatus legal. Los cambios de estatus son asignados por un funcionario de Registro Legal apelando a sus conocimientos jurídicos.

Actualmente, el SACORE registra un nuevo expediente cada vez que le son emitidos los documentos por un nuevo proceso a un interno, ya sea mediante un reingreso o durante su permanencia en el centro penitenciario. Un expediente legal en el SACORE constituye un único proceso del interno. La creación de un nuevo expediente causa que existan internos con más de un expediente legal en el período de permanencia en el centro. Para continuar el cumplimiento de la sanción o medida impuesta al interno, es necesario realizar la unificación de expedientes. Este proceso se realiza manualmente donde el funcionario debe determinar qué expediente legal debe ser por el que se mantenga cumpliendo el interno. Los demás expedientes generan trámites legales pendientes con los órganos de instrucción o justicia penal que emiten los distintos documentos.

Los trámites legales pendientes deben ser controlados desde su generación hasta su resolución, pero SACORE no está diseñado para responder a esta necesidad sino que se controlan manualmente a partir de la adición o actualización de los expedientes legales y de las conciliaciones con los órganos correspondientes.

A pesar de que los centros penitenciarios cuentan actualmente con un sistema informático, las conversiones de la situación legal en el expediente legal de un interno son un proceso complejo. Esta situación problemática lleva a enunciar el siguiente **problema**: ¿Cómo controlar la situación legal de los internos en el Sistema Penitenciario Cubano a partir de la gestión de las conversiones en el expediente legal?

El **objeto de estudio** se centra en el proceso de gestión del expediente legal en los sistemas penitenciarios, siendo el **objetivo general**: Desarrollar el módulo Expediente Legal del subsistema Registro Legal del SIDEP.

El **campo de acción** de la investigación se enmarca en la situación legal de los internos y sus trámites legales pendientes en el Sistema Penitenciario Cubano.

La **idea a defender** durante la investigación es: el desarrollo del módulo Expediente Legal del subsistema Registro Legal del SIDEP permitirá la gestión de las conversiones de la Situación Legal de los internos.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación.
2. Diseñar el módulo Expediente Legal del subsistema Registro Legal del SIDEP.
3. Implementar el módulo Expediente Legal del subsistema Registro Legal del SIDEP.
4. Realizar pruebas de calidad al módulo Expediente Legal del subsistema Registro Legal del SIDEP.

Los objetivos específicos se desglosaron en las siguientes **tareas de la investigación**:

1. Análisis de soluciones nacionales e internacionales enmarcadas en el objeto de estudio.
2. Descripción de las herramientas y tecnologías para el desarrollo módulo Expediente Legal del subsistema Registro Legal del SIDEP.
3. Diseño de los diagramas de clases del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
4. Diseño de los diagramas de interacción del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
5. Diseño de diagramas de estado para las entidades que lo requieren.
6. Diseño de la Base de datos del módulo Expediente Legal del subsistema Registro Legal del SIDEP
7. Elaboración de los diagramas de componentes del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
8. Implementación del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
9. Diseño del diagrama de despliegue del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
10. Diseño de los casos de prueba del módulo Expediente Legal del subsistema Registro Legal del SIDEP.
11. Aplicación de las pruebas módulo Expediente Legal del subsistema Registro Legal del SIDEP.

Durante la investigación se utilizarán los siguientes **métodos científicos**:

Métodos teóricos

- Analítico–sintético: se utiliza para el análisis de las herramientas, tecnologías, lenguajes y la metodología de desarrollo seleccionadas, identificando las características que las distinguen y se refleja la valoración de los autores sobre las mismas.
- Modelación: se utiliza en el diseño del módulo a implementar.

Métodos empíricos

- Análisis documental: se realiza con el fin de realizar un estudio de la bibliografía referente a las herramientas, metodología y tecnologías para dar solución a la problemática.

El presente documento está compuesto por 3 capítulos:

Capítulo 1. Marco Teórico: Incluye un estado del arte de los sistemas a nivel nacional e internacional que dan respuesta a la gestión del expediente legal de los internos en los sistemas penitenciarios. Se describen las tecnologías para desarrollar, así como la metodología de desarrollo de software y las herramientas utilizadas.

Capítulo 2. Diseño del sistema: A partir de un análisis de los casos de uso definidos y la arquitectura del SIDEP se construyen los diagramas de clases del diseño, de interacción y los diagramas de estados para las entidades que lo requieren. Se diseña el modelo de datos y se describen las tablas físicas de la base de datos.

Capítulo 3. Implementación y prueba: Incluye los diagramas de componentes, el diagrama de despliegue y ejemplos de código de las diferentes clases que explican el funcionamiento de los componentes. Se define una estrategia de prueba y se aplica mostrando los resultados obtenidos.

Capítulo 1. Marco teórico

1.1 Introducción

El objetivo de este capítulo es ofrecer el marco teórico en el que se desarrolla la presente investigación, creando un basamento para los próximos capítulos. Para ello, se realiza un estudio de los sistemas informáticos de gestión penitenciaria y cómo manejan el expediente legal de los internos. También se realiza un estudio del ambiente de desarrollo seleccionado y establecido para el SIDEP, especificando las principales características de la metodología, lenguajes, tecnologías y herramientas establecidas que se utilizan en la implementación de la solución.

1.2 Marco conceptual

El marco conceptual de la investigación establece los términos fundamentales para un mejor entendimiento de los conceptos del negocio. (3) A continuación se hace mención a los conceptos identificados y que son necesarios para la comprensión de esta investigación:

Acusado: La persona a quien se le haya decretado la medida cautelar de prisión provisional. (4)

Asegurado: La persona a quien se le impone una medida de seguridad reeducativa de internamiento. (4)

Causa: identificación de un proceso legal en el tribunal. (5)

Conversión: Todo cambio que se produzca durante la permanencia en el centro de los acusados, sancionados y asegurados que modifique la situación legal, tipicidad delictiva, antecedentes penales, progresión o regresión en régimen, rebaja de sanción o pérdida de esta, cambio de delito y radicación de causa. (5)

Denuncia: identificación de un proceso legal en el órgano de instrucción. (5)

Egreso: Se considera egreso cuando el interno, en dependencia de su estatus legal, cumple su sanción o medida de seguridad, se le otorga la libertad anticipada, (5) fallece o se evade.

Expediente de Peligrosidad: identificación de un proceso legal para un asegurado.

Expediente en Fase Preparatoria: identificación de un proceso legal en la fiscalía.

Estatus Legal: Clasificación legal asignada por el Sistema Penitenciario Cubano al que infringe la ley: Acusado Pendiente a Juicio, Acusado Falto de Documentos, Sancionado o Asegurado.

Expediente Legal: El expediente legal está conformado por documentos legales expedidos por las autoridades competentes que amparan su internamiento, en él se recogen los aspectos más sobresalientes de la trayectoria penal y penitenciaria del interno. Contiene los documentos que se remiten con el detenido, así como los que se obtienen y confeccionan a su ingreso, durante su permanencia y egreso, lo cual permite el control del acusado, sancionado o asegurado. (6)

Evasión: Período de tiempo en el que el interno se encuentra evadido del Sistema Penitenciario Cubano.

Ingreso: Se considera ingreso a toda remisión al lugar de internamiento por orden de la autoridad judicial competente de un acusado, sancionado y asegurado que no tenga un expediente abierto en el sistema. (5)

Interno: Denominación genérica que identifica a los acusados, sancionados o asegurados en el proceso de ejecución de la sanción de privación de libertad, medida cautelar de prisión provisional y mediada reeducativa de internamiento respectivamente. (6)

Preventiva: Al período de tiempo en el que el interno estuvo detenido antes de ingresar al Sistema Penitenciario Cubano.

Proceso: Es el respaldo documental y legal de la condena de un interno en el Sistema Penitenciario Cubano.

Quebrantamiento (de la sanción): Tiempo determinado por las evasiones o las ausencias al permiso de un interno.

Reingreso: Se denomina reingreso a toda remisión al lugar de internamiento por orden de la autoridad judicial competente de un acusado, sancionado y asegurado que tenga un expediente abierto en el sistema. (5)

Responsabilidad civil: Es la obligación de restituir lo dañado, reparar el daño moral y adoptar las medidas necesarias para que el inmueble sea desocupado y restituido al organismo al cual un interno provocó algún tipo de daño. (5)

Sanción: Aplicación de algún tipo de pena o castigo a un individuo ante determinado comportamiento considerado inapropiado, peligroso o ilegal al cometer un delito. (5)

Sanción accesoria: Sanción de privación de derechos que comprende la pérdida del derecho al sufragio activo y pasivo, así como del derecho a ocupar cargo de dirección en los órganos correspondientes a la actividad político-administrativa del Estado, en unidades económicas estatales y en organizaciones de masas y sociales. (5)

Sancionado: La persona ejecutoriamente sancionada a privación de libertad o a trabajo correccional con internamiento. (6)

Situación Legal: Condición que define al interno durante su estancia en el Sistema Penitenciario Cubano.

Trámite legal pendiente: “Se considera trámite legal pendiente, aquellas situaciones carentes de solución que presentan los sancionados y asegurados, que precisan de la determinación legal de la autoridad judicial competente.” (6)

Habiendo definido los principales conceptos para un mejor entendimiento del negocio, se procede al estudio de sistemas nacionales e internacionales en sistemas penitenciarios para valorar el empleo de las mismas en la problemática planteada.

1.3 Soluciones informáticas nacionales e internacionales

Actualmente el desarrollo informático ha propiciado que las organizaciones penitenciarias basen su desempeño en el uso de aplicaciones informáticas para la gestión de la información de su población penal. A continuación se explica el sistema SACORE y el módulo Situación Legal del Sistema de Gestión del Interno como soluciones nacionales y los sistemas internacionales Sistema de Gestión Penitenciaria de Venezuela, Sistema de Registro Penitenciario de Perú y OFFENDERTRAK Corrections Management System de la compañía Motorola.

1.3.1 SACORE

El SACORE es el sistema informático utilizado actualmente en el Sistema Penitenciario Cubano. Entre sus principales funcionalidades está el registro de la información referente a la situación legal de los internos. Una vez que un interno ingresa a un centro penitenciario se le crea un expediente legal en el SACORE y el módulo Situación Legal se encarga de recoger el motivo de ingreso y todos los datos fundamentales que emiten los órganos de instrucción o de justicia penal. SACORE no hace distinción entre qué datos deberían recogerse por cada motivo de ingreso, sino que existe una vista principal donde se reflejan los campos que en ocasiones pueden ser datos nulos en el sistema.

La actualización del expediente legal no permite mantener el histórico del flujo de movimientos por órganos de un proceso debido a que los datos se sobrescriben manteniendo solamente los datos actuales.

Si un interno está implicado en otro proceso, SACORE crea un nuevo expediente legal y los funcionarios son los encargados de realizar una unificación de todos los expedientes legales. La unificación de expedientes conlleva a tomar decisiones jurídicas y realizar cálculos manuales de fechas que definen la situación legal de los internos.

SACORE mantiene un registro de las responsabilidades civiles, sanciones accesorias, evasiones, trámites legales pendientes y preventivas identificando el expediente legal implicado. Los trámites legales pendientes deben ser registrados manualmente a pesar de que son generados a partir de conversiones bien identificadas en el Sistema Penitenciario Cubano. SACORE aporta a la solución propuesta en la presente investigación, el mantenimiento de este registro con la identificación de los procesos en los que están implicados estos elementos.

Las funcionalidades que implementa el SACORE no garantizan la completa gestión de las conversiones del expediente legal de los internos; introduce trabajo complejo para los funcionarios cuando un sistema informático debería apoyar la ejecución de los procesos en un negocio y no garantiza el almacenamiento de toda la información legal de un interno.

1.3.2 Módulo Situación Legal del Sistema de Gestión del Interno

El Sistema de Gestión del Interno (SIGI), actualmente SIDEP, estaba compuesto por siete subsistemas, dentro de los cuales se encontraba Registro Legal. Este subsistema contenía tres módulos: Ingreso, Situación Legal y Datos Generales; que constituían los únicos módulos en desarrollo.

SIGI se desarrolló utilizando Grails 1.2.2, pero sin aprovechar la arquitectura que el marco de trabajo brindaba. La poca experiencia de los desarrolladores en ese momento provocó que se incurriera en los siguientes problemas:

- No existían pautas definidas para la interfaz de usuario.
- A pesar de seguir la metodología Proceso Unificado del Software, a la especificación de los casos de uso le seguía la implementación sin tener en cuenta el diseño de la solución.
- La lógica del negocio se implementaba en los controladores existiendo la capa de servicio de Grails que posibilita la implementación de esta lógica.
- No se concebía la reutilización de componentes comunes incurriendo en la duplicación de código para dar solución a un mismo requisito.

A pesar de existir estos problemas, en el 2010 se implementó el módulo Situación Legal del SIGI (1) que incluye “información sobre los procesos judiciales del individuo(a), los trámites legales pendientes a los que puede estar sujeto así como el régimen por el cual transita.” (1). El expediente legal definido para ese entonces abarcaba toda la información del interno en su tránsito por el Sistema Penitenciario Cubano: datos generales, situación legal, movimientos dentro y fuera del centro penitenciario y la vinculación al trabajo, al estudio y a las actividades educativas. Actualmente el expediente legal lo constituyen solamente los datos legales definidos a partir de los documentos procedentes de los órganos de instrucción, fiscalías y tribunales.

El módulo Situación Legal estuvo estructurado por cinco paquetes: Generales, Documentos, Procesos, Régimen y Trámites legales pendientes (1):

- El paquete Generales permite gestionar la clasificación, las categorías, fechas delictivas y el estatus legal de los internos. Las categorías actualmente son gestionadas a partir de un nuevo módulo del SIDEP: Categorías (7). Este paquete, además, no maneja los estados del estatus legal

permitiendo que el usuario actualice el estatus sin registrar la fecha en que cambia. La implementación de otras funcionalidades del sistema requieren conocer qué proceso determinó un cambio de estatus y el diseño del paquete Generales no lo posibilita.

- El paquete Trámites legales pendientes permite generar algunos trámites automáticamente, pero no todos los requeridos. Por ejemplo, no genera los trámites legales pendientes de sanción conjunta dejando la decisión al usuario del sistema. Esto posibilita que un interno pueda estar cumpliendo por dos procesos lo cual constituye una ilegalidad en el Sistema Penitenciario Cubano.
- El paquete Régimen permite registrar la clasificación en régimen del interno y la fecha de clasificación. Actualmente la clasificación en régimen es el resultado de un estudio criminológico y es colegiado por un grupo de funcionarios. Para gestionar la clasificación en régimen se creó el módulo Comisión de clasificación del subsistema Área de Ingreso, Observación, Evaluación y Diagnóstico. (8)
- El paquete Procesos no maneja los estados de los procesos automáticamente sino que el usuario es el encargado de precisar qué estado tiene cada proceso en un momento dado. Esto introduce el error de mantener dos procesos con el estado cumpliendo incurriendo en el mismo problema del paquete Trámites legales pendientes.
- El paquete Documentos permite registrar los documentos de los procesos y los datos de los documentos, pero no hace distinción entre los datos de los diferentes tipos de documentos que reciben los internos. Esta solución impide tener identificados qué datos son propios de cada documento y consecuentemente qué datos son establecidos por cada órgano.

La solución dada por este módulo para el tratamiento de la situación legal de los internos en el Sistema Penitenciario Cubano aún refleja problemas en cuanto a la generación automática de trámites legales pendientes, las asignaciones de estatus legal, la definición de los estados de los procesos y la identificación de los datos por documentos emitidos por los órganos correspondientes.

El módulo Situación Legal, a pesar de concebir la dependencia con el ingreso de un interno a prisión, deja fuera de su alcance el manejo del estado del expediente legal de los internos. Para el SIDEP es importante mantener la concepción de que el ingreso define la situación legal inicial de los internos.

Los problemas referentes al control de la legalidad de los internos en el Sistema Penitenciario Cubano, que preceden el inicio de la informatización del Expediente Legal, no fueron resueltos con el diseño e implementación del módulo Situación Legal del SIGI. Esto conduce a la necesidad de un nuevo diseño que permita una solución ajustada a las necesidades reales del negocio.

1.3.3 Sistema de Gestión Penitenciaria

El Sistema de Gestión Penitenciaria (SIGEP) es una aplicación web desarrollada con el fin de automatizar los procesos penitenciarios en la República Bolivariana de Venezuela. Implementa “una solución de amplio alcance basada en la confección de un expediente carcelario para la automatización entre otros del proceso de Ejecución de la Pena y las Decisiones Judiciales.” (1)

En el SIGEP un individuo(a) puede tener varios Expedientes Penitenciarios, cada uno de ellos correspondiente al tránsito de este por el sistema penitenciario. Cada Expediente Penitenciario puede tener varios procesos judiciales, un estado, detenciones y ejecución de la pena. El estado del Expediente Penitenciario puede ser abierto o cerrado. Las detenciones evidencian los períodos en que el individuo(a) ha estado detenido fuera del sistema penitenciario y la ejecución de la pena tiene como objetivo mantener toda la información relacionada con el proceso judicial que está cumpliendo el individuo(a). (1)

Un proceso judicial contiene los delitos, las fases del proceso, los abogados, los órganos del Ministerio Público, la síntesis del hecho, el tipo de procedimiento y la jurisdicción. Cada proceso judicial tiene un estado que puede ser abierto cuando se inicia el proceso, reabierto si se desunifican los procesos, en ejecución una vez dictada la sentencia y cerrado cuando en alguna de las fases por las cuales transita el individuo(a) se dicte una decisión concluyente. (1)

El Expediente Penitenciario del SIGEP agrupa todos los procesos judiciales de un individuo(a), lo que impide la duplicidad de información en el sistema. Para el SIDEP se concibe también la idea de mantener un único expediente al cual se le asocien todos los procesos atribuidos a un interno en un período de tiempo de permanencia en el sistema penitenciario.

Sin embargo, el SIGEP maneja información que difiere de la información requerida por las leyes cubanas como: los documentos que avalan los procesos judiciales, los datos registrados en ellos, los estados de los procesos, la descripción del delito, la detección de las acumulaciones de ejecuciones y el registro de la condena. (9)

El Expediente Penitenciario del SIGEP persigue el mismo objetivo que el Expediente Legal del SIDEP pero el cómo maneja la información y los datos almacenados no están acordes a las leyes cubanas. Por tanto el SIGEP no es una solución viable para dar respuesta a las necesidades en el Sistema Penitenciario Cubano.

1.3.4 Sistema de Registro Penitenciario

El Instituto Nacional Penitenciario (INPE) de Perú es el órgano encargado de organizar y administrar las actividades y procesos del registro penitenciario de la institución (10). Para la gestión de los internos cuenta con un software (Sistema de Registro Penitenciario) que, entre otras funcionalidades, les permite organizar, centralizar y administrar la información sobre la situación jurídica y el cumplimiento de los procesos de ingreso de los internos a nivel nacional, registrando todos sus datos generales y las Diligencias Judiciales de los internos siguiendo los procedimientos descritos en el Manual de Procedimientos y Actividades de Registro Penitenciario del INPE; por esa misma vía reciben, tramitan y registran las Resoluciones Judiciales concernientes al interno. (11)

En el Sistema de Registro Penitenciario se registra un expediente personal para el interno conformado por los datos personales del interno, y toda la documentación (Social, Legal, Psicológica, de Salud, Educación, entre otras) del interno al momento de su ingreso. En este expediente se almacena y archiva los documentos de ingreso, detenciones, sentencias, clasificación, evaluación en el tratamiento y egreso del interno, generados durante su permanencia en el establecimiento penitenciario, desde su ingreso hasta su egreso. (11)

Para cada interno el Sistema de Registro Penitenciario gestiona documentos que contienen: (11)

- El oficio y/o resolución judicial que contiene el mandato de detención.
- Copias certificadas de sentencias para los casos de los internos sentenciados.
- Informe de clasificación.
- Récord de conducta y sanciones disciplinarias del interno.
- Resultado de las actividades educativas, laborales, recreativas, religiosas y otras en las que haya participado el interno.
- Copia de los oficios de orden de libertad del internos.
- Otros documentos que tengan relación con el interno.

La información del expediente personal del interno se puede procesar por medios manuales e informáticos en base a la información que proporcionan las unidades orgánicas de registro, seguridad y tratamiento, archivando estos en lugares apropiados con determinadas medidas de seguridad por orden alfabético en formato duro. (11) El archivo de documentos legales en formato duro, es una de las problemáticas de esta investigación que el Sistema de Registro Penitenciario no resuelve. Sin embargo mantiene la idea de la gestión de las clasificaciones legales del interno en función de los datos de los documentos. Un aporte de este sistema a la solución de la presente investigación, es la clasificación del estatus legal de los internos a partir de los documentos legales. Para ello es necesario tener informatizados estos documentos y el Sistema de Registro Penitenciario de Perú no ofrece esta funcionalidad.

El Sistema de Registro Penitenciario informatiza el expediente personal del interno basándose en las leyes de Perú, por lo que el uso de este software en el Sistema Penitenciario Cubano conlleva la reelaboración del mismo, lo que descarta su utilización.

1.3.5 OFFENDERTRAK Corrections Management System

OFFENDERTRAK es una solución de software empresarial proporcionada por Motorola, que ofrece a los trabajadores de los sistemas penitenciarios la posibilidad de manejar eficientemente grandes cantidades de información de los internos. (12) Permite administrar, procesar, archivar, recuperar y compartir la información de los delincuentes y sus incidencias. (13)

Con este software es posible: (13)

- Registrar los datos personales de los individuos.
- Clasificar a los individuos atendiendo a distintos criterios.
- Registrar proceso judicial de los individuos.
- Archivar el expediente del individuo una vez dado de libertad.
- Reingresar al individuo y evaluar su clasificación atendiendo al nuevo delito y los antiguos registrados.

Una característica relevante de este software es que permite configurar el flujo de trabajo de la informatización del expediente legal del interno en función de los requerimientos del sistema penitenciario que lo use. Además, garantiza la integración con otros sistemas legales informáticos (12)

Sin embargo, este software es propietario. El alto costo por la adquisición de OFFENDERTRAK (6.4 millones de dólares en el estado de Mississippi, Estados Unidos (14)) descarta la implantación de este sistema en el Sistema Penitenciario Cubano.

1.4 Metodología, herramientas y tecnologías

Para el diseño e implementación del módulo Expediente Legal se emplea el ambiente de desarrollo seleccionado por el equipo de arquitectura del SIDEP, al que se integrará la solución que propone esta investigación. A continuación se mencionan las características del ambiente de desarrollo empleado.

- Para modelar la base de datos, el ER/Studio 8.0, herramienta líder de modelado de datos para el diseño y construcción de bases de datos a nivel físico y lógico.
- Para el diseño del módulo, el Visual Paradigm for UML 6.1. Con el Visual Paradigm se modelaron los paquetes, las clases, las interacciones, los componentes, de estado y el despliegue del módulo.
- Para la implementación del módulo se usó Eclipse STS 2.5.0 como entorno de desarrollo, para el control de versiones VisualSVNServer y SVNTortoise y el contenedor de aplicaciones web Apache Tomcat 6.0.
- Se usó Oracle 11 g R2 como gestor de bases de datos.
- Para la programación del módulo se usó el marco de trabajo de desarrollo web Grails, que usa Groovy como lenguaje de programación.
- Se usó JavaScript como lenguaje del lado del cliente y para la interfaz visual se usó dojo y AJAX.
- Se usó RUP como metodología de desarrollo de software.

1.5 Conclusiones

En este capítulo se describieron los conceptos relacionados con los procesos a informatizar para un mejor entendimiento del negocio. Se analizaron soluciones informáticas para sistemas penitenciarios y se demostró que no resuelven la problemática planteada evidenciando la necesidad de realizar una nueva propuesta que se ajuste a las necesidades del Sistema Penitenciario Cubano. Se realizó un estudio de la metodología, tecnologías, lenguajes y herramientas seleccionadas para el SIDEP. Este estudio arrojó que:

1. Una elección factible para desarrollar el SIDEP es la metodología RUP, puesto que esta metodología constituye una guía de cómo se debe desarrollar una aplicación de gran escala.

2. El marco de trabajo seleccionado, que incluye los lenguajes de programación descritos, permite implementar mayor cantidad de funcionalidades en un menor tiempo.
3. Las herramientas seleccionadas facilitan el desarrollo en cuanto al control de las versiones, el modelado y la implementación del módulo.

En este capítulo, se obtuvieron los elementos conceptuales necesarios para iniciar la construcción del módulo Expediente Legal del SIDEP.

Capítulo 2. Diseño del sistema

2.1 Introducción

En el presente capítulo se hace mención a los casos de uso del módulo Expediente Legal, se describe el marco de trabajo de desarrollo web y la arquitectura definida para el SIDEP. Con la descripción de los casos de uso y la arquitectura se procede a diseñar el módulo mediante la elaboración de sus diagramas de clases y sus diagramas de secuencia. Contiene además el diagrama de paquetes, los diagramas de estados para las entidades que lo requieren y una descripción del modelo entidad relacional de la base de datos.

2.2 Resumen de los casos de uso del módulo Expediente Legal

En el documento Especificación de casos de uso del módulo Expediente Legal (41), se describen los casos de uso que reflejan el funcionamiento del sistema en cuanto a la gestión del expediente legal del interno en el Sistema Penitenciario Cubano. El entendimiento de estos casos de uso es necesario para diseñar el módulo y para su posterior implementación. A continuación se listan los casos de usos definidos y un resumen que describe su comportamiento.

Tabla 1: Resumen de los casos de uso

2.3 Concepción del expediente legal en el SIDEP

El módulo Expediente Legal del SIDEP permite la creación de un expediente único asignado al interno desde que ingresa al centro penitenciario hasta que egresa.

El Expediente Legal está conformado por una situación legal que enmarca datos generales independientes a los procesos, los procesos, los quebrantamientos de la sanción y tiene un estado –*abierto* o *cerrado*- Se dice que el expediente está *abierto*, cuando el interno haya efectuado un ingreso y esté transitando por el Sistema Penitenciario Cubano cumpliendo sanción, medida cautelar, medida de seguridad o estando de libertad anticipada. Al egresar, el expediente toma el estado de *cerrado* o *deshabilitado*. Un expediente toma estado *cerrado* cuando se le otorga al interno un egreso definitivo. Una vez que se cierra un expediente, no se vuelve a trabajar sobre él. Un expediente *abierto* puede tomar otros dos estados: *habilitado* o *deshabilitado. Habilitado* cuando el interno está cumpliendo la sanción

dentro de un centro penitenciario y *deshabilitado* cuando el interno no se encuentra en un centro penitenciario (Figura 1).

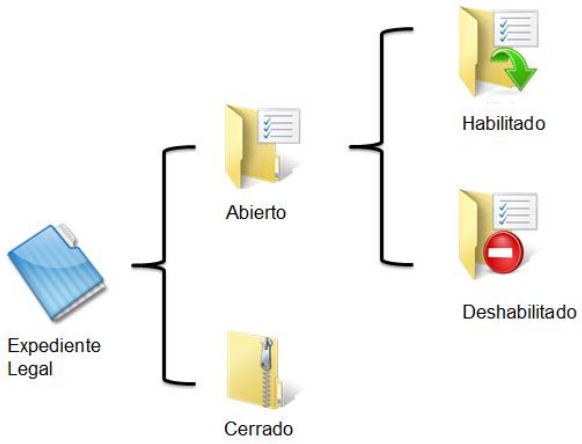

Figura 1: Estados del Expediente Legal

Un proceso está compuesto por documentos legales expedidos por las autoridades competentes que amparan su internamiento, trámites legales pendientes e ingresos del interno. Cada proceso tiene un estado (cumpliendo, cumplido, por cumplir, pendiente, total conjunta, conjunta, interrumpido) que es determinado a partir de los datos de los documentos.

Para diseñar el Expediente Legal es necesario tener en cuenta la arquitectura del marco de trabajo de desarrollo web definido para el SIDEP.

2.4 Marco de trabajo de desarrollo web

Grails es el marco de trabajo para aplicaciones web seleccionado por el equipo de arquitectura para desarrollar el SIDEP. Desarrollado sobre el lenguaje de programación Groovy. Proporciona un entorno de desarrollo estandarizado y oculta gran parte de los detalles de configuración al programador, siguiendo paradigmas como *Convención sobre configuración* o *No te repitas* (del inglés, Don't Repeat Yourself). (42)

*"*Una aplicación Grails contiene lo necesario para desarrollar desde el primer momento. En ella se definen un conjunto de paquetes ordenados según su arquitectura y tipos de clases que vayan a almacenar para que al relacionarse entre sí generen la aplicación final." (43) (Figura 2)

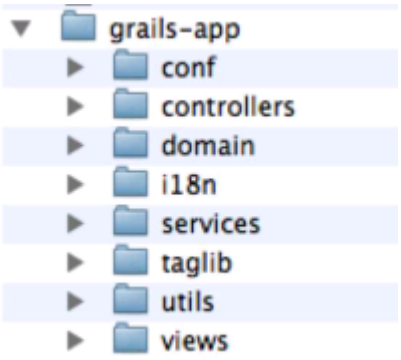

Figura 2: Estructura de un proyecto Grails

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

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

Grails no solo es un marco de trabajo de desarrollo web de la plataforma Java sino que es considerado una plataforma de trabajo donde convergen varias tecnologías, tales como Hibernate, SiteMesh y Spring. (44)

La arquitectura de Grails soporta la adición de plugins para cuestiones de seguridad, AJAX, búsquedas, reportes, servicios web y una comunidad de desarrolladores a nivel mundial. (42)

El desarrollo de plugins para una aplicación Grails es una característica que el equipo de arquitectura del SIDEP decidió utilizar. La aplicación consta de 48 módulos distribuidos en 7 subsistemas. En la Figura 3 puede observarse la estructura de plugins del SIDEP, agrupados por los conceptos:

1. *Proyecto:* proyecto Grails (SIDEP).
2. *Subsistemas*: 7 plugins de Grails que representan los 7 subsistemas.
3. *Paquetes*: representan los 48 módulos ubicados en sus respectivos subsistemas.
4. *Componentes:* representan los componentes resultantes de la implementación del SIDEP.

El módulo Expediente Legal es un paquete que pertenece al subsistema Registro Legal del SIDEP.

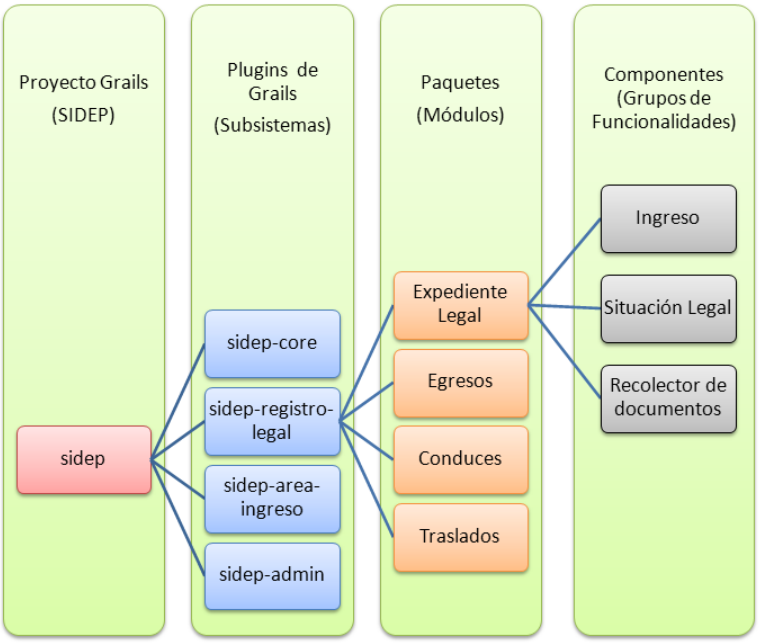

Figura 3: Estructura del SIDEP

2.4.1 Modelo Vista Controlador

Respecto al diseño, Grails sigue el patrón Modelo-Vista-Controlador (MVC), el cual provee un mecanismo para construir una capa web estableciendo que los componentes de una aplicación web deben organizarse en 3 capas distintas según su misión: (43)

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

En términos de aplicación web, los controladores son los responsables de interceptar las peticiones HTML del navegador y generar la respuesta correspondiente. Para evitar que las clases de la capa de control estén sobrecargadas de código fuente dedicado a la implementación de la lógica del negocio, los controladores delegan esta responsabilidad a los componentes de la Capa de servicios:

- *Capa de servicios:* contiene los componentes encargados de implementar la lógica de negocio de la aplicación. (43)

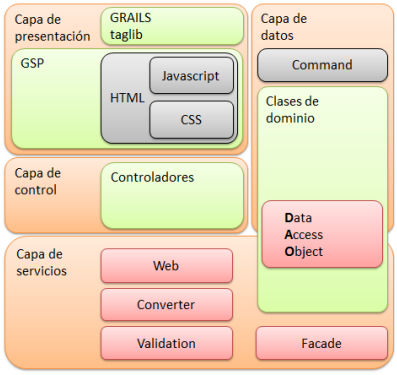

Figura 4: Patrón MVC en Grails

La aplicación del patrón MVC en el diseño permite obtener componentes más pequeños y fáciles de mantener, y la posibilidad de reutilizar el código en mayor medida.

2.4.2 Modelo de dominio

El núcleo de una aplicación Grails es su modelo de dominio, que no es más que la definición de un conjunto de clases del dominio con sus relaciones. Una clase dominio representa datos persistentes y por defecto constituye una entidad de la base de datos.

Una característica interesante de las clases dominio de Grails, son las convenciones usadas para representar las relaciones entre clases (Figura 5). En una clase dominio se usa:

- static hasMany: para dejar constancia que la clase contenedora de la sentencia contiene muchos objetos de la clase ubicada en la cláusula después del signo de asignación.
- static belongsTo: para indicar que la clase contenedora de la sentencia pertenece o indica la parte N de una relación 1:N con la clase ubicada en la cláusula después del signo de asignación.
- static hasOne: para indicar que la clase contenedora de la sentencia contiene un objeto de la clase ubicada en la cláusula después del signo de asignación.
- La instanciación de una clase: para indicar una relación de 1:1 o 1:N con la clase que la contiene instanciada.

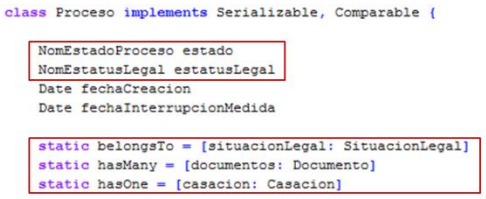

Figura 5: Relaciones entre clases en las clases de dominio de Grails

2.4.3 Los controladores

Las clases controladoras son las responsables de recibir las solicitudes del usuario, aplicar la lógica del negocio sobre el modelo y decidir la vista que se debe mostrar a continuación. Debido a las convenciones de Grails, las clases controladoras serán ubicadas en la carpeta grails-app/controllers, tendrán la estructura *<NombreDelControlador>Controller.groovy* y responderán a la URL *http://servidor:puerto/Aplicación/ <nombreDelControlador>/<nombreDeLaAcción>/<id>.*

En el módulo, se nombran las clases controladoras atendiendo a dos aspectos: el nombre de la funcionalidad o la gestión de una clase de dominio específica. Por ejemplo, en el diseño de la

funcionalidad de ingresar un interno en el SIDEP, se creó un controlador con nombre *IngresoSitLegalController*, y para la gestión de los procesos del interno, *ProcesoRegistroLegalController.*

2.4.4 Vistas: Groovy Server Pages

Las vistas en una aplicación MVC son las clases responsables de mostrar al usuario el estado del sistema y las acciones que tiene a su disposición. En Grails se desarrollan las vistas mediante Groovy Server Pages (GSP), una versión simplificada de JSP (del inglés Java Server Pages) que permite intercalar expresiones en el código HTML (43), por ejemplo:

```
<html>
<head>...</head>
<body>
   <g:set var="miVariable">
      Hoy es: ${new Date()}
   </g:set>
    <p>
      Lo que tengo que decir es:
      <strong> ${miVariable} </strong>
   </p>
</body>
</html>
```

Figura 6: Ejemplo de código de una página GSP

2.4.5 Los servicios

El equipo de Grails no sugiere la inclusión de lógica de negocio en la capa de control, y recomienda hacerlo mediante servicios (al hacerlo se asegura una fase de mantenimiento menos complicada). (45) Un servicio es una clase cuyo nombre sigue el patrón <*NombreDelServicio*>*Service.groovy* y que se aloja en la carpeta *grails-app/services.* La asignación de nombres de los servicios sigue la misma convención de la asignación de nombres a los controladores. Para usar un servicio, cualquier controlador, librería de etiquetas, etc., puede declarar una variable de tipo el servicio, lo que activará la inyección automática de la dependencia por parte de Spring.

Grails controla el ciclo de vida de los servicios decidiendo cuándo se crean instancias de los mismos. Por defecto todos los servicios utilizan el patrón singleton: solo existe una instancia de la clase que se inyecta en todos los artefactos que declaren la variable correspondiente. Esto es posible gracias a las políticas de creación de instancias explicadas en el Manual de desarrollo web con Grails. (43)

Los servicios que implementan lógica de negocio que no procesan los datos de un interno en una sesión iniciada por un usuario del módulo, serán singleton. Para el caso que se requiera mantener información visible solamente para el usuario que está conectado, es necesario cambiar este tipo de comportamiento.

Para lograr una instancia única para cada sesión de usuario, es necesario declarar los servicios como se muestra en la siguiente figura:

```
class SituacionLegalService {

    static scope = "session"
```

Figura 7: Declaración de la variable *scope* para la creación de instancias de un servicio Grails

De esta forma, los usuarios del módulo pueden estar realizando peticiones a una misma acción de un controlador, que gestione lógica de negocio mediante *SituacionLegalService* (la clase tomada como ejemplo en la Figura 7) y recibir información de los internos que hayan sido seleccionados en cada sesión de usuario.

Grails implementa el patrón inversión de control (IoC), con el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste sólo contenga la lógica necesaria para hacer su trabajo. En un controlador solo basta definir una variable con el nombre del servicio que se desea emplear. No es necesario instanciar el servicio, ni configurarlo de ningún modo antes de usarlo. En su lugar, Grails usa el marco de trabajo Spring para ese tipo de tareas. (43)

Cuando se crea un componente en la aplicación, Grails configura Spring para que gestione su ciclo de vida (cuándo se crea, cuántas instancias se mantienen activas a la vez, cómo se destruyen, etc.) y sus dependencias (qué otros componentes necesita para realizar su trabajo y cómo conseguirlos). (43)

El objetivo de esta técnica es mantener los componentes lo más sencillo que sea posible, incluyendo código que tenga relación con la lógica de negocio. Así, la aplicación será más fácil de comprender y mantener.

2.4.6 Arquitectura en capas

El equipo de Grails define una arquitectura en capas compuesta de la siguiente forma: (44)

- Capa web: encargada de la presentación (compuesta por los controladores y las GSP).
- Capa de servicios: encargada de la implementación de la lógica del negocio (compuesta por los servicios).

- Capa de dominio: encargada de persistir los datos en la base de datos (compuesta por las clases del dominio).

Según la arquitectura de Grails: (Figura 8)

- Las clases de la capa de dominio pueden ser utilizadas en las demás capas de la aplicación.
- Los controladores, de la capa web, pueden utilizar todos los tipos de clases de la aplicación.
- No existe relación entre las GSP y los servicios.

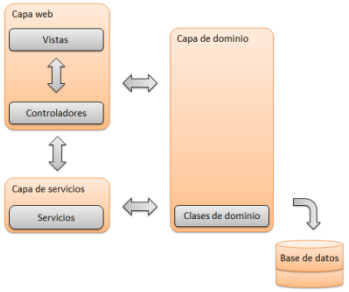

Figura 8: Arquitectura en capas de Grails

Conociendo las características de la arquitectura de Grails y el funcionamiento de sus clases, se procede a diseñar el módulo Expediente Legal del SIDEP.

2.5 Diagrama de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Está compuesto por los siguientes elementos: clase: atributos, métodos y visibilidad y relaciones: herencia, composición, agregación, asociación y uso. (47)

Los diagramas de clases con estereotipos web se confeccionan para diseñar el flujo de acciones descritas en los casos de uso y a partir de la arquitectura en capas definida:

- En la capa web se ubican las vistas y los controladores:
    - Los controladores se encargan de recibir o manejar las peticiones de los usuarios u otros eventos del sistema. En consecuencia, ordenan generar las interfaces de usuario o archivos JSON –para cuando se requiera algún dato del servidor, solicitándolos mediante

AJAX–, redirigen las peticiones hacia otro controlador u ordenan aplicar la lógica de negocio del módulo apoyándose en las clases servicios ubicadas en la capa de servicios.

    - Las vistas son las encargadas de generar las interfaces de usuario.

- En la capa de servicios se ubican los servicios en los que se implementa la lógica de negocio del módulo Expediente Legal. En estas clases se implementa, además, el acceso a datos apoyándose en las clases de dominio.
- En la capa de dominio se ubican las clases –de dominio– que representan los conceptos, artefactos y objetos del negocio cuyos datos son los que se persistirán en la base de datos.

. A continuación se muestra el diagrama de clases del diseño con estereotipos web del caso de uso Registrar proceso por el que cumple, que describe el registro del primer proceso de un interno en el sistema. Para lograr una mejor visualización del diagrama, se muestra fragmentado por capas:

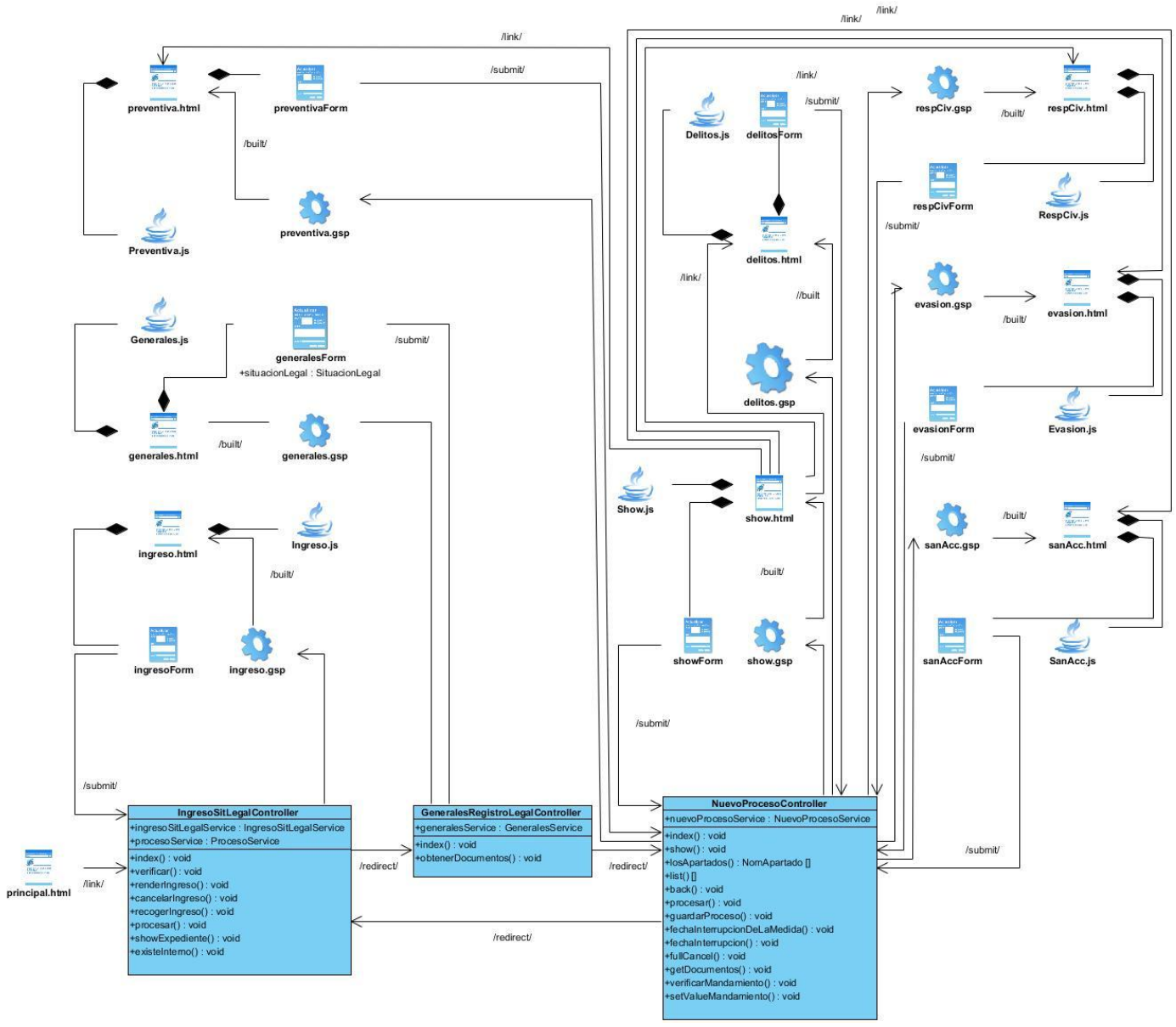

Figura 9: Capa web en el diagrama de clases del diseño con estereotipos web del caso de uso Registrar proceso por el que cumple

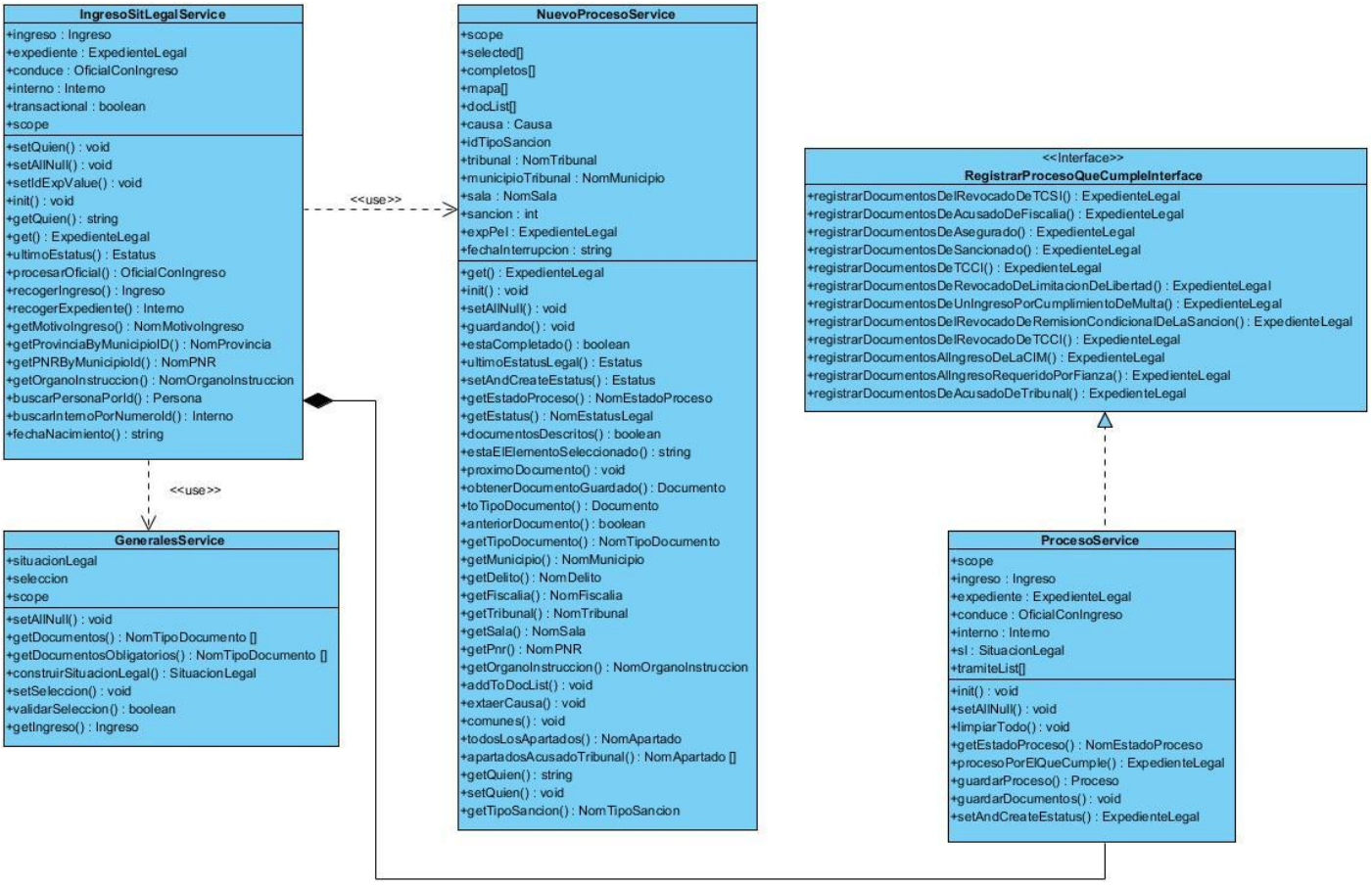

Figura 10: Capa de servicios en el diagrama de clases del diseño con estereotipos web del caso de uso Registrar proceso por el que cumple

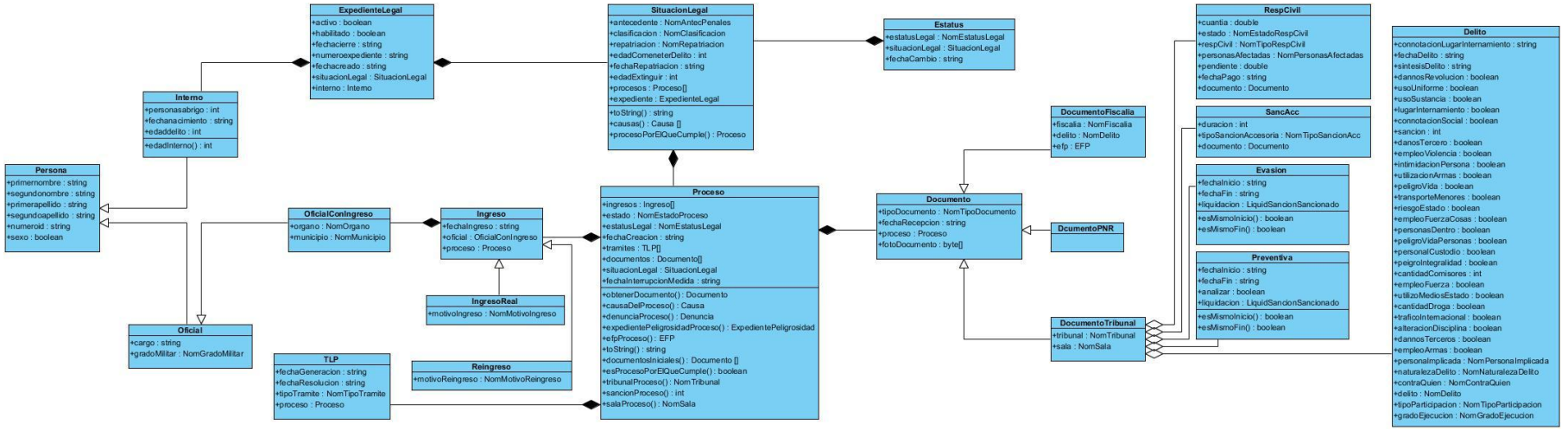

Figura 11: Capa de dominio en el diagrama de clases del diseño con estereotipos web del caso de uso Registrar proceso por el que cumple

2.5.1 Patrones utilizados

En la capa web se emplea el patrón MVC explicado en la sección 2.4.1 Modelo Vista Controlador. En las capas de servicio y de dominio se utilizan los patrones Bajo Acoplamiento, Alta Cohesión, Experto y Creador, los cuales son patrones para asignar responsabilidades (GRASP). Estos patrones “no introducen ideas novedosas; son una mera codificación de los principios básicos más usados” (47) en el diseño.

El uso del patrón Bajo Acoplamiento se evidencia cuando una clase referencia a otra en forma de atributo (X tiene una variable de tipo Y), es una subclase directa o indirecta de otra (X hereda de Y), tiene un método que en su implementación referencia una instancia de otra o implementa una interfaz (Y es una interfaz y X la implementa). Las clases con bajo acoplamiento no se afectan por cambios de otros componentes, son fáciles de comprender por separado y fáciles de reutilizar. (47)

El patrón Alta Cohesión se pone de manifiesto cuando una clase tiene responsabilidades moderadas en un área funcional y colabora con otras para llevar a cabo las tareas. (47) En este caso se encuentran las clases de la capa de servicios del caso de uso Registrar proceso por el que cumple. (Figura 10)

Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y las mejoras. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización. Las clases altamente acopladas a menudo generan un bajo acoplamiento. (47)

Estos patrones pueden ser vistos de forma separada, aunque están fuertemente ligados; de hecho si aumenta demasiado la cohesión del sistema, se obtiene un bajo acoplamiento entre las clases, y por el contrario si se reduce el acoplamiento, aumentada la cohesión.

Se utiliza el patrón Experto cuando se le da la responsabilidad a varias clases que disponen de la información necesaria, a cumplir con una determinada tarea. En el modelo de dominio del módulo y formando parte de las clases del caso de uso Registrar proceso por el que cumple (Figura 11), se encuentran las clases *SituacionLegal* y *Proceso.* Estas clases tienen la responsabilidad de participar en el proceso de asignación o modificación del estatus legal del interno. Para ello es necesario conocer el *proceso* por el que el interno se encuentra cumpliendo. La clase *Proceso* cumple con esta responsabilidad de forma parcial, pues es ella la que conoce su estado –mediante su atributo *estadoProceso*–. Sin embargo no es suficiente, puesto que un interno puede tener varios procesos, contenidos en la clase *SituacionLegal,* por lo que esta clase, es la experta en la detección del proceso por el que cumple el interno.

Con la aplicación de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. (47)

El patrón Creador es mayormente aplicado en la capa de servicios del módulo, asignándoles a las clases de esta capa la responsabilidad de crear objetos. En el diagrama de la Figura 10 la clase *IngresoSitLegalService,* tiene la responsabilidad de crear las instancias que representan al interno, su expediente legal, su ingreso y al oficial que lo condujo hasta el centro penitenciario.

La aplicación de este patrón soporta un bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. (47)

2.6 Diagramas de Interacción

Los diagramas de interacción son diagramas que describen cómo los grupos de objetos colaboran para conseguir algún fin, es decir, la forma en cómo un cliente (actor) u objetos (clases), se comunican entre sí ante una petición. Estos diagramas muestran los objetos, así como los mensajes entre estos objetos. Los diagramas de interacción capturan el comportamiento de los casos de uso y pueden expresarse de dos formas, mediante diagramas de secuencia o diagramas de colaboración. (47)

Las clases del diagrama –fragmentado por capas– de la sección 2.5 Diagrama de Clases interactúan de la siguiente forma:

> Inicialmente se cuenta con una interfaz principal que hace una petición al controlador principal del módulo, quien analiza si lo que sucederá a continuación es un ingreso. De ser positiva su respuesta dirige el flujo del caso de uso para el controlador de ingreso quien ordena a una página servidora que genere una vista para que el usuario introduzca los datos que serán guardados en el servicio destinado a atender los datos del ingreso.
>
> Luego el controlador que gestiona los datos generales de la Situación Legal ordena a su página servidora que genere una vista que contiene los datos básicos de la Situación Legal del interno. El servicio que gestiona los datos generales de la Situación Legal del interno, determina qué documentos se mostrarán a continuación en dependencia del motivo de ingreso seleccionado por el usuario en la vista anterior. Otra tarea que hace este servicio es inicializar el componente Recolector de documentos, encargado del registro de los datos plasmados en los documentos que se remiten con el ingreso del interno.

En dependencia de los documentos a mostrar, el controlador principal del Recolector de documentos ordena mostrar sus respectivas interfaces de usuario. Además, el registro de los delitos, sanciones accesorias, responsabilidades civiles o preventivas es tarea de este controlador. Todos los datos se guardan momentáneamente en memoria. Cuando el usuario registra los datos del último documento, el controlador de ingreso ordena a su correspondiente clase de servicio guardar estos datos. Por último, las clases de dominio realizan la acción de persistencia de los datos en la base de datos.

2.7 Diagramas de estado

Los diagramas de estado describen “visualmente los estados y eventos más interesantes de un objeto, así como su comportamiento ante un evento” (47). Un diagrama de estado puede aplicarse a clases, casos de uso o conceptos. En el módulo Expediente Legal existen algunas clases que cambian su estado en dependencia de determinados eventos que ocurren en el sistema, por lo que los diagramas de estado posibilitan entender estas transiciones fácilmente.

Una de las clases que cambia su estado es la clase Proceso. La Figura 12: Diagrama de estado para la clase Proceso muestra los cambios de estado de la clase mencionada cuando el interno ingresa al Sistema Penitenciario Cubano como acusado.

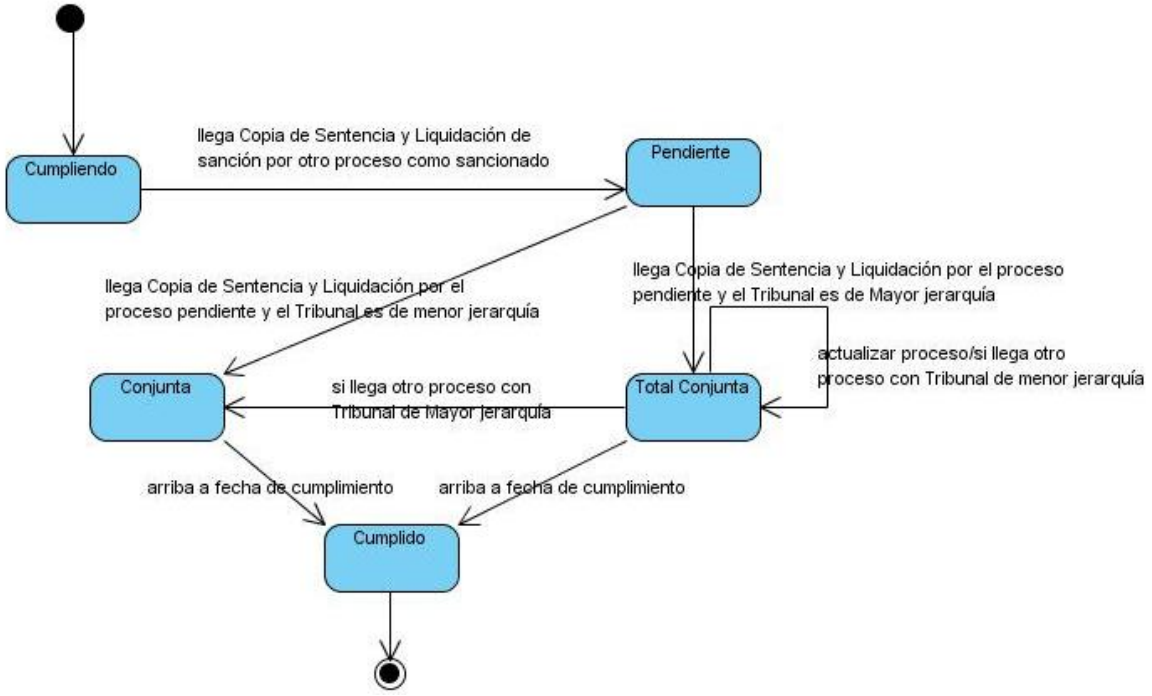

Figura 12: Diagrama de estado para la clase Proceso

El Expediente Legal es otra de las clases que cambia su estado. La Figura 13: Diagrama de estado de la clase Expediente Legal del interno, representa el cambio de estado del expediente legal que fue descrito en la sección 2.3 Concepción del expediente legal en el SIDEP.

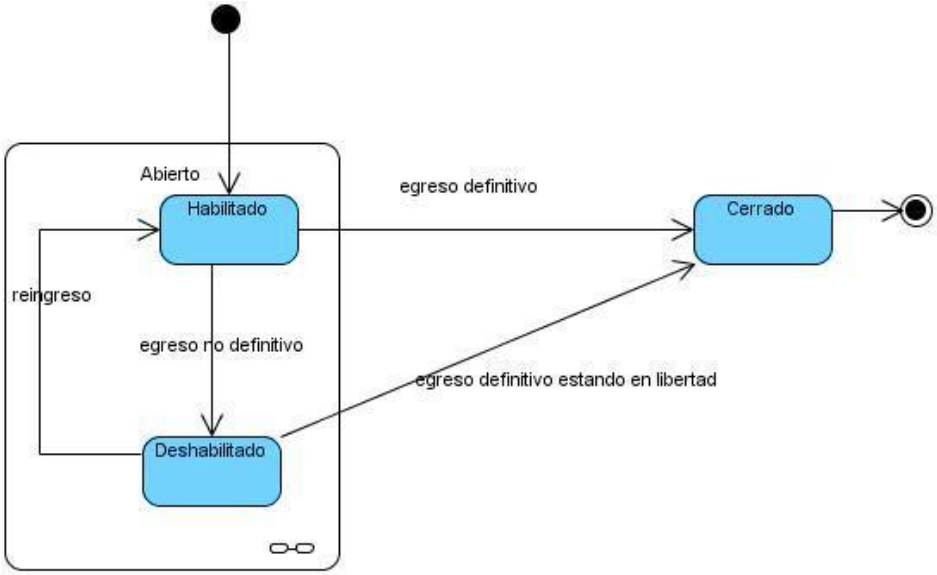

Figura 13: Diagrama de estado de la clase Expediente Legal del interno

El Estatus legal es la entidad más importante en el módulo. Su estado cambia cuando se registran o actualizan los procesos para el interno. Este cambio se puede apreciar en el diagrama de la Figura 14. Un interno puede tomar uno de los cuatro estatus al ingresar y varía con las conversiones en su situación legal.

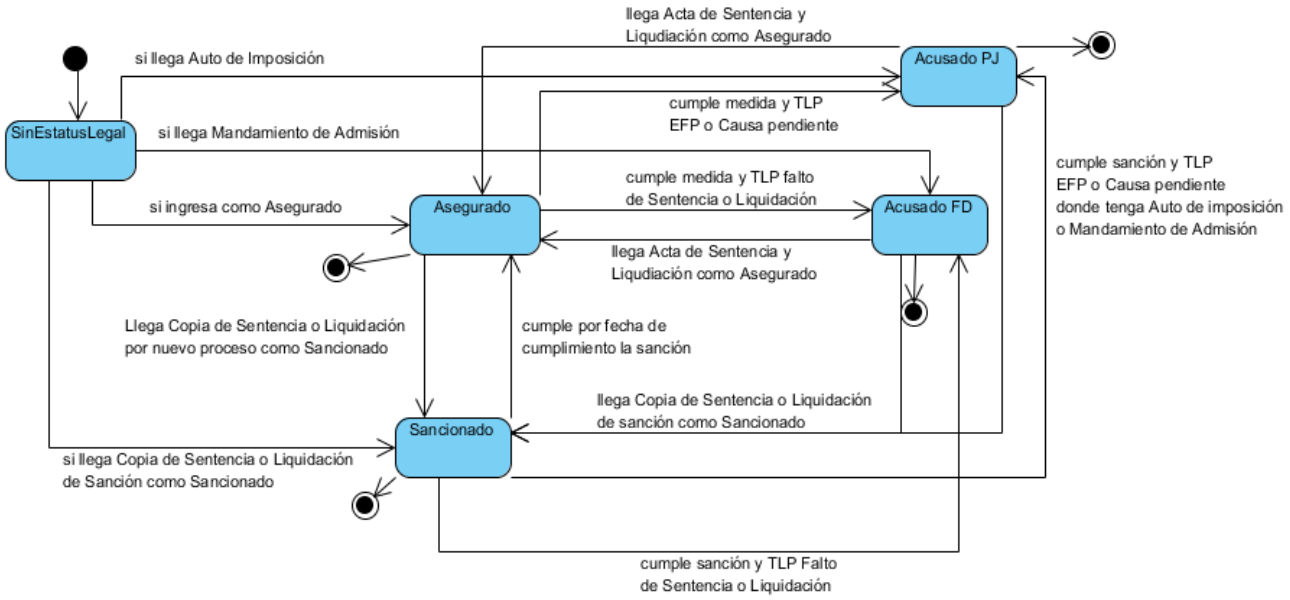

Figura 14: Diagrama de estado de la clase Estatus legal de un interno

2.8 Diseño de la Base de Datos

El proceso de diseño de la base de datos se realiza basándose en los diagramas resultantes de las clases del diseño.

Durante el diseño de la base de datos se utilizaron patrones de diseño de base de datos como árboles fuertemente codificados (para representar jerarquías donde es bien conocida la estructura), árboles estructurados (para representar las herencias entre las tablas), y máquina de estado. (48)

En el Sistema Penitenciario Cubano un interno puede tener varios expedientes. Si el interno se encuentre cumpliendo en un centro penitenciario tendrá un expediente abierto; cuando egrese, el expediente se cerrará automáticamente. La relación puede observarse en la Figura 15:

Un interno tiene siempre una situación legal y un expediente abierto por lo que la relación entre SITUACION_LEGAL y el EXPEDIENTE_LEGAL es de 1:1. En la Figura 15 queda evidenciado que para crear un expediente legal, es necesario conocer la situación legal del interno y el interno que se está creando.

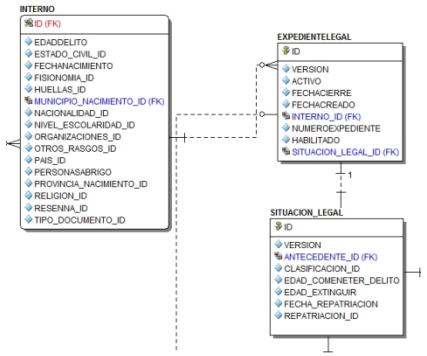

Figura 15: Relación SITUACION_LEGAL, EXPEDIENTE_LEGAL, INTERNO

Por conceptos de trabajo explicados en los Procedimientos de Trabajo de Registro Legal (6) una situación legal contiene entre otros aspectos, todos los procesos por los que se está procesando al interno los cuales modifican el estatus legal del mismo. Es por ello que la relación entre SITUACION_LEGAL, ESTATUS y PROCESO quedan de la siguiente forma:

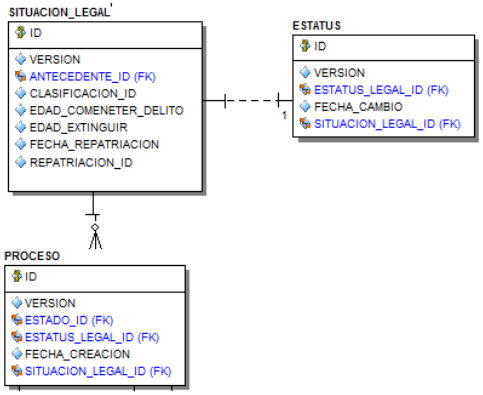

Figura 16: Relación entre SITUACION_LEGAL, PROCESO y ESTATUS

Uno de los requisitos del cliente es saber la fecha de cambio del estatus de un interno, pues el interno en el sistema puede ingresar como acusado y egresar como sancionado. Además, en ocasiones se requiere conocer el proceso que modificó el estatus legal del interno en determinado momento. Como se muestra en la Figura 16, en la tabla

ESTATUS se almacenarán todos los cambios de estatus legal para un interno. De esta forma se aplica el patrón máquina de estado para dar solución a los requisitos mencionados.

Un proceso son todas las características legales que encierran una causa añadida por un tribunal a un interno. En caso de que el tribunal aún no haya sancionado al interno, entonces está definido por la denuncia que se le levanta en los órganos de instrucción o la medida cautelar de la fiscalía. En caso de ser una persona asegurada, estará definido por su expediente de peligrosidad. Todos estos datos están plasmados en los documentos con los que el interno ingresa a prisión. En el proceso es donde quedan registrados los trámites legales pendientes del interno. (Figura 17)

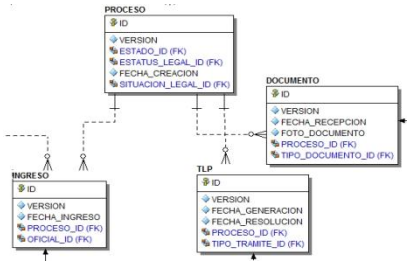

Figura 17: Proceso

En el proceso queda registrado, además, los datos del ingreso de un interno en un centro penitenciario. Un ingreso puede ser ingreso real o reingreso.

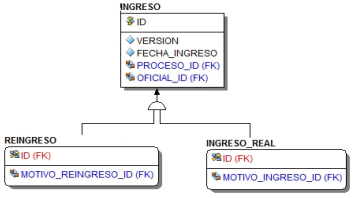

Figura 18: Estructura del ingreso

Los documentos contienen los datos requeridos que garantizan las conversiones de la situación legal de los internos: las causas, las denuncias, los expedientes en fase preparatoria, los expedientes de peligrosidad, los tribunales, los órganos de instrucción, las fiscalías y los tribunales.

2.9 Conclusiones

En este capítulo se mostró un resumen de los casos de uso del módulo para obtener un mayor entendimiento del negocio a diseñar. A partir del estudio de los casos de uso, se definió la concepción del Expediente Legal en el SIDEP.

Se describió la arquitectura definida para el SIDEP y se explicaron los diferentes tipos de clases que componen del marco de trabajo de desarrollo web Grails. Teniendo en cuenta esta arquitectura se procedió al diseño del módulo donde se realizaron los diagramas de paquetes, de clases con estereotipos web, de secuencia y de estado. Se describió, además, el modelo entidad relacional de la base de datos para finalizar el diseño.

Este diseño fue explicado basándose en la solución dada al caso de uso Registrar proceso por el que cumple, que servirá como guía para la implementación del módulo en el Capítulo 3. Implementación y prueba.

Capítulo 3. Implementación y prueba

3.1 Introducción

El presente capítulo tiene como objetivo explicar la implementación del módulo, que tiene como entrada el resultado del diseño. Como parte de la implementación se muestran los diagramas de componentes y de despliegue. Se define una política de tratamiento de errores y se incluye el diseño de las pruebas y los resultados obtenidos de la aplicación de las pruebas al módulo.

3.2 Implementación del módulo Expediente Legal

El flujo de trabajo Implementación, de RUP, explica cómo desarrollar, organizar, realizar pruebas e integrar los componentes implementados basándose en las especificaciones de diseño (18). En este se implementan las clases y objetos en ficheros fuente, binarios y ejecutables. Todos los elementos implementados forman el modelo de implementación, se integran de forma incremental para en caso de que ocurran fallos sean más fáciles de detectar. El resultado final de este flujo de trabajo es un sistema ejecutable (17).

3.2.1 Diagrama de Componentes

"Un diagrama de componentes muestra la estructura de los componentes, incluyendo clasificadores que especifican componentes, y artefactos que los implementan. En esta directriz se identifican las situaciones en que crear estos diagramas puede resultar útil." (18) Es un diagrama de clases a gran escala. Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la implementación de aplicaciones informáticas. Un módulo del sistema se puede representar como un paquete de componentes.

En el diagrama de componentes principal se muestra el módulo Expediente Legal como componente principal implementado. Se relaciona con los componentes que representan a los módulos Egresos y Comisión de Clasificación del SIDEP.

La interfaz visual del módulo es gestionada mediante el componente Plantilla SIDEP, el cual usa Dojo para la creación y animación de los componentes de interfaz de usuario. El componente *App.js* es un archivo con funcionalidades comunes para todos los módulos del SIDEP.

La interfaz *Table* le permite a todos los módulos generar tablas en la interfaz de usuario capaces de soportar cualquier tipo de datos. Es por ello que el módulo Expediente Legal la implementa, al igual que el buscador del sistema, capaz de buscar personas, internos, oficiales, trabajadores, entre otros elementos.

Otros componentes con el que se relaciona el módulo Expediente Legal es la librería de etiquetas del SIDEP (SidepTagLib), para componer las interfaces visuales de la solución y la librería de componentes de seguridad que el componente administrativo del SIDEP (SidepAdmin) brinda.

El módulo está compuesto por tres submódulos: Ingreso, Reingreso y Situación Legal (Figura 19). Este último contiene los componentes Proceso, Responsabilidades civiles, Trámites legales pendientes, Casación, Quebrantamientos de la Sanción y el Recolector de documentos. Estos submódulos usan la clase SelectedChecked que está ubicada en el paquete *util* del módulo.

La clase SelectedChecked fue creada para recordar la selección de los componentes visuales de tipo *check* del lenguaje HTML.

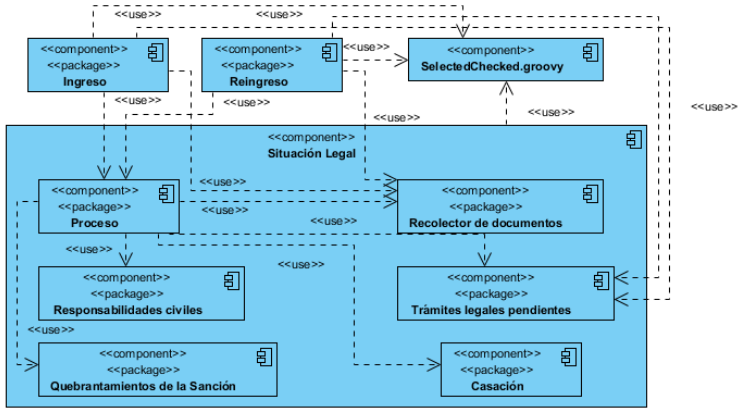

Figura 19: Componente Expediente Legal

3.2.2 Descripción del funcionamiento de los componentes

En esta sección se expone el funcionamiento de las clases *NuevoProcesoService.groovy* que conforma el componente Recolector de documentos y *DescripcionCUService.groovy* que sirve de apoyo en la aplicación de la lógica del negocio en el módulo y pertenece al componente Situación Legal. Además, se manifiesta el funcionamiento de los componentes Ingreso y Proceso, subcomponentes principales del componente Expediente Legal.

3.2.2.1 La clase *NuevoProcesoService.groovy*

El componente más importante desarrollado en el módulo fue el Recolector de documentos. Se compone por un conjunto de clases distribuidas por todas las capas de la aplicación cuya clase principal es *NuevoProcesoService.groovy*. Su objetivo es informarle al controlador del componente qué documento corresponde mostrar al usuario en un momento determinado.

Los principales atributos de esta clase son:

- *mapa*: mapa de valores que representa los posibles documentos que podría seleccionar el usuario o los documentos que se fueran a mostrar a un funcionario.
- *selected*: lista de elementos que fueron seleccionados o que se mostrarán en dependencia de la situación de actualización de un proceso.

- *completos*: lista que contiene los documentos de la selección que ya fueron procesados.
- *docList*: lista que contiene los objetos de tipo Documento que fueron procesados por el Recolector de documentos.

Su funcionamiento base es simular un flujo web apoyándose en los métodos *proximoDocumento()* y *anteriorDocumento()*, que actúan en dependencia de si el usuario desea visualizar el próximo o el anterior documento. Un flujo web encapsula una secuencia de pasos que guían al usuario durante la ejecución de una tarea del negocio. (49)

```
def proximoDocumento(def proximo) {
    if (!(proximo in completos)) {
        completos.add proximo
    }
    selected.remove proximo
}
```

```
def anteriorDocumento() {
    def atras = ""
    if (completos != []) {
        Collections.reverse(completos)
        atras = completos.get(0)
        completos.remove(completos.get(0))
        selected.add 0, atras
        Collections.reverse(completos)
        return true
    }
    false
}
```

Figura 20: Métodos del Recolector de documentos

Una vez que la lista de documentos *selected* esté vacía *NuevoProcesoService.groovy* le anuncia el fin de su funcionamiento al controlador, quien dirigirá el flujo de acciones hacia el controlador que lo haya invocado.

3.2.2.2 La clase *DescripcionCUService.groovy*

Uno de los requisitos descritos en los casos de uso relacionados con el ingreso y con el registro del proceso por el que cumple, son los valores por defecto en varios campos en dependencia del motivo de ingreso seleccionado por el usuario. La clase *DescripcionCUService* se encarga de estos valores y ayuda al Recolector de documentos en esta cuestión. Para ello se apoya en sus dos atributos:

- *cu*: que define el caso de uso que esté ejecutándose.
- *seccion*: que define la sección del caso de uso que esté en ejecución.

Un ejemplo de ello lo muestra el siguiente fragmento de código, que garantiza estos valores por defecto en dependencia del caso de uso que se esté ejecutando:

```
Map valores() {
    def map = [:]
    switch (cu) {
        case "proceso por el que cumple":
            switch (seccion) {
                case 1: //Sección 1: "Registrar documentos del revocado de TCSI"
                case 9: //Sección 9: "Registrar documentos del revocado de remisión condicional de la sanción"
                case 7: //Sección 7: "Registrar documentos de revocado de limitación de libertad"
                case 10: // Sección 10: "Registrar documentos del revocado de TCCI"
                    map["tipoSancion"] = NomTipoSancion.get(3) //Privación de libertad temporal
                    break
                case 6: //Sección 6: "Registrar documentos de TCCI"
                    map["apartado"] = NomApartado.get(3) // apartado 3
                    map["tipoSancion"] = NomTipoSancion.get(4) // TCCI: Trabajo correccional con internamiento
                    break
                case 8: //Sección 8: "Registrar documentos de un ingreso por cumplimiento de multa"
                    map["apartado"] = NomApartado.get(5) // apartado 5
                    map["tipoSancion"] = NomTipoSancion.get(6) // Multa
                    break
            }
            break
        case "actualizar proceso por el que no cumple":
            switch (seccion) {
```

Figura 21: Fragmento de código de la clase *DescripcionCUService.groovy*

3.2.2.3 El ingreso

El componente Ingreso es el encargado de registrar a un interno por primera vez en el sistema.

A continuación se describe el funcionamiento de este componente (Figura 22):

1. El controlador de ingreso muestra la interfaz web que contiene los datos básicos del interno, del Expediente Legal, del ingreso y del oficial que conduce al interno al centro penitenciario.

2. Los datos son procesados por el controlador *IngresoSitLegalController.groovy*, quien se apoya en *IngresoSitLegalService.groovy* para guardar toda la información en memoria.
3. *GeneralesController.groovy* usando métodos descritos en *GeneralesService.groovy,* construye una lista de documentos que deben mostrarse al usuario y muestra un formulario con los datos generales de la situación legal del interno.
4. El servicio *DocumentosService.groovy* obtiene el resultado anterior e inicializa el Recolector de documentos adicionando los documentos que serán visualizados a la lista *selected* del servicio descrito en la sección 3.2.2.1.
5. El Recolector de documentos almacena los datos de los documentos introducidos por el usuario.
6. El Recolector de documentos dirige el flujo para el controlador principal del componente Ingreso.
7. El componente Ingreso registra un nuevo interno en el sistema y en dependencia de los documentos, los datos de los documentos y el motivo de ingreso le asigna un estatus legal.

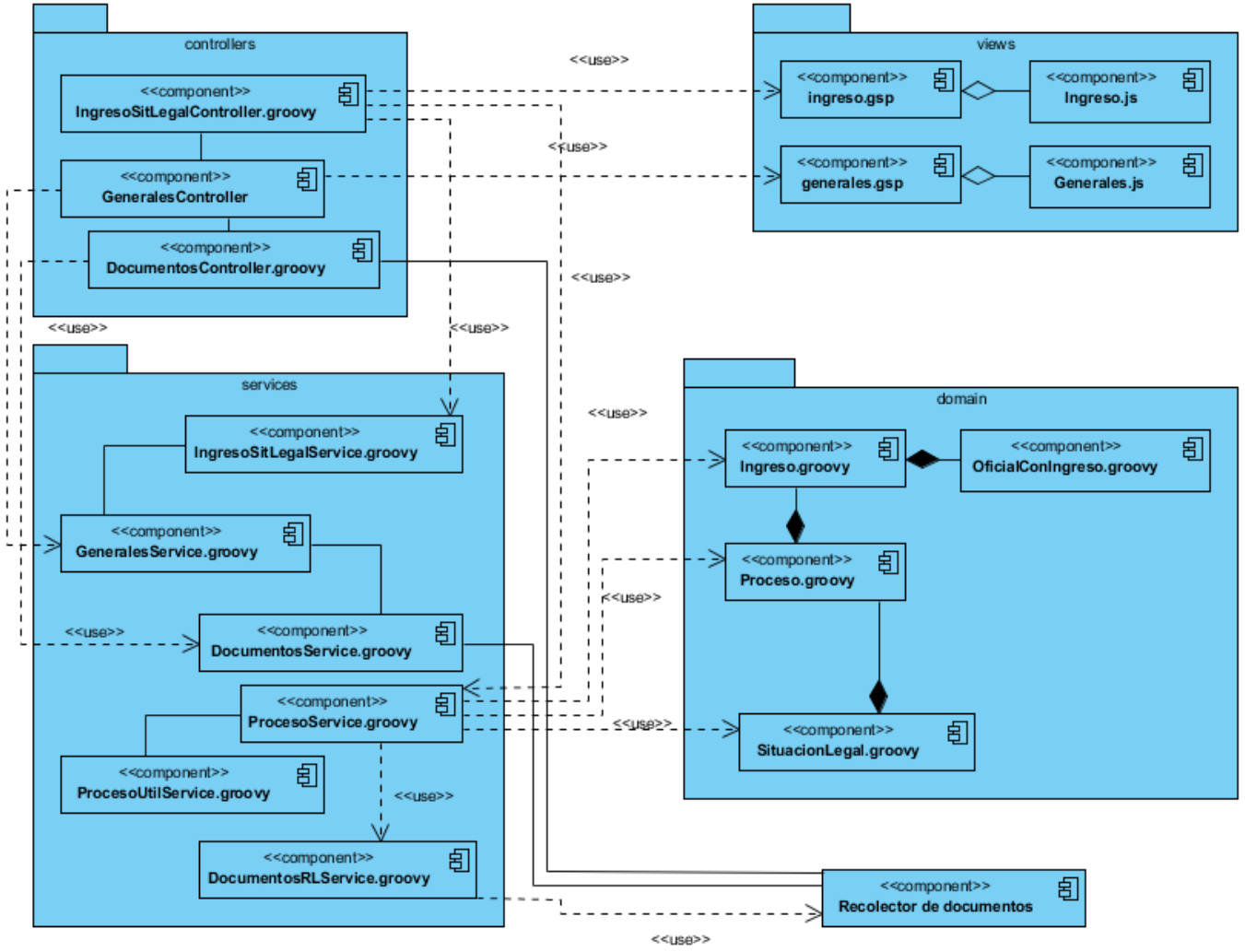

Figura 22: Componente Ingreso

En el paso 7, se le asigna un estatus legal al interno registrado en el sistema. Este es el paso base para garantizar las conversiones de la situación legal: estar registrado en el sistema, tener una situación y un estatus legales, un proceso por el cual estar cumpliendo dentro de prisión y documentos que respaldan este proceso. El mismo está descrito en la clase *ProcesoService.groovy*:

```
if (ingreso) {
    switch (ingreso.motivoIngreso.id) {
        case 9: //Revocacion del TCSI
            proceso = guardarProceso(1) // 1: Cumpliendo
            expediente = registrarDocumentosDelRevocadoDeTCSI(proceso)
            break
        case 3:  //Acusado de Fiscalia
        case 8:  //acusado de fiscalia militar
            proceso = guardarProceso(2) // 2: Pendiente
            expediente = registrarDocumentosDeAcusadoDeFiscalia(proceso)
            break
        case 28: // Asegurado de Tribunal Municipal
        case 6: // Asegurado por Consejo Menores
            proceso = guardarProceso(1) // 1: Cumpliendo
            expediente = registrarDocumentosDeAsegurado(proceso)
            break
        case 5: //  Sancionado de Tribunal Municipal
        case 4: // Sancionado de Tribunal Provincial
        case 10: // Sancionado por Tribunal Militar
            proceso = guardarProceso(1) // 1: Cumpliendo
            expediente = registrarDocumentosDeSancionado(proceso)
            break
        case 17: //TCCI
```

Figura 23: Ejecutando las secciones del caso de uso Registrar proceso por el que cumple para asignar un estatus legal

En la siguiente figura se muestra el código de la asignación de estatus legales para el caso que se esté ingresando un Acusado de Tribunal:

```groovy
/** *
 * @description: Registrar documentos de Acusado de Tribunal
 */
def registrarDocumentosDeAcusadoDeTribunal(def proceso) {
    def mandamiento = proceso.obtenerDocumento(NomTipoDocumento.get(4)) // madamiento
    if (mandamiento) {
        def expediente
        if (mandamiento.apartado.id == 1 || mandamiento.apartado.id == 4) {
            expediente = setAndCreateEstatus(proceso, "Acusado Pendiente a Juicio")
        }
        if (mandamiento.apartado.id == 2) {
            expediente = setAndCreateEstatus(proceso, "Acusado Falto de Documentos")
        }
        return expediente
    }
}
```

Figura 24: Código fuente que finaliza el caso de uso Proceso por el que cumple cuando el interno ingresa como Acusado de Tribunal

Uno de los documentos que se le recogen al Acusado de Tribunal es el Mandamiento de Admisión. En el mismo se registra el apartado, y según este, si es el *1* o el *4* el sistema le asigna el estatus legal de *Acusado Pendiente a Juicio*, y si es el *2* se le asigna el estatus *Acusado Falto de Documentos*. Existen más apartados, pero para este caso específico el Recolector de documentos solo muestra el 1, el 2 y el 4. Para lograrlo, es necesario que las clases del Recolector de documentos tengan en cuenta esta situación. En la Figura 25 se muestran dos métodos implementados en *NuevoProcesoService.groovy* que devuelven todos los apartados, –el primer método– y los apartados específicos para el Acusado de Tribunal, –el segundo método–:

```groovy
/**
 * @description: obtener los apartados para mosrar en el mandamiento de Admisión
 */
def todosLosApartados() {
    NomApartado.list()
}

def apartadosAcusadoTribunal() {
    def lista = []
    lista << NomApartado.get(1)
    lista << NomApartado.get(2)
    lista << NomApartado.get(4)
    lista
}
```

Figura 25: Código fuente en *NuevoProcesoService.groovy* que devuelve los apartados

El controlador principal del Recolector de documentos se encarga de decidir a qué método invocar según la petición del usuario. El código de la Figura 26 pertenece a la implementación de la interfaz de usuario correspondiente a los datos del documento Mandamiento de Admisión. Muestra la utilización de la etiqueta *s:select* de la librería de etiquetas del SIDEP (*SidepTagLib.groovy*). Esta etiqueta está configurada para que ejecute la acción *losApartados* del controlador *NuevoProcesoController.groovy.*

De esta forma se delega la responsabilidad de decidir qué mostrar al controlador. Esta acción del controlador se encarga de descubrir qué es lo que está haciendo el usuario del sistema. El método *getQuien()* de *NuevoProcesoService.groovy* le informa al controlador quién solicitó mostrar la interfaz del Mandamiento de Admisión. En el código de la Figura 27 si la primera condición no se cumple indica que no se está ejecutando un ingreso y por ende el componente Ingreso no fue quien solicitó esta acción; entonces el controlador principal del Recolector de documentos le ordena enviar al componente de interfaz de usuario una lista con todos los apartados.

En caso de que la condición se cumpla, se hace necesario saber con qué objetivo la interfaz de usuario del Mandamiento de Admisión fue mostrada. El método *getSeccion()* de *NuevoProcesoService.groovy* devuelve un

número que representa la sección del caso de uso Registrar proceso por el que cumple; cuando este valor sea igual a *3* entonces el usuario que esté trabajando con el módulo se encuentra efectuando un ingreso de un interno cuyo motivo de ingreso es Acusado de Tribunal y el controlador le envía una lista con los apartados *1, 2* y *4* a la interfaz de usuario. En caso contrario enviará todos los apartados.

```
<s:select controller="nuevoProceso"
          readOnly="${documento?.apartado ? true: false}"
          req="true"
          action="losApartados"
          name="documento.apartado.id"/>
```

Figura 26: Código en la vista *mandamiento.gsp* del Recolector de documentos que consume la acción *losApartados* de *NuevoProcesoController.groovy*

```groovy
/**
 @description: Acción definida para el s:select de los apartados en el Mandamiento de Admisión.
 */
def losApartados = {
    def lista = []
    if (nuevoProcesoService.getQuien() == "ingresoDocumentos") {
        if (nuevoProcesoService.getSeccion() && nuevoProcesoService.getSeccion() == 3) {
            lista = nuevoProcesoService.apartadosAcusadoTribunal()
        } else {
            lista = nuevoProcesoService.todosLosApartados()
        }

    } else {
        lista = nuevoProcesoService.todosLosApartados()
    }
    list(lista)
}
/**
 * @description: convierte una lista a JSON.
 */
def list(lista) {
    def items = []
    items << [id: -1, descripcion: message(code: 'default.select.text')]
    lista.each {
        items << [id: it.id, descripcion: it.toString()]
    }
    render([identifier: "id", items: items] as JSON)
}
```

Figura 27: Código en *NuevoProcesoController.groovy* que convierte al formato JSON los apartados

El componente Reingreso presenta un funcionamiento similar: ingresa personas al sistema, pero esta vez, la misma debió haber estado en un lugar de internamiento y habérsele otorgado algún egreso para que como consecuencia se deshabilite su Expediente Legal en el sistema.

El reingreso puede ser de dos formas, por nuevo delito o por un proceso antiguo que estaba cumpliendo. Si el usuario selecciona la opción *Nuevo delito* el flujo de este componente se comportaría como el de ingreso, pero esta vez en el

paso 7, el sistema analiza la Situación Legal del interno y en dependencia de sus valores, le asigna un nuevo estatus legal y lo clasifica en régimen[2]. Si reingresa por el proceso por el que estaba cumpliendo, el componente configura el Recolector de documentos en función del motivo de reingreso del interno.

3.2.2.4 El componente Proceso

Formando parte del componente Situación Legal del módulo Expediente Legal, se encuentra el componente Proceso, encargado de actualizar cada uno de los procesos que sean registrados para el interno.

Un interno puede tener muchos procesos, pero uno solo cumpliendo. Este componente es capaz de detectar el proceso por el que el interno está cumpliendo. Esta identificación es importante, porque de acuerdo al tipo de proceso (por el que cumple o por el que no cumple), será su forma de ser actualizado. La identificación del tipo de proceso está escrita en el controlador *ActualizarProcesoController.groovy* del componente:

[2] La clasificación en régimen es una funcionalidad de otro módulo del sistema. En este caso, el módulo utiliza esa funcionalidad para dar automáticamente una clasificación al interno.

```
def decide = {
    if (!params.id) {
        redirect action: index
        return
    } else {
        Proceso proceso = Proceso.get(params.id.toString().toLong())
        if (!actualizarProcesoService.isProcesoPorElqueCumple(proceso)) {
            redirect controller: "actualizarProcesoNoCumple", params: [id: params.id]
            return
        } else {
            if (!actualizarProcesoService.tratadoComoProcesoConIngreso(proceso)) {
                redirect controller: "actualizarProcesoNoCumple", params: [id: params.id]
                return
            }
            redirect(controller: "actualizarProcesoConIngreso", params: [id: params.id])
            return
        }
    }
}
```

Figura 28: Forma de decidir cómo actualizar un proceso en el controlador principal de actualización del componente Proceso

Un proceso está cumpliendo cuando:

- Su estado tiene como valor “Cumpliendo” o “Total conjunta”, o
- Es único.

La forma de actualizar los procesos en dependencia de su tipo está descrita en los servicios del componente. Si el componente detecta que se actualizará un proceso y este proceso es por el cual ingresó el interno y por el que cumple, entonces desviará el flujo de actualización hacia el paquete *procesoConIngreso.implement* de los servicios, de otro modo, la lógica de negocio a aplicar sería la que se alberga en el paquete *noCumple.implement.*

Como se puede apreciar en el diagrama de componentes del módulo Expediente Legal de la sección 3.2.1, el componente Proceso usa el Recolector de documentos y así cumple con el objetivo de los casos de uso de actualización de procesos, que es, adicionarles documentos. En las clases

ActualizarProcesoConIngresoService.groovy y *ActualizarProcesoNoCumpleService.groovy* se inicializa el componente recolector en dependencia de los documentos que pueda admitir la actualización del proceso seleccionado. Al finalizar, uno de los servicios ubicados en los paquetes de la Figura 29, actualiza el proceso seleccionado.

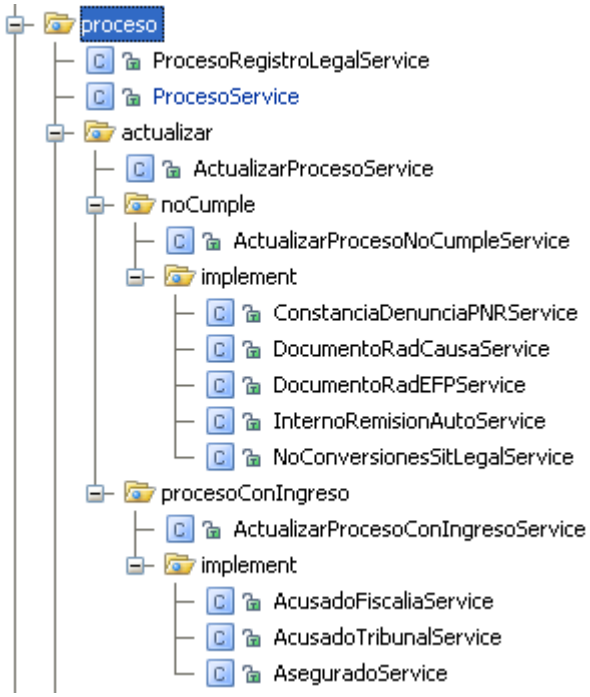

Figura 29: Distribución de la lógica del negocio para actualizar un proceso en el componente Proceso

3.2.3 Tratamiento de errores

Ningún sistema es perfecto, ni está excepto de fallos internos o externos a él. Es por ello que en el módulo Expediente Legal se definió una estrategia para evitar los errores de entrada de datos y tratar los deslices que no dependan de él, que contiene:

1. validación en el cliente,
2. validación en el servidor,

3. y tratamiento de excepciones. (Figura 30).

Los parámetros de entrada fueron validados en el cliente –con JavaScript– y en el servidor –con Groovy o usando los propios *constraints* de las clases de dominio–, con el objetivo de garantizar la integridad de los datos procesados.

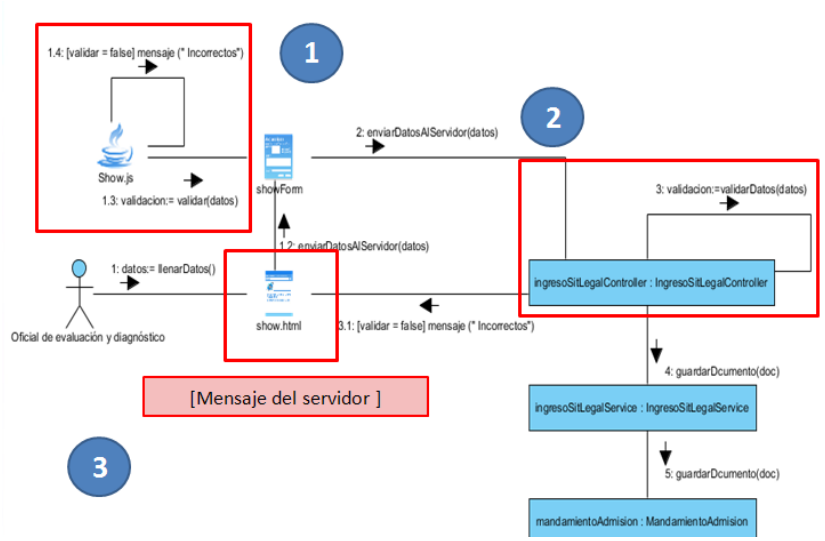

Figura 30: Tratamiento de errores del módulo Expediente Legal

Para la validación de los datos de entrada en el servidor se definieron clases servicio que serán invocados por el controlador que reciba estos parámetros. La nomenclatura de estas clases se compone por el *<nombre de la funcionalidad que estará validando>ValidatorService.groovy* e implementan la interfaz definida en la siguiente figura:

```
interface SituacionLegalValidatorInterfaceService {

    /**
     * devuelve una lista de errores
     * */
    List validate(def params)

}
```

Figura 31: Interfaz que implementan los servicios de validación en el módulo Expediente Legal

Los *constraints* son invocados antes de guardar mediante el método *validate*(). Constituyen la garantía de que los datos que se van a persistir no contienen errores. En la siguiente figura se muestra la definición del *constraints* de la clase del dominio correspondientes a los datos del Interno:

```
static constraints = {
    personasabrigo(nullable: true)
    fechanacimiento(nullable: true)
    edaddelito(nullable: true)
    pais(nullable: true)
    estadoCivil(nullable: true)
    tipoDocumento(nullable: true)
    provinciaNacimiento(nullable: true)
    municipioNacimiento nullable: true
    religion(nullable: true)
    fisionomia(nullable: true)
    resenna(nullable: true)
    otrosRasgos(nullable: true)
    nivelEscolaridad(nullable: true)
    nacionalidad(nullable: true)
    organizaciones(nullable: true)
    numeroid size: 0..11, validator: {val, obj ->
        try {
            if (obj.tipoDocumento.id == 3 || obj.tipoDocumento.id == 5) {
                if (!(val?.length() == 11)) return ["size"]
                if (!val.isNumber()) return ["format.number"]

                new SimpleDateFormat("yyMMdd").parse(val.substring(0, 6))

                if (!(val.substring(6, 11) == "00000")) {
                    return ["format.end.cero"]
                }
                boolean masculino = val.charAt(10).toString().toInteger() % 2 == 0;
                if ((masculino && !obj.sexo) || (!masculino && obj.sexo)) {
                    return ["caracter.sex"]
                }
            } else {
                def diferencia = (new Date().getYear()) - (obj.fechanacimiento.getYear());
                if (diferencia < 17) {
                    return ["and.fechanacimiento"]
                }
            }
        } catch (Exception e) {
            return ["format.message.error"]
        }
    }
}
```

Figura 32: Constraints de la clase Interno

Existen situaciones no deseadas inherentes al módulo que pudieran suceder. El módulo debe ser capaz de saber cuándo ocurren y qué hacer en caso de ocurrencia. Es por ello que todas las acciones con alto riesgo de errores internos o externos fueron tratadas como se muestra en la siguiente figura:

```
def mostrarQuebrantamientos = {
    try {
        if (!quebrantamientosSancionService.idSitLegal) {
            redirect controller: "procesoRegistroLegal", action: "check"
            return
        }
        render view: "${urlbase}/quebrantamientos", model: [interno: quebrantamie
    } catch (SituacionLegalException s) {
        flash.errorMessage = s.getMessage()
        render view: "${urlbase}/quebrantamientos"
        return
    } catch (Exception e) {
        flash.errorMessage = "${message(code: "error.desconocido")}"
        render view: "${urlbase}/quebrantamientos"
        return
    }
}
```

Figura 33: Try-catch implementado en las acciones con riesgo ocurrencia de errores en el servidor

Cuando el error es externo al módulo, se captura como una *Exception*, informándole al usuario de la ocurrencia de un error externo al módulo. En caso de que sea una violación de las funcionalidades implementadas, entonces se le informa al usuario capturando y mostrando la excepción del módulo a través de la clase *SituacionLegalException.groovy.* En este último caso, el sistema deshace todos los cambios que hasta el momento del error pudieron haber sucedido en la base de datos.

3.2.4 Diagrama de despliegue

"Los diagramas de despliegue muestran a los nodos procesadores, la distribución de los procesos y de los componentes." (47) Define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software.

Como se pudo conocer en la sección 3.2.1 Diagrama de Componentes, el módulo Expediente Legal está compuesto por tres submódulos: Ingreso, Reingreso y Situación Legal. El despliegue de la solución implementada se realizó según la distribución y características de la fuerza de trabajo de la Dirección de Establecimientos Penitenciarios. El Sistema Penitenciario Cubano en su estructura, cuenta con dos tipos de centros penitenciarios (Figura 34): receptores y prisiones no receptores[3].

El módulo Expediente Legal se desplegaría de la siguiente forma:

- Centros penitenciarios receptores: Componentes Ingreso, Reingreso y Situación Legal.
- Centros penitenciarios no receptoras: Componente Situación Legal.

[3] Los centros penitenciarios receptores son las destinados a efectuar el ingreso en el Sistema Penitenciario Cubano. Después de un proceso de clasificación del interno, una comisión se encarga de darle ubicación en un centro penitenciario no receptor, donde cumplirá su sanción.

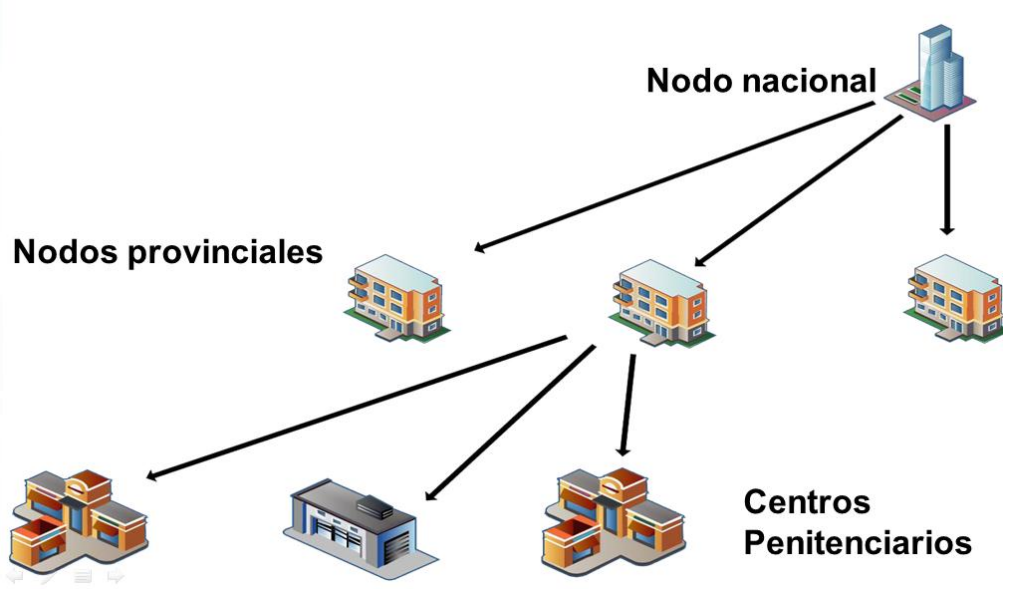

Figura 34: Estructura del Sistema Penitenciario Cubano

En ambos tipos de centro penitenciario el despliegue se realizará según el siguiente diagrama:

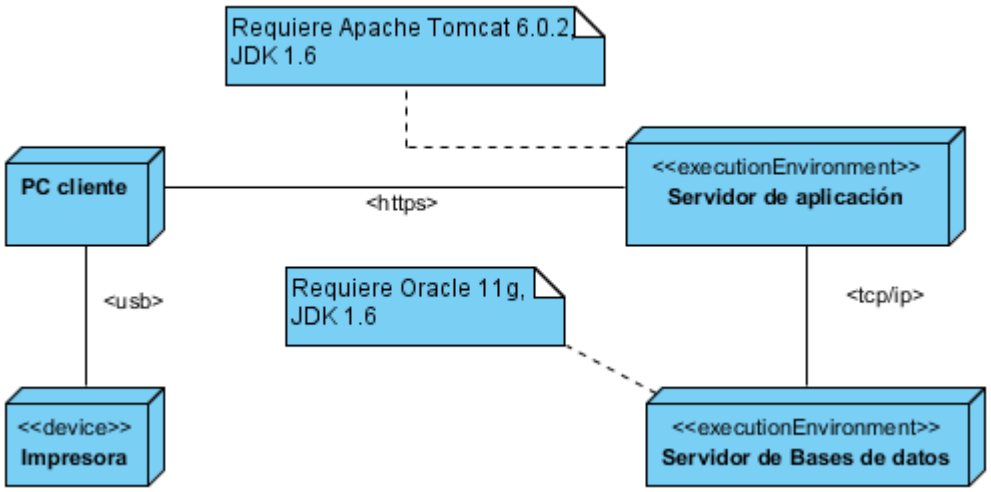

Figura 35: Diagrama de despliegue para el módulo Expediente Legal.

3.3 Pruebas

Las pruebas de software tienen como objetivo proporcionar información sobre la calidad del producto bajo pruebas a la parte interesada. Se centran principalmente en la evaluación de la calidad del producto. (50)

Las pruebas realizadas al módulo se hicieron con el objetivo de encontrar el mayor número de errores en la implementación, de poner en evidencia la solución, de probar todas las funcionalidades descritas en los casos de uso, de no convencerse de que el módulo funcione bien, sino de buscar y encontrar errores. (50)

Para cumplir con el objetivo planteado se procedió a aplicar las pruebas usando el método de caja negra el cual se centra en los requisitos funcionales del software. Este método intenta descubrir errores, tales como (51):

- Funciones incorrecta o ausentes
- Errores de interfaz
- Errores en la estructura de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y de terminación.

Los métodos de caja negra verifican las especificaciones funcionales y no consideran la estructura interna del programa. Se confeccionan sin el conocimiento interno del producto. No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados. (50)

La técnica de pruebas seleccionada fue el diseño de casos de pruebas con el objetivo de probar las funcionalidades del módulo, fijando su atención en la validación de las funciones, métodos, servicios, y descripciones de caso de uso específicamente.

El nivel de prueba seleccionado fue prueba de desarrollador. Este nivel de prueba indica los aspectos de diseño e implementación de las pruebas más adecuadas 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 de 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. (50)

La siguiente tabla muestra un resumen de las no conformidades detectadas en cada iteración de las pruebas realizadas al módulo.

Iteración/ Componente	Proceso	Ingreso	Reingreso	Total
Iteración 1	20	23	10	53
Iteración 2	7	3	4	14
Iteración 3	0	0	0	0
Total	27	26	14	67

3.4 Conclusiones

En este capítulo se obtuvo como resultado los diagramas de componentes y de despliegue del módulo Expediente Legal. El funcionamiento de los componentes del módulo fue explicado utilizando como base el caso de uso Registrar proceso por el que cumple. Se definió y se implementó una política de tratamiento de errores que respalda el funcionamiento del módulo ante el suceso de los mismos. Por último, se aplicaron pruebas de calidad al módulo utilizando la estrategia definida, pudiendo eliminar la aparición de no conformidades.

Conclusiones

Como resultado de esta investigación se obtuvo el módulo Expediente Legal, encargado de controlar la situación legal de los internos en el Sistema Penitenciario Cubano a partir de la gestión de las conversiones en el Expediente Legal, logrando la asignación de estatus legal automática al interno, la generación automática de los principales trámites legales pendientes, el registro de todos los datos de los documentos para la posterior asignación de estatus y evita la duplicación de expedientes. Como parte de la solución:

- Se realizó un estudio de soluciones nacionales e internacionales, en búsqueda de un software que cumpliera con los requisitos requeridos por el Sistema Penitenciario Cubano, resultando necesaria la implementación del nuevo módulo.
- Se analizaron las herramientas, lenguajes y la metodología de desarrollo para el trabajo en el módulo destacando las características que las hacen factibles para la propuesta de solución.
- Se diseñó el módulo, obteniéndose los diagramas de paquetes, de clases, de secuencia y de estado que facilitaron la implementación del mismo.
- Se diseñó y construyó la base de datos del módulo para la persistencia de los datos.
- Se implementó el módulo Expediente Legal compuesto por tres submódulos: Ingreso, Reingreso y Situación Legal, que garantizan las conversiones de la situación legal a través de su integración.
- Se obtuvieron los diagramas de componentes que muestran la distribución física de los componentes del módulo.
- Se aplicaron pruebas de calidad, probando las funcionalidades del módulo hasta eliminar la aparición de no conformidades.

De esta forma se le da cumplimiento al objetivo general demostrando el desarrollo del módulo Expediente Legal del subsistema Registro Legal del SIDEP permite la gestión de las conversiones de la Situación Legal de los internos.

Recomendaciones

Se recomienda:

- Diseñar una solución para la comunicación con los sistemas informáticos de la Dirección Nacional de Identificación e Inmigración y Extranjería para garantizar la correcta identificación de los internos al ingresar al Sistema Penitenciario Cubano.
- Implementar una solución para configurar la aplicación de la lógica del negocio, que se centre en:
    - La asignación de estatus legal al interno en dependencia de los datos de los documentos en el ingreso.
    - La modificación del estatus legal del interno en dependencia de la adición o actualización de los procesos.
    - La creación de nuevos tipos de documentos y sus datos.
    - La relación de los motivos de egreso con los motivos de reingreso, así como los documentos por motivo de reingreso.
- Diseñar la integración con los sistemas informáticos de los órganos de justicia penal y órganos de instrucción para la resolución de los trámites legales pendientes.

Referencias bibliográficas

1. **Gómez Llano, José Antonio y Consuegra Peña, Yohairo Benito.** *Diseño e Implementación del módulo Situación Legal del Sistema Penitenciario Nacional. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de La Habana : s.n., 2010.

2. **Batista, Yanet del Risco.** 0208_Proyecto Técnico Prisiones Cuba.

3. **Hernández León, Rolando Alfredo y Coello González, Sayda .** *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.* Ciudad de la Habana : EDUNIV Editorial universitaria, 2002. ISBN: 959-16-0343-6.

4. **Lorenzo, Yanay Viera.** 5107_Glosario de Términos. Prisiones Cuba.

5. PROCEDIMIENTOS DE TRABAJO DE LA ESPECIALIDAD CONTROL PENAL. *I. DEL NIVELDE CENTRO PENITENCIARIO, TRABAJO Y ESTUDIO Y PRISIONES SANATORIO DEL VIH-SIDA.*

6. *PROCEDIMIENTOS DE TRABAJO DE REGISTRO LEGAL.* [PDF]

7. **Lorenzo, Yanay Viera.** 0114_Especificación de casos de uso. Categorías.

8. —. 0114_Especificación de casos de uso. 2011.

9. **Gongora Rodríguez, Alien y Pupo Diéguez, Yordanys.** Diseño e implementación de la Capa de Presentación de las funcionalidades relacionadas con el registro de decisiones y la ejecución de la pena del módulo Situación Jurídica. Ciudad de la Habana : s.n., 2008.

10. **INPE, Oficina de Sistemas del.** Instituto Nacional Penitenciario. [En línea] Julio de 2009. [Citado el: 30 de noviembre de 2011.] http://www.inpe.gob.pe/contenidos.php?id=478&np=352&direccion=1.

11. **Instituto Nacional Penitenciario. Dirección de Registro Penitenciario.** *Manual de Procedimientos y Actividades de Registro Penitencairio del INPE.* [pdf] 2008.

12. **Motorola Solutions.** Offendertrak - Motorola Solutions USA. *Offendertrak.* [En línea] 2012. [Citado el: 3 de febrero de 2012.] http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/Public+Sector+Applications/Public+Safety+Applications/Jail+Management+Applications/Offendertrak_US-EN.

13. **Corrections.com.** Offendertrak Corrections Management System. [En línea] Corrections Media, 18 de enero de 2001. [Citado el: 15 de abril de 2012.] www.corrections.com/articles/7279.

14. **PR Newswire Association LLC.** Motorola to Provide Corrections Management System to State of Mississippi Department of Corrections (DOC). [En línea] 2012. [Citado el: 3 de febrero de 2012.] http://www.prnewswire.com/news-releases/motorola-to-provide-corrections-management-system-to-state-of-mississippi-department-of-corrections-doc-73959797.html.

15. Real academia Española. *Diccionario de la Lengua Española - Vigésima segunda edición.* [En línea] 18 de enero de 2012. http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=informaci%C3%B3n.

16. La Ingeniería de Software / Software Engineering. [En línea] 2012. [Citado el: 16 de enero de 2012.] http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#IngSoft.

17. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. 84-7829-036-2.

18. **IBM Corp.** Rational Unified Process. *Classic RUP for SOMA.* [En línea] IBM Corp, 2006. [Citado el: 17 de febrero de 2012.] http://10.5.5.199:8086/RUP/.

19. **Instituto Tecnológico Superior de Calkiní en el Estado de Campeche.** ITESCAM. [En línea] 2012. [Citado el: 21 de enero de 2012.] www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r19670.DOC.

20. **Visual Paradigm.** UML, BPMN and Database Tool for Software Development. [En línea] 2012. [Citado el: 21 de enero de 2012.] http://www.visual-paradigm.com/.

21. **Free download manager - Sitio de descargas de software.** Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) por Visual Paradigm International Ltd. - reporte y descarga. [En línea] [Citado el: 21 de enero de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

22. **Dat@Market Solutions.** Embarcadero ER/Studio, Software - Herramientas de Desarrollo de Software. [En línea] 2011. [Citado el: 22 de enero de 2012.] http://www.guiadesolucionestic.com/software-del-sistema/herramientas-de-desarrollo/herramientas-de-desarrollo-de-software/972-embarcadero-erstudio.

23. **Computación Avanzada y Sistemas Empresariales, S.A. de C.V.** Embarcadero ER/Studio. [En línea] [Citado el: 22 de enero de 2012.] http://www.casenet.com.mx/index.php?option=com_content&view=article&id=58&Itemid=59.

24. **SpringSource.** SpringSource. [En línea] 2011. [Citado el: 28 de 11 de 2011.] http://www.springsource.com/developer/sts.

25. SpringSource. [En línea] 2012. [Citado el: 2 de febrero de 2012.] http://www.springsource.org/sts.

26. **Apache Software Foundation.** Apache Tomcat. [En línea] 1999-2011. [Citado el: 28 de 11 de 2011.] http://tomcat.apache.org.

27. **VISUALSVN SERVER.** VISUALSVN SERVER // Subversion Server for Windows . [En línea] 2012. [Citado el: 21 de enero de 2012.] http://www.visualsvn.com/server/.

28. **CollabNet.** tortoisesvn.tigris.org. [En línea] 2009. [Citado el: 4 de febrero de 2012.] http://tortoisesvn.tigris.org/.

29. **Mora, Roberto Canales.** Introducción a TortoiseSVN. [En línea] 2012. [Citado el: 4 de febrero de 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=tortoisesvn.

30. **Gartner, Inc. and/or its Affiliates.** Technology Research | Gartner Inc. [En línea] 2012. [Citado el: 2 de febrero de 2012.] http://www.gartner.com/technology/home.jsp.

31. Oracle. [En línea] 2012. [Citado el: 2 de febrero de 2012.] http://www.oracle.com/lad/products/database/index.html.

32. **Linea de Codigo.** Hola Mundo en DOJO. [En línea] [Citado el: 2 de febrero de 2012.] http://lineadecodigo.com/dojo/hola-mundo-en-dojo/.

33. **Leon, Eduardo.** *Tutorial Visual Paradigm for UML.*

34. **FullBook.** Programación: html, javascript, xhtml, ajax, php. [En línea] 2012. [Citado el: 2 de febero de 2012.] http://www.fullbook.com.ar/stuff/otros_temas_de_desarollo_web/programacion_html_javascript_xhtml_ajax_php/44-1-0-425.

35. **Escuela de Groovy y Grails.** ¿Groovy? ¿Grails? - Escuela de Groovy. [En línea] [Citado el: 23 de enero de 2012.] http://www.escueladegroovy.com/informacion/groovy-grails.

36. **Oracle.** ¿Qué es Java y por qué lo necesito? *Java.* [En línea] [Citado el: 27 de enero de 2012.] http://www.java.com/es/download/faq/whatis_java.xml.

37. —. Conozca más sobre la tecnología Java. *Java.* [En línea] [Citado el: 27 de enero de 2011.] http://www.java.com/es/about/.

38. Conceptos de HTML. [En línea] [Citado el: 1 de febero de 2012.] http://www.htmlya.com.ar/temarios/descripcion.php?cod=68&punto=1.

39. **aulaClic.** ¿Qué es el SQL? . *Tema 1. Introducción.* [En línea] 2001. [Citado el: 28 de enero de 2011.] http://www.aulaclic.es/sql/t_1_1.htm.

40. **Ayuntamiento de Vitoria-Gasteiz.** Blog de Tecnologías. *GORM.* [En línea] 24 de mayo de 2010. [Citado el: 14 de noviembre de 2011.] http://blogs.vitoria-gasteiz.org/ti/tag/grails-gorm-persistencia-hibernate/.

41. **Lorenzo, Yanay Viera.** 0114_Especificación de casos de uso. Situación Legal.

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

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

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

45. **SpringSource, a division of VMware.** Groovy and Grails - Java | SpringSource. *Tools and Frameworks Developers Love.* [En línea] 2012. [Citado el: 24 de noviembre de 2011.] http://www.springsource.com/developer/grails.

46. **Sparx Systems Pty Ltd.** Sparx Systems - Tutorial UML 2 - Diagrama de Paquete. [En línea] [Citado el: 16 de febrero de 2012.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html.

47. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN: 970-17-0261-1.

48. **Blaha, Michael.** *Patterns Of Data Modeling.* s.l. : Taylor & Francis. ISBN: 9781439819890.

49. **SpringSource, a division of VMware.** Spring Web Flow | SpringSource.org. [En línea] 2012. [Citado el: 6 de diciembre de 2011.] http://www.springsource.org/spring-web-flow.

50. **DEPARTAMENTO DE INGENIERÍA Y GESTIÓN DE SOFTWARE.** Entorno virual de aprendizaje del Graduado UCI. [En línea] 2012. [Citado el: 16 de febrero de 2012.] http://evapostgrado.uci.cu/file.php/368/Bibliografia/Conf_FT_Prueba_06-07.pdf.

51. Ingeniería de Software II JOBP. [En línea] [Citado el: 4 de mayo de 2012.] http://sites.google.com/site/ingenieriadesoftwareiijobp/pruebas-de-caja.

52. **Dos ideas.** Los lenguajes específicos de dominio. [En línea] [Citado el: 27 de enero de 2012.] http://www.dosideas.com/noticias/actualidad/487-los-lenguajes-especificos-de-dominio.html.

53. **Masadelante.com.** ¿Qué es un navegador, explorador o buscador? [En línea] 2012. [Citado el: 14 de abril de 2012.] http://www.masadelante.com/faqs/que-es-un-navegador.

54. **Programación en castellano.** Conceptos básicos de ORM (Object Relational Mapping). Programación en Castellano. [En línea] 2011. [Citado el: 29 de enero de 2012.] http://www.programacion.com/articulo/conceptos_basicos_de_orm_object_relational_mapping_349.

55. Tutoriales de Programacion Java: Spring 3 - Parte 1: Introducción. [En línea] 2 de septiembre de 2010. [Citado el: 28 de enero de 2012.] http://www.javatutoriales.com/2010/09/spring-parte-1-introduccion.html.

56. **Rational Software Corporation.** *RUP. "Rational Unified Process".* 2003.

57. **Aguilar Bello, Marisol, y otros, y otros.** Diagrama de clases | ingeniería de Software. *Desarrollo de un software para la sistematizacion de la información del centro de conciliación de la Fundación Tecnológica Antonio de ArévaloTecnar.* [En línea] 12 de febrero de 2012. [Citado el: 16 de febrero de 2012.] http://frank0110.blogspot.com/2012/04/diagrama-de-clase.html.

Glosario de términos

Acusado FD: Acusado falto de documentos.

Acusado PJ: Acusado Pendiente a Juicio.

AJAX: Asynchronous JavaScript And XML .

DNI: Dirección Nacional de Identificación.

EFP: Expediente en Fase Preparatoria.

Hibernate: Es una marco de trabajo de mapeo objeto-relacional para entornos Java. No sólo se ocupa del mapeo desde las clases Java a las tablas de las bases de datos (y desde los tipos de datos de Java a los tipos de datos de SQL), sino que también facilita la consulta y recuperación de datos.

HTML: HyperText Markup Language.

Lenguaje de dominio específico: En el desarrollo de software, un lenguajes específico de dominio (domain-specific language - DSL) es un lenguaje de programación dedicado a un problema de dominio en particular, o una técnica de representación o resolución de problemas específica. (52)

Lugar de internamiento: Centro penitenciario.

Navegador (web): Programa que permite visualizar páginas web en la red además de acceder a otros recursos, documentos almacenados y guardar información. (53)

ORM: Técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de nuestra aplicación. (54)

Población penal: Conjunto de internos.

RUP: Rational Unified Process

Spring: Marco de trabajo de java que gestiona el trabajo de las capas en aplicaciones de java, provee desde plantillas para trabajar con JDBC o invocación de Web Services y JMS, pasando por sus propias soluciones, ORM o MVC (web), hasta integración con otros frameworks, como Struts 2, Hibernate, JSF, etc., haciendo uso de muchos buenos principios de programación. (55)

TCCI: Trabajo Correccional Con Internamiento.

TCSI: Trabajo Correccional Sin Internamiento.

Transición: Relación entre dos estados; indica que, cuando ocurre un evento, el objeto pasa del estado anterior al siguiente. (47)

UML: Unified Modeling Language.