Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Desarrollo de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP.

AUTOR: Carlos Naya Milian

TUTORA: Ing. Linet Lores Sánchez

CO – TUTOR: Ing. Dasiel Cordero Morales.

La Habana, 2012.

Año del 53 Aniversario de la Revolución.

"(…) aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos…"

Ernesto “Che” Guevara

Declaración de autoría

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

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

Carlos Naya Milian

_____

Firma del autor

Ing. Linet Lores Sánchez

_____

Firma de la tutora

Ing. Dasiel Cordero Morales

_____

Firma del co-tutor

Datos del contacto

Tutora: Ing. Linet Lores Sánchez.
Correo Electrónico: linetls@uci.cu

Co-tutor: Ing. Dasiel Cordero Morales.
Correo Electrónico: dcordero@uci.cu

Autor: Carlos Naya Milian.
Correo electrónico: cnaya@estudiantes.uci.cu

Dedicatoria

A mi mamá Teresita que ha bendecido siempre mis decisiones y ha sido mi guía todos estos años, sin su ternura, apoyo y dedicación no hubiese podido ser la persona que soy hoy, todo lo que he logrado en la vida te lo debo mamá, te quiero mucho.

A mi abuela Gloria que aunque no se encuentre físicamente siempre tendrá un espacio dentro de mi corazón.

A mis abuelos Estela y Benito por regalarme momentos de felicidad, amor y comprensión.

A mi papá Carlos por ser más que un padre, un gran amigo, por inspirarme amor, bondad, honestidad e incontables virtudes que ameritan que él sea un ejemplo a seguir para mí.

A mi hermanita Ily, espero que este trabajo te sea de inspiración, te quiero muchísimo.

A mis amistades por haber estado junto a mí durante todo este tiempo, agradecer los buenos momentos que he compartido junto a ellos, sus sabios consejos y compañía. No quiero mencionar nombres porque podría faltarme alguno y no me gustaría eso, así que a “todos” les dedico este trabajo.

A todos mis compañeros de aula con los que he permanecido los cinco años de la carrera y he tenido el regocijo y la satisfacción de compartir buenos momentos.

A mis compañeros del proyecto que han permanecido siempre a mi lado.

A todos los profesores que he tenido el honor y la dicha de conocer.

A la Revolución Cubana y a nuestro Comandante en Jefe Fidel Castro por ser los principales autores de que hoy tengamos una Universidad tan bella como es la Universidad de las Ciencias Informáticas para el desarrollo y el conocimiento de los jóvenes cubanos y otorgar mi más profundo agradecimiento por permitirme estudiar en tan prestigioso centro.

A todos los usuarios de la Dirección de Establecimientos Penitenciarios que utilizarán la aplicación.

Agradecimientos

A mi tutora Linet Lores Sánchez y a mi co-tutor Dasiel Cordero Morales, por haber estado presentes cada vez que los he necesitado, por sus consejos, preocupación y seguimiento durante el desarrollo de este Trabajo de Diploma.

A mi profesora Liset Schery Sánchez, una mujer muy especial que me brindó siempre su asistencia y su inmenso conocimiento con una gentil sonrisa.

Especial gratitud a la profesora Yanay Viera Lorenzo por su colaboración y buenos consejos para la realización de este trabajo.

A mis compañeros y amigos que juntos hemos transitado por este camino y que de una forma u otra, me han ayudado en la realización de este trabajo.

A mis compañeros del Proyecto Prisiones Cuba por poder contar con ellos para cualquier inquietud.

A todos los que de una manera u otra me brindaron su ayuda, tiempo y conocimiento.

Resumen

Actualmente en el sistema penitenciario cubano existen tres sistemas informáticos para gestionar la información: Sistema Automatizado para el Control del Recluso (SACORE), Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios (SACDEP) y Sistema de Automatización de Incidencias de la Dirección de Establecimientos Penitenciarios (SAIDEP). Estos sistemas no manejan la información de los procesos de ocupaciones, decomisos y hallazgos; además, el control físico de los internos, el control de acceso de los vehículos y el personal, así como la gestión de los medios técnicos son controlados manualmente, lo que provoca que hayan actividades que no se lleven a cabo, que la información no se gestione correctamente o que sufra pérdidas a lo largo del flujo de trabajo en Prisiones.

El presente trabajo tiene como objetivo el desarrollo de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP), los cuales registran los datos de los medios técnicos y del armamento empleado por los oficiales, los vehículos que acceden al centro, los planes y recuentos que se le realizan al interno así como las ocupaciones, decomisos y hallazgos que se le realicen a este o a sus familiares. Como parte de la solución se generarán los diferentes artefactos propios de las fases de diseño e implementación, teniendo como punto de partida el análisis de los requisitos de software establecidos de acuerdo con el cliente y haciendo uso de las tecnologías y herramientas definidas en la arquitectura del proyecto.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

En el año 1989 comienza en Cuba el sistema de informatización de los centros penitenciarios, con la digitalización de los datos principales del recluso y ciertos aspectos de control penal. Pero no es hasta algunos años más tarde que a raíz del cumplimiento de la orden 43/99 del Vice Ministro Primero se crea un Sistema Automatizado para el Control del Recluso, SACORE por sus siglas, el cual comenzó a dar respuesta en gran medida a muchos de los problemas existentes en el momento. El sistema culminó su desarrollo a finales del 2002 y se empezó a poner en marcha a principios del 2003 y cuenta en la actualidad con tres módulos principales: Control Penal, Reeducación Penal y el Orden Interior. Dichos módulos permiten gestionar los datos principales del recluso y los aspectos de control penal existentes.

A pesar de las facilidades que ha brindado el SACORE durante sus años de uso, aún cuenta con funcionalidades incompletas o pendientes de dichos módulos, por no contar la institución con el tiempo para su análisis, diseño y programación, y por carecer de las condiciones tecnológicas para su implantación posterior. A la par con este sistema en los centros penitenciarios se utilizan otros dos sistemas como son el Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios (SACDEP) y el Sistema de Automatización de Incidencias de la Dirección de Establecimientos Penitenciarios (SAIDEP).

A partir del Plan 20 x 50 la Universidad de las Ciencias Informáticas lleva acabo la modernización informática del Sistema Penitenciario Cubano a través del Sistema Informativo de la Dirección de Establecimientos Penitenciario (SIDEP), siguiendo al pie de la letra las disposiciones legales del MININT, los Órganos de Justicia Penal y el Estado Cubano. Este proyecto integraría de forma única los tres sistemas que existen en estos momentos en la Dirección de Establecimientos Penitenciarios, SACORE, SACDEP y SAIDEP, quedando conformado por siete subsistemas.

Actualmente el órgano Prisiones no cuenta con un sistema que permita llevar el inventario del equipamiento y armamento, el control de acceso a los centros penitenciarios, el control físico de los internos así como las ocupaciones, decomisos y hallazgos que tienen lugar en el Sistema Penitenciario Cubano. Por lo que estos procesos aún se realizan de manera manual y no se guardan

en formato digital, lo cual puede traer consigo que la información no esté correctamente organizada, posea errores o se extravíe. Sumado a esto, actualmente:

- No se le da seguimiento al Plan de recuentos físicos y sorpresivos que se elabora en los centros penitenciarios por lo que los recuentos no se realizan controladamente.
- El control de acceso se registra en el Libro de ocurrencias lo cual imposibilita conocer en tiempo rápido, por parte de la dirección, los vehículos y personal que accede a los centros penitenciarios.
- No hay respuesta rápida a los problemas encontrados en los recuentos físicos, sorpresivos u ordinarios provocando alteraciones en la vida cotidiana del centro penitenciario.
- No hay un control de la entrega de las ocupaciones realizadas a los familiares de los internos provocando pérdidas de los objetos.
- No hay un control de las medidas tomadas con los internos que se les ocupen o decomisen objetos lo que puede incidir en la repetitividad de las indisciplinas.
- El inventario del armamento y equipos son registrados en formato duro y pueden producirse indebidas modificaciones o pérdidas de dichos inventarios.
- El uso de la tarjeta 4-4-19 y de la Tablilla de Control Físico de movimientos de internos en el proceso de control físico imposibilita agilizar dicho proceso, además pueden ocurrir modificaciones, pérdidas o desorganización de dichos materiales.
- Existen problemas para garantizar el control de la seguridad y acceso a los centros penitenciarios por parte de vehículos no autorizados.

A partir de lo antes expuesto surge como **problema a resolver**: la gestión actual de los procesos de control del equipamiento y el armamento en los centros penitenciarios así como el control de los internos dificulta el correcto funcionamiento de la Seguridad Penal y el Orden Interior en el sistema penitenciario cubano respectivamente.

En correspondencia con lo anterior surge como **objeto de estudio**: los procesos de control de equipos, armamentos y de los internos en Sistemas Penitenciarios.

Se plantea como **objetivo general**: desarrollar los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP a partir del análisis de los

requisitos de software establecidos con el cliente y haciendo uso de la arquitectura y las tecnologías definidas por el proyecto.

Queda definido como **campo de acción**: los procesos de control de equipos, armamentos, acceso de vehículos e internos en el sistema penitenciario cubano.

Teniendo como **idea a defender**: con el desarrollo de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP se logrará gestionar de manera completa y adecuada los procesos de control del equipamiento, de los internos y del armamento en los centros penitenciarios de Cuba.

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

1. Elaborar el marco teórico de la investigación.
2. Diseñar los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.
3. Implementar los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.
4. Realizar pruebas de calidad a los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.

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

- Análisis de sistemas informáticos para centros penitenciarios que implementen soluciones enmarcadas dentro del objeto de estudio.
- Descripción de las herramientas, metodología y tecnologías para dar solución al problema planteado.
- Elaboración de los diagramas de clases del diseño de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.
- Elaboración de los diagramas de interacción de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.
- Diseño de la base de datos.
- Implementación de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.

- Elaboración del diagrama de despliegue.
- Diseño de los casos de prueba de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.
- Realización de las pruebas de calidad a los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.

Dentro del **marco conceptual** se pueden encontrar los siguientes conceptos:

- Control físico: es el conjunto de acciones ejecutadas para el control de la población interna y de sus movimientos, se realiza mediante la Tablilla de Control Físico de movimientos de internos, los recuentos ordinarios, sorpresivos, físicos y las tarjetas 4-4-19.
- Recuento ordinario: es la acción y resultado de contar y controlar la presencia física de los internos por las autoridades penitenciarias, desde su ingreso y ubicación exacta en el lugar de internamiento para el cumplimiento de la sanción o medida impuesta.
- Recuento sorpresivo: se efectúan para controlar la presencia física de los internos, es una acción de control y tiene como objetivo, contribuir a la prevención y detección oportuna de evasiones u otras violaciones del régimen penitenciario.
- Recuento físico: el recuento físico se ejecuta para comprobar la presencia física e identificación de los internos, de acuerdo al plan elaborado al efecto. También se realiza cuando en los recuentos ordinarios o sorpresivos falte algún interno, o cuando la situación operativa así lo aconseje. Para su realización se emplea la Tarjeta de Control Físico (4-4-19).

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

Métodos teóricos:

- Analítico-sintético: se utiliza para realizar el estudio de las herramientas, tecnologías y metodología, destacando las características que las hacen factibles para dar solución al problema planteado.
- Modelación: se utiliza en el diseño del módulo a implementar.

Métodos empíricos:

- Observación: se realiza a partir del desempeño de los autores en el proyecto de desarrollo y con el fin de identificar el marco de trabajo del proyecto.

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

El presente documento está estructurado en tres capítulos. Los capítulos contienen lo siguiente:

Capítulo I: Fundamentación Teórica

Se efectúa un estudio referente a la situación actual de algunas soluciones informáticas correspondientes a sistemas penitenciarios. Se describen, además, las principales herramientas y tecnologías a utilizar para el desarrollo del problema.

Capítulo II: Diseño del sistema

Se expone una representación gráfica de los diagramas de clases e interacción del diseño, así como el diseño de la Base de Datos (BD). Se puntualizan las funcionalidades del sistema y la arquitectura que se utiliza en el proyecto para el desarrollo de la aplicación.

Capítulo III: Implementación y Prueba

En este capítulo se generan el modelo de despliegue y los diagramas de componentes, se implementa el sistema y se concluye realizando pruebas de funcionalidad para comprobar si la aplicación cumple sus objetivos.

Capítulo I. Fundamentación Teórica

1.1. Introducción

En el presente capítulo se explican el funcionamiento de los procesos de control de equipamiento y armamento, control físico y ocupaciones, decomisos y hallazgos en algunos sistemas penitenciarios desplegados y utilizados en el país y en el extranjero. Se describen las herramientas a utilizar en el diseño e implementación de los módulos, realizando una breve descripción que justifique su elección.

1.2. Estudio del estado del arte

Hoy en día los sistemas informáticos de gestión de información son tan útiles que las instituciones que manejan grandes volúmenes de información se han renovado y han ido adquiriendo estos sistemas. Después de realizar una búsqueda de aplicaciones informáticas que estén dentro del objeto de estudio de la investigación, se presentan los siguientes sistemas informáticos.

Sistema Automatizado para el Control del Recluso (SACORE)

El MININT no está ajeno a las tecnologías informáticas, y ha destinado un porciento de ellas para automatizar el Sistema Penitenciario, con el objetivo de reorganizar y facilitar el trabajo dentro de las Instituciones Penales del país, garantizando la gestión y control de los datos y asegurando las leyes vigentes para la Dirección de Establecimientos Penitenciarios.

Con la existencia del SACORE muchos datos son manejados y conservados en formato duro, o a través de herramientas informáticas que no ofrecen grandes facilidades para su gestión. No se puede recurrir a este sistema para dar solución a la problemática debido a que no posee funcionalidades que den cumplimiento a la gestión de los procesos de control de medios técnicos, control de acceso, control físico u de ocupaciones, decomisos y hallazgos.

Sistema de Gestión Penitenciaria (SIGEP)

El SIGEP es un software diseñado e implementado para el sistema penitenciario de Venezuela por la Universidad de las Ciencias Informáticas, tributando al proceso de informatización y modernización de las instituciones de la República Bolivariana de Venezuela a raíz de los acuerdos del ALBA (Alternativa Bolivariana para las Américas).

El SIGEP realiza el control de medios técnicos en los centros penitenciarios y la planificación de las requisas y decomisos que se realizan a los internos y visitantes del centro.

En cuanto al control de medios técnicos permite:

- El control de todo el armamento, equipos y medios de seguridad con que disponen en el parque de armas.
- El registro de la entrega, devolución o dar de baja de un armamento al parque de armas.
- La búsqueda de una entrega dado un criterio de búsqueda.

En cuanto a la planificación de las requisas y decomisos permite:

- El registro de los diversos tipos de requisas: individual, colectiva, extraordinaria, ordinaria, general, en locales y a las visitas.
- El control de todos los elementos no permitidos que pueden ser incautados en una requisa dando la posibilidad de registrar un nuevo elemento, eliminar y modificar los mismos.

A pesar de que este sistema posee más funcionalidades que el SACORE con respecto al control de los internos y al control de los medios técnicos en los centros penitenciarios, se ajusta a las leyes establecidas por el gobierno venezolano y maneja información como los datos del interno y de los oficiales, así como los tipos de armamentos empleados por los oficiales, que difieren de la información necesaria en el Sistema Penitenciario Cubano.

1.3. Herramientas y Tecnologías a utilizar para el desarrollo de la aplicación

A continuación se realiza una breve descripción de la metodología, herramientas y tecnologías definidas por parte del equipo de arquitectura del proyecto.

1.3.1. Metodología de Desarrollo

Proceso Unificado de Rational

Como metodología de referencia para guiar el trabajo de implementación, se utilizó RUP[1]. RUP es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Usando RUP como metodología de desarrollo el trabajo es más rápido y organizado ya que divide el desarrollo del software en fases (Ver Figura 1), las cuales se especifican a continuación:

- Inicio: el objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- Elaboración: etapa en la que se determinar la arquitectura óptima del proyecto.
- Construcción: se obtiene la capacidad operacional inicial.
- Transición: obtener el producto acabado. (1)

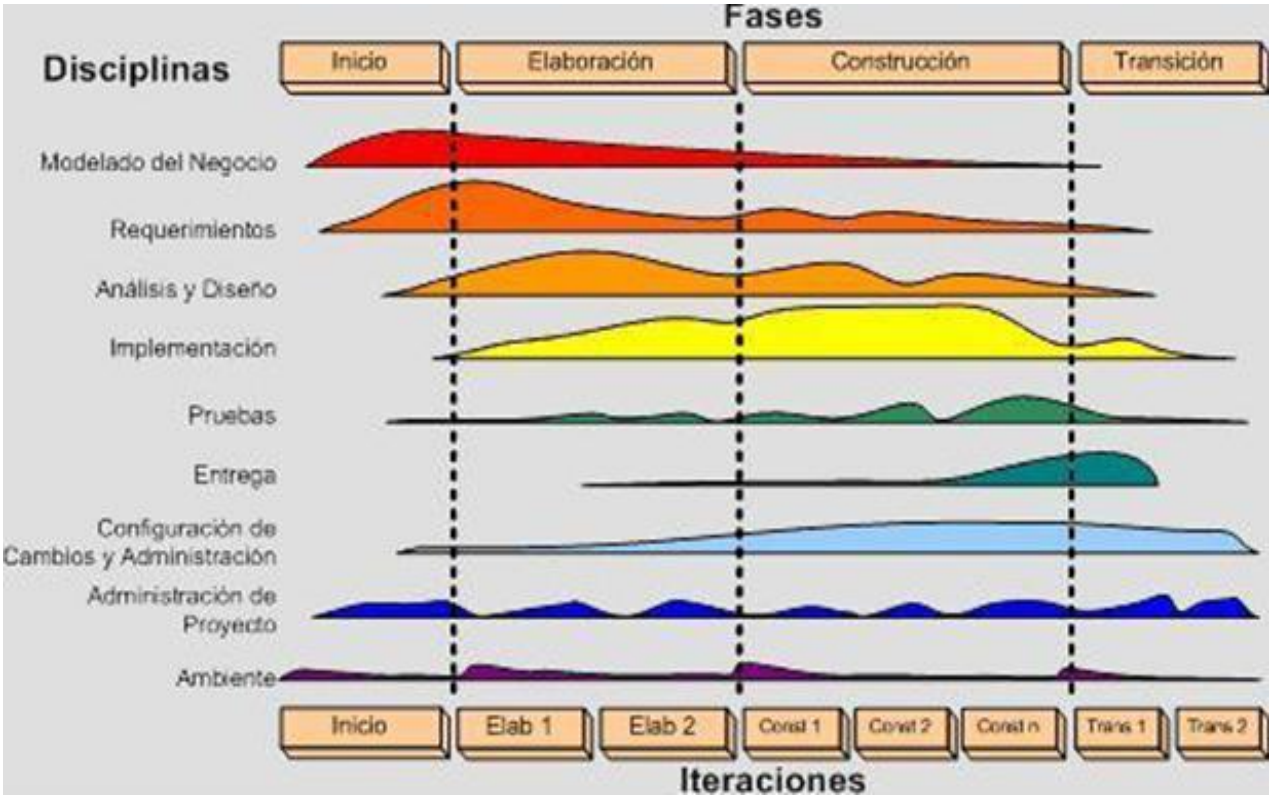

Figura 1 Fases y flujos de trabajo en RUP

[1] Rational Unified Process (Proceso Unificado de Rational)

RUP define los roles y sus respectivas actividades, los productos que se deben generar y los flujos de trabajo de las disciplinas que se especifican como secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.

Es una metodología empleada en sistemas de software de gran envergadura donde se gestiona un cúmulo de información y en el cual trabajan un número considerable de personas.

SIDEP es un software complejo, grande y para el cual se ha conformado un proyecto integrado por un gran número de personas. Por sus características requiere un modelo de desarrollo iterativo e incremental, que permita generar documentación en cada momento del proceso de desarrollo de forma tal que facilite la comunicación entre los miembros del proyecto, la planificación de etapas de desarrollo y permita la gestión de diversos roles dentro del proyecto. Todo ello conlleva al uso de la metodología de desarrollo RUP para el SIDEP.

1.3.2. Lenguaje de modelado

Lenguaje Unificado para la Construcción de Modelos

UML[2] es actualmente el lenguaje de modelado de sistemas de software más destacado y utilizado.

Entre las funciones de UML se encuentran:

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión. (2)

Fundamentalmente facilita a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados. UML ofrece un estándar para escribir un "plano" del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y

[2]Unified Model Language (Lenguaje Unificado para la Construcción de Modelos)

componentes de software reutilizables. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software.

UML permite:

- Configurar el modelado de BD, el modelado de requerimientos, la interoperabilidad, la generación de documentación y la generación de código base para diferentes lenguajes de programación como Java, C# y PHP.
- La integración con herramientas de desarrollo.
- Dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
- A los desarrolladores poder diseñar la documentación del sistema con plantillas de diseño y a los analistas de sistemas poder estimar las consecuencias de los cambios con los diagramas de análisis de impacto, como la matriz y el diagrama de análisis.

UML cuenta con diversos tipos de diagramas, los cuales exponen disímiles aspectos de las entidades representadas. Algunos de ellos son:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de interacción
- Diagrama de componentes
- Diagrama de despliegue

1.3.3. Herramientas de modelado

Visual Paradigm Suite 3.4

Se eligió Visual Paradigm Suite 3.4 como herramienta de modelado de software empleada para generar y representar los artefactos del sistema. La UCI adquirió la licencia de dicha herramienta hace algunos años ya que Visual Paradigm Suite 3.4 es un software privativo.

Su mayor éxito radica en la capacidad de ejecutarse sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma. Constituye un software privativo y sus distintas ediciones son compatibles. (3)

Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Utiliza UML como lenguaje de modelado, ofreciendo soluciones de software que permiten a las organizaciones desarrollar aplicaciones de calidad, más rápido, satisfactorias y a un menor coste. Es muy fácil de usar y presenta un entorno gráfico agradable para el usuario. Facilita una excelente interoperabilidad con otras herramientas CASE [3] y principales IDE [4] de desarrollo.

Embarcadero ER/Studio 8.0.0

La arquitectura del proyecto seleccionó la herramienta ER\Studio para modelar las entidades de los módulos Control de acceso, Control de medios técnicos, Control físico y Ocupaciones, decomisos y hallazgos y sus relaciones entre ellos debido a que se eligió a Oracle como sistema gestor de BD.

Es una poderosa herramienta CASE, para efectuar el modelado lógico y físico de bases de datos Relacionales y Multidimensionales. Entre otras características, ER/Studio permite hacer una separación entre los modelos lógicos y físicos, pero manteniendo una total integración entre ellos, así como con la base de datos. Es posible generar múltiples modelos físicos asociados a un mismo modelo lógico, para diferentes plataformas. (4)

Constituye una herramienta muy fácil de usar. Soporta más de 30 de las bases de datos más importantes de la industria, entre ellas Oracle.

Permite realizar:

- Diseño físico y lógico separados.
- Transformación automática de un diseño lógico a un diseño físico para una plataforma específica de BD.
- Extensa validación de los modelos.
- Validar ampliamente la calidad e integridad tanto del diseño lógico como del diseño físico del modelo.

[3] Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadora)

[4] Integrated Development Environment (Entorno de Desarrollo Integrado)

1.3.4. Entorno de Desarrollo Integrado

Un IDE es un programa informático compuesto por un conjunto de herramientas de programación y puede dedicarse en exclusiva a un solo lenguaje de programación o para varios. Dicho programa es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

SpringSource Tools Suite 2.5.1 (STS) está basado en Eclipse (5), pero también reúne muchas herramientas y asistentes, cuyo objetivo es el de facilitar y agilizar el desarrollo de aplicaciones las cuales utilicen tecnologías como Spring Framework y Spring Web Flow. STS es una herramienta admirable ya que ayuda a manejar de manera muy potente los proyectos Spring, con multitud de asistentes, editores, opciones y configuraciones. STS trae soporte para Groovy y Grails, pues este IDE permite ejecutar aplicaciones Grails en Tomcat Server, lanzar comandos Grails, gestiona distintas versiones de Grails, gestiona dependencias, posee un soporte de depuración mejorado para los proyectos Groovy, mejoras en el tipo de inferencia y soluciones rápidas en el editor de Groovy, y este IDE no es propietario por lo que el equipo de Arquitectura seleccionó a este como el IDE a emplear.

1.3.5. Contenedor web

Apache Tomcat 6.0.3

Apache Tomcat es un contenedor web escrito en Java, por lo que funciona en cualquier sistema operativo que disponga de una máquina virtual Java y desarrollado en un ambiente participativo y abierto. (6) Es un software de código abierto multiplataforma. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad. Apache Tomcat es usado en numerosas aplicaciones web de gran escala y críticas en diversas industrias y organizaciones que se referencian en su sitio oficial.

1.3.6. Controlador de versiones

Subversión 1.6.14

Subversión es un software de sistema de control de versiones. Esta desarrollado sobre software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. (7) Una característica importante de Subversión es que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. El acceso al repositorio en mediante la red, lo que le permite ser usado por personas que se encuentran en distintos ordenadores.

Su principal importancia radica en que varias personas pueden modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomentando así la colaboración. Permite además integrar varias interfaces a entornos de desarrollo, un ejemplo de esto es el TortoiseSVN, el cual provee integración con el explorador de Windows y un ejemplo de desventaja es el manejo de cambio de nombres de archivos que no es completo, pues este es manejado como la suma de una operación de copia y una de borrado.

1.3.7. Sistema Gestor de Bases de Datos

Oracle 11g

Se seleccionó Oracle 11g como sistema gestor para la BD ya que es sólido y garantiza la seguridad e integridad de los datos. Entre sus características se encuentran:

- Ayuda a administrar y almacenar grandes volúmenes de datos.
- Constituye una herramienta de administración gráfica cómoda de utilizar.
- Ayuda al análisis de la información.
- Se caracteriza por su estabilidad y escalabilidad.
- Es multiplataforma.
- Es un software privativo lo cual trae consigo limitaciones para su uso.
- Efectúa recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados apoyando el diseño y optimización de los modelos de datos.

Oracle es muy útil para los desarrolladores ya que los asiste con sus conocimientos de SQL y de construcción de procedimientos almacenados. (8)

1.3.8. Tecnologías

JavaScript Object Notation

JavaScript Object Notation (JSON por sus siglas en inglés) es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. (9)

JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador semántico de JSON.

Dojo Toolkit 1.5

Dojo Toolkit es una librería de clases JavaScript. Dojo es un framework[5] que contiene API (Interfaz de Programación de Aplicaciones) y widgets (controles) para facilitar el desarrollo de aplicaciones web que utilicen tecnología AJAX[6]. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop Apis, widget Apis, abstracción de eventos, almacenamiento de Apis en el cliente, e interacción de Apis con AJAX. Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas, y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Es conocido como "la navaja suiza del ejército de las bibliotecas JavaScript". Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos. (10)

[5] Aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

[6] Javascript Asíncrono And XML: es una técnica de desarrollo web para crear aplicaciones interactivas.

Los complementos de Dojo son componentes pre empaquetados de código JavaScript, HTML[7] y CSS[8] que pueden ser usados para enriquecer aplicaciones web. Facilita componentes para la interfaz de usuario tales como:

- Menús, pestañas.
- Tablas ordenables, gráficos dinámicos.
- Soporte para arrastrar y soltar.
- Formularios y rutinas de validación para los parámetros.
- Calendario, selector de tiempo y reloj.
- Editor online de texto enriquecido.

1.3.9. Lenguajes de programación

Groovy 1.7.8

Es un lenguaje de programación orientado a objetos, ágil, dinámico y de código abierto para la Máquina Virtual de Java. (11) Usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. La mayor parte de código escrito en Java es totalmente válido en Groovy por lo que este lenguaje es de muy fácil adopción para programadores Java.

Es de gran alcance y flexible. Sin embargo, su integración con Java, su dinamismo y sintaxis simple lo convierten en un complemento perfecto para Grails, que es el framework elegido para el desarrollo del SIDEP.

Groovy está basado en paradigmas como “Convención sobre configuración”, simplifica las pruebas mediante pruebas de unidad. Se integra perfectamente con todas las clases existentes de Java y de las librerías.

Lenguaje de Marcas de Hipertexto

[7] HyperText Markup Language (Lenguaje de enmarcado).

[8] Cascading Style Sheets (Hojas de Estilo en Cascada), lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

HTML es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto. Puede describir la apariencia de un documento e incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

Este lenguaje se utilizará como parte de la codificación de las páginas servidoras y como resultado del procesamiento de las mismas. Es el lenguaje de programación por detrás de las vistas que ve el cliente.

CSS

Las hojas con estilo de cascada son el lenguaje empleado para definir la presentación y los estilos de una página HTML.

JavaScript

Lenguaje script utilizado para unir el conjunto de tecnologías usados en la web. Es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Se emplea fundamentalmente integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. (12) Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM. Su principal importancia radica en que tradicionalmente, se venía utilizando en páginas web HTML[9], para realizar operaciones y en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

1.3.10. Marco de trabajo web

Grails 1.3.7

[9] HyperText Markup Language (Lenguaje de Marcas de Hipertexto)

Grails es un framework[10] web de código abierto desarrollado sobre el lenguaje de programación Groovy (el cual a su vez se basa en la plataforma de Java) para aplicaciones web libre. (13) Combina principios tales como "convención sobre configuración" y "No te repitas" junto a una serie de frameworks de código abierto como Hibernate (para mapeo objeto-relacional), Spring (para inyección de dependencia) y SiteMesh (para plantillas). Todo esto, unido al lenguaje dinámico Groovy construido sobre Java.

Grails se puede ampliar a través de plugins. Por su dinamismo es capaz de cortar el ciclo de desarrollo, ahorrando tiempo de trabajo y lo agilizándolo. Incluye un contenedor web, base datos, sistemas de construcción y pruebas. Esta combinación puede reducir el tiempo inicial del proyecto y el tiempo de instalación del desarrollador a minutos en lugar de horas. No hay que instalar o mantener scripts, todo lo que se necesita viene incluido en un paquete fácil de instalar.

Sus principales características son:

- Alta productividad: Grails tiene tres características que intentan incrementar su productividad comparándolo con los frameworks Java tradicionales:
  - Inexistencia de configuración XML
  - Entorno de desarrollo preparada para funcionar desde el primer momento
  - Funcionalidad disponible mediante métodos dinámicos.
- Integración con la plataforma Java: al estar construido sobre la plataforma Java, con lo que es muy fácil integrarlo con librerías Java, Framework y código existente. La mejor característica que Grails ofrece en este ámbito es una integración transparente con clases mapeada mediante el framework Hibernate ORM. Esto representa que aplicaciones existentes que utilicen Hibernate pueden utilizar Grails sin recompilar el código o reconfigurar las clases Hibernate, aprovechando los métodos de persistencia que se mencionan anteriormente. Una consecuencia es que se puede utilizar scaffolding con las clases Java mapeadas con Hibernate. Otra consecuencia es que las capacidades de Grails están totalmente disponibles para estas clases y las aplicaciones que las usan.

[10] Framework(Marco de trabajo)

- Persistencia: el modelo de datos en Grails se graba en la BD utilizando GORM (Grails Object Relational Mapping por sus siglas en inglés). Las clases de dominio se graban en el directorio grails-app/domain.

Los plugins que utiliza el framework son:

- dojo-release-1.3.0
- grails-hibernate-1.3.5
- grails-webflow-1.3.5
- grails-acegi-0.5.3
- grails-spring-security-core-1.0.1
- grails-dojo-1.3.0
- grails-tomcat-1.3.5

1.3.11. Plataforma de programación

Plataforma Java de Edición Empresarial

J2EE[11] es una plataforma de programación parte de la plataforma Java para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones arquitectura en capas.

Define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. Resulta una propuesta atractiva, interesante y de vanguardia que responde, a la demanda actual para el desarrollo de software, bajo el concepto de arquitectura en capas. (14)

[11] Java 2 Platform Enterprise Edition (Plataforma Java de Edición Empresarial)

1.4. Conclusiones parciales

En este capítulo se realizó un análisis de las soluciones informáticas relacionadas con los sistemas penitenciarios, lo que permitió determinar que las mismas no satisfacen las necesidades del SIDEP ya que no se corresponden con las características adaptables a las necesidades de los procesos de control de equipos, armamentos, acceso de vehículos e internos del sistema penitenciario cubano, por lo que se hace necesario desarrollar una aplicación para ello.

La metodología, herramientas, tecnologías y lenguajes que serán utilizados facilitarán el trabajo a realizar en cada una de las disciplinas que serán desarrolladas para darle solución al problema propuesto de la manera más óptima posible de acuerdo con las especificaciones del SIDEP.

Capítulo II. Diseño del sistema

2.1. Introducción

En el presente capítulo se explican los principales elementos para desarrollar el producto, basándose en el análisis previamente realizado. Se describen las funcionalidades de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos. Además, se abordan elementos de importancia para el sistema como la composición de la arquitectura del sistema, los diagramas de clases, los diagramas de interacción y el diseño de la BD.

2.2. Descripción de las funcionalidades

A continuación se muestran las tablas que contienen las funcionalidades de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos teniendo en cuenta que abarcan 15 casos de uso, de los cuales 10 son de complejidad Alta, 1 de complejidad Media y 4 de complejidad Baja.

Nombre Caso de Uso	Descripción
Gestionar medios técnicos	El caso muestra un listado de los medios técnicos existentes, una para el armamento y otra para el resto de los medios técnicos y permite insertar uno nuevo o modificar uno que ya exista.

Tabla 1 Descripción de funcionalidades módulo Control de medios técnicos.

Nombre CU	Descripción
CRUD-RD Vehículo	El caso de uso muestra los vehículos que se han registrado en el día y las opciones de insertar un nuevo vehículo y modificar los datos de un vehículo ya registrado.
Consultar vehículos	El caso de uso permite consultar los datos de los vehículos que visiten al Centro Penitenciario en un rango

	de fecha establecido, por el organismo al que pertenece o al área al que se dirige.

Tabla 2 Descripción de funcionalidades módulo Control de acceso.

Nombre CU	Descripción
Registrar recuento ordinario	El caso de uso permite registrar los datos de un recuento ordinario asociándole los funcionarios implicados en el recuento.
Gestionar recuento sorpresivo	El caso de uso permite registrar los datos de un recuento sorpresivo planificado y no planificado. En el caso de los recuentos planificados se asocian a un recuento registrado en el plan de recuento sorpresivo ya creado.
Gestionar recuento físico	El caso de uso permite registrar los datos de un recuento físico planificado y no planificado. En el caso de los recuentos planificados se asocian a un recuento registrado en el plan de recuento físico ya creado.
Gestionar Plan de recuentos sorpresivos	El caso de uso muestra los Planes de recuentos sorpresivos realizados, permite modificar un plan de recuento sorpresivo o registrar uno nuevo.
Gestionar Plan de recuentos físicos	El caso de uso muestra los Planes de recuentos físicos realizados, permite modificar un plan de recuento físico o registrar uno nuevo.
Consultar recuentos	El caso de uso permite consultar los datos de los recuentos realizados en el Centro Penitenciario por un rango de fecha o por el tipo de recuento.
Consultar planes de recuentos	El caso de uso permite consultar los datos de los planes recuentos realizados en el Centro Penitenciario por un rango de fecha o por el tipo de recuento.

Tabla 3 Descripción de funcionalidades módulo Control físico.

Nombre CU	Descripción
Gestionar ocupación	El caso de uso muestra el listado de las ocupaciones registradas y permite visualizar el listado de ocupaciones registradas, visualizar una ocupación, buscar una ocupación y generar reporte de los diferentes tipos de ocupaciones.
Registrar ocupación	El caso de uso permite registrar la ocupación de retención durante el ingreso al lugar de internamiento. Además posibilita generar el acta de ocupación y del interno.
Gestionar hallazgo	El caso de uso permite visualizar el listado de hallazgos, registrar un hallazgo, buscar un hallazgo, generar reporte de los diferentes tipos de hallazgos.
Registrar hallazgo	El caso de uso permite registrar un nuevo hallazgo en el sistema. Además posibilita generar el acta de hallazgo.
Gestionar decomiso	El caso de uso muestra el listado de los decomisos registrados. Además permite visualizar el listado de decomisos, describir un objeto decomisado, buscar un decomiso, generar reporte de los diferentes tipos de decomisos.
Describir objeto decomisado	El caso de uso muestra una interfaz con los datos del objeto decomisado registrado previamente en la requisa y brinda la posibilidad de insertar nuevos datos.
Consultar ocupaciones, decomisos y hallazgos	El caso de uso brinda la posibilidad de visualizar el listado de ocupaciones, decomisos y hallazgos registrados, y generar un reporte por cada uno de estos.

Tabla 4 Descripción de funcionalidades módulo Ocupaciones, decomisos y hallazgos.

2.3. Marco de trabajo de desarrollo web

Se emplea Grails como marco de trabajo ya que es utilizado para el desarrollo de aplicaciones web sobre plataforma J2EE. Grails utiliza el estilo arquitectónico “Llamada y Retorno” y dentro de este el MVC (Modelo-Vista-Controlador), además presenta una organización de cuatro capas:

- Modelo o capa de datos
- Vista o capa de presentación
- Controlador o capa de control
- Capa de servicios

Grails facilita un conjunto de adaptadores y código de interfaz; lo que posibilita gestionar los diferentes componentes en cada capa del desarrollo, y asegurar que dichos adaptadores funcionan correctamente con tantas combinaciones de los componentes como sea permitido.

2.4. Arquitectura del Sistema

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático.

Arquitectura n-capas

La arquitectura definida para la construcción del SIDEP es una arquitectura n-capas la cual está compuesta de la siguiente manera: la capa de datos o modelo, la capa de presentación o vista, la capa de control, en estas tres capas se aplica el patrón MVC de Grails, y una capa de servicios, que es donde normalmente se implementa la lógica de negocio de los casos de uso.

Se muestra seguidamente la arquitectura n-capas de la aplicación.

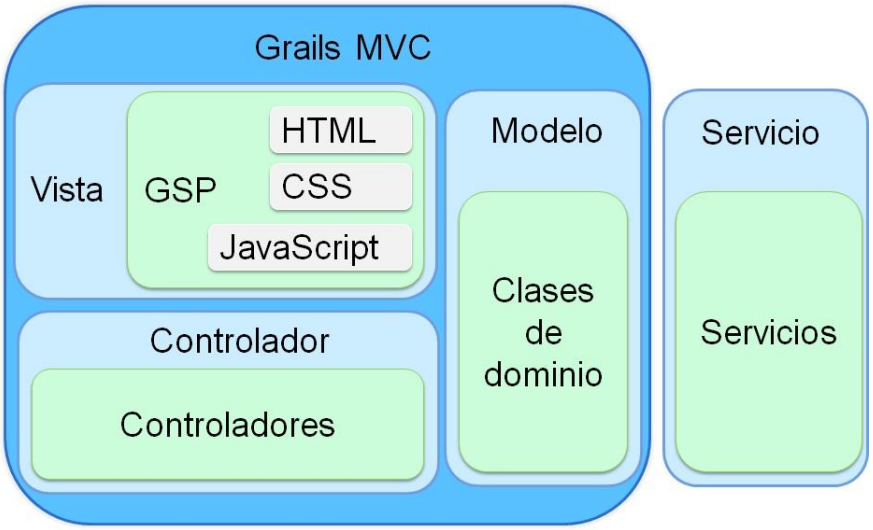

Figura 2 Arquitectura del sistema

A continuación se explican las capas:

Capa de datos o modelo: esta capa está compuesta por las entidades o clases del dominio, en las cuales va a persistir la información de la aplicación. Estas clases son utilizadas por los servicios para hacer las consultas y obtener la información que el usuario solicita o el sistema necesita. Las clases del dominio son entidades fundamentales para el sistema, ya que en ellas persisten los datos con los que posteriormente el cliente va a interactuar, modificar o insertar. Grails utiliza un gestor de persistencia escrito en Groovy sobre Hibernate conocido por GORM[12]para controlar el ciclo de vida de las entidades.

Una vez definidas las propiedades de las clases del dominio, GORM se encarga de:

- Generar las tablas correspondientes en la BD con los campos necesarios para almacenar cada propiedad y proporcionar los métodos de búsqueda y modificación que permitan manipular la entidad.
- Añade automáticamente un campo id a la clase y genera un valor único secuencial para cada instancia de la misma.

[12]Groovy Object Relational Mapping

- Restringe los valores que pueden asignarse a los atributos de cada entidad mediante la propiedad constraints.
- Grails presenta una técnica llamada Scaffolding, la cual permite que una vez creada las clases de dominio se creen las clases controladoras y las vistas necesarias para realizar las operaciones.

Capa de presentación o vista: está formada por plantillas Groovy Server Page (gsp), las cuales son las encargadas de mostrar en el navegador del cliente el contenido que se encuentra en el modelo, mediante código HTML, JavaScript y Cascading Style Sheets (CSS). Se utilizó el framework Dojo para la creación de vistas, utilizando la tecnología Ajax.

Capa de control o controlador: se comunica con las demás capas. Los controladores son los componentes responsables de recibir órdenes del usuario, gestionar la ejecución de la lógica del negocio y después actualizar la vista para mostrar al usuario el estado en que ha quedado el modelo de datos. Son los responsables de interceptar las peticiones HTTP del navegador y generar la respuesta correspondiente, ya sea en el propio controlador o delegando el trabajo en una vista.

Capa de servicios: en esta capa se encuentran los servicios componentes que implementan la lógica de negocio de los casos de uso que componen la aplicación. Tienen una estrecha relación con el modelo de datos, porque en él se hacen llamadas a los métodos que brinda GORM para la consulta de los datos. Todas las entidades de una aplicación Grails que pertenecen al modelo de datos poseen métodos de instancia que facilitan su manipulación y pueden ser llamados desde los servicios. GORM también inyecta una serie de métodos estáticos en las clases persistentes para hacer consultas sobre los datos almacenados en la tabla o tablas correspondientes. Estos pueden ser list(),get(), entre otros. Para realizar búsquedas avanzadas que incluyen muchos campos y comparaciones complejas es recomendable construir consultas mediante Criteria.

2.5. Patrones de diseño

Los patrones de diseño brindan una solución ya probada y documentada a problemas comunes de desarrollo de software. Estos se caracterizan por estar conformados por un conjunto de elementos, como: su nombre, el problema que resuelve el patrón (cuándo aplicar un patrón), la solución al

problema (descripción abstracta del problema) y las implicaciones de cuándo y cómo el patrón debe ser usado así como los beneficios y problemas que implican su uso (costos y beneficios).

2.5.1. Patrones de diseño de la Pandilla de los Cuatro

Los patrones GOF[13]son patrones de diseño que definen una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. De los patrones que propone el grupo de los patrones GOF se utilizará el Singleton[14].

Singleton

Este patrón garantiza que una sola clase sólo tenga una instancia y proporciona un punto de acceso global a ella. (15) La utilización de este patrón se evidencia en los servicios, los cuales son instanciados por los controladores para utilizar las funcionalidades que los mismos brindan. El acceso a ellos es global, lo que significa que todos los controladores acceden a la misma instancia del servicio, accediendo a la misma información contenida en ellos.

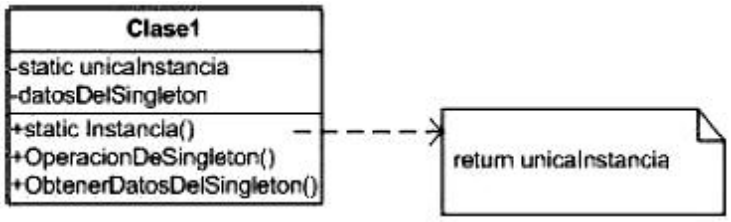

Figura 3 Estructura del Patrón Singleton

2.5.2. Patrones generales de software para asignar responsabilidades

Los patrones GRASP[15] son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software, estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (16) De dichos patrones GRASP se utilizarán de forma

[13]Gang of Four, Banda de los cuatro

[14]Instancia única

[15]General Responsability Assignment Software Patterns(Patrones Generales de Asignación de Responsabilidad de Software)

general los patrones Alta Cohesión, Bajo Acoplamiento, Creador, Controlador y Experto. Los conceptos de cohesión y acoplamiento están íntimamente relacionados. Un mayor grado de cohesión implica uno menor de acoplamiento.

Alta Cohesión

Este patrón explica que la información que almacena una clase debe de ser coherente y debe estar relacionada con la clase. Asigna una responsabilidad de modo que la cohesión siga siendo alta. (16)

Bajo Acoplamiento

Este patrón asigna una responsabilidad para mantener bajo acoplamiento. (16) Declara que las clases no deben estar tan relacionadas entre sí, de manera que en caso de producirse una modificación en alguna de las clases, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. (16)

Controlador

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. (16) Todas las peticiones web son manejadas por un solo controlador, que es el punto de entrada único de toda la aplicación en un entorno determinado. Dicho controlador utiliza el sistema de enrutamiento para asociar el nombre de una acción con la URL solicitada por el usuario.

Experto

Asigna una responsabilidad al más competente en información, donde la clase cuenta con la información necesaria para cumplir la responsabilidad. (16) Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos. Es el patrón que más se usa para asignar responsabilidades.

2.5.3. Otros patrones de diseño utilizados

Inversión de Control (IoC)

Inversión de Control (*Inversion of Control*, IoC según sus siglas en inglés) plantea que las dependencias de un componente no deben gestionarse desde el propio componente para que este solo contenga la lógica necesaria para hacer su trabajo. (17) Cuando se crea un componente en la aplicación, Grails configura a Spring para controlar su ciclo de vida (cuándo se crea, cuántas instancias mantiene vivas a la vez, cómo se destruyen, etc.) y sus dependencias (qué otros componentes necesita para desarrollar su trabajo y cómo seguirlos). El objetivo de esta técnica es mantener los componentes lo más sencillo posible, incluyendo únicamente código que tenga relación con la lógica de negocio. Así la aplicación será más fácil de comprender y mantener.

Objeto de Acceso a Datos (DAO)

Este patrón gestiona las distintas fuentes de datos, además de encapsular la fuente de datos sirve también para ocultar la forma para acceder a ellos. Por otra parte se logra aislar la capa de acceso a datos del resto de la aplicación, logrando con ello un bajo acoplamiento de la misma con las capas superiores. Estos nos permiten contar con diferentes fuentes de datos, como BD, servicios externos, archivos, etc.

2.6. Diagramas de Clases

2.6.1. Descripción de las clases más significativas

A continuación se muestran las tablas con los datos de las clases más significativas de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.

Nombre: Datos de medios técnicos	
Tipo de clase: Entidad	
Atributo	**Tipo**
Medio	Nomenclador
Plantilla	Número
Físico	Número
Necesidad	Número
Por Completamiento	Número
Por Reposición	Número

Tabla 5 Clase de dominio Datos de medios técnicos del módulo Control de medios técnicos.

Nombre: Medio técnico armamento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Estado	Cadena de caracteres
Número de serie	Cadena de caracteres
Arma	Nomenclador

Tabla 6 Clase de dominio Medio técnico armamento del módulo Control de medios técnicos.

Nombre: Vehículo_acceso	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha	Fecha
Hora Entrada	Número
Hora Salida	Número
Marca	Nomenclador
Color	Cadena de caracteres
Organismo al que pertenece	Cadena de caracteres
Número de matrícula	Número
Área	Nomenclador

Tabla 7 Clase de dominio Vehículo_acceso del módulo Control de acceso.

Nombre: Ocupación	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha	Fecha
Hora	Número
Lugar de ocurrencia	Cadena de caracteres
Tipo de ocupación	Nomenclador
No. Expediente Interno	Número
Nombre Oficial responsable	Cadena de caracteres

Tabla 8 Clase de dominio Ocupación del módulo Ocupaciones, decomisos y hallazgos.

Nombre: Decomiso	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha	Fecha
Hora	Número
Lugar de ocurrencia	Cadena de caracteres

Tipo de decomiso	Nomenclador
No. Expediente Interno	Número
Medidas tomadas	Cadena de caracteres
Nombre Oficial responsable	Cadena de caracteres
Descripción del objeto decomisado	Cadena de caracteres

Tabla 9 Clase de dominio Decomiso del módulo Ocupaciones, decomisos y hallazgos.

Nombre: Hallazgo	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha	Fecha
Hora	Número
Lugar de ocurrencia	Cadena de caracteres
Tipo de hallazgo	Nomenclador
Nombre Oficial responsable	Cadena de caracteres
Destino del hallazgo	Cadena de caracteres

Tabla 10 Clase de dominio Hallazgo del módulo Ocupaciones, decomisos y hallazgos.

Nombre: Participante	
Tipo de clase: Entidad	
Atributo	**Tipo**
Primer nombre	Cadena de caracteres
Segundo nombre	Cadena de caracteres
Primer apellido	Cadena de caracteres
Segundo nombre	Cadena de caracteres
Cargo	Cadena de caracteres
Grado militar	Nomenclador

Tabla 11 Clase de dominio Participante del módulo Control físico.

Nombre: Plan de Recuento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha de confección	Fecha
Mes	Nomenclador

Tabla 12 Clase de dominio Plan de Recuento del módulo Control físico.

Nombre: Recuento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Tipo	Nomenclador

Fecha	Fecha
Hora de inicio	Número
Hora de fin	Número
Cantidad de internos recontados	Número
Observaciones	Cadena de caracteres

Tabla 13 Clase de dominio Recuento del módulo Control físico.

Nombre: Responsable del recuento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsable	Booleano

Tabla 14 Clase de dominio Responsable del recuento del módulo Control físico.

Nombre: crearRecuento.html		
Tipo de clase: Interfaz		
Descripción: Muestra los datos de un recuento.		
Atributo	**Tipo**	**Descripción**
fechaRecuento	Date	Guarda la fecha en que se realiza el recuento.
cantInternos	Integer	Contador de internos.
observaciones	String	
funcionario	Funcionario	Oficial participante en el recuento.
responsable	Boolean	Oficial responsable de dirigir el recuento.

Tabla 15 Clase de interfaz crearRecuento.html del módulo Control físico.

Nombre: RecuentoController		
Tipo de clase: Controladora		
Descripción: Es la encargada de gestionar los datos de un recuento.		
Atributo	**Tipo**	**Descripción**
Recuento	Recuento	Posee los datos de un recuento.
controlFisicoService	Service	Se utiliza para guardar un recuento.
Para cada responsabilidad:		
Nombre	**Descripción**	

index()	El método muestra el gsp recuento.
crearRecuentoOrdinario()	El método obtiene los parámetros del recuento.
guardarRecuento()	El método guarda el recuento y después muestra el gsp recuento.

Tabla 16 Clase controladora RecuentoController del módulo Control físico.

Nombre: ControlFisicoService		
Tipo de clase: Service		
Descripción: Es donde se realiza la lógica del negocio		
Atributo	**Tipo**	**Descripción**
Para cada responsabilidad:		
Nombre	**Descripción**	
tiposRecuento()	El método muestra una lista de los tipos de recuentos.	
getTipoRecuento()	El método obtiene el id de un recuento.	
guardarRecuento()	El método guarda el recuento.	
getItems()	El método obtiene una lista de recuentos.	
add()	El método adiciona un recuento si sus datos son válidos.	
getProps()	El método muestra los nombres de los campos de una tabla.	

Tabla 17 Clase de servicio ControlFísicoService del módulo Control físico.

Nombre: recuento.js		
Tipo de clase: JavaScript		
Descripción: Se validan los datos introducidos en la clase crearRecuento.html.		
Atributo	**Tipo**	**Descripción**
Para cada responsabilidad:		
Nombre	**Descripción**	
guardarRecuento()	El método guarda un recuento dado el id si el ingreso de sus datos es válido.	
validar ()	El método valida los datos registrados.	
resetColors()	El método cambia el color del borde de un campo que no sea válido.	

Tabla 18 Clase JavaScript recuento del módulo Control físico.

2.6.2. Diagramas de clases del diseño

Los diagramas de clase de diseño describen gráficamente las descripciones de las clases del sistema y sus interfaces, así como las relaciones que existen entre ellas. Este es el diagrama principal del diseño de un sistema, es un diagrama de estructura estática donde se muestran las clases del sistema y sus interrelaciones.Los diagramas contienen la siguiente información:

- Interfaces

- Atributos
- Métodos
- Dependencias
- Clases y sus asociaciones
- Información sobre los tipos de los atributos

Para el presente documento se seleccionaron cuatro casos de usos, a través de los cuales se mostrará el ciclo de desarrollo. Los casos de uso son: CRUD-RD Vehículo, Gestionar medios técnicos, CRUD-R Registrar recuento ordinario y Gestionar ocupación.

A continuación se muestran los diagramas de clases del diseño de los casos de uso antes mencionados. Para ver el resto de los diagramas de clases, ver el Anexo 1.

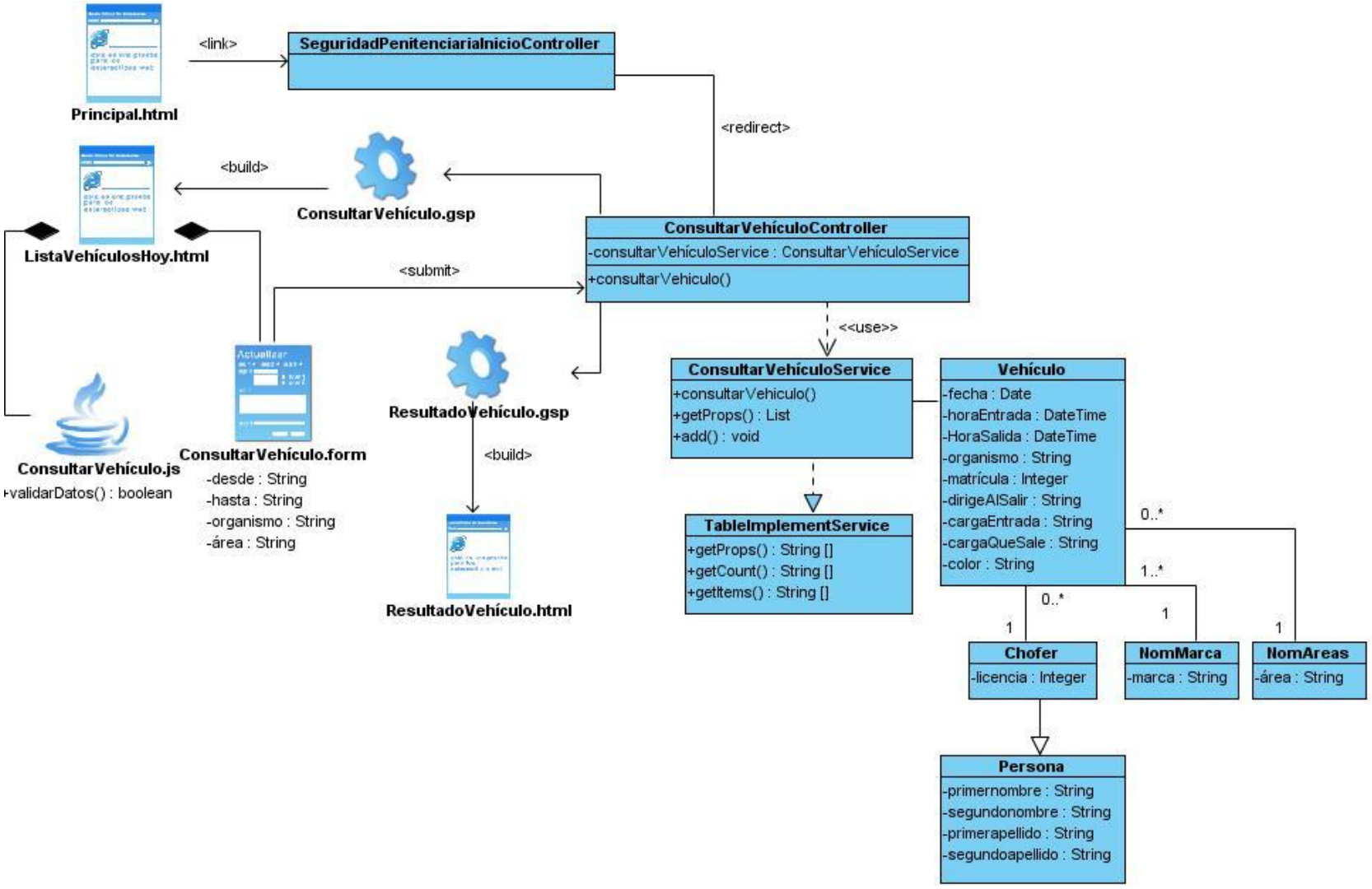

Figura 4 DCD[16] del caso de uso CRUD-RD Vehículo del módulo Control de acceso.

[16] Diagrama de clases del diseño

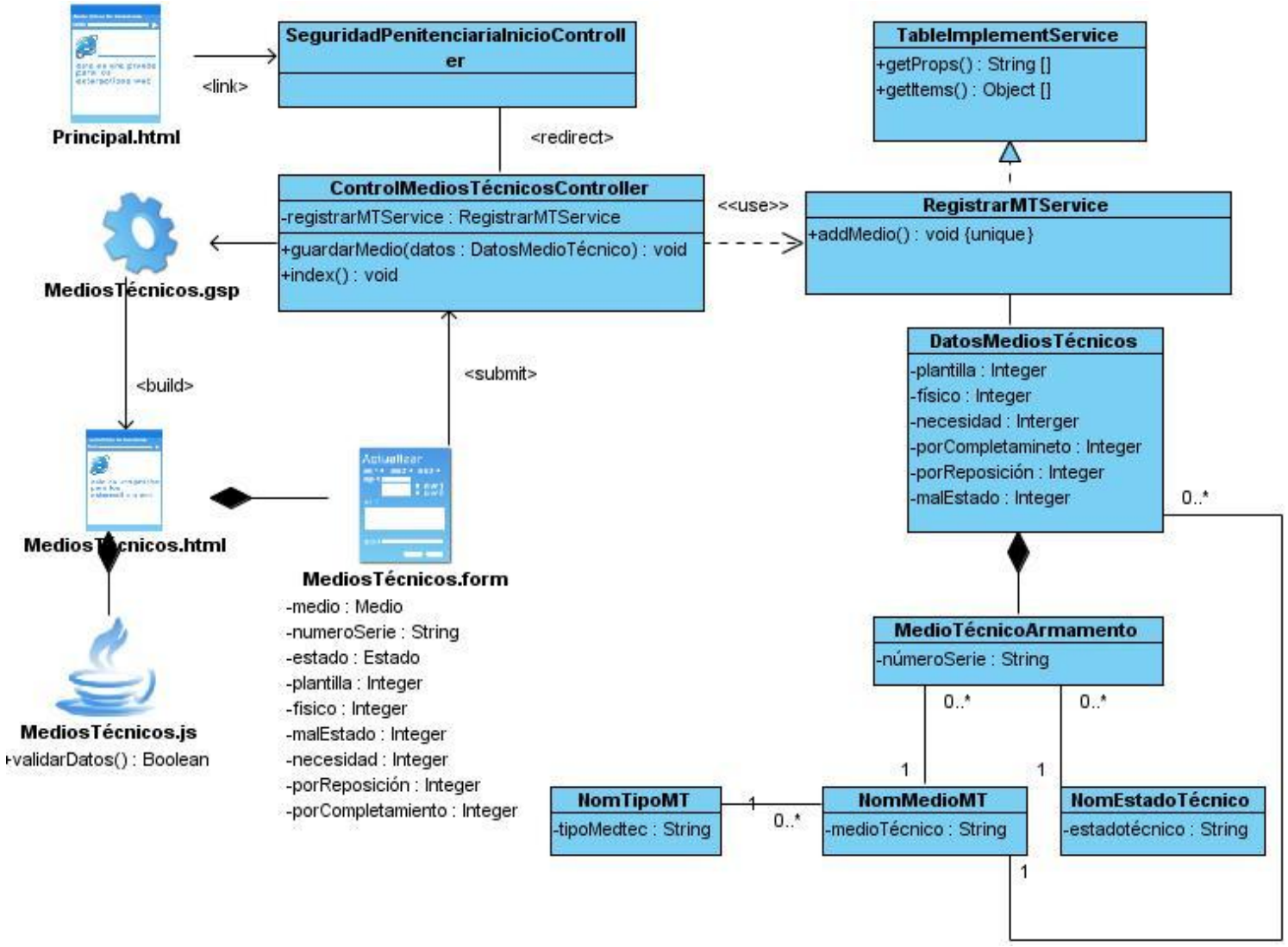

Figura 5 DCD del caso de uso Gestionar medios técnicos del módulo Control de medios técnicos.

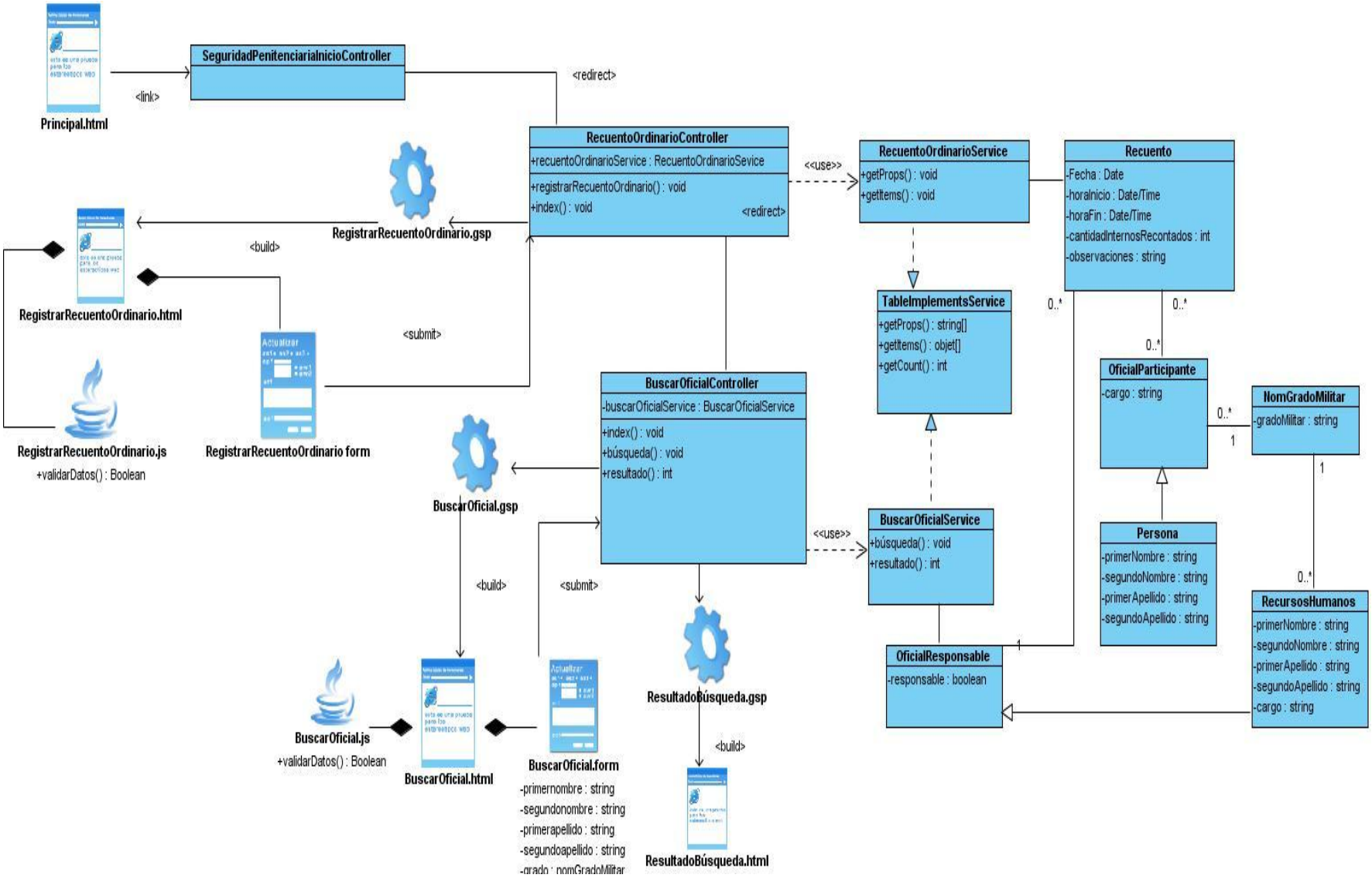

Figura 6 DCD del caso de uso Registrar recuento ordinario del módulo Control físico.

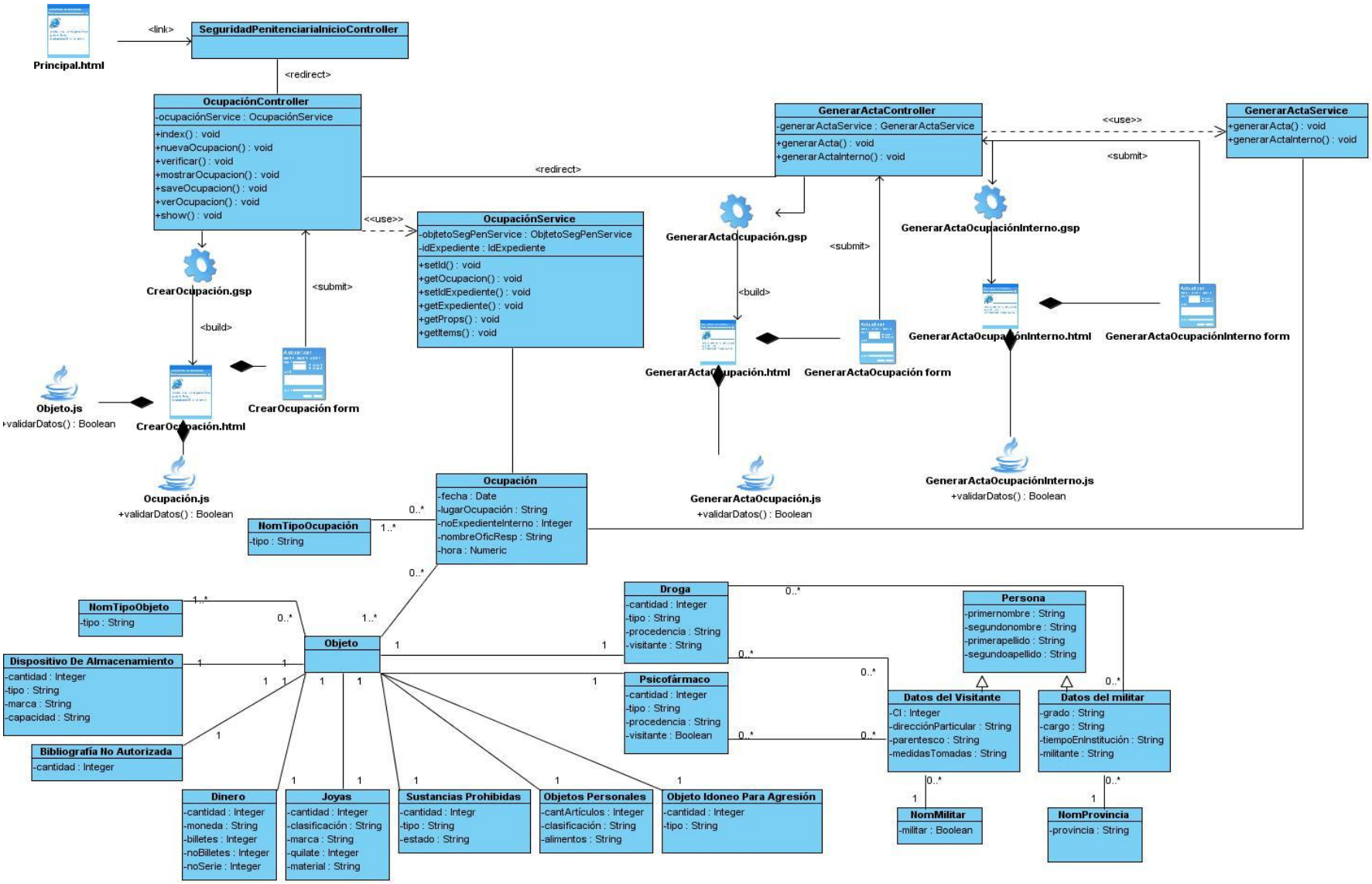

Figura 7 DCD del caso de uso Registrar ocupación del módulo Control Ocupaciones, decomisos y hallazgos.

2.7. Diagramas de Interacción

A continuación se muestran los diagramas de colaboración del caso de uso: CRUD-RD Vehículo Sección1 del módulo Control de acceso. Para ver el resto de los diagramas de interacción, ver el Anexo 2.

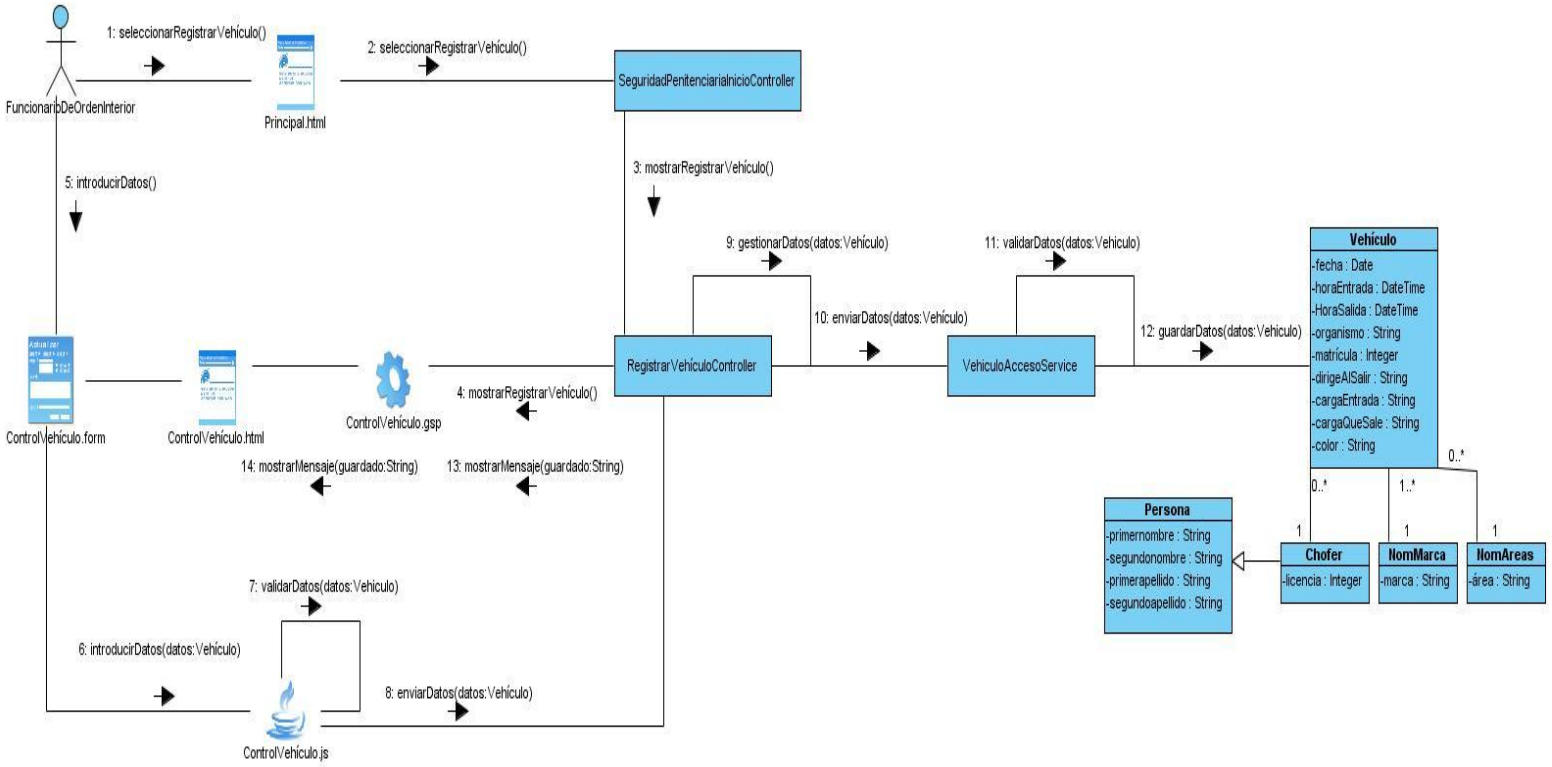

Figura 8 DC[17]del caso de uso CRUD-RD Vehículo Sección1 del módulo Control de acceso.

[17]Diagrama de colaboración

2.8. Diseño de la Base de Datos

Un Diagrama de Entidad Relación (DER) es una herramienta utilizada para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. Teniendo en cuenta los datos de las diferentes tablas contenidas en los DER correspondientes a los módulos Control de acceso, Control físico, Control de medios técnicos y Ocupaciones, decomisos y hallazgos se puede afirmar que se encuentran en Segunda Forma Normal (2FN), ya que están en la Primera Forma Normal (1FN) y, además, no existen dependencias parciales.

Para lograr obtener DER acordes a la solución que se propone y que brinden una organización que facilite su interpretación es necesaria la utilización de patrones de BD. Estos patrones constituyen una guía que especifica cómo debe ser la BD.

En el presente trabajo se hizo del patrón Árbol fuertemente codificado. Este patrón plantea esencialmente que a cada nodo se le asocia una entidad. En términos de BD, las relaciones que existen entre las tablas pueden ser de uno a uno, de uno a muchos, de muchos a muchos, etc. en este caso las relaciones son de uno a muchos (n). El patrón Árbol fuertemente codificado es utilizado para representar jerarquías donde es bien conocida la estructura, es importante representar la correspondencia, por ejemplo las estructuras organizacionales. (18)

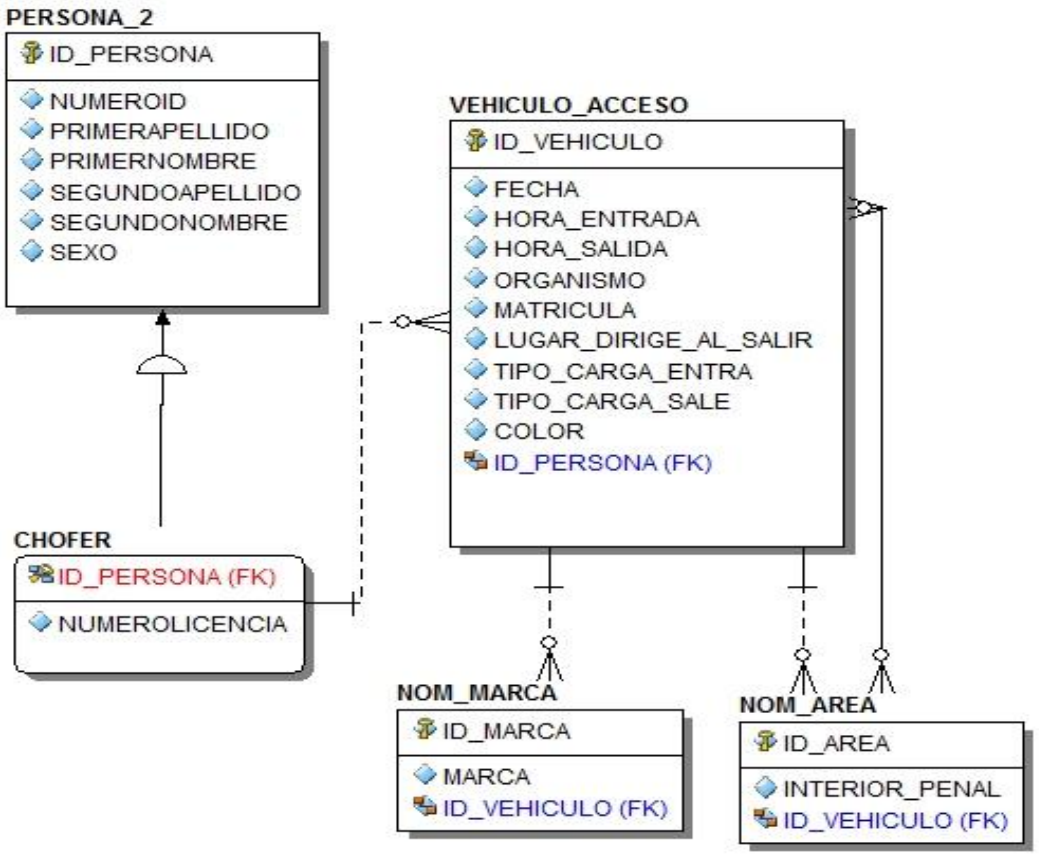

Figura 9 Modelo de datos del módulo Control de acceso.

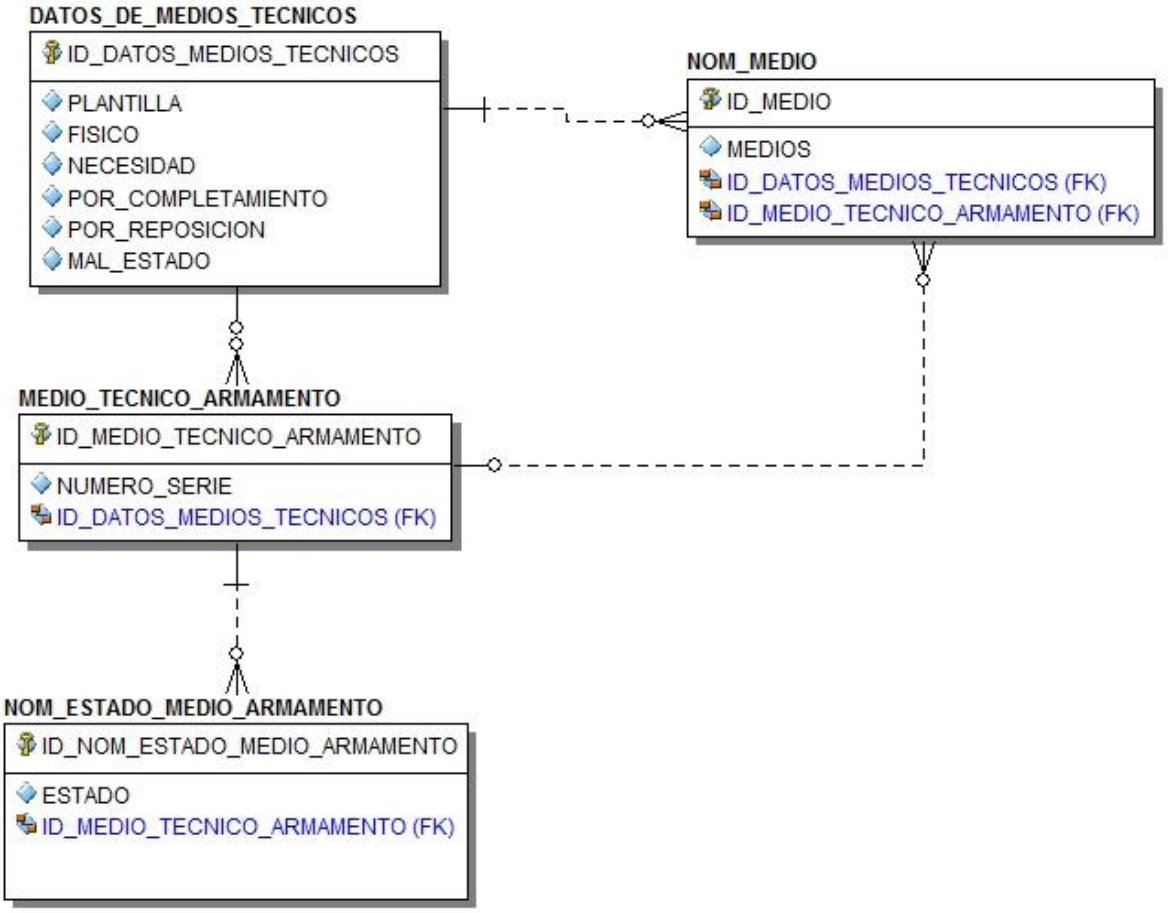

Figura 10 Modelo de datos del módulo Control de medios técnicos.

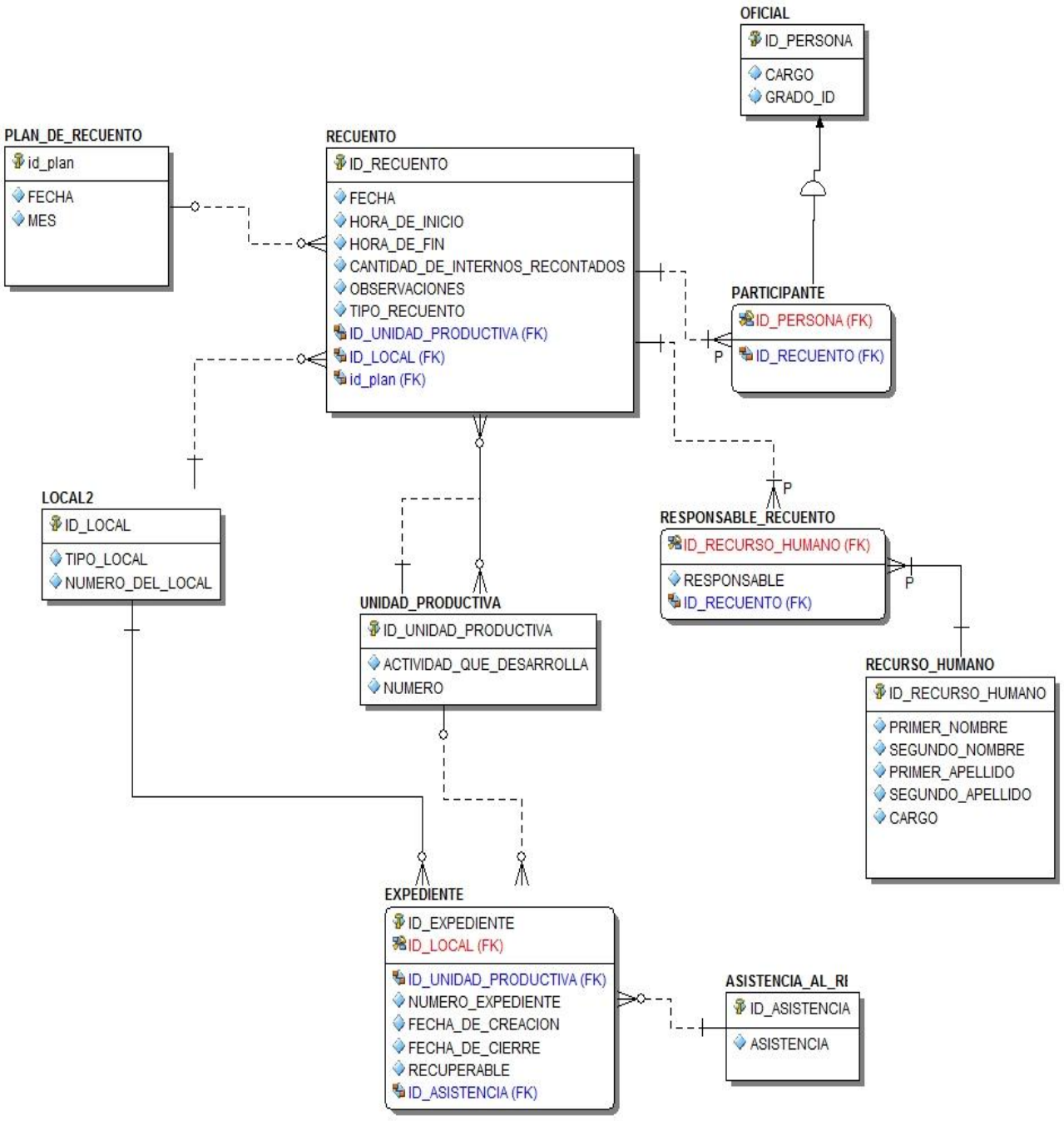

Figura 11 Modelo de datos del módulo Control físico.

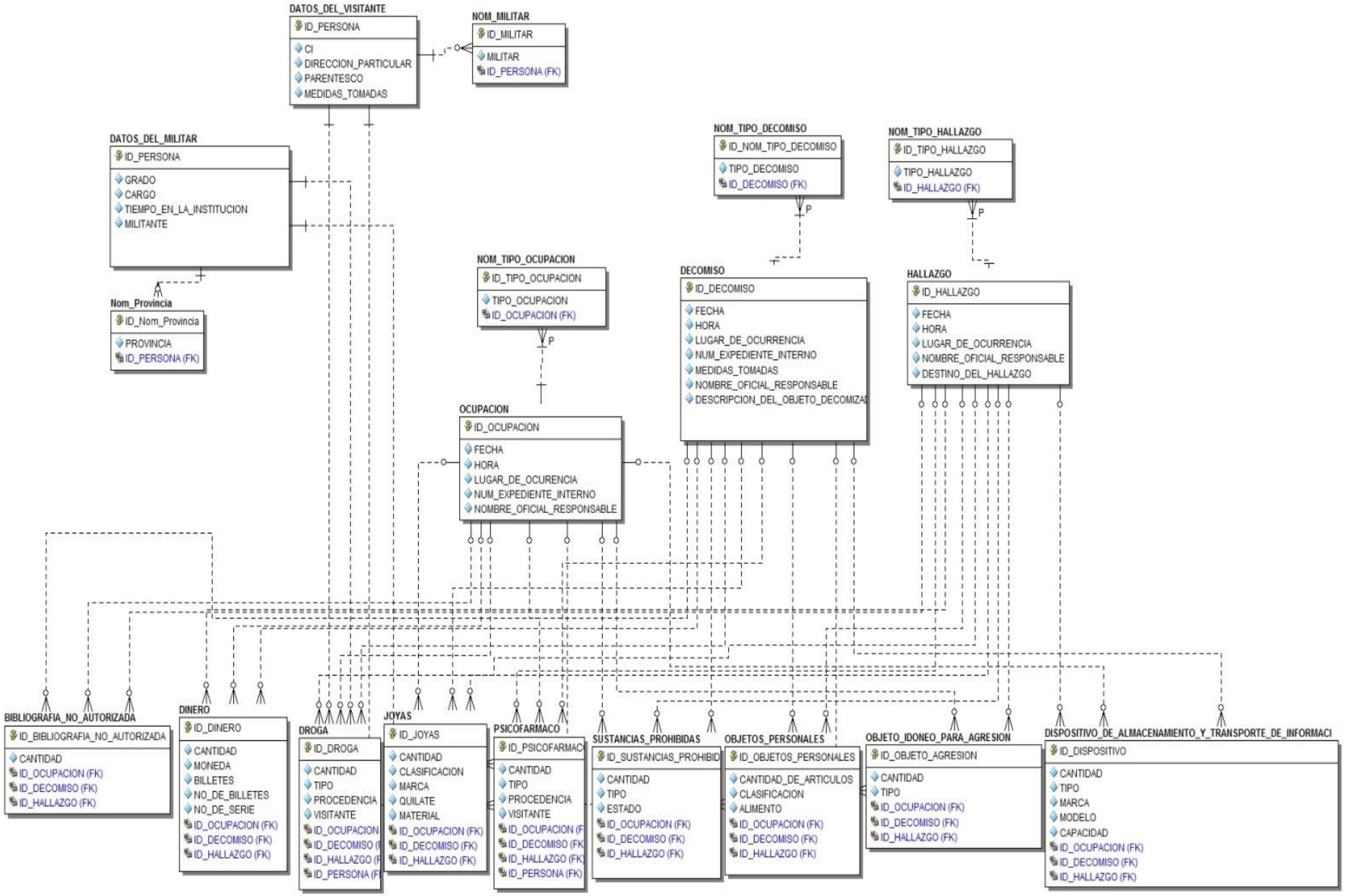

Figura 12 Modelo de datos del módulo Ocupaciones, decomisos y hallazgos.

2.9. Descripción de las tablas.

A continuación de describen los datos de las tablas más significativas de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos.

Vehículo_acceso		
Descripción: Almacena los datos de los vehículos que visiten al Centro Penitenciario.		
Atributo	**Tipo**	**Descripción**
Id	NUMERIC	Identificador de la tabla.
Fecha	DATE	Guarda la fecha en que el chofer visitó al centro.
Hora Entrada	NUMERIC	Almacena la hora en que entró el chofer al centro.
HoraSalida	NUMERIC	Almacena la hora en que salió el chofer al centro.
Marca	VARCHAR	Guarda la marca del vehículo.
Color	VARCHAR	Guarda el color del vehículo.
Organismo al que pertenece	VARCHAR	Guarda el organismo al que pertenece el vehículo.
Número de matrícula	NUMERIC	Guarda el número de la matrícula del vehículo.
Área	VARCHAR	Guarda las áreas que tenga registrado el centro penitenciario según el módulo Aseguramiento Logístico.
Lugar al que se dirige al salir	VARCHAR	Guarda el lugar al que se dirige después de salida.
Tipo de carga que entra	VARCHAR	Guarda el tipo de carga con que entra el vehículo
Tipo de carga que sale	VARCHAR	Guarda el tipo de carga con que sale el vehículo
Primer Nombre	VARCHAR	Guarda el primer nombre del chofer que visitó el centro.
Segundo Nombre	VARCHAR	Guarda el segundo nombre del chofer que visitó el centro.
Primer apellido	VARCHAR	Guarda el primer apellido del chofer que visitó el centro.
Segundo apellido	VARCHAR	Guarda el segundo apellido del chofer que visitó el centro.

Número de Licencia	NUMERIC	Guarda el número de licencia del chofer que visitó el centro.

Tabla 19 Tabla Vehículo_acceso del módulo Control de acceso.

Datos de medios técnicos		
Descripción: Almacena los datos de un tipo de medio en específico.		
Atributo	**Tipo**	**Descripción**
Id	NUMERIC	Identificador de la tabla.
Medio	VARCHAR	Este campo guarda el nombre del medio.
Plantilla	NUMERIC	Guarda la cantidad de medios que debe tener el centro en plantilla.
Físico	NUMERIC	Guarda la cantidad física de medios existentes en el Centro.
Necesidad	NUMERIC	Guarda la cantidad de medios que faltan según lo que está en plantilla con lo que está físicamente
Por Completamiento	NUMERIC	Guarda la cantidad de medios que faltan por completamiento.
Por Reposición	NUMERIC	Guarda la cantidad de medios que faltan por reposición.

Tabla 20 Tabla Datos de medios técnicos del módulo Control de medios técnicos.

Recuento		
Descripción: Almacena los datos de los recuentos.		
Atributo	**Tipo**	**Descripción**
Id	NUMERIC	Identificador de la tabla.
Tipo	VARCHAR	Guarda el tipo de recuento.
Fecha	DATE	Guarda la fecha en que se realizó el recuento.
Hora de inicio	NUMERIC	Almacena la hora en que se inició el recuento.
Hora de fin	NUMERIC	Almacena la hora en que finalizó el recuento.
Cantidad de internos recontados	NUMERIC	Guarda la cantidad de internos contados en el recuento.

Observaciones	VARCHAR	Guarda los resultados del recuento.

Tabla 21 Tabla Recuento del módulo Control físico.

Decomiso		
Descripción: Almacena los datos de los decomisos que se les realizan a los internos.		
Atributo	**Tipo**	**Descripción**
Id	NUMERIC	Identificador de la tabla.
Fecha	DATE	Fecha en que se realiza un decomiso.
Hora	NUMERIC	Hora exacta en que se realizó el decomiso.
Lugar de ocurrencia	VARCHAR	Lugar dónde se realizó el decomiso.
Tipo de decomiso	VARCHAR	Selección del tipo de decomiso realizado.
No. Expediente Interno	NUMERIC	Número de expediente que identifica al interno.
Medidas tomadas	VARCHAR	Descripción de las medidas tomadas con el interno al que se le realizó un decomiso.
Nombre Oficial responsable	VARCHAR	Nombre y apellidos del oficial responsable del decomiso.
Descripción del objeto decomisado	VARCHAR	Descripción detallada del objeto decomisado

Tabla 22 Tabla Decomiso del módulo Ocupaciones, decomisos y hallazgos.

2.10.Conclusiones parciales

En el presente capítulo se realizó el modelo de diseño de los módulos Control físico, Control de medios técnicos, Control de acceso y Ocupaciones, decomisos y hallazgos del SIDEP, específicamente los diagramas de clases, los diagramas de interacción y el diseño de la BD, utilizando como soporte la arquitectura, los patrones de diseño y la descripción de los casos de uso de los módulos, lo que permitió que estos artefactos estuvieron orientados a una solución eficiente y sentar las bases para la posterior implementación de dichos módulos.

Capítulo III. Implementación y Prueba

3.1. Introducción

En el presente capítulo se explican los diagramas correspondientes a la disciplina de Implementación de los módulos Control físico, Control de medios técnicos, Control de acceso y Ocupaciones, decomisos y hallazgos, específicamente los diagramas de despliegue y componentes. Además, se describen las pruebas que se le realizarán a los módulos mencionados con el objetivo de comprobar la eficiencia de los mismos en integración con el sistema.

3.2. Implementación

La implementación define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

Para la implementación de los módulos Control físico, Control de medios técnicos, Control de acceso y Ocupaciones, decomisos y hallazgos se realizan los diagramas de despliegue y los de componentes.

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. También se pueden utilizar para mostrar la estructura de alto nivel del modelo de implementación en términos de subsistemas de implementación, y las relaciones entre elementos de implementación.

La distribución de componentes se realizó por paquetes coincidiendo con la arquitectura de la aplicación. Posteriormente se muestra el diagrama de componentes que representa la relación del módulo Control de acceso con los subsistemas y módulos.

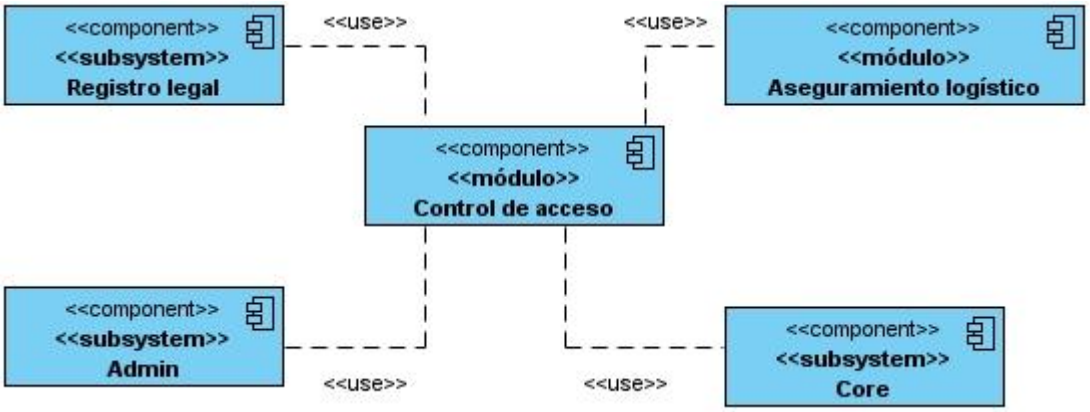

Figura 13 Diagrama de componentes general del módulo Control de acceso.

Seguidamente se muestra la relación que existe entre los componentes que conforman el módulo Control de acceso.

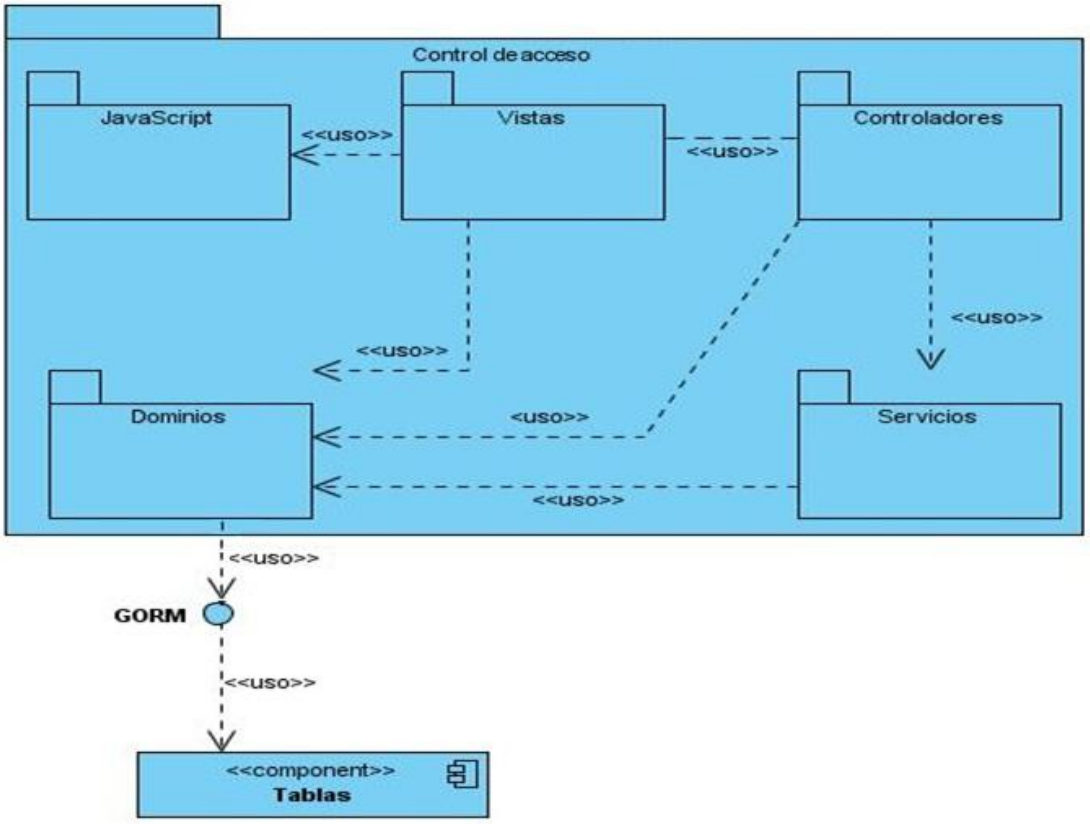

Figura 14 Diagrama de componentes del módulo Control de acceso.

A continuación se detallan los componentes por capas utilizados en el caso de uso CRUD-RD Vehículo correspondiente al módulo Control de acceso.

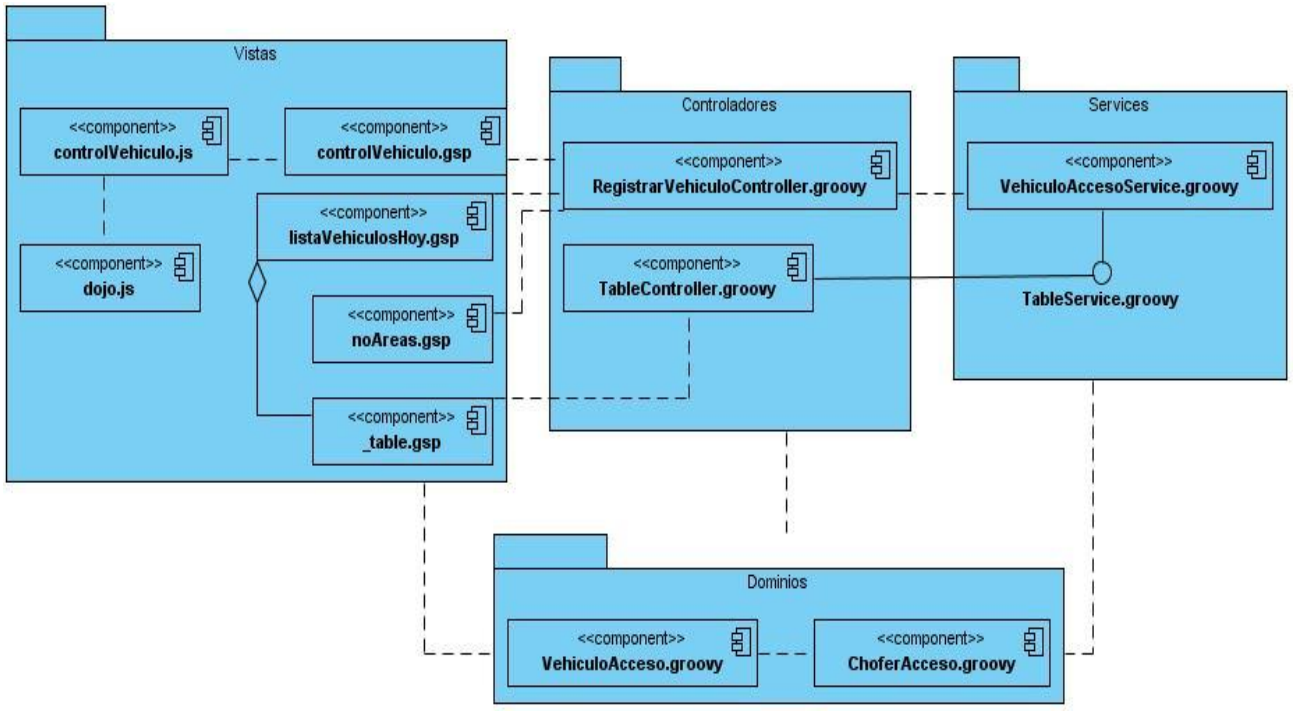

Figura 15 Diagrama de componentes del caso de uso CRUD-RD Vehículo del módulo Control de acceso.

3.2.2. Modelo de despliegue

El modelo de despliegue describe la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos.

Para el correcto despliegue del módulo se necesita un cuarto tecnológico, donde radicarán el Servidor de aplicaciones y el de BD. El servidor de aplicaciones trabaja sobre la distribución de Linux Suse Enterprise Server 10.0 SP2, y el servidor web Apache Tomcat 6.0.25. El servidor de BD también tendrá instalado la distribución de Linux Suse Enterprise Server 10.0 SP2 y tendrá como sistema gestor de base de datos Oracle 11.0.1G Enterprise Edition. El cliente hará uso de la aplicación a través de clientes ligeros con sistema operativo Windows XP SP3, que estarán conectados a un servidor de clientes ligeros.

Teniendo en cuenta que se despliegan los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos, los componentes físicos que intervienen en los procesos de gestión de la información se distribuyen de la siguiente manera: las vistas estarán incluidas en el nodo de la PC cliente, los controladores y los servicios estarán en el nodo del servidor web y las tablas estarán en el servidor de base de datos. El modelo de despliegue se presenta a continuación.

Figura 16 Modelo de despliegue de los módulos Control físico, Control de acceso, Control de medios técnicos y Ocupaciones, decomisos y hallazgos.

Para el despliegue del módulo es necesario que se cumplan las siguientes especificaciones:

Puestos de trabajo (PC cliente):

- Navegador web Mozilla Firefox 3.6.*
- RAM: 512 MB

Servidor de la aplicación (Servidor web):

- JDK 1.6
- Apache Tomcat 6.0.3
- Linux Suse Enterprise Server 10.0 SP2
- Microprocesador: 4 núcleos, 3 GHz
- RAM: 4 GB
- Espacio necesario para la instalación: 250 MB
- Espacio libre requerido: 250 GB

Servidor de Bases de Datos:

- JDK 1.6
- Oracle 11g
- Microprocesador: 4 núcleos, 3 GHz
- RAM: 4 GB
- Espacio necesario para la instalación: 2 GB
- Espacio libre requerido: 1 TB

3.2.3. Seguridad

La seguridad de los módulos Control físico, Control de acceso, Control de medios técnicos y Ocupaciones, decomisos y hallazgos se logró mediante el empleo de Spring Security, el cual proporciona servicios integrales de seguridad para J2EE. Spring Security es un framework altamente adaptable y potente para la autentificación y el control de acceso. (19)

Las principales áreas de seguridad son la autenticación y autorización. La autenticación es el proceso en el que la aplicación pueda o no dar acceso a realizar una acción en la aplicación. La autorización es el proceso que permite decidir si un usuario tiene autorización a realizar una acción dentro de la aplicación. Para lograr la autorización se le dio un código a cada funcionalidad, luego se crean los roles y a estos se les asignan las funcionalidades que podrá realizar ese rol, por último en los controladores mediante una notación de seguridad se aseguran las acciones que se pueden realizar sobre ese controlador. Con este mecanismo se logra que el usuario para poder acceder a una determinada acción debe de tener asignado el rol que corresponda con la acción a realizar sobre la aplicación. A continuación se muestra la notación de seguridad de la clase controladora RegistrarVehiculoController del módulo Control de acceso.

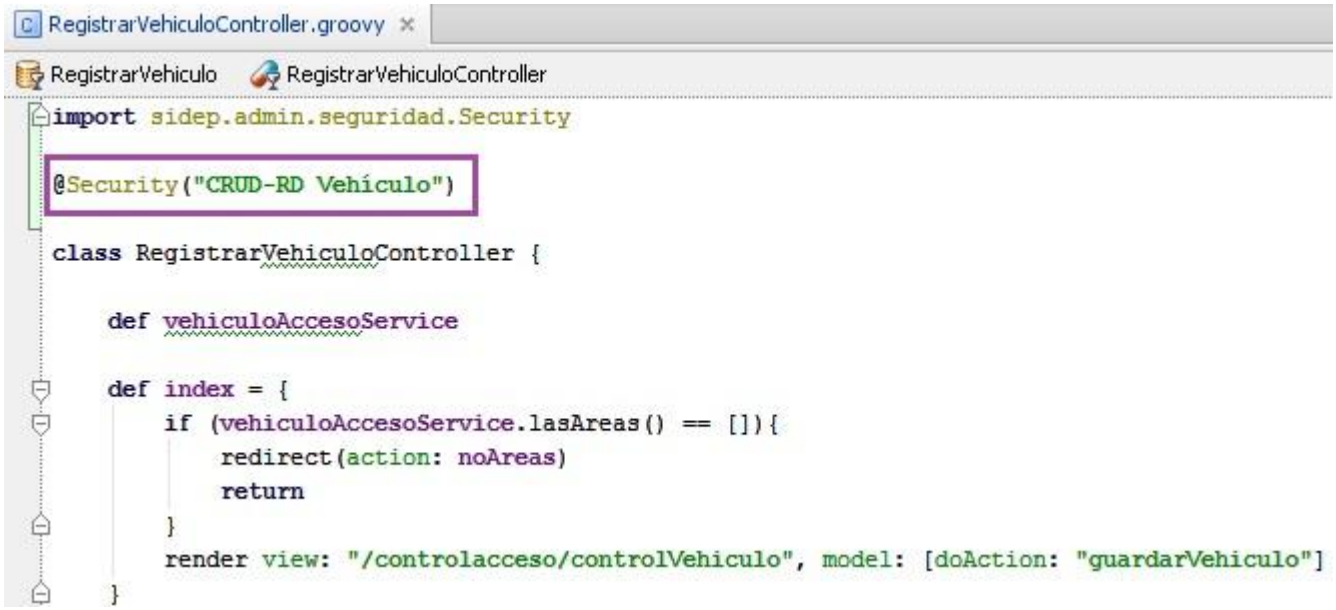

Figura 17 Ejemplo de notación de seguridad de la clase controladora RegistrarVehiculoController del módulo Control de acceso.

3.2.4. Internacionalización

La internacionalización posibilita a una aplicación que la interfaz de usuario soporte distintos idiomas. Al emplear como framework Grails, se creará un fichero para cada idioma que soporte la aplicación en la carpeta grails-app/i18n y en estos ficheros se escriben los mensajes para cada componente. Los nombres de estos archivos empiezan con Messages y luego el Locale del idioma; por ejemplo:

- Messages_es.properties para los mensajes en español.
- Messages_en.properties para los mensajes en inglés.

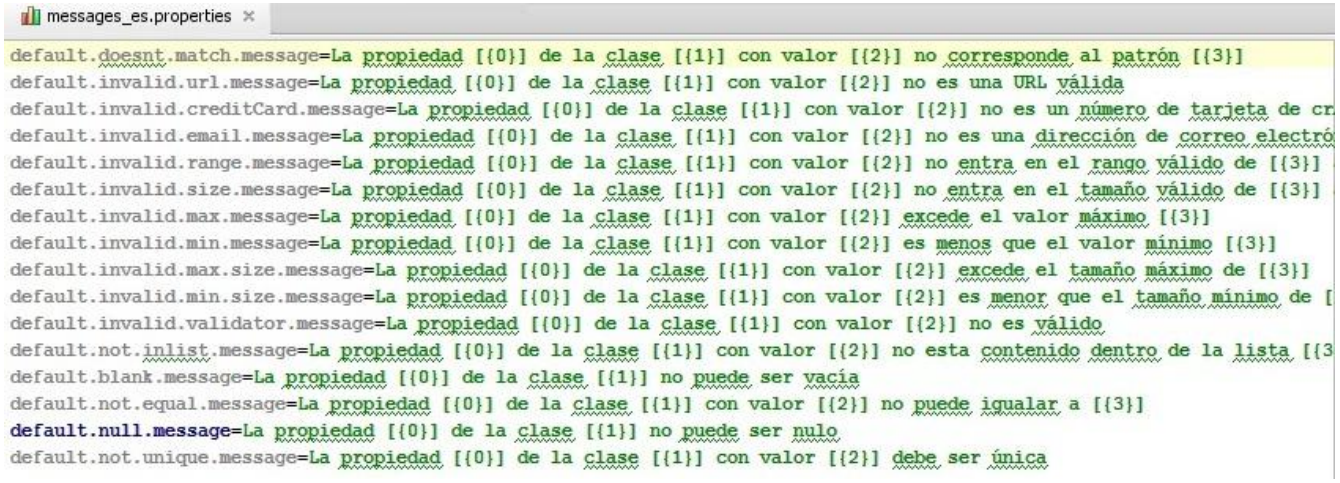

Figura 18 Fichero para los mensajes en español.

3.3. Pruebas

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un sistema informático. Básicamente es una fase en el desarrollo de software, consistente en probar las aplicaciones construidas. Para determinar el nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema.

3.3.1. Estrategia de prueba

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

A continuación se especifica y explica la estrategia seleccionada para las pruebas de los módulos Control físico, Control de acceso, Control de medios técnicos y Ocupaciones, decomisos y hallazgos:

- Método de prueba: método de caja negra.

- Tipo de prueba: funcionalidad.
- Nivel de prueba: prueba de desarrollador.
- Tipo de técnica de prueba: casos de prueba.

Método de caja negra: pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código). (20) El probador proporciona las entradas y estudia las salidas para ver si son o no las esperadas.

Las pruebas de caja negra se centran principalmente en los requisitos funcionales del software y permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento. (20)

Dentro del método de caja negra, la técnica de la partición de equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar. (21)

A pesar de que la técnica partición de equivalencia es la más usada, se empleó el tipo de prueba de funcionalidad basada en casos de prueba debido a que Calisoft, que es la entidad responsable de las pruebas de liberación de los productos informáticos de la UCI, realiza los procesos de prueba haciendo uso de esta técnica en específico, por lo que esta es la que se pone en práctica en el proyecto Prisiones Cuba para determinar la calidad de los procesos que se desarrollan en los diferentes subsistemas.

Funcionalidad: prueba centrada en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos. (22) Se basa específicamente en la ejecución y revisión de las funcionalidades previamente diseñadas para el sistema, estas pruebas se hacen mediante el diseño de modelos de pruebas que buscan evaluar cada una de las opciones con las que cuenta el software. Este tipo de prueba garantiza la función, seguridad y volumen. El factor fundamental que se tiene en cuenta es la capacidad de la aplicación para manejar grandes volúmenes de datos.

Prueba de desarrollador: esta prueba indica los aspectos más adecuados de diseño e implementación de la prueba 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.

Es recomendable que la prueba de desarrollador no cubra únicamente unidades independientes de prueba aisladas.

Casos de prueba: cuando se diseñan los casos de prueba (CP) se tienen las siguientes características:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.

Con los diseños de casos de prueba realizados, se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. A continuación se muestra el CP Registrar vehículo del módulo Control de acceso.

Escenario	Descripción	Fecha	Hora entrada	Hora salida	Marca	Color	Organismo al que pertenece	Número de matrícula	Área	Lugar al que se dirige al salir	Tipo de carga que entra	Tipo de carga que sale	Primer nombre	Segundo nombre	Primer apellido	Segundo apellido	Número de Licencia	Respuesta del sistema	Flujo central
EC 1.1 Registrar exitosamente un vehículo	Se registra un vehículo	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	El sistema almacena los datos del estado en la BD, dejando plasmados los datos del registro.	1.Seleccionar la opción "Control acceso" en el menú. 2.Se llenan los datos. 3.Se oprime el botón Aceptar.
EC 1.2 Cancelar	Se cancela la operación de registrar un vehículo	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	El sistema regresa a la vista de la aplicación donde permite volver a registrar recuento ordinario	1.Seleccionar la opción "Control acceso" en el menú. 2. Se oprime el botón "Cancelar".
EC 1.3 Faltan datos obligatorios	Exiten campos obligatorios que no se han llenado	I	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	El sistema muestra un mensaje "Introduzca los datos obligatorios" indicando los campos obligatorios que faltaron por llenar.Posteriormente el sistema almacena los datos del estado en la BD, dejando plasmados los datos del registro.	1.Seleccionar la opción "Control acceso" en el menú. 2.Se llenan los datos. 3.Se dejan campos obligatorios sin llenar 4.Se oprime el botón Aceptar.
EC 1.4 Datos incorrectos	Exiten campos incorrectos	V	I	V	V	V	V	V	V	V	V	V	V	I	V	V	V	El sistema muestra el mensaje de error "Corrija los datos erróneos introducidos" indicando los campos incorrectos.Posteriormente el sistema almacena los datos del estado en la BD, dejando plasmados los datos del registro del vehiculo.	1.Seleccionar la opción "Control acceso" en el menú. 2.Se llenan los datos. 3.Se llenan campos incorrectos. 4. Se oprime el botón Cerrar.

Figura 19 CP Registrar vehículo del módulo Control físico.

3.3.2. Resultados de la aplicación de la prueba de caja negra

A continuación se describe las no conformidades detectadas en las pruebas de caja negra realizadas a las funcionalidades del módulo Control de acceso, teniendo en cuenta que se realizaron 2 iteraciones.

No	No conformidad	Ubicación	Estado
1	El botón Consultar no funciona.	controlacceso/consultarVehiculo.gsp	Resuelta
2	El botón Guardar no funciona.	controlacceso/controlVehiculo.gsp	Resuelta
3	El label del select Área aparece sin tilde.	controlacceso/controlVehiculo.gsp	Resuelta
4	El JavaScript ControlVehiculo.js no está validando correctamente.	controlacceso/controlVehiculo.gsp	Resuelta
5	El select áreas no muestra sus valores.	controlacceso/controlVehiculo.gsp	Resuelta
6	La tabla no lista los vehículos.	controlacceso/listaVehiculosHoy.gsp	Resuelta

Tabla 23 Tabla de no conformidades (Primera iteración) del módulo Control de acceso.

No	No conformidad	Ubicación	Estado
1	El input Matrícula solo acepta letras.	controlacceso/controlVehiculo.gsp	Resuelta
2	El label del input Matrícula no tiene acento.	controlacceso/controlVehiculo.gsp	Resuelta

Tabla 24 Tabla de no conformidades (Segunda iteración) del módulo Control de acceso.

A continuación se muestra un gráfico donde se evidencian los resultados obtenidos durante las 2 iteraciones de pruebas realizadas al módulo Control de acceso.

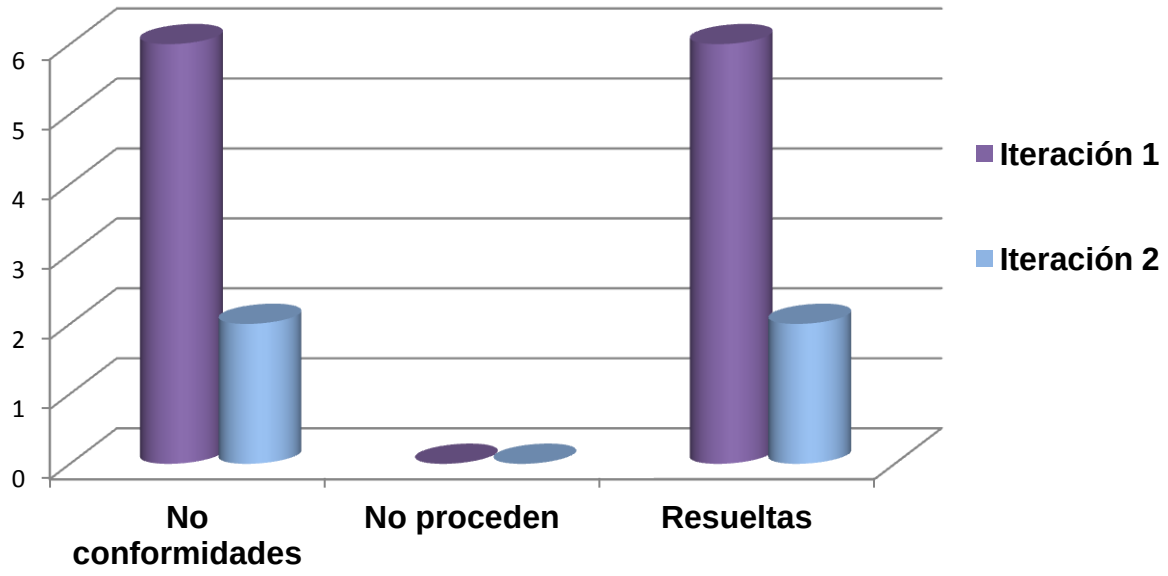

Figura 20 Resultados obtenidos en las pruebas al módulo Control de acceso.

3.4. Conclusiones parciales

En este capítulo se realizó la implementación de la solución correspondiente a los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP, además se realizaron los diagramas de componentes y el modelo de despliegue, lográndose satisfacer las necesidades del cliente de acuerdo a los requisitos capturados en flujos de trabajo anteriores.

Se realizaron pruebas a los módulos antes mencionados lo que permitió detectar deficiencias en la implementación. Estas deficiencias fueron resueltas, propiciando la obtención de una solución acorde a las especificaciones planteadas por el cliente.

Conclusiones generales

Luego de finalizar la investigación y obtener la solución correspondiente a los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP se concluye lo siguiente:

- El análisis de las aplicaciones desarrolladas como soluciones para sistemas penitenciarios existentes permitió reconocer la necesidad de desarrollar dichos módulos como parte integradora del SIDEP, teniendo en cuenta las herramientas, tecnologías y lenguajes más acordes a las necesidades del cliente.
- Los diagramas y el modelo de diseño obtenidos proporcionaron la plataforma para la posterior implementación de una solución orientada a las especificaciones propias de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP.
- Se logró realizar la implementación de una solución conforme a los requisitos previstos con el cliente anteriormente, incorporando dichos módulos como funcionalidades internas del SIDEP.
- Con la aplicación de las técnicas y métodos de pruebas se logró detectar y, posteriormente resolver, las deficiencias existentes en la solución ya implementada.

De esta forma se le dio cumplimiento al objetivo general de desarrollar los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos del SIDEP a partir del análisis de los requisitos de software establecidos con el cliente y haciendo uso de la arquitectura y las tecnologías definidas por el proyecto.

Recomendaciones

Apoyándonos en la investigación realizada y en los resultados obtenidos durante la elaboración de este trabajo, se recomienda lo siguiente:

- Desplegar la aplicación en los centros penitenciarios para determinar si es necesario agregar nuevas funcionalidades que brinden mejor información y servicios al usuario.
- Desarrollar los manuales de usuario de los módulos Control de medios técnicos, Control de acceso, Control físico y Ocupaciones, decomisos y hallazgos para capacitar al personal que hará uso de estos en la aplicación posteriormente.
- Realizar las pruebas de aceptación para obtener un producto que satisfaga completamente los requisitos capturados con el cliente.

Referencias bibliográficas

1. **Carrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David.** *Metodología de desarrollo de software.* 2008.

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

3. Visual Paradigm International. [En línea] [Citado el: 11 de Diciembre de 2011.] http://www.visual-paradigm.com.

4. **SG Software Guru.** Revista digital SGuía. *Embarcadero E/R Studio.* [En línea] 2009. [Citado el: 15 de Enero de 2012.] http://sg.com.mx/guia/node/1454.

5. SpringSource. [En línea] [Citado el: 20 de Enero de 2012.] http://www.springsource.com/developer/sts.

6. **The Apache Software Foundation.** Apache Tomcat. [En línea] 1999. [Citado el: 25 de Enero de 2012.] http://tomcat.apache.org.

7. —. Subversion. [En línea] [Citado el: 28 de Enero de 2012.] http://subversion.apache.org.

8. Oracle. [En línea] [Citado el: 2 de Febrero de 2012.] http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html.

9. Json.org. [En línea] [Citado el: 5 de Febrero de 2012.] http://www.json.org/json-es.html.

10. **The Dojo Foundation.** Dojo Toolkit. [En línea] [Citado el: 10 de Febrero de 2012.] http://dojotoolkit.org.

11. **SpringSource and the Groovy Community.** Groovy. [En línea] [Citado el: 11 de Febrero de 2012.] http://groovy.codehaus.org.

12. librosweb.es. [En línea] [Citado el: 15 de Febrero de 2012.] http://www.librosweb.es/javascript.

13. **SpringSource.** Grails. [En línea] 2009. [Citado el: 20 de Febrero de 2012.] ttp://www.grails.org.

14. **Oracle Corporation and/or its affiliates.** Java 2 Platform . *Enterprise Edition (J2EE) Overview.* [En línea] 2010. [Citado el: 21 de Febrero de 2012.] http://java.sun.com/j2ee/overview.html.

15. **Ecured.** Singleton. [En línea] [Citado el: 1 de Abril de 2012.] http://www.ecured.cu/index.php/Patr%C3%B3n_Singleton.

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

17. **Calderón Sanmartín, Jhon Alexander.** Blog "My Space to share". [En línea] Enero de 2008. [Citado el: 15 de Abril de 2012.] http://jcalderon.wordpress.com/2008/01/04/instalacion-y-configuracion-de-apache-tomcat-60-en-windows-xp.

18. **Ecured.** Patrones de diseño de base de datos. [En línea] [Citado el: 18 de Abril de 2012.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

19. Spring-Security. [En línea] [Citado el: 22 de Abril de 2012.] http://static.springsource.org/spring-security/site.

20. **Universidad de las Ciencias Informáticas.** Conferencia 1 “Introducción al flujo de Pruebas ”. *Entorno Virtual de Aprendizaje.* [En línea] [Citado el: 4 de Mayo de 2012.] http://evapostgrado.uci.cu/mod/resource/view.php?id=12042.

21. **Ecured.** Pruebas de caja negra. [En línea] [Citado el: 15 de Mayo de 2012.] http://www.ecured.cu/index.php/Pruebas_de_caja_negra.

22. **Universidad de las Ciencias Informáticas.** RUP & ISO 9126. *Entorno Virtual de Aprendizaje.* [En línea] [Citado el: 22 de Mayo de 2012.] http://evapostgrado.uci.cu/file.php/368/Bibliografia/ISO-RUP.doc.

23. **Ecured.** Protocolo Seguro de Transferencia de Hipertexto. [En línea] [Citado el: 29 de Mayo de 2012.] http://www.ecured.cu/index.php/Protocolo_seguro_de_transferencia_de_hipertexto.

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

25. **Ecured.** SSL. [En línea] [Citado el: 29 de Mayo de 2012.] http://www.ecured.cu/index.php/SSL.

26. —. Protocolo TCP/IP. [En línea] [Citado el: 30 de Mayo de 2012.] http://www.ecured.cu/index.php/Protocolo_TCP/IP.

27. —. TLS. [En línea] [Citado el: 31 de Mayo de 2012.] http://www.ecured.cu/index.php/TLS.

Bibliografía

1. Código Penal Cubano. Ley Nro. 62. República de Cuba: s.n.

2. Dirección de establecimientos Penitenciarios (DEP), MININT. Reglamento penitenciario cubano. La Habana: s.n., 2010.

3. —. Procedimientos de trabajo Seguridad Penitenciaria. La Habana: s.n., 2010.

4. ALBET. Documento Descripción de funcionalidades: Biblioteca. Solución de Software Sistema de Gestión Penitenciaria. La Habana: s.n., 2008.

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

6. Cupertino, J.R. El Lenguaje Unificado de Modelado. California: s.n., 1998.

7. Crawford, William and Kaplan, Jonathan.J2EE Design Patterns. California: O'Reilly & Associates, 2003.

8. Brito, Nacho. Manual de desarrollo web con Grails. [Online] [Cited: 12 13, 2011.] http://www.manual-de-grails.es.

9. Luca de tenas, Juan Ignacio. Aplicaciones JavaScript. Madrid: EDlClONES ANAYA MULTIMEDIA, 2000.

10. Calahorro, Nacho Brito. Manual de desarrollo web con Grails. s.l. : Graeme Rocher, 2009. v1.0.4.

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

12. Curso de Gestión de la calidad (Curso de verano) Conferencia Tema 3: Evaluación de la calidad de los productos. 2011.

Glosario de términos

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

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

Control de acceso: es el proceso de registro y concesión de permisos para la entrada y salida de vehículos en los centros penitenciarios.

Control de medios técnicos: es el proceso de registro de armamento y medios técnicos (casco, uniforme, radio, etc.) en los centros penitenciarios.

Criteria: buscador dinámico avanzado para realizar consultas sobre el modelo de datos.

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

DOM: *Document Object Model* (Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos). Es una interfaz de programación de aplicaciones (API) que provee un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y usarlos.

Hallazgo: son los objetos que se encuentran en el centro penitenciario y no se le puede atribuir un dueño.

HTTPS: *Hyper Text Transfer Protocol Secure* (Protocolo Seguro de Transferencia de Hipertexto). Combinación del Protocolo de Transferencia de Hipertexto (*Hyper Text Transfer Protocol,* HTTP por sus siglas en inglés) con el protocolo SSL / TLS para proporcionar comunicaciones encriptados y la identificación segura de un servidor web en la red. (23)

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

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

Ocupación: son los objetos que se les extraen a los internos o familiares de estos hasta que egresan del centro penitenciario o por un tiempo determinado respectivamente.

Sistema Penitenciario: es el órgano encargado de garantizar el proceso de ejecución de la sanción de privación de libertad, de la sanción de trabajo correccional con internamiento, la medida de seguridad reeducativa de internamiento y la medida cautelar de prisión provisional. (24)

SSL: *Secure Socket Layer* (Capa de Enchufle Seguro). Servidor de correo que proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico. (25)

TCP/IP: *Transmission Control Protocol/Internet Protocol* (Protocolo de Control de Transmisión/ Protocolo de internet). Conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. (26)

TLS: *Transport Layer Security* (Seguridad de la Capa de Transporte). Es un protocolo mediante el cual se establece una conexión segura por medio de un canal cifrado entre el cliente y servidor. (27)