Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

"Desarrollo del módulo Aseguramiento Logístico del Sistema Informativo de la Dirección de Establecimientos Penitenciarios"

AUTOR:

Daryl Tejada Concepción.

TUTOR: Ing. Alejandro Santana Coteron

CO-TUTOR: Ing. Yanay Viera Lorenzo.

Ciudad de La Habana, 2012.

Año del 54 Aniversario de la Revolución.

"…hagamos el propósito de redoblar nuestro esfuerzo, y juremos ante nosotros mismos que si un día nuestro trabajo nos pareciera bueno, debemos luchar por hacerlo mejor; y si fuera mejor, debemos luchar por hacerlo perfecto, conociendo de antemano que para un Comunista nada será nunca suficientemente bueno y ninguna obra humana será jamás suficientemente perfecta".

Fidel Castro Ruz,
III Congreso del Partido Comunista de Cuba.

DECLARACIÓN DE AUTORÍA

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

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

Daryl Tejada Concepción — Autor

Ing. Alejandro Santana Coteron. — Tutor.

DATOS DE CONTACTO

Tutor: Ing. Alejandro Santana Coteron

ISEC, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: acoteron@uci.cu

Co-Tutor: Ing. Yanay Viera Lorenzo.

ISEC, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: yvieral@uci.cu

AGRADECIMIENTOS

Aprovechando esta oportunidad quisiera decirle “gracias” a todos los que han mostrado interés y preocupación por mí.

En primer lugar a toda mi familia y amigos, por haberme apoyado siempre. Gracias por su profunda confianza, por sus constantes llamadas en espera de alguna noticia, por el apoyo recibido durante mi formación profesional y a lo largo de toda mi vida, sin ustedes no lo hubiese logrado.

También quiero dar gracias a los tutores por haber tenido tanta paciencia y por su apoyo incondicional durante el desarrollo del presente trabajo.

A mis padres, a mi tío Eduardo y mi esposa por haberme brindado su ayuda incondicional en todo momento sin ningún tipo de pretexto, gracias por haber estado ahí siempre que los necesite, sin ustedes este trabajo no hubiera podido ser posible.

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.

Y no porque sea el último deja de ser especial, quiero agradecerle a esta Revolución y a nuestro Comandante Fidel Castro Ruz por darme la oportunidad de estudiar en esta magnífica universidad.

DEDICATORIA

Este trabajo va dedicado con mucho amor y cariño a mi abuelo Marcelino que desgraciadamente no va poder verme graduar, pero que le haría muy feliz ver lo que he podido lograr.

A mi madre por haberme guiado sabiamente, sin su amor, dedicación y apoyo no hubiese sido posible, todo lo que he logrado en la vida te lo debo a ti, te quiero.

A mi familia por ayudarme en momentos difíciles, a mis amigos por estar siempre presentes en lo bueno y lo malo, a mi esposa por quererme tanto, a toda la gente que me aprecia y han estado presentes en mi vida.

RESUMEN

El presente trabajo se enmarca en la investigación, diseño, implementación y prueba del módulo Aseguramiento Logístico del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP), que surge como parte del proyecto de humanización penitenciaria de Cuba.

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.

Este trabajo permitirá a la dirección de establecimientos penitenciarios la gestión de procesos referentes al control de las instalaciones, capacidades y medios técnicos de los establecimientos penitenciarios cubanos.

Índice

Índice de Figuras

Introducción

El Ministerio del Interior, en aras de lograr una mejor gestión en los procesos de la Dirección de Establecimientos Penitenciarios (DEP), en 1989 da sus primeros pasos en la informatización de algunos aspectos como el registro de los datos principales del recluso y los principales aspectos de control penal. La informatización en la DEP logra su mayor auge con la implementación de la orden 43\99 del Vice Ministro Primero del Ministerio del Interior. En el año 2002 culmina el desarrollo del Sistema Automatizado para el Control del Recluso (SACORE), pero no fue hasta enero del 2003 que empezó a usarse dicho sistema. El SACORE posee en estos momentos tres módulos principales: Control Penal, Reeducación Penal y Orden Interior, lo cual le permite recoger los principales datos del recluso y los aspectos penales existentes. En el tiempo que lleva de explotación dicho sistema, a pesar de las facilidades que brinda, es todavía insuficiente pues le quedan áreas por ser informatizadas como son los equipos multidisciplinarios y servicios médicos. A la par del SACORE en los centros penitenciarios se utilizan otros dos sistemas como es 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).

El SACDEP se desarrolló con el objetivo de informatizar los procesos referentes al aseguramiento logístico de la DEP, los cuales son: llevar el control de las capacidades, la infraestructura, la seguridad, los medios técnicos, el transporte, la red hidráulica, la red eléctrica, la telefonía y la radio.

La Universidad de las Ciencias Informáticas (UCI) asumió la modernización informática del Sistema Penitenciario Cubano a través del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP). El SIDEP integraría de forma única los tres sistemas que existen en estos momentos en la DEP: SACORE; SACDEP y SAIDEP. El SIDEP está compuesto por 7 subsistemas dentro de los cuales se encuentra Registro Legal. Uno de los módulos del subsistema Registro Legal es Aseguramiento Logístico el cual abarca toda la gestión de la información que se maneja en el SACDEP y debe incluir nuevos requisitos identificados. Este módulo deberá gestionar la información referente al transporte, la seguridad, las redes técnicas de las estructuras penitenciarias, los medios técnicos y el control de las capacidades. En la DEP las estructuras penitenciarias son todos los tipos de centros penitenciarios y las jefaturas a la que estos centros se subordinan.

Actualmente el SACDEP se encarga de gestionar las capacidades y medios técnicos en los centros penitenciarios con lo cual se tiene un mejor control de la infraestructura de las prisiones, aunque este no maneja todos los datos necesarios para los centros penitenciarios, pues de manera general la información con la que trabaja ha variado desde su puesta en funcionamiento lo cual trae consigo que los datos sean insuficientes o no estén acorde con las necesidades actuales del órgano de dirección de la DEP.

Algunas de las deficiencias que presenta el SACDEP son:

- No permite recoger los datos de algunos locales, tales como las celdas, cubículos, pabellones entre otros, dificultando el control de los medios que se encuentran en algunos locales. Esto provoca pérdidas de los medios y desconocimiento sobre lo sucedido.
- No lleva un control de las capacidades afectadas y fuera de uso, provocando que la ubicación se realice incorrectamente.
- No permite almacenar toda la información de la técnica canina, postas de acceso, así como otras cuestiones de seguridad, trayendo consigo una posible brecha de seguridad. Al no tener toda la información necesaria de las cuestiones de seguridad, los oficiales de los centros penitenciarios pueden desconocer la existencia de un problema.
- No se explota al máximo porque su manejo se hace muy difícil para los funcionarios pues requiere de especialistas en temas técnicos, con conocimientos de temas eléctricos, hidráulicos, informáticos.
- Almacena datos repetidos en varios casos.

Los centros penitenciarios utilizan el SACORE como sistema informático fundamental para controlar la legalidad de los internos, este para poder ubicar a un interno necesita la estructura y la disponibilidad de capacidades y estos datos son manejados por el SACDEP, pero esta interacción entre ambos sistemas no existe, por lo que dificulta el trabajo eficiente del órgano de dirección de la DEP. De igual forma, SACORE gestiona los traslados y los conduces los cuales requieren de una asignación de transporte especializado, dicho transporte se gestiona en el SACDEP y es independiente a la ejecución de los movimientos antes mencionados.

Debido a lo anterior se plantea como **Problema a resolver**: ¿Cómo controlar las instalaciones, capacidades y medios técnicos de las estructuras del Sistema Penitenciario Cubano?

El **objeto de estudio** de la investigación se centra en la gestión de procesos de Aseguramiento Logístico de los Sistemas Penitenciarios, enmarcándose en el **campo de acción** la gestión de procesos de Aseguramiento Logístico del Sistema Penitenciario Cubano.

Para darle solución al problema planteado se trazó como **objetivo general:** Desarrollar el módulo Aseguramiento Logístico del SIDEP.

Objetivos específicos:

- Diseñar el marco teórico de la investigación.
- Diseñar el módulo Aseguramiento Logístico del SIDEP.
- Implementar el módulo Aseguramiento Logístico del SIDEP.
- Probar el módulo de Aseguramiento Logístico del SIDEP.

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

1. Análisis de las soluciones informáticas nacionales e internacionales que implementen procesos con el mismo perfil que Aseguramiento Logístico.
2. Descripción de las herramientas y tecnologías para dar solución al problema.
3. Elaboración de los diagramas de clases del diseño del módulo Aseguramiento Logístico.
4. Elaboración de los diagramas de interacción del módulo Aseguramiento Logístico.
5. Diseño de la base de dato del módulo Aseguramiento Logístico.
6. Realización el modelo de componentes para el módulo Aseguramiento Logístico.
7. Implementación del módulo Aseguramiento Logístico.
8. Elaboración del diagrama de despliegue del módulo Aseguramiento Logístico.
9. Diseño de los casos de prueba para el módulo Aseguramiento Logístico.
10. Realización de las pruebas de caja negra para el módulo Aseguramiento Logístico.

El presente documento consta de los siguientes capítulos**:**

Capítulo 1: Fundamentación Teórica

En este capítulo se analizan otras soluciones informáticas que están relacionadas con el campo de acción, o sea, el estudio del arte. Se describe las principales herramientas y tecnologías definidas por la dirección del proyecto para desarrollar el sistema.

Capítulo 2: Diseño

En este capítulo se aborda la estructura arquitectónica del sistema, los patrones de diseño que se utilizarán y las funcionalidades que son objeto de informatización del módulo Aseguramiento Logístico. Además, se presentarán los diagramas de clases de diseño que responden a dichas funcionalidades, donde se especificarán las responsabilidades de las clases y sus relaciones. Conjuntamente se expondrán los diagramas de secuencia y el diseño de la base de datos para la persistencia de la información.

Capítulo 3: Implementación y Prueba

En este capítulo se mostrarán los diagramas de componentes y por último se diseñarán los casos de prueba y se expondrán los resultados de la aplicación de dichas pruebas.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un análisis referente a otras soluciones informáticas que están relacionadas con el objeto de estudio identificando los elementos que aportan a la solución propuesta y se describirán las principales herramientas y tecnologías definidas por la dirección del proyecto para desarrollar el sistema.

1.2 Marco conceptual

El marco conceptual de la investigación establece los términos fundamentales para un mejor entendimiento de los conceptos del negocio. (1) En esta sección se describe cómo se estructura un centro penitenciario y los conceptos asociados al Aseguramiento Logístico en la DEP.

Los centros penitenciarios están compuestos por diferentes áreas: (2)

- **Área de Evaluación y Diagnóstico**: Parte de la capacidad instalada y en explotación que se destina para la recepción, estudio, caracterización y evaluación, por las diferentes especialidades del penal, de los ingresos o traslados de reclusos.
- **Área de reclusión**: Conjunto de instalaciones divididas en celdas individuales y colectivas, áreas interiores, que incluye comedores, aulas, pasillos, patios, puestos médicos, pabellones conyugales, salones de visita, lugares de producción y servicios.
- **Áreas de servicios**: Conjunto de instalaciones cuya finalidad es prestar un servicio a la población penal, con independencia de su ubicación, y que se integra en el funcionamiento del centro penitenciario (salón de visitas, habitaciones conyugales, camas asistenciales y servicios médicos estomatológicos, cocina comedor, servicios de tintorería, área docente, técnica, deportiva, recreativa o productiva, teatro, biblioteca, celdas disciplinarias y de medidas de seguridad, áreas de entrevistas y servicios religiosos).
- **Área del interior penal**: Es el espacio e instalaciones delimitados por el cordón de seguridad.

Las áreas están compuestas por varios locales en dependencia del tipo de área. A los locales se les asocian determinados medios que difieren por el tipo de local. Los locales del área de reclusión requieren

el control de las literas y el estado de disponibilidad de las camas, por ello se les hace un mayor seguimiento. Entre estos locales se encuentran: (2)

- **Celda:** Estructura interna de un destacamento concebido para albergar de 1 a 3 reclusos en dependencia del área y la disposición de las instalaciones sanitarias independientes.
- **Cubículo:** División estructural de un destacamento en cuya área se pueden albergar de 4 a 21 reclusos en dependencia de sus dimensiones, el tipo de cama, y la cantidad de instalaciones hidrosanitarias.
- **Pabellón:** División estructural de un destacamento cuya área, cantidad de camas e índice de simultaneidad de aparatos permite albergar colectivos de más de 21 reclusos, por lo general constituyendo un solo destacamento.

A continuación se hace mención a los aspectos a tener en cuenta para asignar las literas en los locales del área de reclusión.

Para la instalación de las litaras en los locales del área de reclusión se requiere un espacio mínimo necesario: (2)

- En cuanto al puntal: 2,40 en literas de a dos y 3,20 en literas de a tres.
- Pasillo de circulación: 0.70cm en literas de a dos y 0.80cm en literas de a tres.
- Largo y ancho: 1,95 de largo por 0.75 de ancho, lo que equivale a 3,84 m^2 en literas de a dos y 4,26 m^2 en literas de a tres.

Para la ubicación de los internos en los locales del área de reclusión se definen los tipos de capacidades en los centros penitenciarios: (2)

- **Capacidad afectada parcial:** capacidad afectada parcial como aquella que no reúne algunos de los requisitos higiénicos sanitarios, constructivos o de diseño; pero que mantiene determinados niveles de funcionalidad.
- **Capacidad inhabilitada o fuera de uso:** cuando por la gravedad de las afectaciones no puede ser utilizada.
- **Capacidad instalada:** El número total de camas existentes físicamente en un local de reclusión.

- **Capacidad penitenciaria:** Es la disponibilidad de espacio de habitabilidad y servicios que se le brinda a las diferentes categorías de reclusos en correspondencia con las exigencias del régimen penitenciario y las características estructurales, funcionales y de diseño de la instalación.
- **Capacidad reglamentaria:** La cantidad de camas que de acuerdo al área modular reglamentaria deben existir.

A partir del estudio de los conceptos relacionados al Aseguramiento Logístico en la DEP se analizan algunas soluciones informáticas que de alguna forma manejan estos conceptos.

1.3 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.

1.3.1 Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios

Este sistema controla la información referente al control de capacidades de los establecimientos penitenciarios, permite registrar información de todos los locales del centro, su estructura, dar soporte a una mejor ubicación del interno y además controla los datos referentes a los medios del centro como de las instalaciones eléctricas, hidráulicas, sanitarias y de seguridad. Entre sus funcionalidades se encuentran:

- Mostrar la Capacidad de dormitorios en el área de reclusión.
- Gestionar la Información general de la edificación: clasificación, construcción, datos generales sobre el estado del centro y sobre su estructura interna.
- Gestionar los Órganos que tienen vínculo con el establecimiento.
- Registrar información de las redes técnicas, como el consumo eléctrico, plantas eléctricas, abasto de agua, depósitos de agua, garitas de vigilancia, alcantarillado, entre otros.

Este sistema no resuelve el problema que existe en la organización ya que este trabaja con información que ha cambiado y no existe la integración entre el SACORE y el SACDEP. Aunque las funcionalidades antes mencionadas serán tomadas como punto de referencia para el desarrollo del módulo. El SACDEP es una aplicación de escritorio, este sistema se podría modificar y corregirles las deficiencias actuales y

agregarles las nuevas, pero al ser una aplicación de escritorio se tendría que distribuir e instalar en todos los centro penitenciarios y lo que necesita la DEP es un sistema que se le pueda dar soporte fácilmente y que su despliegue se realice más rápido, por eso la DEP opta por una nueva implementación y que la solución sea una aplicación web.

1.3.2 Sistema de Gestión Penitenciaria (SIGEP)

El SIGEP es un sistema desarrollado por la Universidad de las Ciencias Informáticas, tributa a la Informatización y modernización de las Instituciones Penales de la República Bolivariana de Venezuela a raíz de los acuerdos de la Alianza Bolivariana de las Américas. Este proyecto posee un módulo acerca del control de las capacidades en los establecimientos penitenciarios, el mismo tiene una serie de funcionalidades como son: (3)

- Gestionar locales del área de reclusión.
- Gestionar capacidades de reclusión.
- Gestionar afectaciones de locales y capacidades de reclusión.
- Consultar estado de las capacidades.

Dicho sistema posee algunas de las funcionalidades que necesita el módulo Aseguramiento Logístico del subsistema Registro Legal del sistema SIDEP, pero el proceso de realizar las afectaciones es diferente y no abarca el otro proceso de aseguramiento logístico, además no posee todas las funcionalidades que posee el módulo Aseguramiento Logístico.

1.3.3 Otros sistemas

Se estudiaron otros sistemas del ámbito internacional como son: Spillman Jail, Offendertrak de Motorola, Jail Administration and Management System (JAMS) de Cisco, Jail Management System (JMS) de Sundance Systems Inc, Elite Jails de Syscom, JailTracker, Sentinel de Emerald System Inc, entre otros. Estos sistemas estudiados son privativos y no tienen implementado procesos de Aseguramiento Logístico.

1.4 Herramientas y Tecnologías

El desarrollo de software no sería posible sin las herramientas y tecnologías que le dan soporte, pues de no existir el trabajo sería engorroso, tedioso y muy lento para la obtención de los resultados. A continuación se dará una breve descripción de las herramientas y la metodología que se definieron por la dirección del proyecto para el desarrollo del módulo Aseguramiento Logístico.

1.4.1 Metodología

Para el desarrollo del sistema se utiliza como metodología Rational Unified Process (RUP), esta es una metodología pesada, que se centra en el control de los procesos estableciendo rigurosamente las actividades a desarrollar; este tipo de metodología es eficaz y necesaria en proyectos grandes, donde el ciclo de vida es largo y el personal que trabaja en el mismo esté sujeto a cambios, como es el SIDEP.

RUP se caracteriza por ser dirigido por los casos de usos, centrado en la arquitectura, iterativo e incremental. Este permite un trabajo organizado pues está divido por cuatro fases durante el desarrollo del sistema (Inicio, Elaboración, Construcción y Transición), en cada fase se define qué tareas realizar (Artefacto), por quién (Rol), cuándo (Flujo de Trabajo) y cómo realizarlas (Actividades). (4)

RUP define nueve flujos de trabajo, los cuales son:

- Modelamiento de negocio.
- Requerimientos.
- Análisis y diseño.
- Implementación.
- Prueba.
- Despliegue.
- Administración del proyecto.
- Administración de configuración y cambio.
- Soporte.

Ilustración 1: Representación del proceso ingenieril del software planteado por RUP

1.4.2 Lenguaje de modelado

RUP utiliza el Lenguaje Unificado de Modelado (UML por sus siglas en inglés Unified Modeling Language) para preparar todos los esquemas de un sistema de software. Este lenguaje posee diversos elementos gráficos que se combinan para realizar los diagramas. Como todo lenguaje este tiene definida reglas para poder combinar estos elementos. Es un lenguaje gráfico que cuenta con un grupo de diagramas, los cuales son utilizados para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. UML indica que es lo que supuestamente hará el sistema, pero no cómo lo hará. (5)

Se utilizó la herramienta de modelado Visual Parading for UML 6.4 la cual utiliza la versión 2 de UML. UML 2 modificó el lenguaje de manera tal que permitiera capturar más comportamientos (Behavior). De esta forma, se permitió la creación de herramientas que soporten la automatización y generación de código ejecutable, a partir de modelos UML.

Descripción de los diagramas que posee UML 2: (6)

- Diagrama de clases: Conjunto de clases, interfaces y colaboraciones.
- Diagrama de objetos: Conjunto de objetos y sus relaciones.
- Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.

- Diagrama de Estructura de Composición: Representa la estructura interna de un clasificador (tal como una clase, un componente o un caso de uso), incluyendo los puntos de interacción de clasificador con otras partes del sistema.
- Diagrama de Paquete: Un diagrama que presenta cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos, incluyendo importaciones y extensiones de paquetes.
- Diagrama de Revisión de la Interacción: Los Diagramas de Revisión de la Interacción enfocan la revisión del flujo de control, donde los nodos son Interacciones u Ocurrencias de Interacciones. Las Líneas de Vida los Mensajes no aparecen en este nivel de revisión.
- Diagrama de Secuencia: Un diagrama que representa una interacción, poniendo el foco en la secuencia de los mensajes que se intercambian, junto con sus correspondientes ocurrencias de eventos en las líneas de vida.
- Diagrama de Comunicación (Antes colaboración): Es un diagrama que enfoca la interacción entre líneas de vida, donde es central la arquitectura de la estructura interna y cómo ella se corresponde con el pasaje de mensajes. La secuencia de los mensajes se da a través de un esquema de numerado de la secuencia.
- Diagrama de Máquina de estados: Un diagrama que ilustra cómo un elemento, muchas veces una clase, se puede mover entre estados que clasifican su comportamiento, de acuerdo con disparadores de transiciones, guardias de restricciones y otros aspectos de los diagramas de Máquinas de Estados, que representan y explican el movimiento y el comportamiento.
- Diagrama de actividad: Representa los procesos de negocios de alto nivel, incluidos el flujo de datos. También puede utilizarse para modelar lógica compleja y/o paralela dentro de un sistema.
- Diagrama de Tiempo: El propósito primario del diagrama de tiempos es mostrar los cambios en el estado o la condición de una línea de vida a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados. Los eventos que se reciben se anotan, a medida que muestran cuándo se desea mostrar el evento que causa el cambio en la condición o en el estado.
- Diagrama de componentes: Organización y las dependencias entre un conjunto de componentes.
- Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

1.4.3 Herramientas CASE

CASE corresponde al vocablo inglés: Computer Aided Software Engineering; (Ingeniería de Software Asistida por Computación). El concepto de CASE es muy amplio; una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerarla como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. La atención se centra en el uso de las herramientas, para el desarrollo de proyectos informáticos que tengan como objetivo la automatización de procedimientos administrativos; se puede decir que: las herramientas CASE representan una forma que permite modelar los procesos de negocios de las empresas y desarrollar sistemas de información. En un sentido más amplio, las herramientas CASE se pueden ver como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (7)

Visual Paradigm for UML 6.4

Para el modelado de los artefactos y diagramas generados a lo largo del ciclo de vida del proyecto se decidió emplear Visual Paradigm for UML en su versión 6.4, pues su uso está muy estandarizado a nivel mundial y constituye una herramienta multiplataforma.

Visual Paradigm para UML es una herramienta profesional fácil de utilizar, que soporta el ciclo de vida completo del desarrollo de software en la UCI: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a la rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, permite la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automáticamente en varios formatos, entre ellos PDF (sigla del inglés portable document format, formato de documento portátil), permite el control de versiones. Además soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (8)

Embarcadero ER/Estudio 8.0

El ER/Estudio es una aplicación de modelado de datos para el diseño de bases de datos lógicos y físicos y su construcción. Posee un entorno de diseño que responde a las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de gran tamaño y bases de datos complejas.

La aplicación facilita la capacidad de crear, entender y gestionar los diseños de base de datos de misión crítica dentro de la empresa, ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción de bases de datos automática, ingeniería inversa de bases de datos precisas. Por todo lo planteado el equipo de arquitectura escogió esta herramienta Case para el modelado de datos.

Esta permite hacer:

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

1.4.4 Plataforma de desarrollo

Java Platform Enterprise Edition (JEE) es un conjunto de tecnologías que reduce significativamente el coste y la complejidad de desarrollo, despliegue y gestión de ambiente, centradas en un servidor de aplicaciones y una aplicación de cualquier tamaño a desarrollar. Sobre la base de la Plataforma Java Standard Edition (Java SE), JEE añade las capacidades que proporcionan una plataforma completa para el desarrollo sin contratiempos, que es a la vez estable, segura y rápida. (9)

1.4.5 Lenguajes de desarrollo

Groovy 1.7.8

El equipo de arquitectura seleccionó a Groovy como lenguaje del lado del servidor, pues este es un lenguaje orientado a objetos para la Plataforma Java, como alternativa al lenguaje de programación Java. Es un lenguaje dinámico, similar a Python, Ruby, Perl. Además puede usarse como lenguaje de scripting dentro de la Plataforma Java. (10)

Groovy utiliza una sintaxis con llaves para delimitar bloques, se compila dinámicamente hacia bytecodes para la Máquina Virtual Java, funcionando así con cualquier librería y código Java. El compilador Groovy genera bytecodes Java estándar que pueden usarse dentro de cualquier proyecto Java. Además, la mayoría del código Java es sintácticamente válido en Groovy.

El lenguaje Groovy es un superconjunto del lenguaje Java. En general se puede renombrar un archivo .java en .groovy y va a funcionar, aunque existen algunas incompatibilidades. Además, Groovy tiene algunas características que no existen en Java. Entre las características que distinguen a Groovy se incluyen el tipado estático y dinámico, cláusulas, sobrecarga de operadores, sintaxis nativa para la manipulación de listas y mapas, soporte nativo para expresiones regulares, iteración polimórfica, expresiones embebidas dentro de cadena de caracteres (string).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear aplicaciones web dinámicas. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Con JavaScript se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. JavaScript es una de las múltiples aplicaciones que han surgido para extender las capacidades del lenguaje HTML. Este permite trabajar con los contenidos dentro del documento lo cual facilita realizar validaciones y controlar los eventos que se ejecuten. (11) El equipo de Arquitectura escogió este lenguaje para darle dinamismo a la capa de presentación.

1.4.6 Tecnología

Ajax

AJAX (JavaScript asíncrono y XML), es una tecnología de desarrollo web para crear aplicaciones interactivas. Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor. Ajax no es una tecnología en sí mismo, en realidad se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes, las cuales son: XHTML y CSS para crear una presentación basada en estándares, DOM para la interacción y manipulación dinámica de la presentación, para el intercambio y la manipulación de

información utiliza XML, XSLT y JSON, XMLHttpRequest (Extensible Markup Language / Hypertext Transfer Protocol) para el intercambio asíncrono de información y JavaScript para unir todas las demás tecnologías. (12)

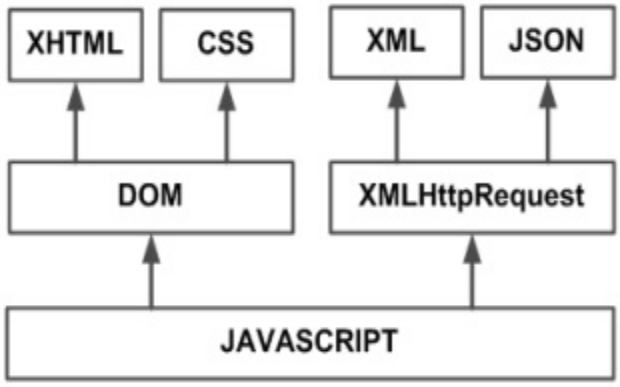

Ilustración 2: Tecnologías agrupadas bajo el concepto Ajax

1.4.7 Frameworks utilizados

Un framework es una estructura de soporte compuesta por componentes genéricos, personalizables, intercambiables y configurables para la organización de un proyecto de software. Para la solución de este sistema se definió por el equipo de desarrollo para la implementación del sistema los frameworks siguientes:

Grails 1.3.7

Grails es una nueva generación de Java framework para el desarrollo de aplicaciones web, este framework está construido sobre cinco fuertes pilares: **Groovy** para la creación de propiedades y métodos dinámicos en los objetos de la aplicación, **Spring** para los flujos de trabajo e inyección de dependencia, **Hibernate** para la persistencia de los datos, **SiteMesh** para la composición de las vistas, **Ant** para la gestión del proceso de desarrollo.

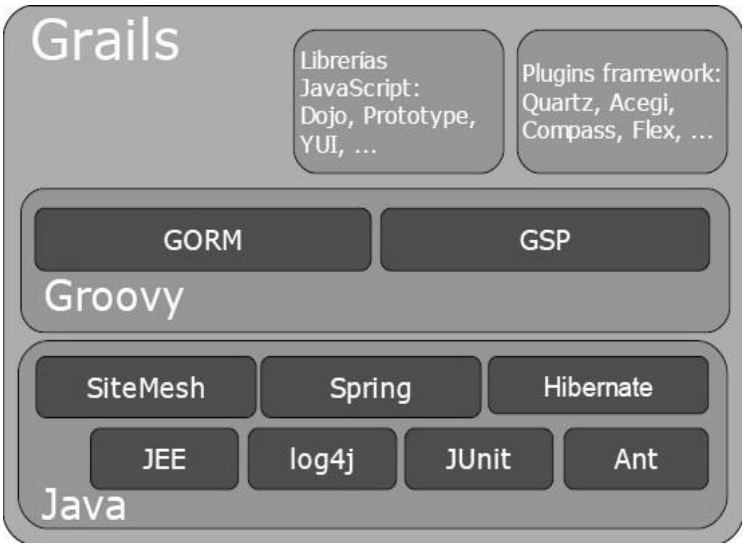

Ilustración 3: Estructura de Grails

Desde el punto de vista del diseño, Grails se basa en dos principios fundamentales, Convención sobre Configuración (Convention Over Configuration) y No te Repitas (Don't Repeat Yourself). En lugar de usar archivos de configuración XML se basa en las convenciones para hacer más fácil, sencillo y productivo el desarrollo, donde ya no es necesario configurar la aplicación mediante XML. La participación de Spring en Grails permite la inyección de dependencia, de forma que cada actor en la aplicación debe definirse una única vez, haciéndose visibles a todos los demás de forma automática. Debido a que aprovecha características dinámicas de Groovy, es ágil y dinámico y posee una mínima configuración, es capaz de acortar el ciclo de desarrollo ahorrando tiempo de trabajo. (10)

DOJO 1.5.0

Dojo es un framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Dojo posee un núcleo ligero que hace que las tareas comunes sean rápidas y sencillas. Animar elementos, manipular el DOM, y consultar con facilidad la sintaxis CSS, todo ello sin sacrificar el rendimiento. Dojo resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente

soportado en el cliente. Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores como Internet Explorer, Mozilla Firefox, Google Chrome, entre otros. (13)

1.4.8 Entorno de desarrollo integrado.

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés Integrated Development Environment) es un programa compuesto por una serie de herramientas que utilizan los programadores para codificar, algunas de las herramientas son: un editor de texto, un compilador, un intérprete, un depurador y un sistema de ayuda para la construcción de interfaz gráfica GUI. Las herramientas ante mencionada pueden estar hechas para soportar un único lenguaje de programación o varios. Los IDE pueden ser aplicaciones por si sola o pueden ser parte de aplicaciones existentes.

Spring Source Tools Suite 2.5.1

Spring Source Tools Suite está basado en Eclipse, pero además incorpora muchas herramientas y asistentes cuyo objetivo es el de facilitar y agilizar el desarrollo de aplicaciones las cuales obviamente utilicen tecnologías como Spring Framework y Spring Web Flow. Spring Source Tool Suite es una fantástica herramienta que ayuda a manejar de manera muy potente los proyectos Spring, con multitud de asistentes, editores, opciones y configuraciones. Spring Source Tool Suite presenta soporte para Groovy y Grails, pues este IDE permite ejecutar aplicaciones Grails en tcServer, lanzar comandos Grails, gestiona distintas versiones de Grails, permite gestionar 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 utilizar. (14)

1.4.9 Servidor de aplicación y gestor de base de datos

Apache Tomcat 6.0.25

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. Tomcat implementa las especificaciones de los servlets y de Java Server Page (JSP) de Sun Microsystems. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Apache Tomcat a menudo se presenta en combinación con el servidor web apache. (15)

Oracle 11g EE

Es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma. (16)

1.5 Conclusiones

El estudio teórico de las características que debía poseer un sistema de gestión penitenciaria y el análisis de algunos sistemas de gestión penitenciaria que tienen procesos de aseguramiento logísticos, confirmó la necesidad de implementar el módulo Aseguramiento Logístico del SIDEP. La metodología, herramientas, tecnologías y lenguajes que fueron utilizados son las adecuadas para el desarrollo del módulo Aseguramiento Logístico del SIDEP.

Capítulo 2: Diseño del Sistema

2.1 Introducción

El objetivo de este capítulo es obtener una visión más específica del sistema, donde se describirán las funcionalidades del módulo Aseguramiento Logístico. Además se explicarán temas relacionados a cuestiones propias del sistema, como son la arquitectura y los patrones de diseño. En este capítulo se realiza la propuesta de los diagramas de clases, los diagramas de interacción y el diseño de la base de datos que servirán de base para la implementación.

2.2 Procesos del negocio

La concepción de una estructura en el sistema penitenciario se inicia cuando la estructura es asignada a la DEP. El Administrador del centro penitenciario define las áreas de la estructura y los locales por cada área. Si los locales no pertenecen al área de reclusión se le asignan los medios que requieren para el trabajo de los funcionarios o para actividades específicas con los internos. En cambio, si son locales del área de reclusión además de asignarle los medios es necesario realizar una asignación de las literas. Para asignar las literas se requiere una verificación del espacio mínimo para los distintos tipos de literas y el pasillo de circulación. En dependencia del espacio se ubican las literas de una, dos o tres plazas.

Durante la vida útil de una estructura se realizan inspecciones para identificar afectaciones o verificar las condiciones de la misma. Este proceso se inicia cuando el Oficial de Aseguramiento Logístico inspecciona los locales de la estructura. Si detecta afectaciones, las registra en el Registro de control de las capacidades. Si la afectación es realizada a un local del área de reclusión es necesario determinar las literas y camas que afecta. En dependencia de las condiciones de la afectación la litera o cama puede quedar afectada o fuera de uso o en el estado en que se encontraba antes de la inspección (ocupada o disponible). El Oficial de Aseguramiento Logístico informa las afectaciones detectadas al nivel de mando superior. Una vez que se le da seguimiento a la afectación se modifican los estados de las literas o camas en caso de que hayan sido afectadas.

2.3 Requisitos funcionales

Un requisito funcional define el comportamiento interno del sistema: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán

llevados a la práctica. (17) A continuación se mencionan los requisitos funcionales del módulo Aseguramiento Logístico:

1. Registrar estructura.
2. Actualizar estructura.
3. Eliminar estructura.
4. Registrar órgano cooperante.
5. Actualizar órgano cooperante.
6. Eliminar órgano cooperante.
7. Registrar local.
8. Actualizar local.
9. Eliminar local.
10. Registrar afectación en un centro.
11. Eliminar afectación.
12. Actualizar afectación.
13. Registrar cordón de seguridad.
14. Actualizar cordón de seguridad.
15. Eliminar cordón de seguridad.
16. Registrar garita.
17. Actualizar garita.
18. Eliminar garita.
19. Registrar cerca.
20. Actualizar cerca.
21. Eliminar cerca.

22. Registrar área.
23. Eliminar area.
24. Registrar posta de acceso.
25. Actualizar posta de acceso.
26. Eliminar posta de acceso.
27. Actualizar datos de la técnica canina.
28. Registrar instructor de la técnica canina.
29. Eliminar instructor de la técnica canina.
30. Consultar unidades caninas.
31. Registrar unidad canina.
32. Eliminar técnico unidad canina.
33. Actualizar unidad canina.
34. Editar datos de la estructura.
35. Registrar abasto de agua.
36. Actualizar abasto de agua.
37. Eliminar abasto de agua.
38. Registrar acometida de agua.
39. Actualizar acometida de agua.
40. Eliminar acometida de agua.
41. Registrar depósito.
42. Actualizar depósito.
43. Eliminar depósito.
44. Registrar redes técnicas sanitarias.

45. Eliminar redes técnicas sanitarias.
46. Actualizar redes técnicas sanitarias.
47. Registrar acometida eléctrica.
48. Actualizar acometida eléctrica.
49. Eliminar acometida eléctrica.
50. Registrar planta generadora.
51. Eliminar planta generadora.
52. Actualizar planta generado.
53. Registrar datos de la red eléctrica.
54. Actualizar datos de la red eléctrica.
55. Registrar consumo electric.
56. Actualizar consumo electric.
57. Registrar medio contra incendio.
58. Actualizar medio contra incendio.
59. Eliminar medio contra incendio.
60. Registrar transporte.
61. Actualizar transporte.
62. Eliminar transporte.
63. Registrar medio de enfrentamiento.
64. Actualizar medio de enfrentamiento.
65. Eliminar medio de enfrentamiento.
66. Registrar medio de seguridad y vigilancia.
67. Actualizar medio de seguridad y vigilancia.

68. Eliminar medio de seguridad y vigilancia.
69. Registrar medio computarizado.
70. Actualizar medio computarizado.
71. Eliminar medio computarizado.
72. Registrar radio.
73. Actualizar radio.
74. Eliminar radio.
75. Registrar frecuencia de trabajo.
76. Actualizar frecuencia de trabajo.
77. Eliminar frecuencia de trabajo.
78. Registrar comunicación portátil.
79. Actualizar comunicación portátil.
80. Eliminar comunicación portátil.
81. Registrar estado técnico.
82. Actualizar estado técnico.
83. Eliminar estado técnico.
84. Registrar pizarra telefónica.
85. Actualizar pizarra telefónica.
86. Eliminar pizarra telefónica.
87. Registrar servicio telefónico.
88. Actualizar servicio telefónico.
89. Eliminar servicio telefónico.
90. Registrar sistema de intercomunicadores.

91. Actualizar sistema de intercomunicadores.
92. Eliminar sistema de intercomunicadores.
93. Registrar soporte del sistema.
94. Actualizar soporte del sistema.
95. Eliminar soporte del sistema.

2.2 Resumen de los casos de uso del módulo Aseguramiento Logístico

Nombre Caso de Uso	Descripción
CRUD-R estructura	El caso de uso permite registrar, actualizar o eliminar los datos de una estructura.
Editar datos de estructura	El caso de uso permite editar los datos de una estructura, pues el mismo es el que permite introducir en el sistema los datos de un tipo de estructura (Jefatura, CTEM, Asentamientos, Centros Penitenciarios).
CRUD-R local	El caso de uso permite registrar, actualizar o eliminar los datos pertenecientes a un local de un área determinada.
CRUD-R órgano cooperante	El caso de uso permite registrar, actualizar o eliminar un órgano cooperante registrado para un centro penitenciario.
CRUD afectación	El sistema muestra la jerarquía de estructuras del centro penitenciario y permite registrar, consultar, actualizar o eliminar una afectación en el sistema. El sistema actualiza los datos: capacidad fuera de uso y capacidad afectada de la estructura según los datos introducidos de la afectación.
Gestionar área	El caso de uso permite crear o eliminar un área dentro de una estructura.
CRUD-R cordón de seguridad	El caso de uso permite crear, modificar o eliminar un cordón de

	seguridad.
CRUD-R cerca	El sistema muestra las cercas que posee el cordón de seguridad y permite crear, modificar o eliminar una cerca y sus composiciones.
CRUD-R garita	El caso de uso muestra las garitas que posee el cordón de seguridad y permite crear, modificar o eliminar una cerca y sus composiciones.
CRUD-R posta de acceso	El caso de uso permite registrar, actualizar o eliminar una posta de acceso de una estructura determinada.
Gestionar técnica canina	El caso de uso permite crear, modificar o eliminar una Unidad Canina y registrar o eliminar un instructor.
CRUD unidad canina	El caso de uso permite crear, actualizar, eliminar o consultar una Unidad Canina.
Gestionar medios computarizados	El sistema muestra un listado con los medios computarizados de una estructura y permite registrar, actualizar o eliminar un medio computarizado.
Gestionar red técnica sanitaria	El sistema muestra un listado con las redes técnicas sanitarias de una estructura y permite registrar, actualizar o eliminar una red técnica sanitaria.
Gestionar transporte	El sistema muestra un listado con los medios de transporte de una estructura y permite registrar, actualizar o eliminar un medio de transporte.
Gestionar medios de enfrentamiento	El sistema muestra un listado con los medios de enfrentamiento de una estructura y permite registrar, actualizar o eliminar un medio de enfrentamiento.
Gestionar medios contra incendios	El sistema muestra un listado con los medios contra incendio de una estructura y permite registrar, actualizar o eliminar un medio contra incendio.

Gestionar medios de seguridad y vigilancia.	El sistema muestra un listado con los medios de seguridad y vigilancia de una estructura y permite registrar, actualizar o eliminar un medio de seguridad y vigilancia.
Gestionar acometida eléctrica	El sistema muestra un listado con las acometidas eléctricas de una estructura y permite registrar, actualizar o eliminar una acometida eléctrica.
Gestionar planta generadora	El sistema muestra un listado con las plantas generadoras de una estructura y permite registrar, actualizar o eliminar una planta generadora.
Actualizar redes eléctricas	El sistema permite actualizar una red eléctrica.
Gestionar consumo eléctrico	El sistema permite registrar el consumo eléctrico, actualizar el consumo eléctrico del año en que se encuentra en ese momento o consultar el consumo eléctrico.

2.3 Marco de trabajo de desarrollo web

Grails es un marco de trabajo que se utiliza para el desarrollo de aplicaciones web sobre la plataforma Java Enterprise Edition. Este marco de trabajo utiliza el patrón de arquitectura Spring MVC (Modelo-Vista-Controlador), este patrón establece que los componentes de un sistema de software debe organizarse en tres capas distintas según su misión; la vista o capa de presentación, el modelo o capa de datos y la capa de control. Grails incorpora otra capa para el manejo de la lógica de negocio que es la de servicio, lo que evidencia una arquitectura n capas. (10)

2.4 Arquitectura del Sistema

La arquitectura de software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. La arquitectura de software es muy importante sobre todo en el desarrollo de software grande y complejo. Esto posibilita entre otras cosas que los

desarrolladores progresen hacia una visión común del sistema. Además contribuye a comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema. (18)

La arquitectura que posee Grails está compuesta por 4 capas, que están dispuesta de la siguiente forma: la vista, el modelo, el controlador y los servicios.

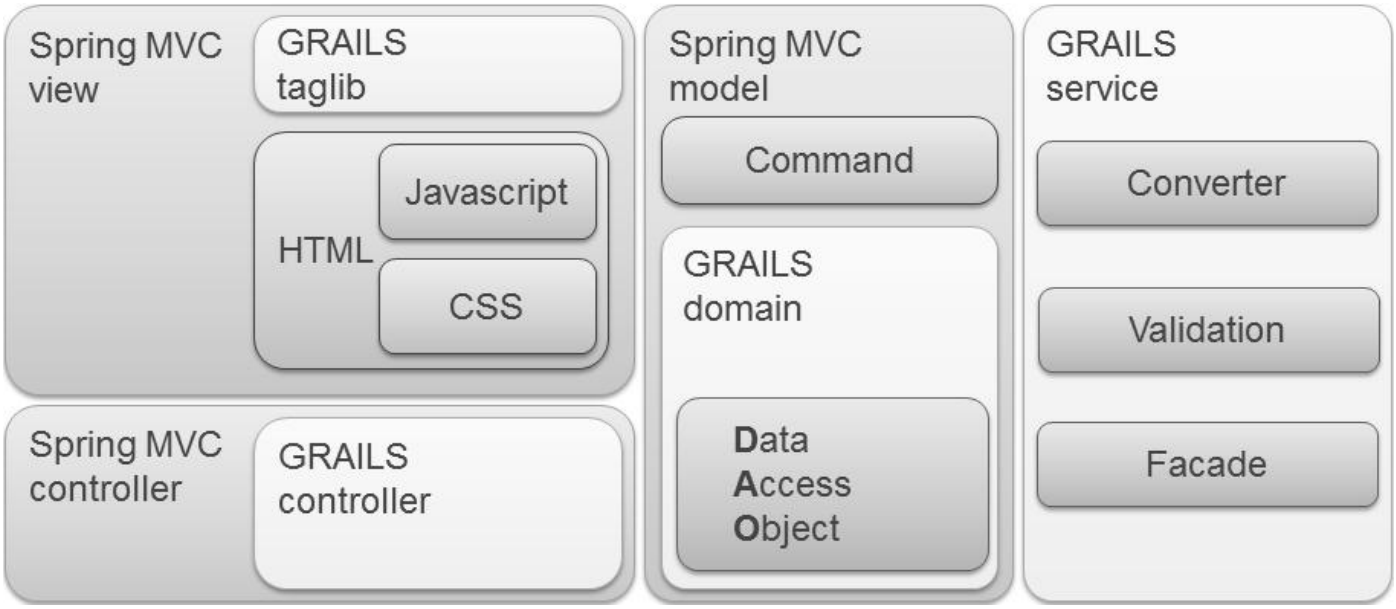

Ilustración 4: Arquitectura Grails

En la ilustración 4 se muestra la arquitectura de Grails, se puede observar cómo interactúan las cuatro capas de la aplicación; a continuación se explicará cada capa:

- La capa vista está compuesta 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), para esta capa Grails tiene el framework SiteMesh para el desarrollo de páginas web (creado por OpenSymphony) que implementa el patrón de diseño decorator para renderizar HTML con componentes aplicables a las páginas webs como cabeceras o funciones de navegación, pero se utilizó el framework Dojo para la creación de vistas, utilizando la tecnología Ajax.
- La capa de control o controlador es la encargada de llevar la comunicación entre las demás capas, esta responde a eventos, mayoritariamente acciones del cliente e invoca cambios en el modelo de

datos o en la vista. Grails utiliza el framework Spring MVC e Inversión de Control para la inyección de dependencia; la cual nos permite en un controlador inyectar un servicio o sea permite la gestión de la instancia de un objeto (la creación y destrucción de los objetos), la gestión transaccional.

- El modelo o capa de datos es donde se crea las clases de dominio y los command. Grails para la abstracción de datos utiliza Grails Object Relational Mapping (GORM) e Hibernate, para mapear objetos a base de dato relacional y representar las relaciones entre los objetos o sea entre las tablas de la base de datos. GORM es un gestor de persistencia escrito en groovy para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos dinámicos (se crean en tiempo de ejecución para cada clase del dominio) que facilita el acceso a datos.
- En la capa de servicio se lleva la lógica de negocio y ella es la encargada de llevar los cambios en el modelo.

2.5 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable. En términos generales, un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto, donde:

- Contexto son las situaciones recurrentes a las que es posible aplicar el patrón.
- Problema es el conjunto de metas y restricciones que se dan en ese contexto.
- Solución es el diseño a aplicar para conseguir las metas dentro de las restricciones.

Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. Han revolucionado el diseño orientado a objetos y todo buen arquitecto de software debería conocerlos. (19)

2.5.1 Patrones de diseño GRASP

Los patrones GRASP (Patrones Generales de Asignación de Responsabilidades) son patrones de diseño que se usan para asignar responsabilidades a una clase. A continuación mostramos los patrones utilizados:

Experto: Asigna una responsabilidad al más competente en información, donde la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de

responsabilidades que suele utilizarse en el diseño Orientado a Objetos. Es el patrón que más se usa para asignar responsabilidades.

Bajo Acoplamiento: Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. El Bajo Acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. (UML y Patrones)

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

Controlador: Asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema. Dicho controlador utiliza el sistema de enrutamiento para asociar el nombre de una acción con la URL solicitada por el usuario.

2.5.2 Patrones de diseño GOF

Los patrones GOF (Gand of Four, Banda de los Cuatro) son patrones de diseño que definen una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

Estos patrones se dividen en tres categorías: los creacionales, los estructurales y los de comportamiento. Seguidamente se expondrán los patrones empleados:

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

2.6 Diagrama de clases

2.6.1 Descripción de las clases más significativas.

Entidad	Descripción
Estructura	Representa los datos generales de las jefaturas, Campamento de trabajo y Estudio Municipal, Asentamientos y centros penitenciarios del Órgano de Prisiones.
Centro penitenciario	Almacena todos los datos de los centros penitenciarios del Órgano de

	Prisiones.
Área	Esta entidad almacena el nombre y el tipo de área que tiene una estructura.
Local	Esta entidad almacena los datos de los locales que tiene un área. Existen varios tipos de locales; oficina, comedor, puesto de mando, celda disciplinaria, etc.
Órgano cooperante	Esta entidad almacena los datos de un órgano que coopera con una estructura.
Afectación	Esta entidad recoge los datos de la afectación de un local.
Cordón de seguridad	Esta entidad recoge los del cordón de Seguridad; así como los datos de las cercas y las garitas que componen dicho cordón de seguridad.
Posta de acceso	Esta entidad recoge los datos de las postas de acceso a una estructura.
Técnica canina	Esta entidad recoge los datos de la técnica canina que tiene un centro penitenciario; así como el instructor y las unidades caninas que tiene la misma.
Medio computarizado	Esta entidad recoge los datos de los medios computarizados que tiene una estructura; como son las impresoras, computadoras, entre otros.
Red técnica canina	Esta entidad recoge los datos de la red técnica canina que tiene una estructura.

2.6.2 Diagramas de clase 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, estos son de estructura estática que muestran las clases del sistema y sus interrelaciones. Normalmente contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Dependencias.

Para consultar los diagramas de clase del diseño ver el anexo 1. A continuación se muestra el diagrama de clase del diseño de los caso de usos CRUD-R Local y CRUD-R Editar Estructura:

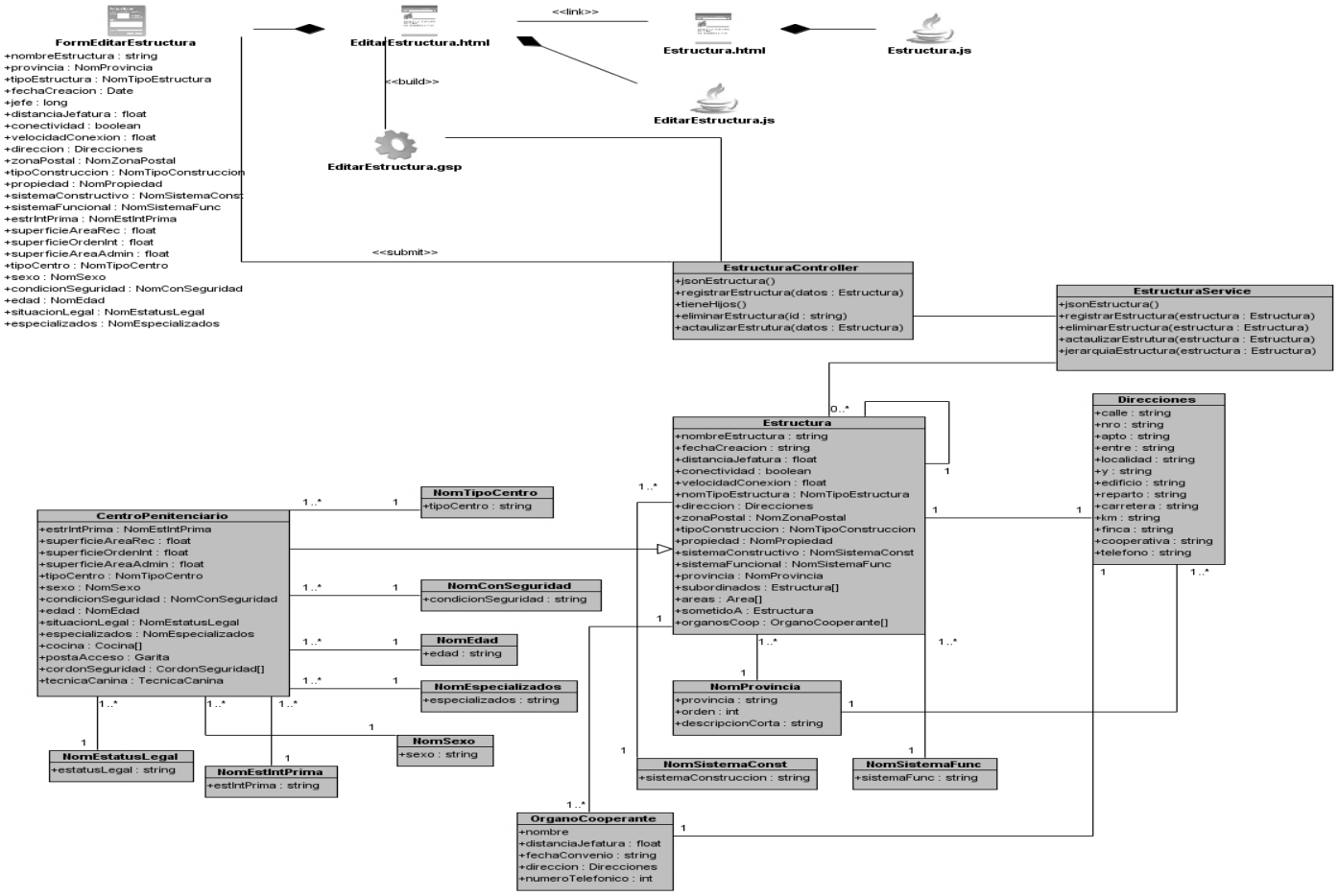

Ilustración 5: DCD CU Editar Estructura

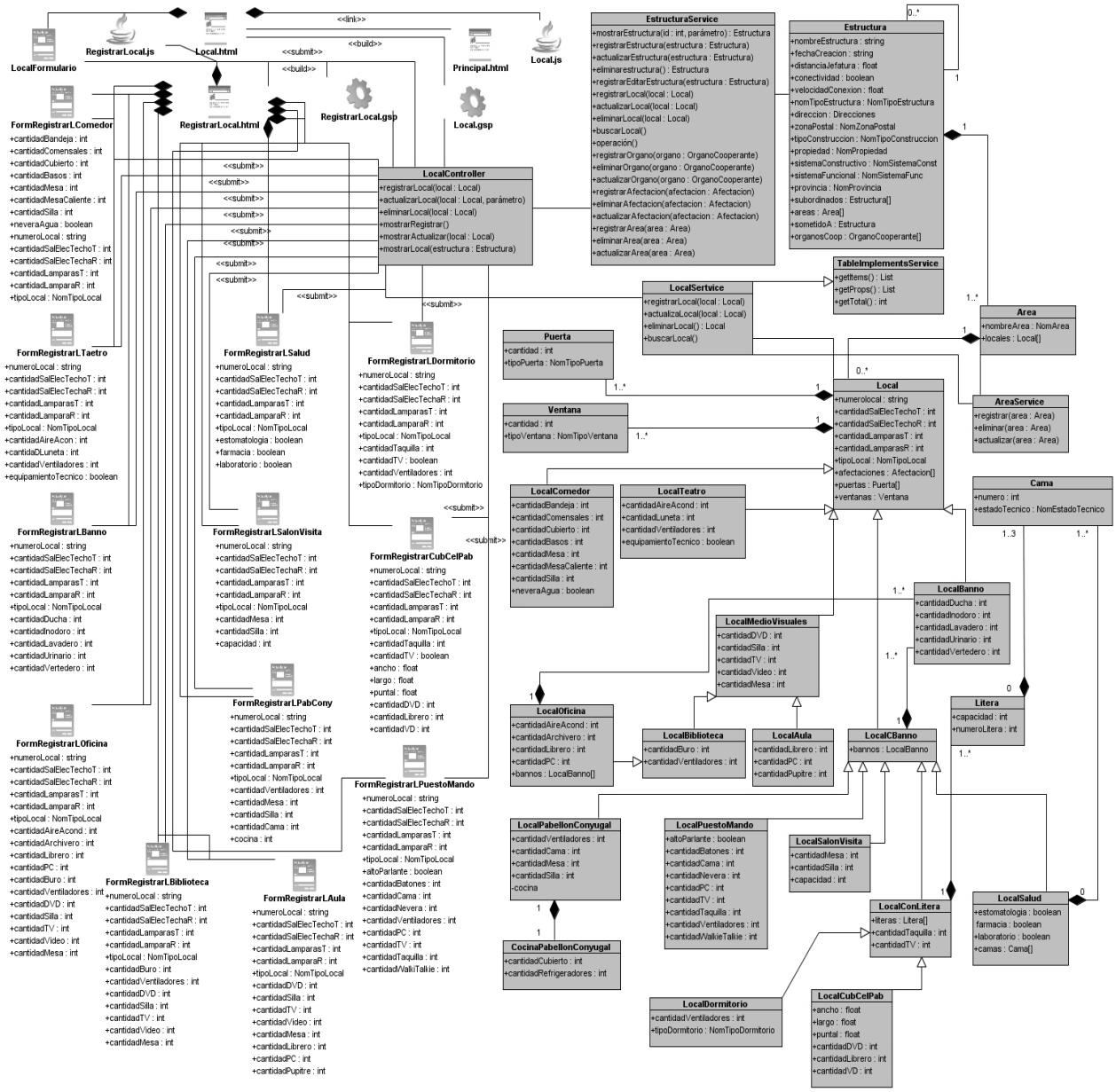

Ilustración 6: DCD CRUD-R Local

2.7 Diagramas de Interacción

Para consultar los diagramas de interacción ver el anexo 2. A continuación se muestra los diagramas de interacción de los caso de usos CRUD-R Local y CRUD-R Editar Estructura:

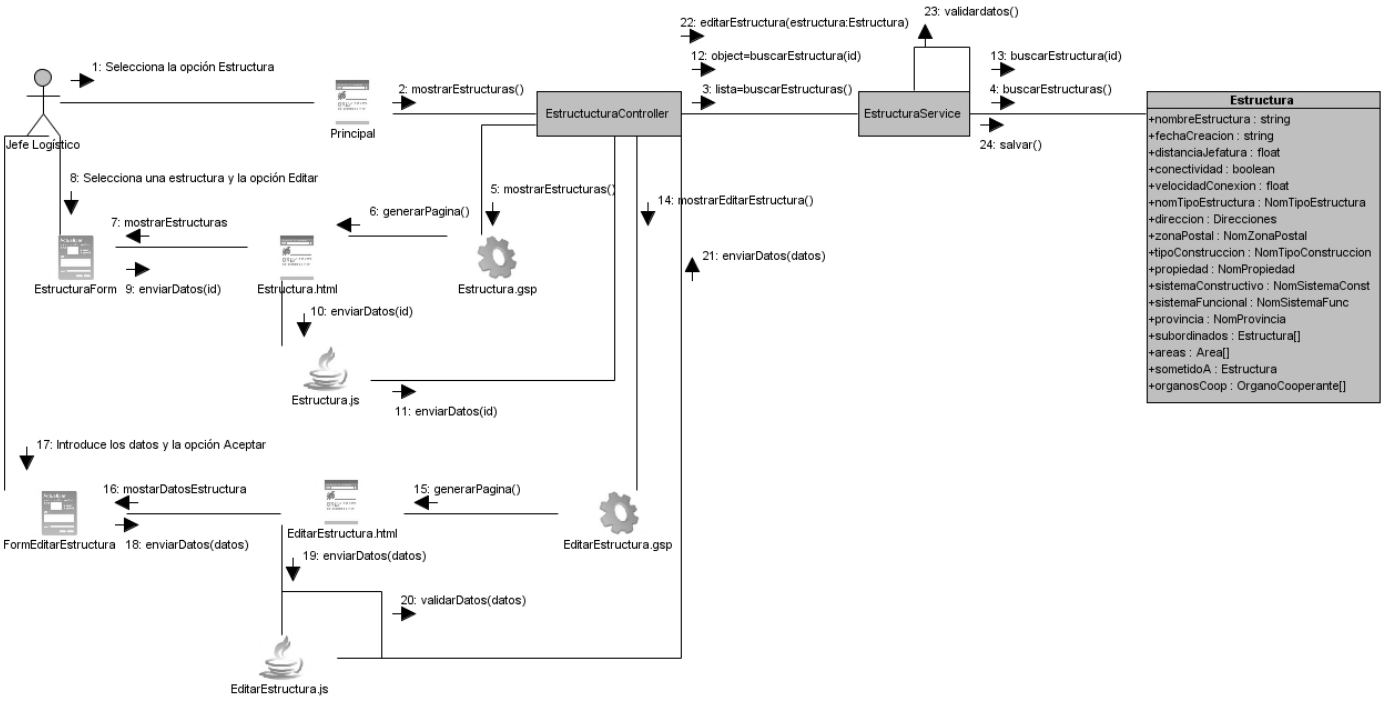

Ilustración 7: DC Editar Estructura

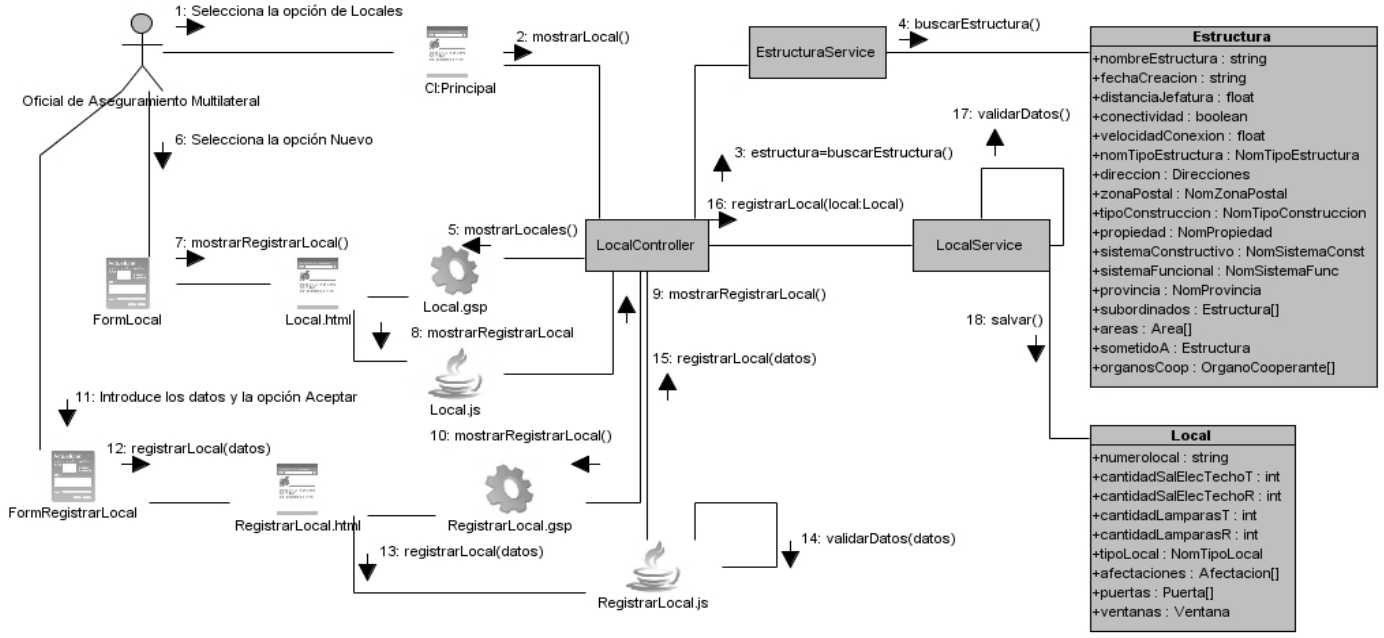

Ilustración 8: DC Registrar Local

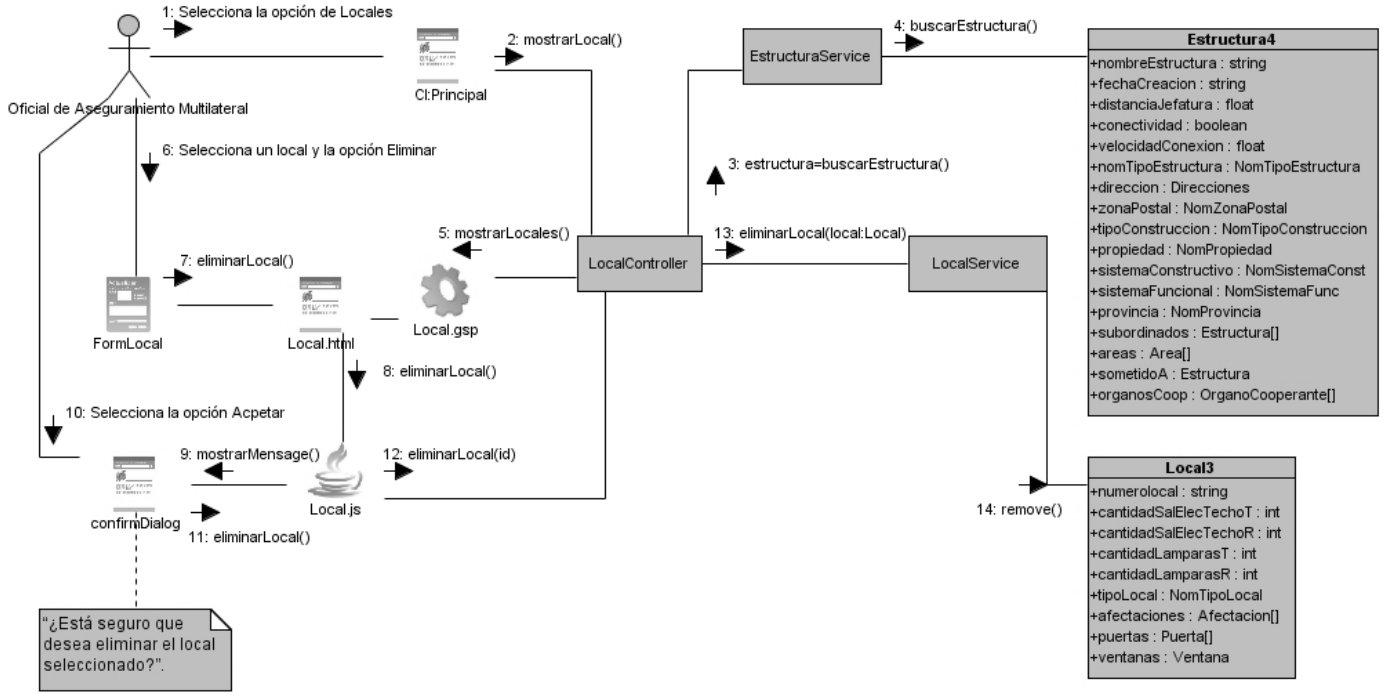

Ilustración 9: DC Eliminar Local

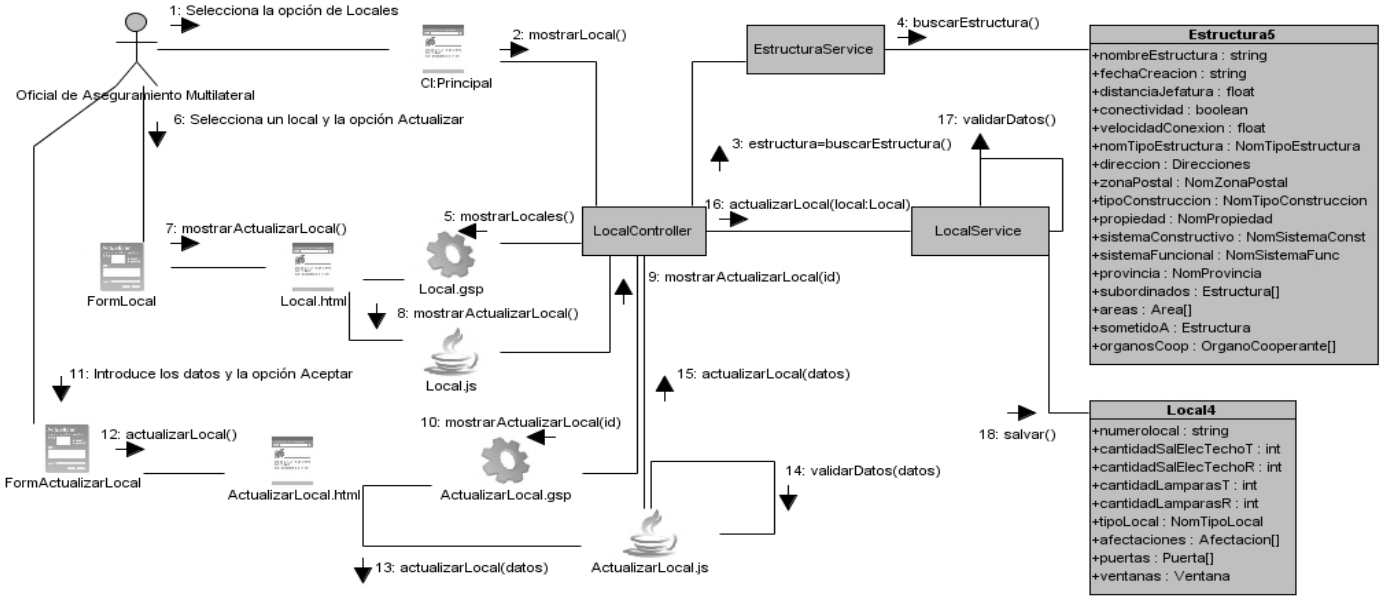

Ilustración 10: DC Actualizar Local

2.8 Diseño de la Base de Datos

Un diagrama o modelo entidad-relación (a veces denominado por sus siglas: ER "Entity Relationship" o "DER" Diagrama de Entidad Relación) es una herramienta 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.

El diseño de la base de datos se encuentra en la tercera forma normal, esta permite incrementar el rendimiento en las operaciones de inserción y actualización de datos.

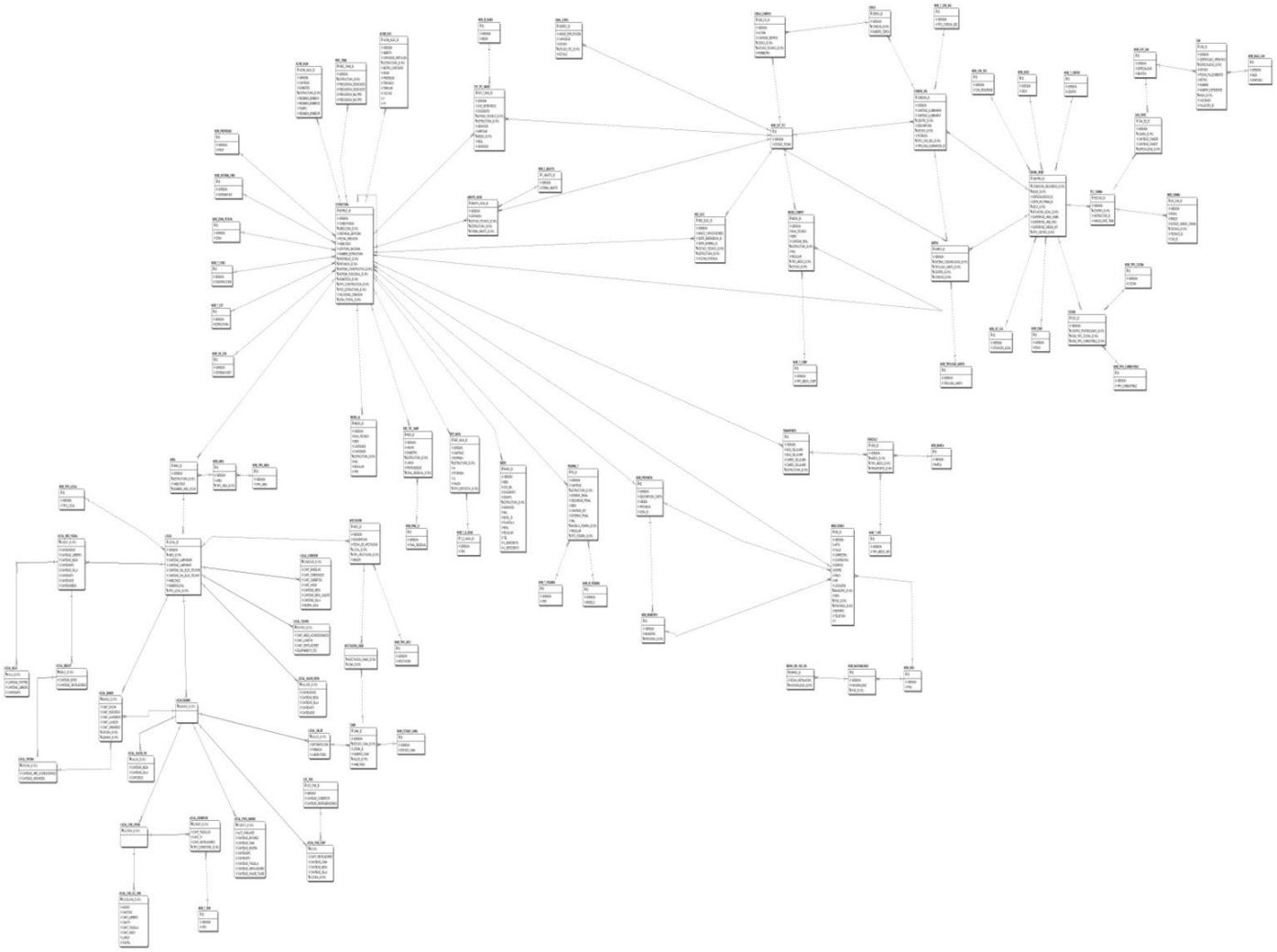

Ilustración 11: Modelo de dato de los locales

2.9 Conclusiones

Este capítulo explica el diseño del sistema, dando así cumplimiento a los objetivos trazados en el mismo, la definición de los patrones utilizados para realizar los diagramas de clases del diseño conjuntamente con los diagramas de interacción correspondientes a cada uno de los escenarios definidos y se especifican dónde se pone de manifiesto estos patrones en los diagramas de clases, también se define el diseño de la Base de Datos. El modelo de diseño realizado en el presente capítulo posibilita la implementación del módulo.

Capítulo 3: Implementación y Prueba

3.1 Introducción

En este último capítulo se mostrarán los diagramas de componente y de despliegue que serán realizados, como se realiza la internacionalización y por último se diseñarán los casos de prueba y se ejecutarán las pruebas de sistema para detectar los errores.

3.2 Diagrama de Componentes

Los Diagramas de Componentes ilustran las piezas del software que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (4)

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. De ahí que los componentes tengan relaciones de trazas con los elementos del modelo que implementan. El diagrama de componentes que representa las relaciones entre los diferentes componentes como realización del modelo de diseño, se dividirá por problemas de espacio en vistas. (4)

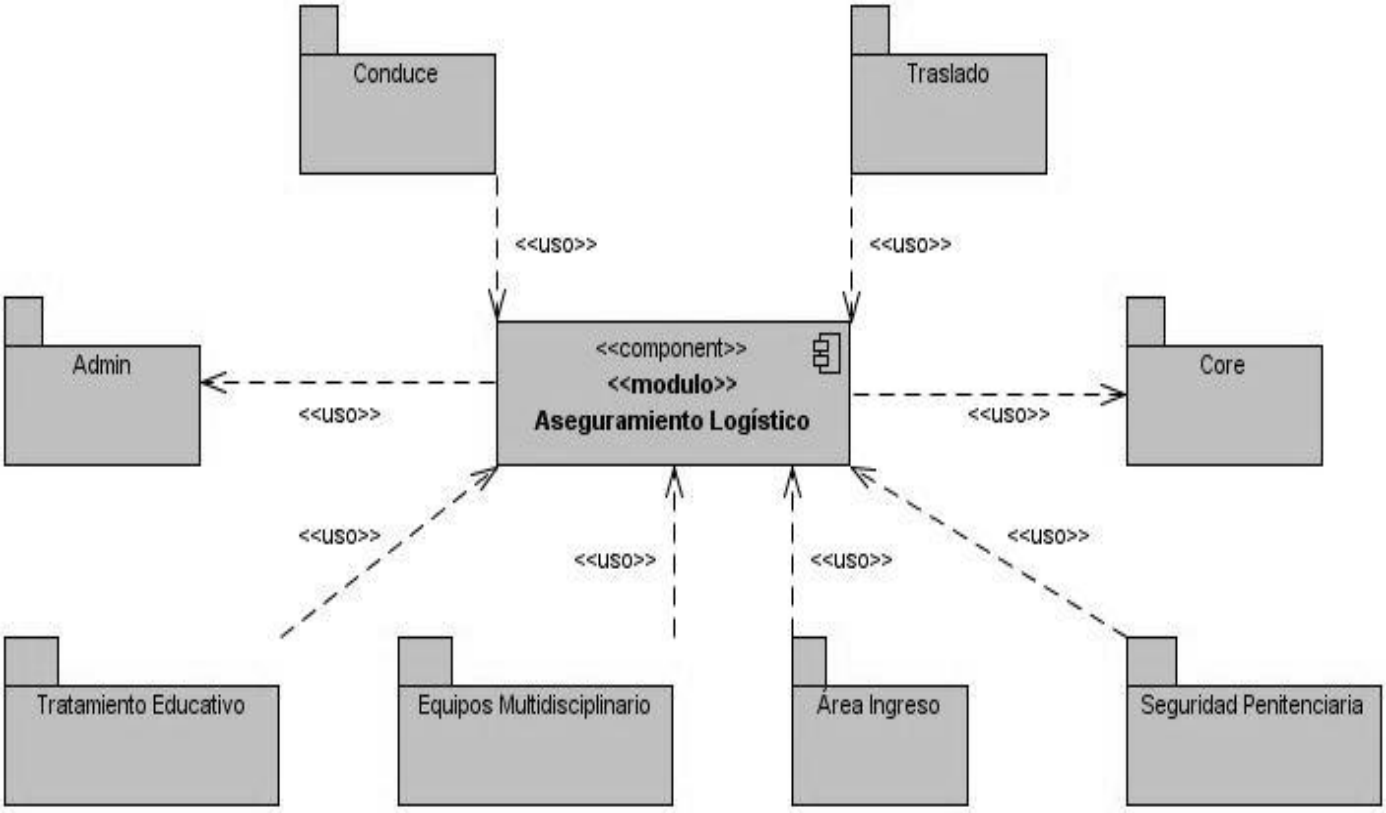

Ilustración 12: Diagrama de componente 1

En la ilustración 12 se muestra el diagrama de componente que representa la relación del módulo Aseguramiento Logístico con los restantes subsistemas y los módulos del subsistema Registro Legal. Como se observa el módulo Aseguramiento Logístico depende de los subsistemas Core y Admin y a su vez los restantes subsistemas dependen del módulo Aseguramiento Logístico.

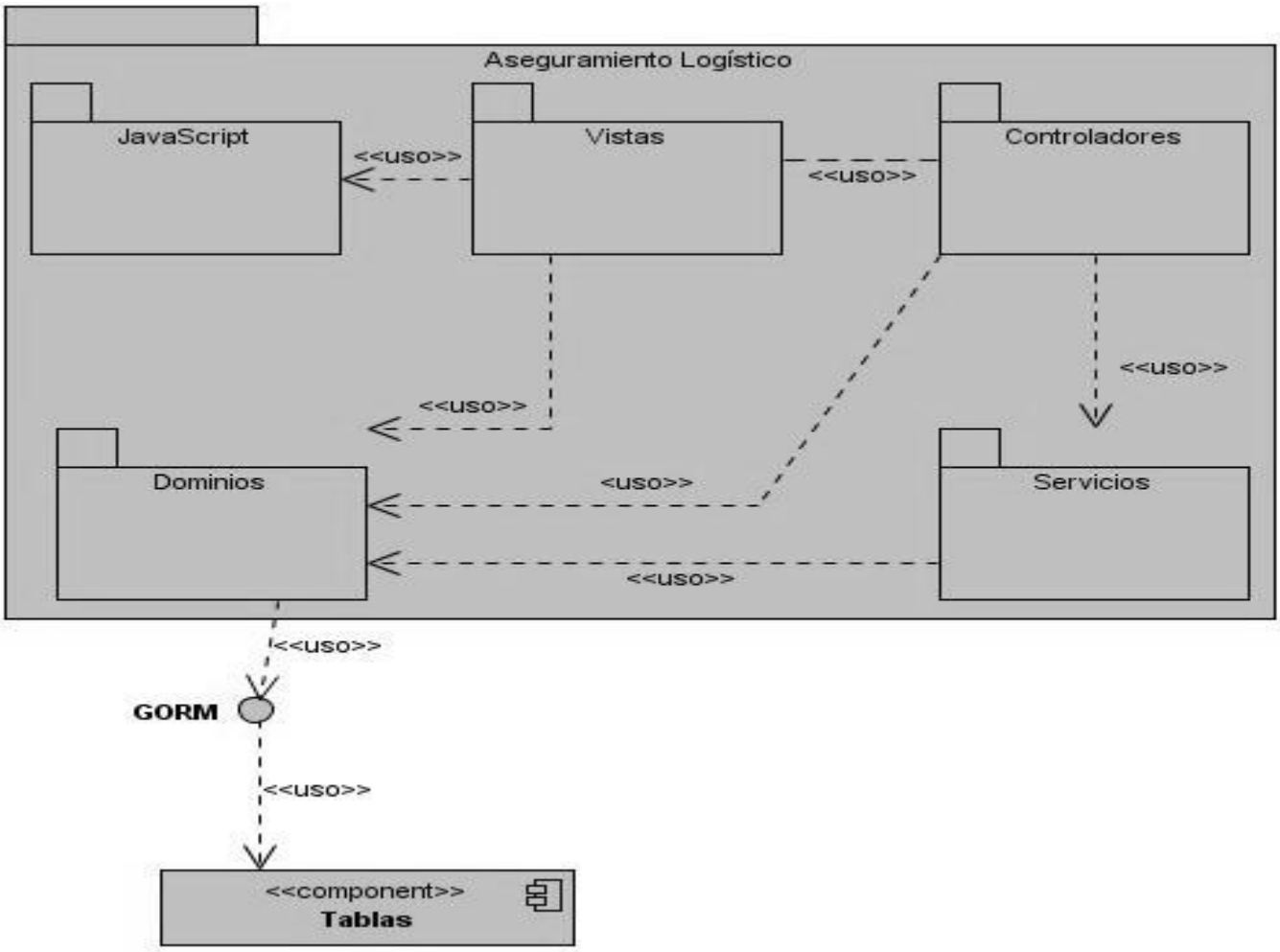

Ilustración 13: Diagrama de Componente 2

En la ilustración 13 se muestra el diagrama de componente que muestra la relación que existe entre los paquetes de componentes que conforman el módulo aseguramiento. En el paquete de componentes Vistas se encuentran la implementación de todas las groovy server page (gsp), las mismas usan los archivos JavaScript que se encuentran dentro del paquete de componentes JavaScript. El paquete de componente Controladores está conformado por los componentes controladores, los cuales se encargan de controlar el flujo de eventos de las vistas, están usan los componentes servicios que se encuentran en el paquete servicios y las clases de dominios que se encuentran en el paquete Dominios. En grails las clases de dominio funcionan como acceso a datos. El acceso a dato se logra mediante GORM que se comunica con las tablas de la base de datos para el manejo de los datos en la misma.

3.3 Internacionalización

La internacionalización permite a una aplicación que la interfaz de usuario soporte distintos idiomas. En grails la forma de trabajar la internacionalización es mediante unos ficheros que se encuentran en la carpeta grails-app/i18n. En esta carpeta se creará un fichero para cada idioma que soporte la aplicación, 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_AR.properties para los mensajes en español de Argentina.
- Messages_en.properties para los mensajes en inglés.
- Messages_es.properties para los mensajes en español.

Ilustración 14: Fichero para los mensajes en español

3.4 Seguridad

La seguridad en el módulo Aseguramiento Logístico, se logró realizando las validaciones y utilizando Spring Security el cual proporciona servicios integrales de seguridad para JEE, las principales áreas de

seguridad es 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 autenticación es cifrada o sea la contraseña es cifrada, se utiliza el Algoritmo Hash Seguro (SHA por sus siglas en inglés Secure Hash Algorithm) con una longitud de salida de 256 bits. La autorización es el proceso que permite decidir si un usuario tiene autorización a realizar una acción dentro de la aplicación. (20) En la autorización se utilizó el modelo Control de Acceso Basado en Roles (RBAC por sus siglas en inglés). Los derechos de acceso se agrupan por rol, y el acceso se restringe a los usuarios que tiene autorizado asumir dicho rol. Para lograr la autorización se 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.

AfectacionController.groovy

Afectacion | AfectacionController | Afectacion Views | Tests

```
import ...

@Security ("eliminarAfectacion,crud-d-Afectacion")
class AfectacionController {

    def afectacionService
    def literaCamaService
    def fileService
    def index = {
        render view: "/registrolegal/aseguramientoLogistico/afectacion"
    }

    def mostrarAreas = {
        list(afectacionService.areas())
    }

    def list(lista) {
        def items = []
        items << [id: -1, descripcion: message(code: 'default.select.text')]
        lista.each {
            items << [id: it.id, descripcion: it.toString()]
        }
        render([identifier: "id", items: items].encodeAsJSON())
    }

    def mostrarlocales = {
        if (params.id) {list(afectacionService.locales(Long.parseLong(params.id)))}
        else {list(afectacionService.localesAll())}
    }
```

Ilustración 15: Ejemplo de la notación de seguridad

La validación al módulo Aseguramiento Logístico se realizó en el cliente y el servidor. Para la validación de datos en el lado del cliente se utilizó JavaScript, se creó expresiones regulares, de forma tal que se asegure que solamente se escriban en los campos de entrada de datos los valores permitidos, por ejemplo si el campo es numérico solo se puede escribir números, al igual que si es alfanumérico o de letras, esto permite que no se introduzcan caracteres no deseados que posibilite la inyección SQL. La validación en el servidor se realizó con groovy usando las propias constraints de las clases de dominio,

con el objetivo de garantizar la integridad de los datos. Los constraints ofrecen una serie de restricciones que permite validar los atributos de la clase de dominio, ejemplo de algunas de las restricciones que posee:

- Max: está restricción se utiliza en los string para la longitud máxima.
- Min: está restricción se utiliza en los string para la longitud mínima.
- Nullable: para saber si el atributo permite valores nulos.
- Unique: para definir se el atributo es único.
- Matches: para el formato que tiene que cumplir el atributo o sea una expresión regular.

En la implementación del módulo Aseguramiento Logístico se utilizaron consultas parametrizadas, estas consultas no concatenan variables, esto evita la inyección SQL. Las consultas parametrizadas se realizaron mediante las facilidades que brinda GORM con la utilización de Hibernate Criteria y Lenguaje de Consulta de Hibernate (HQL por sus siglas en inglés Hibernate Query Language). HQL es completamente orientado a objetos y comprende nociones como herencia, polimorfismo y asociación. Hibernate Criteria es una API que facilita las consultas a la base de datos, permite tratar la composición de la consulta de una forma totalmente orientada a objeto.

3.5 Estándares de codificación

Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

La utilización de un estándar de codificación trae consigo ventajas, tales como:

- Asegurar la comprensión de cada una de las líneas de código.
- Facilitar el mantenimiento posterior a la creación del código.
- Facilitar la tarea de los programadores en el desarrollo del software.

Por las razones antes expuestas se decidió poner en práctica el uso de los estándares de codificación para la implementación del módulo Aseguramiento Logístico. A continuación se describen las notaciones que se utilizaron:

- Camel Case**:** La primera letra de los identificadores y nombres de las variables así como las funciones se escriben con minúscula y si contiene más de una palabra, las mismas tienen su primera letra mayúscula. Ejemplo: notaciónCamelCase.
- Pascal Case**:** La primera letra de los identificadores y nombres de las variables así como las funciones se escriben con mayúscula y si contiene más de una palabra, las mismas tienen su primera letra mayúscula. Ejemplo: NotaciónPascalCase.

A continuación se ejemplifica el uso de las notaciones definidas anteriormente:

- Declaración de variables: se utilizó la notación Camel Case, ejemplo listadoEstructura.
- Atributos: se utilizó la notación Camel Case, ejemplo velocidadConexion.
- Declaración de métodos: se utilizó la notación Camel Case, ejemplo actualizarEstructura.
- Declaración de clases del dominio: se utilizó la notación Pascal Case, ejemplo AcometidaElectrica.
- Declaración de controladores: se utilizó la notación Pascal Case y la terminación Controller, ejemplo EstructuraController.
- Declaración de servicios: se utilizó la notación Pascal Case y la terminación Service, ejemplo EstructuraService.
- Vistas: se utilizó la notación Camel Case, ejemplo gestionarSoporte.
- JavaScript: se utilizó la notación Pascal Case, ejemplo ConsumoElectrico.

Para lograr una mejor organización visual y entendimiento del código se hace referencia a las siguientes normas:

- Declarar las variables en líneas diferentes.
- Se dejó una línea antes y después de cada método.
- Se dejó un espacio en blanco entre operadores lógicos y aritméticos.
- No declarar más de una instrucción por línea.
- Comentar las clases, métodos y los atributos.

3.6 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema.

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. (21)

El diagrama de despliegue está compuesto por un servidor de aplicaciones web, un servidor de base de datos y las computadoras clientes.

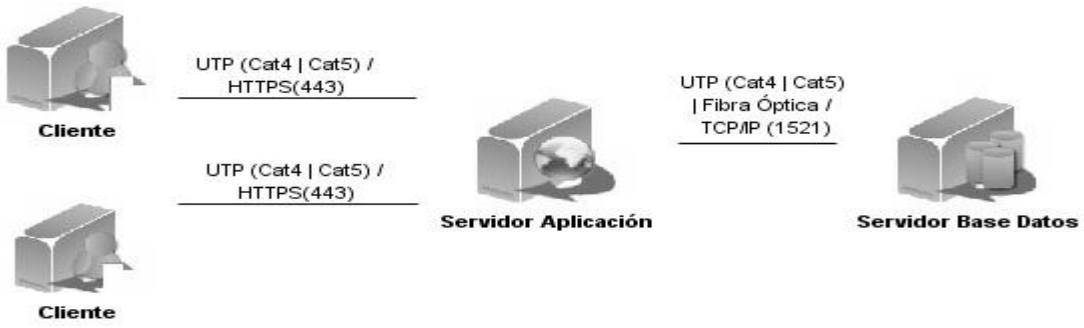

Ilustración 16: Diagrama de Despliegue

Para el despliegue del módulo es necesario:

Puestos de trabajo (PC usuario)

- ✓ RAM: 512 MB
- ✓ Navegador web Mozilla Firefox 3.6 o superior.

En estos nodos se encuentra los HTML que son construidos por las GSP.

Servidor de la aplicación

- ✓ Microprocesador: 4 núcleos, 3 GHz
- ✓ RAM: 4 GB
- ✓ Espacio necesario para la instalación: 250 MB
- ✓ Espacio libre: 250 GB
- ✓ JDK 1.6
- ✓ Apache Tomcat 6.0.25

En este nodo se encuentran las vistas o sea las gsp, los archivos JavaScript, las clases de dominio, los controladores y los servicios.

Servidor de Bases de datos

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

En este nodo se encuentra las tablas de la base de datos, donde se encuentran almacenados los datos de la aplicación.

3.7 Prueba

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de la entrega al cliente del software desarrollado; el éxito de las mismas puede mejorar la percepción de calidad del usuario final y lograr su satisfacción. (22) Existen niveles de prueba, uno de ellos es la prueba de desarrollador. Este tipo de prueba es diseñada e implementada por el equipo de desarrollo. Las pruebas de desarrollo pueden realizarse cruzando los programadores, analistas, u otro rol, de forma tal que no solo sea el desarrollador quien revise su propio trabajo, pues generalmente una persona ajena es la que más errores puede detectar. (23)

3.7.1 Prueba Caja Negra

La prueba de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. (18)

Para la realización de las pruebas se utilizó la prueba de caja negra, la cual se centra principalmente en los requisitos funcionales del software y no consideran la estructura interna del programa; las pruebas de

caja negra son hechas sin el conocimiento interno del producto. Se llevan a cabo sobre la interfaz del software y permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.

Para desarrollar las pruebas de caja negra existen varios tipos de prueba, dentro de esta se encuentra la de función que es la que se utilizó. Este tipo de prueba se centra en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos y ejecutada contra diferentes objetos. La prueba de función debe enfocarse en los requisitos funcionales, las pruebas pueden estar basadas directamente en los casos de usos, requisitos o historias de usuarios. (23)

En la prueba de función se ejecuta cada caso de uso, flujo de caso de uso o función, usando datos válidos e inválidos, para verificar lo siguiente:

- Que los resultados esperados ocurran cuando se usen datos válidos.
- Que sean desplegados los mensajes apropiados de error y precaución cuando se usen datos inválidos.

3.7.2 Diseño de Caso de Prueba

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. Para ver los diseños remitirse al anexo 3.

3.7.3 Resultados

No	No Conformidad	Ubicación	Estado
1	La palabra área aparece sin tilde en la tabla de las áreas creadas.	Aseguramiento Logístico/ Área	Resuelta
2	El botón siguiente no funciona.	Aseguramiento Logístico/ Gestionar Local.	Resuelta
3	El campo Cantidad de Lámparas que lleva solo acepta números	Aseguramiento Logístico/ Gestionar Local.	Resuelta
4	El botón eliminar en la tabla ventana no funciona.	Aseguramiento Logístico/ Gestionar Local.	Resuelta

5	No deja adicionar una puerta	Aseguramiento Logístico/ Gestionar Local.	Resuelta
6	Error de java cuando se adiciona una unidad canina.	Aseguramiento Logístico/Gestionar Unidad Canina	Resuelta
7	Cuando se elimina una unidad canina no se le cambia el estado a la misma	Aseguramiento Logístico/Gestionar Unidad Canina	Resuelta
8	El mensaje de validación está incorrecto.	Aseguramiento Logístico/Gestionar Transporte	Resuelta
9	El mensaje de validación está incorrecto.	Aseguramiento Logístico/Gestionar Técnica Canina	Resuelta

3.8 Conclusiones

Con el desarrollo de este capítulo se ha implementado el diseño propuesto para dar cumplimiento a la propuesta de solución, generándose el diagrama de componentes y el diagrama de despliegue, que muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Seguidamente se realizaron las pruebas de función, a través del método de caja de negra. En general las pruebas de sistema ofrecieron una valiosa información que ayudó a la corrección de los errores identificados y a implementar un producto que satisface las necesidades del cliente.

Conclusiones generales

El presente documento contiene toda la documentación del proceso de desarrollo del módulo Aseguramiento Logístico perteneciente al subsistema Registro Legal del SIDEP. Se realizó un estudio teórico de las características que debía poseer un sistema de gestión penitenciaria y se analizaron algunos de los ya existentes, lo cual arrojó la primera conclusión de la investigación, implementar el modulo Aseguramiento Logístico del SIDEP. Las herramientas y definiciones arquitectónicas puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido. Como propuesta de solución se generan una serie de artefactos, basados en la metodología utilizada (RUP) orientados a elaborar el diseño e implementación del módulo. Los artefactos generados son la prueba del cumplimiento del objetivo general de esta investigación. Los patrones de diseño seleccionados para el desarrollo de la solución propuesta, formalizaron un vocabulario común entre el equipo de desarrollo. La validación de software constituyó un requisito de obligatorio cumplimiento en el proceso de desarrollo de la solución propuesta. Las pruebas de desarrollador realizadas posibilitaron la revisión detallada de cada uno de las funcionalidades desarrolladas, para de esa forma garantizar una mejor calidad. Con esta investigación se obtuvo el módulo Aseguramiento Logístico del SIDEP que da cumplimiento a los requisitos especificados.

Recomendaciones

Tomando como base la investigación realizada y los resultados obtenidos durante la realización de este trabajo, se recomienda lo siguiente:

- Realizar las pruebas de aceptación con el cliente.
- Desarrollar el manual de usuario de los módulos implementados para capacitar al personal que los emplearan.
- Desplegar la aplicación en los centros penitenciarios para analizar si es necesario agregar funcionalidades que brinden otros tipos de informaciones y servicios al usuario.
- Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas.

Bibliografía

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

2. **Zequeira Peña, Tte. Cnel. Dr. C. Alfonso Jesús.** Estudio teórico práctico de la administración de las capacidades en los centros de internamiento de la Dirección de Establecimientos Penitenciarios. *Informe.* Habana : Dirección de Establecimientos Penitenciarios, MININT, 2009.

3. **Pérez Lemes, Lisbel y Rodríguez López, José Miguel.** *Diseño e implementación de los módulos Ubicación, Capacidades y Evaluaciones Técnicas del Sistema de Gestión Penitenciaria.* Habana : UCI, 2010.

4. **JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES.** *El Proceso Unificado de desarrollo del software.* Madrid : sn, 2000.

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

6. **Dan Pilone, Neil Pitman.** *UML 2.0 in a Nutshell .* s.l. : O'Reilly Media, 2005.

7. Universidad EAFIT. [En línea] [Citado el: 12 de enero de 2012.] http://bdigital.eafit.edu.co/ARTICULO/HRU0380410137200505/13705.pdf.

8. **Visual Paradigm International.** Visual Paradigm. [En línea] [Citado el: 14 de enero de 2010.] http://www.visual-parading.com.

9. **Crawford , William y Kaplan , Jonathan.** *J2EE Design Patterns.* California : O'Reilly & Associates, 2003.

10. **Brito, Nacho.** Manual de desarrollo web con Grails. [En línea] [Citado el: 13 de 12 de 2011.] http://www.manual-de-grails.es.

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

12. **DAVE CRANE, ERIC PASCARELLO,DARREN JAMES.** *Ajax in Action.* s.l. : Manning Publications Co, 2006.

13. **Rawld Gill, Craig Riecke, Alex Russell.** *Matering Dojo.* 2008.

14. **SpringSource.** STS. [En línea] [Citado el: 10 de 4 de 2012.] http://www.springsource.org/sts.

15. **Tomcat, Apache.** Apache Tomcat. [En línea] [Citado el: 3 de 3 de 2012.] http://www.apache.org.

16. **ORACLE CORPORATION.** *Oracle Database Documentation Library.*

17. **Sommerville, Ian.** *Ingeniería de software.* 2008.

18. **Pressman, Roger S.** *Ingeniería de Software un enfoque práctico.* s.l. : Quinta Edición., 2005.

19. **Larman, Craig.** *UML Y Patrones. Introducción al análisis y diseño orientado a objeto.* Mexico : s.n.

20. **SpringSource.** springsource. *springsource.* [En línea] [Citado el: 25 de 4 de 2012.] http://static.springsource.org/spring-security/site/docs/3.1.x/reference/introduction.html.

21. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana.** *Diagrama de Despliegue.*

22. **Pressman, Roger S.** *Ingeniería de Software un enfoque práctico.* s.l. : Quinta Edición.

23. **Peña, Ing. Yadira Machado.** *Estrategia de Pruebas de función.* Habana : Universidad de las Ciencias Informáticas.

24. **Zequeira Peña, Dr. Tte.Cnel. Alfonso y Céspedes Quesada, Lic. 1erTte. Aimeé.** *Vocabulario Jurídico Penitenciario.* Ciudad de la Habana : MININT, 2005.

25. **The Dojo Foundation.** DojoTolkit. [En línea] http://www.dojotoolkit.org/.

26. **Software, Departamento de Ingeniería y Gestión de.** EVA. [En línea] [Citado el: 25 de 3 de 2012.] http://evapostgrado.uci.cu/mod/resource/view.php?id=11072.

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

28. **Embarcadero Technologies Inc.** Embarcadero. [En línea] [Citado el: 28 de 11 de 2011.] http://www.embarcadero.com/products/er-studio.

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

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

31. **Subversion.** Subversion. [En línea] 2001-2009. [Citado el: 28 de 11 de 2011.] http://subversion.tigris.org/.

32. **Oracle.** *Oracle Database Documentation Library.* 2005.

33. **Json.org.** Introducción a JSON. [En línea] [Citado el: 28 de 11 de 2011.] http://www.json.org/json-es.html.

34. **JavaScript.com.** JavaScript.com. [En línea] 2011. [Citado el: 28 de 11 de 2011.] http://www.javascript.com/.

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

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

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

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

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

40. **Abdul-Jawad, Bashar.** *Groovy and Grails Recipes.* 2009. ISBN-13 (pbk): 978-1-4302-1600-1, ISBN-13 (electronic): 978-1-4302-1601-8.

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

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

ANEXOS

Anexo 1 Diagramas de clase del diseño

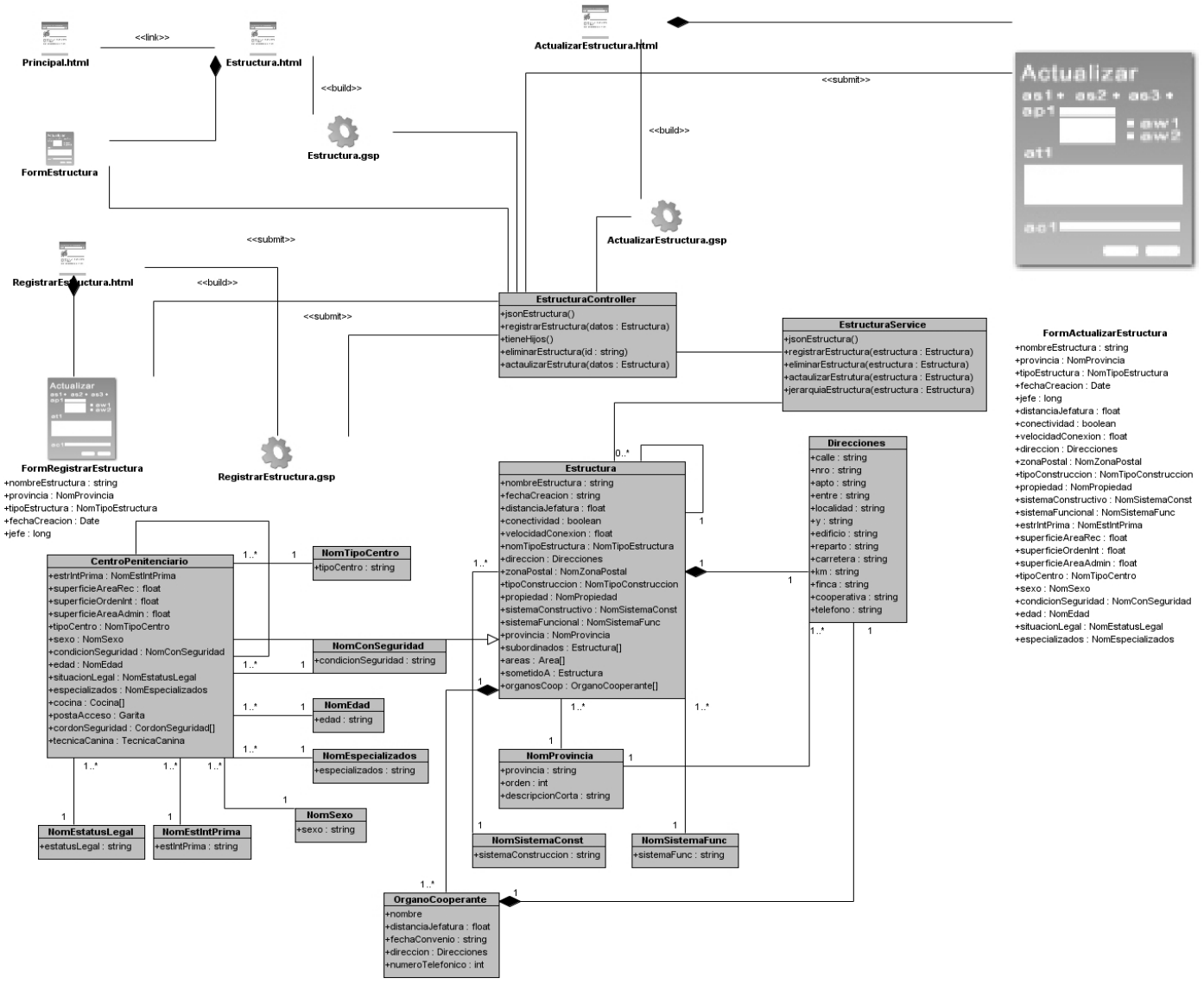

Ilustración 17: DCD CU Estructura

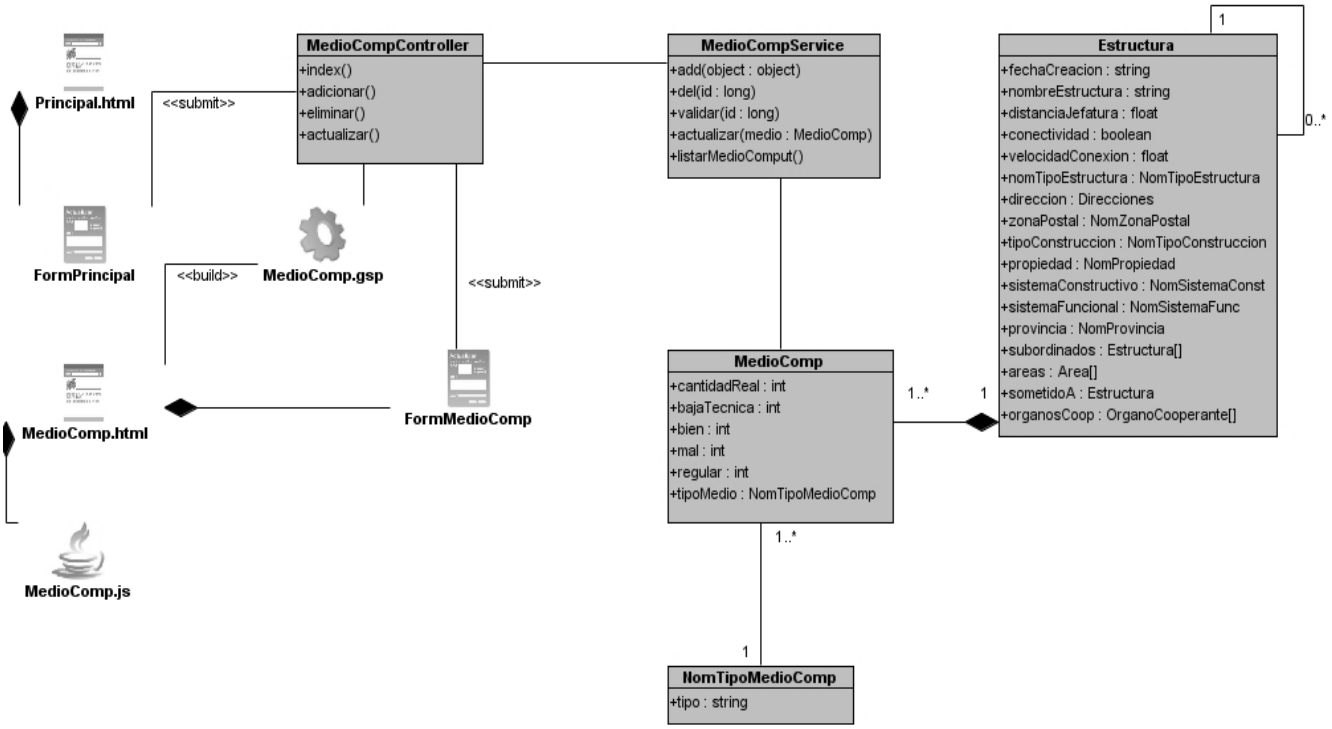

Ilustración 18: DCD CU Medio Computarizado

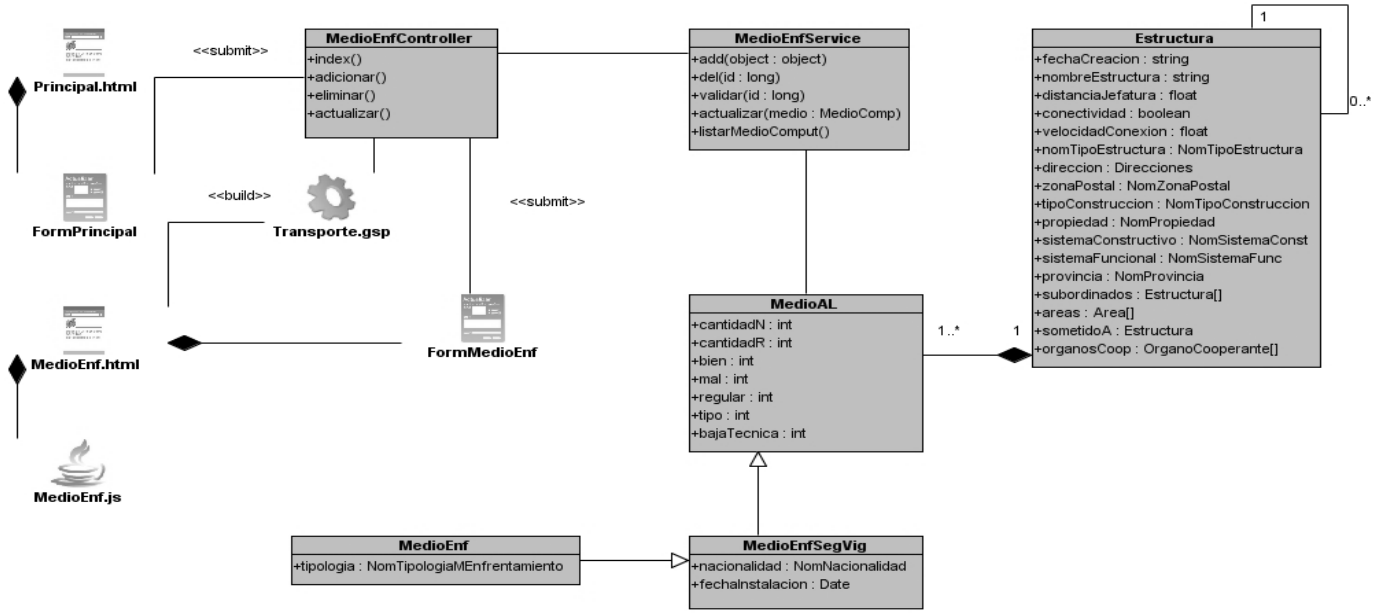

Ilustración 19: DCD CU Medio Enfrentamiento

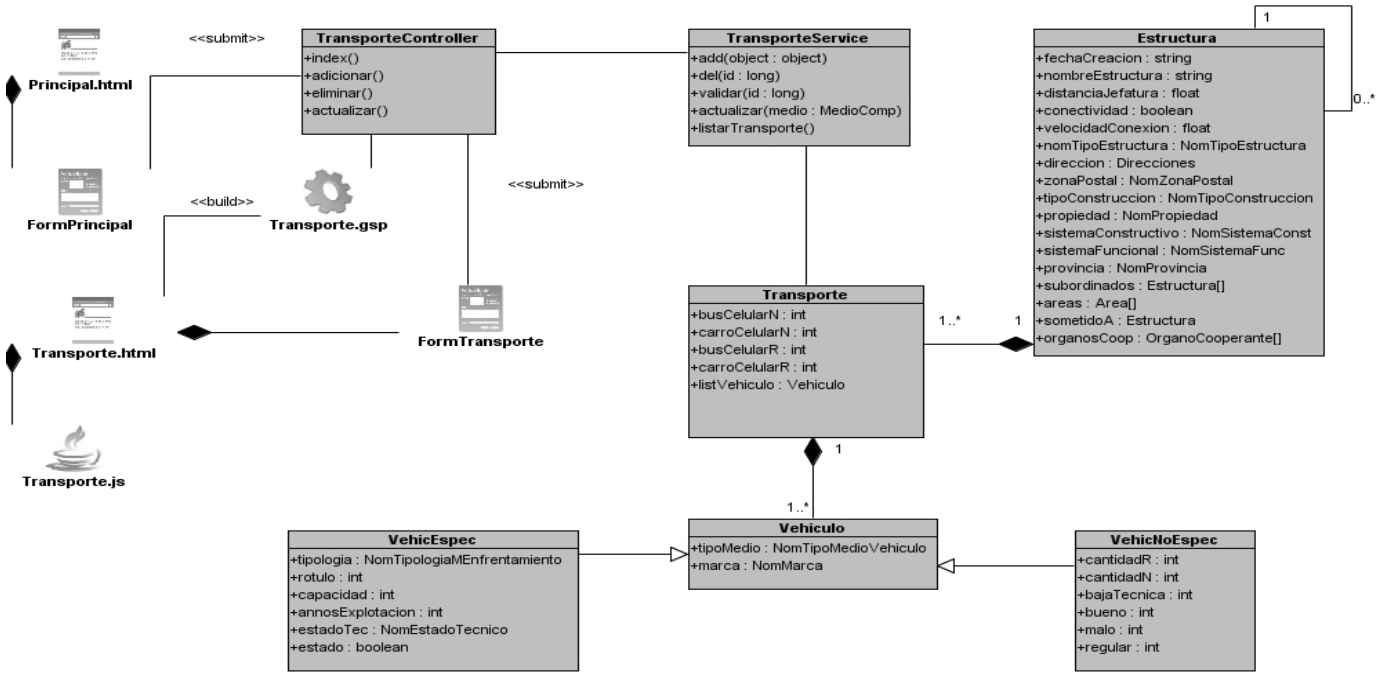

Ilustración 20: DCD CU Transporte

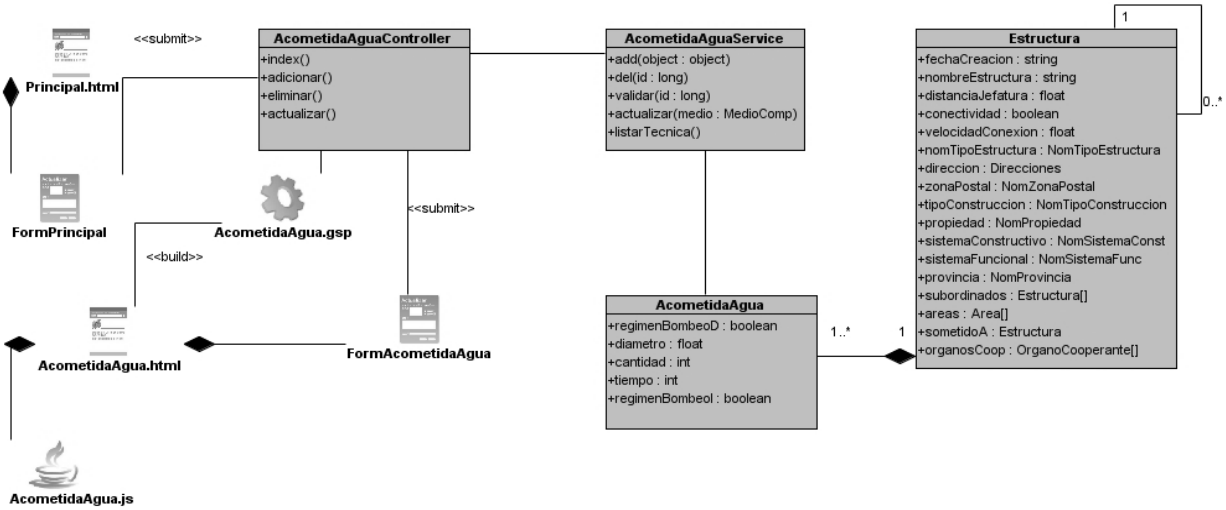

Ilustración 21: DCD CU Acometida de Agua

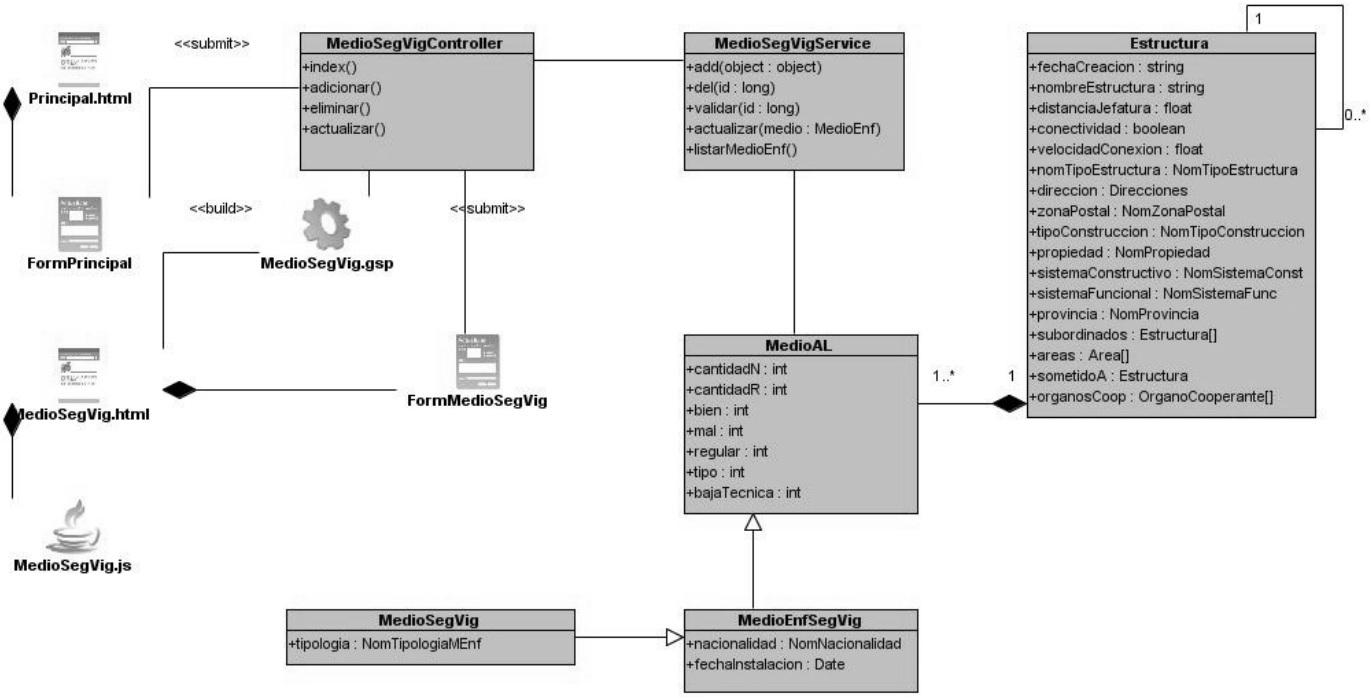

Ilustración 22: DCD CU Medio Seguridad y vigilancia

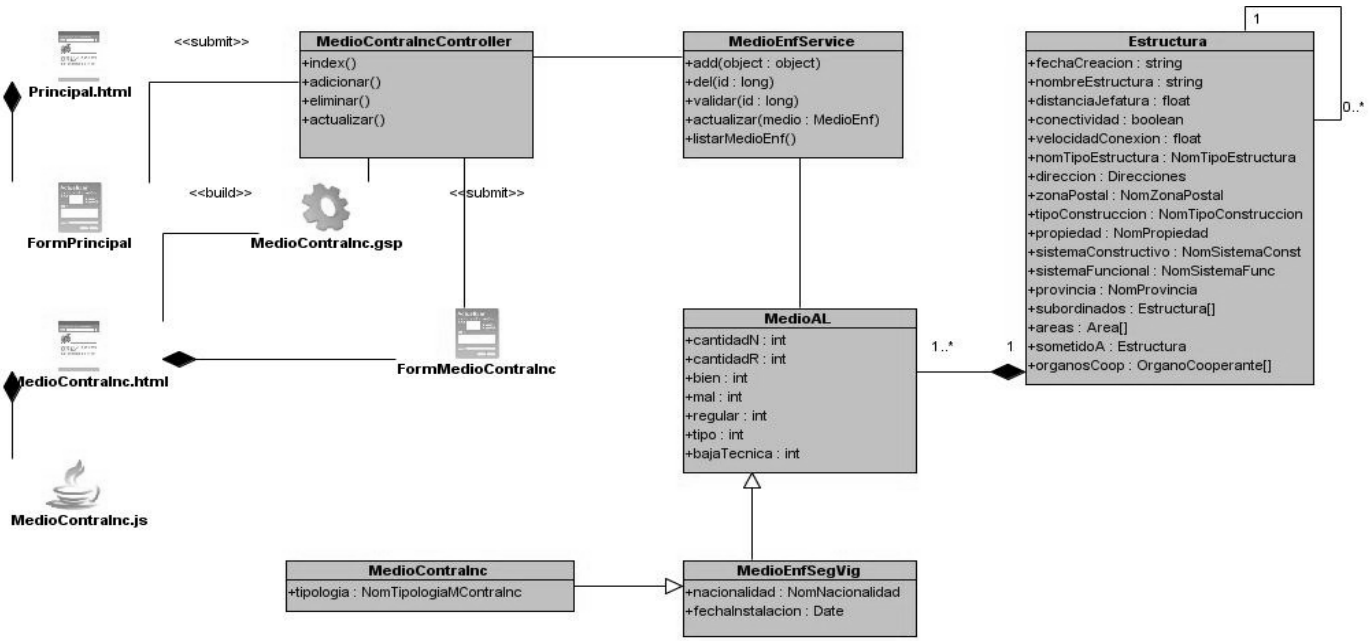

Ilustración 23: DCD CU Medio Contra Incendio

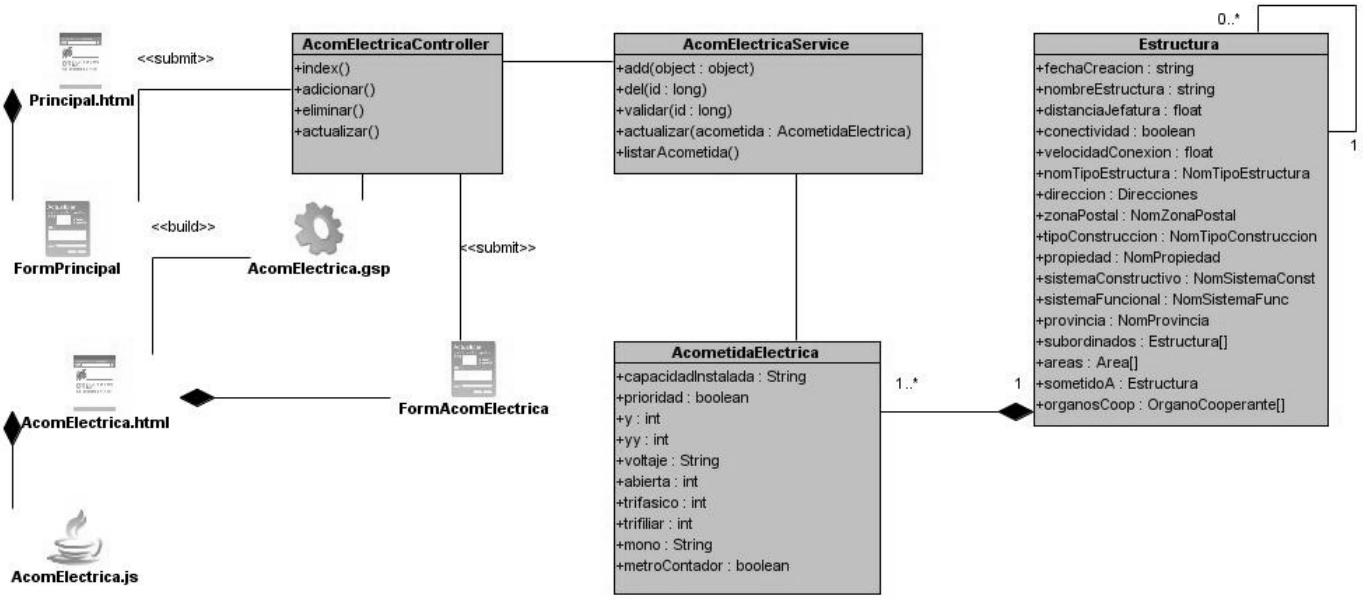

Ilustración 24: DCD CU Acometida Eléctrica

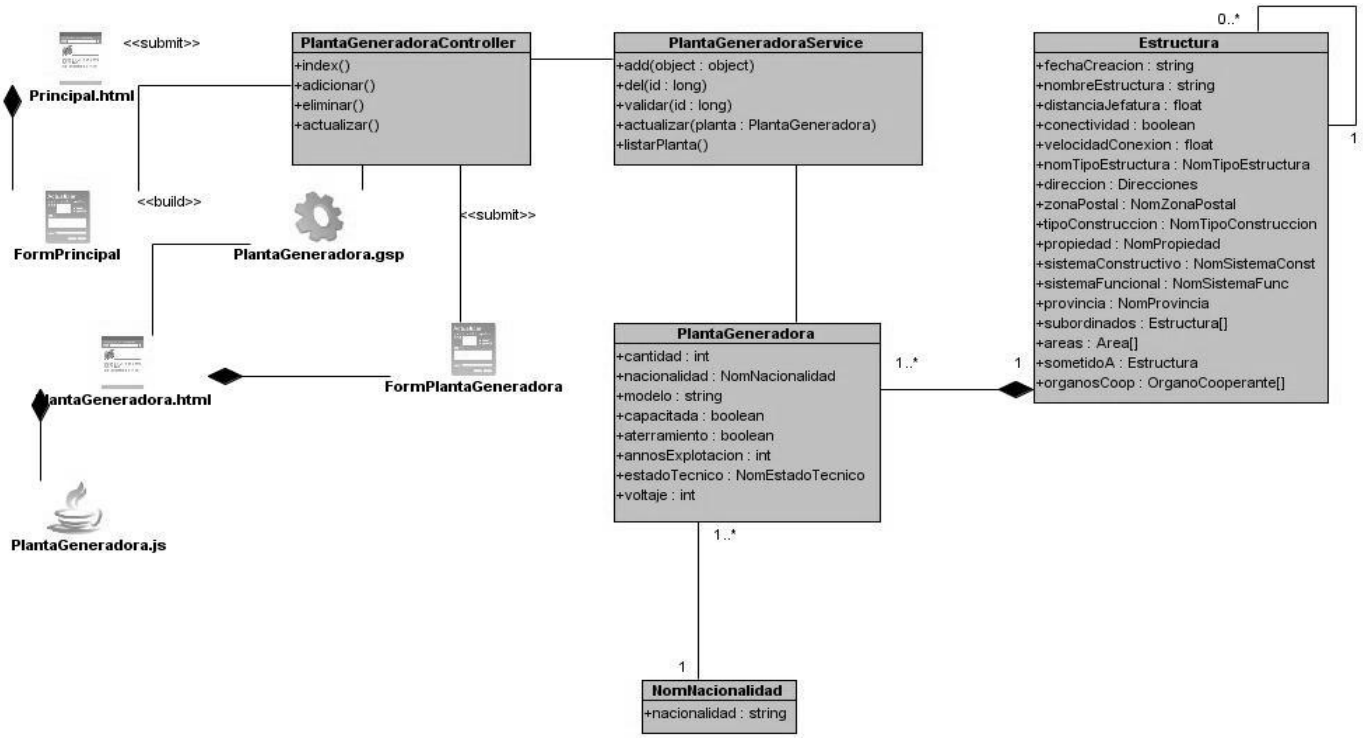

Ilustración 25: DCD CU Planta Generadora

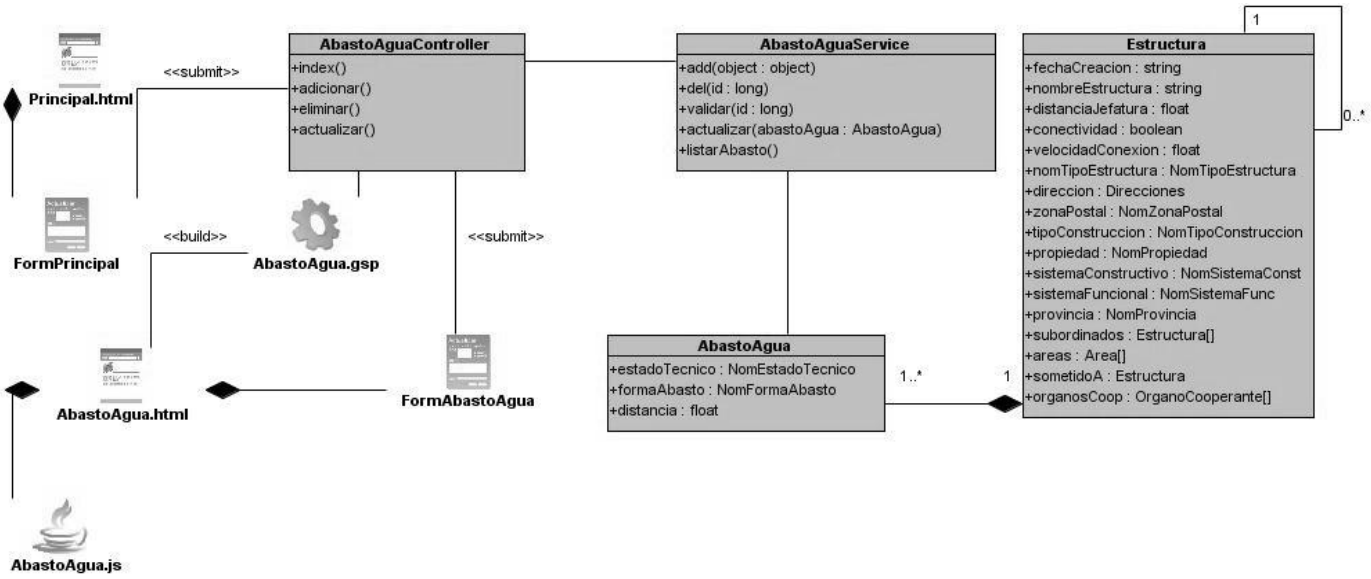

Ilustración 26: DCD CU Abasto Agua

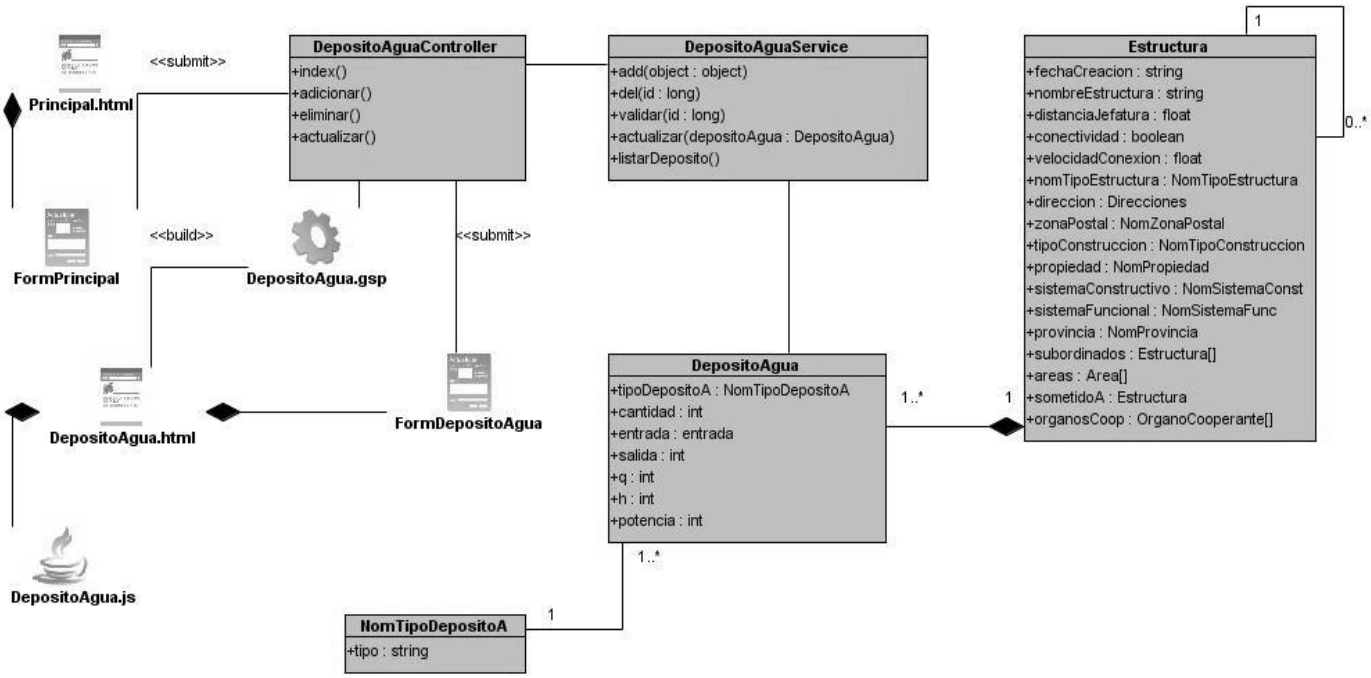

Ilustración 27: DCD CU Depósito Agua

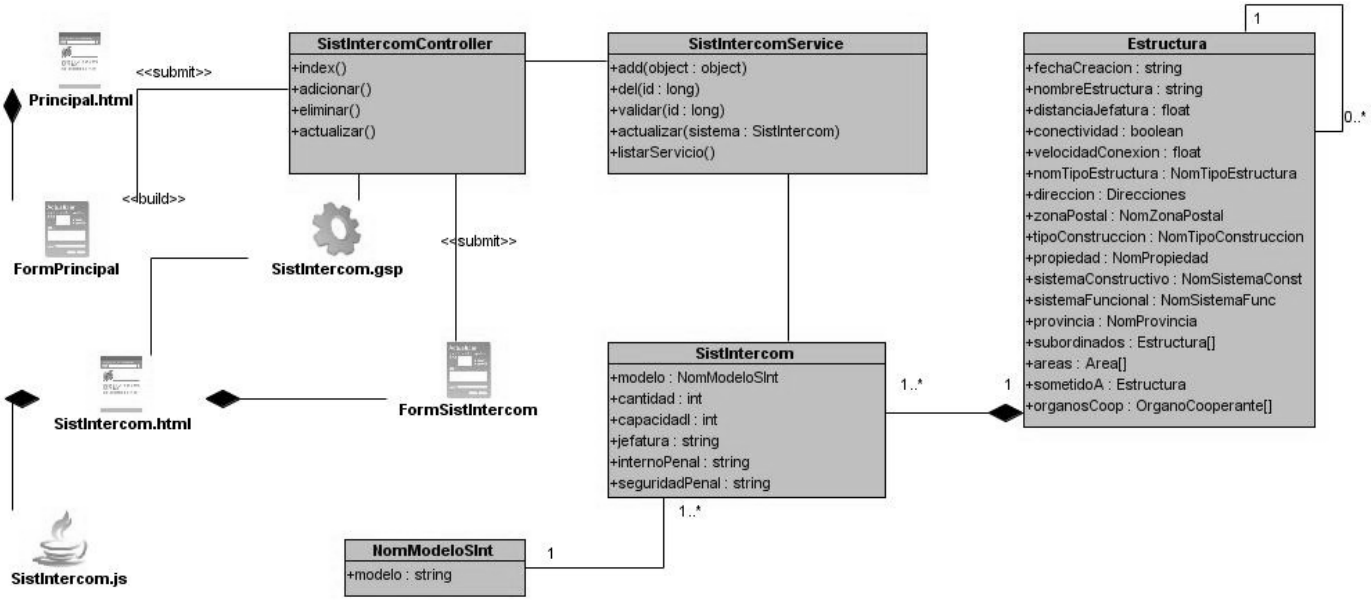

Ilustración 28: DCD CU Sistema Intercomunicadores

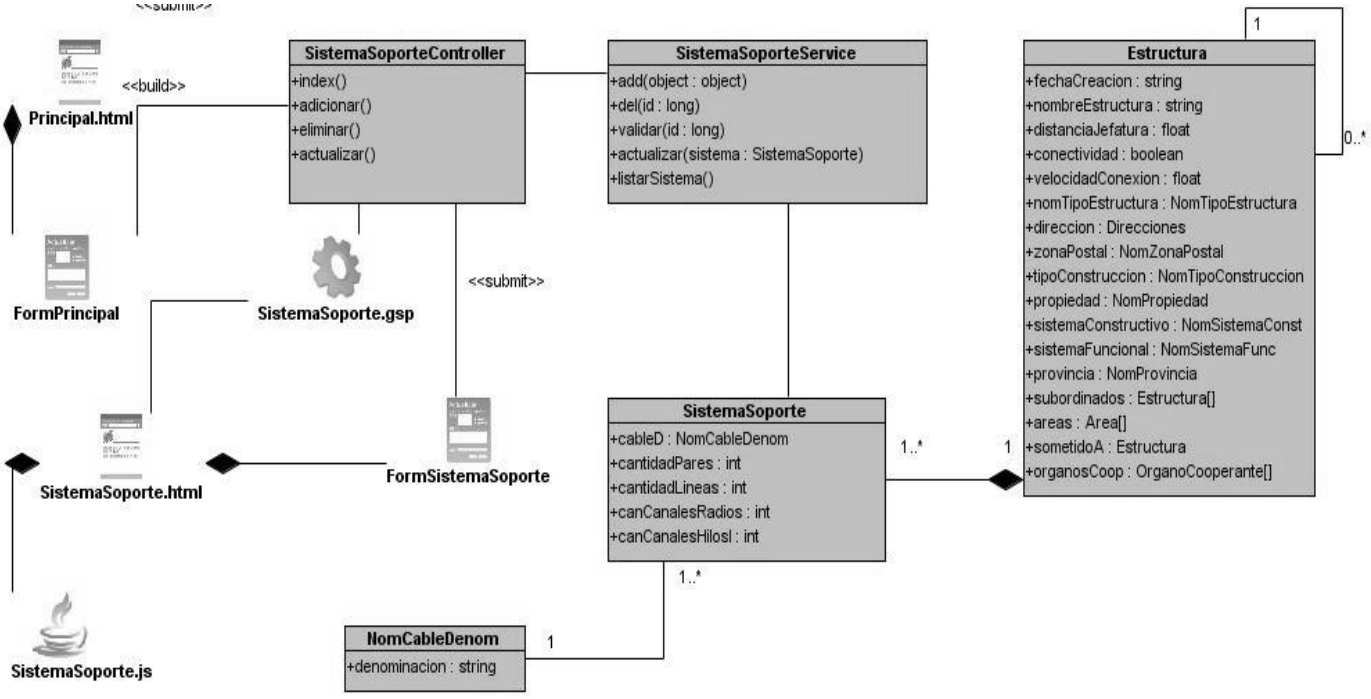

Ilustración 29: DCD CU Sistema Soporte

Anexo 2 Diagramas de Interacción

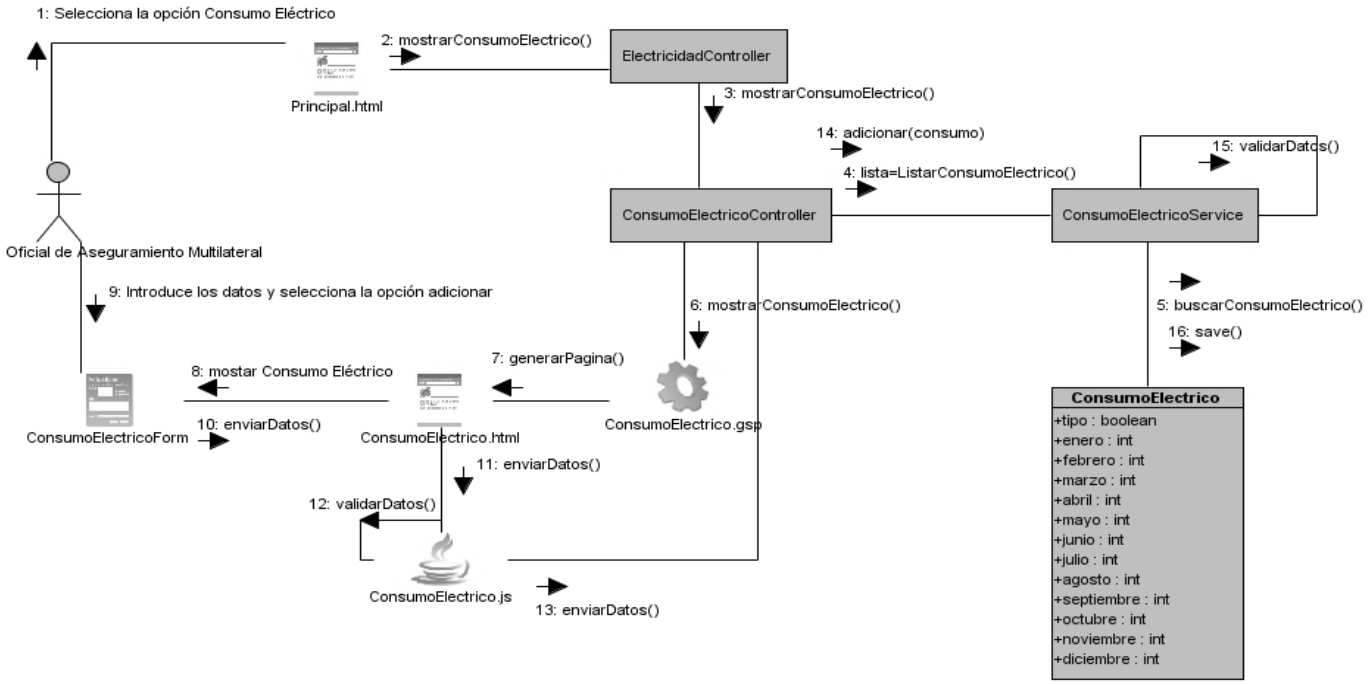

Ilustración 30: DC Registrar Consumo Eléctrico

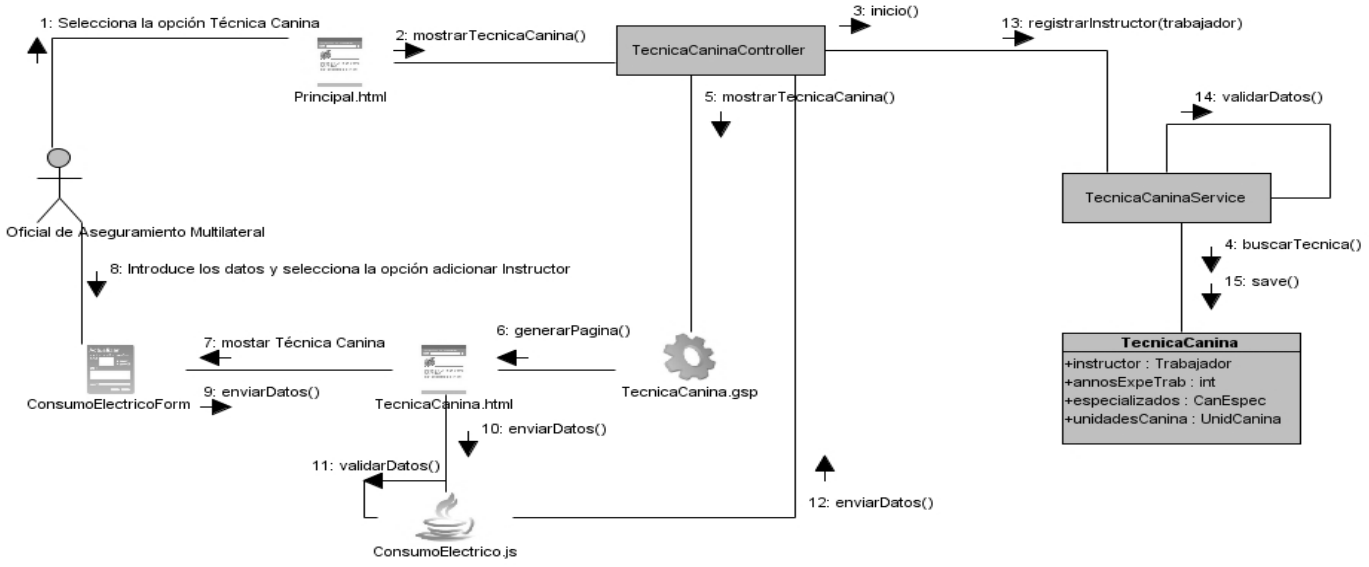

Ilustración 31: DC Registrar Instructor a la Técnica Canina

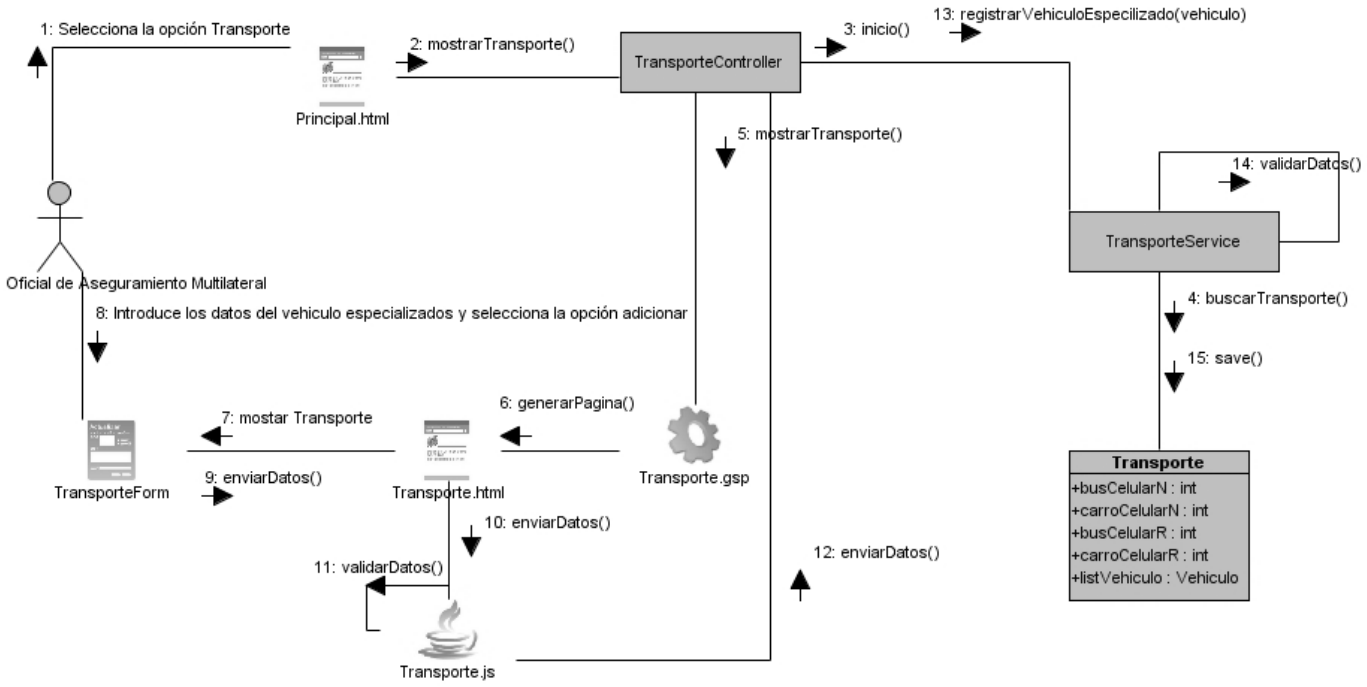

Ilustración 32: DC Registrar Transporte Especializado

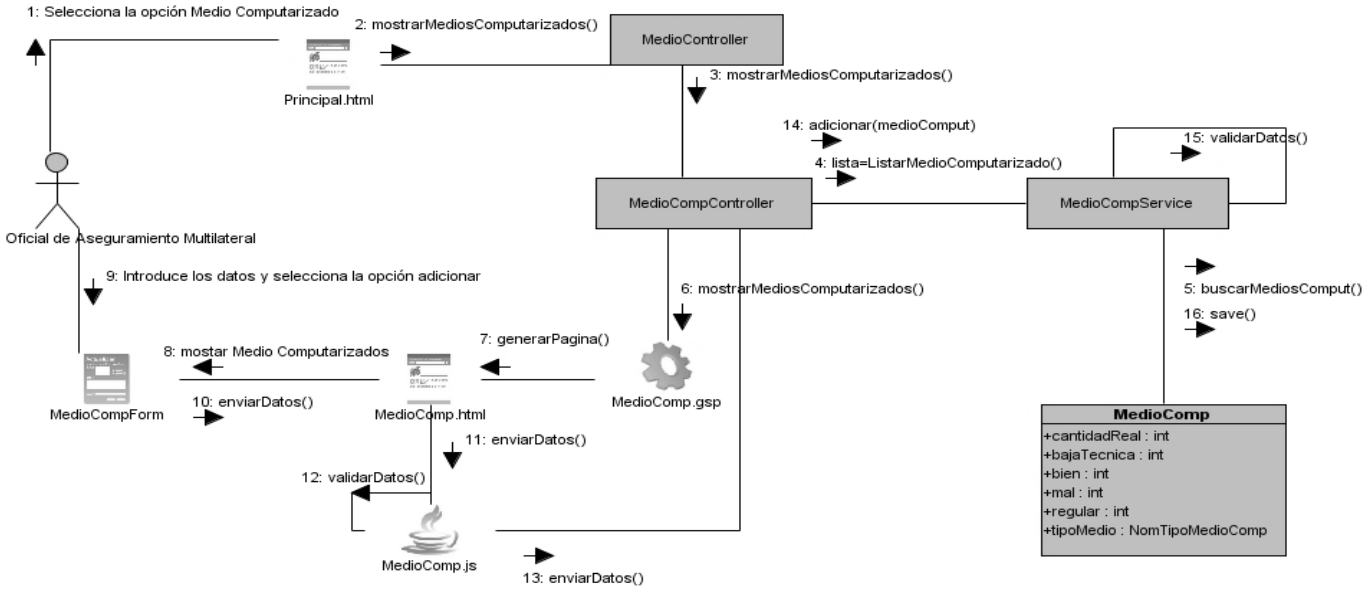

Ilustración 33: DC Registrar Medio Computarizado

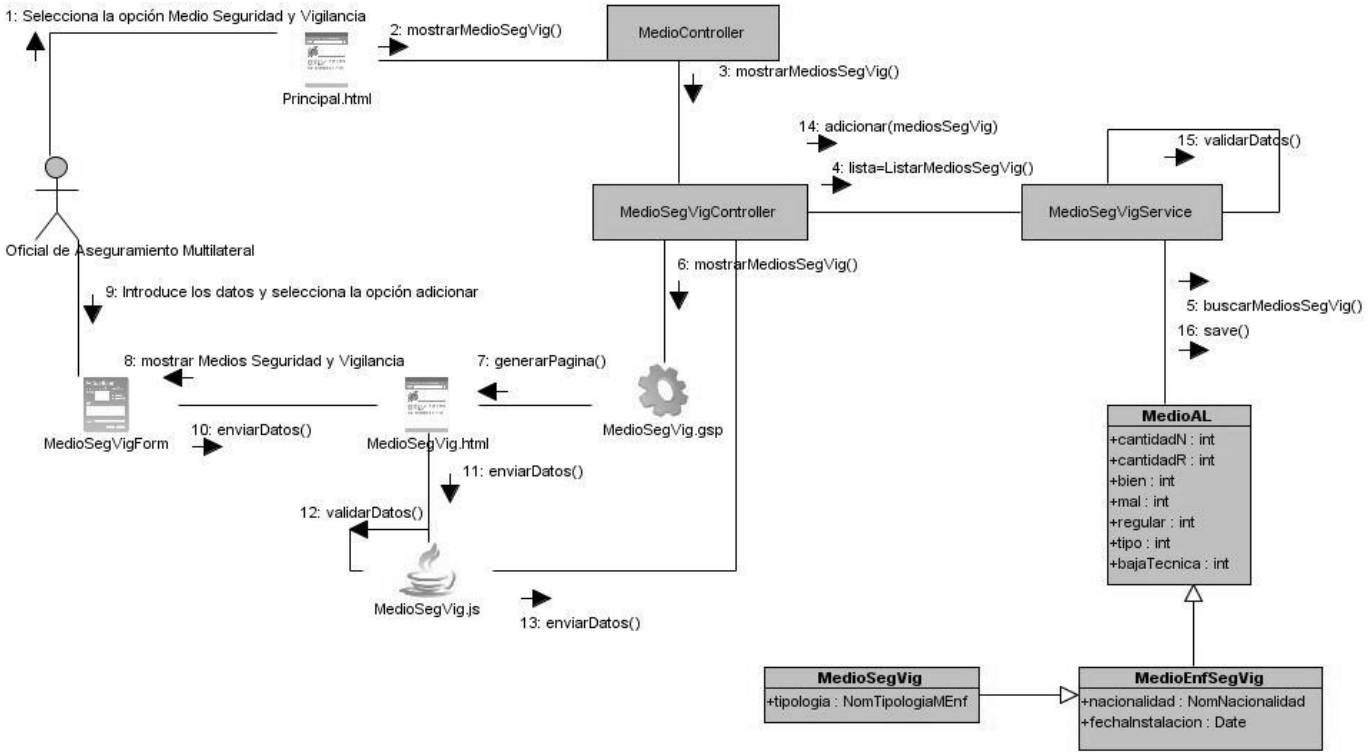

Ilustración 34: DC Registrar Medio Seguridad y Vigilancia

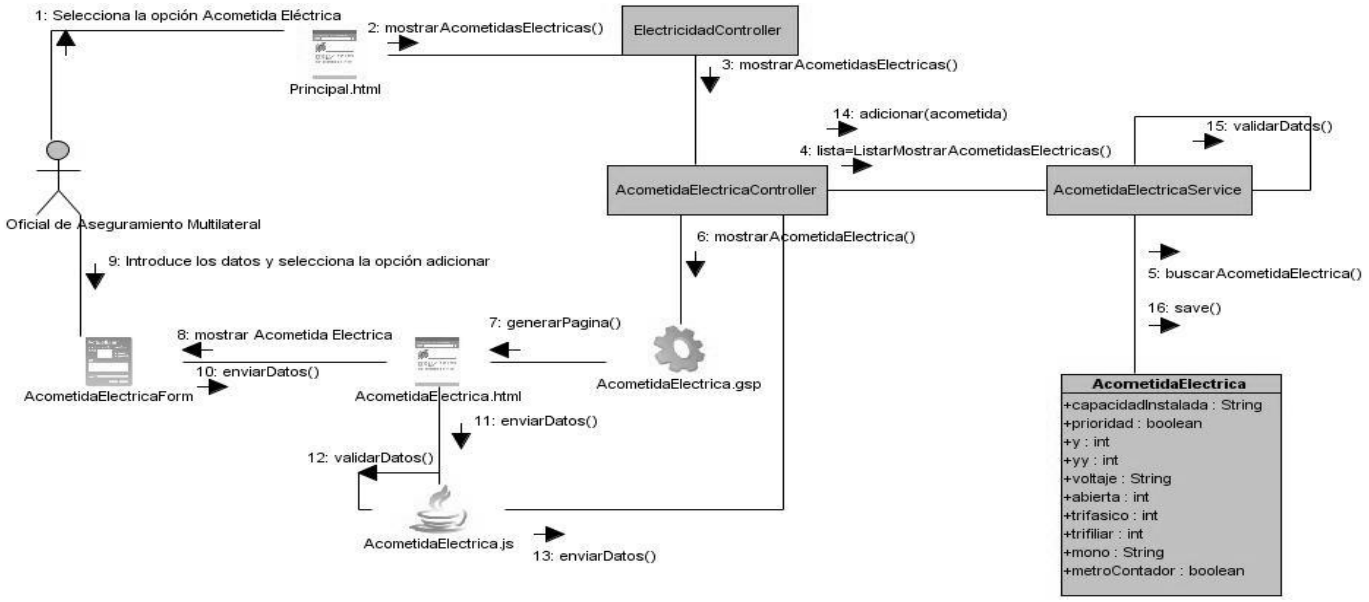

Ilustración 35: DC Registrar Acometida Eléctrica

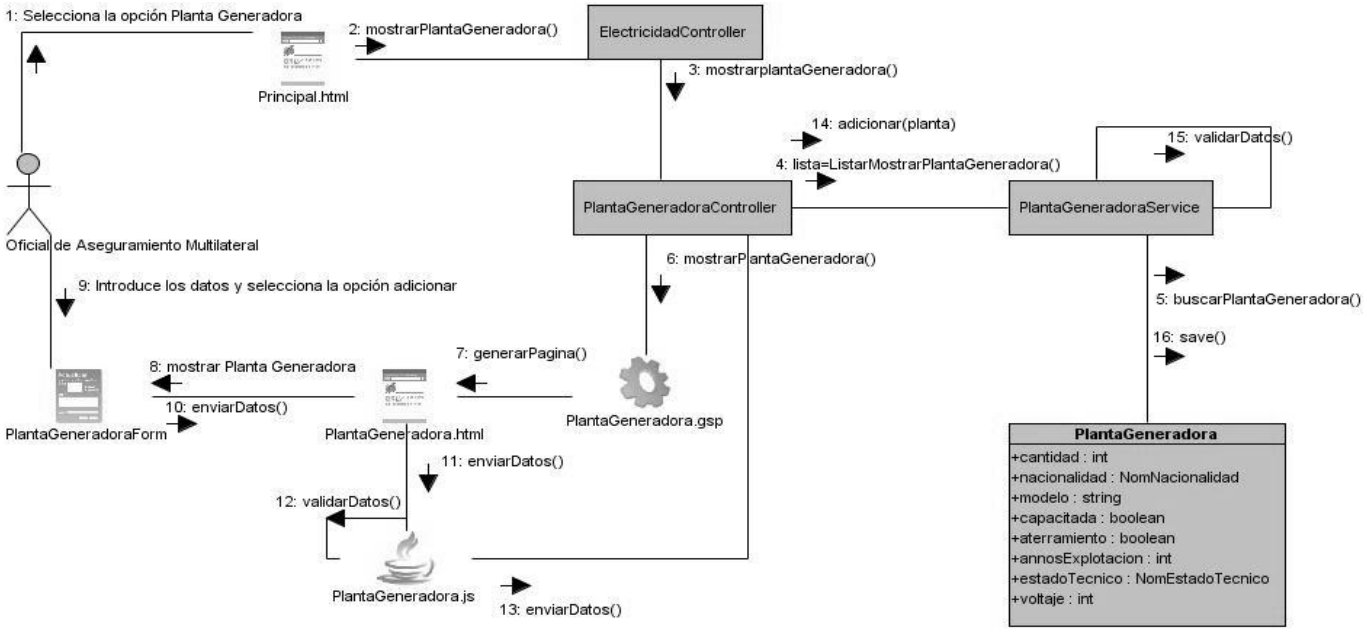

Ilustración 36: DC Registrar Planta Generadora

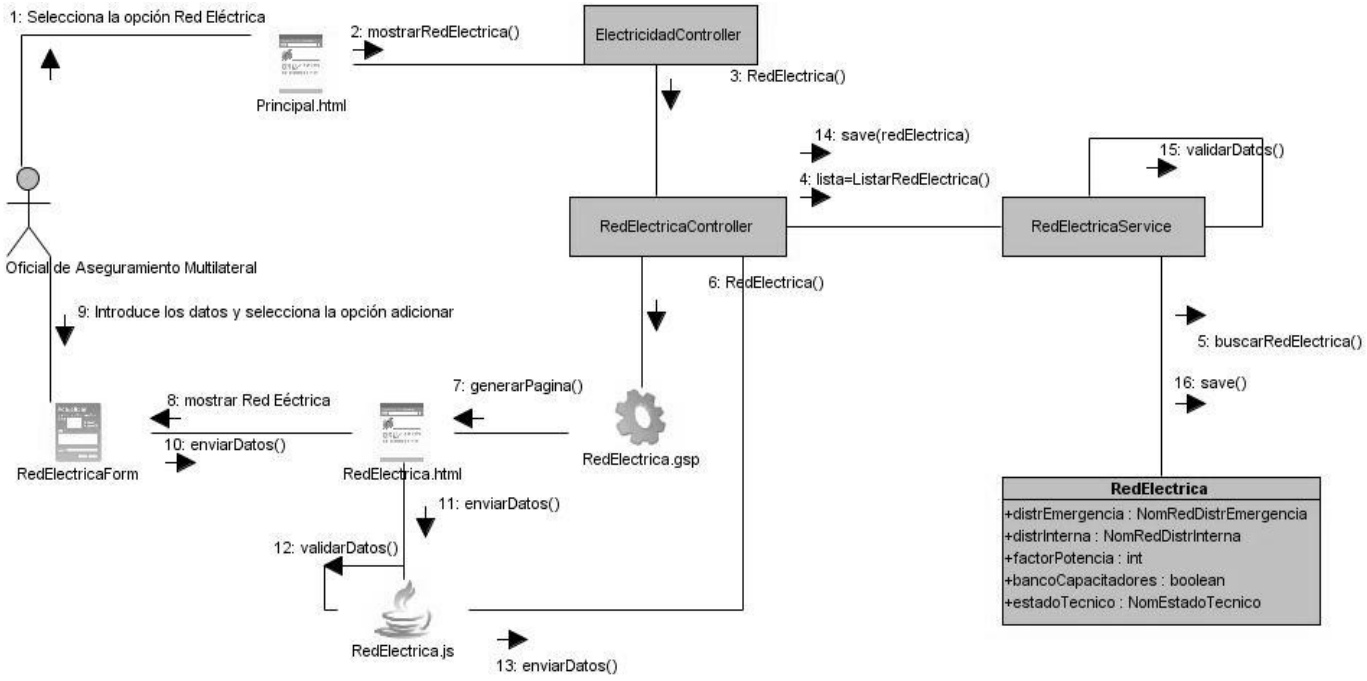

Ilustración 37: DC Actualizar Red Eléctrica

Anexo 3 Diseño de Caso de Prueba

Escenario	Descripción	Nombre	Tipo	Provincia	Fecha de creación	Respuesta del sistema	Flujo central
EC 1.1 Registrar correctamente los datos de la estructura.	*Registrar correctamente los datos de la estructura..*	*V*	*V*	*V*	*V*	*El sistema muestra un arbol jerarquico con las estructuras; valida los datos introducidos y luego registra los datos de la estructura.*	*1. Seleccionar la opción "Estructura".* *2. Seleccionar la estructura superior.* *3. Oprimir el botón "Registrar".* *4. Introducir los datos obligatorios solicitados.* *5. Oprimir el botón Aceptar.*
EC 1.2 Faltan datos obligatorio en el Registrar Estructura.	*Se verifica que no existan campos obligatorios incompletos cuando se registre los datos de la estructura.*	*V*	*V*	*V*	*V*	*El sistema muestra el mensaje de error "Introduzca los datos obligatorios" y señala los campos obligatorios donde no se introdujeron datos.*	*1. Seleccionar la opción "Estructura".* *2. Seleccionar la estructura superior.* *3. Oprimir el botón "Registrar".* *4. Oprimir el botón Aceptar.*

Ilustración 38: CP Estructura

EC 1.3 Cancelar registrar datos de la estructura.	Cancelar registrar datos de la estructura.	V	V	V	V	El sistema no registra datos de la estructura, y oculta los campos de registrar.	1. Seleccionar la opción "Estructura". 2. Seleccionar la estructura superior. 3. Oprimir el botón "Registrar". 4. Introducir los datos obligatorios solicitados. 5. Oprimir el botón Cancelar.
EC 1.4 Eliminar Estructura sin estructuras asignadas a él.	Eliminar Estructura sin estructuras asignadas a él.	NA	NA	NA	NA	El sistema muestra un mensaje "¿Está seguro que desea eliminar la estructura seleccionada?" ; selecciona la opción aceptar y elimimina la estructura.	1. Seleccionar la opción "Estructura". 2. Seleccionar una estructura sin subordinados. 3. Oprimir el botón "Eliminar".
EC 1.5 Eliminar Estructura con estructuras asignadas a él.	Eliminar Estructura sin estructuras asignadas a él.	NA	NA	NA	NA	El sistema muestra un mensaje "No se puede eliminar la estructura seleccionada"	1. Seleccionar la opción "Estructura". 2. Seleccionar una estructura con subordinados. 3. Oprimir el botón "Eliminar".
EC 1.6 Actualizar correctamente los datos de la estructura.	Actualizar correctamente los datos de la estructura..	V	V	V	V	El sistema muestra un arbol jerarquico con las estructuras; valida los datos introducidos y luego actualiza los datos de la estructura.	1. Seleccionar la opción "Estructura". 2. Seleccionar la estructura superior. 3. Oprimir el botón "Actualizar". 4. Introducir los datos obligatorios solicitados. 5. Oprimir el botón Aceptar.
EC 1.7 Faltan datos obligatorio en el Actualizar Estructura.	Se verifica que no existan campos obligatorios incompletos cuando se actualice los datos de la estructura.	V	V	V	V	El sistema muestra el mensaje de error "Introduzca los datos obligatorios" y señala los campos obligatorios donde no se introdujeron datos.	1. Seleccionar la opción "Estructura". 2. Seleccionar la estructura superior. 3. Oprimir el botón "Actualizar. 4. Oprimir el botón Aceptar.
EC 1.8 Cancelar Actualizar datos de la estructura.	Cancelar actualizardatos de la estructura.	V	V	V	V	El sistema no actualiza datos de la estructura, y oculta los campos de registrar.	1. Seleccionar la opción "Estructura". 2. Seleccionar la estructura superior. 3. Oprimir el botón "Actualizar". 4. Introducir los datos obligatorios solicitados. 5. Oprimir el botón Cancelar.

Ilustración 39: CP Estructura 2

Glosario de Términos

API: Interfaz de programación de aplicaciones o application programming interface, conjunto de funciones, procedimientos o métodos que ofrece una librería para ser utilizado por otro software.

Bytecodes: es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

Conduces: Es el proceso que se realiza en el sistema penitenciario, cuando a un recluso se le traslada de la prisión a un juicio, hospital, etc.

DOM: Document Object Model (modelo de objeto del documento) es esencialmente una interfaz de programación de aplicaciones (API) que proporciona 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 manipularlos.

Framework: Marco de trabajo, esquema para el desarrollo y/o la implementación de una aplicación.

GUI: Graphical user interface (interfaz gráfica de usuario) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

JSON: JavaScript Object Notation (objeto de notación de javaScript), es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

UI: User interface (Interfaz de Usuario), es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

XHTML: Es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros.

XML: Extensible Markup Language (lenguaje de marcas extensibles), es un lenguaje que permite definir la gramática de lenguajes específicos.