Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Desarrollo del módulo Conducción del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

Autor: Ernesto Benitez Oliva

Tutor: Ing. Guillermo Enrique Ferras Pérez

Co-Tutora: Ing. Yanay Viera Lorenzo

La Habana, Junio de 2012

Tu tiempo es limitado, de modo que no lo malgastes viviendo la vida de alguien distinto. No quedes atrapado en el dogma, que es vivir como otros piensan que deberías vivir. No dejes que los ruidos de las opiniones de los demás acallen tu propia voz interior. Y, lo que es más importante, ten el coraje para hacer lo que te dicen tu corazón y tu intuición.

Steve Jobs

Declaración de Autoría

Declaramos ser autores 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 firmamos la presente a los ____ días del mes de _____ del año 2012.

Ernesto Benitez Oliva

_____
Firma del Autor

Ing. Guillermo Enrique Ferras Pérez

_____
Firma del Tutor

Ing. Yanay Viera Lorenzo

_____
Firma de la Co-Tutora

Agradecimientos

A mis abuelas, en especial a Romelia, mi mamá, a mi papá, a mis abuelos sobre todo a Argelio, mi padrastro Joel que se ha a acogido como un hijo más; a mis tías en especial a María Josefa que ha sido como una segunda madre, a Odalis, Rudis, Rosalba, a Rubí a mi tío Ricardo, Felito y Diego por preocuparse por mí, por su confianza y apoyo. A mis hermanos Camilo, Daira y Leidi. A mis primas en especial Greisi, Yoenia. A toda mi familia por estar siempre cuidándome y dándome fuerzas para continuar. A mis compañeros que han estado conmigo y que me han ayudado de una u otra forma en especial a Rene, Carlos, Dariel, Vladimir, Alkaid, Bernardo, Leandro, Reinier, Alexander, Juan Carlos, Yaidel, Bárbaro (el sobrino) Juan Carlos Elián, a la tía Beba, Maikel David, Yudisleidy, Deisy, Aparicio, Daril, Yasel y Sulay, al piquete de las fanáticas y todos los demás que faltaron por mencionar. A todos mis amigos de Palma sobre todos al piquete de la Cuba Rafael, Adrián, Sergio, Alain, Omar, Natalí, Yasel Cambara, Miguel, Radames, Daguel y demás; a mis amigos de la playa Abertico, Tinito, David, Mabelita, Rosarito, Sergitin a Rosalia y su abuela Lola por darme apoyo moral y por creer en mí. No me podía faltar mi queridísima cotutora por no haberse dado por vencida y tener paciencia conmigo y a mi tutor Guillermo por el apoyo brindado. A la familia de Carlos por haberme acogido como un miembro más. Por último a todas aquellas personas que me han ayudado a lo largo de estos 5 años de forma directa o indirecta. Muchas gracias a todos, sin su apoyo este trabajo no hubiera sido posible.

Dedicatoria

Especialmente a mi abuela Romelia, a mi mamá a mi revolución por darme la oportunidad de estudiar y todas aquellas personas que me apoyaron de una forma u otra.

Resumen

Actualmente en el Sistema Automatizado para el Control del Recluso (SACORE) no se realiza una planificación de los procesos de conducción lo que provoca la falta de control de los mismos. El presente trabajo propone el diseño, implementación y prueba del módulo Conducción el cual mejora la planificación y ejecución de los conduces en el Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP). Como solución al problema planteado se desarrolló una aplicación web, haciendo uso del framework de desarrollo Grails y lenguaje Groovy, gestor de base de datos Oracle 11g, las tecnologías Dojo, J2EE y la metodología RUP que guió el proceso de desarrollo. Se describen el estudio de estado del arte, la estrategia de pruebas utilizada, así como los diagramas y tablas generados durante los flujos de trabajo desarrollados.

Índice

Índice de figuras

Introducción

En Cuba se ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible. Los recursos de Tecnologías y Sistemas de Información en Cuba son cada vez más utilizados y el Sistema Penitenciario Cubano no ha sido excepción. La automatización del Sistema Penitenciario a nivel nacional comienza en el año 1989, con un sistema de control penal, que permitía la gestión de los principales datos del recluso y algunos aspectos del trabajo de la especialidad.

Los fundamentos de la política penitenciaria cubana están determinados en la Constitución de la República. Le corresponde así dentro de este sistema a los establecimientos penitenciarios, garantizar la ejecución de las sanciones privativas de libertad, alcanzar el fin humanitario de reeducar a los delincuentes y retornarlos a la sociedad como personas útiles.

En cumplimiento de la orden 43/99 del Viceministro Primero del Interior se diseña, programa y pone en funcionamiento el SACORE utilizando la tecnología existente en ese entonces en el Ministerio del Interior (MININT).

En sus seis años de explotación se ha ido perfeccionando a partir de nuevos requerimientos y solicitudes de los usuarios, conteniendo como módulos principales Control Penal, Tratamiento Educativo y Orden Interior; los dos últimos con pocas funcionalidades y salidas pendientes. Además se desarrollaron otros dos sistemas automatizados que complementan la información del SACORE: Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios (SACDEP) y Sistema Automatizado de Incidencias del Departamento de Establecimientos Penitenciarios (SAIDEP).

Actualmente se encuentra en pleno desarrollo un nuevo sistema para la gestión penitenciaria, el SIDEP dentro del cual se encuentra el módulo Conducción dentro del subsistema Registro Legal. La conducción no es más que la acción de presentar bajo custodia a un interno en determinado lugar donde su permanencia es temporal. Este módulo permite gestionar la planificación de las conducciones. Aquí se agrupan los conduces a hospital, a tribunal, a las funerarias y domicilio especificadas en la ley como trámites, además el módulo controla la ejecución de las salidas planificadas y el retorno de los internos.

Entre los problemas que afectan la planificación y ejecución de las conducciones en el Sistema Penitenciario Cubano se encuentran:

- Ninguno de los sistemas informáticos mencionados anteriormente permiten llevar un control del Plan de Seguridad de los conduces que lo requieren lo que provoca que no se tenga en cuenta la peligrosidad del interno que se desea conducir durante la planificación del conduce.
- Los conduces por urgencia quedan registrados solamente en el Libro de ocurrencias; son controlados de manera diferente al resto de los conduces y podrían quedar fuera del control que llevan los diferentes niveles de mando de Prisiones sobre la ejecución de los conduces.
- Hay atrasos en la aprobación de los conduces en los diferentes niveles de mando. Esto provoca demora en su ejecución, incluso aquellos que requieren ser ejecutados en días específicos.
- La gestión de los conduces de mayor severidad y de extranjeros se realizan por vía telefónica y no queda constancia de quien autorizó el conduce; no existe ningún documento legal donde se registre la conducción del interno.

Dada la situación anterior surge como **problema a resolver:** la ineficiente gestión de los procesos de conducción en el Sistema Penitenciario Cubano.

Se define como **objeto de estudio** de la investigación: los procesos de conducción en sistemas penitenciarios, enfocándose en el **campo de acción:** informatización de los procesos de conducción en el Sistema Penitenciario Cubano.

Para darle solución al problema planteado se trazó como **objetivo general**: desarrollar un módulo para optimizar la informatización de los procesos de conducción en el Sistema Penitenciario Cubano a partir del análisis previo realizado por los analistas del SIDEP.

El mismo se divide en los siguientes **objetivos específicos:**

1. Elaborar la fundamentación teórica para guiar el proceso de investigación.
2. Diseñar el módulo de Conducción para facilitar el flujo de trabajo implementación.
3. Implementar el módulo Conducción para mejorar la planificación y ejecución de los procesos de conducción.
4. Realizar pruebas de calidad al módulo Conducción para validar el producto desarrollado.

Se plantea como **idea a defender**: Con el desarrollo del módulo Conducción se mejorará la informatización de los procesos de conducción del Sistema Penitenciario Cubano.

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

1. Análisis de las soluciones informáticas nacionales e internacionales que implementen los procesos con el mismo perfil que Conducción.
2. Descripción de las herramientas y tecnologías para dar solución al problema.
3. Diseño del módulo Conducción.
4. Diseño de la base de datos del módulo Conducción.
5. Elaboración del diagrama de componentes del módulo Conducción.
6. Implementación del módulo Conducción.
7. Elaboración del diagrama de despliegue.
8. Diseño de las pruebas de calidad para el módulo Conducción.
9. Aplicación de las pruebas al módulo Conducción.
10. Respuesta a las no conformidades encontradas.

Los Métodos Científicos que se usarán para dar cumplimiento a las tareas son:

Teóricos:

- Analítico Sintético: se utiliza para el análisis de las herramientas, tecnologías y la metodología de desarrollo seleccionadas, identificando las características que las distinguen y se refleja la valoración de los autores sobre las mismas.
- Modelación: Se utiliza en el diseño del módulo a desarrollar, específicamente en la modelación de los diagramas de clases del diseño, de interacción, en los de componentes y en el diagrama de despliegue.
- Histórico lógico: este método fue utilizado para el análisis de sistemas informáticos con perfiles similares a los de conducción.

Empíricos:

- Análisis documental: Está dado por el análisis de los documentos generados por los analistas para el módulo Conducción: especificación de casos de usos, glosario de términos, diagramas de entidades, entre otros, extrayendo y analizando los principales elementos relacionados con el objeto de estudio.

El presente trabajo se encuentra dividido en tres capítulos. El primero titulado Fundamentación Teórica tiene como objetivo realizar un estudio de los sistemas informáticos con el mismo perfil de Conducción existentes en el mundo y en el país. Además, se analizarán las herramientas, tecnologías y la metodología a utilizar para el desarrollo de las tareas relacionadas con el diseño e implementación del SIDEP.

El capítulo dos titulado Diseño del Sistema, se muestra la descripción de la arquitectura y de los patrones de diseño a utilizar. Así como la descripción de los principales casos de usos mediante la utilización de los diagramas de clases del diseño y de comunicación. También se realizará el diseño de la base datos para el módulo Conducción.

El tercer capítulo nombrado Diseño e Implementación se generan los diagramas de despliegue y componentes, se implementa el sistema y se concluye realizando pruebas de caja negra para comprobar si el módulo cumple los requisitos especificados.

Capítulo I: Fundamentación Teórica

1.1. Introducción

Este capítulo tiene como objetivo fundamental ofrecer el marco teórico en el que se desarrolla el trabajo, además se explica la propuesta solución planteada por el SIDEP. Además, se efectuará el estudio del ambiente de desarrollo, ofreciendo las principales características de la metodología, lenguaje y herramientas que se utilizarán en la implementación de la solución.

1.2. Análisis de sistemas informáticos de gestión penitenciaria

1.2.1. Sistema de Gestión Penitenciaria

El Sistema de Gestión Penitenciaria (SIGEP) fue creado con el fin de automatizar los procesos penitenciarios en la República Bolivariana de Venezuela que se encargará de informatizar todo el recorrido de los penados por el sistema. Este permite el registro de toda la población privada de libertad y el seguimiento progresivo del privado de libertad desde que ingresa al Servicio Penitenciario hasta que egresa del mismo. Es decir, hace el seguimiento del privado de libertad desde que el interno se encuentra en el régimen intramuros, hasta que sale a cada una de las medidas alternativas de cumplimiento de la pena. (8)

La Salida Transitoria es aquella que implica un retorno del privado de libertad al establecimiento esta puede ser un traslado al hospital, a tribunal o uno de los permisos especiales especificados en la ley. (8)

A continuación se describen brevemente los procesos involucrados con salidas transitorias.

Planificación de Salida Transitoria

El proceso de planificación de una salida transitoria se inicia una vez que el tribunal correspondiente a la causa de un interno envía una orden de salida transitoria al establecimiento penitenciario donde este se encuentra recluido. Si la orden del tribunal no establece una fecha y hora para la salida, el Coordinador de Seguridad y Custodia es el encargado de establecerlas. (8)

Ejecución de Salida Transitoria

El proceso de ejecución de Salida Transitoria se inicia llegada la fecha y hora en que debe ser ejecutada la salida transitoria planificada previamente. En este momento, el Coordinador de Seguridad y Custodia determina los custodios que se responsabilizarán con la tarea de

acompañar al interno al lugar a donde debe ser trasladado, y luego regresarlo al establecimiento penitenciario. (8)

En caso de tratarse de un traslado al hospital por urgencia médica, no se parte de una planificación previa, se ejecuta dicha salida cuando el médico del establecimiento la considere imprescindible para la salud del interno y el director lo apruebe.

Tanto en prisiones Venezuela como en prisiones Cuba se realizan salidas a hospitales, al tribunal y otros lugares con regreso al centro penitenciario. También en ambos se recogen atributos similares como la hora y fecha en que se ejecutan los procesos. Las salidas por urgencia no tienen planificación previa y los procesos se inician a partir de una decisión.

A pesar de estas similitudes este sistema no se puede utilizar porque en él no se realiza un control del plan de seguridad referente a la misma.

1.2.2. Sistema Automatizado para el Control del Recluso

En el actual sistema informático vigente en los centros penitenciarios cubanos, el Sistema Automatizado para el Control del Recluso (SACORE) se registran solamente los conduces al tribunal, pero no se le realiza ningún tipo de planificación después de ser registrado como es llevar el control del plan de seguridad del conduce, no se registran los conduces por urgencia; los conduces de mayor severidad y de extranjeros no quedan registrados en la aplicación. Tampoco se tiene en cuenta las solicitudes y decisiones que avalan dichas solicitudes para ejecutar los conduces.

1.3. Metodología, Herramientas y Tecnologías a utilizar

A continuación se analizan la metodología, herramientas y tecnologías utilizadas durante la solución del problema planteado:

1.3.1. Metodología

Como metodología de referencia para guiar el trabajo de implementación, se utilizará Rational Unified Process (RUP). RUP es una metodología de desarrollo de software, un conjunto de procesos adaptables al contexto y necesidades de cada organización. Se caracteriza por estar dirigido por casos de uso que son los que definen los requisitos obtenidos con la modelación del negocio y la captura de requisitos y a partir de las necesidades identificadas por el usuario, en cada fase de desarrollo el equipo pasa por todos los flujos de trabajo, refinando el trabajo

realizado en las iteraciones anteriores y añadiendo nuevos elementos y enriqueciendo el producto final cada vez más. (12)

1.3.2. Herramientas de modelado

Visual Paradigm for UML 6.4 Enterprise Edition: Es una suite completa de herramientas de Ingeniería de Software Asistida por Computadora (CASE), para realizar el modelado de los procesos a desarrollar. Permite la generación de todos los diagramas generados por la metodología RUP en sus diferentes fases. Visual Paradigm se basa en el lenguaje de modelado Unified Modeling Language (UML). Permite elaborar todos los tipos de diagramas de clases. (13)El Visual Paradigm fue utilizado para modelar los diagramas de clases del diseño, de interacción, de componentes y de despliegue para una mejor comprensión del diseño del módulo Conducción.

ER/Studio 8.0: Es una herramienta de modelado de datos para el diseño y construcción de bases de datos a nivel físico y lógico. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. (16)

1.3.3. Herramientas de desarrollo

NetBeans 7.0: El NetBeans IDE es un programa de código abierto escrito completamente en Java. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java. Cuenta con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (19)

Apache Tomcat 6.0: 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. (14)

TortoiseSVN 1.6.14: Es un software de sistema de control de versiones. Está desarrollado sobre software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. El acceso al repositorio es mediante la red, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Su principal importancia radica en que varias personas pueden modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomentando así la colaboración. (17)

Oracle Database 11g Release 2: Es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Oracle es portable porque se puede instalar en la gran mayoría de los sistemas operativos y también tiene una gran capacidad de almacenamiento. Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia. Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente. Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos. Tiene buen rendimiento y hace buen uso de los recursos. Es un sistema multiplataforma, disponible en Windows, Linux y Unix. Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. (15)

1.3.4. Tecnologías

Java Platform Enterprise Edition (J2EE): 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), J2EE añade las capacidades que proporcionan una plataforma completa para el desarrollo sin contratiempos, que es a la vez estable, segura y rápida. J2EE no es solo una plataforma o una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones. Ofrece muy buenas perspectivas para la implementación de software empresarial para aquellos sistemas informáticos que requieran basar su arquitectura en productos basados en software libre. (19)

Grails 1.3.7: Es nuevo framework para la plataforma Java que se basa en el lenguaje dinámico Groovy. Grails se ha desarrollado con una serie de objetivos en mente:

- Ofrecer un framework web de alta productividad para la plataforma Java.
- Ofrecer un framework consistente que reduzca la confusión y que sea fácil de aprender.
- Ofrecer documentación para las partes del framework relevantes para sus usuarios.
- Proporcionar lo que los usuarios necesitan en áreas que a menudo son complejas e inconsistentes:
    - Framework de persistencia potente y consistente.

- Patrones de visualización potentes y fáciles de usar con GSP (Groovy Server Pages).
- Bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.
- Buen soporte de Ajax que sea fácil de extender y personalizar.(18)

Dojo Toolkit 1.5: Es una herramienta de código abierto JavaScript (Open Source) para desarrollar interfaces de aplicaciones Web Dinámicas. Está compuesto por Widgets que son componentes de código en JavaScript pre-empaquetados que puede ser utilizados para enriquecer sitios web con varias características que trabajan a través de la mayoría de los navegadores, tales como: Menús, Tabs, Tooltips y Tablas ordenables. También es posible utilizar temas para mejorar la apariencia de la aplicación como Nihilo, Soria y Tundra. (3)

1.3.5. Lenguajes

UML: Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar. UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. (4) A continuación se muestran los diagramas de UML que se obtendrán durante la presente investigación:

- Diagrama de clases.
- Diagrama de interacción.
- Diagrama de componentes.
- Diagrama de despliegue.

Java: Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. El desarrollo

de aplicaciones se apoya en un gran número de clases preexistentes. Al ser un lenguaje puro Orientado a Objetos le otorga gran reusabilidad y la independencia de la plataforma permite que los programas escritos en el lenguaje Java puedan ejecutarse igualmente en cualquier tipo de hardware, lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo.(21)

Groovy 1.7.8: Es un lenguaje de programación orientado a objetos que emplea el framework Grails. Groovy usa una sintaxis muy parecida a Java. Además, la mayoría del código Java es sintácticamente válido en Groovy. (18)

HTML: Es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto, es decir, un lenguaje que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento. Utiliza marcas para describir la forma en la que deberían aparecer el texto y los gráficos en un Navegador web que, a su vez, están preparados para leer esas marcas y mostrar la información en un formato estándar. Los navegadores se encargan de interpretar el código HTML de los documentos, y de mostrar a los usuarios las páginas web resultantes del código interpretado. (23)

JavaScript: Lenguaje script utilizado para unir el conjunto de tecnologías usados en la web. JavaScript es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Una de las grandes aportaciones de JavaScript a la creación de interfaces web es la posibilidad de acceder al contenido de los campos de los formularios para realizar acciones sobre los valores introducidos por el usuario, modificarlos y validarlos. (5) Maneja objetos dentro de nuestra página Web y sobre ese objeto podemos definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas. Es dinámico, responde a eventos en tiempo real como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. (22)

1.4. Propuesta de solución

Para resolver la problemática planteada se realizó la siguiente propuesta de solución:

En el sistema existen dos formas diferentes para registrar los conduces: a partir de una solicitud de conducción que luego se convierte en una decisión y registrando el conduce sin previo aviso.

Ambas formas convergen en la planificación del conduce que consiste en asignar los funcionarios, el transporte y se confecciona el plan de seguridad referente al mismo. Una vez transcurrida la fecha del conduce se registra la ejecución que provoca un cambio en el estado del conduce el cual pude ser planificado, ejecutado y cancelado en caso que no se realice el conduce (Ver figura 1).

La propuesta solución diseñada por Prisiones Cuba para el módulo Conducción brinda algunas funcionalidades que no brindan los dos sistemas anteriormente analizados como: que permite asignar vehículos al conduce; también asignar funcionarios al conduce; gestionar un plan de seguridad referente al proceso.

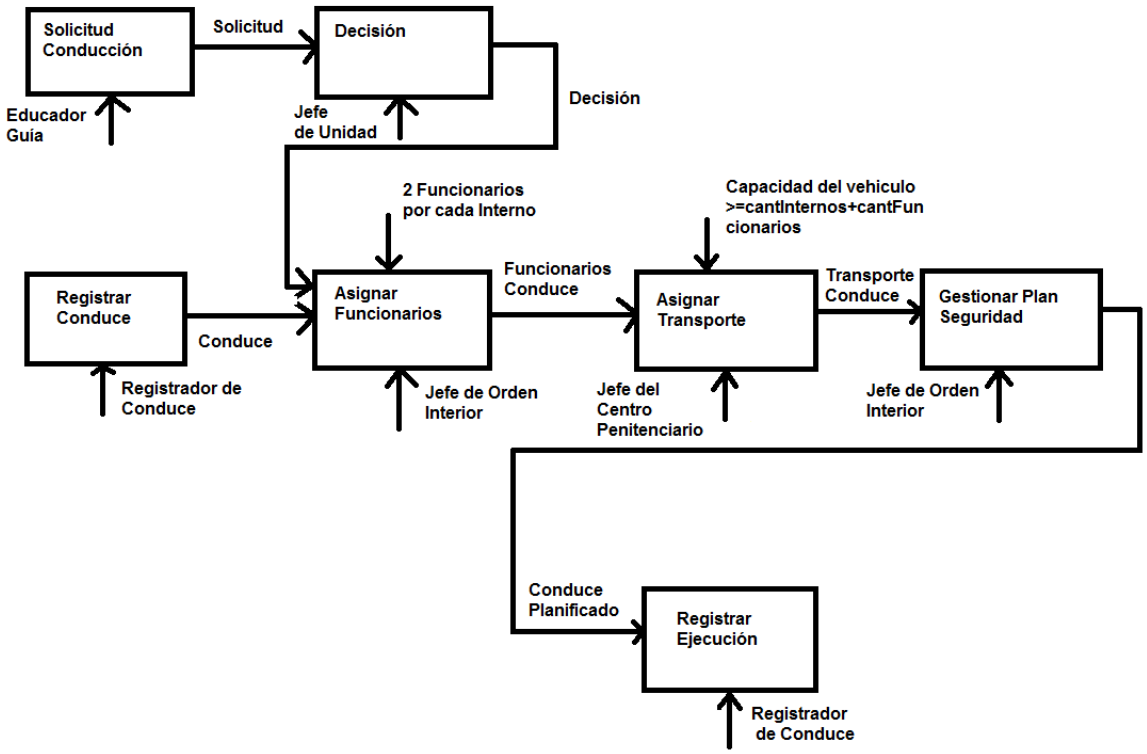

Figura 1: Propuesta solución.

1.5. Conclusiones parciales

Partiendo de un análisis de los sistemas informáticos que gestionan los procesos penitenciarios cubano y venezolano se arribó a la conclusión de que no ofrecen las funcionalidades requeridas para llevar a cabo los procesos de conducción. A raíz de esto surge la necesidad de diseñar e

implementar un módulo del SIDEP capaz de gestionar la ejecución y planificación de los procesos de conducción del Sistema Penitenciario Cubano. Se realizó un análisis de la metodología RUP, las herramientas de modelado Visual Paradigm y ER/Studio, como herramientas de desarrollo se utilizó el entorno de desarrollo integrado(IDE) NetBeans, como servidor web Apache Tomcat, TortoiseSVN para el control de versiones, como gestor de bases de datos Oracle Database, como plataforma Java 2 Enterprise Edition, fue empleado el marco de trabajo para el desarrollo de aplicaciones web Grails y como complemento Dojo para la interfaz de la aplicación. El lenguaje de modelado empleado fue UML, los lenguajes por parte del servidor fueron Java y Groovy; por parte del cliente HTML y JavaScript para validar los campos de las vistas.

Capítulo II: Diseño del sistema

2.1. Introducción

En este capítulo se explicará la arquitectura que propone Grails como framework para el desarrollo de aplicaciones web y se describirán los requisitos definidos para el módulo Conducción de manera que ambas cuestiones permitan el diseño del módulo. Se presentan los diagramas de clases del diseño haciendo mención a los patrones de diseño utilizados y los diagramas de interacción, así como el diseño de la base de datos para la persistencia de la información.

2.2. Descripción de las funcionalidades del módulo Conducción

A continuación se muestran los requisitos capturados por el equipo de analistas del proyecto:

Requisitos	Nombre	Descripción
R 1	Registrar los datos del conduce	Permite registrar un conduce según el tipo de motivo de conduce seleccionado.(Anexo 4, tabla 31)
R 2	Actualizar conduce	Permite actualizar los datos y recursos de conduces que fueron previamente registrados. (Anexo 4, tabla 32)
R 3	Asignar funcionarios al conduce	Permite asignar los funcionarios que se destinarán para conducir los internos.(Anexo 4, tabla 33)
R 4	Buscar conduce	Permite buscar los conduces que pudiera tener el acusado, sancionado o asegurado en el sistema.(Anexo4, tabla 34)
R 5	Registrar ejecución de conduce	Permite registrar si el conduce fue ejecutado o no y los datos de por qué no se ejecutó en ese caso.(Anexo 4, tabla 35)
R 6	Consultar conducciones	Permite consultar las conducciones según el criterio seleccionado.(Anexo 4, tabla

		36)
R 7	Asignar transporte al conduce	Permite asignarle vehículos al conduce.(Anexo 4, tabla 37)
R 8	CRUD-D[1] Plan de seguridad	Permite consultar o actualizar el Plan de Seguridad del conduce.(Anexo 4, tabla 38)
R 9	Registrar Conduce de urgencia	Permite registrar los datos de un conduce por urgencia.(Anexo 4, tabla 39)

Tabla 1: Descripción de los casos de uso.

2.3. Arquitectura del Sistema

La arquitectura definida para el SIDEP está basada principalmente en las funcionalidades y facilidades que propone el framework de desarrollo Grails.

Grails es un framework de desarrollo para aplicaciones web, dentro de la plataforma Java, que utiliza el lenguaje de programación Groovy; su arquitectura está basada en tres capas principales como se muestra en la figura 2:

- La capa web, la capa de datos y la capa de servicios. La capa de web se compone de dos partes principales: vistas y controladores.
  - Las vistas son responsables de hacer la interfaz de usuario y se implementan mediante GSP, que son una extensión de JSP y puede incluir código Groovy.
  - Los Controladores manejan y coordinan la aplicación mediante la recepción de las acciones del usuario desde la vista y actúan sobre ellos, por ejemplo, al interactuar directamente con el modelo de dominio, delegando acciones a un controlador diferente o una capa diferente, o re direccionando a una vista diferente.
- La capa de datos o modelo está compuesto por las clases del dominio o entidades; estas clases son utilizadas por los servicios para hacer las consultas y obtener la información que el usuario solicita o el sistema necesita. Las clases del dominio son entidades fundamentales para el sistema, ya que en ellas persisten los datos con los que posteriormente el cliente va a interactuar, modificar o insertar.
- La capa de servicios es la responsable de implementar la lógica de negocio de la aplicación.(2)

[1] CRUD-D: patrón de CU, CRUD parcial sin la funcionalidad de eliminar (delete).

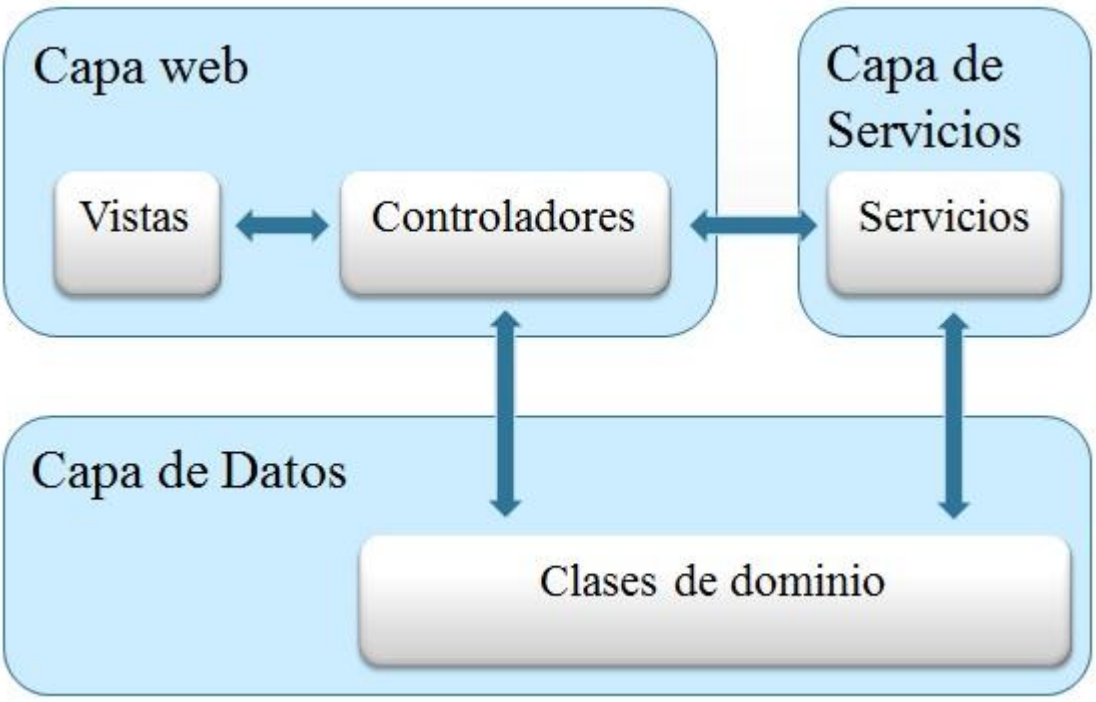

Figura 2: Arquitectura de una aplicación Grails.

Grails como framework para el desarrollo de aplicaciones web implementa la arquitectura cliente-servidor (Ver figura 3); esta es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta.

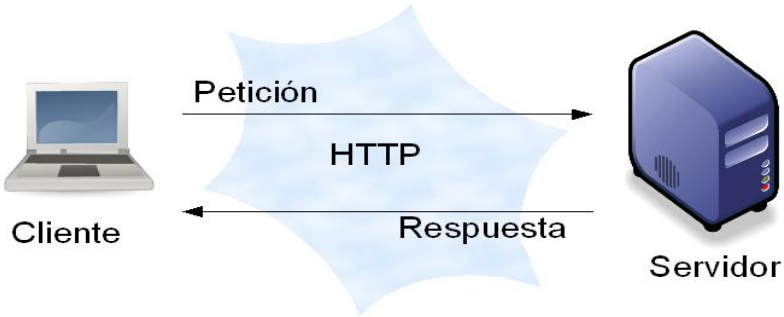

Figura 3: Arquitectura cliente-servidor.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debido a la centralización de la

gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

2.4. Patrones de diseño

Los Patrones de Diseño son modelos de trabajo enfocados a dividir un problema en partes de modo que sea posible abordar cada una de ellas por separado para simplificar una solución. Un patrón de diseño es un conjunto de reglas que describen como afrontar tareas y solucionar problemas que surgen durante el desarrollo de software. (4)

Modelo Vista Controlador (MVC): El patrón MVC separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos el modelo, la vista y el controlador. Grails estructura las aplicaciones utilizando el patrón MVC, pero le agrega otra capa a este patrón la capa de servicios. La capa de servicios evita que se coloque demasiado código en la capa de Control facilitando la fase de mantenimiento de la aplicación.

El siguiente diagrama muestra el flujo que normalmente tiene un caso de uso en Grails:

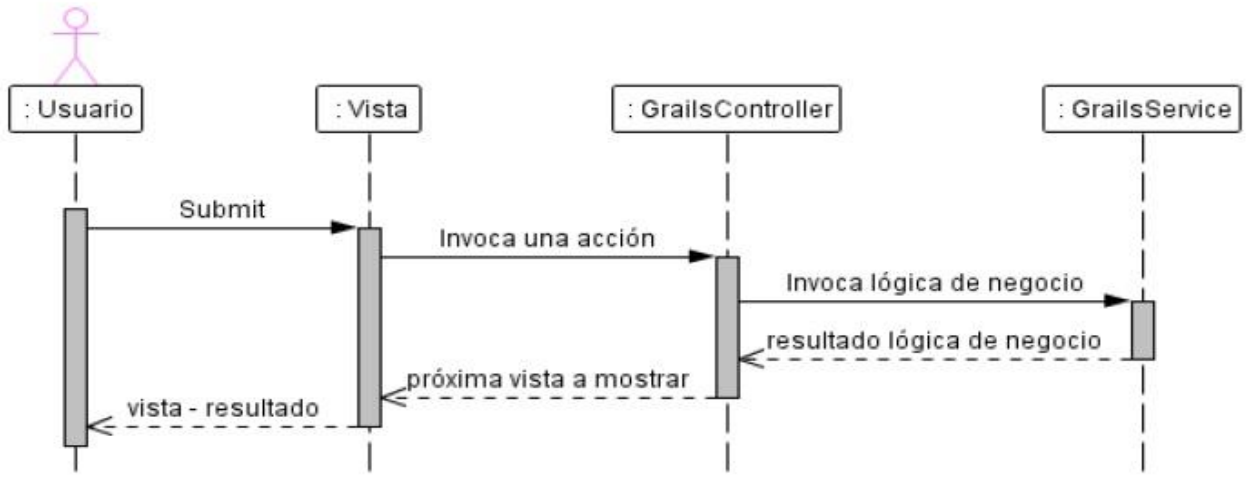

Figura 4: Patrón modelo vista controlador. (2)

1. El usuario pulsa un link en la vista, o un botón *Submit* de un formulario.

2. La solicitud llega a la acción correspondiente del controlador; según lo configurado en el mapeo de URLs.
3. El controlador invoca al servicio encargado de la implementación del caso de uso.
4. En base al resultado de la invocación, el controlador decide cual es la próxima vista a mostrar, y solicita a la capa de presentación que se la muestre al usuario. (2)

Experto: Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad. Este patrón se pone de manifiesto en la clase de dominio conduceJuicio la cual es experta en almacenar información de los conduces a juicio. (4)

Creador: Este patrón plantea asignar a la clase B la responsabilidad de crear una instancia de la clase A si se cumple una de las siguientes condiciones:

1. B contiene A.
2. B agrega A.
3. B tiene los datos de inicialización de A.
4. B registra A.
5. B utiliza A muy de cerca.(4)

Este patrón se pone en la clase de dominio *Circular115* que instancia un objeto de tipo *DecConCir115* donde en la primera clase se registran los datos de la segunda. En la clase Conduce se crea una instancia de un objeto de tipo *PlSeg* (Plan de Seguridad). También se evidencia en las clases de dominio que utilizan nomencladores como Conduce que utiliza los nomencladores *NomMotivoConduce* y *NomEstadoConduce*.

Alta Cohesión: Este patrón plantea que la información que almacena una clase debe ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. (4) Este patrón se evidencia en la clase *Conduce*, donde la información que almacena se encuentra relacionada con esta.

Bajo Acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (4) Este patrón se manifiesta en las relaciones entre clases que

existen el módulo conducción donde las mismas se encuentran relacionadas lo mínimo posible entre ellas.

Inversión de Control (IoC): Es otro patrón utilizado en Grails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que este solo contenga la lógica necesaria para hacer su trabajo.(2) Este se pone de manifiesto en el *controlador Circular115Controller* donde se define una variable con el nombre del *servicio* que necesita emplear (*def circular115Service*), pero no se ocupa de instanciar el *servicio* ni de configurarlo de ningún modo antes de poder usarlo.

Singleton: Grails controla el ciclo de vida de los *servicios*, decidiendo cuando se crean instancias de los mismos. Por defecto todos los *servicios*: solo existe una instancia de la clase que inyecta en todos los artefactos que declaren la variable correspondiente. (2)

Controller: El patrón Controller propone asignar la responsabilidad de controlar el flujo de eventos de un sistema, a clases específicas llamadas controladores. Los controladores no ejecutan las tareas sino que las delegan en otras clases, con las que mantiene un modelo de alta cohesión. Este patrón se pone de manifiesto en las clases controladoras del módulo Conducción que delegan las tareas a los servicios que es donde se implementa la lógica del negocio. (4)

2.5. Descripción de las clases

En la siguiente tabla se describen las principales clases del dominio, vistas, JavaScript, controladores y servicios:

Entidad	Tipo	Descripción
Conduce	Clase de Dominio	Almacena todos los datos referentes al conduce por el que pueda transcurrir un interno figura 4.
Circular115	Clase de Dominio	Almacena todos los datos referentes al conduce por Circular115 figura 5.
ConduceJuicio	Clase de Dominio	Almacena todos los datos referentes al conduce realizado al Tribunal figura 6.

OtrosConduces	Clase de Dominio	Almacena los datos referentes a otros conduces por el que pueda transcurrir un interno figura 7.
conduceTribunalCJ	Vista	Muestra los campos referentes al conduce a juicio.
circular115	Vista	Muestra los campos referentes al conduce por circular115.
Lugar	Vista	Muestra los campos referentes a la clase de dominio a OtrosConduces.
ConduceTribunalCJ	JavaScript	Valida los campos de la vista conduceTribunalCJ.
Circular	JavaScript	Valida los campos de la vista circular115.
Lugar	JavaScript	Valida los campos de la vista lugar.
ConduceTribunalController	Controlador	Procesa los datos del conduce a Juicio.
Circular115Controller	Controlador	Procesa los datos del conduce por circular115.
OtrosConducesController	Controlador	Procesa los datos de otros conduces.
ConduceJuicioService	Servicio	Se implementa el método salvar (); el cual salva objetos de tipo ConduceJuicio.
Circular115Service	Servicio	Se implementa el método salvar (); el cual salva objetos de tipo Circular115.
OtrosConduceService	Servicio	Se implementa el método salvar (); el cual salva objetos de tipo OtrosConduces.

Tabla 2: Descripción de las principales clases del módulo.

2.5.1. Diagrama de clases del diseño con estereotipos web

En la ingeniería de software, un diagrama de clases en UML es un tipo de diagrama de estructura estática que describe la estructura de un sistema por el que muestra el sistema de clases, sus atributos, las operaciones (o métodos), y las relaciones entre las clases. (5) A continuación se muestra el diagrama de clases del diseño con estereotipos web de los casos de uso Registrar conduce y Buscar conduce:

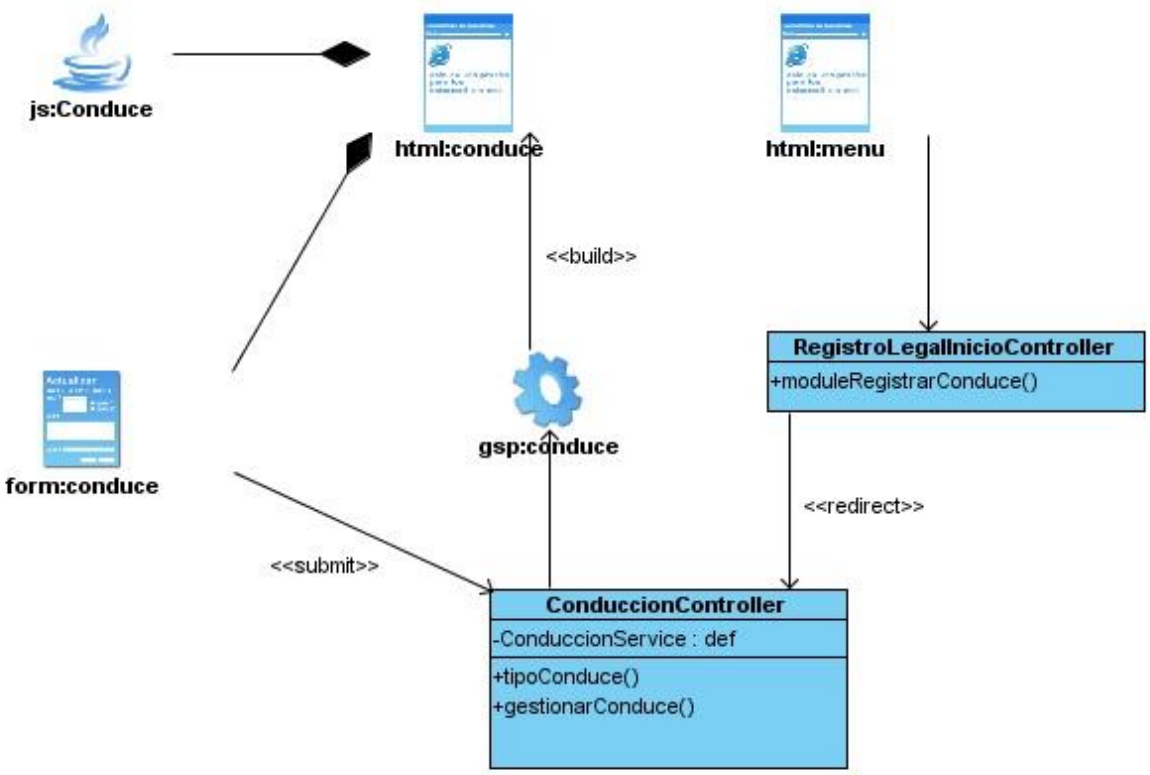

Figura 5: Capa web del diagrama de clases del diseño del caso de uso Registrar Conduce.

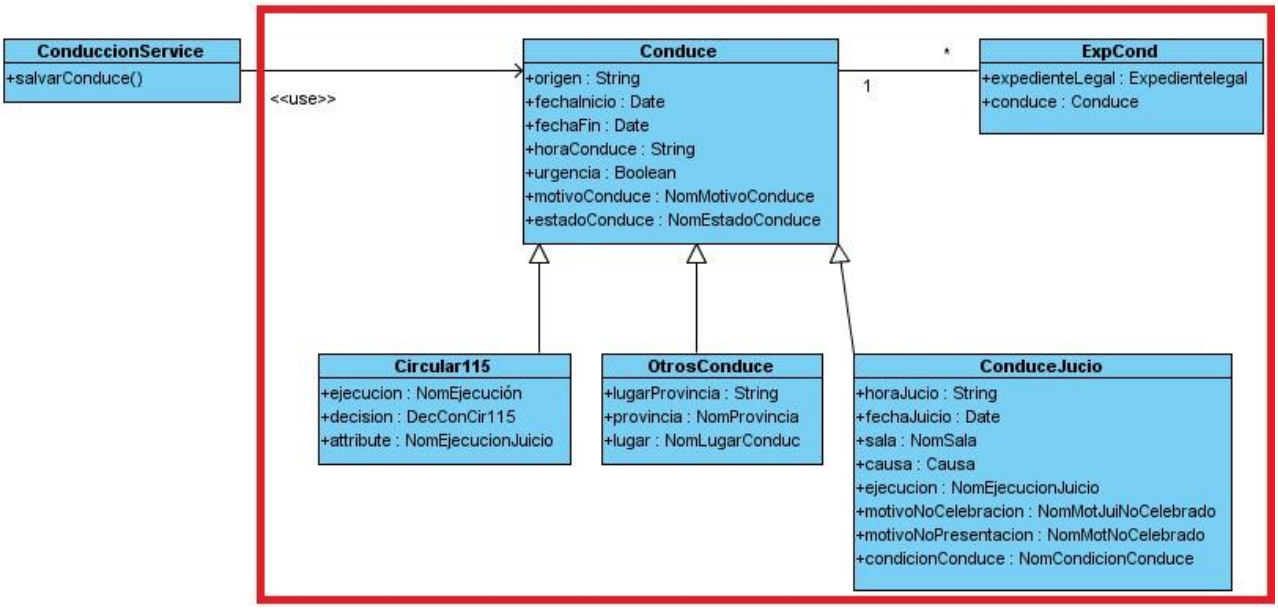

Figura 6: Capa de Datos del diagrama de clases del diseño del caso de uso Registrar Conduce enmarcada en rojo.

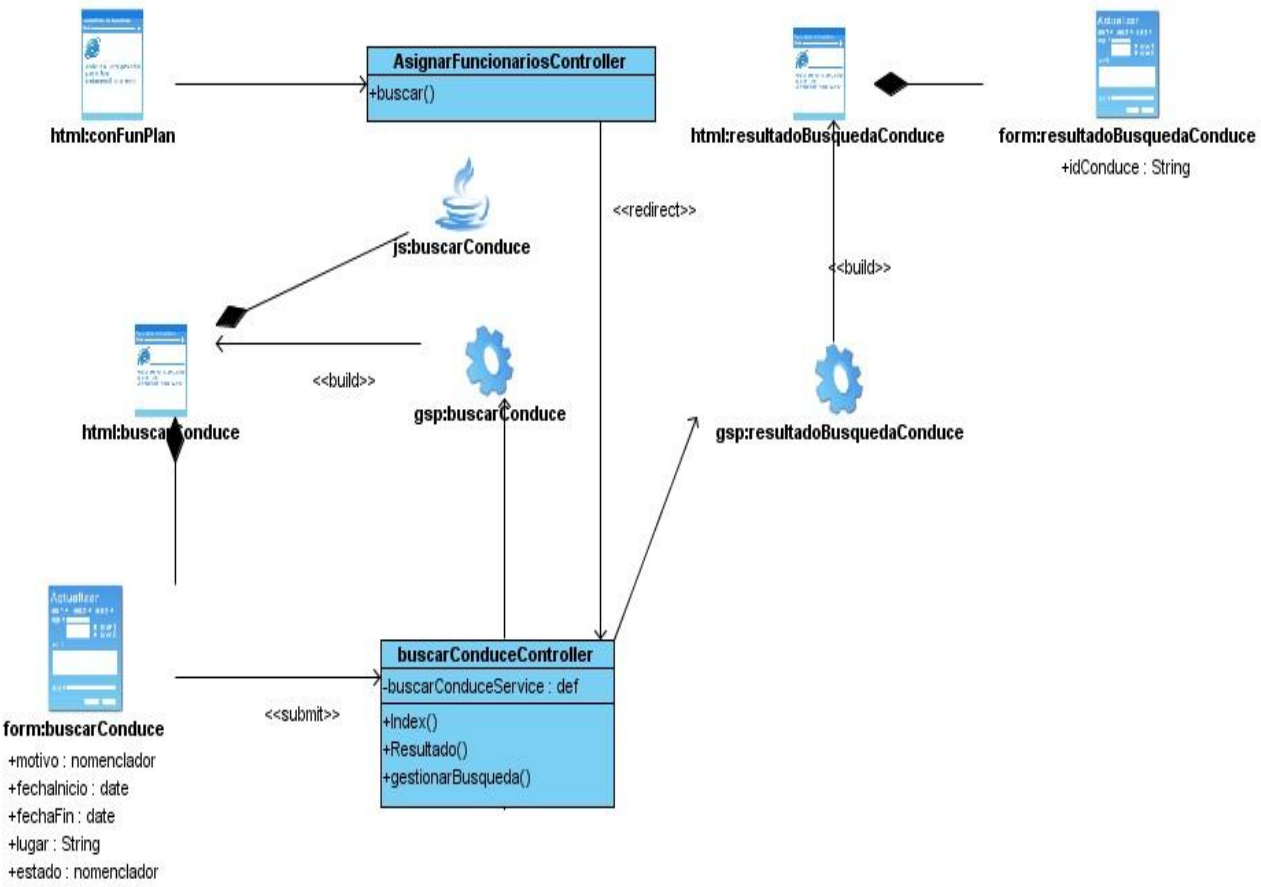

Figura 7: Capa web del diagrama de clases del diseño del caso de uso Buscar conduce.

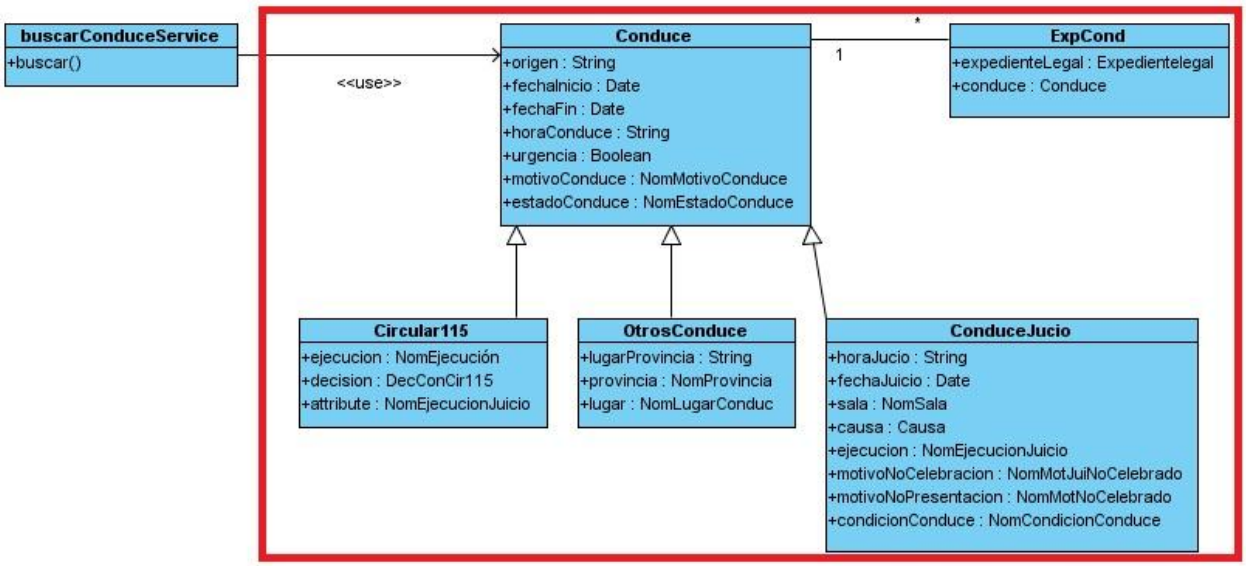

Figura 8: Capa de Datos del diagrama de clases del diseño del caso de uso Buscar Conduce enmarcada en rojo.

El resto de los diagramas de clases diseño se encuentran en el Anexo 1.

2.5.2. Diagrama de Interacción

Un diagrama de interacción es un diagrama que muestra el comportamiento dinámico de un sistema, enfocándose en las relaciones entre objetos. El diseño orientado a objetos tiene por objeto definir las especificaciones lógicas del software que cumplan con los requisitos funcionales, basándose en la descomposición por clases de objetos. Un paso esencial de esta fase es la asignación de responsabilidades entre los objetos y mostrar cómo interactúan a través de mensajes, expresados en diagramas de interacción. Estos presentan el flujo de mensajes entre las instancias y la invocación de métodos. (4) Para la presente investigación se presentarán los diagramas de colaboración de los casos de uso Registrar conduce y Buscar conduce:

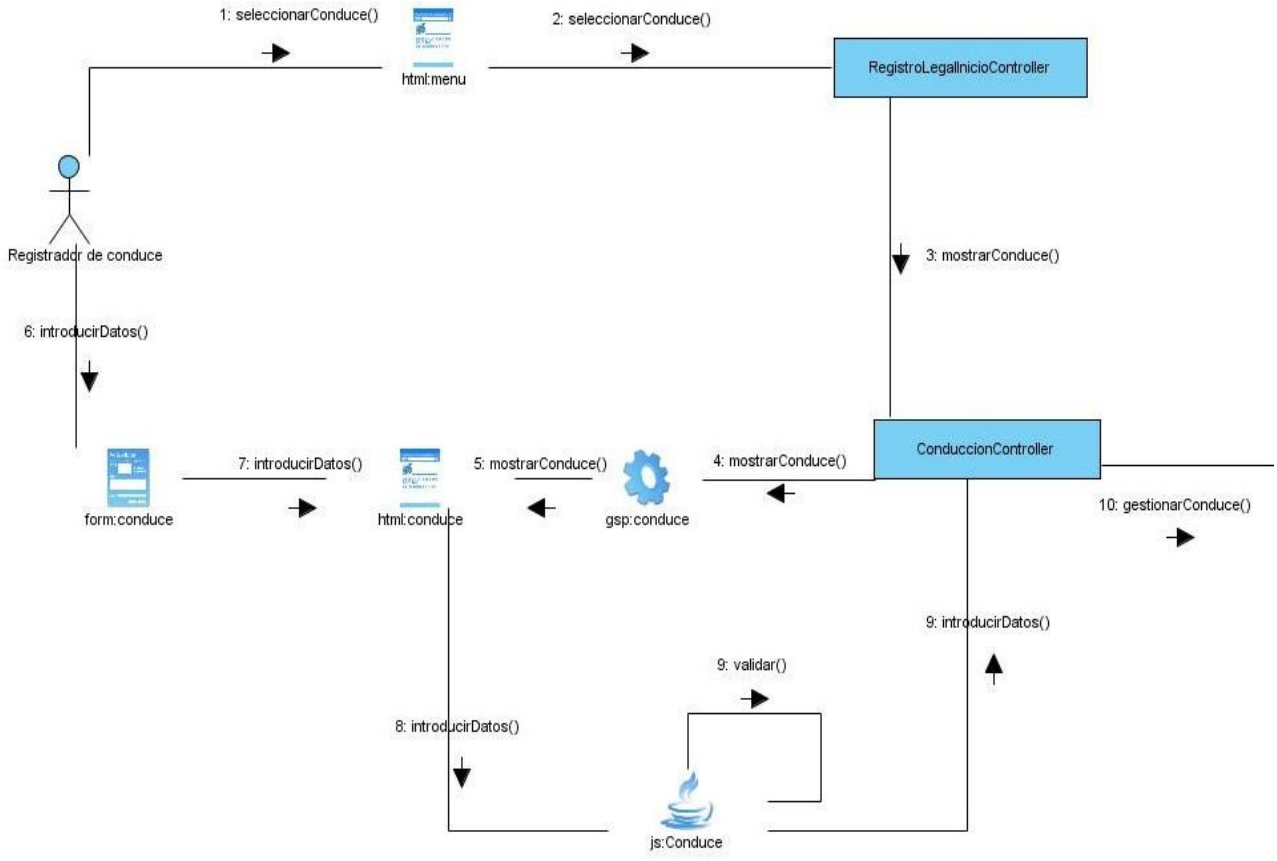

Figura 9: Capa web del diagrama de interacción Registrar Conduce.

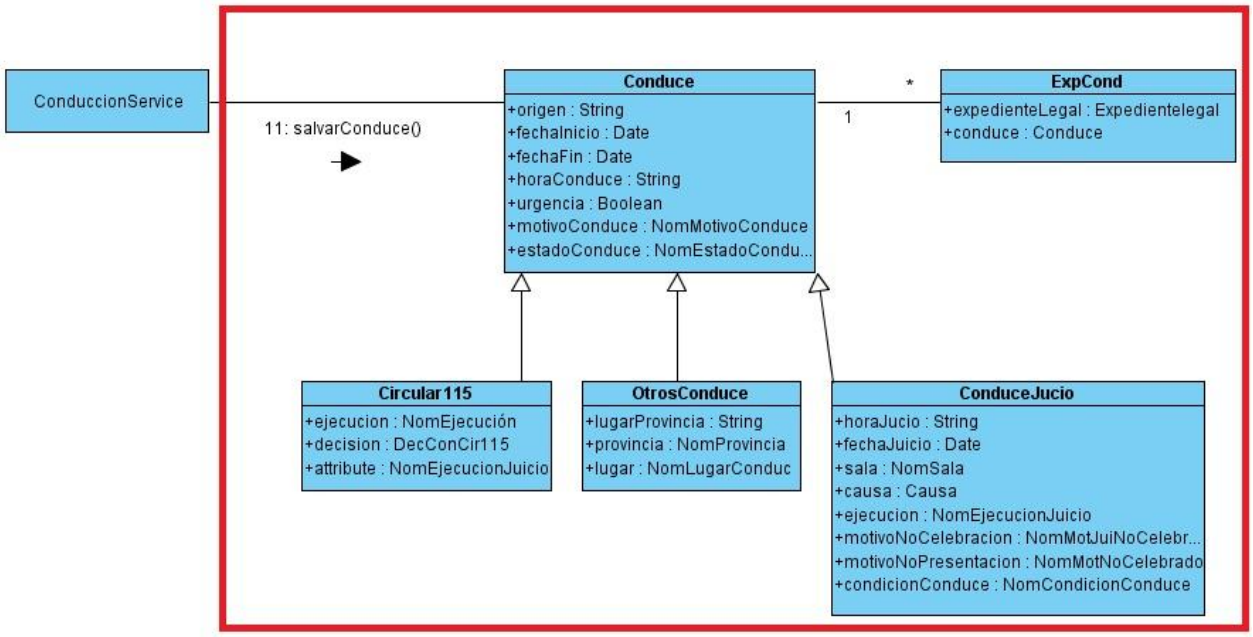

Figura 10: Capa de Datos del diagrama de interacción del caso de uso Registrar Conduce enmarcada en rojo.

En el diagrama de la figura 9 el flujo se inicia cuando el usuario selecciona en el menú principal la opción de registrar conduce esta acción hace que se muestre la página principal del módulo *registrarConduce* que brinda la posibilidad de seleccionar el tipo de conduce que se desea realizar y de registrar los datos según el tipo de conduce seleccionado; los datos son validados en el JavaScript *RegistrarConduce* y de ser correctos son enviados al controlador *ConduccionController* donde se procesan los datos y son enviados al servicio para ser guardados en la base de datos.

En la figura que se muestra a continuación se muestra el diagrama de interacción del caso de uso Buscar conduce.

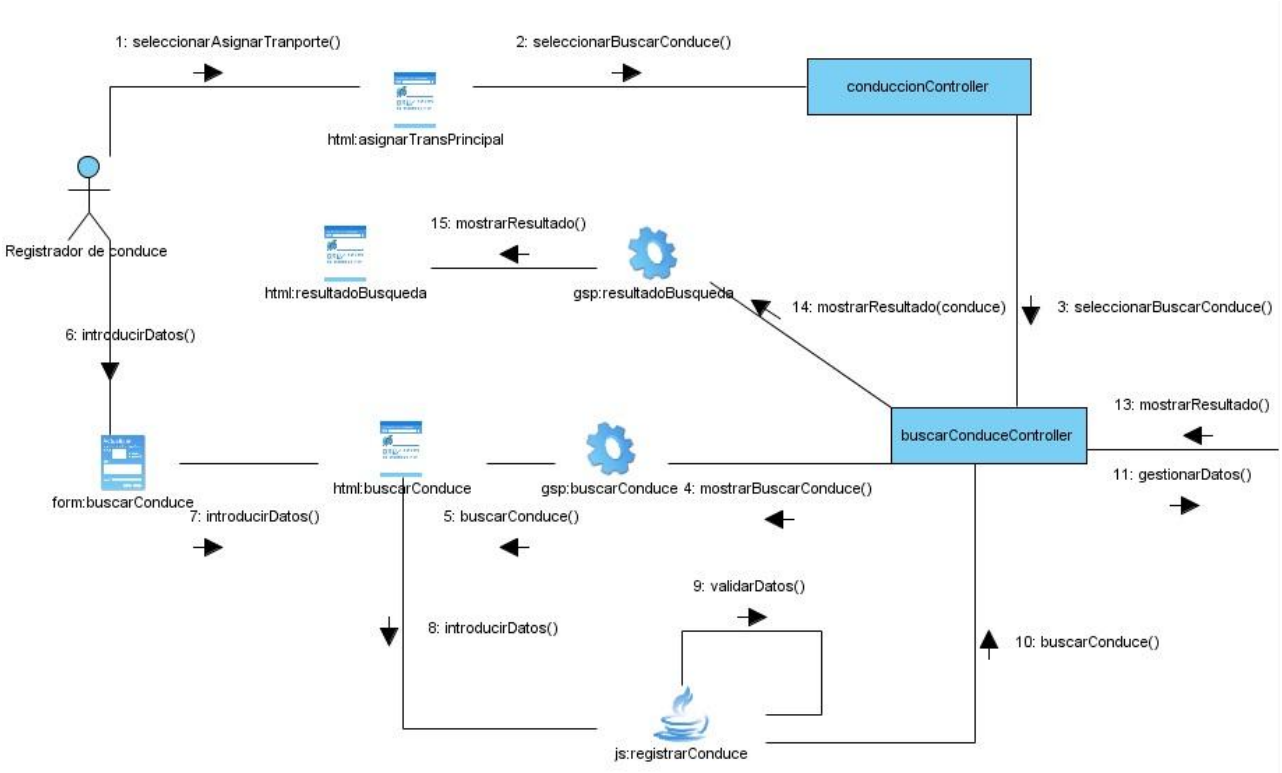

Figura 11 : Capa web del diagrama de interacción Buscar Conduce.

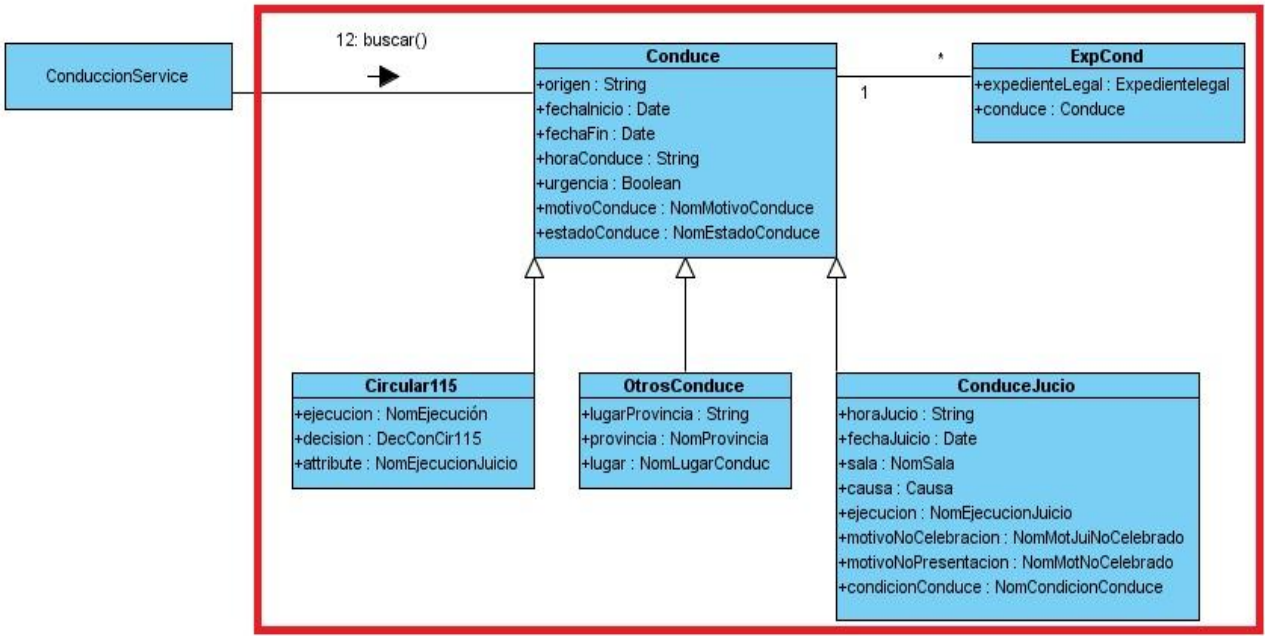

Figura 12: Capa de Datos del diagrama de interacción del caso de uso Buscar Conduce enmarcada en rojo.

En la figura 11 el flujo se inicia cuando el usuario selecciona el botón buscar en algunas de las vistas del módulo conducción que brinden esta funcionalidad. Esta acción es procesada por el controlador *BuscarConduceController* el cual muestra la vista *buscarConduce*; esta muestra los campos por los que se puede buscar un conduce, brindando la opción de buscar. Los datos introducidos en la vista son enviados al controlador buscarConduce el cual utilizando un criterio de búsqueda implementado en el servicio *buscarConduceService* realiza la búsqueda, el controlador muestra la vista *resultadoBusqueda* con los conduces encontrados.

Para ver el resto de los diagramas de iteración, ver el Anexo 2.

2.6. Diseño de la Base de datos

Una base de datos es un conjunto de información relacionada entre sí, referente a un tema o propósito en particular. El módulo Conducción cuenta con 18 tablas sin contar los nomencladores a continuación se muestran las principales relaciones:

La relación principal en el diseño de la base de datos es la relación de herencia que existe entre las entidades de tipo *ConduceJuicio*, *Circular115* y *OtrosConduces* con la entidad *Conduce;* debido que en la planificación del conduce la mayoría de las veces se interactúa con la información de estas entidades ya sea registrando, modificando o consultando datos en estas.

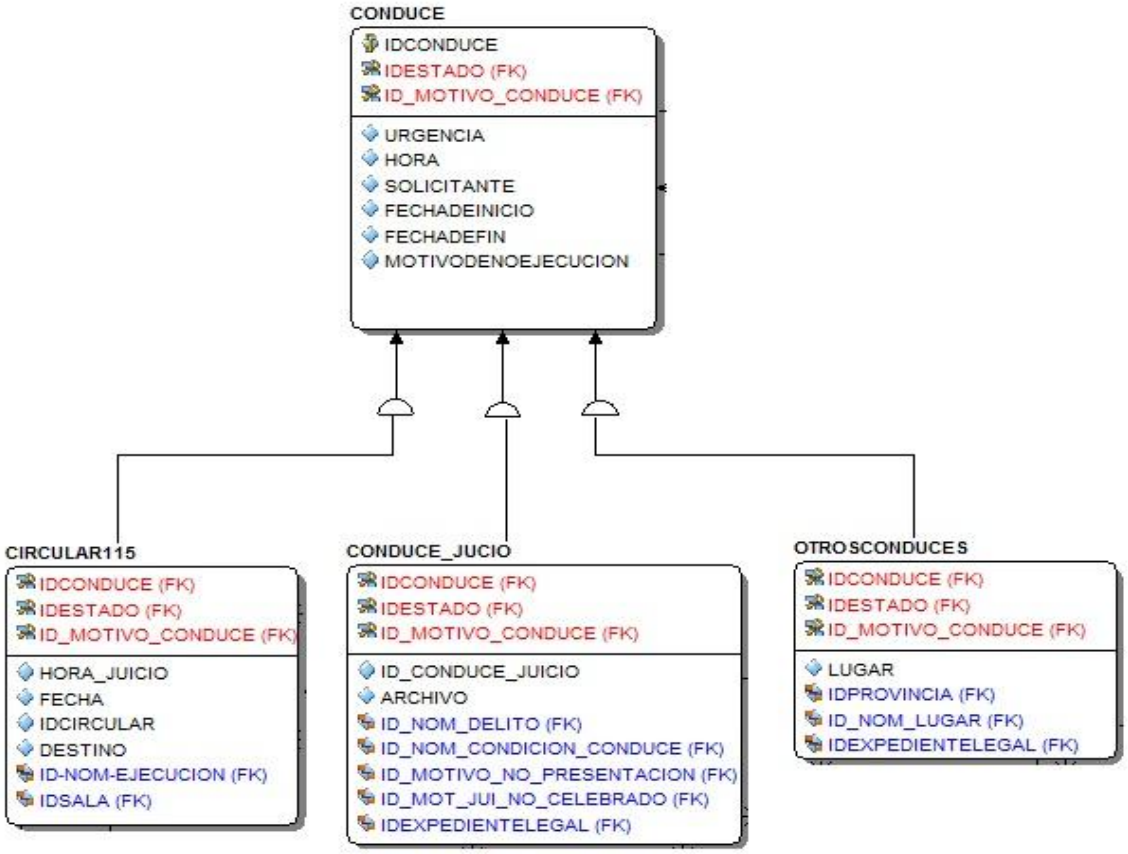

Figura 13 Relación de herencia.

También existe una relación mucho a mucho entre la clase *Conduce* y *VehicEspc* (Vehículo Especializado), *Conduce* y *Oficial*. Este tipo de relación genera una tabla nueva en la cual se encuentran las llaves de ambas tablas como se muestra en la figura 14 y 15.

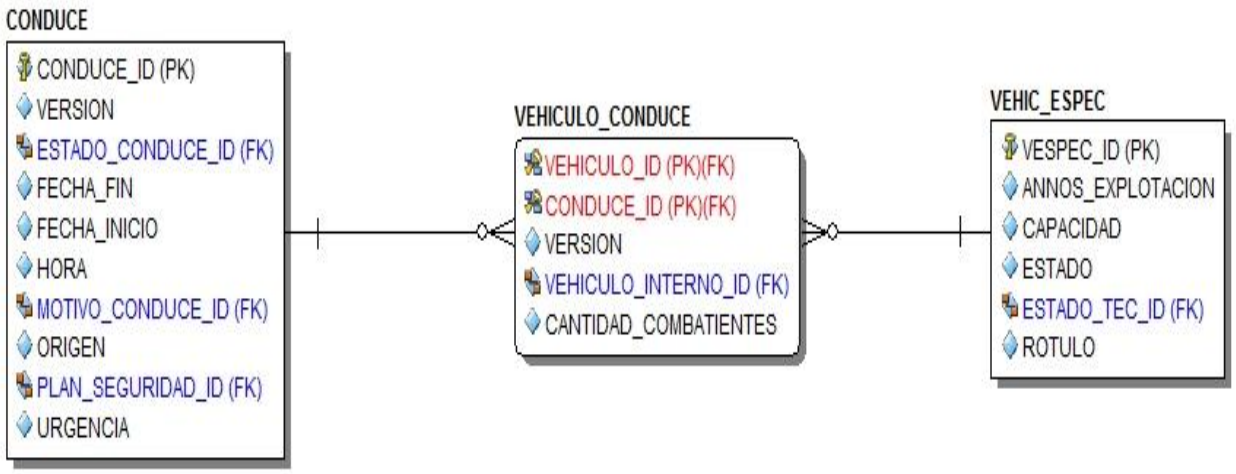

Figura 14 Relación Conduce Vehículo Especializado.

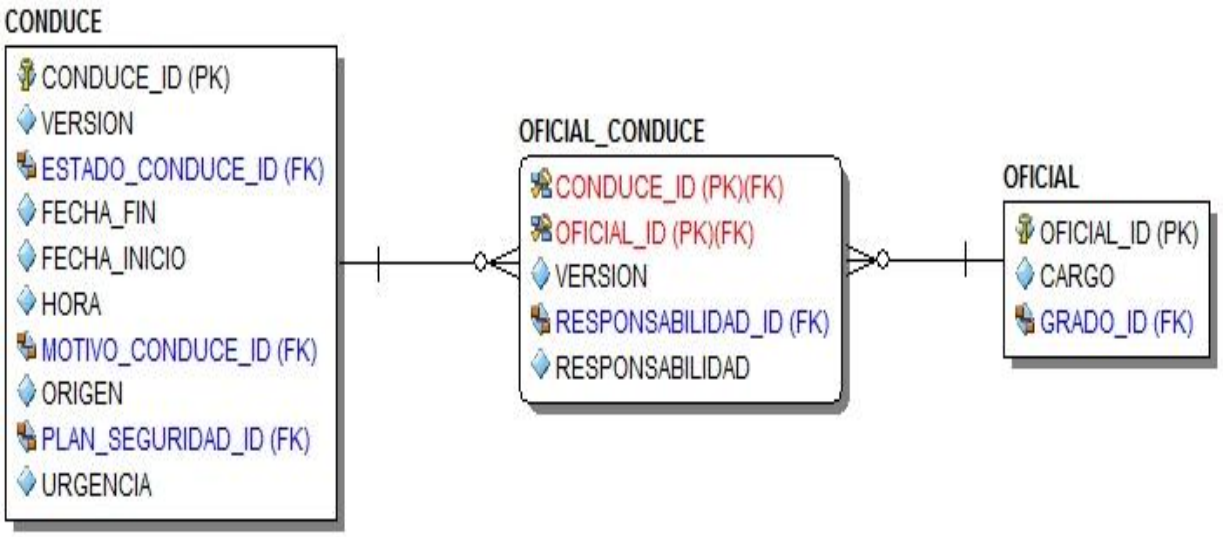

Figura 15 Relación Oficial Conduce.

En la figura 16 se muestra la relación entre la entidad Conduce y Pl_seg (plan de seguridad); en la figura 17 como se relaciona la entidad Conduce con la entidad Interno.

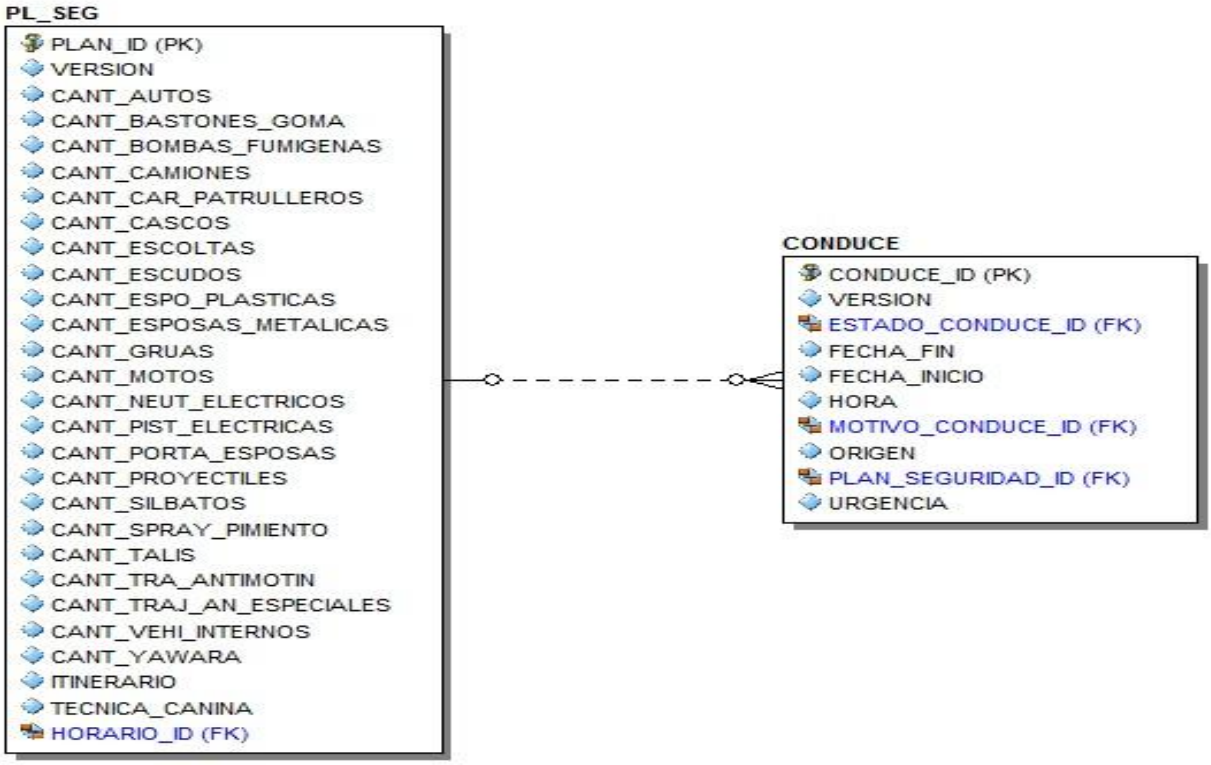

Figura 16 Relación entre el Conduce y el Plan de seguridad.

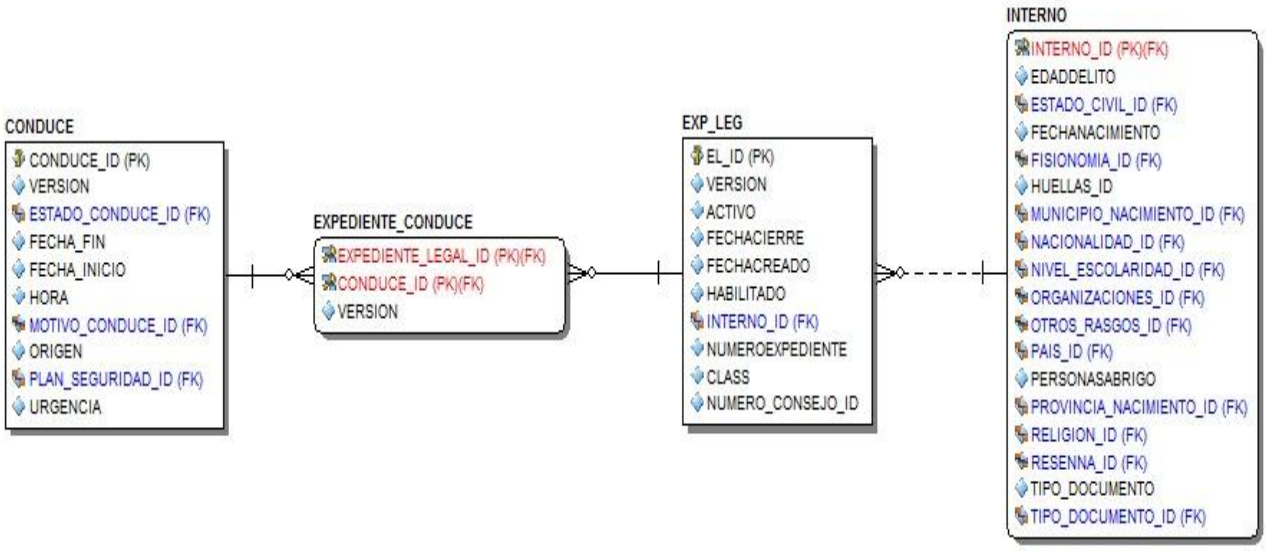

Figura 17 Relación entre el Conduce y el Interno.

En la tabla que se muestra a continuación se realiza una descripción de los atributos de las principales entidades del módulo Conducción:

Nombre de la entidad		Conduce
Descripción de la entidad		Almacena los datos del conduce.
Nombre del atributo	**Descripción**	**Tipo**
Origen o interés	Este campo guarda el nombre de la persona o centro que solicita el conduce del interno.	Cadena de caracteres
Motivo	Este campo guarda el motivo del conduce.	Cadena de caracteres
Fecha de inicio	Este campo guarda la fecha de inicio del conduce.	Fecha
Fecha de fin	Este campo guarda la fecha de fin del conduce.	fecha
Estado	Este campo guarda el estado en el que se encuentra el conduce.	Cadena de caracteres
Motivo de no ejecución	Este campo guarda el motivo de la no ejecución del conduce.	Cadena de caracteres
Hora	Este campo almacena la hora a la que se produce el conduce.	Número

Tabla 3: Descripción de los atributos de la entidad Conduce.

Nombre de la entidad		Conduce a tribunal
Descripción de la entidad		Almacena los datos de los conduces de presentación a juicio que no son celebrados.
Nombre del atributo	**Descripción**	**Tipo**
Motivo de no celebración	Este campo guarda el motivo de por qué el juicio no fue celebrado.	Cadena de caracteres
Condición	Este campo guarda la condición con que el interno es presentado a juicio.	Cadena de caracteres
Ejecución de juicio	Este campo almacena si se celebró el juicio o no.	Cadena de caracteres
Motivo de no presentación (sólo para cuando el conduce al Tribunal es cancelado)	Este campo almacena el motivo de no presentación a juicio de un interno.	Cadena de caracteres
Fecha de juicio	Este campo almacena la fecha del juicio.	Fecha
Hora de juicio	Este campo almacena la hora de juicio.	Número

Tabla 4 : Descripción de los atributos de la entidad Conduce a tribunal.

Nombre de la entidad		Otros conduces
Descripción de la entidad		Almacena los datos específicos de otros conduces que no son presentación a juicio.
Nombre del atributo	**Descripción**	**Tipo**

Lugar	Este campo guarda el lugar al que se va a realizar el conduce.	Cadena de caracteres
Provincia	Este campo guarda la provincia del lugar al que será conducido el interno.	Cadena de caracteres
Lugar en provincia	Este campo guarda el lugar en provincia donde irá el interno.	Cadena de caracteres

Tabla 5 : Descripción de los atributos de la entidad Otros conduces.

2.7. Conclusiones parciales

La utilización de los patrones de diseño utilizados como el Controller, Singleton entre otros, ayudó al desarrollo de la solución propuesta y proporcionó elementos reusables en el diseño del sistema de software.

A través de la realización del modelo de diseño, se logró agilizar el proceso de desarrollo, generando solo la documentación necesaria que sirviera de guía para la construcción del software.

Se desarrollaron varios artefactos como: modelo de diseño, modelo de datos y modelo de implementación, quedando la aplicación lista para entrar al flujo de pruebas.

Capítulo III: Implementación y prueba

3.1. Introducción

En este capítulo se abordará todo lo relacionado con la implementación del módulo Conducción de acuerdo a su diseño. Se muestran los diagramas correspondientes al flujo de trabajo de implementación del módulo Conducción, específicamente los diagramas de despliegue y componentes. Además, se define la estrategia de prueba a aplicar y se muestran los resultados.

3.2. Implementación

En esta fase de RUP se describe cómo los elementos de diseño se implementan en componentes, formando el modelo de implementación. Este modelo es considerado el artefacto más significativo dentro del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes y el diagrama de despliegue. El resultado final de esta fase es un sistema ejecutable. (8)

3.2.1. Diagrama de componentes

Un diagrama de componentes es un diagrama UML que representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables, o paquetes. El diagrama de componente muestra la organización y las dependencias entre un conjunto de componentes. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. (4)

El siguiente diagrama muestra la relación del módulo Conducción con los demás componentes del sistema:

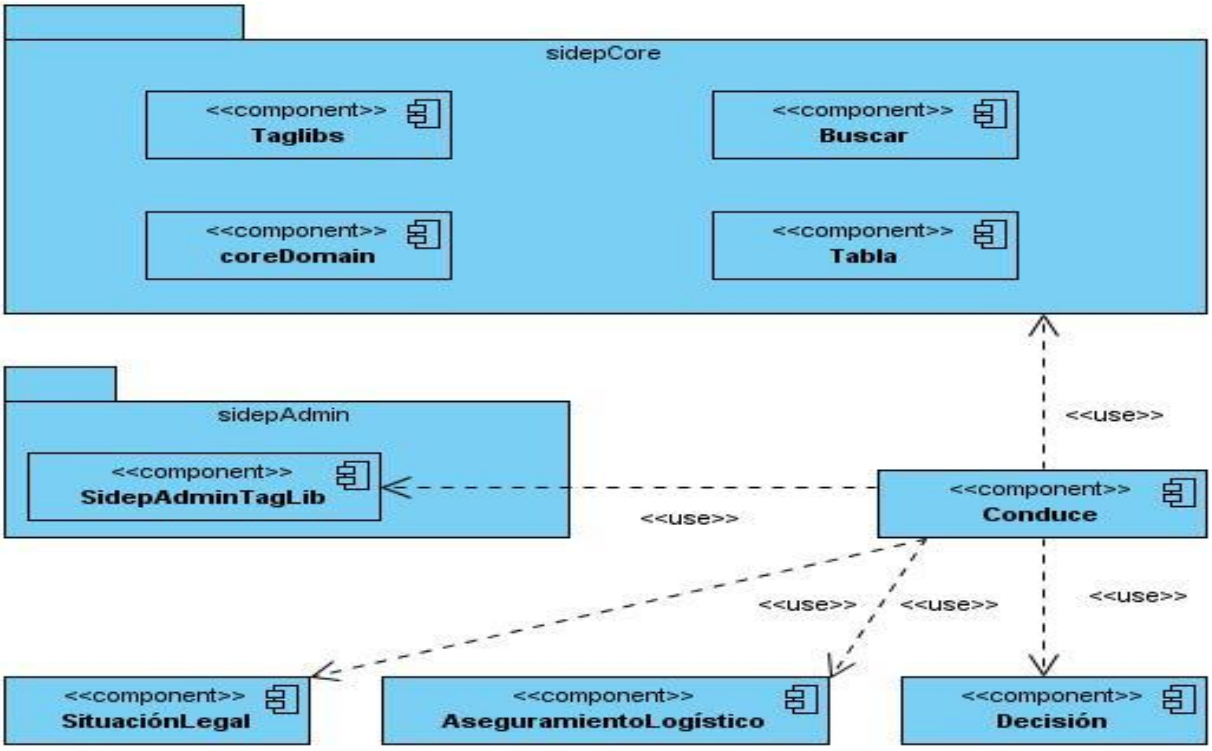

Figura 18: Relación del módulo Conducción con los demás módulos del sistema.

El componente *Conduce* se relaciona con el paquete de componentes *sidepCore* el cual contiene los buscadores, los *taglibs*, las clases y nomencladores, así como el componente tabla; estos componentes que se encuentran en el *sidepCore* son comunes para todos los módulos del proyecto. También se relaciona con el componente de *AseguramientoLogistico* el cual le brinda al conduce los expedientes de los internos a conducir. El componente *Decisión* le brinda al conduce un tipo de decisión por la que se puede crear un conduce por *Circular115*. Se relaciona con el paquete de componentes *sidepAdmin* del cual importa componentes que le brindan seguridad a la aplicación.

En la figura 19 se muestra el diagrama de componente del módulo Conducción.

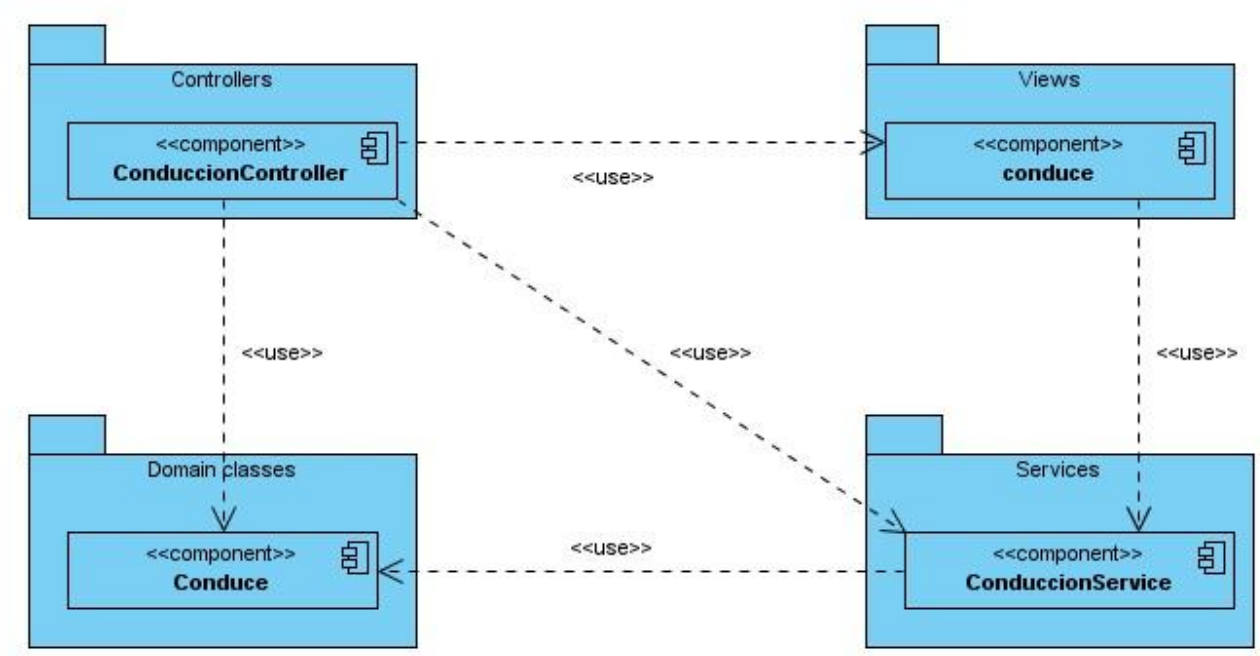

Figura 19: Diagrama de componentes del módulo Conducción.

En la figura anterior se muestran los componentes que contiene el módulo Conducción y la relación que existe entre estos. El paquete de componentes *Views* presenta componentes de tipo vista que son las encargadas de mostrar la información al usuario. Dentro del paquete de componentes *Domain clases* se van a encontrar componentes de tipo clases de dominio que persisten la información en la base de datos. En el interior del paquete de componentes *Service* se encuentran los componentes que siguen el patrón *XXXService.groovy* encargados de contener la lógica de negocio de la aplicación. El paquete de componentes *Controllers* el cual se relaciona con todos los demás paquetes contiene componentes de tipo *XXXController.groovy* y es el responsable gestionar el flujo de eventos de la aplicación.

3.2.2. Implementación de las clases del dominio

Las clases de dominio mantienen las relaciones existentes entre ellas siguiendo la estructura del modelo de relaciones que propone Grails. Estas relaciones pueden ser de uno-uno, uno-muchos, muchos-uno, muchos-muchos. Grails introduce elementos de lenguaje específicos de dominio, de tal manera que las validaciones de los campos de un objeto de dominio se realizan mediante el uso de la variable estática *constraints* como se muestra en la figura 20.

```
class ConduceJuicio extends Conduce {
    String horaJucio
    Date fechaJuicio
    NomSala sala
    Causa causa
    NomEjecucionJuicio ejecucion
    NomMotJuiNoCelebrado motivoNoCelebracion
    NomMotNoCelebrado motivoNoPresentacion
    NomCondicionConduce condicionConduce
    byte[]/* java.sql.Blob */archivo

    static hasMany = [delitos: NomDelito]

    static constraints = {
        ejecucion nullable:true
        horaJucio nullable: true
        archivo nullable: true
        motivoNoCelebracion nullable: true
        motivoNoPresentacion nullable: true
        sala nullable: true
    }

    static mapping = {
        id column: 'cjui_id'
    }
}
```

Figura 20: Clase de dominio conduceJuicio.

En la clase de dominio *conduceJuicio* se declaran los atributos referentes a los conduces realizados a juicios, se le especifican si pueden ser nulos en la sentencia con el patrón *atributo nullable: true;* también se le dice que esta clase va ser hija de la clase *Conduce* en la sentencia *extends Conduce*; además se le especifica que esta clase va a tener una lista de delitos en la propiedad estática siguiente: static hasMany = *[delitos: NomDelito]*. De esta forma se le cambia el identificador que Grails le asigna por defecto a la tabla ConduceJuicio, por cjui_id: static *mapping: id column: " cjui_id".*

3.2.3. Implementación de las vistas.

La vista es la responsable de mostrarle al usuario el estado del sistema y las acciones que tiene a su disposición. En Grails las vistas se desarrollan mediante *Groovy Server Pages* (gsp), una versión simplificada de JSP que permite intercalar expresiones en el código HTML.

En el fragmento de código (figura 21) de la vista *conduceTribunalCJ* se muestran los campos de la plantilla _*conduce* mediante el uso de la etiqueta de Grails *g:render* a la cual en el atributo *template* se le pasa la dirección de la acción del controlador que visualiza la plantilla siguiendo el siguiente formato */subsistema/controlador/método*. Debajo de la plantilla se muestran los campos propios del conduce a juicio como *provincia de tribunal* que es de tipo lista desplegable utilizando la etiqueta *s:select (*etiqueta propia del proyecto de tipo lista desplegable*)*; dentro de esta etiqueta se declara el atributo *req="true"* que significa que el campo es requerido, al atributo *nom* se le pasa el nombre de clase que contiene la lista de elementos que se desea mostrar. Se muestran campos de tipo fecha y hora utilizado las etiquetas *s:dateBox* y s:timerBox ambas creadas por el proyecto. Posteriormente se implementa una tabla utilizando la etiqueta *s:table* en la cual se especifica que va a listar de un servicio (*action="listFromService"*), se le especifica el nombre del servicio en el atributo *bean.* También muestra dos botones los cuales se declaran con etiqueta *s:button* y se encuentran dentro de la etiqueta *s:bottonBar* la cual posiciona los botones en la parte inferior derecha de la vista.

```
<g:render template="/registrolegal/conduces/conduce" />
<s:clear/>

<input type="hidden" id="motivoConduce.id" name="motivoConduce.id"/>

<s:select label="Provincia de tribunal" req="true" id="provincia.id" name="provincia.id" nom="NomProvincia" publisher="true"/>

<s:select label="Municipio de tribunal" req="true" id="municipio.id"  nom="NomMunicipio" publisher="true" disabled="true"/>

<s:select label="Tribunal" req="true" id="causa.tribunal.id" name="causa.tribunal.id" nom="NomTribunal" disabled="true"/>

<s:select label="Condición" req="true" id="condicionConduce.id" name="condicionConduce.id" nom="NomCondicionConduce"/>

<s:select label="Sala" req="true" id="conduceTribunalCJ.sala" name="sala.id" nom="NomSala"/>

<s:dateBox label="Fecha del juicio" req="true" id="fechaJuicio" name="fechaJuicio"/>

<s:timerBox name="horaJuicio" label="Hora del juicio" id="conduceTribunalCJ.horaJucio"/>

<s:clear/> <s:title title="Delitos"><br/>
<input type="hidden" id="delito.id" name="delito.id" />
<s:select label="Delito" name="delitos.id" nom="NomDelito"/>
<s:table id="tablaDelito" name="tablaDelito"  action="listFromService" bean="delitoService" label="delitos"  update="false"/>
<input type="hidden" id="delitoDireccion" value="${createLink(controller: "delito", action: "addDelito")}" />
<input type="hidden" id="borrandoDelitos" value="${createLink(controller: "delito", action: "delDelito")}" />
<s:clear/>
<s:buttonBar>
    <s:button label="Adicionar" name="conduceTribunalCJ.adicionarDelito"/>
    <s:button label="Cancelar" name="conduceTribunalCJ.cancelarDelito"/>
</s:buttonBar>
<s:clear/>
</s:title>
```

Figura 21: Vista conduceJuicio.

3.2.4. Implementación de los controladores.

El controlador es el responsable de recibir las solicitudes del usuario, manda a ejecutar la lógica de negocio al servicio sobre el modelo y decide la vista que debe mostrar a continuación.

En la figura 22 se muestra un fragmento del método *procesarConduceJuicio* del controlador *conduceTribunal*. En este método se obtienen los internos que se encuentran listados en el servicio *internoService* para registrarlos posteriormente; se convierten los datos de tipo fecha, se obtiene el archivo con la sentencia *params.nombreCampo.getBytes()*. Se verifica que la causa que se va a registrar no esté repetida. Posteriormente se crea el objeto de tipo conduce a Juicio y se le asignan los valores capturados en los parámetros a dichos atributos; se obtienen los delitos

del *servicio delitosServices* para asignarlos al atributo de ese tipo del objeto conduce a juicio para posteriormente ser salvado en la base de datos. La sentencia *params* es un mapa de los parámetros de solicitud de entrada que asocia los nombres de los parámetros con sus valores

```
def procesarConduceJuicio = {
    def causa

    def ex = internoService.items
    def expedientes = []
    ex.each {
        expedientes << Expedientelegal.get(it.toString().toLong())
    }


    params.urgencia = params.urgencia == "on"
    params.fechaInicio = new SimpleDateFormat("yyyy-MM-dd").parse(params.fechaInicio)
    params.fechaFin = new SimpleDateFormat("yyyy-MM-dd").parse(params.fechaFin)
    params.fechaJuicio = new SimpleDateFormat("yyyy-MM-dd").parse(params.fechaJuicio)
    params.archivo = !params.archivo ?: params.archivo.getBytes()

    conduccionService.params = params
    def list = conduccionService.buscarC(params)
    if (!list || list.size() <= 0) {
        causa = new Causa(params.causa)
        conduccionService.salvarCausa(causa)

    } else
    if (list.size() >= 0) {
        causa = list.get(0)

    }
    ConduceJuicio conduce = new ConduceJuicio(params)
    conduce.causa = causa
    conduce.motivoConduce=NomMotivoConduce.get(2.toLong())
    if(params.horaJuicio!=""){
    conduce.horaJucio=params.horaJuicio
    }
    if(params.hora!=""){
        conduce.hora=params.hora
    }

    conduce.estadoConduce=NomEstadoConduce.get(1)

    def delitos = delitoService.items
    for (int r=0; r < delitos.size(); r++) {
        conduce.addToDelitos(delitos[r])
    }
```

Figura 22: Método tipoConduce del controlador Conducción.

3.2.5. Implementación de los servicios

Un servicio es una clase cuyo nombre sigue el patrón *XXXService.groovy*, en él es donde se implementa la lógica de negocio de la aplicación. Para usar un servicio cualquier controlador puede declarar una variable del tipo de servicio que se desea llamar y mediante ésta acceder a los métodos implementados en él. A continuación se muestra en la figura 23 el método *situacionLegal*.

```
def situacionLegal(){
def expedientes=internoService.getExpedintes()
def sLegal=[]
def objExp

for(int i=0;i<expedientes.size();i++){
objExp=(Expedientelegal.get(expedientes[i].toString().toLong()))
    sLegal[i]=SituacionLegal.findByExpediente(objExp)
}

return sLegal
}
```

Figura 23: Método situacionLegal del servicio ConduceJuicioService.

En este método se obtienen los id de expedientes de los internos que se van a conducir en una lista. Luego se recorre la lista de id de expedientes y dado ese id se obtiene el objeto de tipo *Expediente Legal* y con ese objeto se busca la *Situación Legal* referente al mismo y se retorna una lista con la situación legal.

3.2.6. Implementación de los JavaScript

En los JavaScript se realiza la validación de los campos y conectan los eventos de las vistas a una determinada acción de un controlador para ser procesados.

```
delInterno:function(id) {
    sidep.core.app.send({
        url: "/sidep/buscarInterno/borrarInterno",
        content: { 'id' : id},
        load: function(params) {
            dijit.byId('tablaInternos').refresh();
            dijit.byId('tablaProcesos').refresh();
        }
    });
},

mostrarDialog: function(){
    sidep.core.app.mostrarDialog({
        title: "Buscar",
        href: dojo.byId('urlBase').value,
        id: "buscarDialog"
    });

},
```

Figura 24: Funciones del JavaScript ConduceTribunalCJ.

En la función *delInterno* a la cual se le pasa el identificador del expediente de un interno; envía este identificador al método *borrarInterno()* del controlador *buscarInterno* el cual recorre la lista de *items* del *servicio InternoService* que almacena los expedientes de interno que se encuentran en memoria en ese momento; este método busca el identificador y de encontrarlo lo elimina de la lista de *items*. Posteriormente se actualiza la tabla de internos.

La función *mostrarDialog* muestra en un diálogo el buscador de expediente legal de un interno; se definen los parámetros *title* que contiene el título del *dialogo*; en el campo *href* se le define la dirección del tipo de buscador que se desea utilizar; en este caso esta dirección va a estar en el atributo *value* de un campo de tipo input que se encuentra oculto en la vista, y en el atributo *id* define el identificador del dialogo.

3.2.7. Seguridad

La seguridad es un aspecto primordial en cualquier aplicación web. La seguridad en el módulo Conducción va a estar dada según el rol que tenga el usuario que va acceder a la aplicación. Cada usuario de la aplicación va tener asignado un determinado rol y según este varia el tipo de funcionalidad a la que puede acceder. Para garantizar este aspecto se utilizó una sentencia de

seguridad *@Security("clave")* establecida por el proyecto, la cual fue implementada en el controlador del menú del subsistema Registro Legal de nombre *RegistroLegalInicioController* (figura 20).

```
//Modulo conduces
@Security("registrarConduce")
def moduleConducePrincipal = {
    redirect controller: "conduccion", action: "index"
}
```

Figura 25 : Seguridad en el controlador RegistroLegalInicioController.

Esta sentencia de seguridad también es declarada en el controlador que muestra la página principal de cada funcionalidad del módulo Conducción como en el controlador *ConduccionController* (figura 26).

```
@Security("registrarConduce")
class ConduccionController {
    def conduccionService
    def delitoService
    def internoService
    def planSeguridadService
    def conduceJuicioService
    def procesosCausaPService

     def urlbase = "/registrolegal/conduces/"

    def index = {
        render view: "${urlbase}index"
    }
```

Figura 26 : Seguridad en el controlador ConduccionController.

La clave que se escribe dentro de la sentencia de seguridad fue establecida por el equipo de seguridad del proyecto, la cual va a ser diferente para cada caso de uso del módulo como se muestra en la figura 27.

Nombre	Clave	Tipo
Asignar funcionarios al conduce	asignarFuncConduce	
Asignar transporte al conduce	asignarTransConduce	
Consultar conducciones	consultarConducciones	
Actualizar conduce	actualizarConduce	
Registrar y actualizar conduce de urgencia	crud-d-ConduceUrgencia	
Registrar ejecución de conduce	registrarEjecucionConduce	
Registrar datos del conduce	registrarConduce	

Figura 27 : Claves de seguridad para el módulo Conducción.

3.2.8. Tratamiento de errores

El tratamiento de errores es un pilar fundamental en cualquier aplicación web debido a que contrarresta posibles errores y guía al usuario por el correcto flujo de eventos por el cual debería seguir. Para ello en la aplicación se les da tratamiento a los mismos en los JavaScript y en los Controladores.

Tratamiento de errores en los JavaScript

En las clases JavaScript se realizan dos pasos cuando ocurre algún error en la validación de los campos de la vista. El primer paso es marcar en rojo el borde del campo que tiene el error y el segundo paso es mostrar un mensaje en la parte superior de la vista “Los campos en rojo son requeridos” tal y como se muestra en la figura 28.

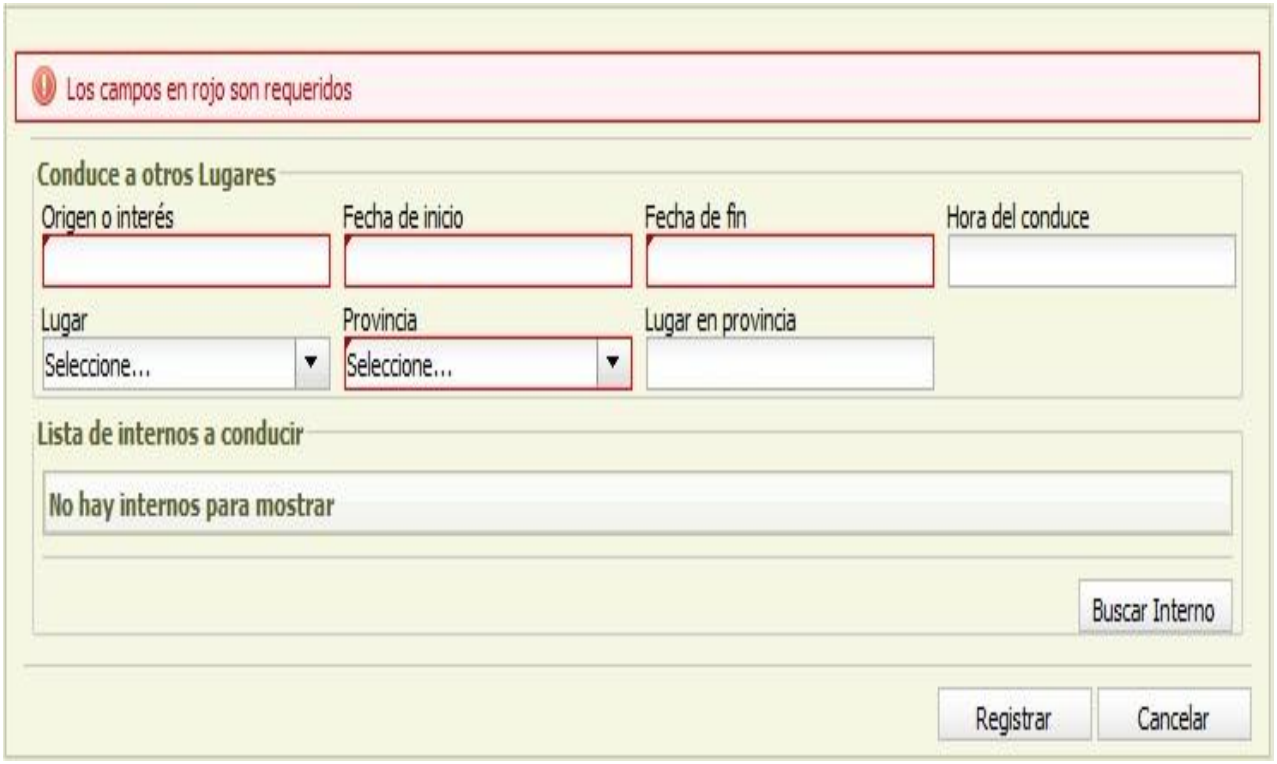

Figura 28: Vista lugar.

Para lograr este resultado en la vista se valida cada campo buscando que cumpla con las especificaciones requeridas y de no cumplir se pone en falso la variable *valido* y se marca el borde del campo con el color rojo utilizando la siguiente sintaxis *sidep.core.app.cambiarColorBorde ("idCampo", true)* tal como se muestra en la figura siguiente.

```
validacionConduce: function(){
 var valido = true;

 if(dojo.byId('origen.id').value == ""){
       sidep.core.app.cambiarColorBorde("origen.id");
       valido = false;
   }else sidep.core.app.cambiarColorBorde("origen.id", true);

    var fechaI=dojo.byId('fechaInicio.id').value;
    var patron = new RegExp(/^\d{1,2}\/\d{1,2}\/\d{2,4}$/)
    if (fechaI =="" || !fechaI.match(patron) ){
        sidep.core.app.cambiarColorBorde('fechaInicio.id');
        valido = false;
    }
    else{
        sidep.core.app.cambiarColorBorde("fechaInicio.id", true);
    }

    var fechaF=dojo.byId('fechaFin.id').value;
    if (fechaF =="" || !fechaF.match(patron) ){
        sidep.core.app.cambiarColorBorde('fechaFin.id');
        valido = false;
    }
    else{
        sidep.core.app.cambiarColorBorde("fechaFin.id", true);
    }

    var patronHora = new RegExp(/^[0-9]{2,2}:[0-9]{2,2}$/)
    var hora=dojo.byId('hora.id').value
    if (hora !="" && !hora.match(patronHora) ){
        sidep.core.app.cambiarColorBorde('hora.id');
        valido = false;
    }
    else{
        sidep.core.app.cambiarColorBorde("hora.id", true);
    }

    return valido;
},
```

Figura 29: Función validacionConduce referente a la vista lugar.

La función *validacionConduce* después de validar todos los campos de la vista retorna la variable *valido* que de ser *true* indica que no hay errores en los campos y de ser *false* lo contrario. La misma es ejecutada desde la función *gestionarConduceTribunal()*, que verifica si *validacionConduce()* retorna verdadero entonces envía los datos del formulario *lugar* al controlador *OtrosConduces* para ser procesados; de lo contrario si *validacionConduce()* retorna falso se muestra un mensaje de error (figura 30).

```
gestionarOtrosConduces:function(){
    if(!sidep.registrolegal.conduces.lugar.validacionConduce() ){
    document.getElementById('lugar').submit();
    }else{
    sidep.core.app.mostrarMensaje("Los campos en rojo son requeridos", "error");
    }
},
```

Figura 30: Función gestionarConduceTribunal referente a la vista lugar.

Tratamiento de errores en el controlador

Para el tratamiento de errores en la clase controladora se siguen básicamente tres pasos el primero es detectar el error; el segundo es informárselo al usuario mediante un mensaje y el tercero es re direccionarlo a la vista donde cometió el error como se muestra en el fragmento de código del método *procecesarConduceJuicio* del controlador ConduceTribunalController que procesa los datos del mismo tipo de conduce como se muestra a continuación.

```
if(sc.validate()){
  sc.save()
  println"***La solicitud se registró correctamente***"
}else{
 flash.error = "Se produjeron errores a la hora registrar un solicitud de conducción"
 redirect(action: "conduce")
 println(sc.errors)
}
```

Figura 31: Fragmento de código del método procesarConduceJuicio del controlador ConduceTribunal.

En el fragmento de código mostrado primero se valida el objeto de la clase que se quiere guardar para ello se utiliza el método *validate()* que verifica las reglas de validación definidas en los *constraints* de la clase del objeto. Si el método es falso entonces habrá errores en la propiedad *errors* del objeto de dicha clase. En caso que *validate()* devuelva verdadero, se aplican las reglas del negocio pendientes y se guarda la instancia del objeto en la base de datos mediante una llamada al método *save().* El método *save()* permite insertar el registro en la base de datos o actualizarlo si ya existía.

3.2.9. Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama de UML que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

Para el correcto despliegue del módulo se necesita una estructura tecnológica, que estará compuesta básicamente por una o muchas PC clientes, el Servidor Web y el de Base Datos como se muestra en la figura 32.

Para las computadoras clientes:

- Se requiere tengan tarjeta de red.
- Se requiere tengan al menos 128 MB de memoria RAM.
- Se requiere al menos 100MB de disco duro.
- Procesador 800 MHz como mínimo.

Para los servidores:

- Se requiere tarjeta de red.
- Se requiere tenga al menos 512MB de RAM.
- Se requiere al menos 40GB de disco duro.
- Procesador 2.0 GHz como mínimo.

La computadora cliente estará comunicada con el servidor web mediante cables de tipo UTP de categoría 4 o 5 utilizando el protocolo de seguridad HTTPS por el puerto 443. El servidor Web se comunica con el servidor de bases de datos mediante cable UTP categoría 4 o 5 o por fibra óptica; utilizando el protocolo TCP/IP por el puerto 1521.

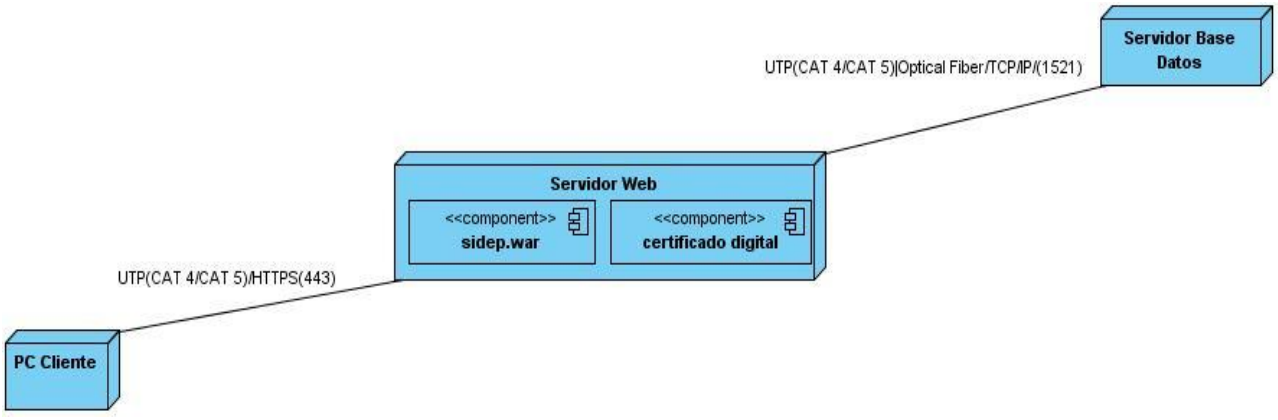

Figura 32: Diagrama de despliegue.

La arquitectura lógica de tres capas de la aplicación se puede desplegar físicamente de la siguiente forma:

El componente Conduce se encontrará ubicado en el servidor de web; los componentes de tipo vista se mostrarán en un navegador web el cual se encontrara ubicado en la PC cliente; por cada componente de tipo clase de dominio se va generar una tabla en el servidor de base de datos.

3.2.10. Métricas

Para la medición de algunas variables tales como complejidad de implementación, reutilización, complejidad del mantenimiento y cantidad de pruebas se realizaron las métricas tamaño operacional de clase y relaciones entre clases al módulo Conducción.

Las métricas Permiten medir de forma cuantitativa la calidad de los atributos internos del software. Esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. (11)

Tamaño operacional de clase TOC

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

Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase. La cual fue calculada para cada clase de la siguiente forma:

Responsabilidad	Categoría	Criterio
	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Tabla 6 : Criterio de evaluación responsabilidad.

Donde la variable *Prom* es el promedio de la cantidad de procedimientos.

Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase; esta fue calculada de la siguiente forma:

Complejidad de implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Tabla 7 : Criterio de evaluación para la complejidad de implementación.

Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase; la cual fue obtenida de la forma siguiente:

Reutilización (N)	
Clasificación	**Valores**
Baja	N > 2*Promedio
Media	Promedio < N < 2* Promedio
Alta	N <= Promedio.

Tabla 8 : Criterio de evaluación para la reutilización.

A continuación se muestran los resultados obtenidos tras la aplicación de la métrica tamaño de clase:

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
ActPrincipalService	3	Baja	Baja	Alta
ActualizarConService	8	Media	Media	Media
ActVehiConService	5	Media	Media	Media
AsigFunService	11	Alta	Alta	Baja
ListarOficialService	5	Media	Media	Media
ListConSinFunService	3	Baja	Baja	Alta
ListConFunPlanService	4	Baja	Baja	Baja
ListVehiculosConService	10	Alta	Alta	Baja
BuscarConduceService	1	Baja	Baja	Alta
ResultadoConducesService	3	Baja	Baja	Alta
ConsultarCService	3	Baja	Baja	Alta
ConsultVConService	3	Baja	Baja	Alta
ListFunConService	3	Baja	Baja	Alta
ListFunConService	3	Baja	Baja	Alta
ListConPlanService	3	Baja	Baja	Alta
ListConSinPlanService	3	Baja	Baja	Alta
ListOficialConService	3	Baja	Baja	Alta
VehiConPlanService	4	Baja	Baja	Baja
ActConduceService	3	Baja	Baja	Alta
RegistrarEjecucionService	4	Baja	Baja	Baja
UrgenciaService	12	Alta	Alta	Baja
Circular115Service	1	Baja	Baja	Alta
ConduceJuicioService	2	Baja	Baja	Alta
DelitoService	3	Baja	Baja	Alta
InternoService	5	Media	Media	Media
OtrosConduceService	2	Baja	Baja	Alta
ConduccionService	4	Baja	Baja	Baja
DecCircularService	3	Baja	Baja	Alta
ProcesosCausaPService	3	Baja	Baja	Alta

Tabla 9 : Análisis de la métrica TOC.

Variables	Baja	Media	Alta
Responsabilidad	76%	14%	10%
Complejidad	76%	14%	10%
Reutilización	24%	14%	62%

Tabla 10 : Resultados de la métrica TOC.

Luego de haber aplicado esta métrica, se obtuvo como resultado que las 29 clases analizadas del diseño son pequeñas en su totalidad. Esto demuestra que las mismas no tienen grandes responsabilidades por lo que permite la reutilización de clases demostrando que el sistema tendrá una implementación a nivel medio. Por tanto se puede afirmar que los resultados obtenidos por la métrica son positivos.

Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Responsabilidad: Responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta. (11)

Complejidad del mantenimiento: Grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto. (11)

Reutilización: Grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. (11)

Cantidad de pruebas: Número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, modulo, clase, conjunto de clases) diseñado. (11)

Para los cuales están definidos los siguientes criterios y categorías de evaluación que se muestran en la siguiente tabla:

Atributo	Categoría	Criterio
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	<=Promedio
Cantidad de pruebas.	Baja.	<=Promedio

	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

Tabla 11 : Criterio de evaluación para métrica RC.

A continuación se muestran los resultados obtenidos tras la aplicación de la métrica relaciones entre clases:

Clase	Cantidad de Relaciones de Uso	Acoplamiento
ConduccionController	5	Alto
ActualizarConController	5	Alto
AsignarFuncionariosController	2	Medio
AsigTransConController	2	Medio
BuscarConController	2	Medio
ReResultBusController	6	Alto
ConsultarCController	6	Alto
ConPlanSController	11	Alto
RegistrarEjecucionController	6	Alto
CondUrgenciaController	4	Alto
BuscarInternoController	2	Medio
Circular115Controller	2	Medio
ConduceTribunalController	3	Alta
DelitoController	1	Bajo
OtrosConducesController	3	Alto

Tabla 12 : Análisis de la métrica RC.

Figura 33: Comportamiento de los valores de dependencia.

Cantidad de clases 15	Ninguno	Baja	Media	Alta
Acoplamiento	0%	7%	36%	57%
Complejidad de mantenimiento		54%	33%	13%

Cantidad de pruebas		54%	33%	13%
Reutilización		7%	33%	60%

Tabla 13 : Resultados de la métrica RC.

Se puede concluir que el diseño propuesto está entre los límites aceptables de calidad, debido a que los atributos de calidad se encuentran en un nivel satisfactorio. La Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 60%.

Pruebas de calidad

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un sistema informático. Para realizar las pruebas a una aplicación es necesario definir la estrategia a seguir.

3.3. Estrategia de prueba

Para realizar las pruebas de calidad en el módulo Conducción se definió la siguiente estrategia de pruebas:

Método de prueba: El método de prueba seleccionado es caja negra. Es aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. (26)

Técnica de Prueba: La técnica de prueba seleccionada son los casos de pruebas. Son un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados, desarrollados para cumplir un objetivo en particular o una función esperada. Siempre es ejecutada como una unidad, desde el comienzo hasta el final. (26)

Debe verificar:

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

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

Tipo de prueba: Las pruebas de función son aquellas que se realizan fijando su atención en la validación de las funciones, métodos, servicios, caso de uso. (26)

Nivel de prueba

Prueba de desarrollador: La prueba de desarrollador indica los aspectos de diseño e implementación de las pruebas más adecuadas que debe llevar a cabo el equipo de desarrolladores, a diferencia de la prueba independiente. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas de desarrollador que la diseñó e implementó. Tradicionalmente, la prueba de desarrollador se consideraba, principalmente, con respecto a la prueba de unidad. Algunos desarrolladores también realizan pruebas de integración de diferentes niveles, aunque depende básicamente de la cultura y otros aspectos contextuales. Es recomendable que la prueba de desarrollador no cubra únicamente unidades independientes de prueba aisladas. (26)

3.3.1. Diseño de los casos de prueba

A continuación se muestra el caso de prueba para el caso de uso Registrar conduce:

Escenario	Descripción	Motivo de conduce	Origen o interés	Fecha inicio	Fecha fin	Hora	Lugar	Provincia	Respuesta del sistema	Flujo central
EC 1.1 Registrar datos del conduce	Se registran los datos del conduce en dependencia del motivo del conduce	V	V	V	V	V	V	V	El sistema registra los datos.	1. Selecciona la opción “Registrar Conduce”. 2. Selecciona el campo "Motivo de Conduce" y presiona el botón siguiente. 3. El sistema muestra los campos referentes al conduce. 4. Introduce los datos y oprime el botón "Registrar".
EC 1.2 Cancelar	Se cancela la operación.	NA	NA	NA	NA	NA	NA	NA	Se cancela el registro del conduce.	1. Selecciona el botón “Cancelar”.
EC 1.3 Faltan datos obligatorios	Indica que faltan datos que son obligatorios para el registro de la	I	V	V	V	V	V	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "motivo de conduce" falto por	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".

	visita.								llenar.	
		V	I	V	V	V	V	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "fecha inicio" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".
		V	V	I	V	V	V	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "fecha fin" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".
		V	V	V	I	V	V	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "fecha fin" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".

		NA	V	V	V	I	V	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "hora" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".
		NA	V	V	V	V	I	V	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "lugar" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".
		NA	V	V	V	V	V	I	Muestra el mensaje “Rectifique los campos en rojo” indicando que el campo "provincia" falto por llenar.	1. Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Registrar".
EC 1. 4 Conduce a tribunal	Se registran los datos del conduce al tribunal	NA	V	V	V	V	I	V	El sistema registra los datos.	1. Introduce los datos del conduce a tribunal y oprime el botón "Registrar".

EC 1. 4 Conduce a tribunal	Se registran los datos del conduce al tribunal	NA	V	V	V	V	I	I	El sistema registra los datos del conduce por circular 15.	1. El sistema muestra una tabla de decisión de conducción de circular115. 2. Selecciona el campo una decisión y presiona el botón siguiente. 3. El sistema muestra los campos referentes al conduce por circular115. 4. Introduce los datos y oprime el botón "Registrar".

Tabla 14 : Caso de prueba del caso de uso Registrar Conduce.

3.3.2. Resultado de las pruebas

Se realizaron tres iteraciones. En la primera iteración se detectaron 10 no conformidades que se muestran en la tabla 15.

No	No conformidad	Ubicación	Estado
1	La fecha fin del conduce tiene que ser mayor que la fecha de inicio.	Conduce/Registrar Conduce	Resuelta
2	Adicionar internos al conduce solo cuando tengan el expediente habilitado.	Conduce/Registrar Conduce	Resuelta
3	Brindar la posibilidad de que el usuario pueda eliminar de la tabla vehiculoConduce.	Conduce/Asignar Transporte	Resuelta
4	Quitar el campo motivo del conduce de la vista lugar.	Conduce/Registrar Conduce	Resuelta
5	Quitar el campo estado del conduce de la vista conduce.	Conduce/Registrar Conduce	Resuelta
6	Ponerle la tilde a la palabra rótulo.	Conduce/ Asignar Transporte	Resuelta
7	Ponerle la tilde a la palabra vehículo.	Conduce/ Asignar Transporte	Resuelta
8	Ponerle la tilde a la palabra número.	Conduce/Registrar Conduce	Resuelta
9	Brindar la posibilidad de que el usuario pueda eliminar de la tabla oficialConduce.	Conduce/Asignar Funcionario	Resuelta
10	Mostrar mensaje de error “Tiene que asignarle un vehículo al conduce”	Conduce/ Asignar Transporte	Resuelta

Tabla 15 : Tabla de no conformidades de la primera iteración.

En la segunda iteración se encontraron 5 no conformidades (Ver tabla 16) y en la tercera no se encontraron.

No	No conformidad	Ubicación	Estado
1	No se le informa al usuario cuando en conduce a	Conduce/Registrar Conduce	Resuelta

	Juicio no se registra correctamente.		
2	No se le informa al usuario cuando en conduce por circular115 no se registra correctamente.	Conduce/Registrar Conduce	Resuelta
3	No se le informa al usuario cuando en conduce a otros lugares se registra correctamente.	Conduce/Registrar Conduce	Resuelta
4	No se informa cuando al conduce no se asigna un interno.	Conduce/Registrar Conduce	Resuelta
5	No se informa cuando la causa está repetida.	Conduce/Registrar Conduce	Resuelta

Tabla 16 : Tabla de no conformidades de la segunda iteración.

3.4. Conclusiones parciales

La validación en las distintas clases de la aplicación ayudó a mejorar y asegurar la calidad del desarrollo de la misma, permitiendo reducir el número de errores y su posterior corrección. La aplicación de la métrica RC demostró que la cantidad de pruebas a realizar es mínima, así la baja complejidad de mantenimiento del módulo. Las pruebas realizadas dieron la medida de cómo el módulo iba mejorando en cada iteración hasta que no se encontraron no conformidades, en su mayoría fáciles de corregir.

Conclusiones generales

Como resultado del trabajo realizado se desarrolló el módulo Conducción del SIDEP a partir de los requisitos de software establecidos con el cliente y haciendo uso de la arquitectura propuesta por el proyecto.

Se realizó un estudio de las tecnologías y herramientas a utilizar en el diseño e implementación del módulo así como del análisis de los requisitos de software y del modelo de negocio correspondientes.

Las actividades de diseño e implementación fueron ejemplificadas a través de diagramas de clases y fragmentos de código fuente cumpliendo así con los objetivos propuestos para este trabajo.

La realización de las pruebas de funcionalidad comprobó el adecuado funcionamiento de las principales funcionalidades del módulo, cumpliendo con los requisitos planteados por el cliente.

Recomendaciones

Se recomienda:

- Darle seguimiento a los procedimientos de trabajo en las prisiones cubanas por si surgen cambios en la planificación y ejecución de los procesos de conducción.
- Realizar las pruebas de aceptación del módulo Conducción para que el cliente valide la aplicación.

Referencias bibliográficas

1. *"Grails in Action"*2009GreenwichManning Publications Co
2. **Brito, Nacho.** Manual de desarrollo web con Grails. [En línea] [Citado el: 13 de 12 de 2011.] http://www.manual-de-grails.es.
3. **The Dojo Foundation.** DojoTolkit. [En línea] http://www.dojotoolkit.org/.
4. Larman, C. *UML y Patrones.*
5. Pérez, J. A. *Introducción a JavaScript.*
6. *PROYECTO TÉCNICO V1.0.*
7. Requejo, A. E. (2008). *Diseño e implementación de las capas de Negocio y Acceso a dato de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP.* Ciudad de La Habana.
8. Rational Software Corporation.*RUP."Rational Unified Process".* 2003.
9. EVA.*EVA.*[Online][Cited:marzo8,2012.] http://evapostgrado.uci.cu/mod/resource/view.php?id=11072.
10. Arregui, J. J. (2005). *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* Madrid.

12. **JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES.** *El Proceso Unificado de desarrollo del software.* Madrid : sn, 2000.
13. **Visual Paradigm International.** Visual Paradigm. [En línea] [Citado el: 14 de mayo de 2010.] http://www.visual-parading.com.
14. **Tomcat, Apache.** Apache Tomcat. [En línea] [Citado el: 3 de 3 de 2012.] http://www.apache.org.
15. **ORACLE CORPORATION.** *Oracle Database Documentation Library.*
16. **Embarcadero Technologies Inc.** Embarcadero. [En línea] [Citado el: 28 de 11 de 2011.] http://www.embarcadero.com/products/er-studio.
17. **Subversion.** Subversion. [En línea] 2001-2009. [Citado el: 28 de 11 de 2011.] http://subversion.tigris.org/.
18. **Abdul-Jawad, Bashar.** *Groovy and Grails Recipes.* 2009. ISBN-13 (pbk): 978-1-4302-1600-1, ISBN-13 (electronic): 978-1-4302-1601-8.
19. **Netbeans.** Netbeans. [En línea] [Citado el: 28 de 11 de 2011.] http://netbeans.org/kb/trails/platform.html.

20.**Oracle**.Oracle. [En línea] http://docs.oracle.com/javaee/1.4/tutorial/doc/

21. **Java**.Java. [En línea] http://www.java.com/es/about/

22.Goodman,Danny. Javascript_Bible_4th_Edition.

23. Eidos. Lenguaje HTML.

24. EVA BD 1. Patrones_de_diseno_de_BD.

Bibliografía

1. Arregui, J. J. (2005). *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* Madrid.
2. Brito, N. (s.f.). *Manual de desarrollo web con Grails.*
3. Calahorro, N. B. (s.f.). *Manual de desarrollo web con Grails.*
4. Eidos, G. (2000). *Lenguaje HTML.*
5. Goodman, D. (s.f.). *Javascript_Bible_4th_Edition.*
6. *http://www.dojotoolkit.com/*. (s.f.).
7. Larman, C. (s.f.). *UML y Patrones.*
8. Pérez, J. A. (s.f.). *Introducción a JavaScript.*
9. (s.f.). *PROYECTO TÉCNICO V1.0.*
10. Requejo, A. E. (2008). *Diseño e implementación de las capas de Negocio y Acceso a dato de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP.* Ciudad de La Habana.
11. SMITH, G., & LEDBROOK, P. (2009). *"Grails in Action".* Greenwich: Manning Publications Co.

Glosario de término

AJAX: JavaScript asíncrono y XML.

API: Interfaz de Programación de Aplicaciones**.** Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas vulgarmente "librerías").

CASE: *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Computadora).

CSS: Hojas de estilo en cascada (*Cascading Style Sheets* por sus siglas en inglés).

DOM: *Document Object Model*.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

HTML: *Hyper Text Markup Language* (Lenguaje de Etiquetas de Hipertexto).

HTTP: *Hypertext Transfer Protocol.*

HTTPS*: HTTP Secure*.

IDE: *Integrated Development Enviroment* (Ambiente Integrado de Desarrollo).

IPP: *Internet Printing Protocol.*

JVM: *Java Virtual Machine* (Máquina Virtual de Java).

MVC: Modelo Vista Controlador.

Plugins: Un plugin o plug-in, es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.

RUP: *Rational Unified Process.*

TCP/IP: *Transmission Control Protocol/Internet Protocol.*

UML: Lenguaje de Modelado Unificado.

URL: *Uniform Resource Locator* (Localizador Uniforme de Recursos).

UTP: Unshielded Twisted Pair.

XML: *Extensible Markup Language.*

Anexos

Anexo 1 Diagramas de clases del diseño

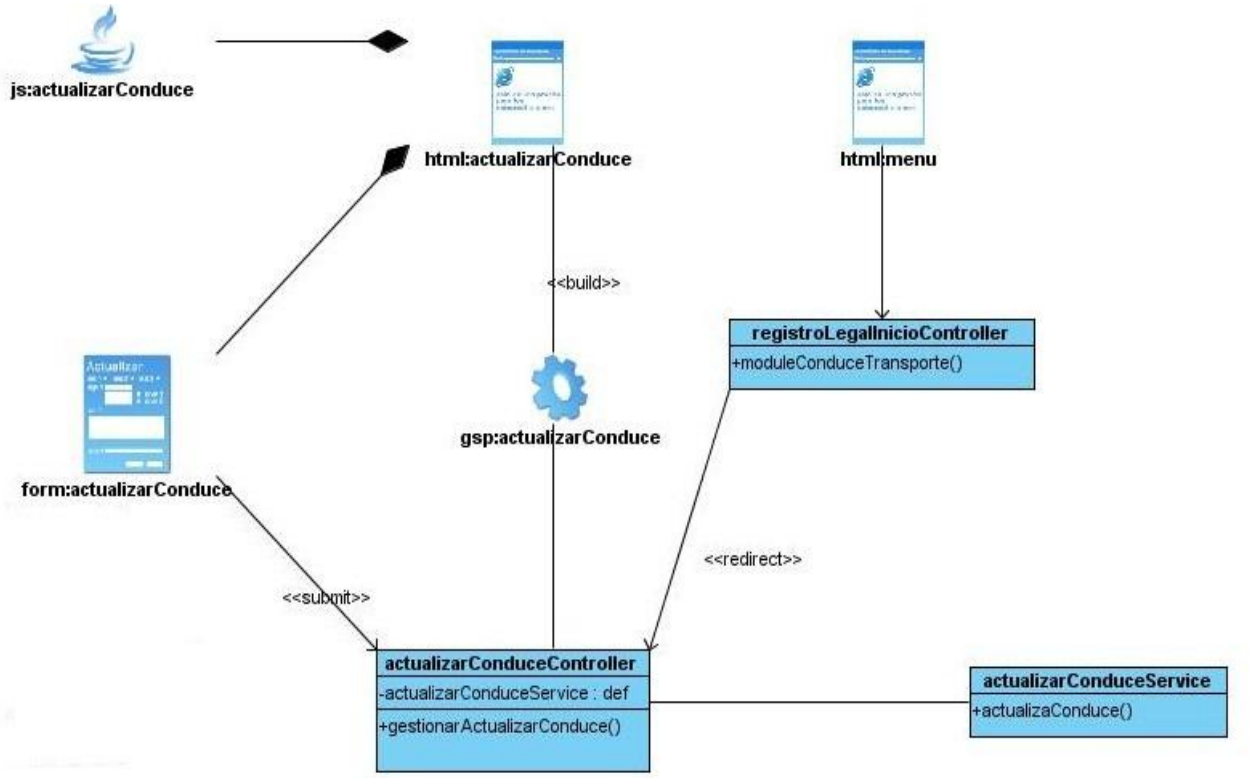

Figura 34: Capa web del diagrama de Clases del diseño Actualizar Conduce.

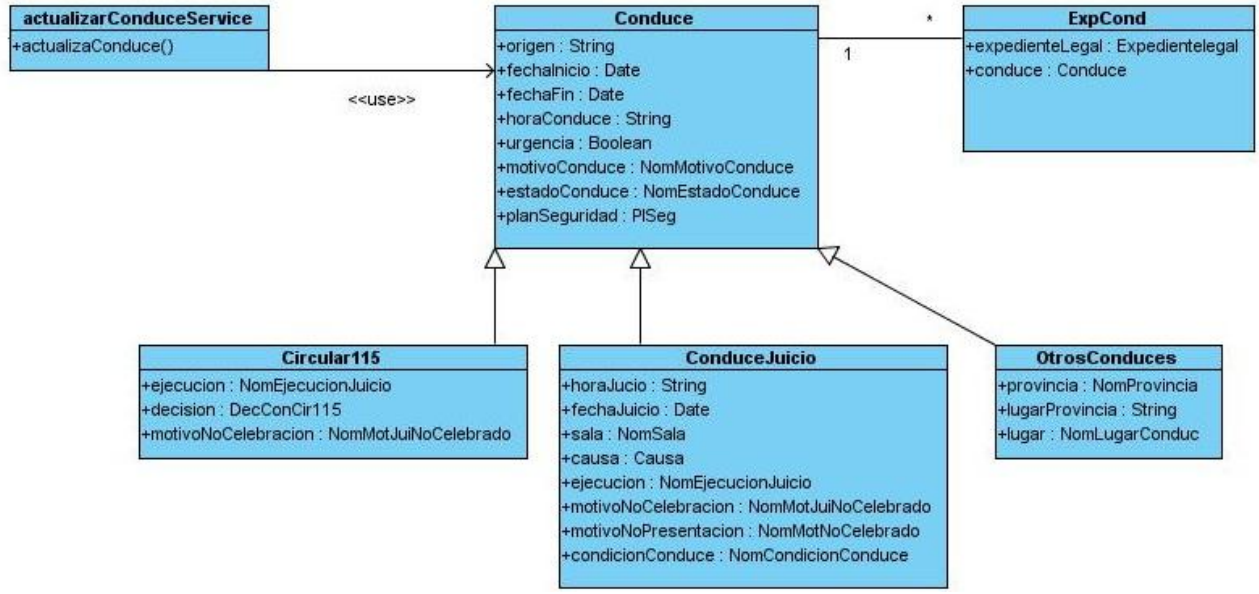

Figura 35: Capa datos y capa de servicio del diagrama de Clases del diseño Actualizar Conduce.

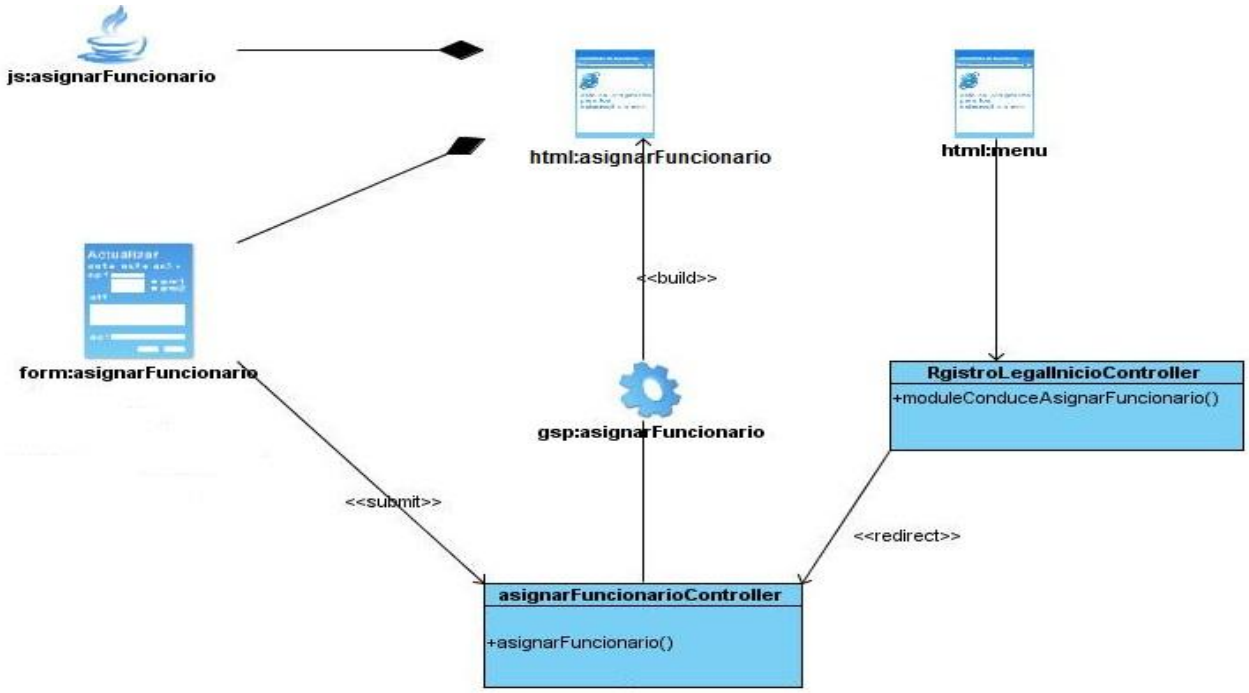

Figura 36: Capa web del diagrama de Clases del diseño Asignar Funcionarios al Conduce.

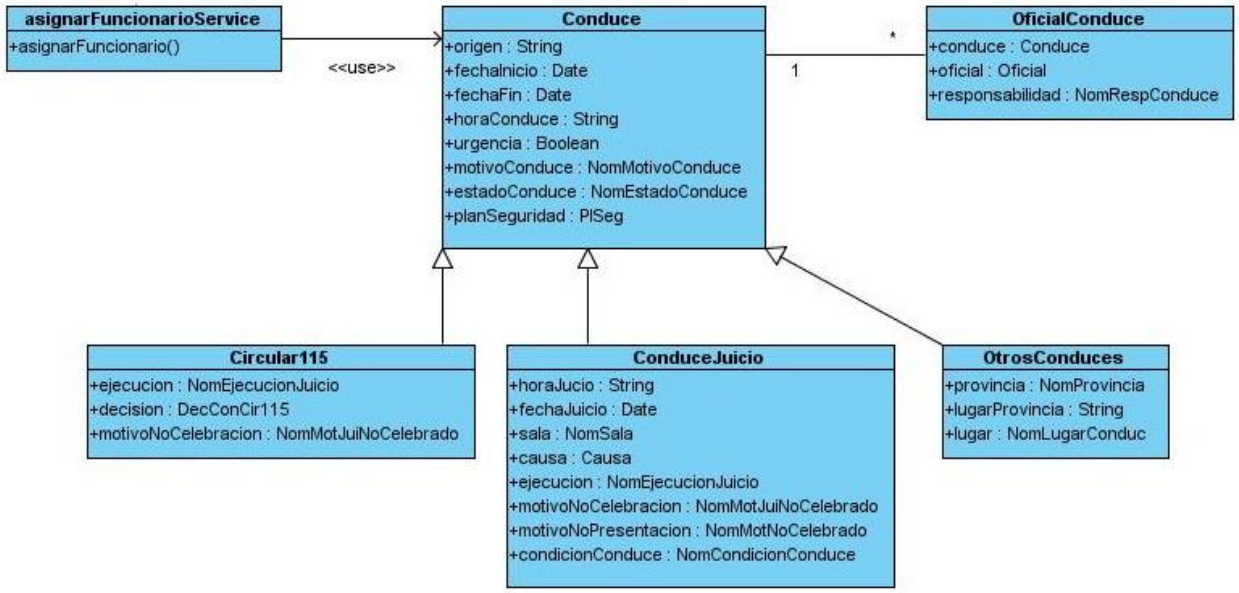

Figura 37: Capa datos y capa de servicio del diagrama de Clases del diseño Asignar Funcionarios al Conduce.

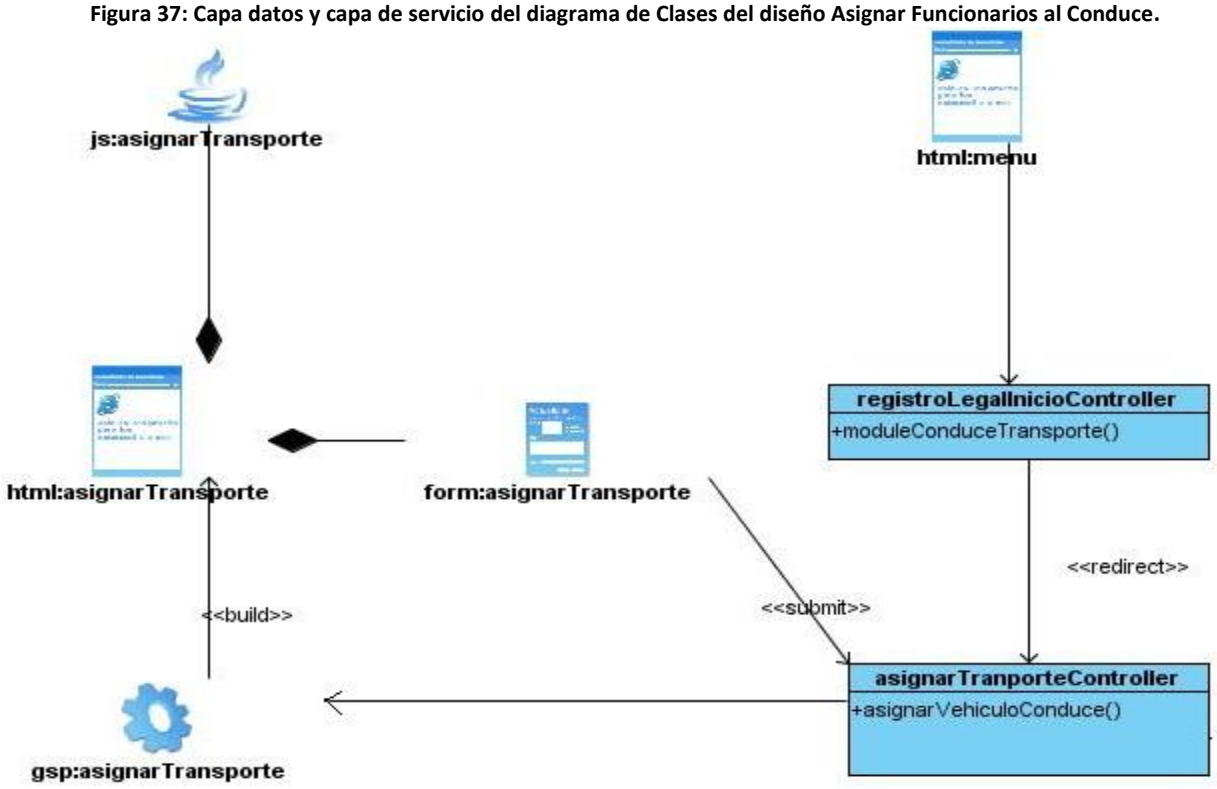

Figura 38: Capa web del diagrama de Clases del diseño Asignar Transporte Conduce.

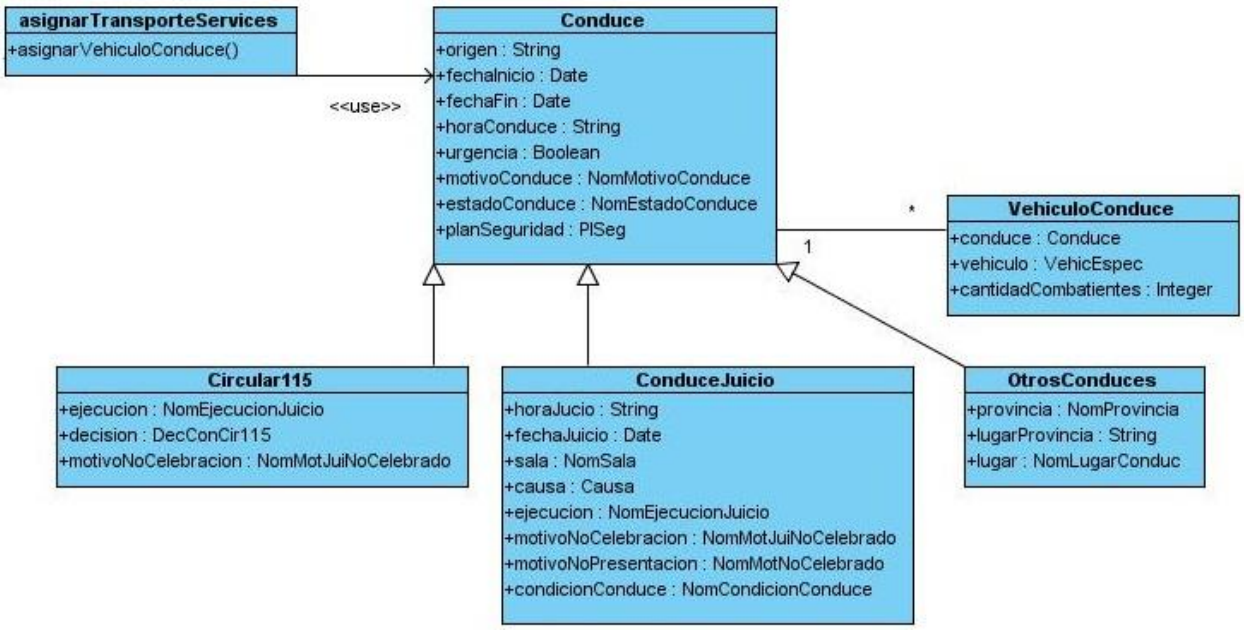

Figura 39: Capa de datos y capa de servicio del diagrama de Clases del diseño Asignar Transporte Conduce.

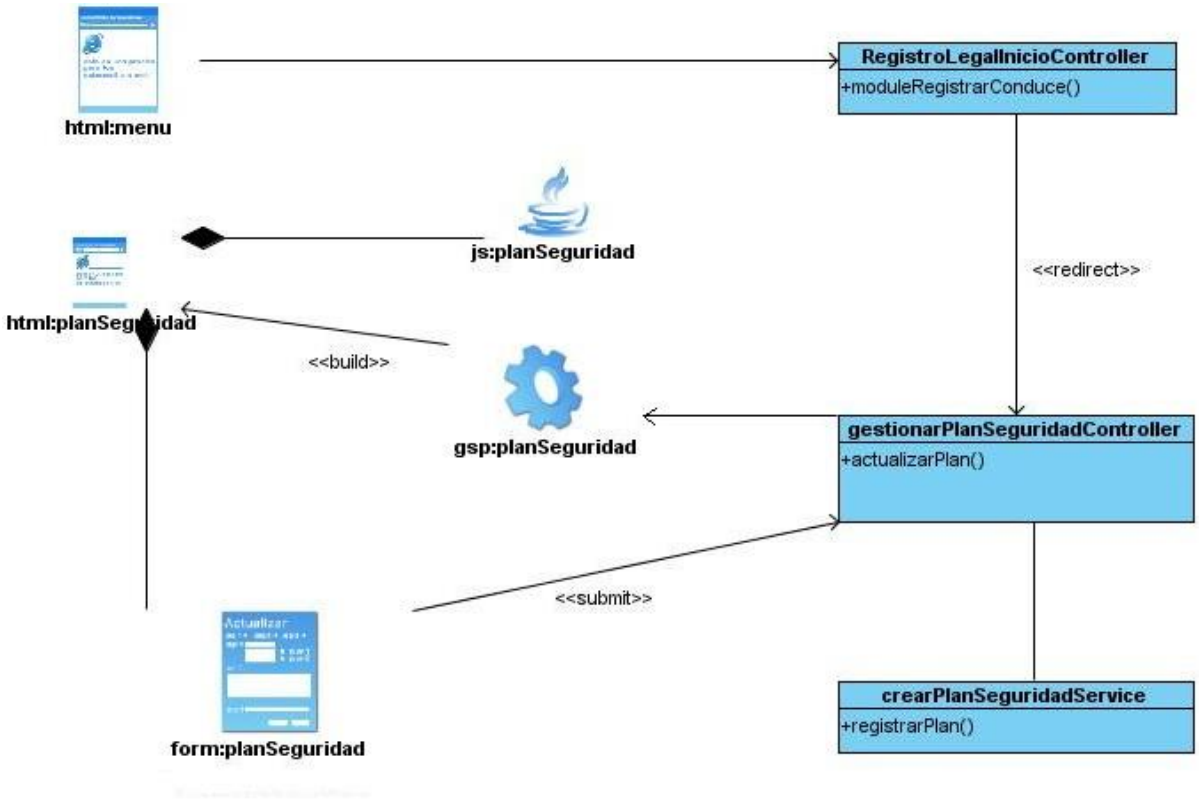

Figura 40 : Capa web del diagrama de Clases del diseño Crear Plan de Seguridad.

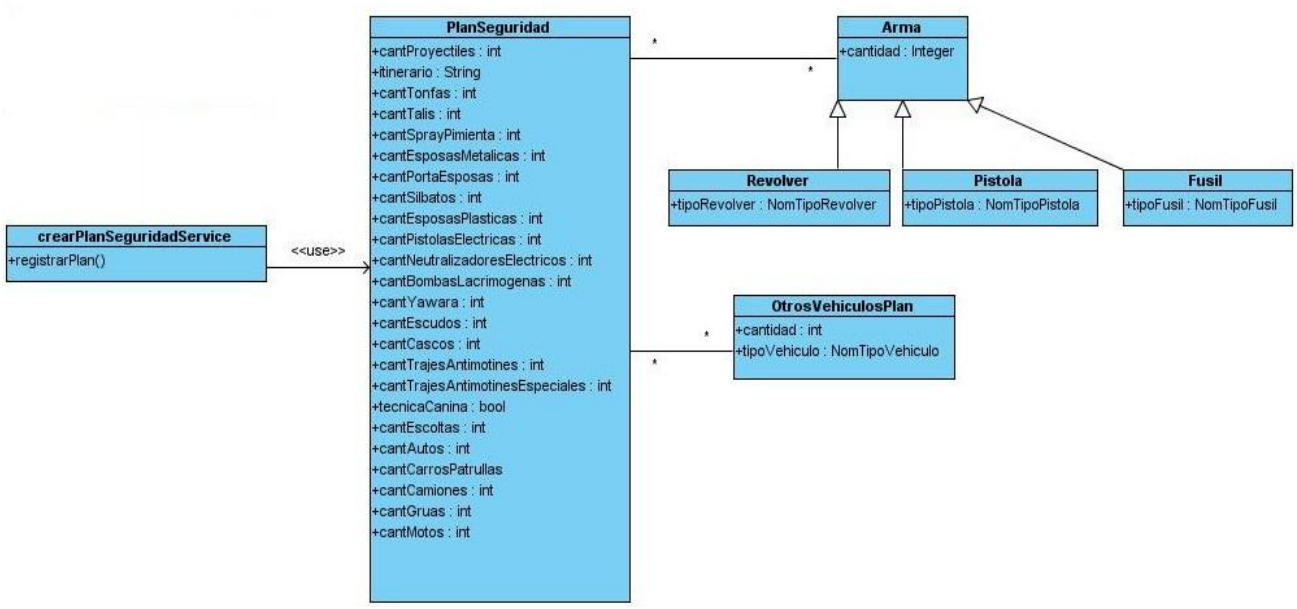

Figura 41 : Capa de datos y capa de servicio del diagrama de Clases del diseño Crear Plan de Seguridad.

Anexo 2 Diagramas de Comunicación

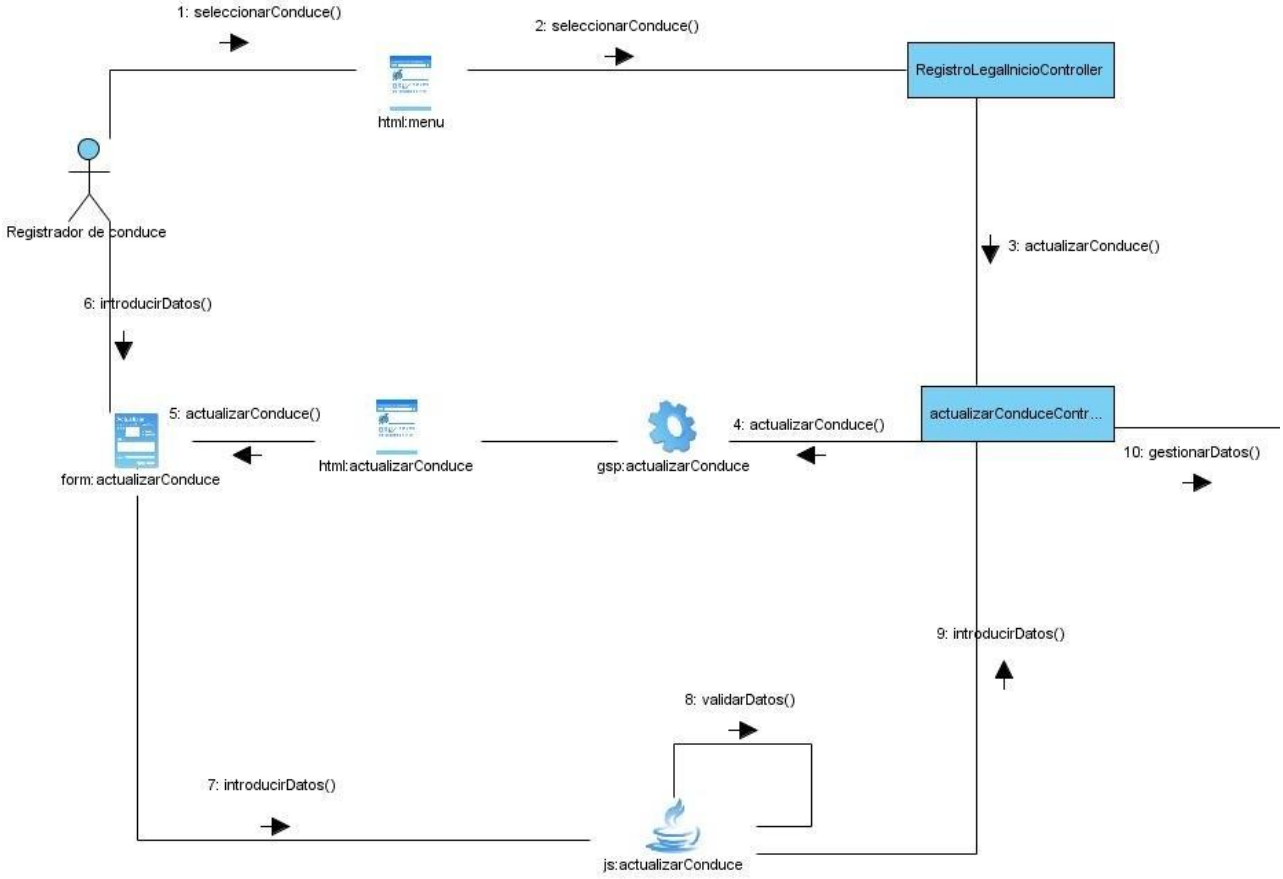

Figura 42: Capa web del diagrama de Comunicación Actualizar Conduce.

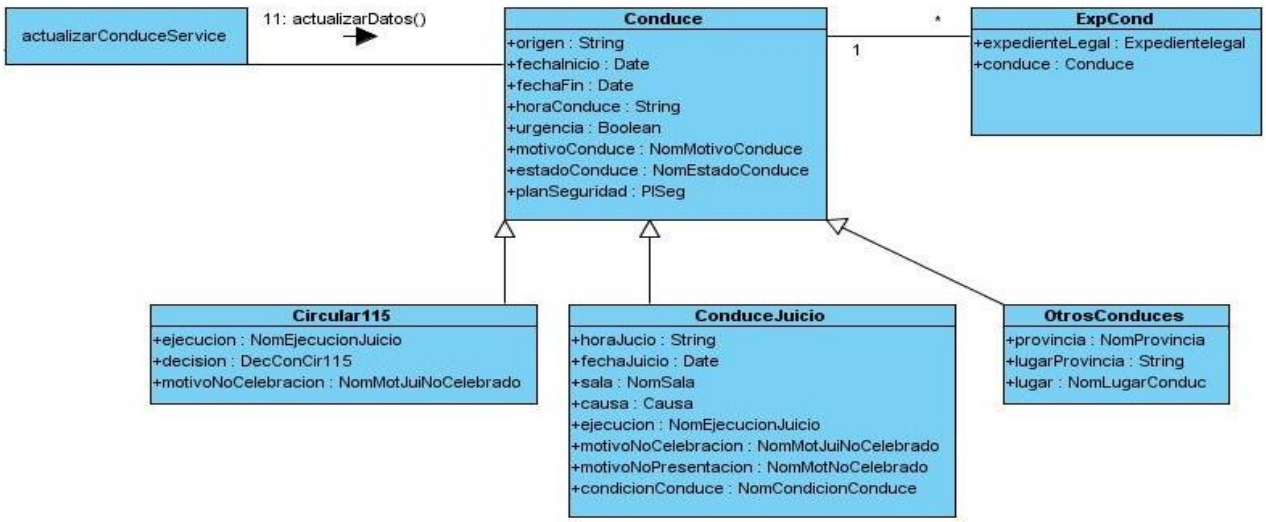

Figura 43: Capa de datos y capa de servicio del diagrama de Comunicación Actualizar Conduce.

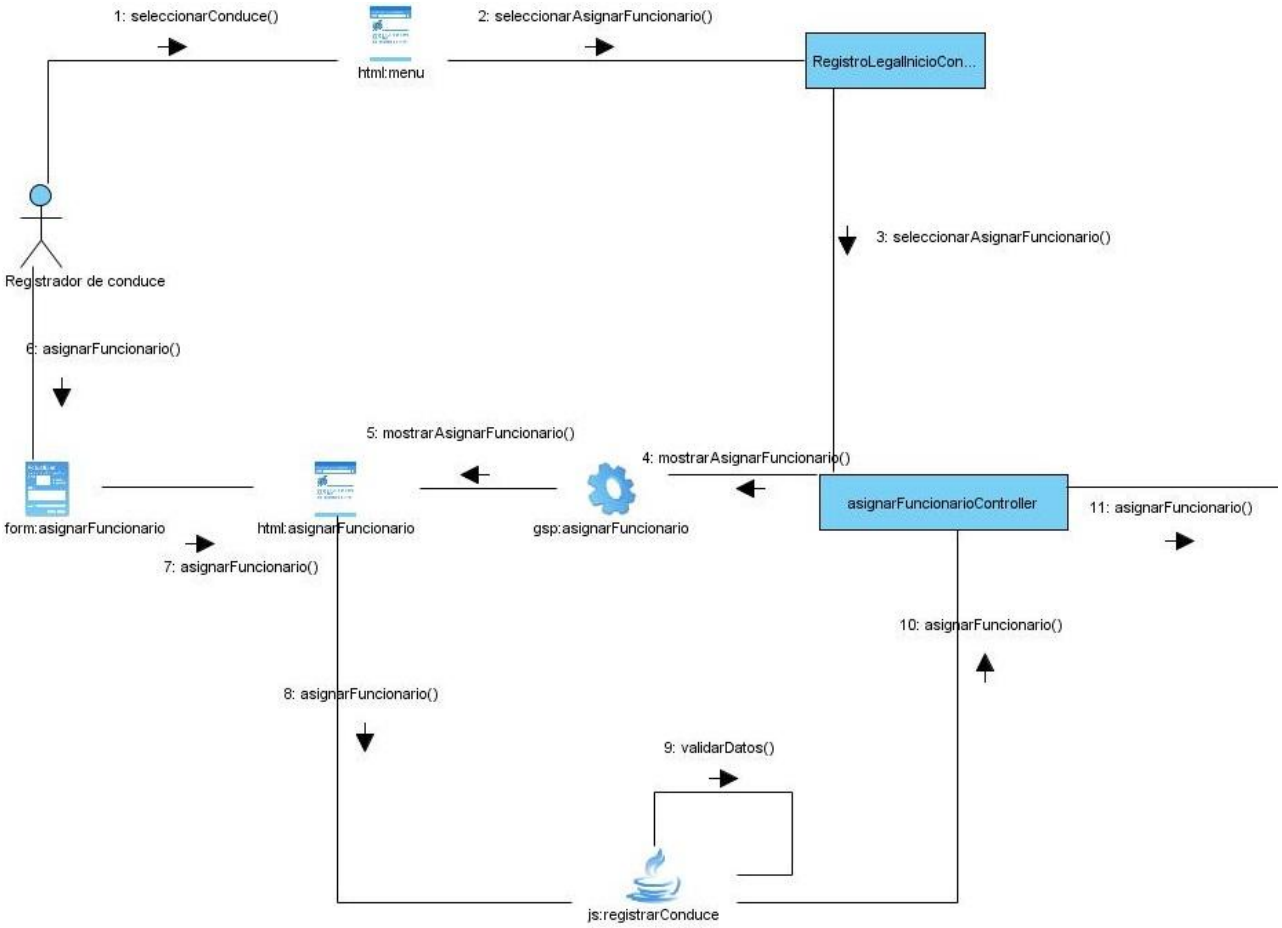

Figura 44: Capa web del diagrama de Comunicación Asignar Funcionarios al Conduce.

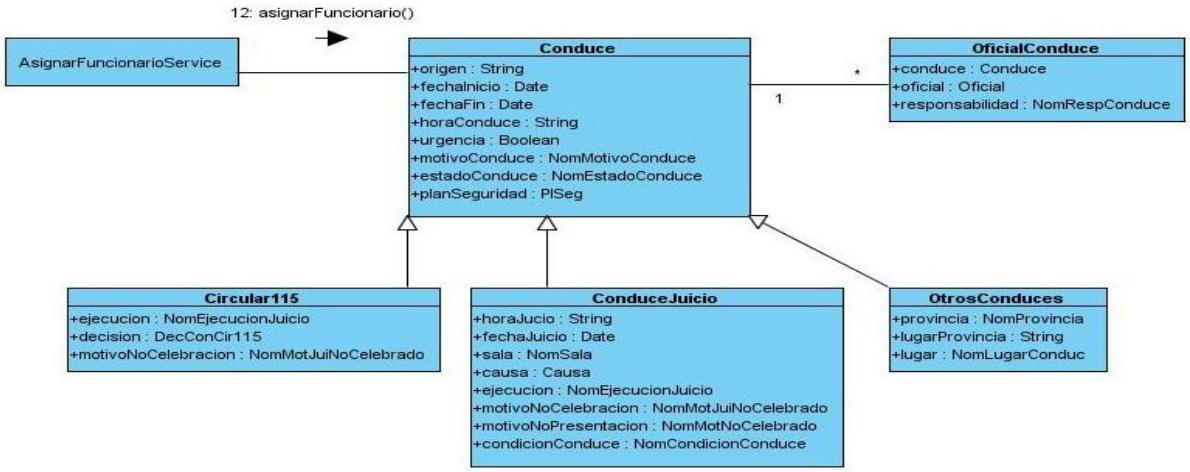

Figura 45: Capa de datos y capa de servicio del diagrama de Comunicación Asignar Funcionarios al Conduce.

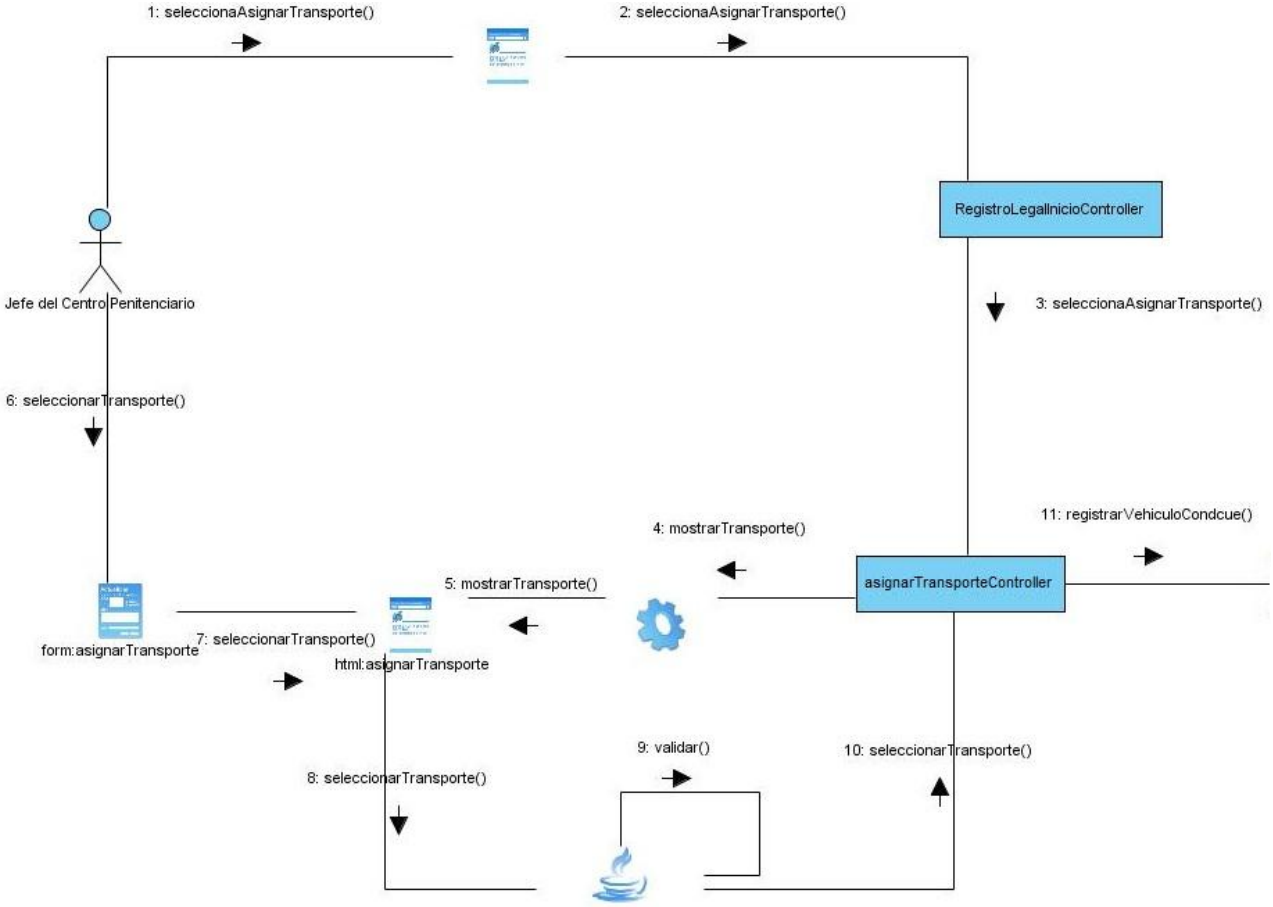

Figura 46: Capa web del diagrama de Comunicación Asignar Transporte al Conduce.

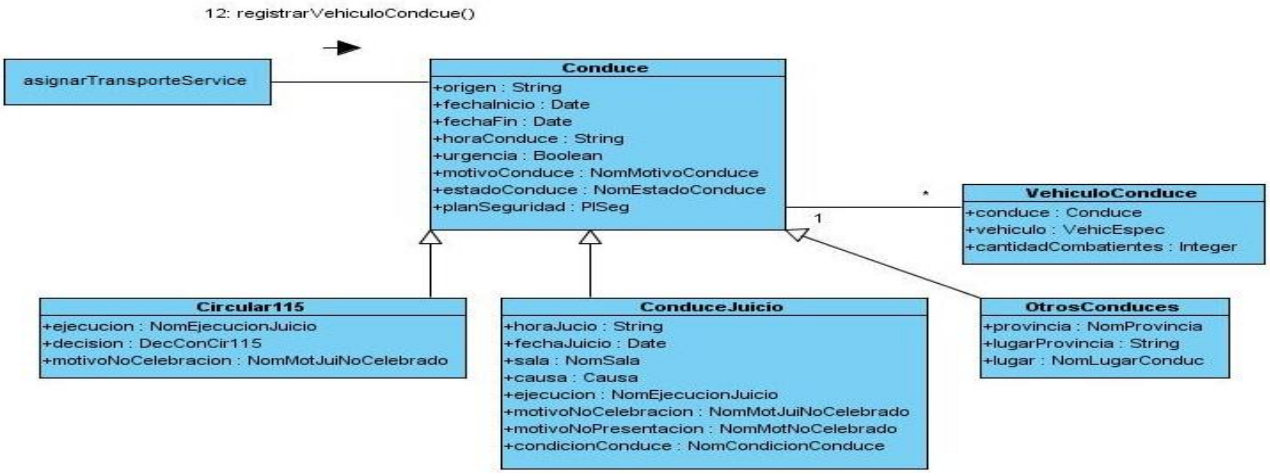

Figura 47: Capa de datos y capa de servicio del diagrama de Comunicación Asignar Transporte al Conduce.

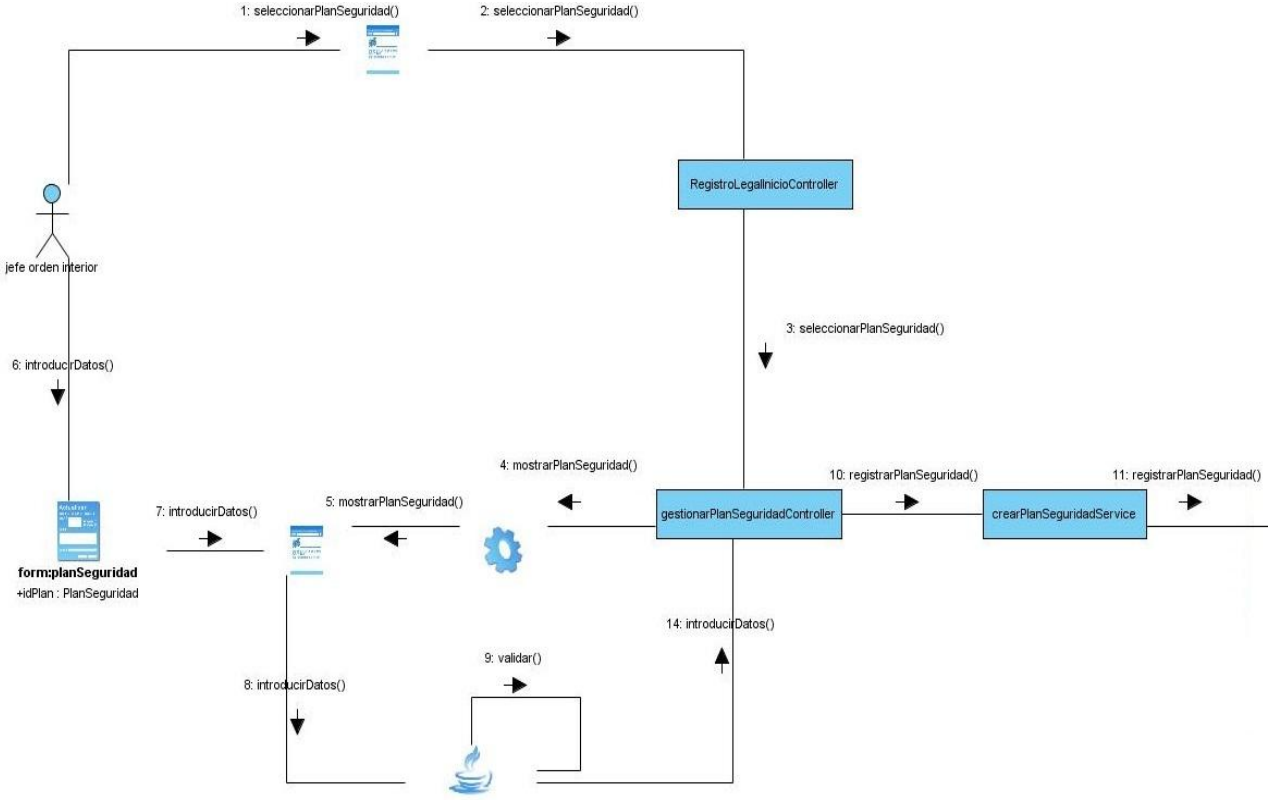

Figura 48: Capa web del Diagrama de Comunicación Crear Plan de Seguridad.

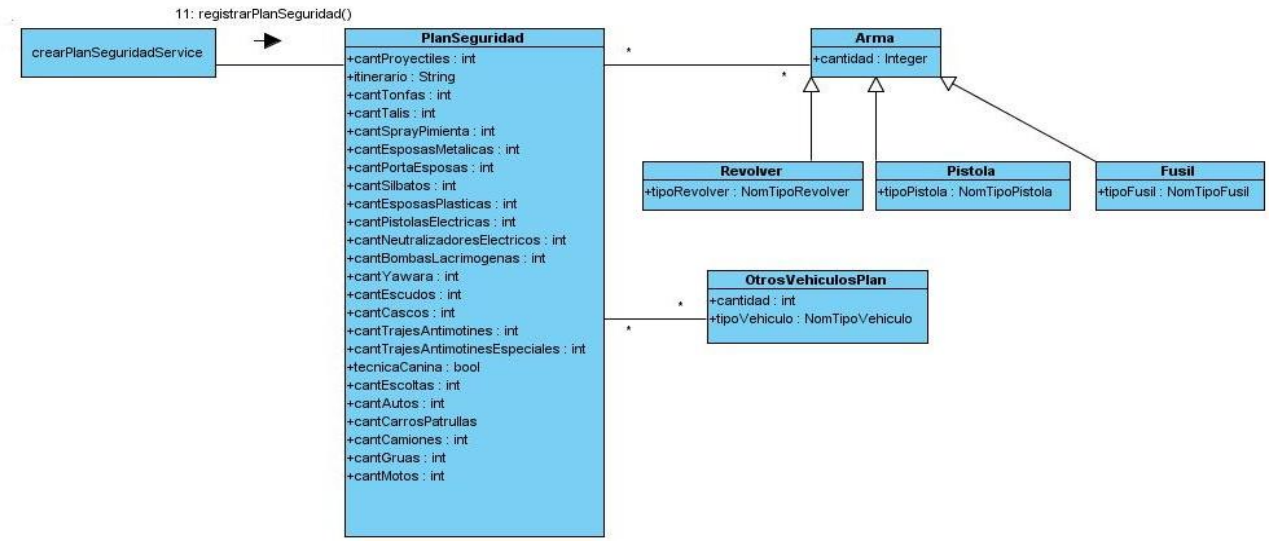

Figura 49: Capa de datos y capa de servicio del diagrama de Comunicación Crear Plan de Seguridad.

Anexo 3 Descripción de las principales clases del módulo

Clase de dominio: planSeguridad	
Atributo	**Tipo**
cantEscoltas	Integer
cantAutos	Integer
cantCarPatrulleros	Integer
cantCamiones	Integer
cantGruas	Integer
cantMotos	Integer
cantProyectiles	Integer
itinerario	String
cantBastonesGoma	Integer
cantTalis	Integer
cantSprayPimiento	Integer
cantEsposasMetalicas	Integer
cantVehiInternos	Integer
cantPortaEsposas	Integer
cantSilbatos	Integer
cantEspoPlasticas	Integer
cantPistElectricas	Integer
cantNeutElectricos	Integer
cantBombasFumigenas	Integer
cantYawara	Integer
cantEscudos	Integer
intcantCascos	Integer
cantTraAntimotin	Integer
cantTrajAnEspeciales	Integer
tecnicaCanina	Boolean

Tabla 17: Descripción de la clase de dominio Plan de Seguridad.

Clase de dominio: conduce	
Atributo	**Tipo**
origen	String
motivoConduce	NomMotivoConduce
fechaInicio	Date
fechaFin	Date
estadoConduce	NomEstadoConduce
hora	DateTime
urgencia	Boolean

Tabla 18: Descripción de la clase de dominio Conduce.

Clase de dominio: circular115	
Atributo	**Tipo**
causa	Integer
annoCausa	Integer
destino	String
provincia	NomProvincia
tribunal	NomTribunal
municipio	NomMunicipio
sala	NomSala
fecha	Date
ejecucionJuicio	NomEjecucion
horaJuicio	DateTime

Tabla 19: Descripción de la clase de dominio Circular115.

Clase de dominio: conduceTribunalCJ	
Atributo	**Tipo**
causa	Integer
motivoNoPresentacion	NomMotivoNoPresentacion
motivoNoCelebracion	NomMotivoNoCelebracion
condicionConduce	NomCondicionConduce
annoCausa	Integer
destino	String
provincia	NomProvincia
tribunal	NomTribunal
municipio	NomMunicipio
sala	NomSala

Tabla 20: Descripción de la clase de dominio Conduce a Tribunal.

Clase de dominio: lugar	
Atributo	**Tipo**
lugar	NomLugar
provincia	NomProvincia
lugarProvincia	NomLugaProvincia

Tabla 21: Descripción de la clase de dominio Conduce a Lugar.

Java Script: Circular115	
Nombre	**Descripción**
gestionar	Envía los datos del formulario de la vista circular115 para ser guardados.
validacion	Este método valida todos los atributos de la vista circular115.

Tabla 22: Descripción delJavaScript Circular115.

Java Script: ConduceTribunalCJ	
Nombre	**Descripción**
validacion	Este método valida todos los atributos de la vista conduceTribunalCJ.
validarDelito	Valida el tipo de delito.
limpiarDelito	Limpia el campo tipo de delito.
adicionarDelito	Envía el campo del tipo de delito al controlador Delito para ser insertado en la tabla delitos.
borrarDelito	Envía el id del tipo de delito en la tabla de delitos al controlador Delito para ser borrado de esta.

Tabla 23: Descripción delJavaScript Conduce a Tribunal.

Java Script: Lugar	
Nombre	**Descripción**
validacion	Este método valida todos los atributos de la vista conduce lugar.
gestionar	Envía los datos del formulario de la vista lugar al controlador OtrosConduces para ser guardados.

Tabla 24: Descripción delJavaScript Conduce a Lugar.

Controlador: Circular115Controller	
Atributo	**Tipo**
circular115Service	Circular115Service
Nombre	**Descripción**
procesarCircular	Se procesan los datos de la vista circular115 y se le envían al serviciocircular115Service para ser guardados
conduce	Este método levanta la vista circular115.

Tabla 25: Descripción del Controlador Circular115.

Controlador: ConduceJuicioController	
Atributo	**Tipo**
conduceJuicioService	ConduceJuicioService
Nombre	**Descripción**
procesarConduceJuicio	Se procesan los datos de la vista conduceTribunalCJy se le envían al servicio ConduceJuicioService para ser guardados.
conduce	Este método levanta la vista conduceTribunalCJ.

Tabla 26: Descripción delControlador Conduce a Juicio.

Controlador: OtrosConducesController	
Atributo	**Tipo**
otrosConduceService	OtrosConduceService
Nombre	**Descripción**
procesarOtrosConduce	Se procesan los datos de la vista lugar y se le envían al servicio OtrosConduceServicepara ser guardados.

conduce	Este método levanta la vista lugar.

Tabla 27: Descripción delControlador Otros Conduces.

Servicio: Circular115Service	
Nombre	**Descripción**
salvar	Se salvan los datos de la vista circular115 en la base de datos.

Tabla 28: Descripción delServicio Circular115.

Servicio: ConduceJuicioService	
Nombre	**Descripción**
salvar	Se salvan los datos de la vista conduceTribunalCJen la base de datos.

Tabla 29: Descripción delServicio Conduce a Juicio.

Servicio: OtrosConducesService	
Nombre	**Descripción**
salvar	Se salvan los datos de la vista lugar en la base de datos.

Tabla 30: Descripción del Servicio Otros Conduce.

Anexo 4 Descripción de los casos de uso

Objetivo	Registrar los datos del conduce de los internos en el sistema.
Actores	Registrador de conduce
Resumen	El Caso de Uso se inicia cuando el Registrador de conduce selecciona el (los) interno(s) a conducir e ingresa los datos del conduce, el sistema registra los datos del conduce del interno y finaliza el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El Registrador de conduce debe estar autenticado en el sistema.
Postcondiciones	Se registran los datos del conduce de un interno. Se registra una solicitud o una decisión en dependencia del motivo de conduce.
Flujo de eventos	
Flujo básico Registrar los datos del conduce	

	Actor	**Sistema**
1.	El Registrador de conduce selecciona la	

	opción “Registrar conduce”.	
2.		Se ejecuta el CU incluido “Buscar interno”.
3.	El Registrador de conduce selecciona el interno a conducir y oprime el botón “Adicionar “.	
4.		El sistema valida que el interno no haya egresado por cualquier motivo. Si el interno egresó, ver el flujo alterno 4a. “Interno egresado”.
5.		El sistema muestra el listado de internos. El listado contiene: • Primer nombre • Segundo nombre • Primer apellido • Segundo apellido • Número de identidad El sistema permite buscar un nuevo interno o eliminar un interno del listado.
6.	El Registrador de conduce oprime el botón “Siguiente”. Si el Registrador de conduce oprime el botón “Nuevo”, regresa al paso 2 del flujo básico. Si el Registrador de conduce oprime el botón “Eliminar” para elimina un interno, ver el flujo alterno 6a. “Eliminar interno”.	
7.		El sistema valida los internos a ser conducidos. Si hay más de un interno en el conduce y uno de ellos tiene VIH/SIDA, ver el flujo alterno 7a. “Interno enfermo de VIH/SIDA”.

		Si hay más de un interno en el conduce y uno de los internos es de Mayor Severidad, ver el flujo alterno 7b. “Interno de Mayor Severidad”. Si hay más de un interno en el conduce y uno de los internos tiene el tipo de sanción “Privación perpetua de libertad” o es de nacionalidad diferente a la cubana, ver el flujo alterno 7c. “Interno de categoría especial”.
8.		El sistema muestra los campos para ingresar los siguientes datos del conduce (Ver **¡Error! No se encuentra el origen de la referencia.**): • Origen o interés • Motivo • Fecha de inicio • Fecha de fin • Hora
9.	El Registrador de conduce introduce los datos solicitados. Si el Registrador de conduce selecciona como motivo de conduce “Al tribunal para la celebración del juicio oral o notificación de alguna resolución judicial”, ver flujo alterno 9a. “Conduce al Tribunal para celebración del juicio”.	
10.		El sistema muestra el campo: • Lugar • Provincia Por defecto la provincia debe ser la provincia donde esté ubicado el interno.