Universidad de las Ciencias Informáticas

Facultad 2

Título: "Desarrollo del módulo Traslados del Sistema Informativo de la Dirección de Establecimientos Penitenciarios"

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

Autor: José Mario Hernández Ronda

Tutor: Ing. Yanay Viera Lorenzo
Co-Tutor: Ing. Guillermo Enrique Ferras Pérez

Ciudad de La Habana

2012

DECLARACIÓN DE AUTORÍA

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

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

José Mario Hernández Ronda.

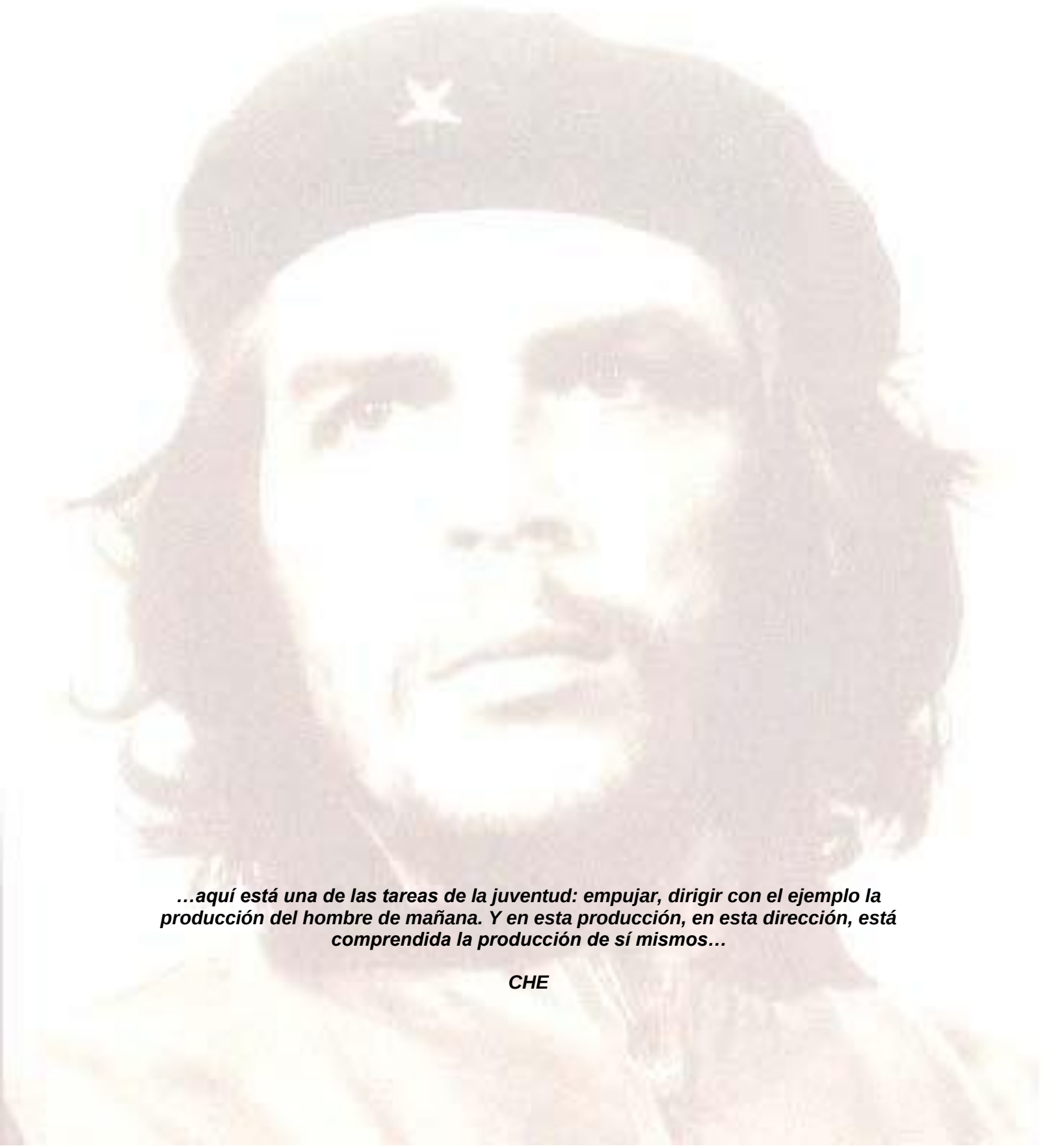
_____

Firma del Autor

Ing. Yanay Viera Lorenzo.

_____

Firma del Tutor



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

CHE

AGRADECIMIENTOS

Agradezco especialmente a mis padres por ser la luz guía en mi vida, por brindarme su amor, dedicación, educación, ejemplo y apoyo, algo de lo que estaré agradecido siempre. A mi tía Gloria, por estar ahí siempre que la necesité. A Beba y Lincoln por apoyarme en los momentos duros. A mi amigos de la 6 y a mi gente de la 2 por estar siempre a mi lado y que cada uno de ellos aportó su granito de arena en el transcurso de mi carrera y en la realización de esta tesis. A esa obra infinita que es la Revolución Cubana por haberme dado la posibilidad de llegar a ser alguien en la vida. En general a todas las personas que me han ayudado y apoyado para lograr este sueño. Gracias a todos ustedes por ayudarme a llegar hasta aquí…

DEDICATORIA

A mis padres por estar siempre apoyándome, por ser un ejemplo para mí. A mi hermano por ser mi inspiración para llegar hasta aquí. A la memoria de los míos que ya no están, que estoy seguro de que estarían orgullosos al verme ahora.

RESUMEN

Como parte del desarrollo del Sistema Penitenciario Cubano se decide crear el Sistema Informativo de la Dirección de Establecimientos Penitenciarios, que es el encargado de informatizar los procesos del Sistema Penitenciario Cubano. Dentro de estos procesos se encuentra toda la gestión referente a los traslados de los internos la cual no está informatizada completamente, es por eso que se decide desarrollar el módulo Traslados. Para esto se diseñó el módulo elaborando los diagramas de clases y de secuencia atendiendo a la arquitectura definida y haciendo uso de los patrones de diseño. Además se construyó el diagrama de componentes, obteniéndose el código de la aplicación y el diagrama de despliegue. Finalmente se realizaron las pruebas de calidad en 3 iteraciones que arrojaron no conformidades las cuales fueron resueltas. Lo antes expuesto se realizó utilizando la metodología RUP, plataforma de desarrollo Java y el marco de trabajo Grails.

Palabras claves: interno, sistema penitenciario cubano, traslado

INDICE

INTRODUCCIÓN

Con el desarrollo actual de las tecnologías y la consolidación de la era de la información, en Cuba se han tomado medidas para informatizar varios sectores de la sociedad. Entre los sectores informatizados se encuentra el Ministerio del Interior (MININT) debido a su vital importancia para asegurar la seguridad ciudadana, y dentro de éste se encuentra el órgano Prisiones, cuyo proceso de automatización comenzó alrededor del año 1989 con un sistema informático para el registro y control de la legalidad de los acusados, sancionados y asegurados. Este sistema permitía gestionar los datos del recluso[1], los procesos penales y otros datos del sistema de trabajo en vistas a lograr la reinserción de los reclusos en la sociedad. A medida que se fue perfeccionando el sistema de control y en cumplimiento de la orden 43/99 del Viceministro Primero del Interior, en el año 2002 se desarrolla el Sistema para el Control de Reclusos (SACORE), utilizando la tecnología existente en ese entonces, pero su implantación no es hasta el año 2003.

SACORE brinda ventajas para la gestión de la información del trabajo en el Sistema Penitenciario, pero no es suficiente pues le faltan áreas por informatizar como son los equipos multidisciplinarios y servicios médicos. También se utiliza el Sistema Automatizado de Capacidades de la Dirección de Establecimientos Penitenciarios (SACDEP) y el Sistema de Automatización de Incidencias de la Dirección de Establecimientos Penitenciarios (SAIDEP), los cuales son también insuficientes para gestionar eficientemente todo el sistema penitenciario, ya que su utilización es un poco engorrosa, no hay integración entre ellos y no cubren todas las funcionalidades del área en la que se emplean. Por la necesidad de integrar los tres sistemas con el objetivo de optimizar los procesos y la modernización del sistema informático en el sistema penitenciario, se desarrolla en la Universidad de las Ciencias Informáticas el Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

Este sistema informático está dividido en 7 subsistemas en correspondencia a los procesos de trabajo que se realizan en las áreas de registro legal, tratamiento y seguridad de los acusados, asegurados y sancionados en los establecimientos penitenciarios.

[1] Actualmente: interno.

El subsistema Registro Legal está compuesto por varios módulos, entre los que se encuentra el módulo Traslados que es el encargado de gestionar la planificación y ejecución de los traslados inter-unidades o interprovinciales, registrando la salida y la llegada al destino.

Actualmente SACORE gestiona el traslado de internos de un centro a otro, pero aún carece de funcionalidades importantes para la completa planificación y ejecución de los traslados. Algunos de los problemas presentes son:

- No se planifican los traslados con los funcionarios y vehículos necesarios para ejecutar el traslado, que dificulta el control de los recursos y la correcta ejecución del traslado.
- No se le da seguimiento al Plan de Aseguramiento de los traslados que lo requieren, puede provocar que no se aseguren los traslados de internos con cierto nivel de peligrosidad dando lugar a escenarios propicios para la ocurrencia de incidencias.
- No se controla el vencimiento de los traslados según el período de tiempo establecido para su ejecución.
- No se controlan los traslados que son prorrogados para nuevas fechas de ejecución sino que se tratan como nuevos traslados en el sistema.

Por lo antes planteado surge como **problema** para la presente investigación: ¿Cómo contribuir a la informatización de los procesos de traslados en el Sistema Penitenciario Cubano?

Definiendo a su vez como **objeto de estudio**: la informatización de los Sistemas Penitenciarios.

Se plantea como **objetivo general:** Desarrollar el módulo Traslados del SIDEP para contribuir a la informatización de los procesos de traslados del Sistema Penitenciario Cubano y queda definido como **campo de acción** la informatización de los procesos de traslados en el Sistema Penitenciario Cubano.

Del objetivo general se desglosan los siguientes **objetivos específicos**:

1. Diseñar el módulo Traslados del SIDEP.
2. Implementar el módulo Traslados del SIDEP.

En aras de dar cumplimiento a los objetivos planteados se trazan las siguientes **tareas de la investigación**:

1. Análisis de las soluciones informáticas nacionales o internacionales con el mismo perfil.
2. Descripción de las herramientas y tecnologías para dar solución al problema.

3. Diseño de los diagramas de clases del módulo Traslados.
4. Elaboración de los diagramas de interacción del módulo Traslados.
5. Diseño de la base de datos del módulo Traslados.
6. Elaboración de los diagramas de componentes del módulo Traslados.
7. Implementación del módulo Traslados.
8. Elaboración del diagrama de despliegue del módulo Traslados.
9. Diseño de los casos de prueba para el módulo Traslados.
10. Aplicación de las pruebas al módulo Traslados.

El documento de tesis consta de los siguientes capítulos:

Capítulo 1: Fundamentación Teórica

En este capítulo se presenta un estudio referente a otras soluciones informáticas existentes relacionadas con el control de los traslados en centros penitenciarios tanto a nivel nacional como internacional. Se describirán, además, las herramientas, tecnologías y metodología definidas para el SIDEP.

Capítulo 2: Diseño

En este capítulo se abordarán las características del sistema, los patrones de diseño que se utilizarán y las funcionalidades a informatizar en el módulo Traslados. A partir de estos elementos, se construirán los diagramas de clases e interacción asociados a las funcionalidades y el diseño de la base de datos para la persistencia de la información.

Capítulo 3: Implementación y Prueba

En este capítulo se construirán los diagramas de componentes, se expondrán segmentos de códigos que demuestran la implementación del módulo Traslados y elabora el modelo de despliegue. Para concluir el capítulo, se define una estrategia de pruebas a aplicar al módulo Traslados y se expondrán los resultados de la aplicación de dichas pruebas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se presenta un estudio referente a otras soluciones informáticas existentes relacionadas con la planificación y ejecución de los traslados en centros penitenciarios tanto a nivel nacional como internacional. Se describirán, además, las herramientas, tecnologías y la metodología definidas para el SIDEP.

1.2 Estado del arte de los sistemas penitenciarios

La investigación realizada a los sistemas informáticos existentes, mostró que estos sistemas cuentan con la posibilidad de identificar a cada uno de los individuos que ingresan al sistema penitenciario, registrando sus datos personales, rasgos físicos y señas particulares, entre otros datos de interés. El análisis de cada uno de los sistemas que a continuación se mencionarán, parte de las funcionalidades que brindan.

1.2.1 Sistema Automatizado para el Control del Recluso

El Sistema Automatizado para el Control del Recluso (SACORE) (1), surge para dar cumplimiento a la Orden 43/99 del Vice-Ministro Primero del Ministerio del Interior de Cuba y tiene las siguientes características:

- Garantiza respuestas inmediatas a las solicitudes de información de los diferentes órganos e instituciones del estado como son: Jefatura del MININT, Ministerio de Justicia, Tribunales, Fiscalías, MINED, INDER, FMC, MINFAR.
- Recoge prácticamente la totalidad de la información de los reclusos en todas las especialidades.
- Tiene más de 200 reportes impresos.
- Permite la recuperación dinámica a partir de una solicitud de búsqueda.
- Los partes que se emiten son obtenidos de forma automatizada.
- Permite el traslado automático de todos los datos del recluso al nivel nacional.

El SACORE permite registrar traslados, de los cuales registra los internos que van a ser trasladados con la provincia y el centro de destino, pero esta gestión presenta deficiencias como:

- No se registra un Plan de Seguridad.

- No se registra un oficial responsable.
- No se especifica el tipo de traslado ni de ejecución.
- No se asignan los vehículos de transporte para los traslados.
- No se tiene en cuenta la etapa de planificación del traslado ni de ejecución.

El SACORE no tiene en cuenta la decisión tomada para llevar a cabo el traslado en caso de la aprobación de esta, así como la prórroga de los traslados una vez pasado el tiempo asignado para su ejecución. Por otra parte, la introducción de nuevas tecnologías no han generado todas las transformaciones funcionales y organizativas en los sistemas de trabajo y aún no es óptimo el aprovechamiento de éstas.

Las funcionalidades presentes en el SACORE para la ejecución de los traslados no cubren todas las necesidades actuales de los centros penitenciarios por lo que no es una solución factible para dar solución al problema planteado en la presente investigación.

1.2.2 Sistema de Gestión Penitenciaria

El Sistema de Gestión Penitenciario Venezolano (SIGEP) desarrollado a partir del año 2006 en la Universidad de las Ciencias Informáticas, da respuesta a las necesidades de gestión, información y apoyo a la toma de decisiones de la Dirección General de Custodia y Rehabilitación del Recluso (DGCRR). Este sistema gestiona la planificación de traslados interpenales a partir de órdenes de los tribunales o de la dirección general. También controla la ejecución de los traslados, la salida y la llegada al centro de destino. Este sistema no es factible para su utilización en nuestro país debido a que las decisiones no provienen del mismo nivel de mando de prisiones. El traslado es planificado y se le asignan los oficiales responsables del mismo en el momento de su ejecución, y al ser otro sistema penitenciario, los traslados de forma general se manejan diferente a como lo hace el Sistema Penitenciario Cubano.

1.3 Metodología, herramientas, tecnologías y lenguajes

La metodología, herramientas, tecnologías y lenguajes que se utilizarán en el desarrollo del módulo Traslados fueron definidos por el equipo de arquitectura del SIDEP.

1.3.1 Metodología

Como metodología para el desarrollo del software se decide utilizar Rational Unified Process (RUP), por ser un proceso:

- Iterativo e incremental, lo cual permite dividir el proyecto en pequeños subproyectos para desarrollarlo en distintas etapas e iteraciones que resultan en un incremento del producto.
- Dirigido por casos de uso, el cual es uno de los métodos más utilizados y efectivos para reflejar los requisitos. Estos no solo sirven para especificar los requisitos, ellos son los encargados de guiar el ciclo de vida del proyecto.
- Centrado en la arquitectura, lo cual permite organizar o estructurar el sistema en sus partes más relevantes e ir refinando esta estructura progresivamente. RUP define “un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto” (2).

RUP propone flujos de trabajo en los que se definen las secuencias de actividades, quiénes las deben desarrollar y los artefactos a generar. La figura 1 muestra las fases de RUP y los flujos de trabajo que contiene.

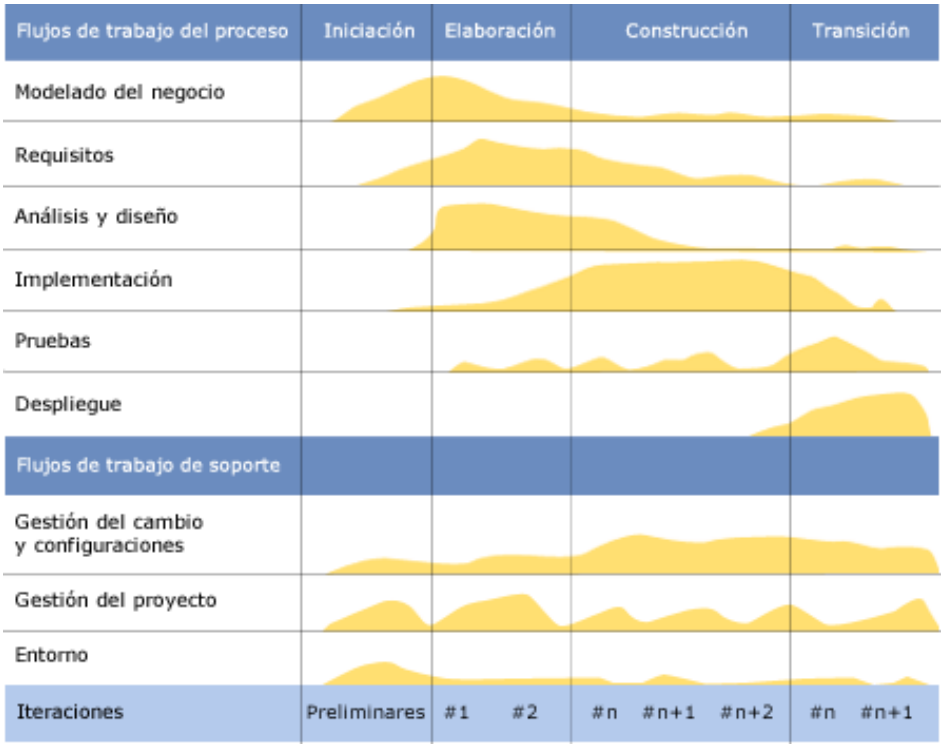


Figura 1 Ciclo de vida de RUP

Fases

- Inicio: En esta etapa se describe el negocio, síntesis de arquitectura posible y el alcance del proyecto.

- Elaboración: Se define la línea base de la arquitectura y una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos.
- Construcción: Se obtiene un producto documentado, listo para su uso, se obtienen uno o varias versiones que han pasado por pruebas.
- Transición: Se obtiene un producto final listo para su uso. Puede implicar reparación de errores.

Flujos de Trabajo

- Modelación del negocio: Se describen y se detallan las necesidades del negocio.
- Requerimientos: Se traducen las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Los requerimientos se trasladan dentro de la arquitectura de software.
- Implementación: Se comienza a crear el software ajustándose a la arquitectura para que tenga el comportamiento esperado.
- Instalación: Se producen varias versiones del producto y se realizan actividades.
- Prueba: Se verifica si el comportamiento logrado es correcto y si el desarrollo ha sido acorde a la arquitectura.
- Administración del proyecto: Se administran horarios y recursos.
- Administración de Configuración y Cambios: Se controlan las versiones del proyecto.
- Ambiente: Actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto.

En el desarrollo de este trabajo se generan artefactos pertenecientes a los flujos de trabajo Análisis y Diseño (abarca solo las actividades del diseño), Implementación y Pruebas.

1.3.2 Lenguaje de Modelado

UML 6.4 es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema (3). UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para

documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software pero no especifica en sí mismo qué metodología o proceso usar.

1.3.3 Herramienta CASE

1.3.3.1 Visual Paradigm 3.4

Visual Paradigm es una herramienta CASE concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Dentro de sus características se aprecia que soporta UML. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (4). Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux)
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación
- Capacidades de ingeniería directa e inversa
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad

Actualmente la Universidad de las Ciencias Informáticas cuenta con la licencia para el uso de la herramienta. Por estas características es que se utiliza Visual Paradigm 3.4 for UML 6.4 Enterprise Edition como herramienta de modelado de software para la construcción de los diagramas del diseño y la implementación.

1.3.3.2 Embarcadero ER/Estudio 8.0

ER/Studio es una herramienta que modela los datos, se usa para el diseño y la construcción lógica y física de base de datos. Su ambiente es de gran alcance, de varios niveles del diseño.

ER/Studio ofrece las siguientes facilidades:

- Capacidad fuerte en el diseño lógico
- Sincronización bidireccional de los diseños lógico y físico

- Construcción automática de Base de Datos
- Reingeniería inversa de Base de Datos
- Documentación basada en HTML
- Un Repositorio para el modelado

Esta herramienta es utilizada para todo el diseño de la base de datos del módulo Traslados.

1.3.4 Plataforma de Desarrollo

Java Platform Enterprise Edition (JEE) es una plataforma de programación para desarrollar y ejecutar en software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

1.3.5 Lenguajes de Desarrollo

1.3.5.1 Groovy 1.7.8

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. La especificación JSR 241 se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java (5).

Groovy usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. El bytecode generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java. Groovy puede usarse también de manera dinámica como un lenguaje de scripting.

1.3.5.2 JavaScript

JavaScript es un lenguaje de programación interpretado. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web, permitiendo crear efectos dinámicos en las páginas. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). JavaScript es una de las múltiples aplicaciones que han surgido para extender las capacidades

del lenguaje HTML. Este permite trabajar con los contenidos dentro del documento lo cual facilita realizar validaciones y controlar los eventos que se ejecuten (6).

1.3.6 Tecnología Utilizada

1.3.6.1 Ajax

Ajax es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

1.3.7 Frameworks Utilizados

Un framework es un marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes software para el desarrollo rápido de aplicaciones (7). Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.3.7.1 Grails 1.3.7

Grails es un framework para aplicaciones web libre desarrollado sobre el lenguaje de programación Groovy. Está construido sobre cinco fuertes pilares: Groovy para la creación de propiedades y métodos dinámicos en los objetos de la aplicación, Spring para los flujos de trabajo e inyección de dependencia, Hibernate para la persistencia de los datos, SiteMesh para la composición de las vistas y Ant para la gestión del proceso de desarrollo (8). En la figura 2 se muestra gráficamente la estructura de Grails.

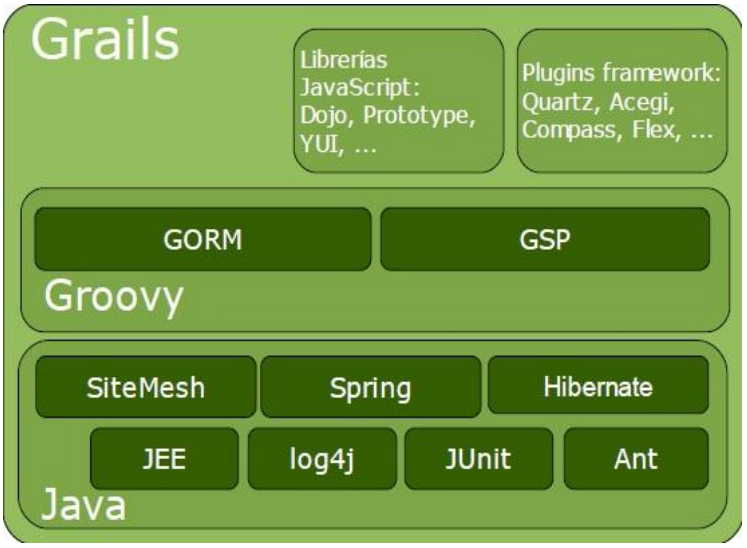


Figura 2 Estructura de Grails

Grails sigue los principios Don't Repeat Your Self (No te repitas) y Convention over Configuration (Convención sobre configuración). Grails es algo más que un framework MVC, también ofrece capa de persistencia, capa de servicio, contenedor de servlets y gestor de bases de datos.

1.3.7.2 Dojo 1.5

Dojo es un framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de Interfaz de Usuario (UI), drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX (9).

Dojo resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos.

1.3.8 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios (10). Un entorno de desarrollo integrado es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs

pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

1.3.8.1 NetBeans 6.9

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos (10). Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

1.3.9 Servidor de Aplicaciones y Gestor de Base de Datos

1.3.9.1 Apache Tomcat 6.0.25

Apache Tomcat funciona como un contenedor web escrito en java, por lo que funciona en cualquier sistema operativo que disponga de una máquina virtual Java y desarrollado en un ambiente participativo y abierto. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP). Es usado en numerosas aplicaciones web de gran escala y críticas en diversas industrias y organizaciones que se referencian en su sitio oficial (11).

1.3.9.2 Oracle 11g EE

Oracle es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

Ayuda a administrar y almacenar grandes volúmenes de datos, se caracteriza por su estabilidad y escalabilidad, es multiplataforma, constituye una herramienta de administración gráfica cómoda de utilizar, ayuda al análisis de la información.

1.4 Conclusiones parciales

En este capítulo se analizaron los Sistemas de Gestión Penitenciarios SACORE y SIGEP que actualmente se utilizan para apoyar la gestión de los Sistemas Penitenciarios y se da a conocer las funcionalidades y deficiencias de los mismos. A partir de este análisis se llegó a la conclusión de que no solucionan la problemática planteada.

Se expusieron las características de las tecnologías y herramientas definidas para el SIDEP, teniendo como principales recursos las tecnologías libres, eslabón indispensable en las nuevas concepciones de informatización que se aplican en el país. Se definió la utilización de la metodología RUP para el desarrollo de software, con el lenguaje de modelado UML y utilizando las herramientas CASE Visual Paradigm y Embarcadero ER/Estudio para el modelado de los datos, la plataforma de desarrollo a utilizar es JEE, utilizando los lenguajes de programación Groovy y JavaScript, se emplea también la tecnología Ajax utilizando los frameworks Grails 1.3.5 y Dojo 1.5.0, como entorno de desarrollo integrado se utiliza NetBeans 6.9 y como gestor de base de datos Oracle 11g EE.

CAPÍTULO 2: DISEÑO

2.1 Introducción

En este capítulo se abordarán casos de uso definidos para el módulo Traslados, los patrones de diseño que se utilizarán y la arquitectura definida para el SIDEP. A partir de estos elementos se realizan los diagramas de clases del diseño y los diagramas de secuencia asociados a los casos de uso. Se realiza, además, el diseño de la base de datos para la persistencia de la información.

2.2 Resumen de los casos de uso del módulo Traslados

El módulo tiene un total de 10 casos de uso, los cuales se describen a continuación:

Tabla 1: Descripción de las funcionalidades del módulo Traslado.

Caso de Uso	Nombre	Descripción
1	Planificar traslado	El Caso de Uso permite registrar traslados a partir de las decisiones aprobadas permitiendo ingresar los datos necesarios y asignándole el estado Aprobado al traslado.
2	Asignar funcionarios al traslado	El Caso de Uso permite asignar todos los funcionarios que se destinarán para trasladar los internos y el sistema registra los datos.
3	Buscar Traslado	El sistema muestra los criterios de búsqueda para buscar los traslados que pudiera tener el interno en el sistema, el Usuario introduce los datos y el sistema brinda la posibilidad de seleccionar un traslado.
4	Registrar ejecución de traslado	El sistema muestra los traslados aprobados para registrar si fue ejecutado o no y en caso de no ejecutarse, el motivo por el cual no se ejecutó. El sistema registra el estado.
5	Consultar traslados	El sistema muestra los traslados por los criterios seleccionados por el Consultante del traslado y permite visualizar los detalles.
6	CRUD-D[2] Plan de seguridad	El Caso de Uso permite crear, consultar o actualizar el Plan

[2] CRUD-D: patrón de caso de uso CRUD parcial.

		de Seguridad del traslado, los crea o actualiza y el sistema almacena la información.
7	Asignar vehículos al traslado	El Caso de Uso permite asignar los vehículos que se destinarán para trasladar los internos, los asigna y el sistema registra los datos.
8	Actualizar estado del traslado a vencido	El Caso de Uso se inicia cuando se arribe a la fecha máxima para ejecutar el traslado y aún no se haya planificado, el sistema asigna al estado del traslado el valor de “Vencido”, el sistema registra los datos.
9	Gestionar Plan de traslados	El Caso de Uso permite crear o actualizar el Plan de traslados anual para todo el Sistema Penitenciario Cubano. El usuario crea el plan anual pudiendo actualizarlo en el transcurso del año.
10	Consultar planes pasados	El sistema muestra los campos para consultar un Plan de traslados de otro año distinto al vigente. El usuario selecciona un año y el sistema muestra el Plan del año.

2.3 Arquitectura del Sistema

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software. Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para la guia en el desarrollo de software dentro de un sistema informático (13).

Estas arquitecturas están definidas muchas veces por el tipo de tecnología a la cual se enfrenta un programador o grupo de programadores, por lo cual algunos tipos de arquitectura son más recomendables que otras para ciertas tecnologías.

2.3.1 Arquitectura Cliente-Servidor

La arquitectura cliente servidor se encuentra dentro de la clasificación de estilo llamada-retorno (Ver Figura 3). El cliente y el servidor generalmente están localizados en diferentes sistemas, sin embargo pueden encontrarse en el mismo. El cliente es la entidad que hace la petición por un servicio a uno o varios servidores, el cliente también es el encargado de mantener y procesar todo el diálogo con el usuario. El servidor es la entidad que provee el servicio correspondiente a la

petición, es decir, que responde las peticiones de los clientes. El servicio debe procurar el resultado, el cual es retornado. Las peticiones se llevan a cabo mediante un conjunto de elementos basados en hardware y software que permite establecer un enlace entre los clientes y los servidores.

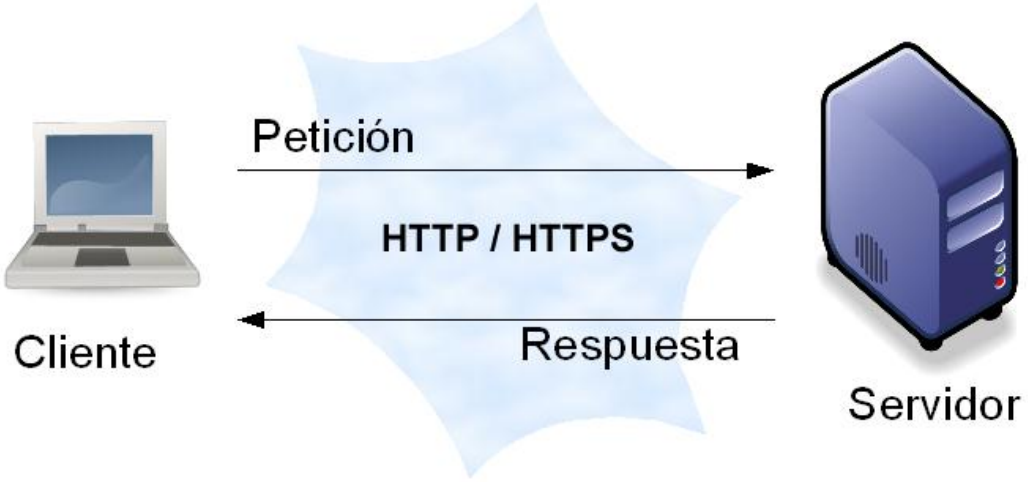


Figura 3: Arquitectura Cliente-Servidor

2.3.2 Arquitectura Modelo Vista Controlador (MVC)

Grails se basa en la arquitectura MVC, en el cuál se aísla la lógica del dominio de la presentación de la información, esta arquitectura establece que los componentes del sistema deben organizarse en 3 capas distintas dependiendo de su función (ver figura 4). La estructura de cada capa se explica a continuación:

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera, se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.
- **Vista:** Esta capa presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Esta capa responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

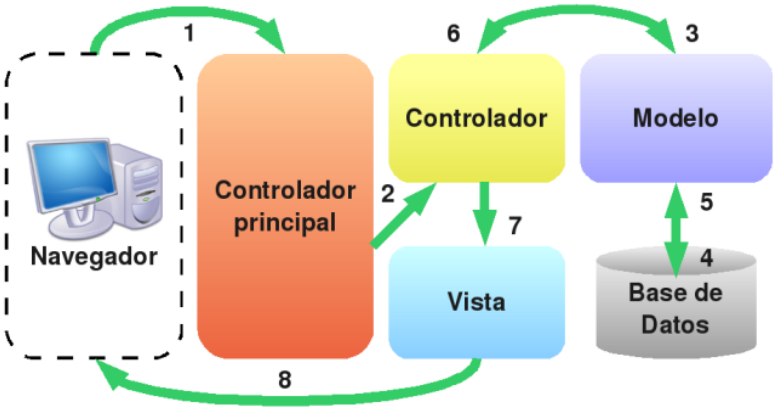


Figura 4: Arquitectura Modelo Vista Controlador (MVC)

Grails propone una cuarta capa que se describe a continuación (ver figura 5):

- **Servicios:** Esta capa contiene los componentes encargados de implementar la lógica del negocio sobre el modelo de datos, que sirve de apoyo al controlador para gestionar toda la lógica del negocio.

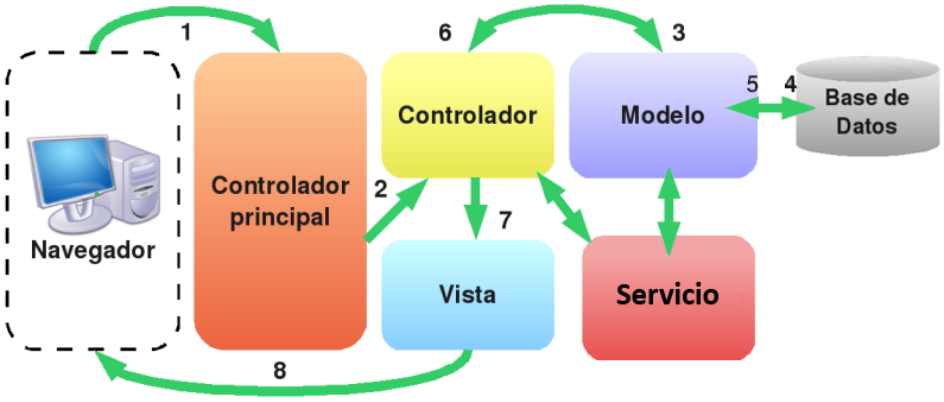


Figura 5: Arquitectura en capas de Grails.

2.4 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan (14). Para el desarrollo del módulo, el marco de trabajo mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad, ya que el mismo framework los implementa. Dentro de los patrones implementados se encuentran:

2.4.1 Patrones GRASP

Los patrones GRASP son una serie de buenas prácticas de aplicación recomendable en el diseño de software para la asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns".

Los patrones GRASP utilizados en el diseño del sistema son los siguientes:

2.4.1.1 Controlador

Todas las peticiones Web son manejadas por un solo controlador, que es el punto de entrada único de toda la aplicación en un entorno determinado. Dicho controlador utiliza el sistema de enrutamiento para asociar el nombre de una acción con la URL solicitada por el usuario. Este patrón se pone de manifiesto en los controladores de la aplicación, donde cada uno es responsable de controlar el flujo de información del caso de uso al cual pertenece.

2.4.1.2 Experto

Asigna una responsabilidad a la clase experta en la información que contiene. Se evidencia en la abstracción del modelo de datos. Cada una de estas clases son expertas en la información que maneja ya que contienen lo básico y esencial para garantizar la información necesaria cuando se solicite.

2.4.1.3 Alta Cohesión

La asignación de responsabilidades con alta cohesión se evidencia por ejemplo en los servicios, los cuales son los encargados de colaborar con otras clases para realizar diferentes operaciones y crear objetos, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.

2.4.1.4 Bajo Acoplamiento

Se evidencia en la poca dependencia entre las clases, con el objetivo de disminuir el nivel de complejidad del sistema así como el trabajo a realizar para llevar a cabo alguna modificación sobre alguna de ellas. Este patrón proporciona una mejor escalabilidad del sistema.

2.4.2 Inversión de Control (IoC)

La Inversión de Control es un patrón que se utiliza para proporcionar una menor dependencia entre los componentes utilizados en la aplicación y fomentar el reúso de los mismos. Manteniendo

estos componentes lo más sencillos posibles facilitando el mantenimiento y la comprensión del sistema.

2.4.3 Singleton

Este patrón garantiza una única instancia para una clase y el acceso global a dicha instancia. La utilización de este patrón se evidencia en los servicios, los cuales son instanciados por los controladores para utilizar las funcionalidades que los mismos brindan. El acceso a ellos es global, lo que significa que todos los controladores acceden a la misma instancia del servicio, accediendo a la misma información contenida en ellos.

2.5 Diagrama de Clases

Los Diagramas de Clases son los diagramas principales de diseño para un sistema, presentan las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. Durante el diseño, se utilizan para satisfacer los detalles de las implementaciones.

2.5.1 Descripción de las clases

A continuación se describen algunas de las clases más significativas del módulo Traslados.

Tabla 2: Descripción de la clase Traslado.

Nombre	Traslado
Tipo de Clase	Clase del dominio
Atributo	**Tipo**
decision	DecTraslado
ejecutado	Boolean
estadoTraslado	NomEstadoTraslado
horaEjecucion	String
fechaEjecucion	Date
plan	PlSeg

Tabla 3: Descripción de la clase TrasladoService.

Nombre	TrasladosService
Tipo de Clase	Servicio
Atributo	**Tipo**
traslado	Traslado
registroLegalService	RegistroLegalService
listInternosService	ListInternosService
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
listarInternos (Object params)	Adiciona a la tabla de internos el nuevo que se le pasa por parámetros.
usuarioOnline()	Devuelve el usuario que está registrado en el sistema.
salvar(Traslado t)	Salva en la base de datos el traslado que se le pasa por parámetros.

Las descripciones de las demás clases podrán consultarse en el Anexo 1.

2.5.2 Diagrama de clases de diseño con estereotipos web

A continuación se muestra el diagrama de clases del diseño con estereotipos web el caso de uso Planificar Traslado, porque es un caso de uso de complejidad alta y prioridad crítica.

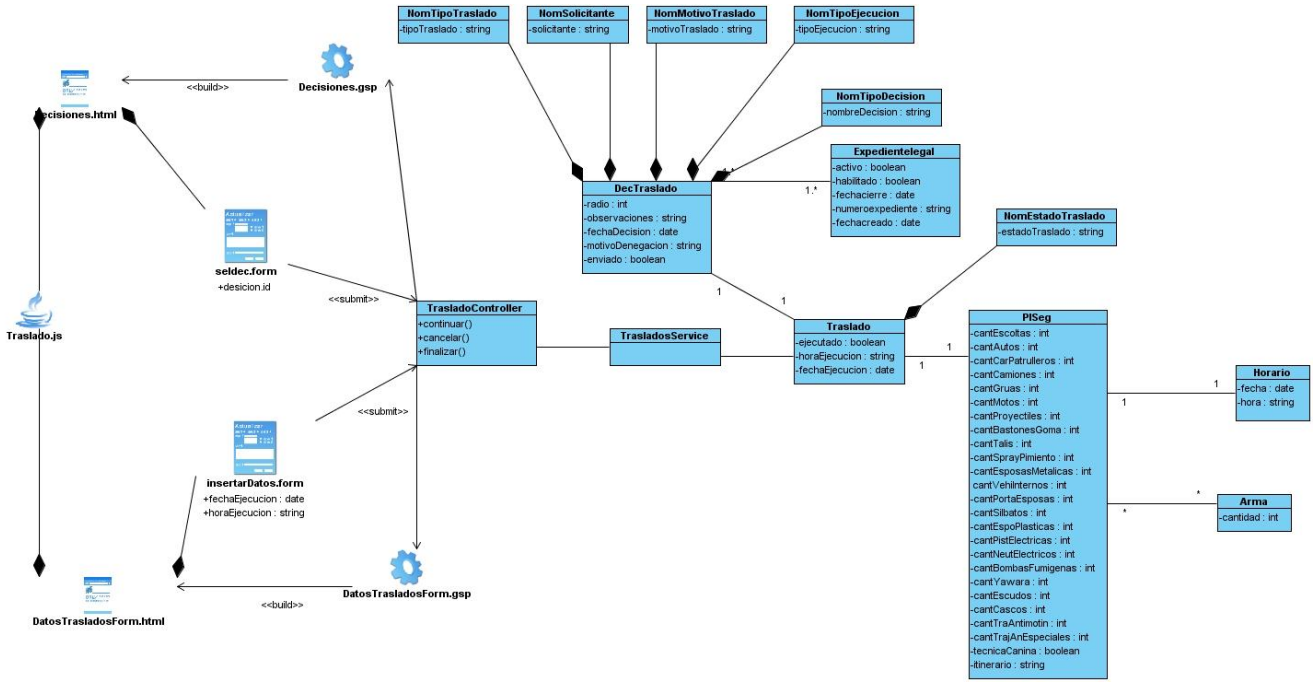


Figura 6: Diagrama de clases con estereotipos web del CU Planificar Traslados.

Puede consultar los demás diagramas de clases en el Anexo 2.

2.6 Diagramas de Interacción

En el flujo de diseño se utilizan los diagramas de interacción, estos incluyen las interacciones entre las clases del diseño a través de mensajes, que describen las operaciones que realiza cada clase para colaborar con otras con el objetivo de dar cumplimiento a la petición del usuario.

A continuación se muestra el diagrama de secuencia del caso de uso Planificar Traslado.

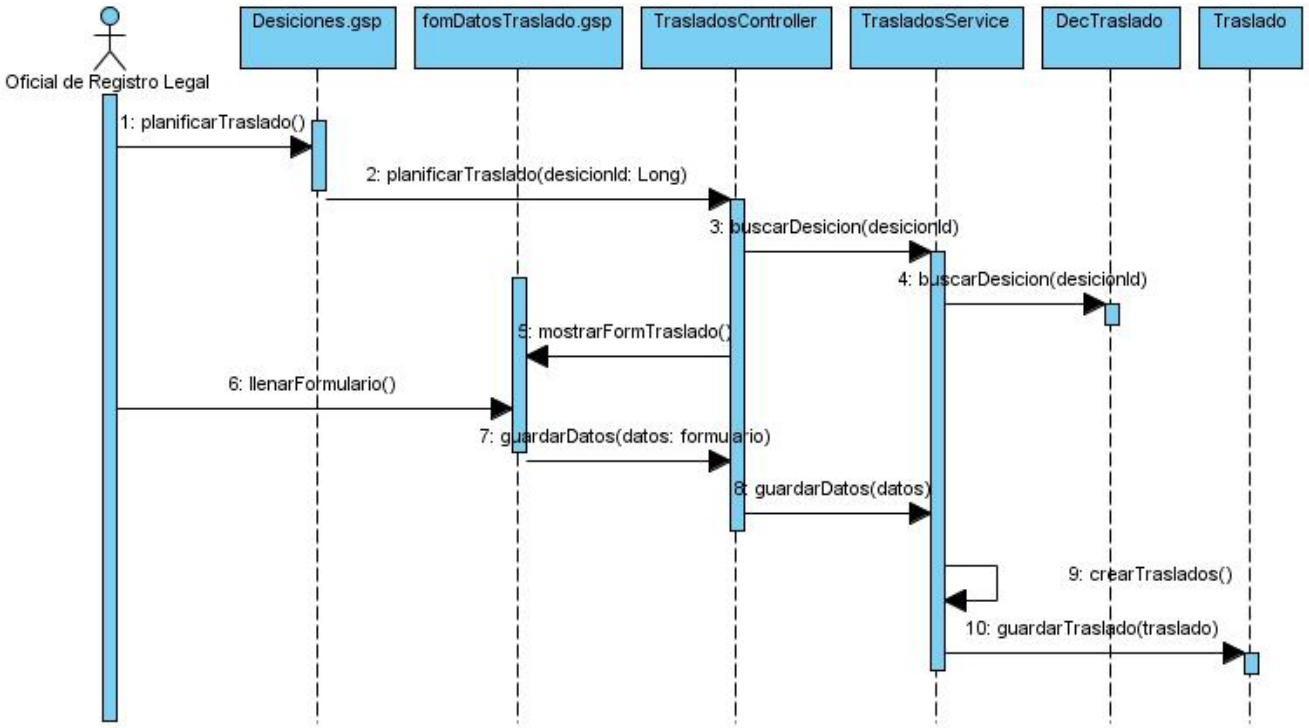


Figura 7: Diagrama de secuencia del CU Planificar Traslados.

Para consultar los demás diagramas de secuencia, dirigirse al Anexo 3.

2.7 Diseño de la base de datos

Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada. Puesto que un diseño correcto es esencial para lograr los objetivos fijados para la base de datos.

Durante el proceso de diseño de una base de datos se deben seguir algunos principios fundamentales por ejemplo, se deben evitar los datos redundantes, ya que malgastan el espacio y propician la ocurrencia de errores e incoherencias. El segundo principio plantea la importancia de que la información sea correcta y completa. Si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta y, por tanto, las decisiones que tome a partir de esos informes estarán mal fundamentadas.

Para lograr un buen diseño de base de datos es necesario dividir la información en tablas basadas en temas para reducir los datos redundantes, esto proporciona al gestor de base de datos la información necesaria para reunir la información de las tablas cuando así se precise. Además

ayuda a garantizar la exactitud e integridad de la información y satisface las necesidades de procesamiento de los datos y de generación de informes.

El proceso de diseño de la base de datos se realiza basándose en los diagramas resultantes de las clases del dominio del módulo. Durante este proceso se utilizaron patrones de diseño de base de datos que permitieron crear una base de datos más sólida, ya que estos constituyen una guía para la creación de la misma. Dentro de los patrones utilizados se encuentran:

- Árboles fuertemente codificados que permite que cada entidad esté asociada a un nivel del árbol. Este patrón se utiliza en la representación de las estructuras en jerarquía, soportando tantos niveles como la jerarquía requiera.
- Árboles simples que plantea que los elementos a almacenar pueden ser almacenados en la misma entidad y no pueden existir ciclos, es decir, una entidad no puede ser padre de ella misma en la jerarquía.

En la **¡Error! No se encuentra el origen de la referencia.** se muestra el diseño de la base de datos del módulo Traslados.

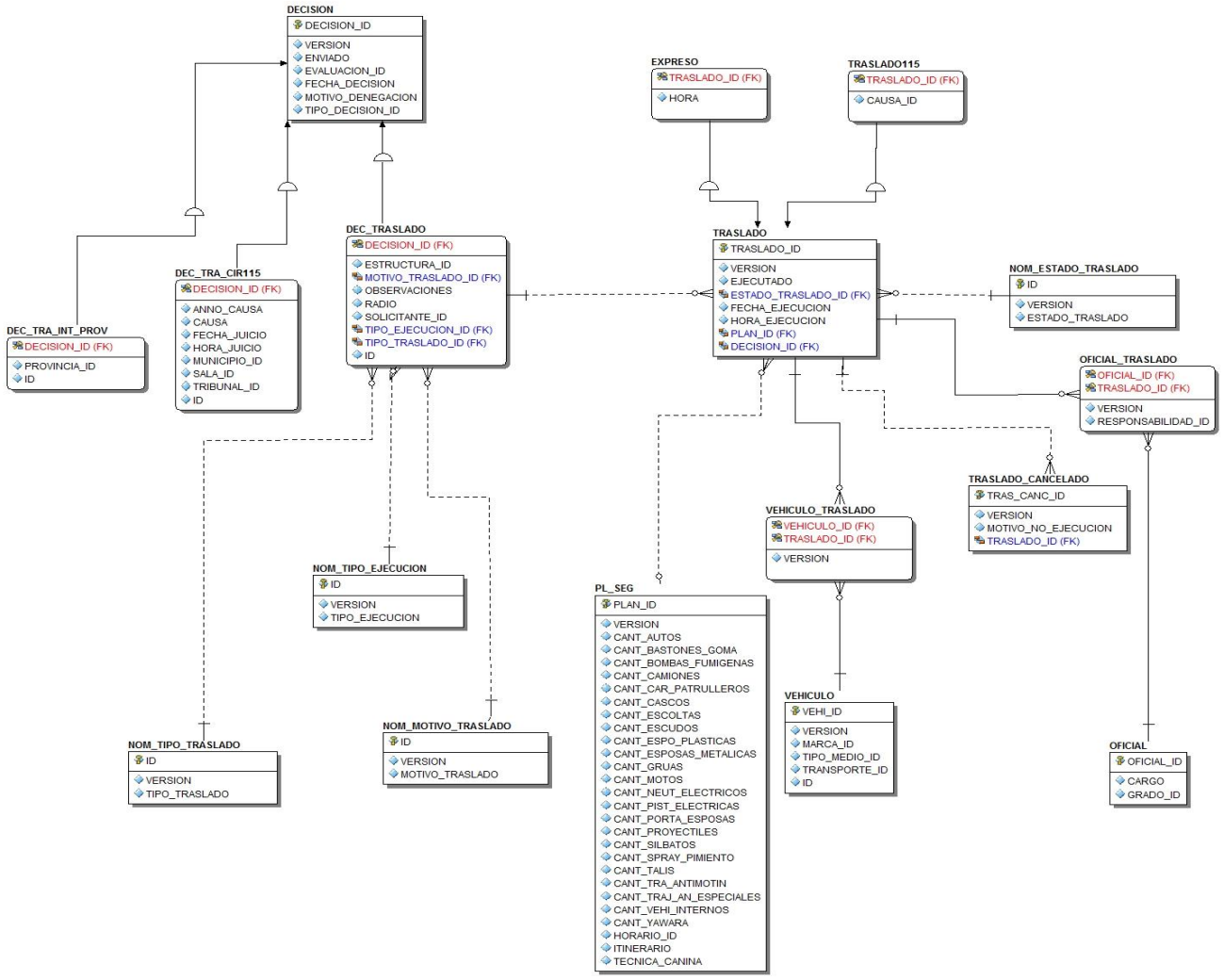


Figura 8: Diagrama Entidad Relación del módulo Traslados

2.7.1 Descripción de las tablas

A continuación se muestra la descripción de la tabla Traslado, la más importante de la base de datos.

Tabla 4: Descripción de la tabla Traslado.

Nombre	**Traslado**	
Descripción	Almacena los datos de los traslados	
Atributo	**Tipo**	**Descripción**
TRASLADO_ID	NUMBER(19,0)	Guarda el id del traslado, número

		hasta 19 dígitos, no puede ser nulo
EJECUTADO	BOOLEAN	Guarda true si el traslado fue ejecutado, no puede ser nulo
ESTADO_TRASLADO_ID	NUMBER(19,0)	Guarda el id del valor de NomEstadoTraslado, no puede ser nulo
FECHA_EJECUCION	DATE	Guarda la fecha de ejecución del traslado, no puede ser nulo
HORA_EJECUCION	VARCHAR2(255 BYTE)	Guarda la hora de ejecución del traslado, no puede ser nulo
PLAN_ID	NUMBER(19,0)	Guarda el id del plan de seguridad asociado al traslado, no puede ser nulo
DECISION_ID	NUMBER(19,0)	Guarda el id de la decisión de traslado, no puede ser nulo

Las demás descripciones de las tablas de la base de datos se encuentran en el Anexo 4.

2.8 Conclusiones parciales

En este capítulo se resumieron de forma general los casos de uso del módulo Traslados y se explicó la arquitectura definida para el SIDEP. Esto posibilitó definir la estructura de las clases mediante el diagrama de clases del diseño y haciendo uso de los patrones de diseño, los diagramas secuencia y el diseño de la base de datos. Con los artefactos obtenidos durante el diseño se puede dar paso a la implementación del módulo Traslados.

CAPITULO 3: IMPLEMENTACION Y PRUEBAS

3.1 Introducción

En este capítulo se presenta el diagrama de componentes y se explica la organización de los mismos en paquetes. Se explican, además, fragmentos de código de la implementación de las diferentes clases del módulo Traslados y se detalla cómo se le da tratamiento a los errores del módulo. Se elabora, además, el diagrama de despliegue haciendo alusión a la ubicación de los componentes en los distintos nodos. Por último, para detectar defectos del módulo se define una estrategia de pruebas y se muestran los resultados de la aplicación de las pruebas.

3.2 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos del software que entran en la fabricación de aplicaciones informáticas (15). Son usados para estructurar el modelo de implementación partiendo de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

El diagrama de compontes se elaboró teniendo en cuenta la arquitectura definida para el desarrollo de la aplicación. Se ha estructurado de forma que se pueda apreciar la relación de cada uno de los componentes con los restantes y que le dan sustento a las funcionalidades del software. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente, se realizan por partes.

La figura 9 muestra los componentes del módulo Traslados y los componentes con los cuales se relaciona. El paquete de componentes Traslados se relaciona con el componente Decisiones de Traslado, el cual pertenece al módulo Decisiones del SIDEP. Utiliza el componente Dojo.js para la capa de presentación, y el componente Sidep.css para los estilos visuales. Se relaciona también con el componente Plan de Seguridad, que es un componente común para varios módulos y contiene las clases que son utilizadas para gestionar la información referente a los planes de seguridad de los traslados. Del plugin Core, se relaciona con los Nomencladores y el componente Oficial y del plugin Admin se utiliza el componente Seguridad, el cual se utiliza para restringir el acceso de los usuarios a las diferentes funcionalidades del módulo.

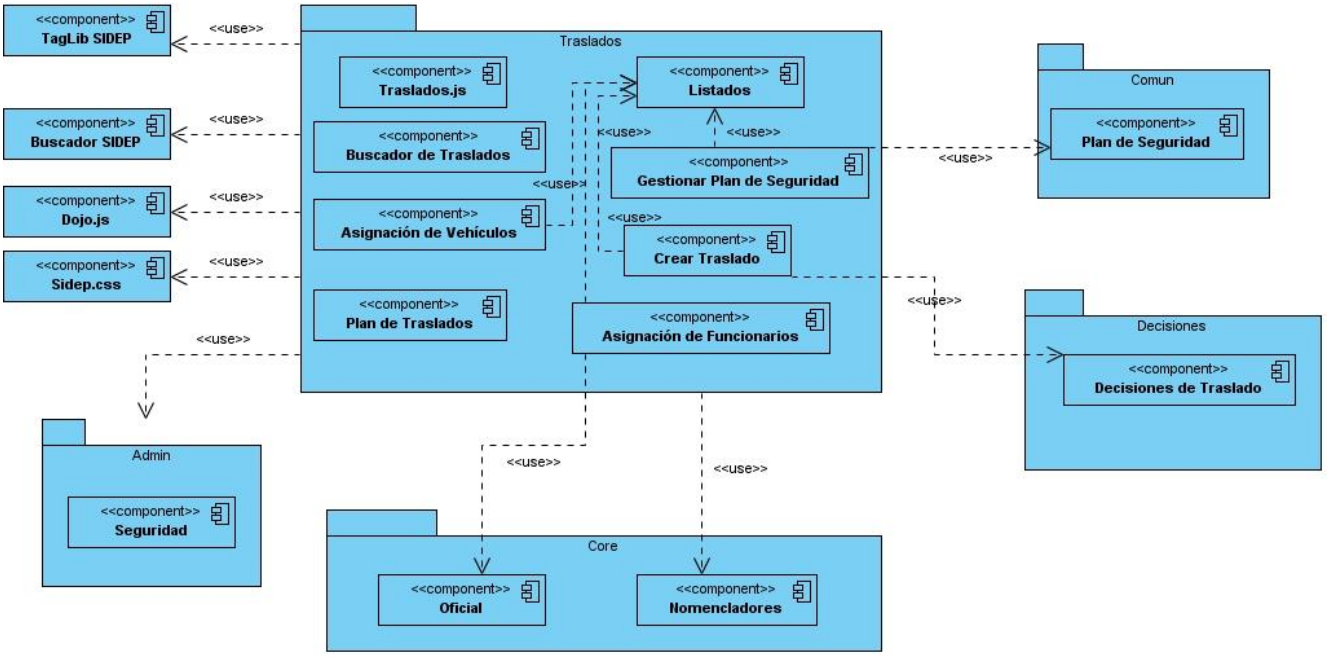


Figura 9: Diagrama de Componentes del módulo Traslados.

Para un mejor entendimiento del paquete de componentes Traslados, en el Anexo 5 se exponen los diagramas de los componentes que lo componen.

3.3 Implementación del módulo Traslados

Uno de los componentes más importantes del módulo Traslados es Crear Traslado, el cual está compuesto por todas las clases que intervienen en la correcta planificación de los traslados. Para planificar un traslado el controlador **TrasladoController** utiliza los métodos que se muestran a continuación para controlar todo el flujo de información que maneja el mismo.

```
def continuar = {
    def desicion = DecTraslado.findById(params.buscar.resultados.radio.toString().toLong())
    trasladosService.listarInternos(desicion.exps)
    if (desicion instanceof DecTraCir115) {
        render view: "${urlbase}datos115Form", model: [desicion: desicion]
    } else {
        render view: "${urlbase}datosTrasladosForm", model: [desicion: desicion]
    }
}
```

Figura 10: Método Continuar de TrasladoController

```
def finalizar = {
    def traslado = new Traslado()
    traslado.decision = DecTraslado.get(params.desicion.id)
    traslado.fechaEjecucion = new SimpleDateFormat("yyyy-MM-dd").parse(params.traslado.fechaEjecucion)
    traslado.horaEjecucion = params.traslado.horaEjecucion.toString()
    traslado.ejecutado = false
    traslado.prorrogado = false
    traslado.estadoTraslado = NomEstadoTraslado.get(1.toLong())
    trasladosService.salvar(traslado)
    redirect controller: "buscarTr"
}

def fin115 = {
    def tr115 = new Traslado115()
    def c = Causa.findByCausa(DecTraCir115.get(params.desicion.id).causa)
    tr115.decision = DecTraCir115.get(params.desicion.id)
    tr115.fechaEjecucion = new SimpleDateFormat("yyyy-MM-dd").parse(params.traslado.fechaEjecucion)
    tr115.horaEjecucion = params.traslado.horaEjecucion.toString()
    tr115.ejecutado = false
    tr115.estadoTraslado = NomEstadoTraslado.get(1.toLong())
    tr115.prorrogado = false
    tr115.causa = c
    trasladosService.salvar(tr115)
    redirect controller: "buscarTr"
}
```

Figura 11: Métodos Finalizar de TrasladoController

Otra clase importante de este componente es **TrasladosService** el cual sirve de apoyo a la clase **TrasladoController** para registrar los traslados utilizando los siguientes métodos:

```
def salvar(Traslado t) {
    if (t.validate()){
        t.save()
       }else{
            t.properties.each{
        println it
        }
    }
}
```

```
def salvar115(Traslado115 t) {
    if (t.validate()){
       t.save()
       }else{
            t.properties.each{
        println it
        }
    }
}
```

Figura 12: Métodos Salvar de TrasladosService

Otro componente importante es Asignación de Funcionarios el cual utiliza los servicios para primeramente mostrar todos los traslados a los cuales se le pueda realizar la asignación de recursos, ya sea humanos o vehículos, y luego de seleccionado un traslado, se registran los datos en la base de datos. A continuación se muestran los principales métodos de **asigFuncionarioService**:

```
public List getItems(Object params) {
    items = []
    def lista = []
    def resultado = Traslado.findAllByEstadoTraslado(NomEstadoTraslado.findByEstadoTraslado("Aprobado"))

    resultado.each {
        if (!OficialTraslado.findByTraslado(it))
            lista.add(it)
    }
    resultado = lista
    resultado.each {
        def item = [:]
        item.id = it.id
        item.tipoTraslado = """
            <p style="margin: 2px;">
            <i id="${it?.decision.tipoTraslado}">${it?.decision.tipoTraslado}</i>
            </p>
        """
        item.sel = """<input dojotype="dijit.form.RadioButton" id="${it?.id}"
            onclick="document.getElementById('traslado.resultado').value = this.value;"  name="radio"
            value="${it?.id}" type="radio" />"""
        item.fechaEjecucion = """
            <p style="margin: 2px;">
            <i id="${it?.fechaEjecucion}">${it?.fechaEjecucion}</i><br />
            </p>
        """
        item.destino = """
            <p style="margin: 2px;">
            <i id="${it?.decision.estructura.nombreEstructura}">${it?.decision.estructura.nombreEstructura}</i>
            </p>
        """
        items << item
    }
    ApplicationUtils.pag(items, params)
}
```

Figura 13: Método getItems de AsigFuncionarioService

El método mostrado anteriormente muestra todos los traslados planificados que no tienen asignado aún recursos humanos. De los traslados muestra el tipo de traslado, la fecha de ejecución y el centro de destino del mismo. Esta clase utiliza también el método mostrarOficial() en el cual se apoya en **mostrarOficialesService** que muestra los oficiales seleccionados para asumir una responsabilidad en el traslado y lo agrega a una tabla que se muestra en la vista. Este método se muestra a continuación:

```
def mostrarOficial(OficialTraslado oftraslado) {
    def list = mostrarOficialesService.items
    def b = true
    list.each {
        if (it.oficial.id == oftraslado.oficial.id && oftraslado.responsabilidad == null)
            b = false
    }
    if (b)
        mostrarOficialesService.add(oftraslado)
}
```

Figura 14: Método MostrarOficial de AsigFuncionarioService

3.4 Tratamiento de errores

Durante el desarrollo del módulo Traslados se tomaron varias medidas para garantizar la integridad de los datos procesados. Dentro de estas medidas se encuentran la validación del lado del cliente utilizando Javascript y las validaciones del lado del servidor utilizando Groovy y las constraints de las clases del dominio.

Para la validación de los datos introducidos en las vistas se utilizan los archivos Javascript de las páginas los cuales procesan los datos a nivel de cliente sin necesidad de realizar el envío al servidor. Un ejemplo de la forma en que se validan los datos es la siguiente:

```
finalizar :function() {
    if (dojo.byId("datos") != null)
        if (dijit.byId("traslado.horaEjecucion").value != null)
            dojo.byId("datos").submit();
        else {
            sidep.core.app.mostrarMensaje("Debe seleccionar la hora de ejecución.", "error");
            sidep.core.app.cambiarColorBorde("traslado.horaEjecucion");
        }
},
```

Figura 15: Tratamiento de errores a nivel de cliente

Para la validación de los datos de entrada en el servidor, se manejaron los datos de forma tal que el módulo sea capaz de saber cuándo son correctos e incorrectos. Es por ello que en todas las acciones con probabilidad de ocurrencia de errores los datos se manejan de la siguiente forma:

```
def continuar = {
    if(params.idTraslado!=null){
        if (params.idTraslado) {
            asigFuncionarioService.setTraslado(params.idTraslado.toString().toLong())
        } else {
            asigFuncionarioService.setTraslado(params.radio.toString().toLong())
        }

        asigFuncionarioService.listarInternos(asigFuncionarioService.getTraslado().decision.exps)
        redirect action: "oficialesList"
    }
    else{
        flash.error = "Debe seleccionar un traslado."
        render view: "${urlbase}trasladosList"
    }
}
```

Figura 16: Tratamiento de errores a nivel de servidor

Los constraints son elementos importantes en las clases de dominio, constituyen la garantía de que los datos que se van a persistir no contienen errores. En la siguiente figura se muestra la definición del constraints de la clase del Traslado:

```
static constraints = {
    plan(nullable: true)
    decision(nullable: false)
    ejecutado(nullable: false)
    prorrogado(nullable: false)
    estadoTraslado(nullable: false)
    horaEjecucion(nullable: false)
    fechaEjecucion(nullable: false)
}
```

Figura 17: Constraints de la clase Traslado

3.5 Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación (16). Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

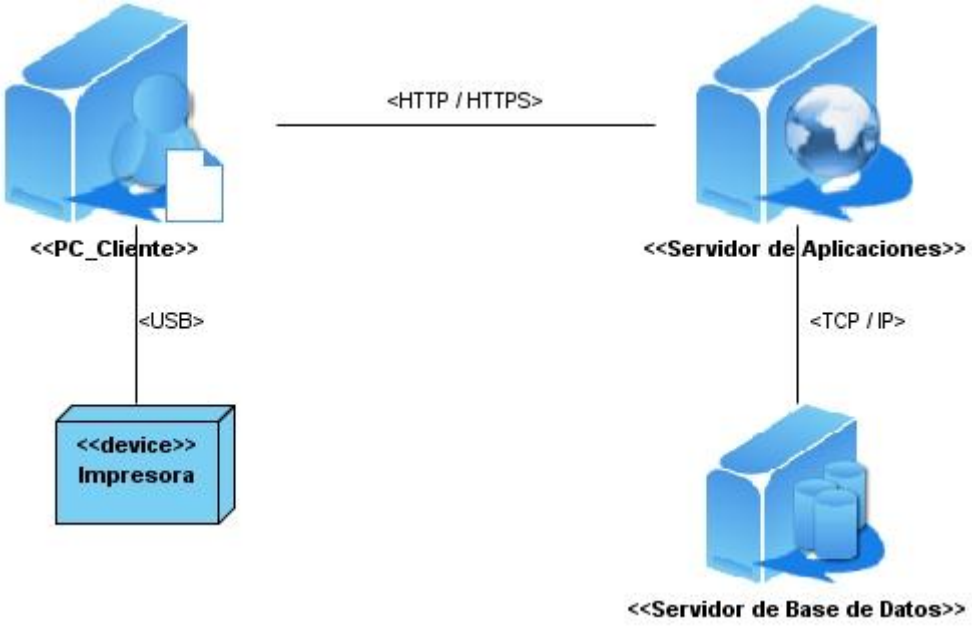


Figura 18: Diagrama de despliegue del módulo Traslados

El diagrama de despliegue del módulo Traslado muestra la distribución del sistema en función de los nodos que lo forman. La máquina cliente se conecta al servidor web que es el encargado de responder las peticiones y ejecutar las funcionalidades, y este se conecta a la base datos para la gestión de datos en caso de que sea necesario. El componente Traslados se encuentra en el servidor de aplicaciones y es accesible a través del cliente para realizar todas las funcionalidades disponibles. El dispositivo de impresión se debe incluir para la impresión de los modelos de traslado.

3.6 Pruebas

Las pruebas son procesos en los cuales una aplicación o componente es ejecutado en condiciones específicas con el objetivo de detectar los errores del mismo. Los resultados son observados y registrados, y se hace una evaluación teniendo en cuenta los mismos.

Las pruebas realizadas al módulo Traslados se realizan utilizando el método caja negra el cual se lleva a cabo sobre la interfaz del software, sin tener en cuenta el comportamiento interno y la estructura del mismo. No validan funciones implementadas que no hayan sido descritas en las especificaciones funcionales del diseño, por tanto los errores asociados a ellas no serán encontrados. Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes
- Errores en la interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y de terminación

El tipo de prueba escogido fue el de función utilizando la técnica de diseño de casos de prueba con el objetivo de probar las funcionalidades del módulo, utilizando las descripciones de caso de uso para la validación de los datos introducidos que son utilizados por las funciones, métodos y servicios. Los casos de prueba pretenden demostrar que:

- Las funciones del software son operativas
- La entrada se acepta de forma correcta
- Se produce una salida correcta
- La integridad de la información externa se mantiene

3.6.1 Diseño de los casos de prueba

El objetivo del diseño de casos de prueba es demostrar que las funcionalidades de la aplicación aceptan los datos correctos, los procesan y devuelven el resultado esperado. Un caso de prueba está diseñado correctamente cuando tiene altas probabilidades de descubrir algún error no descubierto anteriormente. A continuación se muestra el diseño de casos de prueba para el caso de uso Planificar Traslado.

Descripción general: El caso de uso se inicia cuando el Oficial de Registro Legal selecciona la opción “Planificación del traslado”, el sistema muestra las decisiones de traslado, el Oficial de Registro Legal selecciona una de las decisiones, ingresa los datos necesarios del traslado y el sistema registra los datos del traslado asignándole el estado de planificado y finaliza el caso de uso.

Condiciones de ejecución:

1. El Oficial de Registro Legal debe de estar autenticado en el sistema.
2. Para planificar un traslado debe haberse registrado una decisión de traslado o prórroga de traslado con evaluación de aprobado.

SC Registrar datos del traslado:

Escenario	Descripción	Fecha de ejecución	Hora	Respuesta del sistema	Flujo Central
EC 1.1 Registrar satisfactoriamente datos del traslado.	Registra satisfactoriamente datos del traslado.	V	V	El sistema muestra las decisiones de “Traslado” o “Prórroga de traslado” con Evaluación de “Aprobado” que no han sido planificados aún en el centro penitenciario; verifica los internos a ser trasladados, valida los datos introducidos y luego registra los datos del traslado.	1. Seleccionar la opción “Planificación del traslado”. 2. Seleccionar una decisión. 3. Oprimir el botón “Planificar”. 4. Introducir los datos obligatorios solicitados. 5. Oprimir el botón Cerrar.
EC 1.2 Interno	Permite saber los	NA	NA	El sistema muestra un	1. Seleccionar la opción

enfermo de VIH/SIDA	internos que están enfermos de VIH/SIDA.			mensaje de error "Hay interno(s) que padecen de VIH/SIDA" y señala los internos que padezca la enfermedad.	"Planificación del traslado". 2. Seleccionar una decisión. 3. Oprimir el botón "Planificar".
EC 1.3 Interno con conduce al Tribunal	Permite señalar al interno involucrado para presentación del juicio.	NA	NA	El sistema muestra el mensaje "Hay internos con conduce para presentación a juicio, ¿desea continuar?" y señala el interno involucrado. Permite regresar a la vista "Registrar datos del traslado" si no se selecciona cancelar.	1. Seleccionar la opción "Planificación del traslado". 2. Seleccionar una decisión. 3. Oprimir el botón "Planificar". 4. Seleccionar motivo(Al Tribunal para la celebración del juicio oral o notificación de alguna resolución judicial.) 5. Oprimir el botón Aceptar.
EC 1.4 Los datos ya existen en el sistema	Se valida que ya exista el traslado en el sistema.	I	V	El sistema muestra un mensaje "El traslado ya existe en el sistema" y regresa a la vista Registrar datos del traslado.	1. Seleccionar la opción "Planificación del traslado". 2. Seleccionar una decisión. 3. Oprimir el botón "Planificar". 4. Introducir datos que ya existen.
		V	I	El sistema muestra un mensaje "El traslado ya existe en el sistema" y regresa a la vista Registrar datos del traslado.	1. Seleccionar la opción "Planificación del traslado". 2. Seleccionar una decisión. 3. Oprimir el botón "Planificar". 4. Introducir datos que ya existen.

EC 1.5 Errores en los datos introducidos	Se verifica que no existan campos obligatorios incorrectos cuando se registre datos del traslado.	I	V	El sistema muestra el mensaje de error “Corrija los datos erróneos introducidos” y señala los datos erróneos. Regresa a la vista “Registrar datos del traslado”.	1. Introducir los datos señalados como incorrectos. 2. Se oprime el botón Cerrar.
		V	I	El sistema muestra el mensaje de error “Corrija los datos erróneos introducidos” y señala los datos erróneos. Regresa a la vista “Registrar datos del traslado”.	1. Introducir los datos señalados como incorrectos. 2. Se oprime el botón Cerrar.
EC 1.6 Faltan datos obligatorios.	Se verifica que no existan campos obligatorios incompletos cuando se registre datos del traslado.	I	V	El sistema muestra el mensaje de error “Introduzca los datos obligatorios” y señala los campos obligatorios donde no se introdujeron datos. Regresa a la vista “Registrar datos del traslado”.	1. Introducir los datos obligatorios incompletos. 2. Se oprime el botón Cerrar.
		V	I	El sistema muestra el mensaje de error “Introduzca los datos obligatorios” y señala los campos obligatorios donde no se introdujeron datos. Regresa	1. Introducir los datos obligatorios incompletos. 2. Se oprime el botón Cerrar.

				a la vista “Registrar datos del traslado”.	
EC 1.7 Fecha de ejecución inválida	Permite mostrar cuando una fecha de ejecución es inválida.	I	V	El sistema muestra el mensaje “La fecha de ejecución introducida excede la fecha válida para ejecutar el traslado”. Regresa a la vista “Registrar datos del traslado”.	1. Introducir la fecha de ejecución incorrectamente. 2. Se oprime el botón Cerrar.
		V	I	El sistema muestra el mensaje “La fecha de ejecución introducida excede la fecha válida para ejecutar el traslado”. Regresa a la vista “Registrar datos del traslado”.	1. Introducir la hora incorrectamente. 2. Se oprime el botón Cerrar.
EC 1.8 Cancelar registrar datos del traslado.	Se cancela el registrar datos del traslado.	NA	NA	El sistema no registra datos del traslado, y regresa a la vista “Registrar datos del traslado”.	1. Seleccionar la opción “Planificación del traslado”. 2. Seleccionar una decisión. 3. Oprimir el botón “Planificar”. 4. Introducir los datos obligatorios solicitados. 5. Oprimir el botón Cancelar.

Descripción de las variables:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Fecha de ejecución	Campo de selección	No	Se debe escoger una fecha de ejecución válida.
2	Hora	Campo de texto	No	Se debe poner una hora válida, caracteres entre [0-9]

Las pruebas aplicadas al módulo Traslados arrojaron en la primera iteración un total de 19 no conformidades, 4 en la segunda iteración y en la tercera no se detectaron no conformidades.

3.7 Conclusiones Parciales

En este capítulo se desarrolló el diagrama de componentes del módulo Traslados y se expusieron fragmentos de códigos de la aplicación y con la explicación de las principales funcionalidades. Se elaboró, además, el diagrama de despliegue describiendo la ubicación de los componentes en los nodos y se realizaron pruebas de calidad al módulo que culminaron en una tercera iteración con la ausencia de no conformidades detectadas.

CONCLUSIONES GENERALES

Se desarrolló el módulo Traslados del SIDEP que permite la planificación y el registro de la ejecución de los traslados en el Sistema Penitenciario Cubano. Los resultados obtenidos permiten plantear las siguientes conclusiones:

- Se estudiaron los sistemas SACORE y SIGEP demostrándose que no resuelven el problema planteado.
- Se analizaron las tecnologías, metodología, lenguajes y herramientas definidas por el equipo de arquitectura del SIDEP para el desarrollo del módulo.
- Los artefactos obtenidos durante el diseño facilitaron la implementación del módulo.
- Se implementó el módulo Traslados que permite a partir de decisiones tomadas planificar y mantener un control de la ejecución de los traslados entre centros penitenciarios.
- Las pruebas realizadas arrojaron la ausencia de no conformidades en una tercera iteración.

De esta forma se le da cumplimiento al objetivo de la investigación.

RECOMENDACIONES

Después de haber desarrollado el módulo Traslados se recomienda lo siguiente:

- Darle seguimiento a nuevos requisitos que surjan a partir de cambios en las legislaciones vigentes para la actualización del módulo.
- Implementar, en una segunda iteración del módulo, la funcionalidad de eliminar el Plan de Seguridad.
- Implantar el módulo en un ambiente real para su explotación.

BIBLIOGRAFÍA CITADA

1. **KNIGHT, T. C. H.** *Sistema Informativo de la Dirección de Establecimientos Penitenciarios*, 2005.

2. **JACOBSON, I.; G. BOOCH**. *El Proceso Unificado del Desarrollo del Software*. . 435 p.

3. Definición de UML. www.mastermagazine.info. [En línea] 26 de 02 de 2005. [Citado el: 22 de Enero de 2012.] http://www.mastermagazine.info/termino/7006.php.

4. Freedown Load Manager. [En línea] [Citado el: 7 de Enero de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

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

6. JavaScript.com. JavaScript.com. [En línea] 2011. [Citado el: 10 de 12 de 2011.] http://www.javascript.com/.

7. Nuevas Tecnologías. [En Línea] [Citado el: 12 de Enero de 2012] http://www.slideshare.net/reingsys/nuevas-tecnologas-reingsys-31309

8. Desarrollo Web con Grails. [En línea] [Citado el: 10 de Enero de 2012] http://es.scribd.com/doc/73695471/Desarrollo-Web-Con-Grails-1-1-X

9. dojotoolkit.org. Dojo Toolkit. [En línea] [Citado el: 28 de 11 de 2011.] http://dojotoolkit.org/.

10. Qué es un entorno de desarrollo integrado, IDE. [En línea] [Citado el: 10 de Enero de 2012] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/

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

12. Definiciones: NetBeans [En línea] [Citado el: 11 de Enero de 2012] http://gutl.jovenclub.cu/wiki/definiciones/net_beans

13. Definición de Arquitectura Software [En línea] [Citado el: 20 de Enero de 2012] www.mastermagazine.info/termino/3916.php

14. Patrones de diseño [En línea] [Citado el: 22 de Enero de 2012] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

15. Modelo de Implementación: Diagramas de Componentes y Despliegue [En línea] 2010. [Citado el: 14 de Abril de 2010.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

16. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana**. Trabajo de Investigación y Exposición. 2010.

BIBLIOGRAFÍA CONSULTADA

1. **Gálvez Casanova, Dayneris de la Caridad y Echemendía Lazo, José Livan** . *Análisis y diseño de los Módulos Traslados y Conducción del Sistema de Gestión Penitenciario Cubano.* LA HABANA : s.n., 2010.
2. *PROCEDIMIENTOS DE TRABAJO DE REGISTRO LEGAL.* [PDF].
3. Rational Software Corporation. *RUP. "Rational Unified Process".* 2003.
4. Metodologías tradicionales y metodologías ágiles. *www.eumed.net.* [En línea] 2009. [Citado el: 15 de 12 de 2011] http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.
5. Metodologías de desarrollo de software. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 20 de Enero de 2012.] http://eva.uci.cu.
6. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN: 970-17-0261-1.
7. ¿Qué es una Herramienta Case? [citado el 10 de Febrero de 2012][En línea], Disponible en http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
8. Embarcadero Technologies Inc. Embarcadero. [En línea] [Citado el: 10 de 12 de 2011.] http://www.embarcadero.com/products/er-studio.
9. **Brito, Nacho.** *Manual de desarrollo web con Grails. JavaEE, como siempre debió haber sido.* 2009. ISBN: 978-84-613-2651.
10. **Dearle, Fergal.** *Groovy for Domain-Specific Languages.* s.l. : Packt Publishing Ltd., 2010. ISBN 978-1-847196-90-3.
11. Bienvenido a NetBeans y www.netbeans.org. netbeans. [En línea] 2011. [Citado el: 03 de Febrero de 2012.] http://netbeans.org/index_es.html.
12. Apache Software Foundation. Apache Tomcat. [En línea] 1999-2011. [Citado el: 28 de 11 de 2011.] http://tomcat.apache.org.
13. Oracle. *Oracle Database Documentation Library.* 2005.
14. **Manso Rodríguez, Ludmila Yirenis y Pizarro Barata, Maydalis**. *Diseño e implementación de los módulos Pertenencias y MedidasDisciplinarias del SIGEP Venezuela.* LA HABANA : s.n., 2010.

15. **Bertolami, Leandro y Kamil, Pablo.** *Grails: El Santo Grial de Java .* 2008.
16. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad).Parte II [En línea] 2007 [Citado el: 22 de Enero de 2012] http://jorgesaavedra.wordpress.com/category/patrones-grasp/
17. Diagrama de Clases [En línea] 2009 [Citado el: 22 de Enero de 2012] http://egdamar877.blogspot.com/2009/05/expocicion.html
18. Conceptos básicos del diseño de una base de datos [En línea] 2007 [Citado el: 22 de Enero de 2012] http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx
19. Técnicas de prueba [En línea] 2010 [Citado el: 22 de Febrero de 2012] http://indalog.ual.es/mtorres/LP/Prueba.pdf

ANEXOS

Anexo 1: Descripción de las clases del dominio

Tabla 5: Descripción de la clase OficialTraslado.

Nombre	OficialTraslado
Tipo de Clase	Clase del dominio
Atributo	**Tipo**
responsabilidad	NomRespTraslado
oficial	Oficial
traslado	Traslado

Tabla 6: Descripción de la clase VehiculoTraslado.

Nombre	VehiculoTraslado
Tipo de Clase	Clase del dominio
Atributo	**Tipo**
vehiculo	Vehiculo
traslado	Traslado

Tabla 7: Descripción de la clase Traslado115.

Nombre	Traslado115
Tipo de Clase	Clase del dominio
Atributo	**Tipo**
decision	DecTraslado

ejecutado	Boolean
estadoTraslado	NomEstadoTraslado
horaEjecucion	String
fechaEjecucion	Date
plan	PlSeg
causa	Causa

Tabla 8: Descripción de la clase TrasladoCancelado.

Nombre	TrasladoCancelado
Tipo de Clase	Clase del dominio
Atributo	**Tipo**
traslado	Traslado
motivoNoEjecucion	String

Tabla 9: Descripción de la clase TrasladoController.

Nombre	TrasladoController
Tipo de Clase	Controlador
Atributo	**Tipo**
trasladosService	TrasladosService
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
insertar()	Carga la vista que contiene el listado de las decisiones de traslados aprobadas.

continuar()	Carga la vista que contiene los datos de la decisión seleccionada y un formulario con los datos del traslado.
cancelar()	Regresa a la vista que contiene el listado de las decisiones de traslados aprobados.
finalizar()	Guarda en la base de datos los datos del traslado aprobado.

Tabla 10: Descripción de la clase AsigFuncionarioController.

Nombre	AsigFuncionarioController
Tipo de Clase	Controlador
Atributo	**Tipo**
asigFuncionarioService	AsigFuncionarioService
oftraslado	OficialTraslado
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
insertar()	Carga la vista que contiene el listado de los traslados aprobados.
continuar()	Guarda el traslado en el servicio y llama a la función oficialesList().
oficialesList()	Le asigna el traslado al OficialTraslado y le asigna también un oficial y una responsabilidad en el caso de que ninguno de estos sea nulo, y carga la vista que muestra los datos del traslado con sus respectivos oficiales.
mostrarOficial()	Añade un oficial al listado de oficiales asignados al

	traslado.
asigResp()	Carga la vista que contiene el listado de las responsabilidades que puede tener un oficial en el traslado.
asignar()	Guarda la responsabilidad en el servicio y vuelve a la vista que muestra los datos del traslado con sus respectivos oficiales.
finalizar()	Guarda en la base de datos los datos del OficialTraslado.

Tabla 11: Descripción de la clase AsigTransporteController.

Nombre	AsigTransporteController
Tipo de Clase	Controlador
Atributo	**Tipo**
asigTransporteService	AsigTransporteService
vehtraslado	VehiculoTraslado
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
insertar()	Carga la vista que contiene el listado de los traslados aprobados.
continuar()	Guarda el traslado en el servicio y llama a la función vehiculosList().
vehiculosList ()	Le asigna el traslado al VehiculoTraslado y le asigna también un vehiculo en el caso de que no sea nulo, y carga la vista que muestra los datos del traslado con sus respectivos vehículos.

mostrarVehiculo()	Añade un vehiculo al listado de vehiculos asignados al traslado.
finalizar()	Guarda en la base de datos los datos del VehiculoTraslado.

Tabla 12: Descripción de la clase BuscarTrController.

Nombre	BuscarTrController
Tipo de Clase	Controlador
Atributo	**Tipo**
buscarTrasladoService	BuscarTrasladoService
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
index()	Carga la vista que contiene el formulario con los criterios de búsqueda de los traslados aprobados.
busqueda()	Carga la vista que contiene un listado de traslados que cumplen con los criterios de búsqueda.
continuar ()	Carga la vista que contiene los datos del traslado seleccionado.

Tabla 13: Descripción de la clase BuscarTrController.

Nombre	CambiarEstadoController
Tipo de Clase	Controlador
Atributo	**Tipo**
cambiarEstadoService	CambiarEstadoService

Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
insertar ()	Carga la vista que contiene el listado de los traslados planificados para que el usuario seleccione el que quiere modificar.
continuar ()	Guarda el traslado en el servicio y carga la vista que contiene los datos del traslado y permite modificarlos.
finalizar ()	Modifica el estado del traslado siempre que este sea válido y guarda los datos en la base de datos.
Finalizar115 ()	Obtiene los datos del traslado en caso de que este sea de tipo Traslado115 y modifica su estado siempre que este sea válido y guarda los datos en la base de datos.

Tabla 14: Descripción de la clase AsigFuncionarioService.

Nombre	AsigFuncionarioService
Tipo de Clase	Servicio
Atributo	**Tipo**
registroLegalService	RegistroLegalService
mostrarOficialesService	MostrarOficialesService
buscarTrabajadorService	BuscarTrabajadorService
traslado	Traslado
oficial	Oficial
responsabilidad	NomRespTraslado

Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
setTraslado(Object t)	Crea un objeto de tipo Traslado.
getTraslado()	Devuelve el objeto de tipo Traslado.
usuarioOnline()	Devuelve el usuario que está registrado en el sistema.
setOficial(Object o)	Crea un objeto de tipo Oficial.
getOficial()	Devuelve el objeto de tipo Oficial.
setResp(Object r)	Crea un objeto de tipo nomenclador NomRespTraslado.
getResp()	Devuelve el objeto de tipo nomenclador NomRespTraslado.
salvarOficiales(OficialTraslado o)	Salva el objeto OficialTraslado pasado por parámetros.
addTabla(OficialTraslado oftraslado)	Adiciona el objeto OficialTraslado a la tabla donde se muestran.

Tabla 15: Descripción de la clase AsigVehiculoService.

Nombre	AsigVehiculoService
Tipo de Clase	Servicio
Atributo	**Tipo**
registroLegalService	RegistroLegalService
mostrarVehiculosService	MostrarVehiculosService
buscarVehiculoService	BuscarVehiculoService
traslado	Traslado

vehiculo	Vehiculo
Funcionalidades	
Nombre de la Funcionalidad	**Descripción**
setTraslado(Object t)	Crea un objeto de tipo Traslado.
getTraslado()	Devuelve el objeto de tipo Traslado.
usuarioOnline()	Devuelve el usuario que está registrado en el sistema.
setVehiculo(Object o)	Crea un objeto de tipo Vehiculo.
getVehiculo()	Devuelve el objeto de tipo Vehiculo.
salvarVehiculos(VehiculoTraslado v)	Salva el objeto VehiculoTraslado pasado por parámetros.
addTabla(VehiculoTraslado vehtraslado)	Adiciona el objeto VehiculoTraslado a la tabla donde se muestran.

Anexo 2: Diagramas de clases del diseño

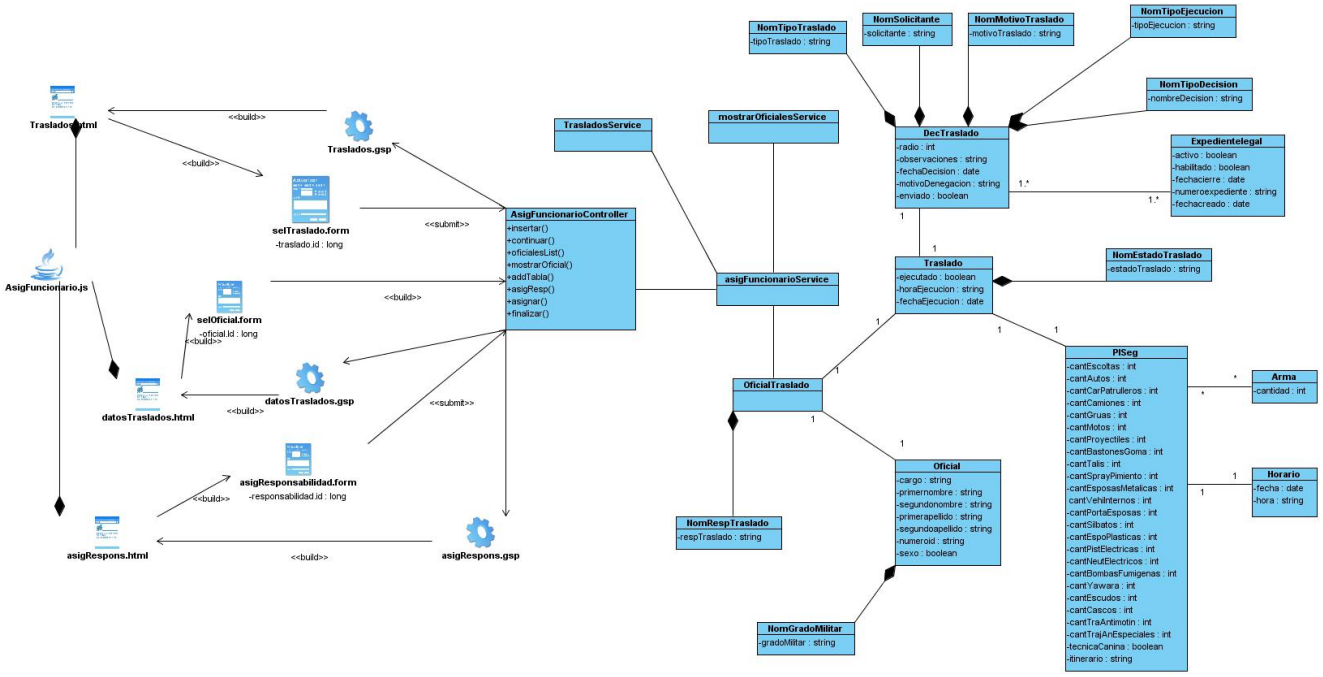


Figura 19: Diagrama de clases con estereotipos web del CU Asignar Funcionarios.

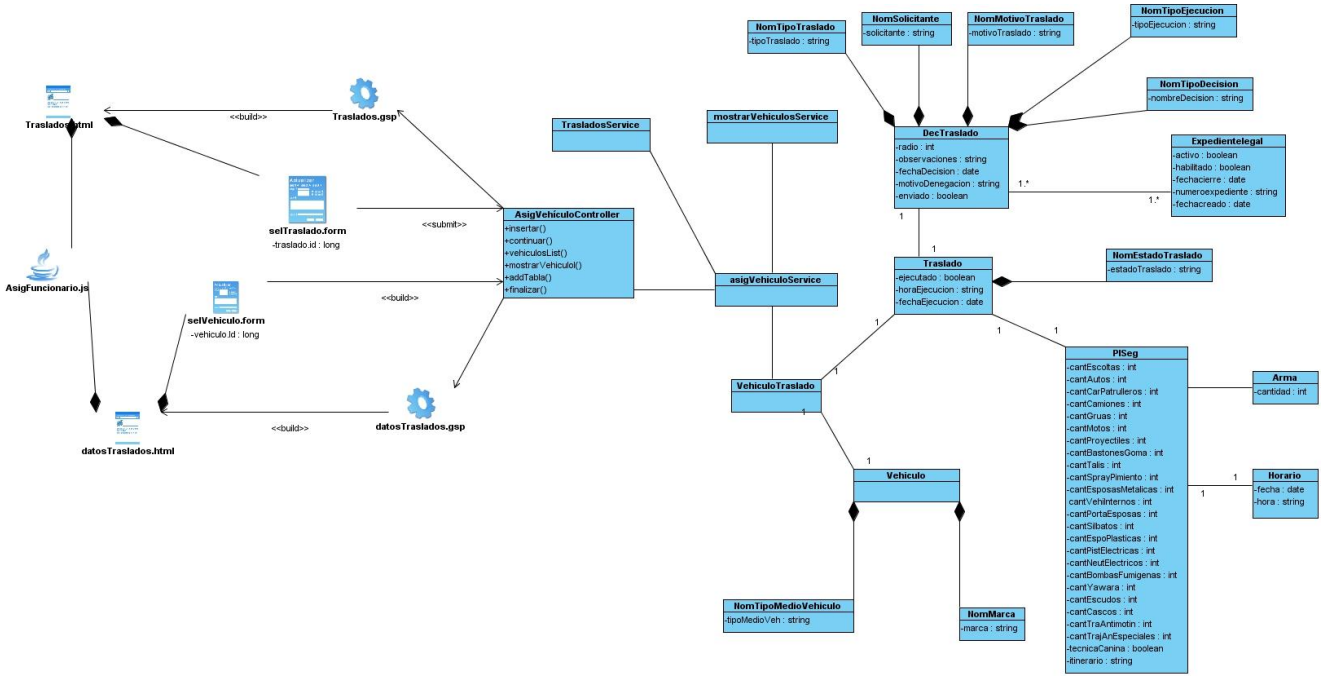


Figura 20: Diagrama de clases con estereotipos web del CU Asignar Vehículos.

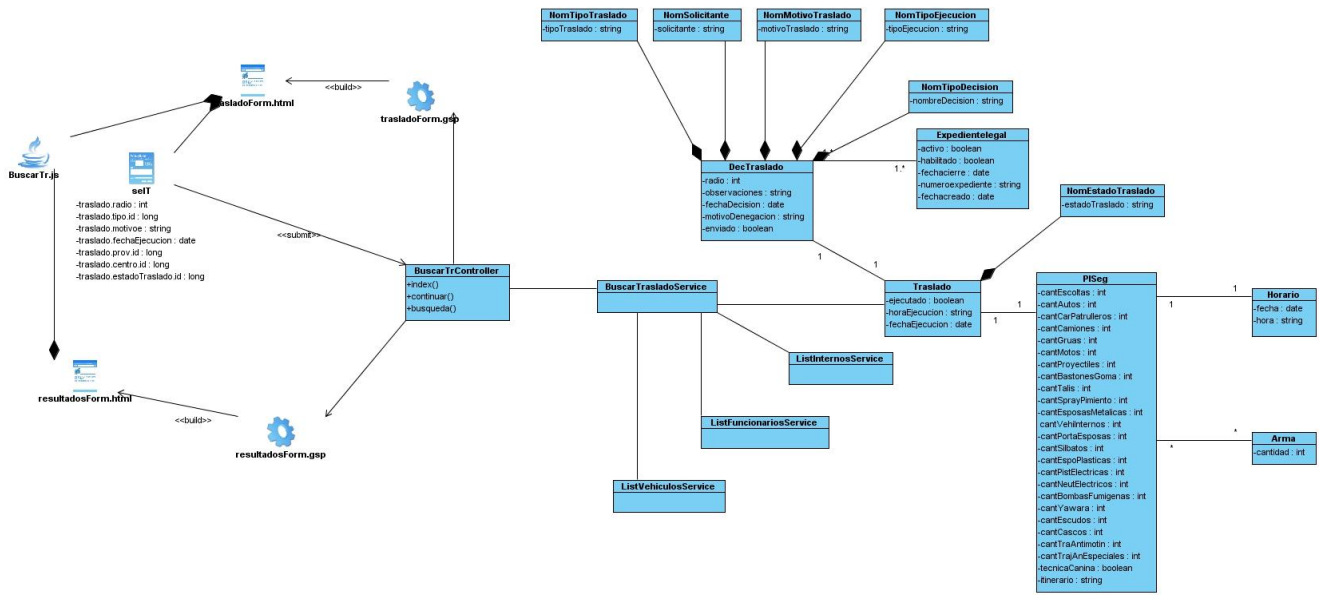


Figura 21: Diagrama de clases con estereotipos web del CU Buscar Traslado.

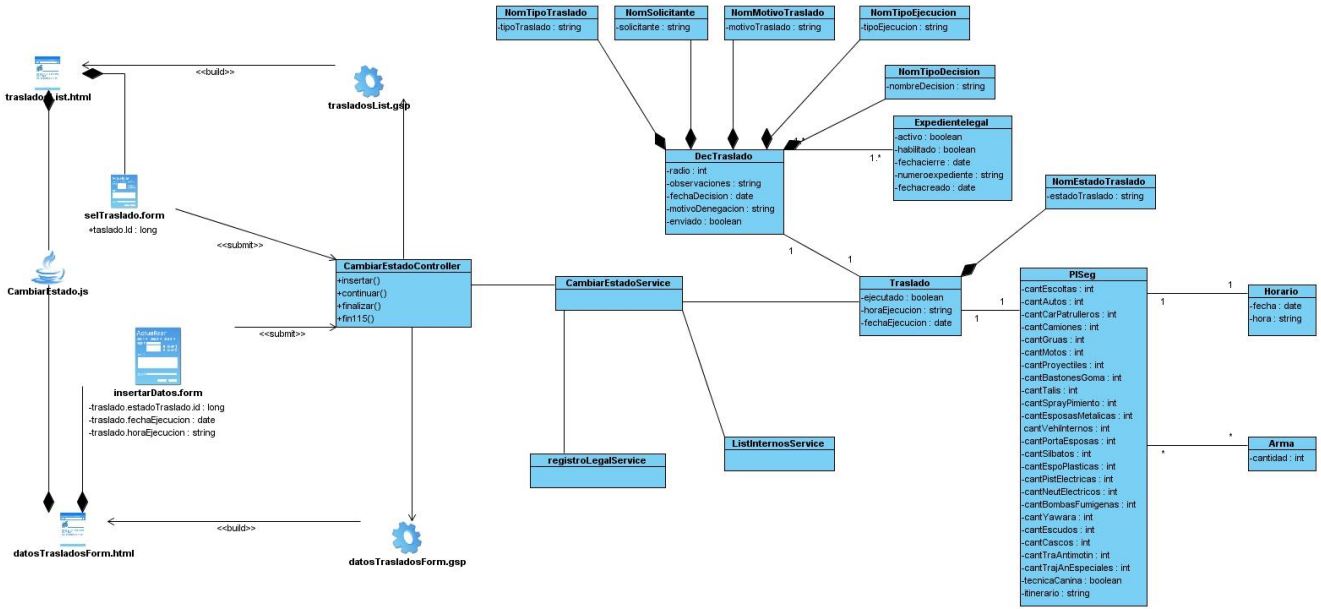


Figura 22: Diagrama de clases con estereotipos web del CU Registrar Ejecución.

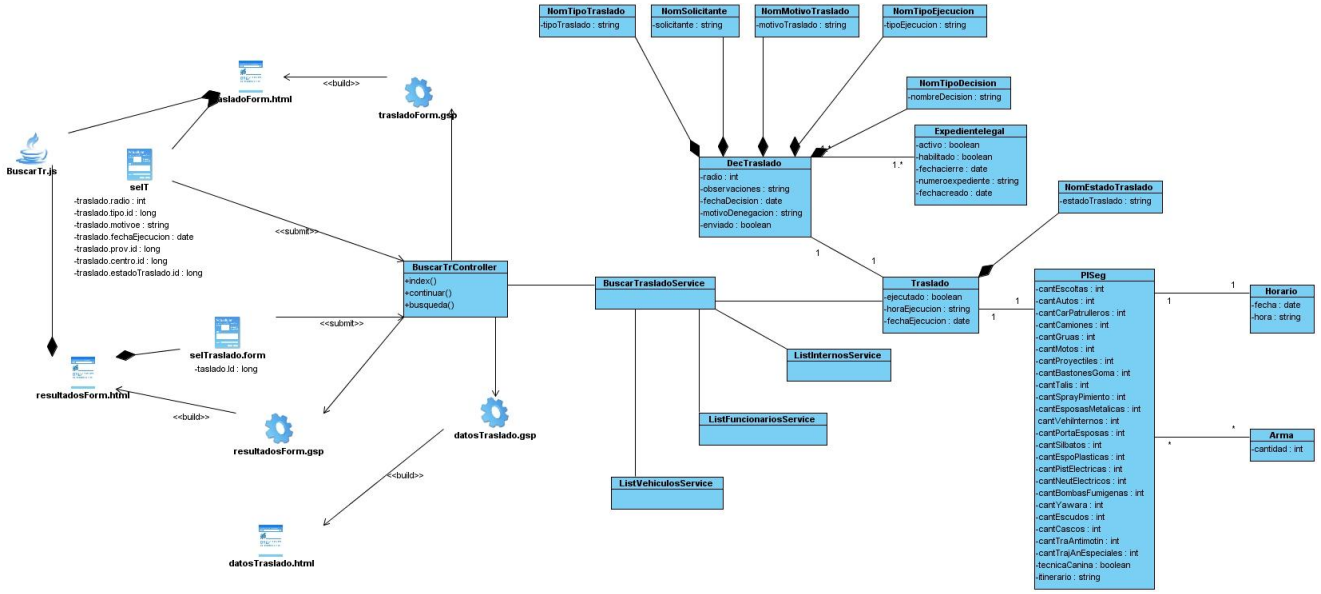


Figura 23: Diagrama de clases con estereotipos web del CU Consultar Traslados.

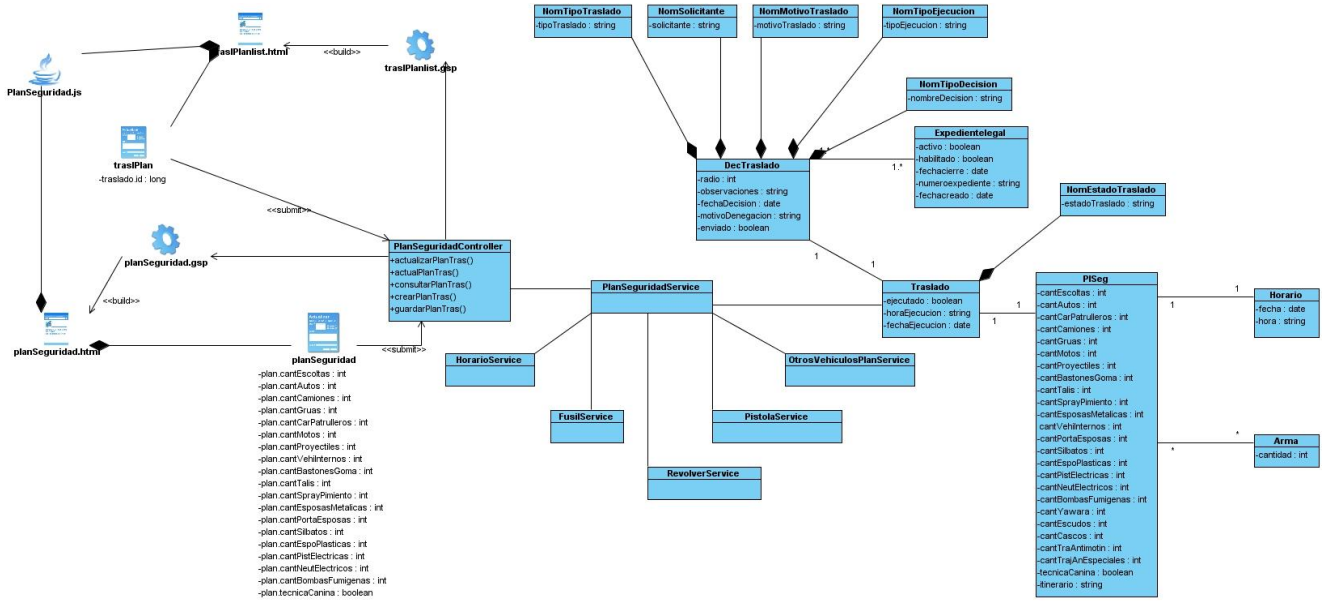


Figura 24: Diagrama de clases con estereotipos web de la sección 1 del CU CRUDD Plan de Seguridad.

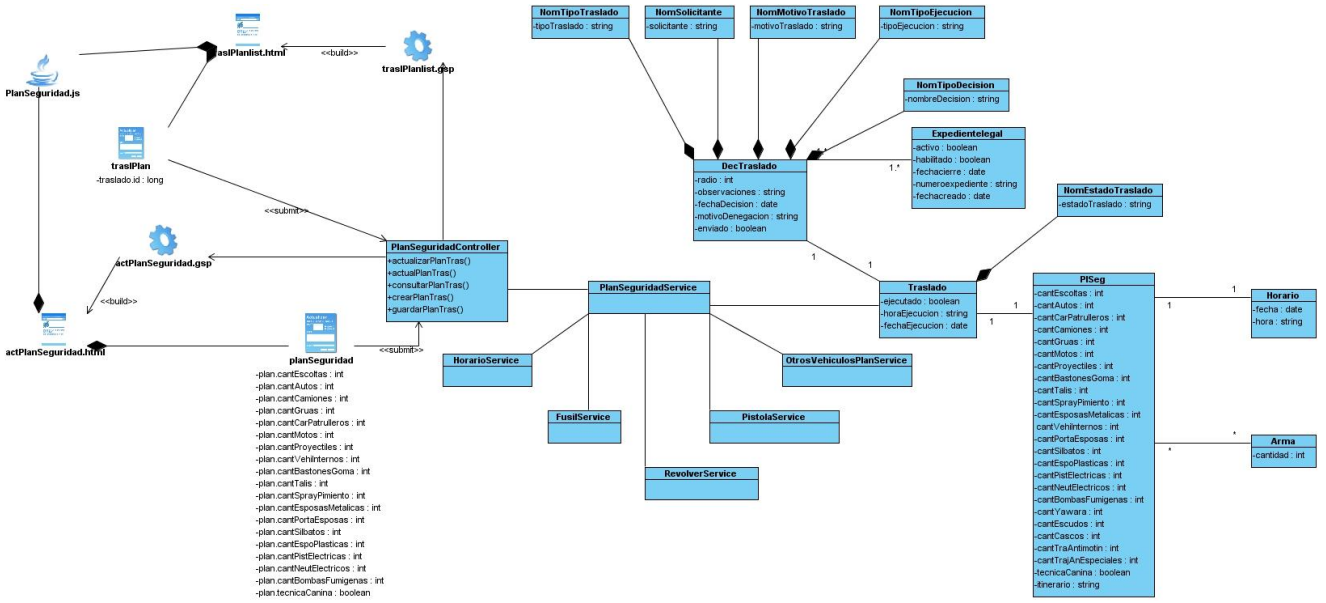


Figura 25: Diagrama de clases con estereotipos web de la sección 2 del CU CRUDD Plan de Seguridad.

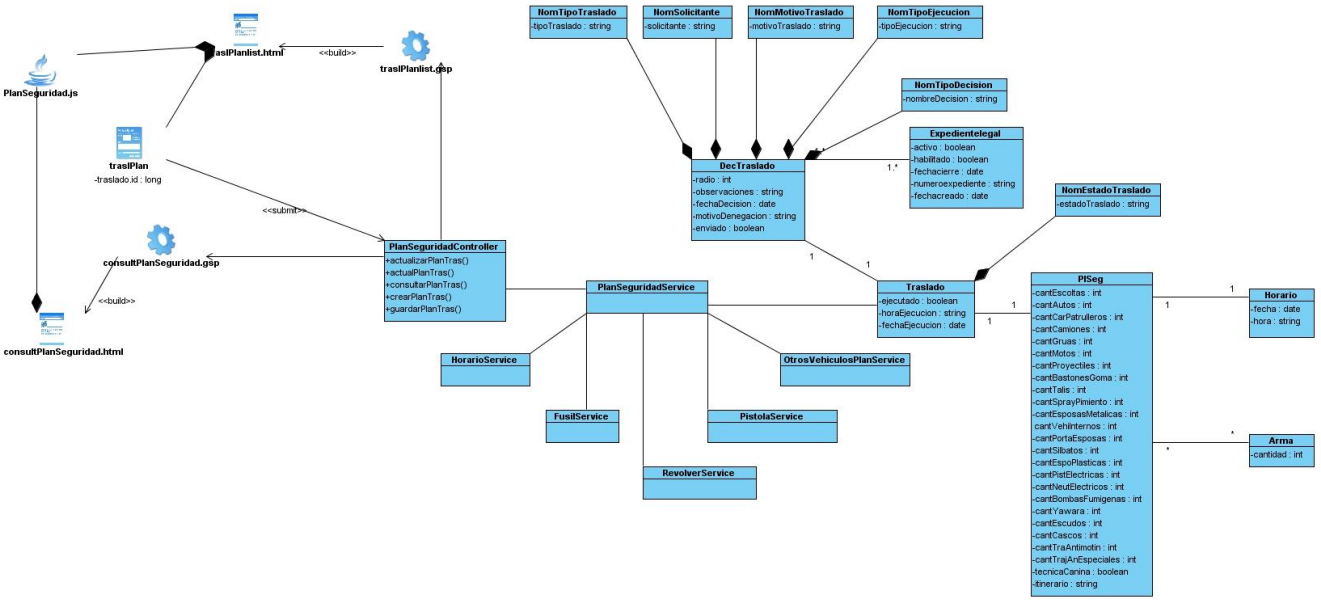


Figura 26: Diagrama de clases con estereotipos web de la sección 3 del CU CRUDD Plan de Seguridad.

Anexo 3: Diagramas de secuencia

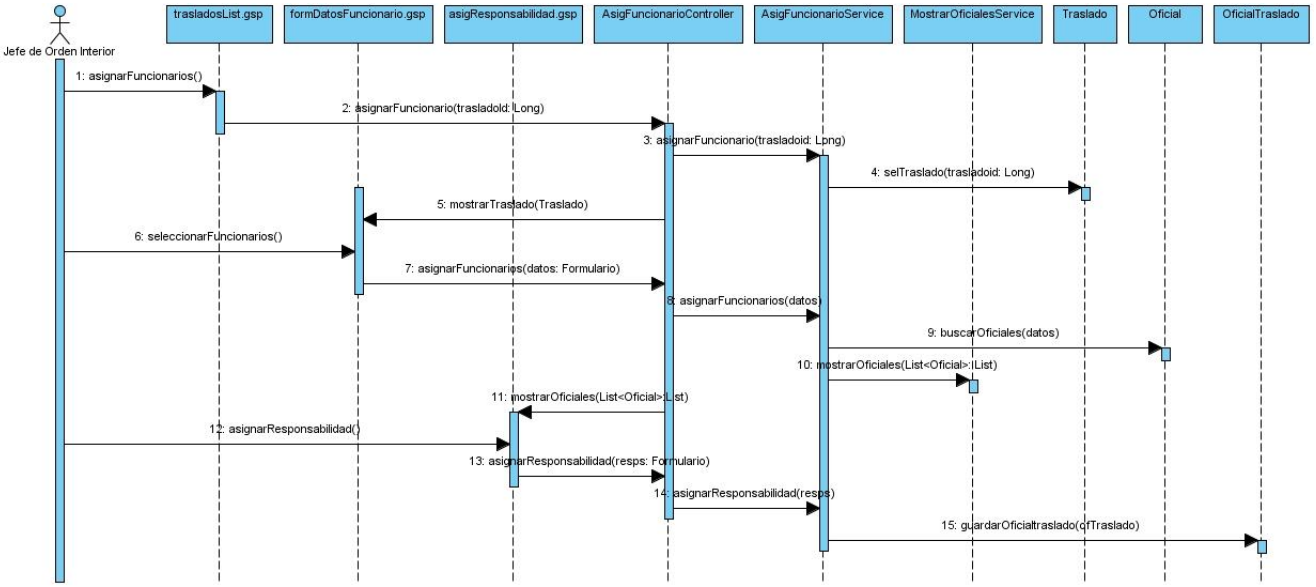


Figura 27: Diagrama de secuencia del CU Asignar Funcionarios.

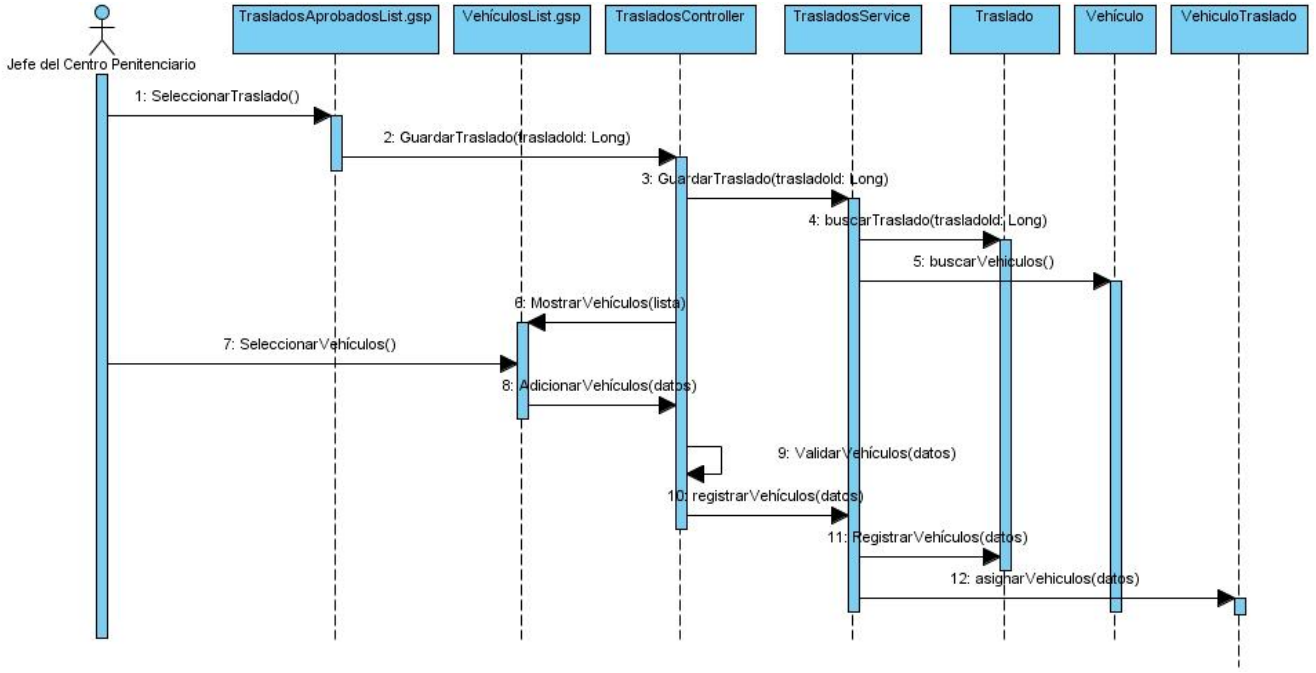


Figura 28: Diagrama de secuencia del CU Asignar Vehículos.

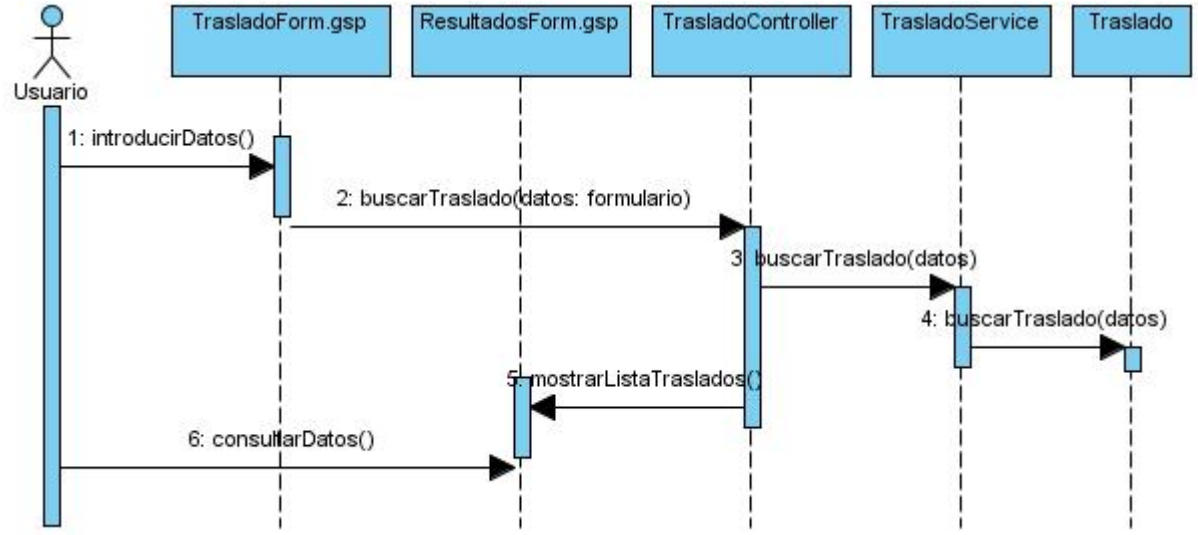


Figura 29: Diagrama de secuencia del CU Buscar Traslados.

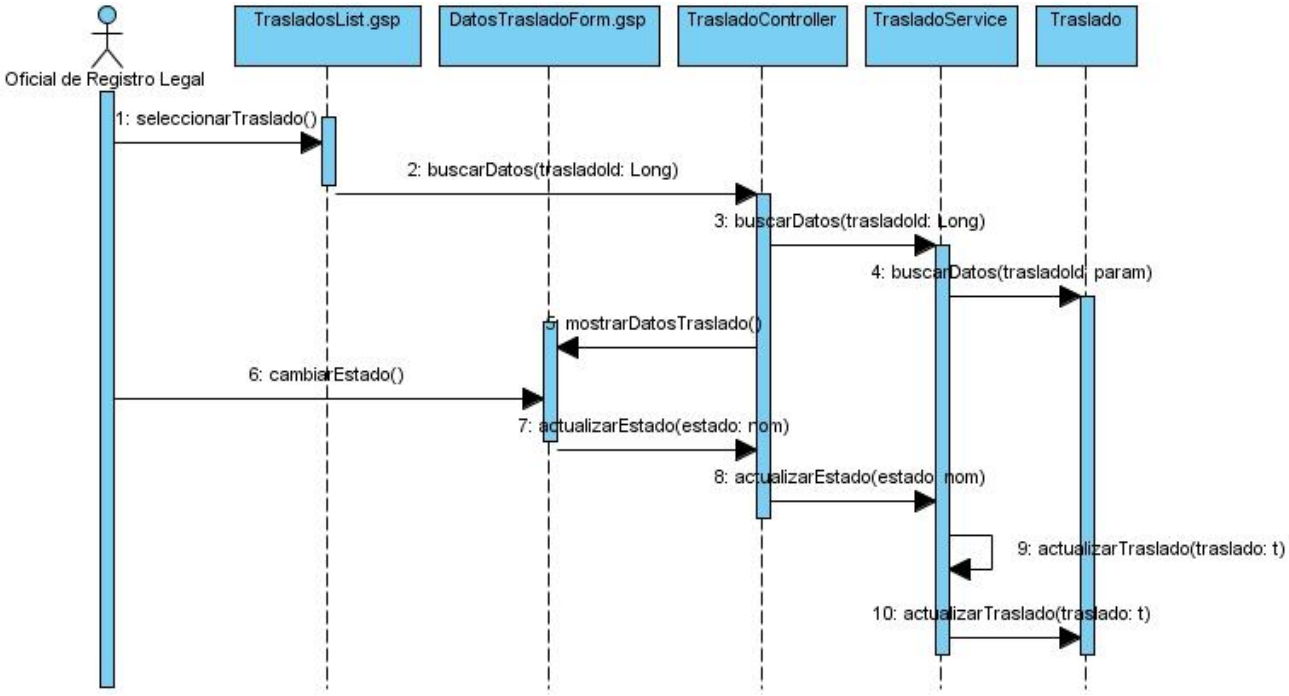


Figura 30: Diagrama de secuencia del CU Registrar Ejecución.

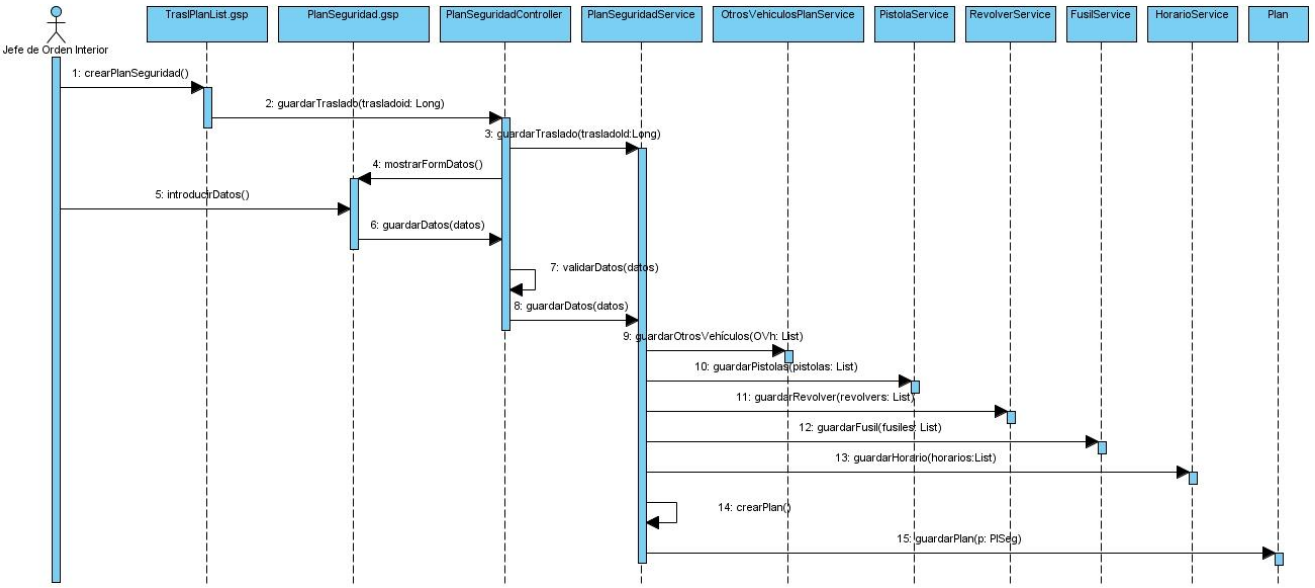


Figura 31: Diagrama de secuencia de la sección 1 del CU CRUDD Plan de Seguridad.

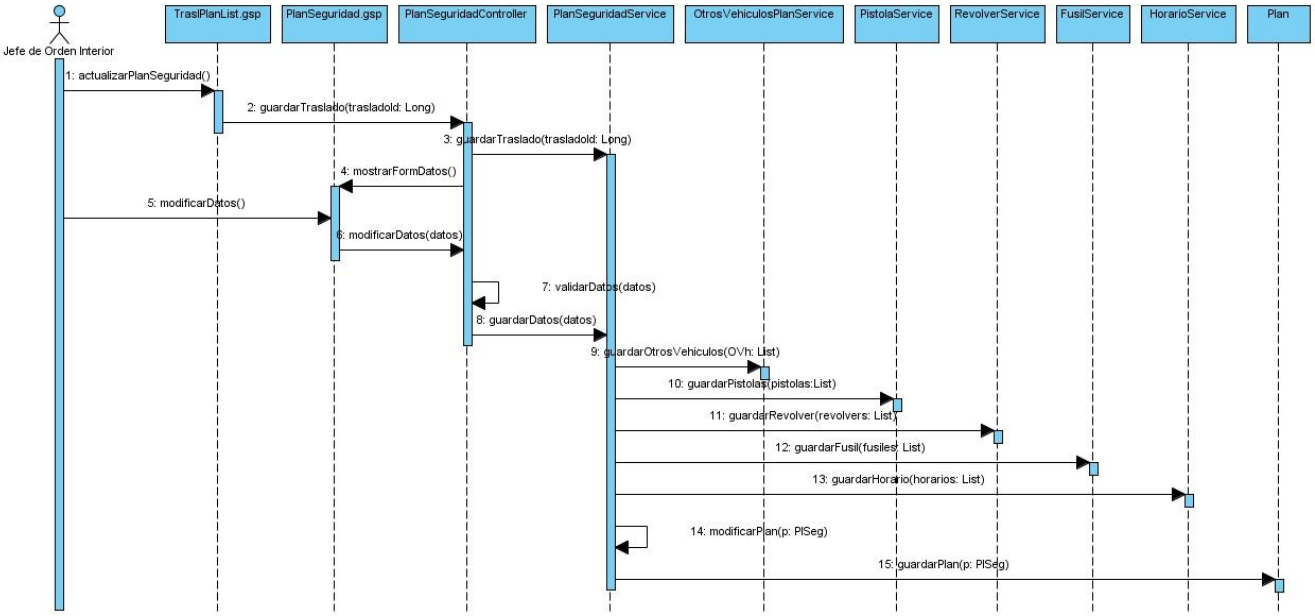


Figura 32: Diagrama de secuencia de la sección 2 del CU CRUDD Plan de Seguridad.

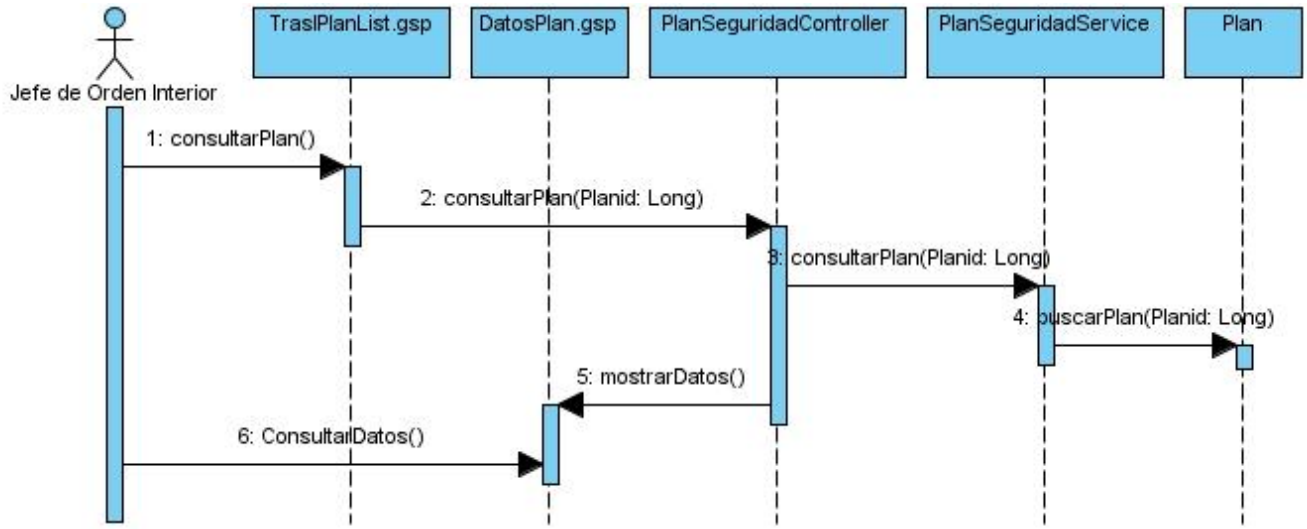


Figura 33: Diagrama de secuencia de la sección 3 del CU CRUDD Plan de Seguridad.

Anexo 4: Descripción de las tablas de la base de datos

Tabla 16: Descripción de la tabla OficialTraslado.

Nombre	Oficial_Traslado	
Descripción	Almacena los datos de los oficiales asignados a un traslado	
Atributo	**Tipo**	**Descripción**
TRASLADO_ID	NUMBER(19,0)	Guarda el id del traslado, número hasta 19 dígitos, no puede ser nulo
OFICIAL_ID	NUMBER(19,0)	Guarda el id del oficial, número hasta 19 dígitos, no puede ser nulo
RESPONSABILIDAD_ID	NUMBER(19,0)	Guarda el id del valor de NomResponsabilidad, no puede ser nulo

Tabla 17: Descripción de la tabla OficialTraslado.

Nombre	Vehículo_Traslado	
Descripción	Almacena los datos de los vehículos asignados a un traslado	
Atributo	**Tipo**	**Descripción**
TRASLADO_ID	NUMBER(19,0)	Guarda el id del traslado, número hasta 19 dígitos, no puede ser nulo
VEHICULO_ID	NUMBER(19,0)	Guarda el id del vehículo, número hasta 19 dígitos, no puede ser nulo

Tabla 18: Descripción de la tabla OficialTraslado.

Nombre	Traslado_Cancelado

Descripción	Almacena los datos de los traslados que han sido cancelado	
Atributo	**Tipo**	**Descripción**
TRASLADO_ID	NUMBER(19,0)	Guarda el id del traslado, número hasta 19 dígitos, no puede ser nulo
MOTIVO_NO_EJECUCION	VARCHAR2(255 BYTE)	Guarda el motivo por el cual no fue ejecutado

Tabla 19: Descripción de la tabla PlSeg.

Nombre	Pl_Seg	
Descripción	Almacena los datos de los planes de seguridad asociados a los traslados	
Atributo	**Tipo**	**Descripción**
CANT_AUTOS	NUMBER(19,0)	Guarda la cantidad de autos para el traslado
CANT_BASTONES_GOMA	NUMBER(19,0)	Guarda la cantidad de bastones de goma
CANT_BOMBAS_FUMIGENAS	NUMBER(19,0)	Guarda la cantidad de bombas fumígenas
CANT_CAMIONES	NUMBER(19,0)	Guarda la cantidad de camiones
CANT_CAR_PATRULLEROS	NUMBER(19,0)	Guarda la cantidad de carros patrulleros
CANT_CASCOS	NUMBER(19,0)	Guarda la cantidad de cascos
CANT_ESCOLTAS	NUMBER(19,0)	Guarda la cantidad de escoltas
CANT_ESCUDOS	NUMBER(19,0)	Guarda la cantidad de escudos

CANT_ESPO_PLASTICAS	NUMBER(19,0)	Guarda la cantidad de esposas plásticas
CANT_ESPOSAS_METALICAS	NUMBER(19,0)	Guarda la cantidad de esposas metálicas
CANT_GRUAS	NUMBER(19,0)	Guarda la cantidad de carros grúas
CANT_MOTOS	NUMBER(19,0)	Guarda la cantidad de motos
CANT_NEUT_ELECTRICOS	NUMBER(19,0)	Guarda la cantidad de neutralizadores eléctricos
CANT_PIST_ELECTRICAS	NUMBER(19,0)	Guarda la cantidad de pistolas eléctricas
CANT_PORTA_ESPOSAS	NUMBER(19,0)	Guarda la cantidad de porta-esposas.
CANT_PROYECTILES	NUMBER(19,0)	Guarda la cantidad de proyectiles
CANT_SILBATOS	NUMBER(19,0)	Guarda la cantidad de silbatos
CANT_SPRAY_PIMIENTO	NUMBER(19,0)	Guarda la cantidad de spray de pimienta
CANT_TALIS	NUMBER(19,0)	Guarda la cantidad de talis
CANT_TRA_ANTIMOTIN	NUMBER(19,0)	Guarda la cantidad de trajes antimotines
CANT_TRAJ_AN_ESPECIALES	NUMBER(19,0)	Guarda la cantidad de trajes antimotines especiales
CANT_VEHI_INTERNOS	NUMBER(19,0)	Guarda la cantidad de vehículos para trasladar los internos
CANT_YAWARA	NUMBER(19,0)	Guarda la cantidad de yawara

HORARIO_ID	NUMBER(19,0)	Guarda el id del horario asociado al plan de seguridad
ITINERARIO	VARCHAR2(255 BYTE)	Guarda el itinerario del traslado
TECNICA_CANINA	BOOLEAN	Guarda true si el plan de seguridad tiene asociada la técnica canina

Anexo 5: Diagramas de componentes

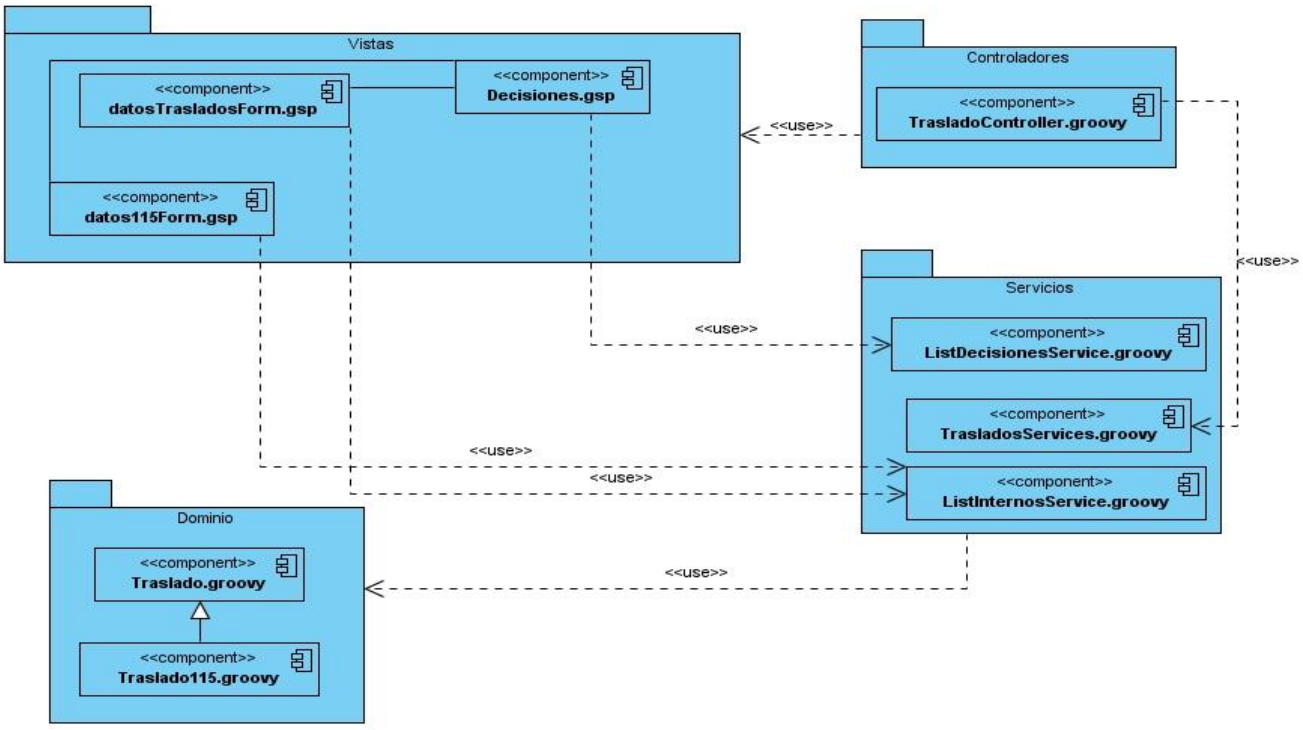


Figura 34: Diagrama de Componentes de Crear Traslado.

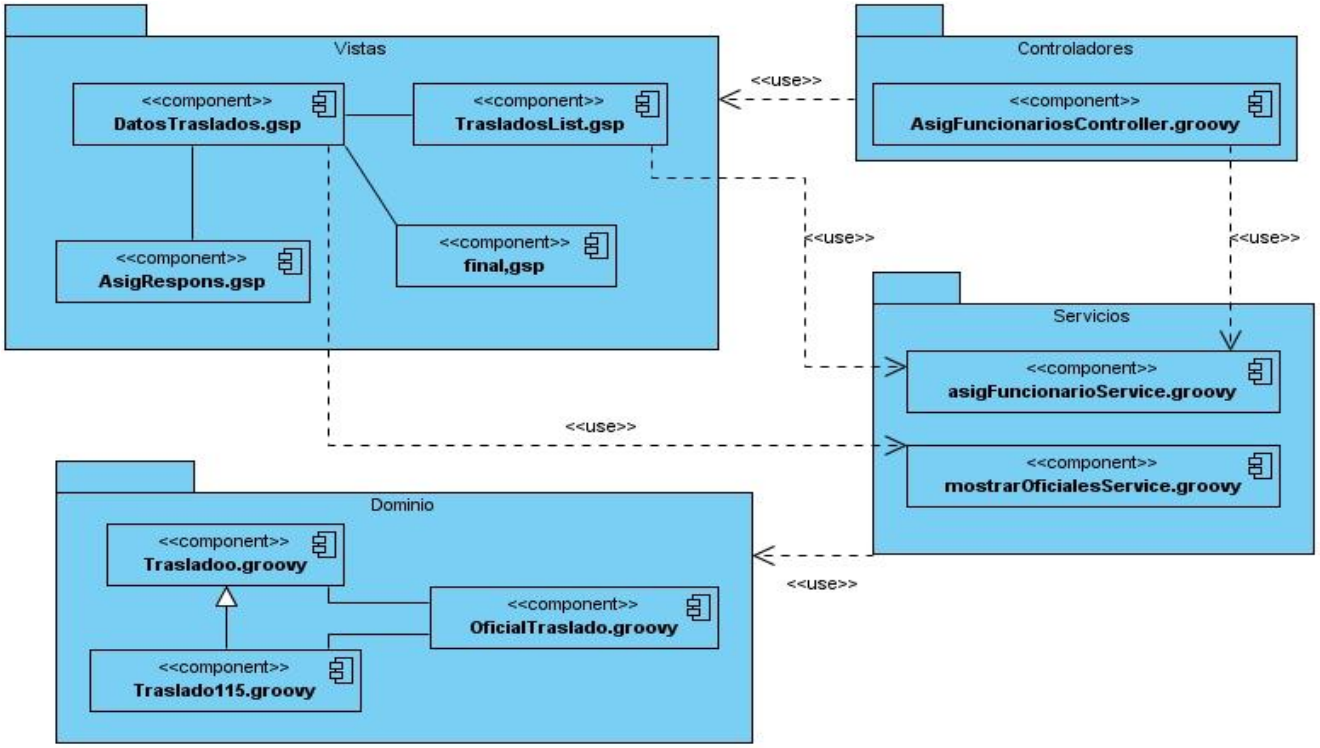


Figura 35: Diagrama de Componentes de Asignación de Funcionarios.

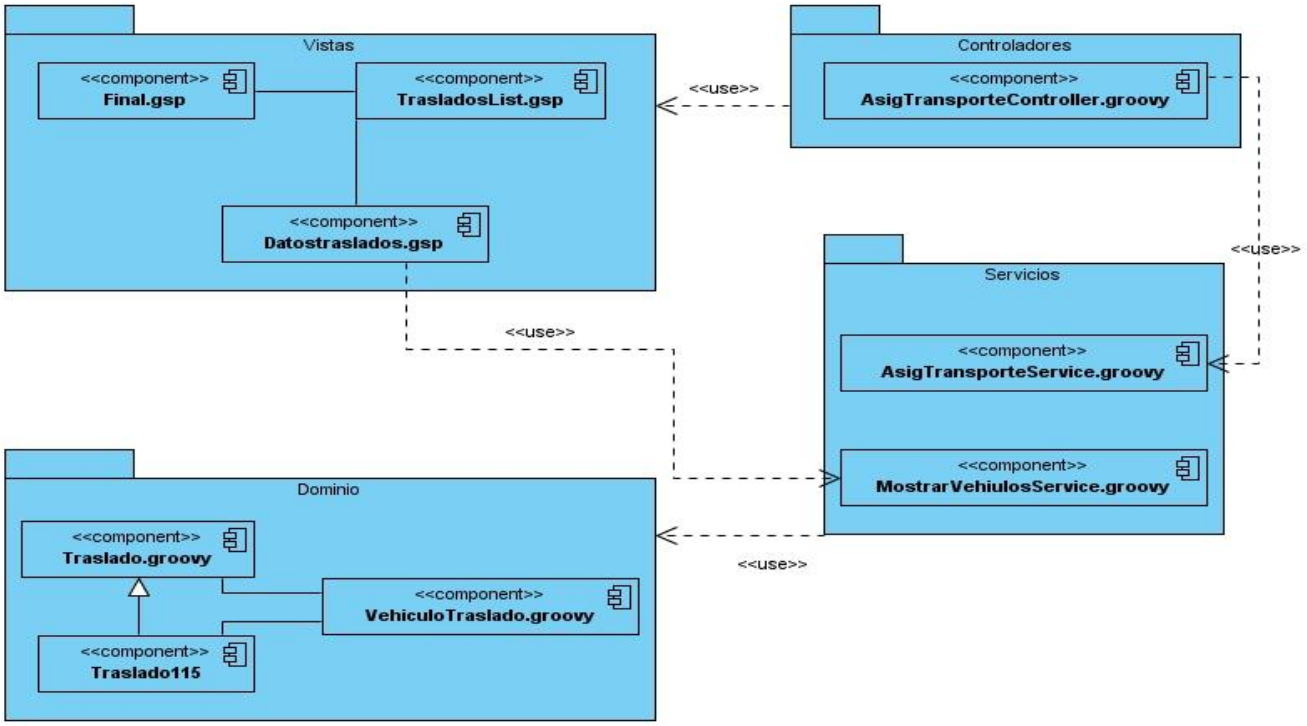


Figura 36: Diagrama de Componentes de Asignación de Vehículos.