República de Cuba

Universidad de las Ciencias Informáticas

Facultad 2

Sistema de Gestión Policial (SIGEPOL)

"Módulo Servicios de Vigilancia y Seguridad Privada"

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

Autores: Yaritza Rivera Ramírez

Eriberto Minguel Pantaleon

Tutor: Ing. Lisbet María González Bravo

"Año 53 de la Revolución"

La Habana, Cuba. Junio de 2011.

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 _____.

_____ _____

Yaritza Rivera Ramírez **Eriberto Minguel Pantaleon**

Autor **Autor**

_____

Ing. Lisbet María González Bravo

Tutor

Dedicatoria

Yaritza:

A mi mamá y mis hermanos que son los amores de mi vida y la razón de mi existencia.

Eriberto:

A mis padres, y a mi hermano que han estado apoyándome en todo momento, que han sido mi guía a lo largo de los años, por su paciencia, su confianza en mí, por cada granito de arena que pusieron en mi vida y en mi futuro para ustedes es este regalo.

Agradecimientos

Yaritza:

Quiero agradecerle de forma especial a mi mamita linda porque su amor incondicional ha sido el motor impulsor de mi vida, porque me ha dedicado su vida y siempre ha confiado en mí. Por darme su apoyo en todo momento y haberme enseñado desde pequeña a ser fuerte, independiente y a luchar por lo que quiero sin importar los obstáculo. En fin por demostrarme que no hay amor más grande que el de una madre. Te amo mima.

A mi hermano mayor "el hombre de mi vida" , por estar a mi lado , aconsejarme y preocuparse por mí en todo momento, por ser no solo hermano sino también mi más fiel amigo y confidente y por el buen ejemplo que me ha dado siempre. Te quiero mucho Tito.

A mi hermanita "la niñita de mis ojos", porque a pesar de ser menor que yo ha sabido escucharme, comprenderme y aconsejarme, por estar junto a mi tanto en mis victorias como en mis derrotas y por ser mi compañera de tantas locuras y diversiones. Te quiero Lila.

A mi mamá Nola (mi abuela), por darme su apoyo en todo momento y preocuparse por mí como una madre.

A mi compañero de tesis por la paciencia y el aguante que ha tenido conmigo.

De manera especial a mis amigas de la vocacional Mavita, Mady y Bety y que han permanecido siempre a mi lado, con las cuales he contado siempre pase lo que pase y les agradezco sobre todas las cosas porque me han demostrado que su amistad no es pasajera, sino que perdura y perdurara a través de los años.

Especialmente a "mis chicas" (Marvita, Yei, Zuly, Carmen, Dayi, Yona, Niusdin, Lisbet, Jany, Yilen), las cuales me han acompañado durante estos 5 años, gracias por aguantarme cada día y por haber sido mi segunda familia desde el principio. Las adoro a todas.

A mi "piquete querido" (Daile y Eve), las que llegaron último, pero que llegaron para quedarse, gracias por los momentos súper lindos que hemos pasado juntas, por preocuparse por mi y sobre todas las cosas por la amistad que me han brindado en todo momento. Las llevo en mi corazón.

A mis amigos Yaritzin, Yaineris, Rainer, Pedro, Tito y Capote por todos los momentos que hemos compartido. Muchas gracias.

Eriberto:

De manera general a todos aquellos que me han acompañado y que de una forma u otra han interactuado conmigo en estos años. Gracias.

Hoy es un día especial para mí, ya han pasado cinco cursos de sacrificios, sueños, alegrías y ha llegado el momento de enfrentarme al mundo como una profesional, son innumerables las personas a las que tengo que agradecerles por ayudarme a llegar aquí.

Primeramente agradecer a mis padres Ana y Eriberto que no pudieron estar presentes hoy aquí pero siempre me han apoyado y dado fuerzas en los momentos difíciles, a ellos que siempre han sabido sacar lo mejor de mí, brindándome todo su amor y cariño, desear mejores padres sería imposible.

A Yunior mi hermano por ser uno de los motivos que mas me impulsaron a seguir adelante, por ser mi mejor amigo desde hace mucho tiempo y estoy seguro que llegaras muy lejos.

A todos mis familiares, por la confianza depositada.

Yaritza gracias por ser mi compañera de tesis y gracias por la paciencia que me tuviste.

A mis amigos de Las Tunas que siempre has estado ahí para mí y a los que conocí luego en la Universidad, a mis compañeros de aula a lo largo de estos cinco años de estudios.

A mi piquete que siempre nos hemos llevado bien desde el primer día de clases Lester, Tito, Yoan y José.

Al equipo de desarrollo de SIGEPOL que nos ayudaron a que este trabajo pueda mostrar hoy sus resultados

A todos aquellos que de una forma u otra ayudaron a la realización de este trabajo.

A todos muchas Gracias.

Yaritza y Eriberto:

A nuestra tutora, por aclararnos las dudas, por su apoyo incondicional durante el desarrollo del presente trabajo de diploma, por su optimismo y por ser un gran ejemplo a seguir profesionalmente.

Resumen

La seguridad ciudadana es uno de los primordiales deberes desde el punto de vista social que todo país debe garantizar, de forma tal, que los ciudadanos queden satisfechos con la actuación de los cuerpos de seguridad ciudadana.

En la actualidad en la República Bolivariana de Venezuela existen muchos problemas de inseguridad social lo cual conlleva que un gran número de ciudadanos se sientan desprotegidos por parte del gobierno. Entre los órganos encargados de garantizar la seguridad ciudadana está el Viceministerio del Sistema Integrado de Policía (VISIPOL).

Debido a los problemas existentes con la seguridad ciudadana, el VISIPOL promueve realizar la aplicación web Sistema de Gestión Policial (SIGEPOL), que tiene como objetivo coordinar las acciones de los órganos policiales, para lograr la adecuación al nuevo modelo de actuación policial y ayudar a combatir el delito en la República Bolivariana de Venezuela.

Como parte del Sistema de Gestión Policial se define el Módulo de Servicios de Vigilancia y Seguridad Privada, debido a que es necesario gestionar la información de las empresas de la República Bolivariana de Venezuela que ofrecen Servicios de Vigilancia y Seguridad Privada.

Para el desarrollo de la aplicación web se emplea: RUP como metodología de desarrollo de software, Java como plataforma de desarrollo, Visual Paradigm como herramientas CASE y Springsource Tool Suite como IDE de desarrollo integrado para la codificación y gestión del código fuente en general, Grails como framework a utilizar y como lenguajes de programación Groovy y Java.

Índice

Índice de Figuras

Introducción

Uno de los problemas graves de la sociedad venezolana es la inseguridad ciudadana, según estudios realizados en 21 de los 24 estados de Venezuela el miedo se ha generalizado, al punto de que el 62 por ciento de las personas consultadas no se sienten seguras cuando están en su casa, 80 por ciento de ellas se consideran inseguras en la calle y el 83 por ciento viajan con miedo en el transporte público. (1)

Los homicidios son el más vistoso indicador de la inseguridad en este país de 27 millones de habitantes. En el 2009 se registraron 16.047 asesinatos, según un informe del Observatorio Venezolano de Violencia (OVV), que destaca la "*impunidad*" y "*corrupción*" como principales causas. (1)

La preocupación demostrada por el Estado por erradicar diversas causas y condiciones generadoras de delitos y de otras manifestaciones de conductas delictuales se expresan de forma clara y palpable en la implementación de un conjunto de políticas y misiones sociales aplicadas en estos últimos años.

Por esta razón el Comandante Hugo Rafael Chávez Frías, Presidente Constitucional de la República Bolivariana de Venezuela, a través del Reglamento Orgánico del Ministerio del Poder Popular para Relaciones Interiores y Justicia, según Decreto Nº 6.733 de fecha 09 de Junio de 2009 y publicado en la Gaceta Oficial de la República Bolivariana de Venezuela Nº 39.196 de fecha 09 de Junio 2009, creó el Viceministerio del Sistema Integrado de Policía (VISIPOL) que tiene como misión promover y alcanzar en el plano interno del Sistema Integrado de Policía una sinergia en el funcionamiento de sus procesos y en la interrelación de los diferentes Cuerpos de Policía y de Vigilancia y Seguridad Privada; y en el plano externo la armonía y complementación con los órganos que participan en la formulación y ejecución de políticas públicas y procedimientos que contribuyan al pacífico disfrute de los derechos de los ciudadanos y de las instituciones.

El VISIPOL, se encuentra dividido en tres niveles, el nivel de apoyo, el nivel sustantivo y el nivel desconcentrado. Dentro del nivel sustantivo se encuentra situada la Dirección General del Servicio de Policía, la Dirección General de Asistencia Técnica y la Dirección General de los Servicios de Vigilancia y Seguridad Privada (DIGESERVISP).

En el VISIPOL no existe un sistema que permita nutrirse de información relevante en el ámbito policial, es por ello que con la finalidad de ayudar a coordinar las acciones de los órganos policiales dicho viceministerio promueve la realización del Sistema de

Gestión Policial (SIGEPOL), para lograr la adecuación al nuevo modelo de actuación policial y ayudar a combatir el delito en la República Bolivariana de Venezuela.

El sistema está compuesto por los siguientes módulos:

- Módulo de Consejo Disciplinario.
- Módulo de Servicio de Policía.
- Módulo de Servicios de Vigilancia y Seguridad Privada.
- Módulo de Funcionario, Armamento y Dotación Policial.
- Módulo de Desviación Institucional.
- Módulo de Análisis de Información.
- Módulo de Administración.

Entre las principales problemáticas que presenta la Dirección General de los Servicios de Vigilancia y Seguridad Privada están:

El control de la información de la mayoría de los procesos que se llevan a cabo en esta dirección como son Protocolizar Actas Constitutivas, Renovar Permisos de Funcionamiento, Protocolizar Actas de Asambleas entre otros, se realizan de forma manual, lo que dificulta la eficiencia de los mismos. Además no cuenta con un registro único y automatizado de las evaluaciones de las supervisiones a las empresas que prestan servicio de vigilancia y seguridad privada.

El objetivo del Módulo de Servicios de Vigilancia y Seguridad Privada es solucionar las situaciones antes mencionadas, de ahí que se plantee el siguiente problema científico.

¿Cómo gestionar la información asociada a la creación, funcionamiento, supervisión y disolución de las empresas de vigilancia y seguridad privada, en la Dirección General de Servicio de Vigilancia y Seguridad Privada?

El objeto de investigación lo constituyen los procesos de control de los servicios de vigilancia y seguridad privada, siendo el campo de acción la supervisión y la regulación para la concesión de permisos de funcionamiento a las empresas de vigilancia y seguridad privada de la República Bolivariana de Venezuela.

Entre las principales funcionalidades del Módulo Servicios de Vigilancia y Seguridad Privada correspondiente al Sistema de Gestión Policial estarán:

- Registrar toda la información y solicitudes de otorgamiento y renovación de la concesión de permisos a personas jurídicas que prestan servicios de seguridad y vigilancia.
- Llevar el registro sistemático de informaciones resultantes de las acciones de supervisión, control y evaluación de las actividades de las personas jurídicas

que prestan servicios de vigilancia y seguridad privada; y de las medidas dictadas en caso de infracciones.

- Supervisar el personal, el armamento y los vehículos que pertenecen a las empresas que brindan servicio de vigilancia y seguridad privada.
- Invalidar o cancelar el permiso a las empresas que brindan servicio de vigilancia y seguridad privada ante el incumplimiento reiterado de las normativas que regulan la materia.

Para darle solución al problema planteado se definió como objetivo general desarrollar un módulo para la gestión de los Servicios de Vigilancia y Seguridad Privada del Sistema de Gestión Policial a partir del análisis de los requisitos de software haciendo uso de las tecnologías y metodologías definidas por el proyecto.

Para cumplir con el objetivo general trazado, se han definido como tareas de investigación:

1. Conocer los procesos que se llevan a cabo en la Dirección General de Servicios de Vigilancia y Seguridad Privada, para alcanzar un resultado acorde a las necesidades del cliente y que dicha solución incluya todos los procesos que ocurren en esta dirección.
2. Analizar las leyes que rigen el funcionamiento de la Dirección General de Servicios de Vigilancia y Seguridad Privada, para garantizar el cumplimiento de estas durante el diseño e implementación del sistema.
3. Valorar la metodología de desarrollo de software, plataforma, lenguaje y el conjunto de herramientas de desarrollo que conforman la línea base de la arquitectura, con el objetivo de garantizar el buen uso de estos recursos para obtener un producto final con mayor calidad.
4. Identificar y seleccionar los patrones de arquitectura y diseño más apropiados para la elaboración del producto de software, para garantizar un diseño robusto y flexible acorde a las necesidades del cliente.
5. Realizar el diseño del software a partir del análisis del negocio.
6. Definir la organización del código e implementar los elementos de diseño en términos de componentes de implementación.

Para cumplir con las tareas antes mencionadas se desarrollará un producto de software flexible, seguro, confiable y orientado a las necesidades del cliente acorde con las tecnologías de punta en el ámbito informático, las leyes venezolanas vigentes y los requerimientos necesarios.

El presente documento está compuesto por 4 capítulos:

En el Capítulo 1 **"Fundamentación del Tema"** se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema, además de analizar los principales procesos que conforman el Módulo de Servicios de Vigilancia y Seguridad Privada.

En el Capítulo 2 **"Metodología y Herramientas de Desarrollo"** se analizan las herramientas y lenguajes de programación fundamentando su uso para el desarrollo del Módulo de Servicio de Vigilancia y Seguridad Privada. Además se plantea la metodología a emplear en el desarrollo del mismo.

En el Capítulo 3 **"Diseño del Sistema"** se construye la solución propuesta, se modelan diagramas de clases que representan las funcionalidades del Módulo de Servicio de Vigilancia y Seguridad Privada aplicando los patrones de arquitectura y diseño seleccionados.

En el Capítulo 4 **"Implementación de la Solución"** se representa el diagrama de componentes que detalla la forma en que está estructurado el sistema, reflejando la transformación de los elementos del modelo del diseño en términos de componentes, ficheros que contienen código fuente, ejecutables, así como las dependencias entre ellos

Capítulo 1: Fundamentación del Tema

1.1 Introducción

Este capítulo está dedicado a investigar sobre la Seguridad Ciudadana en la República Bolivariana de Venezuela además de los procesos que se llevan a cabo en DIGESERVISP y exponer los fundamentos generales que sirvan de soporte teórico en la solución y concepción del problema.

1.2 Seguridad Ciudadana

Se entiende por seguridad ciudadana a "la acción integrada que desarrolla el Estado, con la colaboración de la ciudadanía, destinada a asegurar su convivencia pacífica, la erradicación de la violencia y la utilización pacífica de las vías y espacios públicos. Del mismo modo, contribuir a la prevención de la comisión de delitos y faltas". (2)

La seguridad ciudadana está encaminada a prevenir males como la violencia, las infracciones y el crimen; acciones que generan diversos peligros para la sociedad y para el desarrollo de las actividades económico-socio-cultural de la sociedad en general.

En la actualidad la seguridad ciudadana es uno de los grandes problemas que enfrentan las sociedades contemporáneas, su impacto en la vida de los ciudadanos hace que los gobiernos tomen medidas y realicen planes alternativos para disminuir los niveles de inseguridad entre la población.

Para incrementar la seguridad ciudadana se realizan un conjunto de acciones como es el dictado de leyes de seguridad ciudadana, que regulan los métodos y parámetros de actuación de los Órganos de Seguridad Ciudadana ante una situación, estos fueron creados para velar que se cumplan las regulaciones y las leyes establecidas.

1.2.1 Seguridad Ciudadana en Venezuela

La seguridad de los ciudadanos debe ser atendida de acuerdo al mandato establecido en el artículo 55 de la Constitución de la República Bolivariana de Venezuela, el cual establece "Toda persona tiene derecho a la protección del Estado, a través de los órganos de seguridad ciudadana regulados por ley, frente a situaciones que constituyan amenazas, vulnerabilidad o riesgos para la integridad física de las personas, sus propiedades, el disfrute de sus derechos y el cumplimiento de sus deberes". (3)

La cantidad de delitos que se cometen en esta nación de América del Sur ha originado gran incertidumbre, teniendo en cuenta la situación política vivida en estos últimos

años, generada por una oposición que busca crear caos para tratar de llegar al poder, lo cual ha influido para que el gobierno tome cartas en este asunto con la decisión y firmeza que se requiere para solventar el problema de la inseguridad.

Entre los principales problemas está el desempleo, la vivienda y el descontrol de los cuerpos de policías, pero el mayor es el nivel de inseguridad ciudadana que existe. En este marco nace el VISIPOL, adscrito al Ministerio del Poder Popular para las Relaciones Interiores y Justicia (MPPRIJ), el cual se instaló en el año 2009 con el propósito de adelantar una serie de políticas públicas que estandarizaran los cuerpos de policía para su adecuación al nuevo modelo policial, un modelo que se caracterice por el humanismo, la solidaridad, la participación ciudadana y el respeto a los derechos humanos. EL VISIPOL es el encargado de apoyar al Ministro o Ministra en la planificación, coordinación, formulación, fiscalización, control, evaluación y seguimiento de las políticas, planes, estrategias, lineamientos y directrices del Sistema Integrado de Policía que promuevan el desarrollo del Servicio de Policía. También se encarga de dar cumplimiento a las atribuciones asignadas al Ministerio, como órgano rector del Sistema Integrado de Policía, conformado por el Cuerpo de Policía Nacional, los Cuerpos de Policías Estadales y Municipales, la institución académica nacional especializada en seguridad, el FISPOL[1], el propio Ministerio con competencia en materia de seguridad ciudadana y cualquier otro órgano que determine el Ejecutivo Nacional. (4)

Para llevar adelante la construcción del nuevo modelo de policía que nace de la consulta y concertación ciudadana y del diagnóstico de las potencialidades y debilidades de la policía venezolana se hace necesario dotar al Viceministerio del Sistema Integrado de Policía de tecnología, equipamiento y Aplicaciones Informáticas que apoyen el desempeño policial, tributando así a la disminución de los índices delictivos y de violencia, por lo que se promueve la creación y puesta en marcha del Sistema de Gestión Policial con la finalidad de ayudar a coordinar las acciones de los órganos policiales.

1.3 Marco Legal o Seguridad Ciudadana según la Ley

Para establecer la normativa jurídica que sirve de base para la creación del Sistema de Gestión Policial del MPPRIJ de la República Bolivariana de Venezuela se tomará como referencia:

- Constitución de la República Bolivariana de Venezuela.
- Reglamento Orgánico del MPPRIJ.

[1] FISPOL: Órgano Intergubernamental del Servicio de Policía

- Decreto Ley 699.
- La Ley Orgánica del Servicio de Policía.
- La Ley Orgánica Cuerpo de Policía Nacional.
- Ley del Estatuto de la Función Policial

La Constitución de la República Bolivariana de Venezuela, publicada en Gaceta Oficial Extraordinaria No. 5 453 de la República Bolivariana de Venezuela en Caracas, viernes 24 de Marzo de 2000, expresa:

Artículo 9: Cuando resulte inminente el desbordamiento de la capacidad de respuesta del órgano actuante para controlar la situación, debido a su magnitud o complejidad de la misma, asumirá la responsabilidad de la coordinación y el manejo de ésta, el Órgano de Seguridad Ciudadana que disponga de los medios y la capacidad de respuesta para ello.

Artículo 18. El Consejo de Seguridad Ciudadana tendrá por objeto el estudio, formulación y evaluación de las políticas nacionales en materia de seguridad ciudadana.

Artículo 36: La coordinación nacional de seguridad ciudadana y las coordinaciones regionales de seguridad ciudadana, como administradores de las bases de datos de sus respectivas localidades, procesarán la información suministrada por los integrantes del sistema y generarán los reportes de información relacionados con el comportamiento de la acción delictiva, emergencias y situaciones de desastres.

Artículo 322: Establece la organización de los Órganos de Seguridad Ciudadana, como medio para garantizar la protección de los ciudadanos y sus hogares en el disfrute de los derechos fundamentales, incorpora la creación de instituciones e instrumentos legales que permitan abordar integral y eficazmente la problemática de la inseguridad ciudadana.

La Ley del Estatuto de la Función Policial tiene por objeto regir las relaciones de empleo público entre los funcionarios y funcionarias policiales y los cuerpos de policía de la Administración Pública nacional, estatal y municipal, lo cual comprende:

1. El sistema de dirección y de gestión de la Función Policial y la articulación de la carrera policial.
2. El sistema de administración de personal, el cual incluye la planificación de recursos humanos, procesos de reclutamiento, selección, ingreso, inducción, educación y desarrollo, planificación de la carrera, evaluación de meritos, ascensos, traslados, transferencias, valoración y clasificación de cargos,

jerarquías, escalas de remuneraciones y beneficios , permisos, licencias y régimen disciplinario.

3. Los derechos, garantías y deberes de los funcionarios y funcionarias policiales en sus relaciones de empleo público.

Garantizar la seguridad ciudadana es una función del Estado que se ejerce en los ámbitos Nacional, Estatal y Municipal. El Decreto con Rango, Valor y Fuerza de Ley Orgánica del Servicio de Policía y del Cuerpo Policía Nacional, que se propone, tiene su origen en el artículo 332, numeral 1 de la Constitución de la República Bolivariana de Venezuela.

La designación de un Órgano Rector del servicio de policía proporcionara un mejor ordenamiento, control y proyección de la función policial en los diferentes ámbitos político-territoriales. Y finalmente la homogeneización de los diversos Cuerpos de Policía en aspectos clave para su funcionamiento, como son principios de trabajo, capacitación, respeto a los derechos humanos, rendición de cuentas, indicadores del desempeño, uso de la fuerza, entre otros, contribuirán a elevar su efectividad y eficiencia en el cumplimiento de sus misiones.

Para dar cumplimiento al Reglamento Orgánico del MPPRIJ, la Ley Orgánica del Servicio de Policía y la Ley Orgánica Cuerpo de Policía Nacional se definen las siguientes tareas:

1. Ejecutar las políticas públicas y planes en materia de seguridad ciudadana en el espectro policial y velar por su ejecución.
2. Diseñar y formular políticas integrales en lo que respecta a procedimientos y actuaciones de los cuerpos de policía.
3. Regular, coordinar, supervisar y controlar la correcta prestación del servicio de policía.
4. Establecer los lineamientos administrativos, funcionales y operativos, conforme a los cuales se organizan los cuerpos de policía.
5. Fijar, diseñar, implementar, controlar y evaluar las políticas, estándares, planes, programas y actividades relacionadas con la prestación del servicio de policía.
6. Establecer un sistema único de expedición de credenciales a las funcionarias y funcionarios de los cuerpos de policía.
7. Ejercer el control de desempeño y evaluación de los cuerpos de policía, de acuerdo con estándares que defina el Órgano Rector.
8. Regular, supervisar, controlar, autorizar o revocar las actividades de las personas jurídicas que prestan servicios de vigilancia y seguridad privada.

9. Supervisar, promover y coordinar la implementación de las medidas establecidas en el ordenamiento jurídico inherente a los servicios de vigilancia y seguridad privada.

1.4 Gestión de las Empresas de Vigilancia y Seguridad Privada en Venezuela

La actividad de las empresas que brindan servicios de vigilancia y seguridad es controlada desde el VISIPOL, esta puede concebirse como la disposición de fuerzas que se especializan en brindar servicios de vigilancia y seguridad y responden a una entidad privada que cobra por sus servicios.

Actualmente en la República Bolivariana de Venezuela no existe una manera de tener un estricto control de la creación, mantención y disolución de las empresas que brindan servicios de vigilancia y seguridad privada, lo cual trae consigo las siguientes problemáticas:

Se realiza un control manual de la información recolectada en diferentes formatos con la finalidad de otorgar, renovar, denegar o revocar la concesión de permisos a personas jurídicas que prestan servicios de vigilancia y seguridad privada, lo que dificulta la agilidad del proceso para garantizar el debido funcionamiento de estas empresas, según la normativa legal vigente.

No se cuenta con un registro único y automatizado de las evaluaciones de la prestación de los servicios de vigilancia y seguridad privada, que permita velar por el cumplimiento efectivo y eficaz de esta actividad.

1.5 Propuesta del Proceso Automatizado de la Dirección General de Servicios de Vigilancia y Seguridad Privada.

DIGESERVISP se encarga de llevar el control de las empresas que brindan servicios de vigilancia y seguridad privada, así como también lleva a cabo un conjunto de trámites realizados por dichas empresas. Para los representantes de las empresas realizar estos trámites tienen que presentar la información necesaria en dependencia del trámite a realizar a DIGESERVISP donde la Secretaria es la encargada de recibir los documentos y enviarlos al Revisor.

Una vez que el Revisor posea los documentos, analiza los recaudos y determina si dichos documentos están correctos, en caso favorable el trámite es enviado al Jefe de Grupo que se encargará de asignar dicho trámite al analista encargado de revisar la solicitud, en caso de que los documentos estén incorrectos se devuelven al representante especificándole los errores encontrados.

Luego, el Analista verifica que los documentos presentados estén acordes con las reglas establecidas en el Decreto Ley 699 para este tipo de trámite y a raíz de esto decide si el trámite es procedente o improcedente. En caso de que no proceda se elaboran las observaciones detectadas que no son más que crear una comunicación al representante de la empresa informándole las razones por las cuales su solicitud fue denegada y en caso contrario se puede llevar a cabo cualquiera de los siguientes procesos:

Protocolización de Acta Constitutiva: en caso de que el trámite sea procedente se crea una comunicación para el registrador mercantil y para el representante de la empresa informándole la aprobación de la solicitud presentada y se registran los datos básicos de la empresa. Al culminar dicho proceso la empresa quedará constituida y solo serán necesarios los permisos de funcionamiento para comenzar a brindar los servicios de vigilancia y seguridad privada.

Autorizar Permiso de Funcionamiento: estos permisos son necesarios para que las empresas puedan brindar los servicios para los cuales fueron creadas. En caso de que el analista determine que la solicitud en procedente se crea una comunicación para consultoría donde se revisa el trámite y se realiza el informe de revisión y oficio de aprobación. Si la solicitud cumple con los requisitos se registra su aprobación y se procede a realizar la inspección constatando la existencia de las instalaciones y el debido cumplimiento con los requisitos exigidos en las leyes y reglamentos respectivos, en caso contrario se registra que fue denegada la solicitud. Seguidamente si la inspección fue satisfactoria se crea el Resuelto Ministerial y la Providencia Administrativa que son los documentos que otorgan el permiso de funcionamiento de la Sede Principal autorizando a prestar los servicios por un período de un (1) año, y se le notifica la aprobación de la solicitud presentada al representante de la empresa. De no haber sido conforme la inspección entonces se le comunica al representante de la empresa las razones por las cuales su solicitud fue denegada.

En el caso de que la empresa quiera extender su funcionamiento más allá del período de un (1) año debe realizar el proceso **Renovar Permiso de Funcionamiento**: en el cual, si los documentos presentados por el representante son aprobados se realiza una inspección constatando la existencia de las instalaciones y el debido cumplimiento con los requisitos exigidos en las leyes y reglamentos respectivos y si dicha inspección es conforme entonces se crear la Providencia Administrativa autorizando a la empresa a renovar el permiso de funcionamiento de los servicios aprobados por un período de un (1) año, tanto para sede principal como para sus sedes sucursales y también le notifica al representante de la empresa que fue aprobada la solicitud presentada. En el

caso de que la inspección no sea conforme entonces se le comunica al representante de la empresa las razones por las cuales su solicitud fue denegada.

Cuando el representante desee realizar algún cambio en la empresa entonces se procede a **Protocolizar Acta de Asamblea:** en dicho proceso si los documentos presentados por el representante son aprobados entonces se le comunica al registro mercantil y al representante de la empresa la aprobación de la solicitud presentada y se registran los datos referentes a los puntos tratados en el acta de asamblea.

Al tener constituida la empresa los representantes de estas pueden ampliar sus servicios a otros estados y para esto es necesario realizar el proceso **Autorizar Permiso de Funcionamiento de Sucursal**: en caso de que los documentos presentados por los representantes cumplan con los requisitos establecidos se registra la aprobación de la solicitud presentada. Posterior a esto se realizar una inspección a dicha empresa constatando la existencia de las instalaciones y el debido cumplimiento con los requisitos exigidos en las leyes y reglamentos respectivos y en caso de que esta inspección sea conforme se crea la Providencia Administrativa que otorga el permiso de funcionamiento de la sede sucursal autorizando a prestar los servicios por un período de un (1) año y se crea una notificación al representante de la empresa informándole la aprobación de la solicitud presentada. En caso de existir algún problema con el documento o la inspección no sea conforme entonces se le comunica al representante de la empresa las razones por las cuales su solicitud fue denegada.

Todos los ciudadanos venezolanos tienen el derecho de presentarse en DIGESERVISP para realizar denuncias hacia alguna empresa que presta servicio de vigilancia y seguridad privada en este caso se hace necesario el proceso **Tramitar Denuncia**: en este caso se almacenan los datos de la denuncia, los datos del denunciante y la sede a la cual se le realizó la denuncia. Posteriormente se realiza una inspección a la empresa denunciada para verificar el debido cumplimiento con los requisitos exigidos en las leyes y reglamentos respectivos. Luego se registra el resultado de las denuncias formuladas y que serán analizadas por DIGESERVISP a partir de las inspecciones realizadas a la empresa denunciada.

Realizar Inspección: estas pueden ser planificadas, resultado de una queja hacia alguna empresa o para otorgar el Permiso de Funcionamiento a la empresa. Primeramente se planifica la inspección donde quedan registrados los datos de la empresa y el motivo de la inspección, luego se prepara la inspección a realizar a una empresa registrando la información relevante de la misma así como también el motivo de la inspección. Posteriormente se elabora la providencia administrativa a partir de la

información registrada en la preparación de la inspección de la empresa y se genera automáticamente el número de providencia administrativa. Luego se elabora el acta de inspección donde se registran los datos referentes a la inspección realizada a una empresa y en caso de que no se encuentren deficiencias culmina dicho proceso. En caso de haber encontrado algún problema se elabora el Acta de Requerimiento la cual incluye el listado de documentos a subsanar por el representante debido a irregularidades detectadas durante la inspección. El representante de le empresa tendrá un lapso de 30 días para subsanar dichas irregularidades detectadas durante la inspección. En caso de no haber acudido a subsanar los errores se registran los datos asociados a la suspensión de los servicios que presta la empresa.

Para finalizar los servicios prestados por las empresas es necesario el proceso **Extinguir Empresa:** en el cual se reciben los documentos requeridos para realizar la solicitud de extinción de empresa y posteriormente se registra la extinción de la autorización de los servicios que presta dicha empresa culminando así su funcionamiento.

Para una mejor comprensión de estos procesos ver Anexos.

1.6 Conclusiones

En este capítulo se realizó un análisis de los principales conceptos que deben conocerse referente al marco legal en que se apoya el VISIPOL para el trabajo con las empresas que brindan servicios de vigilancia y seguridad privada, además se explican los procesos que se van a automatizar en términos de negocio.

Capítulo 2: Metodología y Herramientas de Desarrollo

2.1 Introducción

En este capítulo se describe la metodología de desarrollo de software utilizada en la solución, además de las diferentes herramientas usadas en la misma, explicando de cada una de ellas las características fundamentales por las cuales fueron elegidas por parte del equipo de desarrollo.

2.2 Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Las metodologías de desarrollo de software contribuyen a mejorar la calidad y a realizar un software en el tiempo esperado y con el coste estimado.

Van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, proponen qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (5)

Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. Las metodologías indican cómo hay que obtener los distintos productos parciales y finales. Aspectos positivos de éstas son:

- Son interactivas e incrementales.
- Fácil de dividir el sistema en varios subsistemas independientes.
- Se fomenta la reutilización de componentes.

Existen las llamadas Metodologías Tradicionales, conocidas también como metodologías pesadas, las cuales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada, RUP es un ejemplo de dichas metodologías el cual fue escogido para la

elaboración de la solución debido a sus importantes características descritas seguidamente.

2.2.1 Metodología RUP

Para controlar, planificar, organizar y guiar el desarrollo del Módulo de Servicio de Vigilancia y Seguridad Privada se decidió utilizar como metodología de desarrollo del software RUP (Proceso Unificado de Desarrollo de Software).

RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software. Esta metodología está pensada para proyectos grandes en cuanto a tiempo y recursos como es el caso del sistema que se propone; el cual cuenta además con un equipo de desarrollo inestable, pues en su mayoría son estudiantes y profesores que pueden pasar a desarrollar otras funciones en cualquier momento, es por esto que se necesitará una buena organización y abundante documentación para que el proyecto continúe avanzando. Usando esta metodología no se necesita tener al usuario final como parte del equipo de desarrollo, lo cual es muy importante para el desarrollo del sistema ya que el cliente no puede formar parte de dicho equipo debido a que proceden de otro país.

2.2.2 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (6)

Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos, permite la modelación de sistemas con tecnología orientada a objetos. Los diagramas son entes importantes de UML, cuya finalidad es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo. El modelo gráfico de UML tiene un vocabulario en el que se identifican: elementos, relaciones y diagramas.

En el desarrollo de la aplicación se usa UML como lenguaje de modelado por la experiencia que posee el equipo de desarrollo en el trabajo con dicho lenguaje, lo cual contribuyó a disminuir grandemente el tiempo dedicado a modelar el sistema. Además, que la herramienta CASE Visual Paradigm utiliza UML como lenguaje de modelado.

2.3 Herramienta CASE

Las herramientas CASE[2] son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero. Las herramientas CASE permiten organizar y manejar la información de un proyecto informático. Permite que los sistemas (especialmente los complejos, en el caso de SIGEPOL), se tornen más flexibles, más comprensibles y además mejoran la comunicación entre los participantes, con el uso de las herramientas CASE se logra un formato estándar para toda la información referente a la gestión del proyecto.

Estas herramientas pueden servir de apoyo en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, compilación automática, documentación o detección de errores entre otras. (7)

2.3.1 Visual Paradigm para UML

Visual Paradigm para UML es una herramienta CASE profesional, fácil de utilizar y que soporta el ciclo de vida completo del desarrollo de software; usa al UML como lenguaje de modelado lo cual ayuda a la construcción de aplicaciones con calidad, de manera intuitiva y a menor coste; además de que facilita la comunicación entre los miembros del equipo de desarrollo al garantizar el uso de un lenguaje estándar común. (8)

Se decidió utilizar esta herramienta CASE para el modelado del diseño de la aplicación, ya que entre sus principales características y facilidades se encuentran:

- Es una herramienta libre.
- Fácil de instalar, actualizar y usar.
- Soporta el ciclo de vida completo del desarrollo de software.
- El equipo de desarrollo posee experiencia con esta herramienta.
- Integración con distintos Ambientes de Desarrollo Integrados (IDE): se integra fácilmente con varios IDEs de desarrollo, entre los que se encuentra Springsource Tool Suite, que es el que se utilizará para el desarrollo de la aplicación.

2.4 Entorno de Desarrollo

Un entorno de desarrollo integrado (IDE) es un programa informático compuesto por un conjunto de herramientas de programación que a su vez puede dedicarse en

[2] Ingeniería de Software Asistida por Computadoras

exclusiva a un sólo lenguaje de programación o utilizarse para varios lenguajes. Un IDE 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. (9)

2.4.1 Plataforma Java

Una plataforma de desarrollo es el entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones, la plataforma Java es un entorno o plataforma de computación creada por la Sun Microsystems[3], capaz de ejecutar aplicaciones desarrolladas con el uso del lenguaje de programación Java y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino una máquina virtual encargada de la ejecución, y un conjunto de librerías estándares que ofrecen funcionalidades comunes.

Para el desarrollo de la aplicación se seleccionó la Edición Empresarial: J2EE o Java EE (Java Platform, Enterprise Edition).

Una de las principales razones por la que se seleccionó esta plataforma es porque se cuenta con limitaciones legales por parte del cliente: el decreto Ley No. 3.390 de la Gaceta Oficial de la República Bolivariana de Venezuela establece que la Administración Pública Nacional empleará prioritariamente Software Libre Desarrollado con Estándares Abiertos en sus Sistemas, Proyectos y Servicios Informáticos, lo cual permite el uso de otro tipo de software solo en casos de no ser posible la adquisición de Software Libre Bajo Estándares Abiertos, en tal caso, los órganos y entes de la Administración Pública Nacional deberán solicitar ante el Ministerio de Ciencia y Tecnología, la autorización para adoptar otro tipo de soluciones bajo normas y criterios establecidos por ese Ministerio. (10)

J2EE es independiente del sistema operativo, es una plataforma madura y con una amplia gama de proveedores y documentación. Además, el usuario no necesitará de requerimientos de hardware ni de software para acceder al sistema, solo se debe equipar con buen hardware el servidor de aplicaciones.

El motivo principal por el cual el equipo de desarrollo escogió la Plataforma Java como entorno de desarrollo es porque el lenguaje utilizado para elaborar la solución es Groovy y el mismo solo puede ser usado sobre dicha plataforma.

[3] Es una empresa informática de Silicon Valley, fabricante de semiconductores y software. Las siglas SUN se derivan de «Stamford University Network», proyecto que se había creado para interconectar en red las bibliotecas de la Universidad de Stamford.

Los componentes principales de la plataforma son la Máquina Virtual de Java (JVM) y el lenguaje de programación Java.

2.4.2 Lenguaje de Programación Java

Java fue diseñado en 1990 por James Gosling, de Sun Microsystems, como software para dispositivos electrónicos como calculadoras, microondas y la televisión interactiva. (11)

Este lenguaje nace bajo la filosofía de la Programación Orientada a Objetos (OOP) facilitando así abordar la resolución de cualquier tipo de problema, además esto permite lograr una mayor reutilización del código y una forma más eficiente de producir software. Además cabe mencionar que el equipo de desarrollo posee experiencia en el uso de este lenguaje al ser usado en el desarrollo de versiones anteriores del sistema que aquí se propone, esta característica influyó considerablemente a la hora de decidir su uso, pues contribuyó al ahorro de tiempo al no ser necesario el uso de cursos de capacitación para los miembros del proyecto.

2.4.3 Lenguaje de Programación Groovy

Groovy 1.0 apareció el 2 de enero de 2007. Después de varias versiones beta y otras tantas candidatas a release, el 7 de diciembre de 2007 surgió la versión Groovy 1.1 que finalmente fue renombrada a Groovy 1.5 con el fin de notar la gran cantidad de cambios que había sufrido con respecto a la versión 1.0. En diciembre de 2009 se publicó la versión 1.7. (12)

Este lenguaje usa una sintaxis muy parecida a Java y 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. Todo lo anterior unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy hacen que este lenguaje sea de muy fácil adopción para programadores Java; la curva de aprendizaje se reduce mucho en comparación con otros lenguajes que generan bytecode para la JVM, tales como Jython o JRuby. Groovy puede usarse también de manera dinámica como un lenguaje de scripting. (12)

Entre sus principales características encontramos:

- El código es totalmente compatible con Java.
- Tipado estático y dinámico.
- Closures.

- Sobrecarga de operadores.
- Manipulación de listas y mapas.
- Soporte para expresiones regulares.
- Expresiones embebidas dentro de strings.

Este lenguaje fue seleccionado por sus innovadoras características además de que permite obtener un producto con alta calidad en un tiempo menor que otros lenguajes como C# y al unirse con Grails forman una potente herramienta de desarrollo web. En la aplicación descrita Groovy es utilizado para crear propiedades y métodos dinámicos que permiten acelerar y facilitar el desarrollo del sistema.

2.4.4 Máquina Virtual de Java (JVM)

El término JVM se refiere a la especificación abstracta de una máquina de software para ejecutar programas Java. Es el núcleo de ese lenguaje de programación, por lo que resulta imposible ejecutar cualquier programa Java sin la ejecución de alguna implantación de la JVM, o sea, el código no se ejecuta directamente sobre un procesador físico, sino sobre un procesador virtual Java. Es la encargada de traducir los bytecode[4] en las instrucciones nativas, permitiendo la portabilidad de las aplicaciones. (13)

2.5 Framework

En el desarrollo de software, un marco de trabajo (framework en inglés) es una estructura de soporte definida sobre la cual un proyecto puede ser organizado y desarrollado; puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre aplicaciones para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Para desarrollar una aplicación Web es necesario tener en cuenta que esta filosofía se basa en el estilo arquitectónico Cliente-Servidor, por lo que es necesario definir ambos comportamientos.

2.5.1 Dojo

Para darle soporte a toda la gestión de información y al manejo de páginas dinámicas que llegan a los clientes, se plantea para el lado del servidor la utilización de este marco de trabajo, debido a las potencialidades fundamentadas a continuación.

[4] Código resultante de la compilación del código fuente

Dojo contiene APIs [5]y controles para facilitar el desarrollo de aplicaciones Web que utilicen el modelo AJAX. Con la utilización de Dojo se gana en facilidad y agilización del proceso de desarrollo de este tipo de aplicaciones; permite reutilizar código y promueve buenas prácticas de desarrollo y resuelve problemas comunes de compatibilidad entre navegadores y proporciona una gama más amplia de opciones en una sola biblioteca que otros framework del mismo tipo.

Otros de los principales motivos que propiciaron el uso de Dojo por parte del equipo de desarrollo es la experiencia poseída con este framework, y además que cuenta con un mecanismo de creación de componentes lo suficientemente flexible como para modificar los existentes y permitir crear nuevos componentes en correspondencia con las necesidades del proyecto. De dichos componentes los más significativos para el desarrollo del módulo son:

tabs: permiten incluir en una misma vista un conjunto de tabs que me permitirán ver la información de varias vistas sin moverme de página, es muy usado cuando se requiere mostrar en una vista un gran volumen de información.

address: Este componente mediante el uso de una sola línea de código hace posible cargar los componentes necesarios para que el usuario de la aplicación introduzca los datos referentes a una dirección, este puede configurarse de manera que los elementos que incluye varíen en dependencia de lo que necesita el usuario.

functionary: Permite cargar un funcionario y consiste en abrir una ventana emergente al usuario mostrando en su parte derecha todos los funcionarios autenticados en el sistema y en la parte izquierda un conjunto de campos que permiten filtrar los funcionarios para lograr una búsqueda más sencilla.

table: Este componente es el más usado a la hora de mostrar información de la base de datos ya que permite no solo insertar una tabla sino que se acopla con otro componente llamado "action", el cual incluye en una tupla de la tabla las acciones para eliminar, editar o ver detalles.

Además de los componentes antes mencionados también fueron creados los componentes necesarios para crear los formularios de las vistas, entre los más usados están los datePicker, hiddenField, textArea, select, radio y los textField.

[5] Interfaz de Programación de Aplicaciones

2.5.2 Grails

El futuro del desarrollo web está bien claro y se basa en el principio de reutilización de código para aumentar la productividad y disminuir los riesgos de desarrollo. Basado en estos principios surge Grails, un entorno para desarrollo de aplicaciones web sobre la plataforma Java Enterprise Edition. (14)

Grails disminuye el tiempo necesario en tareas repetitivas y permite que un cambio de requisitos no requiera reescribir toda la aplicación, porque la mayoría del código de infraestructura se genera de forma dinámica mientras la aplicación se está ejecutando.

Desde el punto de vista del diseño, Grails se basa en dos principios fundamentales:

Convención mejor que configuración

En lugar de tener que escribir interminables archivos de configuración en formato XML, Grails se basa en una serie de convenciones para que el desarrollo de la aplicación sea mucho más rápido y productivo. Por ejemplo, todas las clases de la carpeta grails-app/controllers serán tratadas como Controladores, y se mapearán convenientemente a las urls de la aplicación. (14)

DRY: Don't repeat yourself (¡No te repitas!)

La participación de Spring Container en Grails permite la inyección de dependencias mediante el patrón IoC (Inversión of Control), de forma que cada actor en la aplicación deba definirse una única vez, haciéndose visible a todos los demás de forma automática. Esto permite ahorrar gran tiempo y evitar errores que puedan surgir a la hora de definir actores del sistema. (14)

Grails permite al programador olvidarse de gran parte de la configuración típica que incluyen los frameworks Modelo Vista Controlador (MVC). Además se aprovecha de un lenguaje dinámico como Groovy para acortar los tiempos de desarrollo y simplemente dejarlos en escribir código, actualizar, testear y depurar fallos. Esto hace que el desarrollo de la aplicación sea mucho más ágil que con otros frameworks MVC.

Entre sus principales características encontramos:

- Se basa en el patrón MVC.
- Soporta Ajax.
- Trata los modelos como clases del dominio.
- Permite insertar contenido dinámico en las vistas o Groovy Server Page (GSP).
- Genera la Base de Datos a partir del dominio.
- Utiliza HTML como lenguaje en las vistas.

Grails está compuesto por un conjunto de elementos esenciales, a continuación se describen los más usados:

GORM

El modelo de datos en una aplicación Grails está compuesto por las clases del dominio, que se ubican en la carpeta grails-app/domain. Grails utiliza GORM (*Grails Object Relational Mapping*), un gestor de persistencia escrito en Groovy, para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos que son creados en tiempo de ejecución para cada clase del dominio que facilitan enormemente las búsquedas.

GORM está construido sobre **Hibernate**, una herramienta de mapeo objeto-relacional, que se encarga de relacionar las clases del dominio con tablas de una base de datos, y las propiedades de las clases con campos en las tablas, además garantiza que cada operación que sea realizada sobre los objetos del modelo de datos será traducida por Hibernate a las sentencias SQL necesarias para quedar reflejado en la base de datos. (14)

GORM proporciona los métodos de búsqueda y modificación que permiten manipular las clases del dominio, además agrega un campo id a la clase y se encargara de generar un valor único para cada instancia de la clase logrando una mayor sencillez en la búsqueda de datos específicos.

Este gestor de persistencia también se encarga de mapear las asociaciones entre clases del dominio, estas asociaciones pueden ser del tipo uno-a-uno, uno-a-muchos y muchos-a-muchos garantizando que al eliminar un dato especifico, este se elimine en cascada y evitando así posibles errores como errores por violación de restricciones de clave foránea. Además del mapeo explicado GORM también se encarga de mapear herencia y composiciones.

Una vez definido el modelo de datos GORM propone un conjunto de herramientas que permiten realizar actualizaciones o insertar datos referentes a una clase del dominio en la base datos mediante el uso de los métodos save() y delete(), los cuales son muy útiles para trabajar con elementos de la base de datos.

GORM permite restringir los valores que pueden asignarse a las propiedades de cada clase del dominio, mediante una propiedad estática con el nombre constraints. Dichos constraints se utilizarán también a la hora de crear el esquema de datos, por ejemplo, en función del tamaño de un String Hibernate decidirá si crear un campo VARCHAR o LONGTEXT y para la generación de vistas mediante Scaffolding. Además de lo antes

expuesto GORM también se encarga de verificar estas restricciones a la hora de guardar información en la Base de Datos. (14)

Plugins

Para facilitar el trabajo Grails dispone de una arquitectura de Plugins con una comunidad de usuarios detrás que ofrecen Plugins para seguridad, AJAX, testeo, búsqueda, informes y servicios web. Este sistema de Plugins hace que añadir complicadas funcionalidades a nuestra aplicación se convierte en algo muy sencillo.

2.6 IDE

SpringSource Tools Suite (STS) está basado en Eclipse, pero además incorpora muchas herramientas y asistentes cuyo objetivo es el de facilitar y agilizar el desarrollo de aplicaciones, las cuales obviamente utilicen tecnologías como Spring Framework y Spring Web Flow, pero que también permite desarrollar aplicaciones por muy sencillas que parezcan. (15)

STS es usado principalmente en el desarrollo del sistema debido a que se integra fácilmente con el lenguaje de programación Groovy y con el framework Grails utilizados en el desarrollo del sistema propuesto. Además, mediante el uso de plugins permite el trabajo con un repositorio central donde se encuentra toda la aplicación, de esta forma los desarrolladores poseen una copia local del sistema permitiendo que solo tengan que preocuparse por subir y actualizar los cambios realizados al sistema.

2.7 Gestor de Base de Datos Oracle 10g release 2

Oracle es un sistema de gestión de base de datos relacional desarrollado por Oracle Corporation[6].

Se considera como uno de los sistemas de bases de datos más completos, y las principales característica de este que lo convierten en una buena opción es que cuenta con un equipo de respaldo el cual dará solución a cualquier problema con la base de dato en un corto período de tiempo, además incluye un conjunto de herramientas que hacen que sea más fácil la integración con otros sistemas que usen Oracle como es el caso de los sistemas con los que trabajará Sigepol.

El uso de este gestor de base de datos permite el manejo de grandes volúmenes de información lo cual es de gran importancia debido a que el sistema que se propone debe llevar el control de todos los trámites realizados por las empresas que brindan

[6] Oracle Corporation es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión.

servicios de vigilancia y seguridad privada siendo necesario registrar un gran cúmulo de información relacionada con dichas empresas.

2.8 Conclusiones

En este capítulo se realizó un análisis de las herramientas y tecnologías a utilizar en la elaboración de la aplicación, exponiendo los factores que impulsaron al equipo de desarrollo a seleccionar RUP como metodología, UML como lenguaje de modelado y Visual Paradigma for UML como herramienta CASE. Además, también se fundamentan las razones por las cuales se seleccionó como parte del entorno de desarrollo: la Plataforma Java, la Maquina Virtual de Java , el lenguaje de programación Java, el lenguaje de programación Groovy, frameworks como Dojo y Grails, SpringSource Tool Suit como IDE y Oracle 10g release 2 como gestor de base de datos.

Capítulo 3: Diseño del Sistema

3.1 Introducción

En este capítulo se realiza el diseño del Módulo Servicios de Vigilancia y Seguridad Privada. Se presentan los patrones de diseño utilizados en la arquitectura del sistema así como los patrones arquitectónicos seguidos. Se describe la arquitectura del SIGEPOL, se presenta el diagrama de paquetes que se propone para estructurar los elementos del módulo. Se muestran además los diagramas de clases de cada uno de estos paquetes divididos por capas, y la descripción de las principales clases del diseño.

3.2 Patrones de diseño y de arquitectura.

"Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software", ya que estos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Es importante destacar que un patrón no es en general una solución en forma de código directamente, sino una descripción de cómo resolver el problema y ante qué circunstancias es aplicable. (16)

Para la confección de los diagramas de clases del diseño del módulo se emplearon un grupo de patrones que se describen a continuación.

3.2.1 Patrones de arquitectura

Modelo-Vista-Controlador (MVC)

Es un patrón de arquitectura de software que permite separar la complejidad de un sistema en tres componentes elementales (Modelo, Vista y Controlador), de esta manera, es posible separar los datos que se manejan en el sistema (modelo), de su presentación al usuario (vista) y de la lógica de control de los eventos (controlador), facilitando la extensibilidad y desacoplamiento.

El patrón MVC es usado debido a que la arquitectura de Grails se basa en dicho patrón dotando dicha arquitectura de todas las fortalezas que el mismo brinda. Este patrón se encuentra situado en la capa de Presentación donde las vistas presentan el modelo en un formato adecuado para interactuar con el usuario, el modelo lo constituyen los datos que se obtienen en el controlador invocando los métodos del negocio necesarios y actualiza la vista solicitada por el usuario.

Tres Capas (Layers)

El modelo tres capas permite desglosar la aplicación en partes lógicas que favorecen a una mejor organización en el desarrollo de la misma. Este estilo es fácilmente acoplable con el MVC. La arquitectura quedaría separada en lo relacionado con la gestión de interfaz (presentación), negocio y acceso a datos, además de las clases del dominio, en el caso de SIGEPOL este patrón se evidencia a la hora de trabajar con los servicios, los controladores y las vistas. El trabajo con dichas capas se evidencia de la siguiente forma las vistas o interfaces son las encargadas solamente de captar y mostrar información, los controladores se encargan de atender los eventos de la interfaz de usuario, invocar al negocio (servicios) según la acción lanzada desde la interfaz, esperar la respuesta del negocio y actualizar la interfaz.

3.2.2 Patrones de diseño

Inversión de Control (IoC)

La inversión de control es un patrón utilizado en Grails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste sólo contenga la lógica necesaria para hacer su trabajo. (16)

Cuando creamos un componente en nuestra aplicación, Grails configura Spring para que gestione su ciclo de vida (cuándo se crea, cuántas instancias se mantienen vivas a la vez, cómo se destruyen, etc.) y sus dependencias (qué otros componentes necesita para realizar su trabajo y cómo conseguirlos). (16)

El objetivo de esta técnica es mantener los componentes lo más sencillos que sea posible, incluyendo únicamente código que tenga relación con la lógica de negocio, y dejar fuera todo el código de fontanería. Así, la aplicación será más fácil de comprender y mantener. (16)

El uso de este patrón es muy importante para el desarrollo del sistema en cuestión, pues el mismo es usado principalmente cuando se crea una instancia de un servicio dentro de un controlador, esto permite que el desarrollador no tenga que preocuparse por gerstionar el ciclo de vida de dicho servicio, lo cual ahorra tiempo y disminuye la posibilidad de cometer errores por parte del equipo de desarrollo a la hora de crear intancias de otras clases.

Avanzados: GOF (Gang of Four)

Instancia única o Solitario (Singleton)

El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia, este patrón es principalmente usado cuando

se declara una instancia de un servicio en un controlador, esta instancia tendrá un acceso total a todas las acciones que puede realizar dicho servicio y además puede ser utilizada en cualquier parte del controlador donde fue declarada. El uso de Singleton permite una mejor organización del código al existir una única instancia por clase.

3.2.3 Patrones de Aplicaciones Empresariales

Template View

La mejor manera de trabajar no es componer la página web con contenido dinámico como lo hace una página estática, sino incluir marcadores que se pueden resolver en las llamadas para obtener información dinámica desde la parte estática de la página que actúa como una plantilla para una respuesta en particular.

Este patrón permite escribir la presentación en la estructura de la página e incrustar marcadores en ella para indicar dónde incluir el contenido dinámico, facilita mucho el desarrollo al permitir incluir contenidos dinámicos dentro de las vistas. En el caso de SIGEPOL este patrón está presente a la hora de incluir la etiqueta ${nombre variable} dentro de una vista, se usa principalmente a la hora de trabajar con los componentes (<sigepol:select>), tablas y otros elementos que muestran información desde la base de datos.

Front Controller

Con el uso de este patrón, se garantiza que exista un único objeto que recibe todas las peticiones provenientes de varias vistas, acto seguido se encarga de procesar las solicitudes. Esto permite centralizar toda la gestión de las solicitudes en un único objeto, ganando en organización.

En el caso del sistema que se propone, este patrón se encuentra en los controladores principales que engloban los procesos de varias vistas como es el ejemplo del controlador EmpresaController, el cual se encarga de recibir todas las solicitudes de las interacciones del usuario referente a la gestión de las empresas que prestan servicios de vigilancia y seguridad privada en la República Bolivariana de Venezuela.

Data Transfer Object

En varias ocasiones desde una vista enviamos objetos que pertenecen a varias clases del dominio, en estos casos se crea una clase que se encarga de capturar estos objetos y enviarlos de manera independiente a la vista.

En el caso del sistema que se describe el patrón Data Transfer Object está presente a la hora de crear los command ya que estos son los encargados de capturar los objetos cuando no existe una entidad que los pueda tratar, la función principal del command es poder trabajar con varias clases del dominio al mismo tiempo de forma sencilla y rápida.

3.3 Descripción de la Arquitectura

Establecer las bases sólidas que soporten la construcción, escalabilidad, correcto funcionamiento y mantenimiento de una aplicación informática, requiere un estudio detallado de los patrones seguidos a lo largo del desarrollo del software, las buenas prácticas de programación, el entorno de desarrollo de la misma, las prestaciones que ha de satisfacer y los requerimientos necesarios para su puesta en marcha. La descripción de la arquitectura define los principales aspectos tomados en cuenta en el diseño de un sistema. La arquitectura de SIGEPOL está organizada desde un enfoque de capas.

Modelo Basado en Capas

Una capa es una agrupación lógica de un conjunto de clases acopladas entre sí. Desde el punto de vista de la ingeniería del software, la división de un sistema en capas facilita el diseño modular (cada capa encapsula un aspecto concreto del sistema) y permite la construcción de sistemas débilmente acoplados (si se minimiza las dependencias entre capas, resultará más fácil sustituir la implementación de una capa sin afectar al resto del sistema). Además, el uso de capas también posibilita la reutilización y el desarrollo en paralelo. El Módulo de Servicios de Vigilancia y Seguridad Privada posee un conjunto de clases agrupadas jerárquicamente en capas (ya sea de presentación, de negocio o de acceso a datos), en dependencia de las funcionalidades que brinda cada una de ellas. Cada capa para completar su funcionamiento delega tareas a la capa que se encuentra por debajo en la jerarquía, brindándole los datos necesarios para las mismas o solicitando de ella la información requerida.

A continuación se muestra la siguiente figura donde se representan los componentes más importantes de la arquitectura del sistema, posteriormente se explican cada una de sus partes.

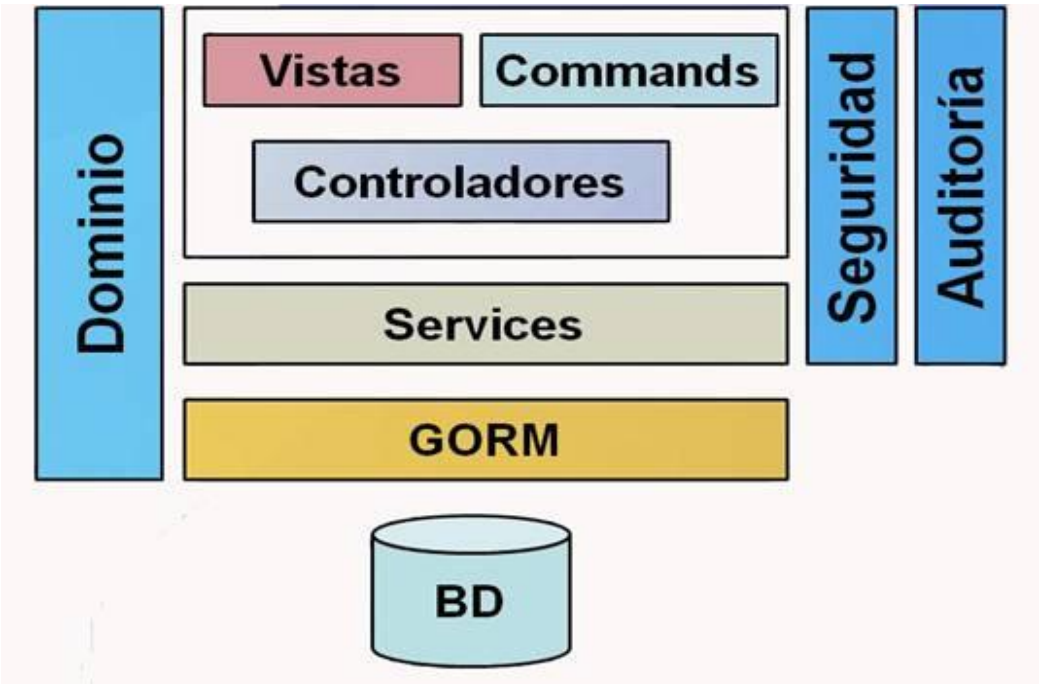

Figura 1: Modelo de la Arquitectura.

Dominio

Se encuentra representado verticalmente teniendo en cuenta que contiene todas las vistas, controladores, command y servicios, muestra el área del problema que se estudiará, como son los procesos que ocurren en la Dirección de Vigilancia y Seguridad Privada del VISIPOL.

Está compuesto por las clases que contienen de manera abstracta la información que el sistema va a manipular mediante las diferentes capas que lo componen, además de hacer cumplir las restricciones del negocio y permitir la transferencia de datos.

Seguridad

Se encarga de controlar los aspectos relacionados con la seguridad de la aplicación mediante el control de la autenticación de los usuarios además de gestionar las políticas de seguridad definidas. Este módulo se desarrolla utilizando el framework Spring Security, en el cual se filtran todas las llamadas a los controladores de la aplicación, verificando en cada una que el usuario esté autenticado y que posea el rol que le permite hacer uso de las funcionalidades del sistema sobre las cuales tiene permiso de acceso. Se representa en la arquitectura de manera vertical debido a que la seguridad es gestionada en todo el sistema.

Auditoría

Es la responsable de guardar todos los cambios que realiza el usuario en el sistema, es decir, guardaría el rol del usuario, el cambio que efectuó, la tabla donde realizó dicho cambio y la hora y fecha del mismo.

Vistas (Views)

Son las que más interactúan con el usuario, además de ser las encargadas de mostrar toda la información requerida por el usuario, no implementan lógica del negocio, solo lo referente a presentación como es el código para controlar las interacciones web, además de los estilos de las páginas, son una aplicación del Modelo Vista Controlador (MVC) y se encargan además de mostrar el estado del sistema y las acciones que el usuario tiene a disposición.

Controladores (Controllers)

Se encargan recibir las solicitudes que provengan de la interacción de los usuarios con las vistas y deciden que vista mostrar, aplican la lógica del negocio y son responsables de las decisiones de presentación como el idioma y los niveles de acceso. Permite la conexión lógica entre las acciones de los usuarios y los servicios del negocio, en Grails se generan controladores para gestionar el ciclo de vida de nuestras entidades, pero lógicamente podemos generar controladores mediante el comando create-controller.

Command

Son clases responsables de validar la información de un formulario cuando no existe una clase del dominio que lo pueda tratar, permite el acceso a atributos de diferentes clases, también posibilita insertar en varias tablas en una misma acción.

Servicios (Services)

Un servicio es una clase cuyo nombre sigue el patrón XXXService.groovy y que se aloja en la carpeta grails-app/services. En los servicios es donde se implementa toda la lógica del negocio del sistema, para luego ser aplicada por un determinado controlador, son los encargados de manejar el acceso a datos, además de lograr que la lógica de negocio sea reutilizable permitiendo invocar el mismo proceso desde distintos controladores o acciones sin repetir código.

GORM

Grails utiliza GORM (Grails Object Relational Mapping), es un gestor de persistencia escrito en Groovy, para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos dinámicos que facilitan enormemente las búsquedas. Mapea las

clases de dominio con tablas de la base de datos. Permite la definición de validaciones automáticas sobre estas clases, de manera que antes de persistir en la base de datos un objeto de este tipo, se ejecutan esas validaciones que evitan que se salve si no cumple las restricciones definidas, devolviendo la colección de errores generada. Se generan automáticamente los métodos necesarios para gestionar la persistencia del objeto y asimismo por defecto aplica la técnica del bloqueo optimista para gestionar esta problemática. Cada operación que se realice sobre los objetos del modelo de datos será traducida por Hibernate a las sentencias SQL necesarias para quedar reflejado en la base de datos. (16)

BD (Persistencia)

Esta capa corresponde a los almacenes de datos, contiene todas las estructuras necesarias para poder almacenar la información del dominio que debe persistir y está contenida físicamente en el servidor de bases de datos. Es única y común a todos los módulos de SIGEPOL y debe garantizar que el modelo de datos esté al menos, en tercera forma normal.

Para un mejor entendimiento de la arquitectura propuesta en el siguiente epígrafe se describe el flujo del caso de uso Registrar Denuncia.

3.4 Diagrama General de Clases del Diseño

El diagrama general de clases del diseño representa la interacción entre las clases de las diferentes capas por las que está integrado el Módulo de Servicios de Vigilancia y Seguridad Privada. Se ha utilizado como elemento característico el color definido en la Figura 2 para identificar las clases pertenecientes a cada capa.

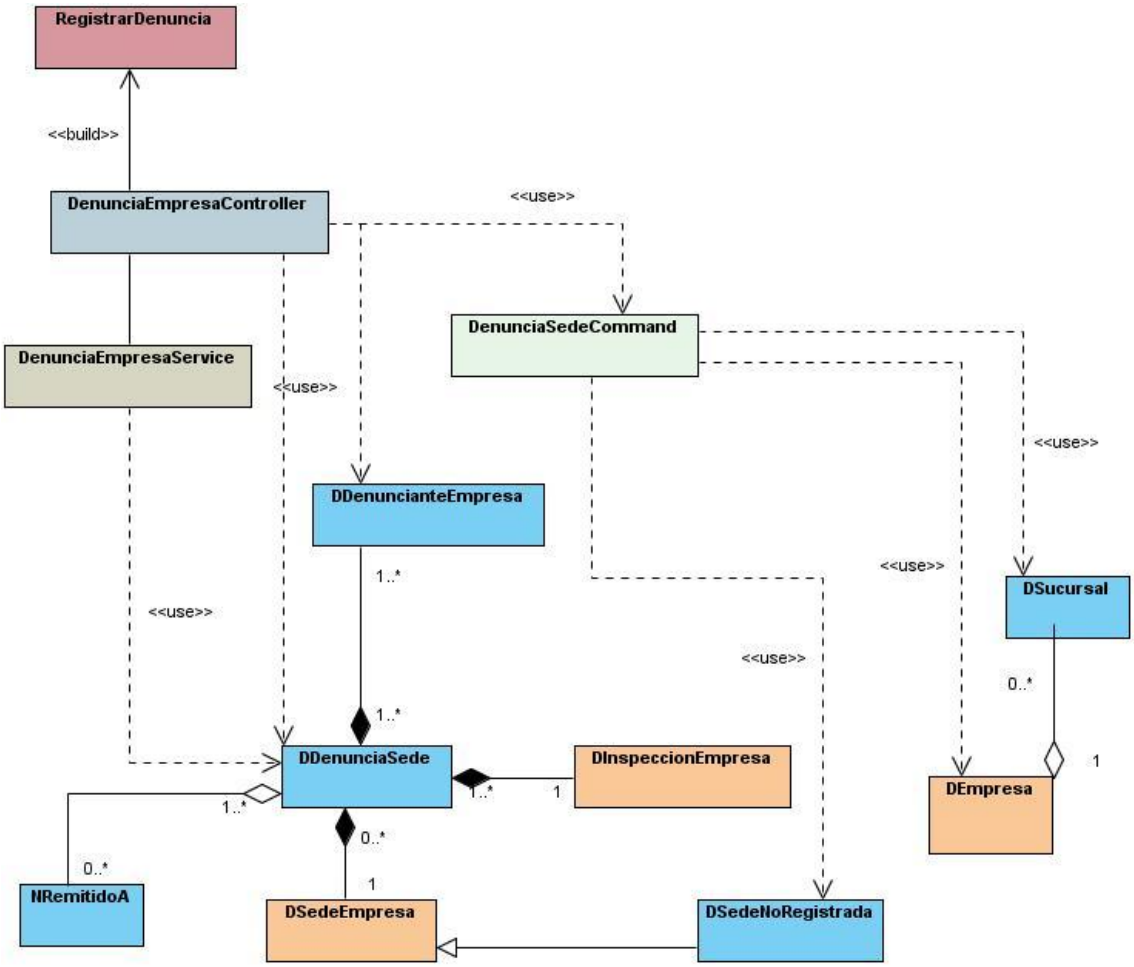

Figura 2: Diagrama General de Clases.

En la capa de presentación está implementado el patrón Modelo-Vista-Controlador en la cual van a estar ubicados los controladores, las vistas y los command los cuales son responsables de mostrar al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles.

El flujo comienza con la interacción del usuario con la vista RegistrarDenuncia, a continuación los datos son enviados al controlador DenunciaEmpresaController encargado de atender todas las solicitudes referentes a la gestión de las denuncias, luego mediante el command DenunciaSedeCommand se validan los datos enviados por el usuario, una vez que estén los datos en el controlador el mismo se encarga de definir las acciones necesarias para registrar la denuncia.

Luego mediante una instancia al servicio DenunciaEmpresaService el cual se encuentra en la capa de negocio se implementa la lógica del negocio referente a las denuncias que puedan ser realizadas a las empresas, a continuación el servicio mediante la acción Registrar en la cual se realiza una consulta a la BD y si esta tiene

un resultado satisfactorio se procede a registrar los datos de dicha denuncia mediante el uso de GORM.

Una vez concluido el registro de los datos el controlador basándose en la información que posee de la ejecución del servicio define qué vista se mostrará a continuación, y así termina el flujo del Caso de Uso Registrar Denuncia.

La seguridad es garantizada a la hora de enviar los datos hacia el controlador mediante una llamada a Spring Security Core.

3.5 Diagrama de Paquetes

Los diagramas de paquetes se usan para reflejar la organización de paquetes y sus elementos, muestran como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes. Es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas y agrupar elementos relacionados de dicho modelo con propósitos organizacionales. (17)

Para el diseño de este módulo las clases se agruparon por paquetes según su funcionalidad y las entidades que gestionan. Los paquetes definidos son: paquete Control de Trámites, paquete Control de Actas de Asamblea y Denuncias a Empresas, paquete Control de Constitución y Permiso de Funcionamiento de Empresas, paquete Control de Inspección y el paquete Común. Cada paquete contiene dos diagramas de clases correspondientes a la Capa de Presentación y a la Capa de Dominio. Esta división se hace necesaria para comprender con más precisión cada uno de los elementos que contiene, debido a la complejidad de los diagramas y el gran número de clases contenidos en los mismos.

A continuación se describe el objetivo de cada paquete para lograr una mayor comprensión:

El Paquete Control de Trámites: agrupa todas las funcionalidades referentes a la gestión de trámites que pueden ser realizados por las empresas que brindan Servicios de Vigilancia y Seguridad Privada.

El sistema incorporará las funcionalidades necesarias para permitir al representante de una empresa consultar el estado de sus trámites en la DIGESERVISP de manera sencilla y rápida.

En el Paquete Control de Actas de Asamblea y Denuncias a Empresas: se encuentran las funcionalidades necesarias para la gestión de las denuncias realizadas a las empresas que brindan Servicios de Vigilancia y Seguridad Privada, además de contener las funcionalidades necesarias para la gestión de dichas empresas.

El sistema una vez implementado incorporará la infraestructura necesaria para almacenar todos los cambios realizados a una empresa mediante las Actas de Asamblea.

En el Paquete Control de Constitución y Permiso de Funcionamiento de Empresas: se encuentran todas las funcionalidades necesarias para la creación de nuevas empresas y para gestionar los permisos de funcionamiento a estas empresas, estos permisos solo son efectivos por un año, por lo que en este paquete también estarán ubicadas las funcionalidades necesarias para renovar los permisos de funcionamiento.

El Paquete Control de Inspección: engloba todas las funcionalidades necesarias para gestionar las inspecciones realizadas a las empresas que brindan Servicios de Vigilancia y Seguridad Privada, estas inspecciones pueden ser producto de una denuncia o planificadas desde DIGESERVISP.

El Paquete Común: se representa de un color diferente porque está compuesto por diferentes funcionalidades que son comunes para todos los módulos que estructuran al SIGEPOL.

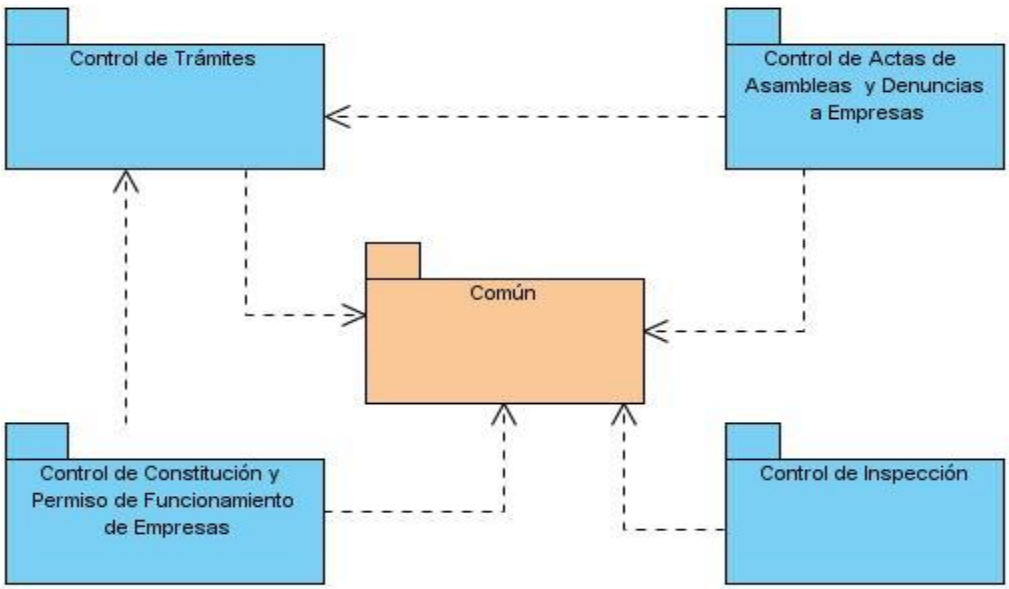

Figura 3: Diagrama de Paquetes del Diseño.

3.6 Diagramas de Clases

3.6.1 Diagramas de Clases de la Capa Presentación

A continuación se muestran los diagramas de clases pertenecientes al Módulo Servicios de Vigilancia y Seguridad Privada, primero se exponen los diagramas pertenecientes a la capa de presentación y luego los que pertenecen a la capa de dominio, una vez expuesto todos los diagramas se procede a la descripción de las principales clases presentes en los mismos.

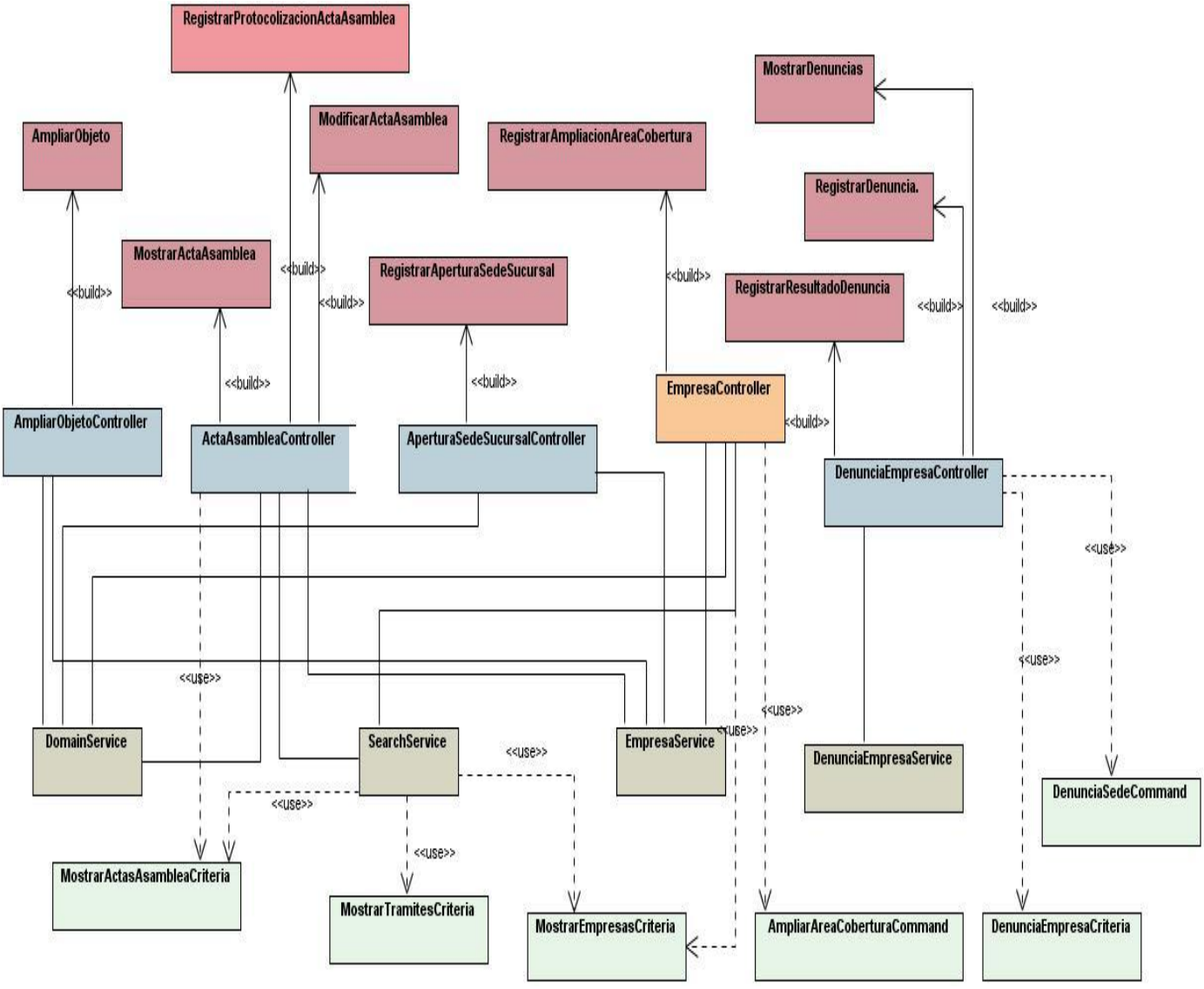

Figura 4: Diagrama de Clases Paquete Control de Actas de Asambleas y Denuncias.

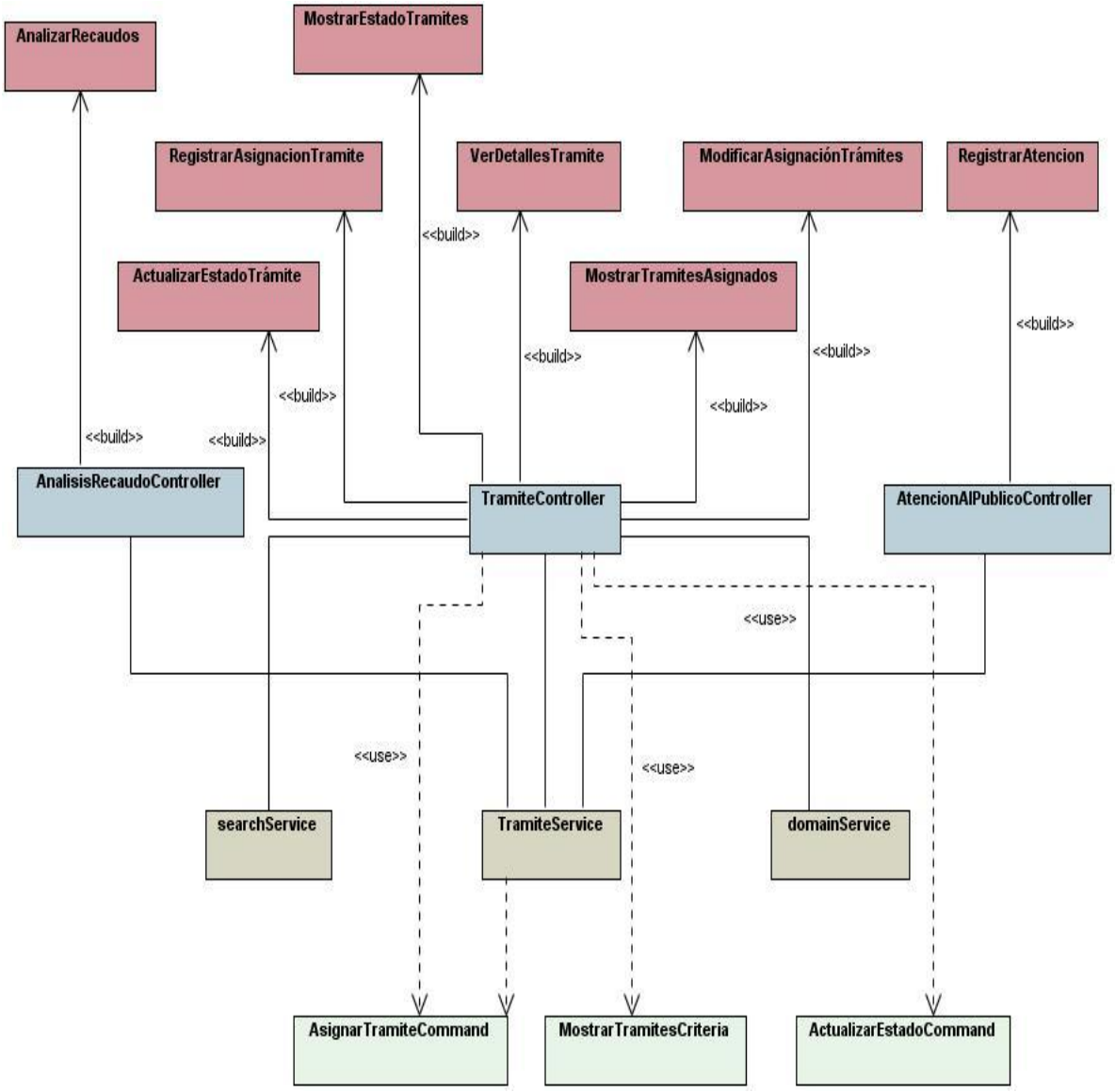

Figura 5: Diagrama de Clases Paquete Trámites.

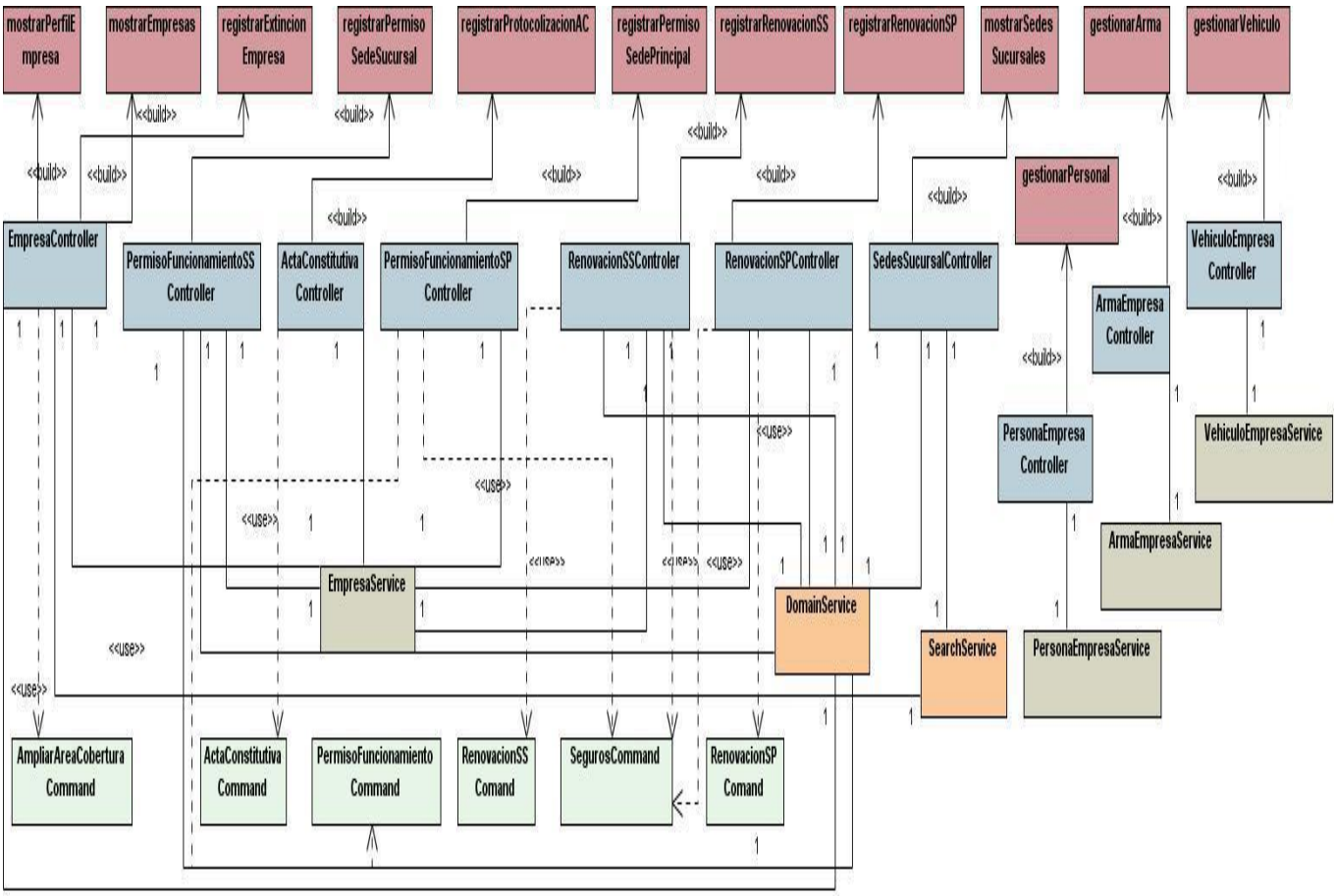

Figura 6: Diagrama de Clases Paquete Control de Constitución y Permiso de Funcionamiento de Empresas.

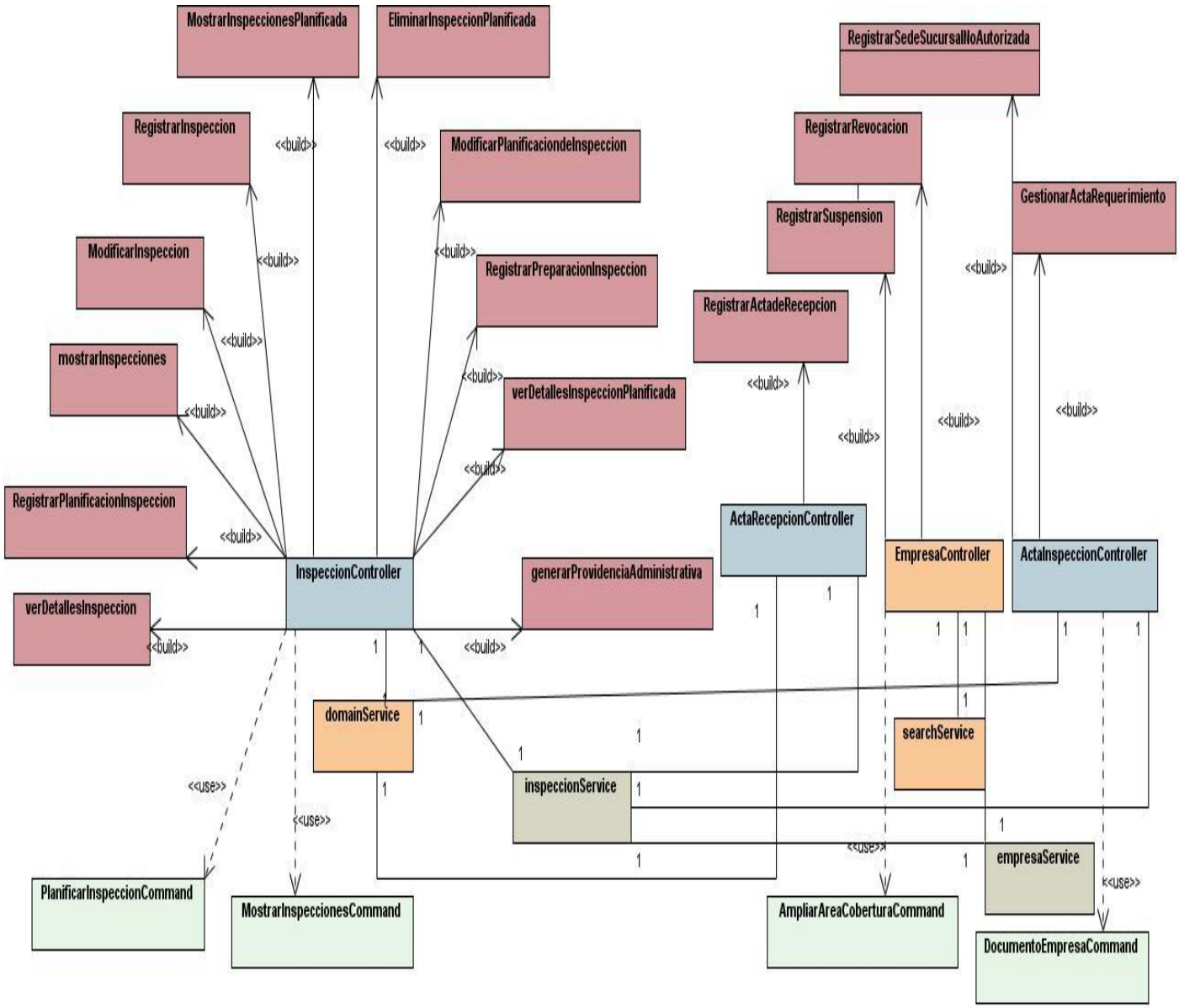

Figura 7: Diagrama de Clases Paquete Control de Inspección.

3.6.2 Diagrama de Clases de la Capa de Dominio

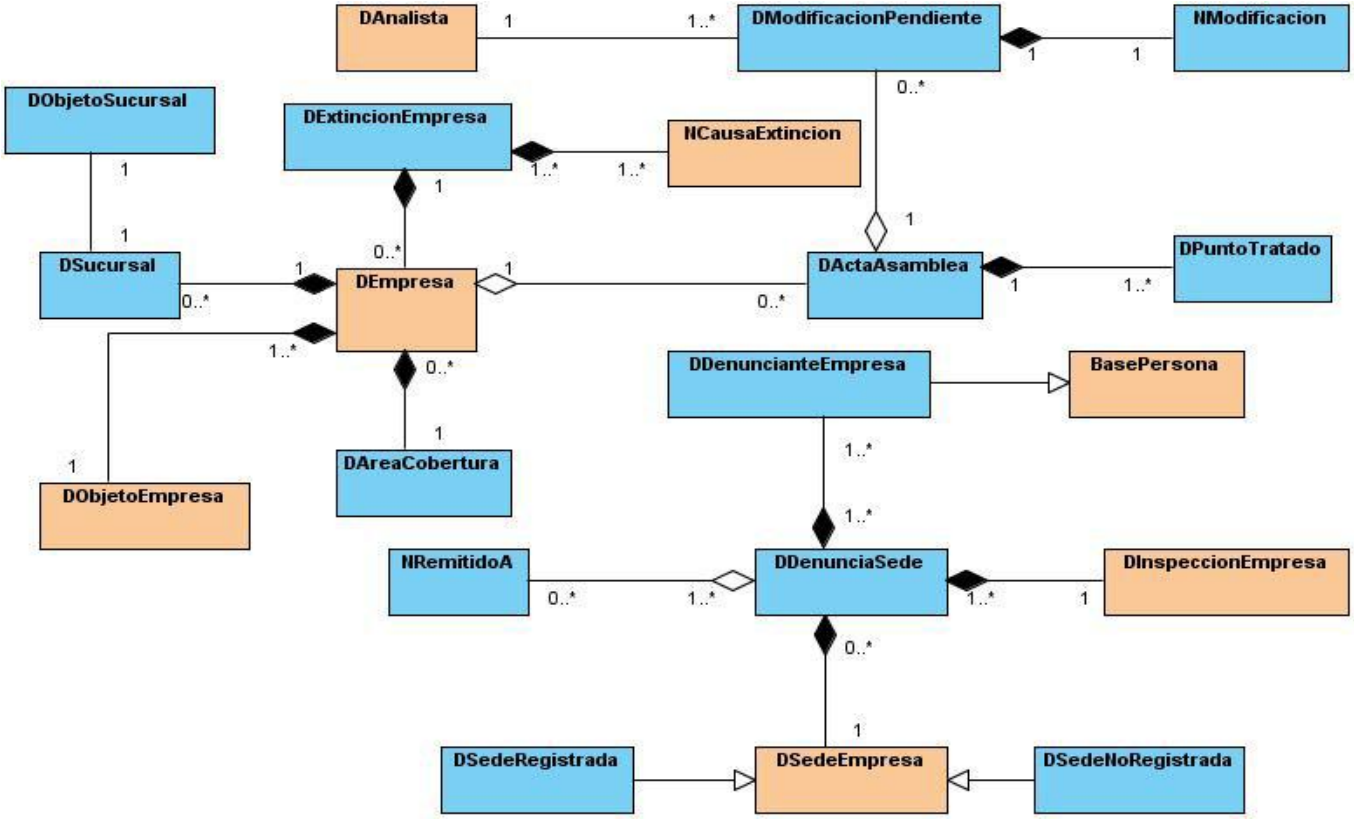

Figura 8: Diagrama de Clases Paquete Control de Actas de Asambleas y Denuncias a Empresas.

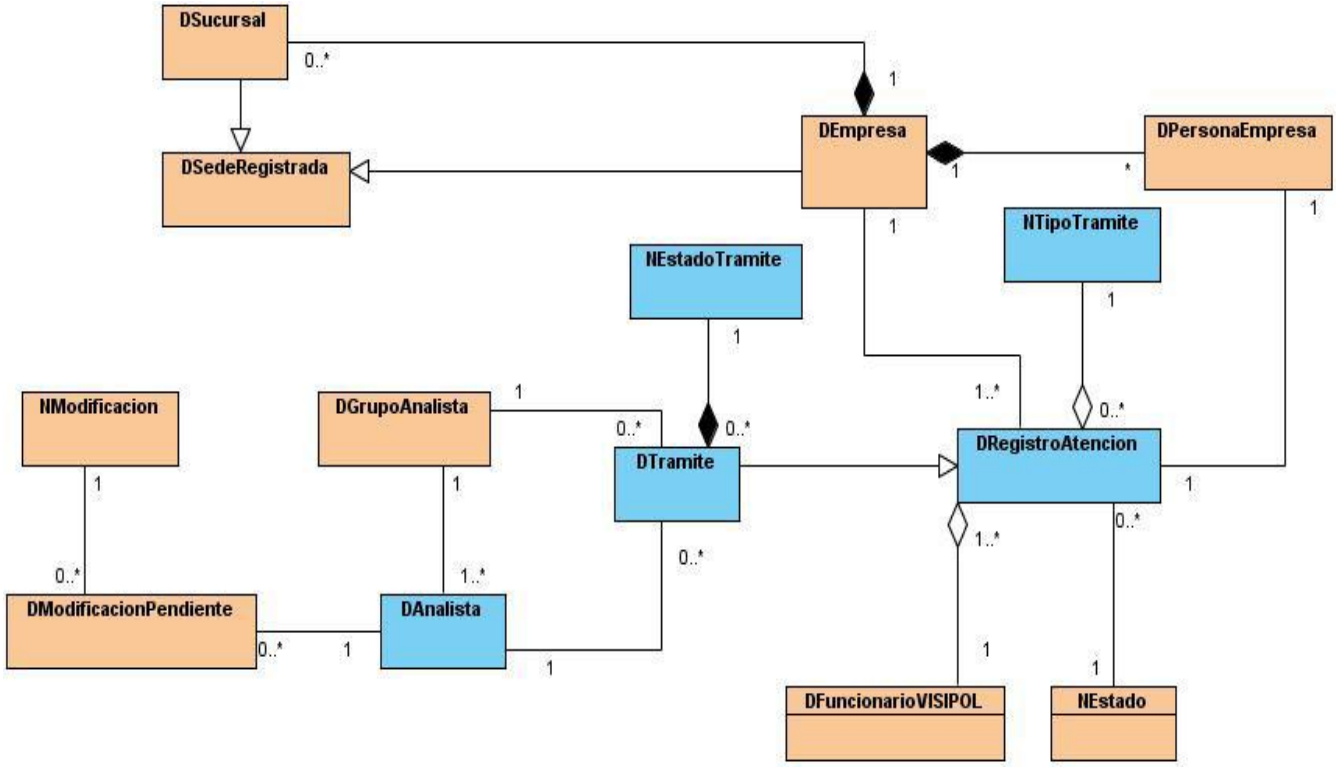

Figura 9: Diagrama de Clases Paquete Control Trámite.

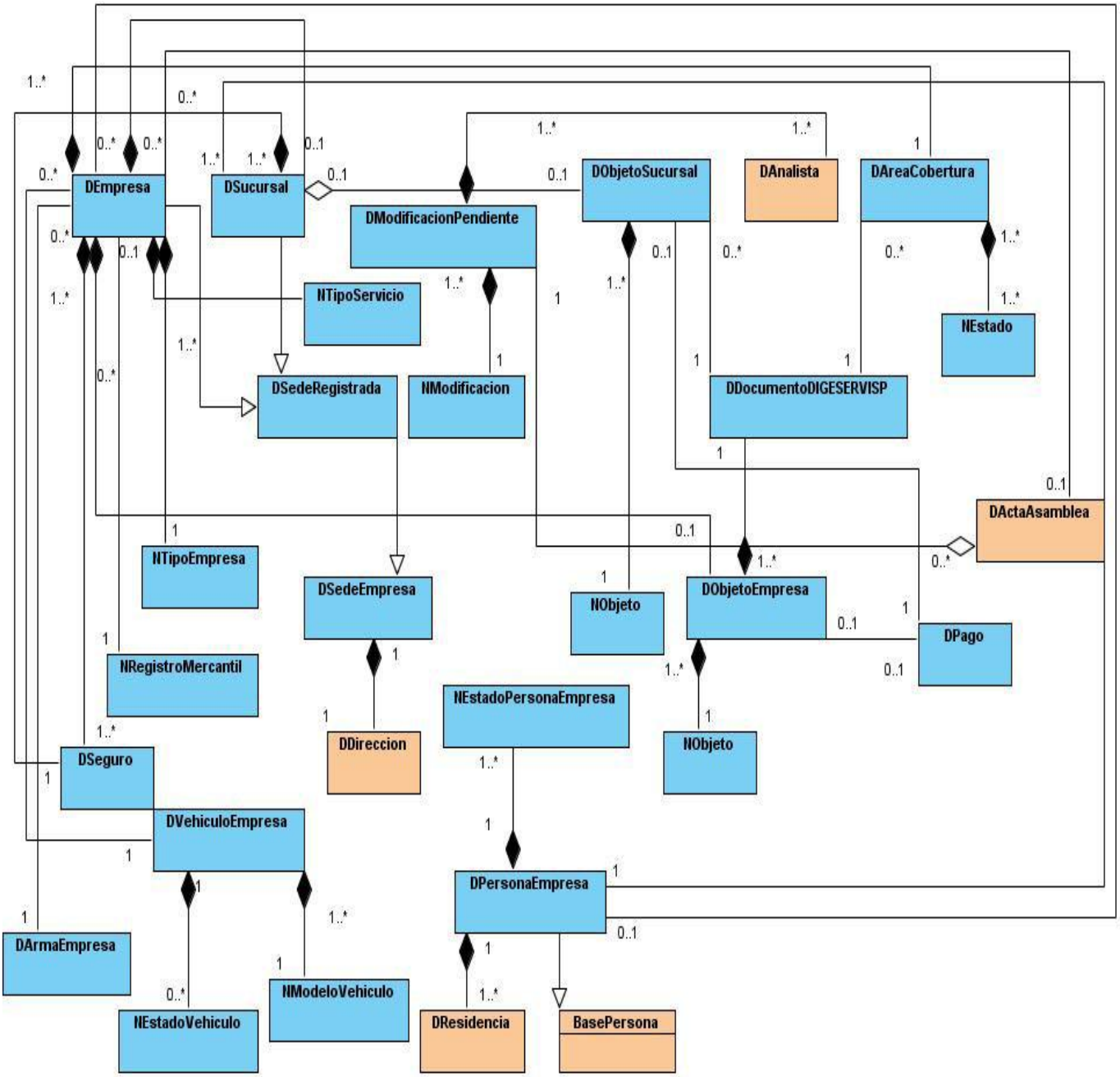

Figura 10: Diagrama de Clases Paquete Control de Constitución y Permiso de Funcionamiento de Empresas.

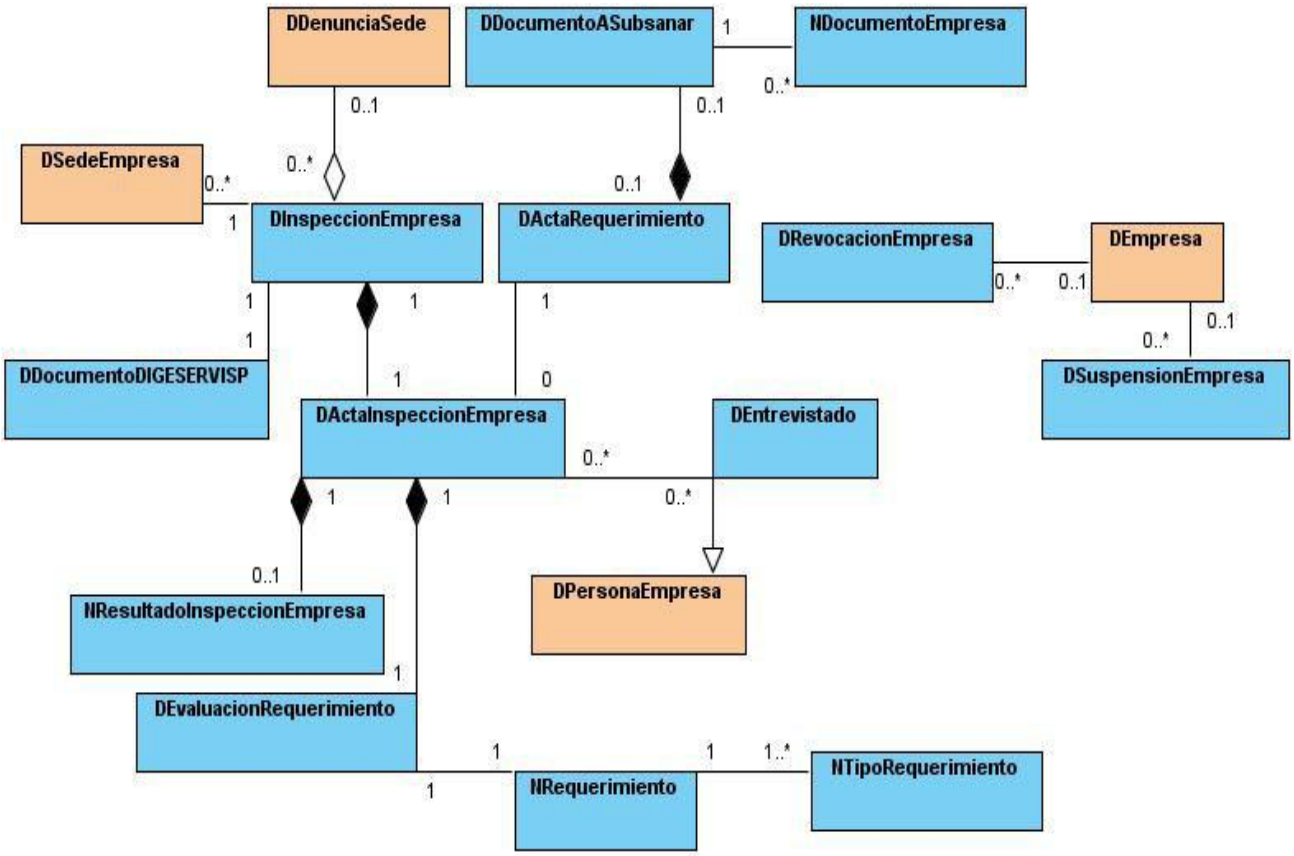

Figura 11: Diagrama de Clases Paquete Control de Inspección.

3.7 Descripción de las principales clases.

Las siguientes clases expuestas a continuación no son todas las que conforman al sistema pues como se puede apreciar en los diagramas pertenecientes a la Capa de Presentación y a la Capa de Dominio, existen un gran número de clases involucradas en el proceso, por lo que las siguientes clases son sólo las más importantes.

3.7.1 Clase mostrarEmpresas

Propósito

Clase de tipo Vista, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, en esta se muestra un formulario con los criterios de búsqueda. Se obtiene de la BD las empresas que cumplan con los criterios de búsqueda introducidos. Se muestran las empresas obtenidas.

Descripción

Figura 12: Mostrar Empresas.

Observaciones

- No aplica

Clase RegistrarPermisoFuncionamientoSedeSucursal

Propósito

Clase de tipo Vista, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, en esta se muestra un formulario con los datos que se deben insertar. El Analista introduce los datos especificados. Se almacenan los datos en la BD.

Descripción

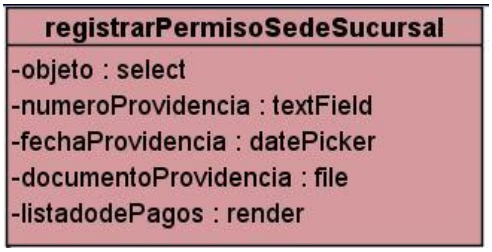

Figura 13: Registrar Permiso de Funcionamiento de Sede Sucursal.

Observaciones

- No aplica.

3.7.2 Clase Registrar Protocolización de Acta Constitutiva

Propósito

Clase de tipo Vista, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, en esta se muestra un formulario con los datos que se deben insertar asociados a la protocolización del acta constitutiva. El Analista introduce los datos especificados. Se almacenan los datos en la BD.

Descripción

Figura 14: Registrar Protocolización de Acta Constitutiva.

Observaciones

- No aplica.

3.7.3 Clase EmpresaController

Propósito

Clase de tipo Controlador, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de controlar las acciones a realizadas en la empresa y decidir las vistas que se deben mostrar referentes a las empresas.

Descripción

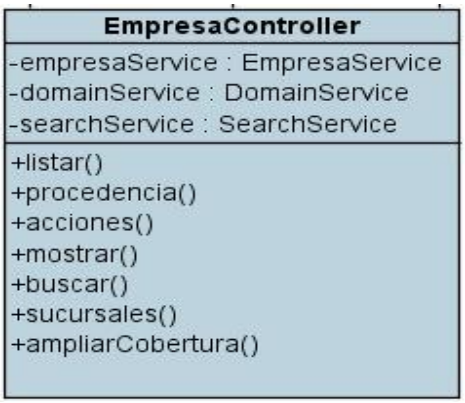

Figura 15: Empresa Controller.

Observaciones

- No aplica.

3.7.4 Clase PermisoFuncionamientoSSController

Propósito

Clase de tipo Controlador, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de registrar las solicitudes del usuario y decidir las vistas que se deben mostrar referentes a los permisos de funcionamientos a las sedes sucursales.

Descripción

Figura 16: Permiso de Funcionamiento Sede Sucursal Controller.

Observaciones

- No aplica.

3.7.5 Clase RenovacionSSController

Propósito

Clase de tipo Controlador, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de registrar las solicitudes del usuario y decidir las vistas que se deben mostrar referentes a las renovaciones de las sedes sucursales.

Descripción

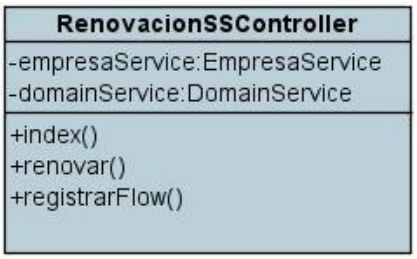

Figura 17: Renovación Sede Sucursal Controller.

Observaciones

- No aplica.

3.7.6 Clase SedesSucursalController

Propósito

Clase de tipo Controlador, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de recibir las solicitudes del usuario y decidir las vistas que se deben mostrar referentes a las sedes sucursales.

Descripción

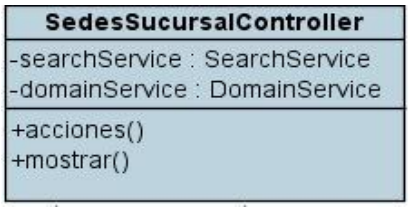

Figura 18: Sedes Sucursal Controller.

Observaciones

- No aplica

3.7.7 Clase EmpresaService

Propósito

Clase de tipo Servicio, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de realizar las acciones de lógica de negocio sobre el modelo referente a la gestión de las empresas.

Descripción

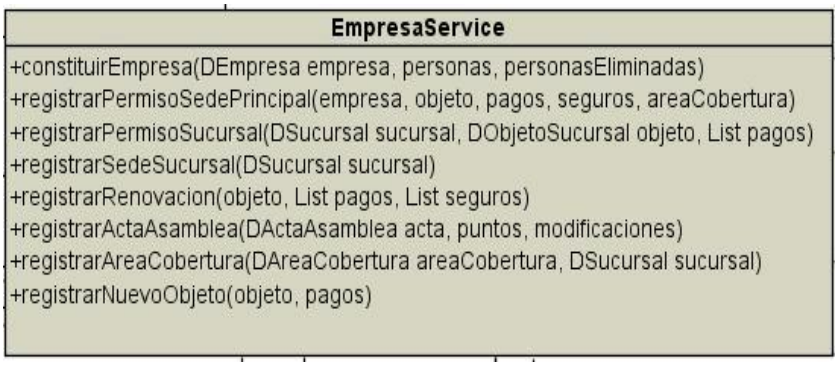

Figura 19: Empresa Service.

Observaciones

- No aplica.

3.7.8 Clase PersonaEmpresaService

Propósito

Clase de tipo Servicio, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de realizar las acciones de lógica de negocio sobre el modelo referente a la gestión del personal de las empresas.

Descripción

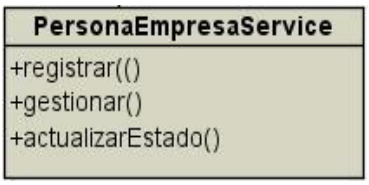

Figura 20: Persona Empresa Service.

Observaciones

- No aplica.

3.7.9 Clase ActaConstitutivaComand

Propósito

Clase de tipo Comando, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de validar la información que llega con un formulario cuando no existe ninguna entidad involucrada en el proceso.

Descripción

Figura 21: Acta Constitutiva Comand.

Observaciones

- No aplica

3.7.10 Clase RenovacionSSComand

Propósito

Clase de tipo Comando, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, encargada de validar la información que llega con un formulario cuando no existe ninguna entidad involucrada en el proceso.

Descripción

Figura 22: Renovación Sede Sucursal Comand.

Observaciones

- No aplica

3.7.11 Clase DEmpresa

Propósito

Clase de tipo Dominio, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas que contiene los datos las empresas.

Descripción

Figura 23: DEmpresa.

Observaciones

- Hereda de la clase DSedeRegistrada.

3.7.12 Clase DPago

Propósito

Clase de tipo Dominio, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas que contiene los datos los pagos.

Descripción

Figura 24: DPago.

Observaciones

- No aplica.

3.7.13 Clase Registrar Extinción de Empresa

Propósito

Clase de tipo Vista, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, en esta se muestra un formulario con los criterios de búsqueda. Se obtiene de la BD las empresas que cumplan con los criterios de búsqueda especificados. Se muestran las empresas obtenidas y un formulario con los datos que se deben insertar. El Inspector selecciona una empresa e introduce los datos especificados. Se almacenan los datos en la BD.

Descripción

Figura 25: Registrar Extinción Empresa.

Observaciones

- No aplica

3.7.14 Clase Registrar Protocolización de Acta Constitutiva

Propósito

Clase de tipo Vista, del paquete Control de Constitución y Permiso de Funcionamiento de Empresas, en esta se muestra un formulario con los datos que se deben insertar asociados a la protocolización del acta constitutiva. El Analista introduce los datos especificados. Se almacenan los datos en la BD.

Descripción

Figura 26: Registrar Protocolización Acta Constitutiva.

Observaciones

- No aplica

3.7.15 Clase ActaInspeccionController.

Propósito

Clase de tipo Controlador, del paquete Control de Inspección, encargada de recibir las solicitudes del usuario y decidir las vistas que se deben mostrar referentes a las actas de inspecciones.

Descripción

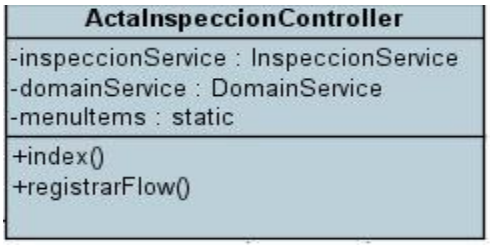

Figura 27: Acta Inspección Controller.

Observaciones

- No aplica

3.7.16 Clase InspeccionController.

Propósito

Clase de tipo Controlador, del paquete Control de Inspección, encargada de recibir las solicitudes del usuario y decidir las vistas que se deben mostrar referentes a las inspecciones.

Descripción

Figura 28: Inspección Controller.

Observaciones

- No aplica

3.7.17 Clase InspeccionService.

Propósito

Clase de tipo Servicio, del paquete Control de Inspección, encargada de realizar las acciones de lógica de negocio sobre el modelo referente a la gestión de inspecciones.

.**Descripción**

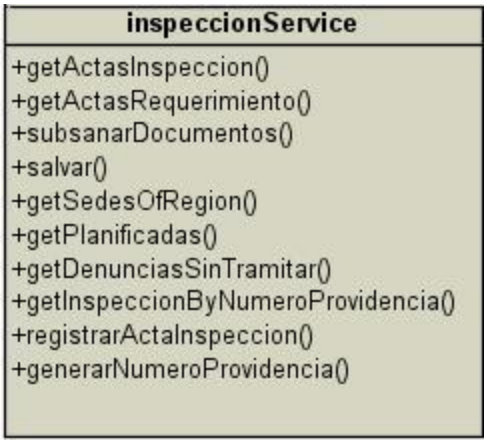

Figura 29: Inspección Service.

Observaciones

- No aplica

3.7.18 Clase AmpliarAreaCoberturaCommand.

Propósito

Clase de tipo Comando, del paquete Control de Inspección, encargada de validar la información que llega con un formulario cuando no existe ninguna entidad involucrada en el proceso.

Descripción

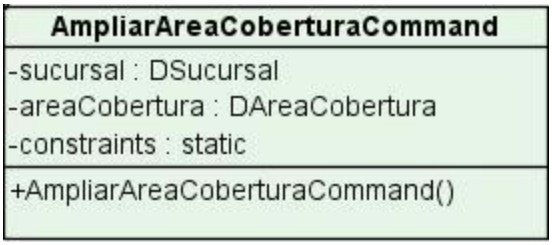

Figura 30: Ampliar Área Cobertura Comand.

Observaciones

- No aplica

3.8 Conclusiones

En este capítulo se han abordado los contenidos referentes a los patrones arquitectónicos utilizados en la elaboración de la solución, haciendo hincapié principalmente en los patrones GOF y a los patrones Arquitectónicos de Aplicaciones Empresariales que están presentes en varias áreas de la aplicación, en el capítulo se describe además la arquitectura del sistema explicando sus principales características, su función como parte de la arquitectura del sistema y el área donde se encuentran.

Capítulo 4: Implementación de la Solución

4.1 Introducción

En el presente capítulo se representan en términos de componentes los elementos del modelo del diseño definidos en el capítulo anterior. A continuación se modela un diagrama de componentes, con los componentes principales que constituyen la aplicación del Módulo Servicios de Vigilancia y Seguridad Privada. Se realiza una breve descripción de cada componente, especificando su propósito y contenido.

Los Diagramas de Componentes ilustran las piezas del software que no son más que los controladores, servicios, vistas y elementos del domino de nuestro sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases u objetos en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (18)

Figura 31: Diagrama de Componentes.

Descripción de los Componentes

4.2 Subsistema Grails-app

Propósito

Contenedor físico de los ficheros que conforman el núcleo de la aplicación.

Contenido

- Subsistema Controller
- Subsistema Domain
- Subsistema Services
- Subsistema Views

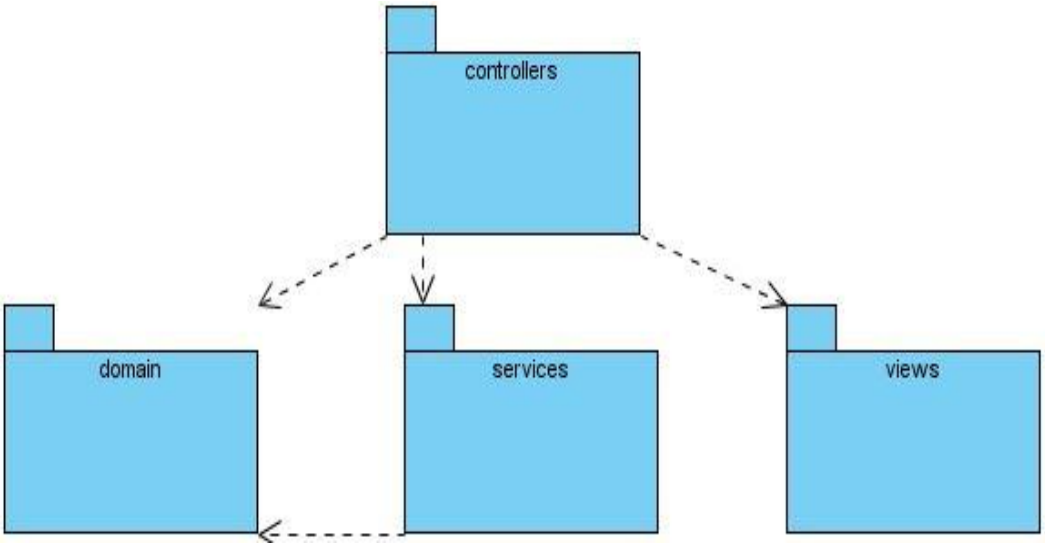

Figura 32: Diagrama de Componentes del Subsistema Grails-app.

Subsistema Controllers

Propósito

Contenedor físico de los ficheros que aplican la lógica del negocio y controlan el flujo en la aplicación, archivos con extensión Groovy.

Contenido

- VehiculoEmpresaController.groovy
- TramiteController.groovy
- SedeSucursalController.groovy
- RenovacionSSController.groovy
- RenovacionSPController.groovy
- PersonaEmpresaController.groovy
- PermisoFuncionamientoSSController.groovy
- PermisoFuncionamientoSPController.groovy
- PerfilEmpresaController.groovy
- InspeccionController.groovy
- EmpresaRESTController.groovy
- EmpresaController.groovy
- DenunciaEmpresaController.groovy
- AtencionAlPublicoController.groovy
- ArmaEmpresaController.groovy
- AnalistaController.groovy
- AnalisisRecaudoController.groovy
- AmpliarObjetoController.groovy

- ActaRecepcionController.groovy
- ActaInspeccionController.groovy
- ActaConstitutivaController.groovy
- ActaAsambleaController.groovy
- SegurosCommand.groovy
- ResultadoDenunciaSedeCommand.groovy
- RenovacionSSCommand.groovy
- RenovacionSPCommand.groovy
- RegistroAtencionCommand.groovy
- PuntoTratadoBinder.groovy
- PlanificarInspeccionCommand.groovy
- PermisoFuncionamientoSSCommand.groovy
- PermisoFuncionamientoCommand.groovy
- PagoBinder.groovy
- ModificacionBinder.groovy
- GestionSupport.groovy
- DocumentoEmpresaCommand.groovy
- DocumentoASubsanarBinder.groovy
- DenunciaSedeCommand.groovy
- AreaCoberturaBinder.groovy
- AmpliarAreaCoberturaCommand.groovy
- ActaConstitutivaCommand.groovy
- ActaAsambleaCommand.groovy

Subsistema Domain

Propósito

Contenedor físico de los ficheros que contienen de manera abstracta la información que el sistema va a manipular, archivos con extensión Groovy.

Contenido

- NRemitidoA.groovy
- DDenunciaSede.groovy
- DDenuncianteEmpresa.groovy
- UNumeroProvidencia.groovy
- NTipoRequerimiento.groovy
- NResultadoInspeccionEmpresa.groovy
- NRequerimiento.groovy

- NMotivoInspeccion.groovy
- NDocumentoEmpresa.groovy
- DInspeccionEmpresa.groovy
- DEvaluacionRequerimiento.groovy
- DEntrevistado.groovy
- DDocumentoASubsanar.groovy
- DActaRequerimiento.groovy
- DActaInspeccionEmpresa.groovy
- DUnidadTributaria.groovy
- DPago.groovy
- DPago.groovy
- HEstadoPersonaEmpresa.groovy
- DPersonaEmpresa.groovy
- DAccionistaPersonaJuridica.groovy
- NTipoVehiculo.groovy
- NModeloVehiculo.groovy
- NMarcaVehiculo.groovy
- NEstadoVehiculo.groovy
- NEstadoArmaEmpresa.groovy
- HEstadoVehiculoEmpresa.groovy
- HEstadoArmaEmpresa.groovy
- DVehiculoEmpresa.groovy
- DArmaEmpresa.groovy
- NTipoSeguro.groovy
- DSeguro.groovy
- DPolizaAccidentePersonal.groovy
- UNumeroRegistroAtencion.groovy
- NTipoTramite.groovy
- NModificacion.groovy
- NEstadoTramite.groovy
- DTramite.groovy
- DRegistroAtencion.groovy
- DPuntoTratado.groovy
- DModificacionPendiente.groovy
- DActaAsamblea.groovy
- NTipoEstandar.groovy
- NRecomendacionEstandar.groovy

- DRecomendacionEstandar.groovy
- DEstandar.groovy
- DDocumento.groovy
- NTipoServicio.groovy
- NTipoEmpresa.groovy
- NRegistroMercantil.groovy
- NObjeto.groovy
- NCausaExtincion.groovy
- DSuspensionEmpresa.groovy
- DSucursal.groovy
- DSedeRegistrada.groovy
- DSedeNoRegistrada.groovy
- DSedeEmpresa.groovy
- DRevocacionEmpresa.groovy
- DObjetoSucursal.groovy
- DObjetoEmpresa.groovy
- DExtincionEmpresa.groovy
- DEmpresa.groovy
- DDocumentoDIGESERVISP.groovy
- DAreaCobertura.groovy
- NRegionGeografica.groovy
- NParroquia.groovy
- NMunicipio.groovy
- NEstado.groovy
- DDireccion.groovy
- DFuncionarioPolicial.groovy
- BasePersona.groovy
- DGrupoAnalista.groovy
- DFuncionarioVISIPOL.groovy
- DAnalista.groovy

Subsistema Services

Propósito

Contenedor físico de los ficheros se encargan de manejar la lógica del negocio, archivos con extensión Groovy.

Contenido

- VehiculoEmpresaService.groovy
- TramiteService.groovy
- SearchService.groovy
- PersonaEmpresaService.groovy
- InspeccionService.groovy
- EmpresaService.groovy
- DenunciaEmpresaService.groovy
- ArmaEmpresaService.groovy
- AnalistaService.groovy
- MostrarTramitesCriteria.groovy
- MostrarEmpresasCriteria.groovy
- MostrarActasAsambleaCriteria.groovy
- InspeccionEmpresaCriteria.groovy
- DenunciaEmpresaCriteria.groovy

Subsistema Views

Propósito

Contenedor físico de las interfaces de usuarios, archivos con extensión GSP.

Contenido

- index.gsp

Acta de Asamblea

- protocolizar.gsp
- mostrar.gsp
- _acta.gsp

Acta Constitutiva.Protocolizar

- registrarDatosEmpresa.gsp
- registrarDatosEmpresa - copia.gsp
- gestionarPersonas.gsp

Acta de Inspección

- index.gsp

Registrar

- sucursalNoAutorizada.gsp
- resultadoInspeccion.gsp
- deberes.gsp
- datosGenerales.gsp
- actaRequerimiento.gsp

Acta de Recepción

- buscar.gsp
- _registrar.gsp

Ampliar Objeto

- sedeSucursal.gsp
- sedePrincipal.gsp
- _crear.gsp

Análisis de Recaudo

- index.gsp
- _recaudos.gsp
- _decision.gsp

Arma Empresa

- listado.gsp
- gestionar.gsp
- actualizarEstado.gsp
- _acciones.gsp

AtencionAlPublico

- registrar.gsp

Common

- seguros.gsp
- _sucursales.gsp
- _pagos.gsp
- _hiddens.gsp

DenunciaEmpresa

- resultado.gsp
- registrar.gsp
- mostrar.gsp

Empresa

- suspender.gsp
- sucursales.gsp
- revocar.gsp
- mostrar.gsp
- extinguir.gsp
- ampliarCobertura.gsp
- _empresas.gsp
- _acciones.gsp

Inspección

- providencia.gsp
- preparar.gsp
- planificar.gsp
- planificadas.gsp
- mostrar.gsp
- _sedes.gsp
- _denuncias.gsp
- _aGenerarProvidencia.gsp

PerfilEmpresa

- mostrar.gsp
- _vehiculos.gsp
- _sucursales.gsp
- _renovacion.gsp
- _personal.gsp
- _permisoFuncionamiento.gsp
- _inspecciones.gsp
- _denuncias.gsp
- _datosBasicos.gsp
- _armas.gsp
- _actasAsambleas.gsp

PersonaEmpresa

- listado.gsp
- gestionar.gsp
- actualizarEstado.gsp
- _acciones.gsp

SedePrincipal

- renovacion.gsp
- permisoFuncionamiento.gsp
- SedeSucursal
- renovacion.gsp
- permisoFuncionamiento.gsp
- mostrar.gsp
- apertura.gsp
- _acciones.gsp

Tramite

- mostrar.gsp
- asignar.gsp

- asignados.gsp
- actualizarEstado.gsp
- _tramites.gsp
- _atramites.gsp
- _analistas.gsp
- _actualizarEstadoLink.gsp

VehiculoEmpresa

- listado.gsp
- gestionar.gsp
- actualizarEstado.gsp
- _acciones.gsp

4.3 Subsistema Web-app

Propósito

Contenedor físico de los ficheros usados para el diseño de la página, archivos con extensión CSS, JS, HTML, JPG, XML.

Contenido

- Subsistema JS

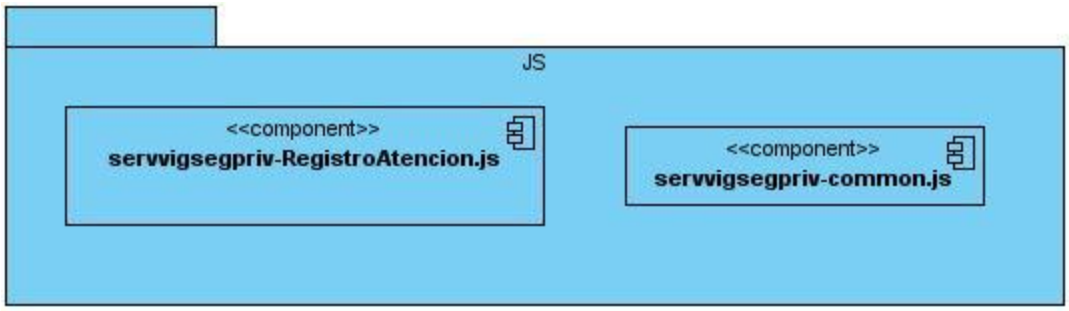

Figura 33: Diagrama de Componentes del Subsistema Web-app.

- resultado.gsp
- registrar.gsp
- mostrar.gsp

Subsistema JS

Propósito

Contenedor físico de los ficheros que permiten el acceso a los objetos de las interfaces, archivos con extensión JS.

Contenido

- Servvigsegpriv-common.js

- Servvigsegpriv-RegistroAtencion.js

4.4 Conclusiones

Los diferentes diagramas de componentes confeccionados en este capítulo, representan en términos de componentes y subsistemas de implementación, los elementos del modelo de diseño obtenidos en el capítulo anterior y establecen las dependencias entre uno y otro.

Conclusiones

El presente trabajo finaliza dando cumplimiento al objetivo general trazado de desarrollar una aplicación informática para los procesos de gestión de las empresas que brindan Servicios de Vigilancia y Seguridad Privada perteneciente a la DIGESERVISP como parte del Sistema de Gestión Policial para la República Bolivariana de Venezuela.

Como resultado del estudio realizado de la metodología y las herramientas usadas en el desarrollo se garantizo un uso eficiente de las mismas por parte del equipo de desarrollo, obteniendo así un producto con la calidad requerida.

Un aspecto de gran utilidad en la confección y concepción del diseño fue la comprensión y utilización de patrones, que permitieron aplicar buenas prácticas y brindar soluciones probadas a problemas comunes.

Se diseñaron los diagraman de componentes que representan la implementación del sistema en términos de componentes, quedando construida una aplicación capaz de garantizar rapidez y efectividad en los procesos de gestión de las empresas que brindan Servicios de Vigilancia y Seguridad Privada de la DIGESERVISP.

Recomendaciones

Realizar las pruebas a la versión operacional obtenida del Módulo de Servicios de Vigilancia y Seguridad Privada, que permitan validar y medir la calidad del producto desarrollado.

Profundizar más en el estudio del framework Grails y el lenguaje Groovy para aprovechar las facilidades que brindan y así reducir el tiempo de desarrollo notablemente.

Incorporar nuevas funcionalidades para garantizar un mayor control sobre las empresas que brindan Servicios de Vigilancia y Seguridad Privada de la República Bolivariana de Venezuela.

Referencias Bibliográficas

1. El miedo se apodera de la mayoría de los venezolanos. [Online] [Cited: Abril 25, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBUQFjAA&url=http%3A%2F%2Flaverdad.com%2Fdetnotic.php%3FCodNotic%3D930&rct=j&q=de%20los%2024%20estados%20de%20Venezuela%20el%20miedo%20se%20ha%20generalizado&ei=KqveTe2jBabY0QH5vvHDCg&usg=AFQjCNGiGIg4.
2. Seguridad Ciudadana. [Online] [Cited: Febrero 15, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=2&ved=0CBkQFjAB&url=http%3A%2F%2Fwww.ilsed.org%2Findex.php%3Foption%3Dcom_docman%26task%3Ddoc_view%26gid%3D180&rct=j&q=Se%20entiende%20por%20Seguridad%20Ciudadana%20a%20%E2%80%9Cla%20acci%C3%B3n%20integrada%.
3. Constitución Vigente. [Online] [Cited: Diciembre 16, 2010.] http://www.analitica.com/BITBLIO/anc/constitucion1999.asp
4. Gaceta Oficial. [Online] [Cited: Enero 21, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fwww.asambleanacional.gob.ve%2Findex.php%3Foption%3Dcom_docman%26task%3Ddoc_download%26gid%3D1732%26%26Itemid%3D190&rct=j&q=Ministra%20en%20la%20planificaci%C3%B3n%2C%20coordi.
5. UNIVERSIDAD PERUANA LOS ANDES. [Online] [Cited: Abril 10, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=3&ved=0CB8QFjAC&url=http%3A%2F%2Fwww.fiupla.edu.pe%2Fjlh%2Fsistemas%2Fmod%2Fresource%2Fview.php%3Finpopup%3Dtrue%26id%3D3455&rct=j&q=Van%20indicando%20paso%20a%20paso%20todas%20las%20actividades%20a%20realiz.
6. Lenguaje Unificado Uml . [Online] [Cited: Mayo 10, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=2&ved=0CB0QFjAB&url=http%3A%2F%2Fwww.buenastareas.com%2Fensayos%2FLenguaje-Unificado-Uml%2F490134.html&rct=j&q=El%20Lenguaje%20Unificado%20de%20Modelado%20(UML)%20es%20un%20lenguaje%20para%20visualizar%2C%20.
7. Visual Paradigm. [Online] [Cited: Diciembre 12, 2010.] http://www.visual-paradigm.com/product/vpuml.
8. Ingenieria Software. [Online] [Cited: Noviembre 22, 2010.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fingdesoftware-karina.blogspot.com%2F&rct=j&q=Estas%20herramientas%20pueden%20servir%20de%20apoyo%20en%20todos%20los%20aspectos%20del%20ciclo%20de%20vida%20de%20desarrollo%20d.
9. Taller de Rails. [Online] [Cited: Mayo 24, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=5&ved=0CC4QFjAE&url=http%3A%2F%2Fwww.scribd.com%2Fdoc%2F55151939%2FTaller-de-Rails&rct=j&q=Un%20entorno%20de%20desarrollo%20integrado%20(IDE)%20es%20un%20programa%20inform%C3%A1tico%20compuesto%20por%20un%20.

10. Decreto 3390 Software Libre. [Online] [Cited: Abril 25, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fwww.scribd.com%2Fdoc%2F9756%2FDecreto-3390-Software-Libre&rct=j&q=el%20decreto%20Ley%20No.%203.390%20de%20la%20Gaceta%20Oficial%20de%20la%20Rep%C3%BAblica%20Bolivariana%20de%.
11. Historia de Java. [Online] [Cited: Mayo 26, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fzarza.usal.es%2F~fgarcia%2Fdoc%2Ftuto2%2FI_2.htm&rct=j&q=Java%20fue%20dise%C3%B1ado%20en%201990%20por%20James%20Gosling%2C%20de%20Sun%20Microsystems%2C%20como%20software%20pa.
12. Groovy. [Online] [Cited: Octubre 10, 2010.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fes.wikipedia.org%2Fwiki%2FGroovy_(lenguaje_de_programaci%25C3%25B3n)&rct=j&q=Groovy%201.0%20apareci%C3%B3%20el%202%20de%20enero%20de%202007.%20Despu%C3%A9s%20de%20varias%20ve.
13. Manual de Java. [Online] [Cited: Enero 16, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fwww.docstoc.com%2Fdocs%2F278164%2FMANUAL-DE-JAVA&rct=j&q=El%20t%C3%A9rmino%20JVM%20se%20refiere%20a%20la%20especificaci%C3%B3n%20abstracta%20de%20una%20m%C3%A1quina%20de%20so.
14. [book auth.] Nacho Brito. *Manual de desarrollo web con Grails.* España : s.n., 2009.
15. SpringSource Tool Suite. [Online] [Cited: Febrero 22, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww.springsource.com%2Fdeveloper%2Fsts&rct=j&q=SpringSource%20Tools%20Suite%20&ei=qrPeTZepDemV0QH2vbGiCg&usg=AFQjCNHWKoR-osNupYeKb6u4N7_0RCFLfA&cad=rja.
16. Patrones de Diseño. [Online] [Cited: Febrero 22, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fmsdn.microsoft.com%2Fes-es%2Flibrary%2Fbb972240.aspx&rct=j&q=%E2%80%9CLos%20patrones%20de%20dise%C3%B1o%20son%20el%20esqueleto%20de%20las%20soluciones%20a%20problemas%20comun.
17. Diagramas de Paquete. [Online] [Cited: Septiembre 30, 2010.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fmaicolsrg.blogspot.com%2F2009%2F09%2Fdiagrama-de-paquetes.html&rct=j&q=Los%20diagramas%20de%20paquetes%20se%20usan%20para%20reflejar%20la%20organizaci%C3%B3n%20de%20paquetes%.
18. Diagrama de Componentes. [Online] [Cited: Mayo 25, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBUQFjAA&url=http%3A%2F%2Fwww.dsi.uclm.es%2Fasignaturas%2F42530%2Fpdf%2FM2tema12.pdf&rct=j&q=Los%20Diagramas%20de%20Componentes%2

0i&ei=UbXeTY3UKqjs0gH4mYCeCg&usg=AFQjCNH2aMYH8Locr7hpha9HPvWKTI4EfQ&cad.

Bibliografía

- **Almaer, Dion**. “The Dojo Toolkit in Practice”. 2006.
- **Craig Larman**. UML y Patrones, Introducción al análisis y diseño Orientado a Objetos.
- **Brito, Nacho**. “Manual de desarrollo web con Grails”.2009.
- **Visual Paradigm**. Visual Paradigm for UML Model –Code-Deploy Platform. [En línea][Citado el: 9 de Enero de 2008.] Disponible en: http://www.visual-paradigm.com/product/vpuml/.
- “**Fundamentos de Ingeniería de Software**”. Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.
- **Scott Davis and Jason Rudolph**. “Getting Started with Grails”. 2010
- **Visual Paradigm**. Visual Paradigm for UML Model –Code-Deploy Platform. [En línea][Citado el: 9 de Enero de 2008.] Disponible en: http://www.visual-paradigm.com/product/vpuml.
- **Jacobson, I. and Booch, G. y Rumbaugh, J**. “El Proceso Unificado de Desarrollo de software”. 2000. 2000.
- **Constitución de la República Bolivariana de Venezuela**. [En línea] [Citado el: 16 de Diciembre de 2007.] Disponible en: **http://www.venezuela-oas.org/Constitucion%20de%20Venezuela.htm**.

Anexos

Requisitos Funcionales del Módulo de Servicios de Vigilancia y Seguridad Privada.

- Registrar los datos de la atención que se le ha brindado a un ciudadano que solicita los servicios de la Dirección General de Servicios de Vigilancia y Seguridad Privada.
- Registrar los datos del análisis de recaudos realizado a un trámite sin verificar especificado.
- Mostrar datos referentes al estado de los trámites correspondientes a criterios de búsqueda especificados.
- Ver datos en detalle de un trámite previamente seleccionado.
- Registrar la asignación de un trámite a uno de los analistas pertenecientes al grupo que se encarga de revisar las solicitudes realizadas por las empresas ubicadas en los estados asignados al mismo.
- Modificar la asignación de un trámite a un analista específico, indicando los motivos para asignar un nuevo analista.
- Mostrar los datos de los trámites asignados a un analista especificado.
- Actualizar el estado de procesamiento un trámite previamente seleccionado.
- Registrar la protocolización del acta constitutiva de una nueva empresa.
- Mostrar datos de las empresas registradas de acuerdo a criterios de búsqueda especificados.
- Registrar el permiso de funcionamiento de la sede principal de una empresa previamente seleccionada.
- Registrar el permiso de funcionamiento de una sede sucursal de una sucursal de empresa previamente seleccionada.
- Registrar la renovación de la sede principal de una empresa previamente seleccionada.
- Registrar la renovación de la sede sucursal de una sucursal de empresa previamente seleccionada.
- Registrar los datos de las solvencias y pólizas de la sede de una empresa previamente seleccionada.
- Registrar la protocolización de un acta de asamblea de una empresa previamente seleccionada.
- Mostrar datos de las actas de asamblea registradas de acuerdo a criterios de búsqueda especificados.

- Registrar los datos referentes a la apertura de una nueva sucursal asociada a una empresa previamente seleccionada.
- Registrar la ampliación del área de cobertura de una empresa previamente seleccionada, entiéndase, que se añaden estados al listado antes definido, con el consecuente registro de los datos de las nuevas sucursales de los estados adicionado.
- Registrar la ampliación del objeto de la sede de una empresa seleccionada anteriormente.
- Mostrar datos de las sedes sucursales registradas de una empresa de acuerdo a criterios de búsqueda especificados.
- Gestionar los datos del personal asociado a una empresa previamente seleccionada, entiéndase por gestionar, las acciones de registro y modificación de los datos y su estado.
- Gestionar los datos de las armas asociadas a una empresa previamente seleccionada, entiéndase por gestionar, las acciones de registro y modificación de los datos y su estado.
- Gestionar los datos de los vehículos asociados a una empresa previamente seleccionada, entiéndase por gestionar, las acciones de registro y modificación de los datos y su estado.
- Registrar los datos de una denuncia realizada.
- Registrar los datos del resultado de la inspección realizada a una sede sobre la que existía una denuncia previa.
- Mostrar datos de las denuncias registradas de acuerdo a criterios de búsqueda especificados.
- Mostrar datos de las denuncias sin tramitar registradas de acuerdo a criterios de búsqueda especificados. Una denuncia sin tramitar es aquella que no tiene asociada una inspección.
- Ver datos en detalle de una denuncia previamente seleccionada.
- Modificar datos de una denuncia previamente seleccionada.
- Registrar los datos de la planificación de la inspección a una sede especifica.
- Mostrar datos de las inspecciones planificadas registradas de acuerdo a criterios de búsqueda especificados.
- Ver datos en detalle de una inspección planificada previamente seleccionada.
- Modificar datos de una inspección planificada previamente seleccionada.
- Eliminar una inspección planificada previamente seleccionada.

- Registrar los datos de la preparación de la inspección a realizar a una empresa, así como obtener información relevante de la misma. Generar la providencia administrativa.
- Generar la providencia administrativa a partir de la información registrada en la preparación de la inspección de la empresa.
- Registrar los datos de una inspección realizada a una sede especifica.
- Gestionar los datos de un acta de requerimiento, entiéndase por gestionar, las acciones de registro y modificación de los datos.
- Registrar los datos referentes a una sede sucursal no autorizada. Una sede sucursal no autorizada es una sede que ha sido constituida sin consentimiento de DIGESERVISP.
- Registrar los datos de suspensión de una empresa previamente seleccionada.
- Mostrar datos de las inspecciones realizadas registradas de acuerdo a criterios de búsqueda especificados.
- Ver datos en detalle de una inspección realizada previamente seleccionada.
- Modificar datos de una inspección realizada previamente seleccionada.
- Registrar los datos de revocación del permiso de funcionamiento de una empresa previamente seleccionada.
- Registrar los datos del acta de recepción de los documentos a subsanar señalados en el acta de requerimientos durante la inspección.
- Registrar los datos de la extinción de una empresa previamente seleccionada.
- Mostrar los datos del perfil de una empresa previamente seleccionada.
- Modificar los datos del acta constitutiva de una empresa previamente seleccionada.
- Modificar los datos del permiso de funcionamiento de sede principal de una empresa previamente seleccionada.
- Modificar los datos del permiso de funcionamiento de sede sucursal de una sucursal previamente seleccionada.
- Modificar los datos de la renovación de la sede principal de una empresa previamente seleccionada.
- Modificar los datos de la renovación de una sede sucursal de una empresa previamente seleccionada.
- Modificar los datos de las solvencias y pólizas de una empresa previamente seleccionada.
- Modificar los datos de un acta de asamblea previamente seleccionada.
- Modificar los datos de una sede sucursal previamente seleccionada.