FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

DISEÑO E IMPLEMENTACIÓN DEL MÓDULO SUSTANCIAS QUÍMICAS DEL SIIPOL

Autor(es): Yunislema González Bermúdez

Adrián Ramírez Almenarez

Tutor(es): Ing. Maylin Díaz Cabrera

La Habana 2012

Año 54 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de_____ del año 2011.

_____ _____

AUTOR **AUTOR**

Yunislema González Bermúdez Adrian Ramirez Almenarez

_____

TUTOR

Ing. Maylin Díaz Cabrera

AGRADECIMIENTOS

De Yunislema:

Primero que todo agradecer a mi mamá y a mi papa, que son las personas que más quiero, a mis abuelas y mi abuelo, a toda mi familia, que de una forma u otra siempre me han apoyado. A mi novio. A todos los presentes. Gracias a mi tutora Maylin y a su esposo Kenny que aun estando lejos nos brindaron toda la ayuda posible, así como también a mi compañero de tesis Adrian y a Gual y los demás profesores de CICPC que de una forma u otra contribuyeron a que saliera este trabajo. A todos los profesores que tuve durante la carrera. A las picuas del 44, Luisa, Yary e Ivett y a las que no son picuas también, Yanet y Yissel. También agradecer a una personita que es muy callada pero muy buena persona Isel o la polilla como le decimos cariñosamente. Creo que ahora viene la parte más difícil, el momento de agradecer a las amistades que han estado conmigo durante estos 5 años. A Surenis que con sus gritos vuelve loco a cualquiera, pero es muy buena persona y amiga. A Yusdania que siempre me ha apoyado en todo momento. Y por ultimo pero muy importante a Orquídia que para mí ha sido como una hermana estos 5 años.

De Adrian:

A mi mamá, abnegada, entregada, cariñosa e incansable, te quiero con la vida y te digo: "ya somos ingenieros". A mi papá, eres mi ejemplo a seguir, el hombre que quiero ser, mi amigo incondicional. A mi hermano, siempre presente, mis triunfos son tuyos. A mi novia, amor no resume cuanto te quiero, gracias por escuchar cada noche en La Niemeyer mi exposición y por ser como eres. A mi familia, que me abraza fuerte y me quiere incondicionalmente. A mis colegas, Javier, Kenny, Nadian, Rodolfo y Ronny (ordenados en orden alfabético, para que no hallan celos), mis amigos ya no de la Lenin, ni del verde, ni de la UCI; mis hermanos de la vida, mis colegas pa' siempre (que cursi jajaja). A Gual, mi otro compañero de Tesis, mi otro hermano de la vida, colega pa' siempre y del estudio, ahora somos los que más sabemos Spring, Hibernate y JSF. A Maylin y Kenny, mejores tutores no pude tener; gracias, ustedes hicieron que la distancia no fuera problema. A Yuni, más que mi compañera de tesis, mi compañera en armas. A Oigres, gracias por hacerme leña el documento y destrozarnos en la predefensa, me hiciste más fuerte. A la UCI, que fue mi hogar estos 5 años y a todos los que de alguna manera me ayudaron a ser ingeniero y más importante a ser Adrian.

DEDICATORIA

A mis padres que me han brindado su apoyo incondicional y han hecho posible que se cumpla mi sueño.

Yunislema

A mis padres, que me ha hecho el hombre que soy, más allá del ingeniero que intento ser.

Adrian

RESUMEN

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) es una de las instituciones más importantes que lucha contra el crimen existente en la República Bolivariana de Venezuela. Parte de esta institución es la División de Investigación y Fiscalización de Sustancias Químicas, perteneciente a la Dirección Contra Drogas. Dicha división tiene la misión de registrar información, investigar y fiscalizar a las empresas que operan en territorio venezolano utilizando de alguna manera las sustancias químicas controladas por el Régimen Legal No. 4 (que son aquellas que por sus propiedades químicas pueden ser utilizadas en la síntesis y elaboración de drogas).

En la actualidad el CICPC emplea la versión 4 del Sistema de Investigación e Información Policial (SIIPOL), sistema concebido como parte del proyecto de modernización del CICPC. La versión 4 del SIIPOL no brinda soporte a los procesos realizados en la División de Investigación y Fiscalización de Sustancias Químicas.

Antiguamente la División de Investigación y Fiscalización de Sustancias Químicas empleaba para gestionar la información que manejaba, al Sistema de Investigación Control y Registro de Operadores Químicos (SICROQ) parte del Sistema Integrado de Información Policial (antiguo sistema empleado por el CICPC); el cual no daba soporte a todos los procesos que realizaba. Para darle solución a este problema y modernizar esta área, surge el presente trabajo, el cual se enmarca en el SIIPOL, nuevo sistema de gestión de información policial.

El objetivo del presente trabajo es diseñar e implementar un módulo Sustancias Químicas que se integre plenamente al SIIPOL y responda a todas las necesidades de la División de Investigación y Fiscalización de Sustancias Químicas. Para dar cumplimiento a tal objetivo, en este trabajo se describen Sistemas de Gestión de Información Policial, se realiza un estudio del entorno de desarrollo que se emplea, se recoge toda la documentación referente al proceso diseño e implementación del módulo y se valida dicha solución mediante la verificación del producto obtenido.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Venezuela ha sido declarada en varias ocasiones uno de los países más violentos del continente Americano, según el Índice de Paz Global (GPI[1]) producido por el Instituto para la Economía y la Paz[2]. (1) La tensa situación en las calles, junto a la corrupción en el aparato policial contribuyen a la inseguridad ciudadana y al estado de paranoia generalizada.

Para combatir esta situación, la República Bolivariana cuenta con el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC), institución que garantiza la eficiencia en la Investigación del delito, mediante su determinación científica, asegurando el ejercicio de la acción penal que conduzca a una sana administración de justicia. (2)

Constituye una necesidad para el gobierno de Venezuela controlar el uso dado a las sustancias químicas controladas por el Régimen Legal No. 4; que son aquellas que por sus propiedades químicas pueden ser utilizadas en la síntesis y elaboración de drogas. De esta manera se contribuye a disminuir las posibilidades de que en el país se produzcan o transporten drogas.

Como parte de la estructura organizativa del CICPC, se encuentra la División de Investigación y Fiscalización de Sustancias Químicas, perteneciente a la Dirección Contra Drogas. Esta división tiene la misión de registrar información, investigar y fiscalizar a las empresas que operan en territorio venezolano utilizando de alguna manera las sustancias químicas controladas por el Régimen Legal No. 4. (3)

Para la gestión de la información, el CICPC cuenta con el Sistema Integrado de Información Policial, carente de una interfaz sencilla y desarrollado sobre la base de una tecnología obsoleta, presenta problemas con las consultas y procesamiento de la información, lo que trae consigo la falta de información actualizada, oportuna y fiable a las entidades de la dirección del CICPC; por lo que no cubre con todas las necesidades de dicha institución.

[1] (Global Peace Index, GPI), un análisis global fundado en 23 indicadores, que propone una clasificación metódica de 153 países en función de su nivel de paz o tranquilidad.
[2] (Institute of Economics and Peace, IEP) Organización internacional independiente y no partidista. Instituto de investigación dedicado a la divulgación con mayor comprensión de los factores claves y métodos de medida de la paz, desde una perspectiva socioeconómica y macroeconómica.

Para corregir estas deficiencias, el CICPC y sus coordinaciones se ven en la necesidad de adquirir un sistema de gestión competitivo, más completo, con tecnología actualizada, donde se pueda gestionar la información generada de una manera confiable y eficaz, disminuir los tiempos de respuesta de las peticiones y cubrir los procesos que se desarrollan en el CICPC. Para dar solución a esta necesidad en el marco de las relaciones Cuba - Venezuela por la colaboración entre los países de la Alternativa Bolivariana para las Américas (ALBA), se firmó el proyecto de modernización del CICPC; el cual concibe la creación del Sistema de Investigación e Información Policial (SIIPOL).

Con el fin de crear un sistema que sustituya las prestaciones del Sistema Integrado de Información Policial, incorpore y mejore las funcionalidades del resto de las áreas operativas del CICPC; se realizó un análisis exhaustivo del funcionamiento de la institución, así como sus procesos de negocio, lo que generó un conjunto de requisitos funcionales y no funcionales, que resumen las capacidades y condiciones que debe cumplir el nuevo sistema.

En la actualidad el SIIPOL se encuentra en su versión 4, pero carece de funcionalidades que permitan gestionar la información relacionada a las empresas que operan sustancias químicas controladas por el Régimen Legal No. 4.

Planteada la **situación problemática** se puede enunciar el **problema a resolver** a partir de la siguiente interrogante: ¿Cómo garantizar el cumplimiento de los requisitos funcionales y no funcionales obtenidos en la Ingeniería de Requisitos en la División de Investigación y Fiscalización de Sustancias Químicas?

El **objeto de estudio** determinado para el presente trabajo de diploma se centra en el desarrollo de Sistemas de Gestión de Información Policial y el **campo de acción** se enmarca en el Sistema de Investigación e Información Policial.

La **idea a defender** se plantea: “Con el diseño e implementación del módulo Sustancias Químicas se garantizará el cumplimiento de los requisitos funcionales y no funcionales obtenidos en la Ingeniería de Requisitos”.

La presente investigación tiene como **objetivo general**: Diseñar e implementar el módulo Sustancias Químicas del Sistema de Investigación e Información Policial.

Para darle cumplimiento al objetivo general se desglosan los siguientes **objetivos específicos**:

- Elaborar el marco teórico orientado al desarrollo de sistemas de gestión de información policial.
- Realizar el diseño e implementación del módulo Sustancias Químicas respetando la arquitectura definida para el SIIPOL.
- Integrar satisfactoriamente el módulo implementado al SIIPOL.
- Validar la propuesta de solución.

Para cumplir con los objetivos propuestos anteriormente se han trazado las siguientes tareas investigativas:

1. Estudiar los conceptos sobre sistemas de gestión de información y sistemas basados en la gestión policial.
2. Investigar otras soluciones informáticas o aplicaciones similares a la que se desea desarrollar.
3. Estudiar la metodología, lenguaje, herramientas y frameworks a emplear en el desarrollo de la solución.
4. Estudiar el modelo de casos de usos asociado al módulo Sustancias Químicas.
5. Estudiar la arquitectura del SIIPOL propuesta por el equipo de arquitectura.
6. Confeccionar el modelo de diseño.
7. Confeccionar el modelo de implementación.
8. Implementar los componentes necesarios para el desarrollo del módulo Sustancias Químicas.
9. Probar los componentes desarrollados.
10. Integrar la solución al SIIPOL.
11. Verificar el correcto funcionamiento del módulo Sustancias Químicas integrado a los demás subsistemas del SIIPOL.

Para el desarrollo de la presente investigación se emplearon los siguientes métodos científicos teóricos:

- **Analítico–Sintético**

  Analizar significa desintegrar, descomponer un todo en sus partes para estudiar en forma intensiva cada uno de sus elementos, así como las relaciones entre sí y con el todo. (4)

  El método sintético es un proceso de razonamiento que tiende a reconstruir un todo, a partir de los elementos distinguidos por el análisis. (4)

En otras palabras, se realiza un análisis de los documentos generados en el levantamiento y captura de requisitos, donde se extraen y analizan los principales elementos relacionados con el objeto de estudio, para luego, reestructurar, recomponer el todo estableciendo las relaciones entre los componentes más significativos, o sea los elementos extraídos durante el análisis.

- **Modelación**

  Se manifiesta en la creación de los modelos y diagramas generados durante las diferentes etapas por las que transita el desarrollo de la aplicación; los cuales representan la propuesta de solución y permiten visualizar el sistema que se pretende desarrollar desde diferentes puntos de vista.

La investigación está estructurada en 3 capítulos que agruparán los contenidos de la siguiente manera:

Capítulo 1. Fundamentación Teórica: se tratan elementos teóricos de la investigación tales como: análisis de sistemas de gestión de información policial, metodología, lenguajes y herramientas de desarrollo que serán utilizadas para implementar la solución.

Capítulo 2. Diseño e Implementación de la propuesta de solución: se muestran los artefactos generados en el modelo de diseño y de implementación. Además, se explican elementos propios del sistema, que se adaptan a la arquitectura definida por el equipo de arquitectos del proyecto.

Capítulo 3. Validación de la solución: en este capítulo se presentan los resultados de las diferentes pruebas realizadas al módulo implementado, para verificar que cumple con los requisitos funcionales y no funcionales obtenidos en la captura de requisitos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

El objetivo de este capítulo es ofrecer el marco teórico en el que se desarrolla la presente investigación, creando un sólido basamento que será el preámbulo de los próximos capítulos. Para ello, se realiza un estudio de la situación actual en la que se encuentran los sistemas de gestión de información policial, lo que sirve como entrada para el análisis del entorno de la institución que se desea automatizar y los sistemas con los que se relaciona. También se realiza un estudio del ambiente de desarrollo seleccionado y establecido por la dirección del proyecto, y se especifican las principales características de la metodología, lenguajes y herramientas establecidas que se utilizan en la implementación de la solución.

1.2 CUERPO DE INVESTIGACIONES CIENTÍFICAS PENALES Y CRIMINALÍSTICAS

El CICPC es el organismo rector del enfrentamiento al delito en La República Bolivariana de Venezuela, integra las investigaciones de tipo científicas y criminalísticas sobre la base de las leyes vigentes y pone los resultados a disposición de los órganos judiciales del Ministerio Público, responsable legal de la investigación. Este organismo tiene como visión ser la Institución indispensable, por su reconocida capacidad científica y máxima excelencia de sus recursos, con la finalidad de alcanzar el más alto nivel de credibilidad nacional e internacional en la investigación del fenómeno delictivo organizado y criminalidad violenta. (2)

Para la gestión de la información el CICPC empleaba el Sistema Integrado de Información Policial, el cual no brindaba suficiente soporte al trabajo realizado en la División de Investigación y Fiscalización de Sustancias Químicas. En la actualidad, el CICPC emplea el Sistema de Investigación e Información Policial en su versión 4, aunque este, carece de un módulo que gestione la información referente a dicha división. Por tal motivo se hace necesaria para la División de Investigación y Fiscalización de Sustancias Químicas como parte de la Dirección Contra Drogas del CICPC, un sistema que gestione todos los procesos que realiza, de manera ágil y eficiente.

1.3 SISTEMAS DE GESTIÓN DE INFORMACIÓN

"Información: comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada." (5)

Se entiende por "*gestión de la información como un proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma."* (6)

Por lo tanto, la gestión de la información implica (7):

- **Determinar la información que se precisa**: Durante la planificación, gestión y supervisión de diferentes procesos se genera mucha información, por tanto, un buen sistema de gestión de la información debe, ayudar a los usuarios a saber qué información necesitan recabar, para tomar diferentes decisiones en distintos momentos.
- **Registrarla y recuperarla cuando sea necesaria**: Es importante guardar la información para futuras referencias. El principio más importante del registro de informaciones es la facilidad con la que pueden recuperarse.
- **Utilizarla**: Se puede utilizar para solucionar problemas, determinar recursos (cantidad y naturaleza) y planear futuros proyectos.

1.4 SISTEMAS DE GESTIÓN DE INFORMACIÓN POLICIAL

"La Seguridad Ciudadana es una situación social, donde predomina la sensación de confianza, entendiéndosela como ausencia de riesgos y daños a la integridad física y psicológica, donde el Estado debe garantizar la vida, la libertad y el patrimonio ciudadano." (8)

A partir de la bibliografía consultada, se entiende por sistema de gestión de información policial, a la *"herramienta informática que permite registrar la información referente a eventos policiales que ocurren dentro de determinado territorio, tales como: delitos, faltas, accidentes, entre otros".*

Los resultados del Índice de Paz Global para el 2011 sugieren que el mundo se ha hecho en poco menos tranquilo en el último año. (9) Dicho resultado hace más evidente y necesaria la digitalización y

centralización de la información policial. Con la implementación de un sistema de gestión de información policial se contribuye a lograr una mayor eficiencia en las gestiones de seguridad ciudadana, lo cual beneficia la calidad de vida en la población. Los sistemas de gestión de información policial agilizan los procesos de investigación y esclarecimiento de hechos delictivos, puesto que permiten interconectar otros centros policiales, brindan información actualizada y de manera rápida, logrando así mayor capacidad de acción. Además, los datos registrados, constituyen una fuente de información de alto valor para la gestión policial y para la toma de decisiones en la institución u organismo que los emplee.

Con el objetivo de conocer la existencia de soluciones informáticas o aplicaciones similares a la que se desea desarrollar se realizó una búsqueda detallada, de la misma resultaron los siguientes sistemas:

1.4.1 SISTEMA INTEGRADO DE INFORMACIÓN POLICIAL

El Sistema Integrado de Información Policial es la aplicación que usaban las dependencias, delegaciones y subdelegaciones del CICPC, entre ellas la División de Investigación y Fiscalización de Sustancias Químicas. Está desarrollado sobre una tecnología obsoleta con servidores de Base de Datos SUN 6500 y base de datos Adabas, con lenguaje de programación Natural para el manejo de los datos; el acceso a este sistema se hace a través de un emulador[3], como Personal Comunication, o un Telnet[4]. El servidor de la aplicación es con tecnología de Sun y Sistema Operativo Solaris. Utilizan un programa llamado Natural Security que es el que valida la gestión de los usuarios para la seguridad del SIIPOL. (10)

1.4.1.1 SISTEMA DE INVESTIGACIÓN CONTROL Y REGISTRO DE OPERADORES QUÍMICOS (SICROQ)

El SICROQ como parte del Sistema Integrado de Información Policial, es un software que fue creado para auxiliar los procesos desarrollados en la División de Investigación y Fiscalización de Sustancias Químicas.

Este sistema no es capaz de registrar un expediente para cada empresa, donde se almacene toda la información utilizada por los investigadores, en apoyo a las labores investigativas relacionadas con las empresas que utilizan sustancias químicas controladas. No permite almacenar los datos de las

[3] Emulador: Es un software que permite ejecutar programas de computadora en una plataforma (arquitectura hardware o sistema operativo) diferente de aquella para la cual fueron desarrollados originalmente.
[4] Telnet: Es una de las formas de conectarse a otra computadora por medio de llamada directa del ordenador terminal al ordenador host y ejecutar secuencias de comandos, al estilo DOS o UNIX.

sucursales, plantas, depósitos, vehículos y choferes de estas empresas, ni brinda la posibilidad de registrar información sobre las actas policiales, las actas de fiscalización o las actas de inspección a empresas, siendo estos, datos y documentos imprescindibles para contribuir con las labores de registro y control de esta división. (3)

Los funcionarios de la División de Investigación y Fiscalización de Sustancias Químicas necesitan una herramienta automatizada que muestre la fecha en la cual una empresa vence su Permiso de Utilización de Sustancias Químicas y el SICROQ no ofrece esta funcionalidad. Para una mejor gestión de la información se requiere comunicación constante con el SIIPOL y otros sistemas informáticos que ofrecen información de interés para las investigaciones realizadas, y el sistema actual no ofrece tal conexión. En muchas ocasiones producto a esta falta de comunicación, se le otorga el permiso a una empresa que está siendo investigada por otra división o que pertenece a una persona incriminada en algún tipo de delito, cuestión que no debería ocurrir según lo establecido por la ley. (3)

De manera general, el SICROQ no cumple con las necesidades informativas de la División de Investigación y Fiscalización de Sustancias Químicas para combatir el uso y la distribución de la droga.

1.4.2 SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL

En la actualidad el CICPC emplea el SIIPOL en su versión 4, este sistema es la sustitución del Sistema Integrado de Información Policial. El SIIPOL cuenta con diferentes módulos, que responden a las necesidades de las distintas áreas con las que cuenta la institución; dicha estructura permite el trabajo en conjunto, sincronizado y de manera rápida entre las diferentes oficinas del CICPC en disímiles regiones del estado. Además, el SIIPOL provee a la institución una conexión con otros sistemas de gestión de información, como el INTT[5] para el control de vehículos y SAIME[6] para el control de personas. También facilita a la institución estadísticas a partir de los datos gestionados por las distintas divisiones y departamentos, lo que permite la generación de gráficos que ilustren los datos.

Aunque el SIIPOL sustituyó los procesos que realizaba el Sistema Integrado de Información Policial, es importante señalar que la versión 4, aún no cuenta con un módulo que cumpla con las necesidades de la División de Investigación y Fiscalización de Sustancias Químicas.

[5] Instituto Nacional de Transporte Terrestre.
[6] Servicio Administrativo de Identificación, Migración y Extranjería.

El SIIPOL cuenta con una arquitectura sólida, está estructurado de manera modular y desarrollado en 3 capas. El entorno de desarrollo, definido para la construcción del SIIPOL, fue producto de un estudio realizado por el equipo de arquitectura, y su uso, establecido como políticas del proyecto; el mismo está compuesto esencialmente por:

- La metodología de desarrollo de software, Proceso Unificado del Software (RUP).
- El lenguaje de modelado, Lenguaje Unificado de Modelado (UML).
- La herramienta CASE, Visual Paradigm 8.
- El lenguaje de programación, Java.
- La plataforma de desarrollo, Java Enterprise Edition (JEE).
- El Entorno Integrado de Desarrollo (IDE), Eclipse.
- Los frameworks, Java Server Faces (JSF) 1.1 en la capa de presentación, Spring 3.1 en la capa de negocio e Hibernate 4.1.2 en la capa de acceso a datos.
- Gestor de bases de datos relacional, Oracle 10g Release 2.

1.5 METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO

Para la construcción del módulo Sustancias Químicas se emplea el ambiente de desarrollo seleccionado para el SIIPOL, software al que se integrará la solución propuesta. A continuación se mencionan las características más relevantes del ambiente de desarrollo empleado, enfatizando las que tienen mayor impacto en el desarrollo de la solución.

1.5.1 METODOLOGÍA DE DESARROLLO

Se define la metodología como: *"el conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal."* (5)

"El proceso de desarrollo de software es aquel en que las necesidades del usuario son traducidas en requisitos de software, los mismos son transformados en diseño y el diseño implementado en código; el código es probado, documentado y certificado para su uso operativo. Concretamente 'define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo'". (11)

Para cumplir con el objetivo de producir eficiente y eficazmente un software que cumpla los requisitos del cliente; es necesario un proceso que sirva como guía para todos los participantes clientes, usuarios, desarrolladores y directores ejecutivos. A partir de esta necesidad la dirección del proyecto realizó un estudio de las diferentes metodologías existentes para guiar el proceso de desarrollo de software.

A la hora de tomar la decisión de cuál metodología de desarrollo usar en la solución, la dirección del proyecto tuvo en cuenta varios puntos definitorios, como: la magnitud del proyecto, la cantidad de miembros del equipo, el tiempo de que se disponía para su culminación, lo distante que se encuentran desarrolladores y clientes geográficamente, las expectativas del cliente y la experiencia de los desarrolladores, entre otros. La magnitud del software a producir hace que las partes a entregar deban quedar bien documentadas internamente; los formalismos se hacen necesarios para prevenir malentendidos que no son deseables en el marco en el que se desenvuelve la producción del software. Por otra parte, el cliente espera un producto que contenga manuales y documentación; este requisito es cumplido de manera más efectiva empleando como metodología de desarrollo el Proceso Unificado de Desarrollo (RUP).

1.5.1.1 PROCESO UNIFICADO DE DESARROLLO (RUP)

El Proceso Unificado del Software, RUP (Rational Unified Process), es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software. (...) utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. De hecho, UML es una parte esencial del Proceso Unificado. (12)

Los autores del Proceso Unificado de Desarrollo destacan que los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases clave —dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al Proceso Unificado. (12)

- **Dirigido por casos de uso**

  Un caso de uso (CU) es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. (…) Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo —avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba

construyen sus casos de prueba. Aunque es cierto que los casos de uso guían el proceso, no se desarrollan aisladamente. Se desarrollan a la vez que la arquitectura del sistema. Es decir, los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso. (12)

- **Centrado en la arquitectura:**

  El papel de la arquitectura software es parecido al papel que juega la arquitectura en la construcción de edificios. (...) Análogamente, la arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción. El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema. (...) Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el software (arquitectura hardware, sistema operativo, sistema de gestión de base de datos, protocolos para comunicaciones en red), (...) consideraciones de implantación, sistemas heredados, y requisitos no funcionales (por ejemplo, rendimiento, fiabilidad). Podemos pensar que la arquitectura de un sistema es la visión común en la que todos los empleados (desarrolladores y otros usuarios) deben estar de acuerdo, o como poco, deben aceptar. La arquitectura nos da una clara perspectiva del sistema completo, necesaria para controlar el desarrollo. (12)

- **Iterativo e incremental:**

  Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deber estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son mini-proyectos. (12)

Estos conceptos —los de desarrollo dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental— son de igual importancia. La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. La eliminación de una de las tres ideas reduciría drásticamente el valor del Proceso Unificado. Es como un taburete de tres patas. Sin una de ellas, el taburete se cae. (12)

La figura 1 representa a RUP en sus dos dimensiones:

- El eje horizontal representa el tiempo en las cuatro fases en las que se descompone el proceso.

- El eje vertical representa la serie de flujos de trabajo que lo construyen gradualmente.

El desarrollo del presente trabajo se centra principalmente en las fases de Elaboración y Construcción y se describe en los flujos de trabajo Diseño e Implementación.

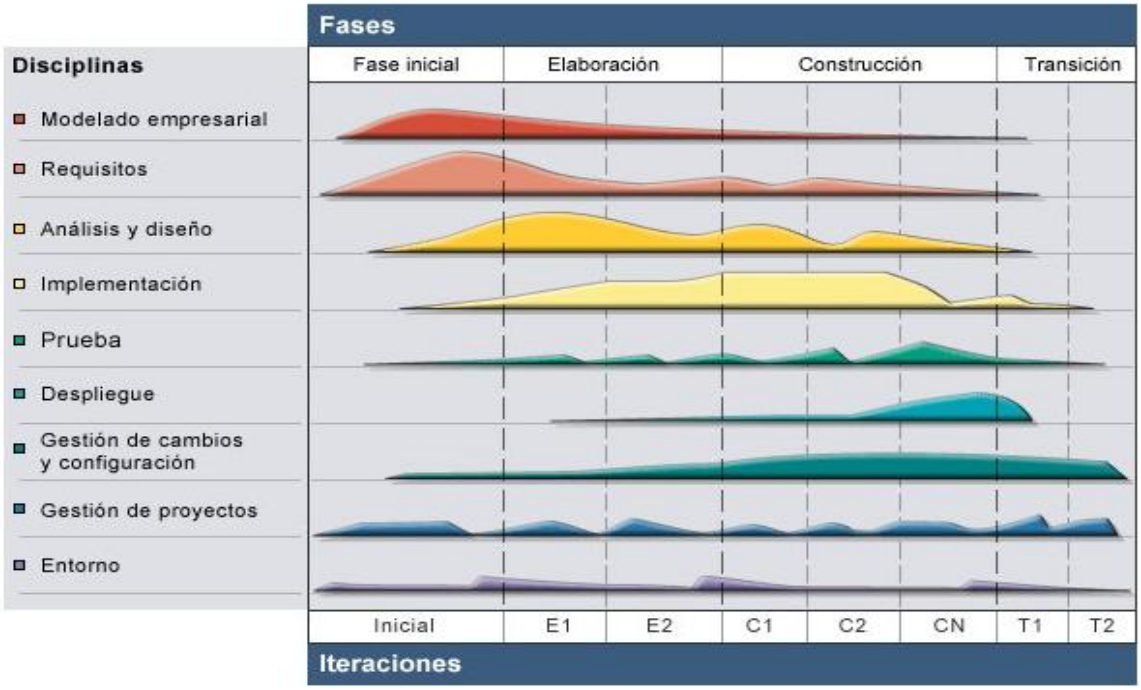


Figura. 1 RUP en 2 Dimensiones.

Fases del proceso en las que se centra la investigación (12):

- **Elaboración:** se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo con el alcance definido.
- **Construcción:** se logra un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene una o varias versiones del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

Flujos de trabajo que describen la investigación (12):

- **Análisis y diseño:** describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación. Durante el desarrollo de este flujo de trabajo se genera el artefacto modelo de implementación.

1.5.2 LENGUAJE DE MODELADO

El modelo capta los aspectos importantes de lo que estamos modelando, desde cierto punto de vista, y simplifica u omite el resto. La ingeniería, la arquitectura y muchos otros campos creativos usan modelos. (...) Los diversos modelos de un sistema de software pueden capturar requisitos sobre su dominio de aplicación, las formas en que los usuarios lo utilizarán, su división en módulos, los patrones comunes usados en su construcción, y otras cosas. (13)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. (...) La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (13)

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. (13)

1.5.3 HERRAMIENTAS CASE

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) 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. Estas herramientas

pueden ayudar 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, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (14)

En la actualidad las tecnologías CASE han ayudado a perfeccionar la calidad y la productividad en el desarrollo de Sistemas de Gestión, entre sus principales objetivos están (14):

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.5.3.1 VISUAL PARADIGM FOR UML 8.0

Visual Paradigm para el Lenguaje Unificado de Modelado (VP-UML) está diseñado para una amplia gama de usuarios, tales como: ingenieros de software, analistas de sistemas, analistas de negocio y arquitectos de sistemas, o para cualquiera que esté interesado en la construcción de forma fiable y a gran escala de sistemas que utilizan un enfoque orientado a objetos. (15)

Visual Paradigm es una herramienta CASE profesional que de igual forma, soporta el ciclo de vida completo del desarrollo de software (…). Ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Utiliza UML como lenguaje de modelado y se integra con herramientas Java tales como: Eclipse, JBuilder, (…) entre otras. Es una herramienta que genera la documentación del proyecto automáticamente en varios formatos como Web o .pdf, soporta la realización de ingeniería tanto directa como inversa y permite control de versiones. Presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas (…). (16)

1.5.4 PLATAFORMA DE DESARROLLO

Una plataforma Java consiste en una máquina virtual (VM) y una interfaz de programación de aplicaciones (API). La máquina virtual de Java es un programa, para un hardware y plataforma de software en particular, que ejecute una aplicación Java. Una API es una colección de componentes de software que se puede utilizar para crear otros componentes de software o aplicaciones. Cada plataforma Java proporciona una máquina virtual y un conjunto de APIs, esto permite que las aplicaciones escritas para la plataforma puedan ejecutarse en cualquier sistema compatible con todas las ventajas del lenguaje de programación Java. (17)

La plataforma Java EE está diseñada para reducir la complejidad del desarrollo de aplicaciones empresariales, proporcionando un modelo de desarrollo, APIs, y entorno de ejecución que permita a los desarrolladores concentrarse en las funcionalidades. (17)

Esta plataforma se desarrolla a través del Java Community Process (JCP), que es el responsable de todas las tecnologías Java. Al trabajar bajo el programa JCP ayuda a asegurar los estándares de la tecnología Java para una estabilidad y compatibilidad entre plataformas. (18)

La plataforma Java EE (Java Enterprise Edition, J2EE hasta la JVM 1.5) incluye varias especificaciones de APIs, tales como: Java Server Faces Technology, Java Server Pages Technology, Java Server Pages Standard Tag Library, Java Database Connectivity (JDBC) entre otras, y define cómo coordinarlas. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel. (19)

Básicamente la selección de Java EE como plataforma proporciona todas las herramientas necesarias para desarrollar y mantener la aplicación objeto de este proyecto, la gran mayoría en software libre y con un soporte comunitario que asegura su estabilidad. Es una plataforma madura, que permite el desarrollo de aplicaciones exigentes en cuanto a servicios y a la calidad de los mismos, rica en funcionalidades y debidamente estandarizada. Aunque la plataforma Java EE no es un estándar todos los proveedores de productos deben regirse por la especificación del API sobre el cual están desarrollando para que su producto sea declarado compatible con la plataforma. (19)

1.5.5 LENGUAJES DE PROGRAMACIÓN

Java es un lenguaje de programación que compila en código intermedio interpretable por una máquina virtual. Algunas de sus principales características son (19):

- **Portabilidad**: Permite desarrollar con un ambiente y desplegar en otro, más exactamente, desarrollar sobre Windows, probar en Linux y desplegar en HP-UX, los tres sistemas operativos utilizados en el desarrollo del proyecto corriendo cada uno sobre un hardware diferente.
- **Seguridad**: Permite que cualquier error por parte de la aplicación no pueda comprometer la seguridad ni la estabilidad del sistema en su totalidad. Todos los errores son registrados de alguna manera por las herramientas utilizadas, de forma que la información puede ser utilizada para corregir el error.
- **Manejo de memoria**: Java es un lenguaje administrado, lo que significa que la memoria utilizada por las aplicaciones no es manejada por ellas sino por la máquina virtual subyacente.

Java ha sido mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores (20):

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Crear programas para que funcionen en un navegador web y en servicios web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital.

1.5.6 ENTORNO INTEGRADO DE DESARROLLO (IDE)

Los IDEs (Integrated Development Environment) son un conjunto de herramientas para el programador, que suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos,

enlace transparente a compiladores y debuggers e integración con sistemas controladores de versiones o repositorios. (21)

El Eclipse es una plataforma de desarrollo de software multilenguaje que contiene un IDE y un sistema de plug-ins para extenderlo. Está escrito en Java y brinda la funcionalidad de programar en este lenguaje por defecto a través del plug-in básico Java Development Tools (JDT). Además de extender los lenguajes que puede soportar el IDE, el sistema de plug-ins permite adicionar otras funcionalidades útiles como editores visuales de distintos tipos de archivos, internacionalización, conexión con repositorios de control de versiones, etc. El Eclipse se distribuye bajo la Eclipse Public License, y es por tanto código abierto y software libre. Debido a este sistema, el IDE Eclipse es muy popular y existen plug-ins de muchos tipos para disímiles funciones. Las principales compañías detrás de los frameworks utilizados (fundamentalmente SpringSource y JBoss) proveen plug-ins que aceleran el desarrollo de software utilizando Eclipse como base. (19)

El IDE utilizado para el desarrollo de la presente investigación es Helios (versión 3.6 de Eclipse), esta versión reúne un conjunto de plug-ins (provistos por defecto y otros adicionados por el equipo de arquitectura), como: WTP (permite desarrollo web, adicionando soporte para servidores), SpringIDE (para trabajo con los contexto de Spring), Subeclipse (para integración con el control de versiones), JBoos Tools (que ofrece un conjunto de módulos como: Visual Page Editor, Hibernate Tools, JST/JSF Tools) entre otros. (19) Otra ventaja del uso del Helios es su posibilidad de integración con Apache Tomcat 7.

1.6 FRAMEWORKS UTILIZADOS

“Un framework es una herramienta que provee servicios básicos a las aplicaciones que lo utilizan, define una arquitectura para la aplicación y reduce la cantidad de código cliente necesario utilizando código preprogramado para abstraer al programador de las tareas más simples permitiendo que se concentre en resolver los problemas del negocio...” (22)

Por tanto un framework constituye la base sobre la cual se puede construir una aplicación, y como toda base su selección es tan importante e influyente en la arquitectura como las especificaciones funcionales y no funcionales.

1.6.1 JAVA SERVER FACES (JSF) 1.1

La tecnología Java Server Faces es un framework que proporciona un grupo de componentes para la creación de aplicaciones web basadas en tecnología Java. Esta tecnología consiste en un conjunto de APIs para representar los componentes, la gestión de sus estados, control de eventos, validaciones del lado del servidor y conversión de datos entre otras. Además, brinda una biblioteca de etiquetas para la adición de componentes a las páginas web y conectar estos a los objetos del lado del servidor. (23)

JSF proporciona un modelo de programación bien definido y varias bibliotecas de etiquetas. Estas características hacen significativamente fácil la construcción y mantenimiento de aplicaciones web. Lo que permite con un mínimo de esfuerzo realizar las siguientes tareas (23):

- Crear una página web.
- Colocar componentes en una página web mediante la adición de etiquetas de componentes.
- Vinculación de componentes con los datos del lado del servidor.
- Establecer la conexión de los eventos generados en los componentes con en el código de la aplicación en el lado del servidor.

La figura 2 muestra la interacción cliente-servidor en una típica aplicación Java Server Faces. En respuesta a una petición del cliente, la página web es representada por el contenedor web que implementa JSF. (24)

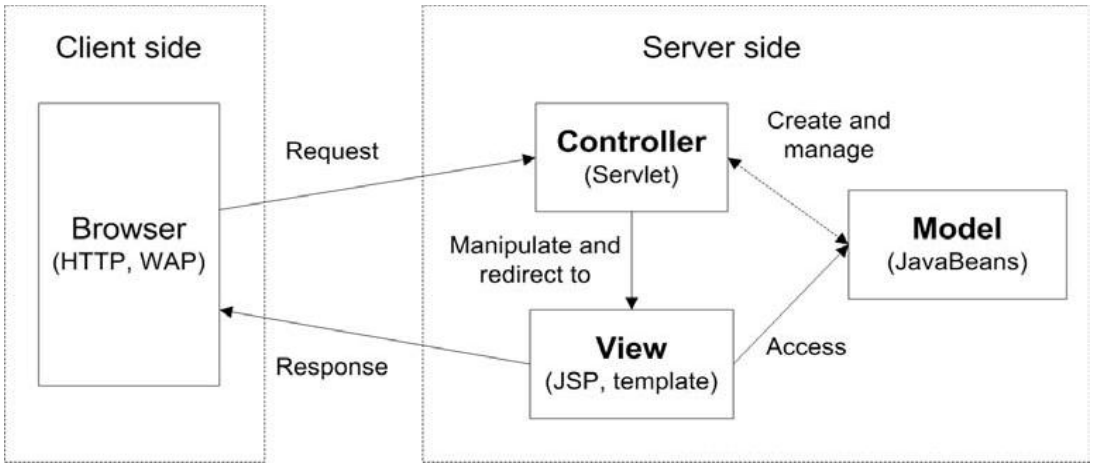


Figura. 2 Respuesta a petición del cliente por JSF.

JSF está basado en el patrón de diseño Modelo Vista Controlador (MVC) y utiliza Java Server Pages (JSP). El empleo de este framework permite procesar los eventos lanzados por los usuarios (por ejemplo, el clic en un botón de la interfaz), mediante la declaración de los event listeners (funciones que se encargan de "escuchar" el lanzamiento de un evento determinado y ejecutar un acción específica).

RichFaces es un framework de código abierto que integra funcionalidades Ajax[7] en aplicaciones JSF sin recurrir a JavaScript. (25)

Son características de RichFaces las siguientes (25):

- Se integra perfectamente en el ciclo de vida de JSF.
- Incluye funcionalidades Ajax, de modo que nunca vemos el JavaScript y tiene un contenedor Ajax propio.
- Contiene un set de componentes visuales, los más comunes para el desarrollo de una ricas aplicación web.
- Soporta css[8].

1.6.2 SPRING 3.1

Spring es un framework modular, lo que permite utilizar sólo aquellas partes que se necesitan, sin tener que agregar el resto. (...) Este Framework admite la gestión de transacciones y diversas opciones para la persistencia de sus datos. También permite integrar de forma transparente AOP (aspect oriented programming, programación orientada a aspectos) a la solución. Spring está diseñado para ser no intrusivo, lo que significa que el código de la lógica de dominio, no tiene ninguna dependencia del propio framework. (26)

Como se puede apreciar en la figura 3, Spring se compone de elementos organizados en alrededor de 20 módulos. Estos módulos se agrupan en: Núcleo principal (Core Container), Acceso a datos / Integración (Data Access / Integration), Web, AOP, Instrumentación (Instrumentation), Prueba (Test). El empleo de Spring no significa basar la aplicación completamente en el framework, el usuario es libre de elegir los módulos que se adapten a su aplicación.

[7] Del término en inglés Asincronous Java Script and XML.
[8] Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla.

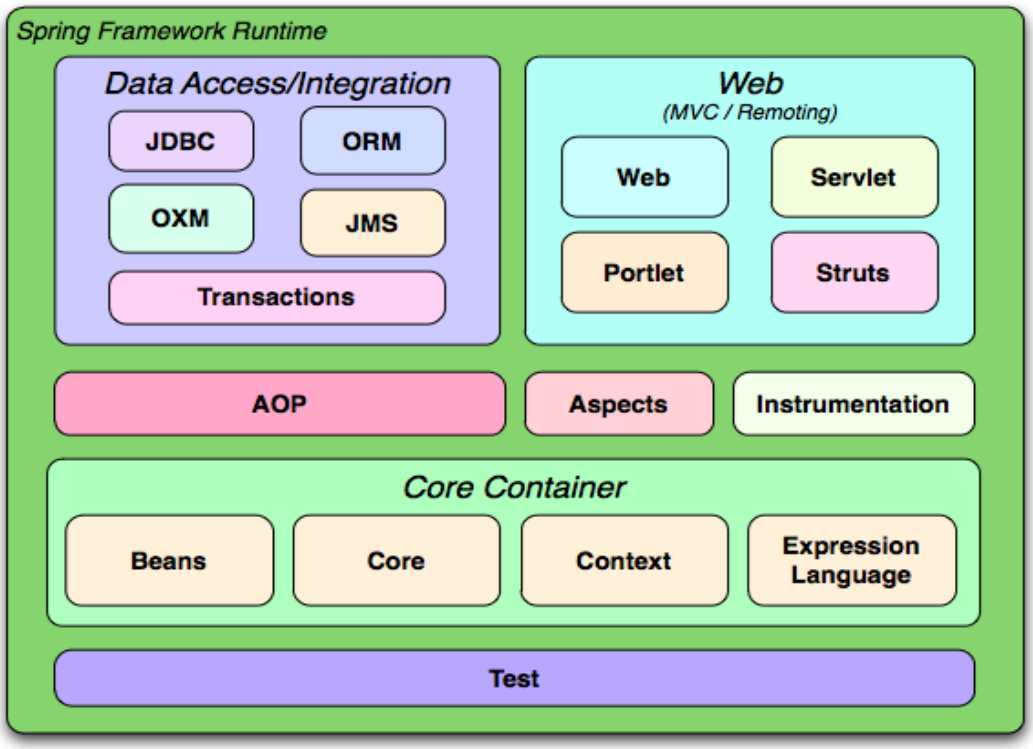


Figura. 3 Estructura de Spring.

Para entender mejor que es Spring a continuación se describen algunas de sus principales características (27):

- **Inyección de dependencia**: Spring promueve un bajo acoplamiento a través de la inyección de dependencias. Cuando se aplica esta técnica, Spring entrega a los objetos sus dependencias. De esta manera se elimina el acoplamiento existente entre un objeto y las dependencias que el framework evitó crear y/o buscar en el mismo. El contenedor de Inversión de Control de Spring (o Inyección de dependencias) se encuentra implementado en los paquetes Beans y Context del Core Container.
- **Orientada a aspectos**: Spring brinda un amplio soporte para la programación orientada a aspectos, lo que permite un desarrollo cohesionado por la separación en la aplicación de la lógica de negocio y los servicios del sistema.

1.6.3 HIBERNATE 4.1.2

Hibernate es una solución que hace uso del Mapeo Relacional de Objetos (ORM) para entornos Java. El término ORM se refiere a la técnica que mapea una representación de datos de un modelo de objetos a un modelo de datos relacional con un esquema basado en SQL[9] (y viceversa).

Hibernate no sólo se encarga de la asignación de las clases Java a tablas de la base de datos (y de tipos de datos Java a tipos de datos SQL), sino que también proporciona consulta y facilidades de recuperación. El objetivo principal de este framework es aliviar al desarrollador de un 95 por ciento de las tareas comunes de persistencia de datos relacionadas con la programación, de esta manera se puede reducir significativamente el tiempo de desarrollo que se emplearía con la manipulación manual de datos en SQL y JDBC. (28)

Hibernate soporta un lenguaje de consulta orientado a objetos (HQL[10]) fácil de usar pero potente a la vez. (…) HQL es extremadamente potente pero algunos desarrolladores prefieren construir consultas dinámicamente utilizando una API orientada a objetos, en vez de construir cadenas de consulta. Hibernate brinda una API intuitiva de consulta Criteria[11] para estos casos. (28)

La figura 4 ejemplifica cómo Hibernate está diseñado para operar con diferentes entornos, donde el framework hace de enlace entre una aplicación que necesita persistir su modelo de objetos y la base de datos. Debido a que es tan genérico, existen muchos parámetros de configuración, afortunadamente, la mayoría de estos parámetros tienen valores predeterminados y se distribuyen un archivo *hibernate.properties*. Este archivo de configuración (que no es el único que proporciona el framework) junto a los archivos de Mapeos XML, permiten al framework saber cómo cargar y guardar los objetos de clases persistentes asignados a tablas de la base de datos. (28)

[9] Por sus siglas en inglés, Lenguaje de Consultas Estructurado.
[10] Por sus siglas en inglés, Hibernate Query Language.
[11] La interfaz *org.hibernate.Criteria* representa una consulta contra una clase persistente en particular.

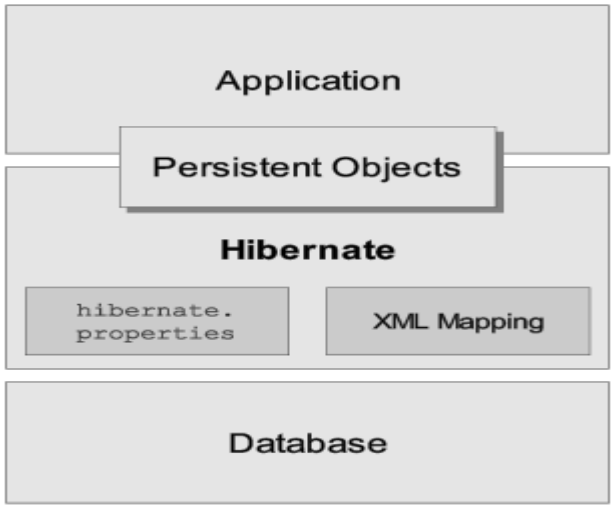


Figura. 4 Perspectiva a alto nivel de la arquitectura de Hibernate.

1.7 SISTEMA GESTOR DE BASE DE DATOS

Oracle es un sistema de gestión de base de datos relacional (RDBMS por el acrónimo en inglés Relational Data Base Management System). Se considera uno de los sistemas de bases de datos más completos. Es un robusto sistema gestor de base de datos multiplataforma, con un conjunto de características que garantizan: la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta y sin causar inconsistencias; que ayudan a administrar y almacenar grandes volúmenes de datos y brindan estabilidad y escalabilidad. (29)

Algunas de las principales características de Oracle son (29):

- **Versatilidad**: Por su cualidad de multiplataforma se pueden crear códigos en Java o en C++ utilizando dicha base de datos tanto en Windows como en Linux. Exporta los datos y los migra desde una plataforma a la otra sin causar problemas de integridad o pérdida de datos, migra con su propio software tanto de SQL Server como de MySQL funcionando sobre cualquier plataforma libre, por lo que es muy útil para los programadores.
- **Potencia**: Ofrece un rendimiento mucho mayor que cualquier otra plataforma de Base de Datos. Permite al administrador asignar sus propias zonas de memoria a los datos y cualidades, se controla en todo momento; tanto el crecimiento como el rendimiento de los distintos esquemas que

componen una Base de Datos sobre Oracle, aunque esto suponga un problema, ya que los administradores deben estar pendientes de su configuración para no sufrir fallos debido a algún problema de almacenamiento.

- **Seguridad**: La seguridad de Oracle como sistema gestor de base de datos es alta, pues al no tener 100 por ciento de integración con Windows lo hace invulnerable a los defectos que posee el sistema operativo, además de poseer un sistema de seguridad muy avanzado.

1.8 PROPUESTA DE SOLUCIÓN

Para materializar la idea que defiende la presente investigación y responder a la situación problemática antes planteada, haciendo uso del ambiente de desarrollo seleccionado para la construcción del SIIPOL, se propone como solución: la creación de un módulo (Sustancias Químicas) que se integre plenamente al SIIPOL: aplicación Web cliente-servidor, donde los usuarios se conectan a través de protocolo seguro (HTTPS) a una Red Privada Virtual (VPN). La misma estará alojada en un servidor de aplicaciones Apache Tomcat 7.0.22 y se conectará a la base de datos, mediante la familia de protocolos TCP/IP. Dicha aplicación estará desarrollada según el patrón n-capas, específicamente en tres niveles, con la integración de los frameworks JSF, Spring e Hibernate (ver figura 5).

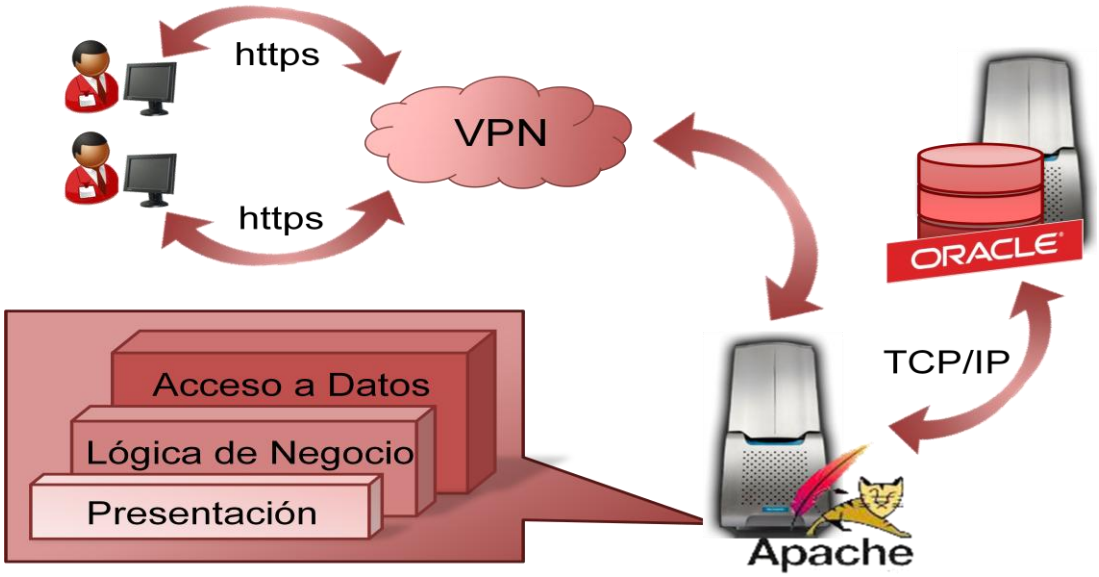


Figura. 5 Propuesta de solución.

1.9 CONCLUSIONES

Con la culminación del presente capítulo se conformó el marco teórico en el que se desarrolla la presente investigación, marco en el que se apoyan los próximos capítulos. Se presentó al CICPC como organismo rector del enfrentamiento al delito en La República Bolivariana de Venezuela y se demostró mediante un estudio realizado a diversos sistemas policiales (como: el Sistema Integrado de Información Policial, el SICROQ y el SIIPOL); la necesidad que presenta la División de Investigación y Fiscalización de Sustancias Químicas de un sistema que gestione todos los procesos que realiza, de manera ágil y eficiente.

También se realizó un estudio de la metodología, lenguajes y herramientas seleccionadas por el equipo de arquitectura. Este estudio arrojó que:

1. La elección correcta y más ventajosa para desarrollar el SIIPOL es la de RUP, puesto que esta metodología constituye una guía de cómo se debe desarrollar una aplicación de tal escala. Permitiendo el seguimiento y monitoreo de un proceso de desarrollo que agilice el trabajo de los implicados en el proyecto, lo cual resultará en la obtención de un producto con la calidad requerida.
2. Los frameworks elegidos para el perfil tecnológico del proyecto son líderes en sus campos. Todos son gratis, de código abierto, y con amplio soporte comunitario y empresarial.
3. El gestor de base de datos empleado es uno de los más poderosos de los existentes en el mercado, lo que da robustez y seguridad al producto.
4. Con la culminación de este capítulo se obtuvieron los elementos conceptuales necesarios para iniciar la construcción de un software que cumpla con los objetivos propuestos.

CAPÍTULO 2. DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE SOLUCIÓN

2.1 INTRODUCCIÓN

El presente capítulo tiene como objetivo principal el diseño e implementación de la propuesta de solución. Primeramente, como entrada al diseño, se realiza un análisis a partir de los artefactos generados en la Ingeniería de Requerimientos realizada por el equipo de analistas. También se explican cuestiones propias del sistema, que se adaptaron a la arquitectura definida por los arquitectos del proyecto. Además se exponen los diagramas generados durante el proceso de diseño de la solución, los cuales conforman fundamentalmente el modelo de diseño y el modelo de datos. Asimismo se exponen los diagramas generados en el flujo de trabajo implementación que conforman el modelo de implementación.

2.2 ENTRADA AL DISEÑO

Como entrada al diseño de la solución, se hace un estudio de los diagramas de casos de usos del sistema a desarrollar. Dicho estudio se realiza con el objetivo de comprender cómo interactúan los casos de usos (CU) entre sí (y con los diferentes actores del negocio). Mientras se realiza el estudio de los diagramas de casos de usos del sistema, se analizan los requisitos funcionales asociados a cada CU y los requisitos no funcionales del sistema. El objetivo de hacer ambas tareas en paralelo es conseguir una comprensión más precisa de los requisitos del sistema; ya que al tener una mejor comprensión de los mismos, se facilita la realización del diseño de la solución.

2.2.1 PROCESO DE NEGOCIO

El proceso de negocio se inicia en el momento que el Gestor de Solicitudes de la Empresa realiza la Solicitud de Permiso de Utilización de Sustancias Químicas. A través de dicho documento la empresa formaliza la solicitud de permiso de utilización de las sustancias químicas que emplea y están recogidas en el Régimen Legal No. 4. Para hacer dicha solicitud, el Gestor de Solicitudes de la Empresa debe especificar las sustancias químicas y las cantidades que utilizará, el motivo por el que hace la solicitud, las

actividades comerciales que realiza, y en caso de ser Transportista Importadora (tipo de actividad comercial) especificar los vehículos que posee. Una vez realizada dicha solicitud, el Gestor de Permisos de Utilización de Sustancias Químicas la revisa y otorga el Permiso de Utilización de Sustancias Químicas a la empresa solicitante (en caso de que la empresa o el representante legal de la misma no posean alguna investigación penal activa). El principal resultado de dicho proceso es la creación de la Carpeta Empresa, carpeta que se sustanciará con solicitudes tales como: Solicitudes de Inclusión, Solicitudes de Extensión, Solicitudes de Renovación y Solicitud de Inspección, para asociar nuevas sustancias químicas al permiso de utilización, aumentar la cantidad a utilizar de alguna sustancia química autorizada, renovar el permiso de utilización otorgado una vez vencido, o solicitudes de inspecciones que se le han solicitado a la empresa. Así mismo se registrarán en la Carpeta Empresa los permisos otorgados a la Empresa en respuesta a las solicitudes realizadas.

2.2.2 DIAGRAMAS DE CASOS DE USO

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Es importante resaltar que los diagramas de casos de uso no están pensados para representar el diseño y no puede describir los elementos internos de un sistema. Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, pero no cómo. (30)

El equipo de analistas del proyecto, dividió el módulo Sustancias Químicas en 3 submódulos (Solicitudes, Experticias y Control de Investigación), a continuación se muestran los diagramas de casos de usos pertenecientes a cada submódulo.

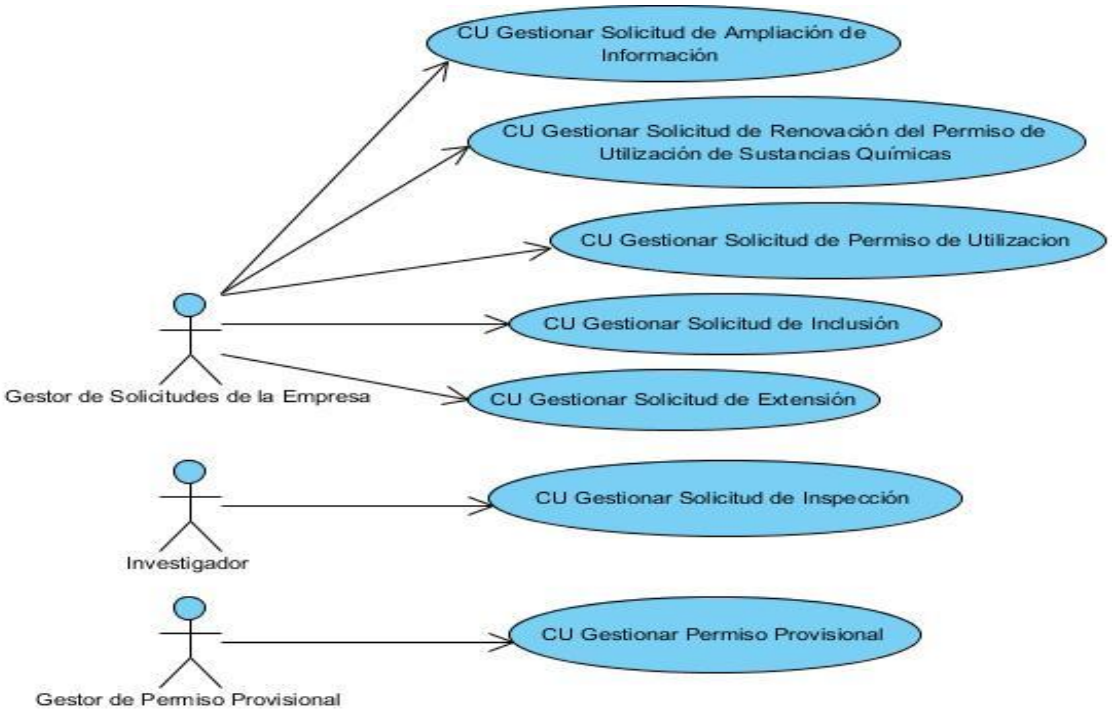


Figura. 6 Diagrama de Casos de Uso. Submódulo Solicitudes.

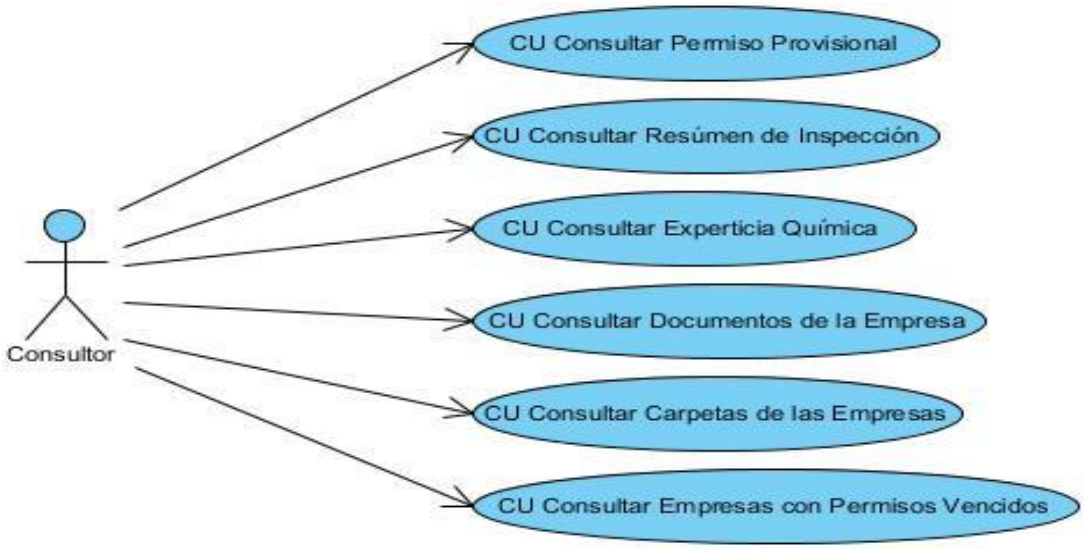


Figura. 7 Diagrama de Casos de Uso. Submódulo Control de Investigación.

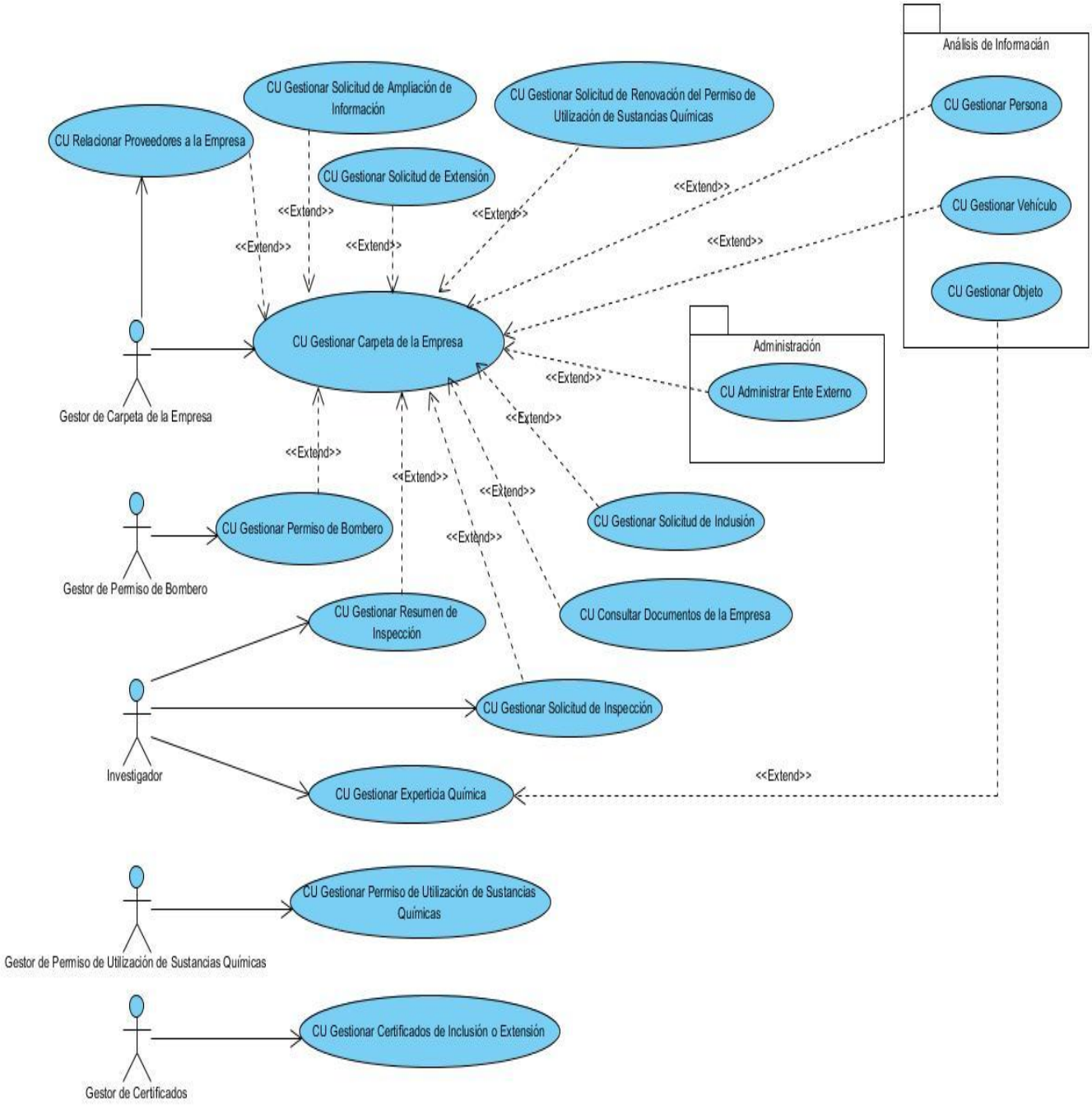


Figura. 8 Diagrama de Casos de Uso. Submódulo Experticia.

Seguidamente se muestra la descripción reducida del CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas del módulo Sustancias Químicas:

Caso de Uso	Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas
Objetivo	Crear, modificar y ver una Solicitud de Permiso de Utilización de Sustancias Químicas.
Actores	Gestor de Solicitudes de la Empresa: (Inicia) Crea, modifica y ve una Solicitud de Permiso de Utilización de Sustancias Químicas.
Resumen	El caso de uso se inicia cuando el actor accede a la opción que le permite crear, modificar o ver una Solicitud de Permiso de Utilización de Sustancias Químicas. Si el actor desea crear una Solicitud de Permiso de Utilización de Sustancia Químicas el sistema permite asociar un Ente Externo como Empresa e introducir y seleccionar los restantes datos de la solicitud. Si el actor desea modificar una Solicitud de Permiso de Utilización de Sustancias Químicas, el sistema muestra sus datos y permite editarlos. Si el actor selecciona la opción de ver una Solicitud de Permiso de Utilización de Sustancias Químicas, el sistema muestra sus datos y permite imprimirlos. Termina el caso de uso.
Complejidad	Alta.

2.2.3 REQUISITOS

Los requisitos son los que especifican las condiciones o capacidades que el sistema debe cumplir estos a su vez se pueden clasificar en dos grupos: requisitos funcionales y requisitos no funcionales. (12)

2.2.3.1 REQUISITOS FUNCIONALES

Los requisitos funcionales que a continuación se mencionan, están incluidos en los casos de usos del módulo Sustancias Químicas.

CU. Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas. (31)

- **RF1** - Incluir una Solicitud de la Empresa.
- **RF2** - Asociar un Ente Externo a la solicitud.
- **RF3** - Asociar vehículos a la solicitud.
- **RF4** - Crear una Carpeta de la Empresa.

- **RF5** - Ver una solicitud de la Empresa.
- **RF6** - Modificar una solicitud de la Empresa.
- **RF7** - Imprimir o exportar a formato PDF los datos de la Solicitud.

CU. Gestionar Solicitud de Renovación del Permiso de Utilización de Sustancias Químicas. (32)

- **RF8** - Incluir una Solicitud de la Empresa.
- **RF9** - Asociar un Ente Externo a la solicitud.
- **RF10** - Asociar vehículos a la solicitud.
- **RF11** - Ver una solicitud de la Empresa.
- **RF12** - Modificar una solicitud de la Empresa.
- **RF13** - Imprimir o exportar a formato PDF los datos de la Solicitud.

CU. Gestionar Solicitud de Extensión. (33)

- **RF14** - Incluir una Solicitud de la Empresa.
- **RF15** - Asociar un Ente Externo a la solicitud.
- **RF16** - Asociar vehículos a la solicitud.
- **RF17** - Ver una solicitud de la Empresa.
- **RF18** - Modificar una solicitud de la Empresa.
- **RF19** - Imprimir o exportar a formato PDF los datos de la Solicitud.

CU. Gestionar Solicitud de Inclusión. (34)

- **RF20** - Incluir una Solicitud de la Empresa.
- **RF21** - Asociar un Ente Externo a la solicitud.
- **RF22** - Asociar vehículos a la solicitud.
- **RF23** - Ver una solicitud de la Empresa.
- **RF24** - Modificar una solicitud de la Empresa.
- **RF25** - Imprimir o exportar a formato PDF los datos de la Solicitud.

CU. Gestionar Solicitud de Ampliación de Información. (35)

- **RF26** - Incluir una Solicitud de la Empresa.
- **RF27** - Asociar un Ente Externo a la solicitud.
- **RF28** - Asociar vehículos a la solicitud.
- **RF29** - Ver una solicitud de la Empresa.
- **RF30** - Modificar una solicitud de la Empresa.
- **RF31** - Imprimir o exportar a formato PDF los datos de la Solicitud.

CU. Gestionar Permiso Provisional. (36)

- **RF32** - Incluir un Permiso Provisional.
- **RF33** - Modificar un Permiso Provisional.
- **RF34** - Ver un Permiso Provisional.
- **RF35** - Imprimir o exportar a formato PDF los datos del Permiso Provisional.
- **RF36** - Adjuntar la fotocopia de la Cédula del solicitante, de la cotización el proveedor y de la solicitud de permiso.

CU. Gestionar Solicitud de Inspección. (37)

- **RF37** - Incluir un Permiso Provisional.
- **RF38** - Modificar un Permiso Provisional.
- **RF39** - Ver un Permiso Provisional.
- **RF40** - Imprimir o exportar a formato PDF los datos del Permiso Provisional.

CU. Gestionar Carpeta de la Empresa. (38)

- **RF41** - Modificar datos de la Carpeta de la Empresa.
- **RF42** - Relacionar proveedores a la Empresa.
- **RF43** - Asociar entes externos a la Empresa.
- **RF44** - Asociar personas como choferes de la Empresa.
- **RF45** - Incluir Permiso de Bomberos en la Carpeta de la Empresa.
- **RF46** - Incluir diligencias de la Empresa.
- **RF47** - Consultar las diligencias de la Empresa.

- **RF48** - Crear Permiso de Utilización de Sustancias Químicas.
- **RF49** - Imprimir o exportar a formato PDF los datos de la Carpeta de la Empresa.

CU. Relacionar Proveedores a la Empresa. (39)

- **RF50** - Consultar un listado de Empresa registradas.
- **RF51** - Asociar una o varias Empresas como proveedores de la Empresa contenida en la Carpeta de la Empresa.
- **RF52** - Especificar para cada proveedor las sustancias químicas que provee.

CU. Gestionar Resumen de Inspección. (40)

- **RF53** - Incluir un Resumen de Inspección.
- **RF54** - Ver un Resumen de Inspección.
- **RF55** - Modificar un Resumen de Inspección.
- **RF56** - Imprimir o exportar a formato PDF un Resumen de Inspección.

CU. Gestionar Permiso de Utilización de Sustancias Químicas. (41)

- **RF57** - Crear un Permiso de Utilización de Sustancias Químicas.
- **RF58** - Ver un Permiso de Utilización de Sustancias Químicas.
- **RF59** - Modificar un Permiso de Utilización de Sustancias Químicas.
- **RF60** - Imprimir o exportar a formato PDF los datos un Permiso de Utilización de Sustancias Químicas.

CU. Gestionar Permiso de Bombero. (42)

- **RF61** - Incluir un Permiso de Bombero.
- **RF62** - Modificar un Permiso de Bombero.
- **RF63** - Ver un Permiso de Bombero.
- **RF64** - Imprimir o exportar a formato PDF los datos del Permiso de Bombero.

CU. Gestionar Experticia Química (43)

- **RF65** - Incluir un Informe de Experticia Química.
- **RF66** - Ver el Informe de Experticia Química.
- **RF67** - Modificar el Informe de Experticia Química.
- **RF68** - Imprimir o exportar a formato PDF el Informe de Experticia Química.

CU. Gestionar Certificado de Inclusión o Extensión. (44)

- **RF69** - Incluir un Certificado de Inclusión.
- **RF70** - Incluir un Certificado de Extensión.
- **RF71** - Ver un Certificado de Inclusión.
- **RF72** - Ver un Certificado de Extensión.
- **RF73** - Modificar un Certificado de Inclusión.
- **RF74** - Modificar un Certificado de Extensión.
- **RF75** - Imprimir o exportar a formato PDF los datos un Certificado de Inclusión o un Certificado de Extensión.

CU. Consultar Documentos de la Empresa. (45)

- **RF76** - Consultar los documentos que se asocian a una Carpeta de la Empresa registrada en el Sistema.
- **RF77** - Mostrar los datos de los elementos seleccionados.
- **RF78** - Imprimir el listado de coincidencias.

CU. Consultar Experticia Química. (46)

- **RF79** - Consultar las Experticias Químicas registradas en el Sistema.
- **RF80** - Mostrar los datos de una Experticia Química seleccionada.
- **RF81** - Imprimir el listado de coincidencias.

CU. Consultar Resumen de Inspección. (47)

- **RF82** - Consultar los Resúmenes de Inspección registradas en el Sistema.
- **RF83** - Mostrar los datos de un Resumen de Inspección seleccionado.

- **RF84** - Imprimir el listado de coincidencias.

CU. Consultar Permiso Provisional. (48)

- **RF85** - Consultar los Permisos Provisionales registrados en el Sistema.
- **RF86** - Mostrar los datos de un Permiso Provisional seleccionado.
- **RF87** - Imprimir el listado de coincidencias.

CU. Consultar Carpetas de las Empresas. (49)

- **RF88** - Consultar las Carpetas de las Empresas registradas en el Sistema.
- **RF89** - Mostrar los datos de una Carpeta de las Empresa seleccionada.
- **RF90** - Imprimir el listado de coincidencias.

CU. Consultar Empresas con Permisos Vencidos. (50)

- **RF91** - Consultar las Carpetas de las Empresas registradas en el Sistema y que posean los permisos vencidos.
- **RF92** - Mostrar los datos de una Carpeta de las Empresa seleccionada.
- **RF93** - Imprimir el listado de coincidencias.

2.2.3.2 REQUISITOS NO FUNCIONALES

Funcionalidad (51)

RNF 1. Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

RNF 2. El sistema permitirá el uso de reportes para presentar información al usuario.

Usabilidad (51)

RNF 3. Los campos de texto tendrán un tamaño estándar de acuerdo con el espacio con que se cuente en el área de la página.

RNF 4. Se evitará el uso de los Botones de Opción en la medida de lo posible, potenciando la utilización de cuadros de selección, a fin de economizar espacio de trabajo.

RNF 5. No se utilizarán textos extensos para las etiquetas de la interfaz de usuario.

Interfaz de usuario (51)

RNF 6. El sistema brindará una interfaz amigable para sus usuarios. El nivel de funcionamiento del sistema deberá corresponder el nivel medio de conocimiento informático de los usuarios.

RNF 7. El sistema proporcionará claridad y buena organización de la información, permitiendo la interpretación correcta e inequívoca de esta.

RNF 8. El sistema aplicará normas de diseño que permitan la distinción visual entre los elementos de las tablas de resultados, a través del uso de colores.

RNF 9. Todos los textos y mensajes en pantalla aparecerán en idioma español. Los errores serán visibles al usuario e incluirán sugerencias de las posibles soluciones.

RNF 10. El sistema presentará los términos capitalizados, es decir, tendrán su primera letra en mayúsculas.

RNF 11. Los textos asociados a los botones de opción deberán estar redactados claramente.

RNF 12. Ante la ocurrencia de un error, el sistema señalará los campos que generan el mismo, ya sea porque contienen información incompleta, o porque se encuentran vacíos.

Seguridad (51)

RNF 15. El sistema manejará la seguridad de acceso y administración de usuarios: otorgamiento de privilegios y roles, asignación de perfiles.

RNF 16. El sistema implementará el uso de campos obligatorios y validaciones para garantizar la integridad de la información que se introduce por el usuario.

Restricciones de diseño (51)

RNF 17. El sistema será una aplicación web centralizada.

RNF 18. El sistema se implementará usando la plataforma JAVA.

RNF 19. El sistema estará basado en un estilo arquitectónico en capas.

RNF 20. El sistema usará el Framework de Presentación JSF para manejar la Capa de Presentación.

RNF 21. El sistema usará el Framework Spring para manejar la Capa de Lógica de Negocio, así como para el manejo de transacciones, concurrencia y seguridad.

RNF 22. El sistema usará el Framework de Persistencia Hibernate para manejar la capa de Acceso a Datos.

2.3 DISEÑO

En el diseño se modela el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis. (12)

En concreto, los propósitos del diseño son (12):

- Adquirir una comprensión a profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables sistemas operativos, tecnologías de distribución y concurrencia.
- Crear una entrada apropiada y un punto de partida para las actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales.

2.3.1 DESCRIPCIÓN DE LA ARQUITECTURA DEL SISTEMA

La arquitectura de software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. (52)

La arquitectura de software es muy importante sobre todo en el desarrollo de software grande y complejo. Esto posibilita entre otras cosas que los desarrolladores progresen hacia una visión común del sistema. Además contribuye a comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema. (52)

La definición de arquitectura no es sencilla, dado que no existe consenso absoluto sobre este concepto a nivel mundial. Para dar basamento conceptual a las actividades concebidas dentro del marco de la actividad arquitectónica del proyecto SIIPOL, el equipo de arquitectura del proyecto se basó en el concepto siguiente:

"Arquitectura es la organización estructural de un sistema, representada por sus componentes, las relaciones entre los mismos, el ambiente y los principios que gobiernan el diseño y su evolución. La AS[12] *es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema".* (19)

El módulo Sustancias Químicas como parte de SIIPOL utiliza e implementa patrones de diseño heredados de los frameworks utilizados, de los cuales se mencionarán a continuación los más usados:

- **Patrón Experto:** asignar una responsabilidad al experto en información -la clase que tiene la información necesaria para realizar la responsabilidad-. (53) Este patrón es utilizado para la modelación de las entidades persistentes que formarán parte de la solución, donde cada responsabilidad es asignada a la clase u objeto que tiene la información necesaria para realizarla o sea la experta en dicha información.
- **Patrón Bajo Acoplamiento:** el acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de demasiados otros elementos. (53) La utilización de este patrón se evidencia en la comunicación a través de las fachadas de cada subsistema con el que se relaciona el módulo.
- **Patrón Alta Cohesión:** más que un diseño directamente implementable en código, se trata de un principio que nos guiará en el diseño, es un objetivo subyacente a tener en cuenta continuamente. (…) Se puede medir el nivel de cohesión en una clase cuanto más enfocado sea su comportamiento. Cada elemento del diseño debe realizar una labor única dentro del sistema, que no se desempeñe en ninguno de los demás elementos del sistema. (53)
- **Patrón Controlador:** aumenta el potencial de reutilización, y asegura que la lógica de la aplicación no se maneja en la capa de interfaz. Un Controlador es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema y define el método para la operación del sistema. (53) Este patrón es implementado en los beans manejados.
- **Patrón Facade:** Una Fachada es un objeto "front-end" que es el único punto de entrada para los servicios de un subsistema; la implementación y otros componentes del subsistema son privados y

[12] Arquitectura de Software.

no pueden verlos los componentes externos. La Fachada proporciona Variaciones Protegidas frente a los cambios en las implementaciones de un subsistema. (53) El módulo Sustancias Químicas utiliza este patrón en la capa de negocio a través de una fachada para cada submódulo, las cuales le proporcionan independencia y portabilidad al mismo.

- **DAO:** concentra la lógica especial de persistencia de las entidades del dominio de la aplicación. Los DAO contienen todas las sentencias de interacción de la aplicación con la base de datos, así como su transformación para la presentación. (19) Este se utiliza en la capa de acceso a datos, con el fin de provocar el menor cambio posible en las clases de la solución, cuando se realice un cambio en la fuente de los datos.
- **Domain Model:** consiste básicamente en usar las técnicas de diseño orientado a objetos para modelar mediante clases el dominio del problema. (…) El Modelo de Dominio es una vía para enfrentar con éxito negocios complejos y constituye el núcleo de la capa de negocio y de todo el sistema. (19) Este patrón se evidencia en las clases persistentes que tiene el módulo, las cuales encapsulan el dominio del problema.
- **Composite view:** crea vistas compuestas de varias sub-vistas de forma modular, flexible y extensible para construir vistas de páginas JSP. (54) Este patrón se evidencia en las interfaces de usuario dentro de las cuales existen interfaces complejas que son divididas en varias vistas, las cuales son incluidas cuando es necesario usarlas.

La aplicación define varios roles para familias específicas de elementos. Cada una de estas familias o roles, se encarga de tareas particulares, se encuentra en lugares específicos y se nombra de una forma específica. Algunas de estas clases serían las siguientes (22):

Tipo de Clase	Función
Beans Manejados	Manejan la lógica por detrás de las páginas de presentación, o sea, mapean los valores de los formularios JSF a sus propiedades. (19)
Convertidores	Un convertidor es un elemento cuya función consiste en convertir la entrada de datos de la interfaz a formas más legibles para la aplicación, y viceversa.
Validadores	El objetivo de un validador es asegurarse de que la información provista por el usuario es válida según las reglas de negocio reflejadas en el sistema.

Fachadas	Las fachadas son los elementos encargados de proveer el punto básico de aplicación de aspectos, transacciones de la aplicación. Además sirven para separar la lógica de cada módulo de las llamadas de los demás.
Daos	Encargados de manejar la persistencia de una entidad cualquiera, tanto para leer como para escribir en la BD.
Entidades	Las entidades son el mecanismo básico de movimiento e información dentro de la aplicación. Las entidades son leídas de la BD por los Daos y luego se mueven por todas las capas de la aplicación hasta la presentación.
Nomencladores	Esta es una clase particular del sistema, que representa el conjunto de valores propios de los conceptos asociados a las entidades del sistema. Es especial debido a que unifica todos los conceptos que pueden modelarse como clasificadores. (19)
Útiles	Estas clases son colecciones de métodos que encapsulan lógica útil para varios lugares de la aplicación. Funcionan como librerías de funcionalidades en vez de librerías de componentes.

El SIIPOL tiene una arquitectura modular, porque cada subsistema es independiente en cierto grado del resto de la aplicación, pero monolítica debido a que todos son dependientes tanto de un conjunto central de configuraciones, componentes y protocolos definidos previamente como de los demás módulos de la aplicación, siendo entonces imposible realmente separar un módulo de la aplicación. (22)

Se definió una arquitectura en paquetes para la organización de las clases dentro de los módulos y submódulos de la aplicación como se muestra en la figura 9. Donde la estructura de paquetes queda especificada de la siguiente manera (19):

Paquete	Función
comun	Este paquete agrupa todas las clases o funcionalidades comunes para todo el módulo.
config	Este paquete contendrá toda la configuración necesaria para realizar las operaciones del módulo.

web	Este paquete estará constituido por clases pertenecientes a la capa de presentación. Puede contener tantos subpaquetes como sean necesarios, siendo los más comunes **bean** (contiene los beans manejados).
facade	Este paquete contendrá todas las clases fachada del módulo, estas clases constituirán los puntos de acceso al módulo por parte de subsistemas y módulos externos. Este paquete contendrá a su vez un paquete **impl** donde estarán todas las implementaciones de las interfaces que se definan.
dao	Este paquete contendrá todas las clases DAO del módulo, las cuales tendrán la misión de encapsular todo el código necesario para manejar las clases persistentes. Este paquete contendrá a su vez un paquete **impl** donde estarán todas las implementaciones de las interfaces DAO que se definan.
util	Paquete que contiene cualquier clase útil necesaria para el procesamiento de las peticiones del subsistema o módulo en la capa de Aplicación.

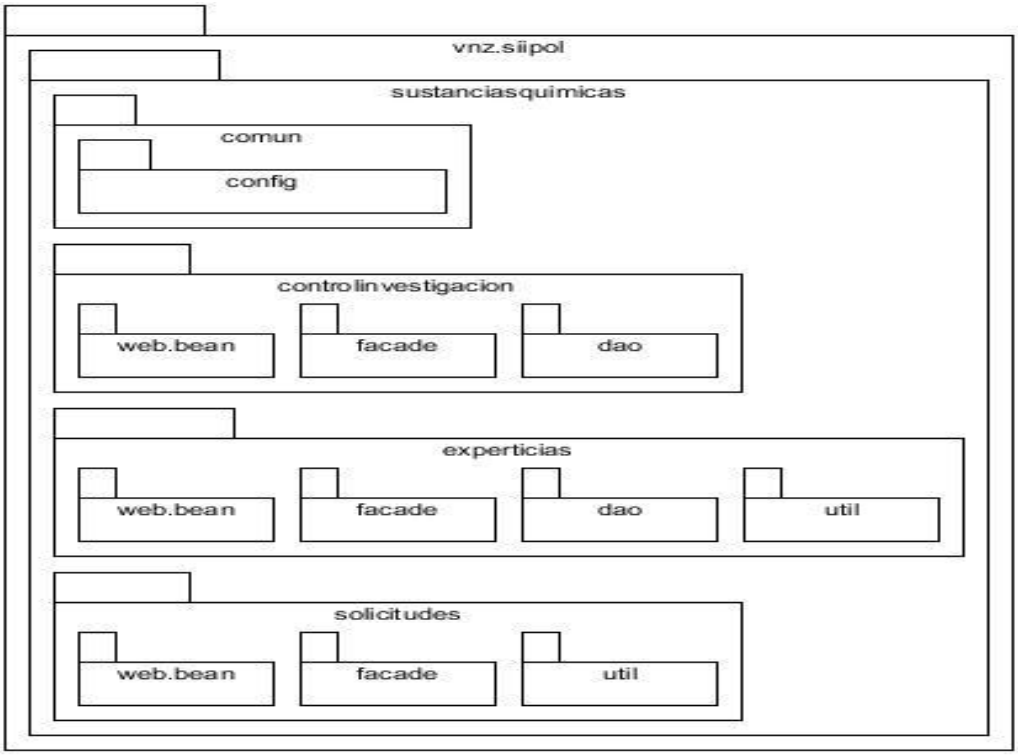


Figura. 9 Estructura por paquetes del módulo Sustancias Químicas de SIIPOL.

Durante el diseño e implementación de la solución se tuvieron en cuenta las principales clases que determinan la estructura de la aplicación de manera general en todos los módulos. Siendo el uso de las mismas obligatorio para lograr una plena integración al SIIPOL. A continuación se brindan las principales características de las mismas.

Clase	Descripción
BaseBean	Constituye la raíz de la jerarquía de beans de respaldo de JSF. Proporciona muchas funcionalidades necesarias para la capa de presentación, como formato y visualización de mensajes, (…) entre otras. (19)
ComunFacade	Constituye la raíz de la jerarquía de las fachadas de todos los módulos. Proporciona muchas funcionalidades necesarias para la capa de presentación, como funciones para consultas básicas entre otras. (19)
EntidadPersistenteBase	Esta clase determina la raíz de la jerarquía de Entidades que se encuentra en el paquete **domain**. Las clases de este estereotipo no suelen realizar ninguna lógica, así que el papel de esta clase se reduce a garantizar la existencia de algunos atributos obligatorios para todas las entidades como el id o el campo activo. (19)
Nomenclador	Esta es una clase particular del sistema, que representa el conjunto de valores propios de los conceptos asociados a las entidades del sistema. Es especial debido a que unifica todos los conceptos que pueden modelarse como clasificadores. (19)
DaoGenerico	Esta clase es la raíz de la jerarquía de DAOs de la aplicación. Contiene numerosas funcionalidades, (…) así como métodos auxiliares de uso obligatorio que determinan la forma que deben tener las Criterias, HQL y SQL de la aplicación. También soportan el paginado y constituye el punto único de acceso a las sesiones de Hibernate y la fuente de datos configurada para la aplicación. (19)

2.3.2 MODELO DE DISEÑO

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve como abstracción de la implementación del sistema y es, de este modo, utilizada como una entrada fundamental de las actividades de implementación. (12)

2.3.2.1 DIAGRAMA DE CLASES DEL DISEÑO

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Los diagramas de clases son diagramas estáticos porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases conocen a qué otras clases o qué clases son parte de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas. (30)

A continuación se muestra el diagrama de clases del diseño del caso de uso, Gestionar Solicitud de Permisos de Utilización de Sustancias Químicas.

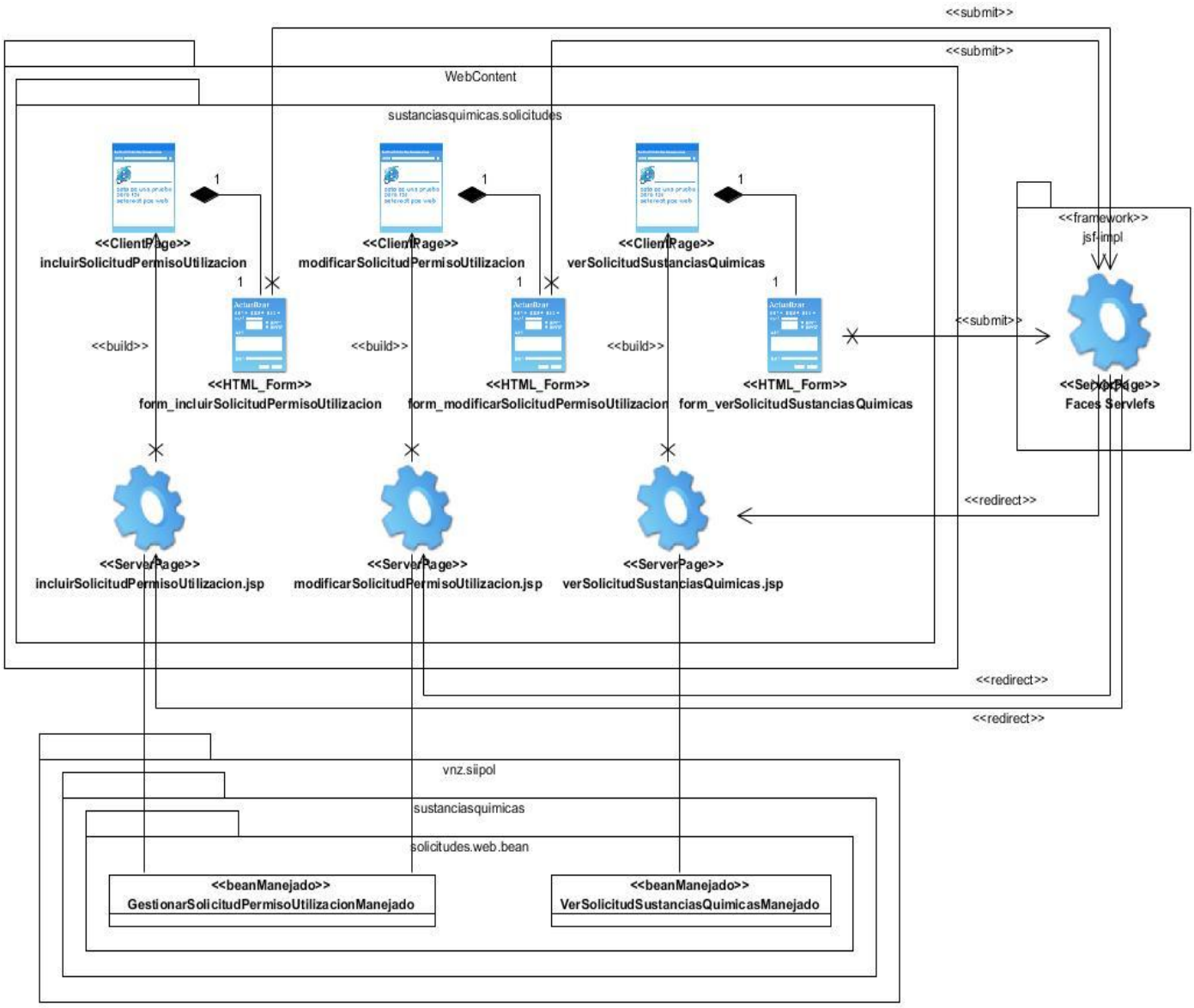


Figura. 10 Diagrama de Clases. CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas – Vista de presentación.

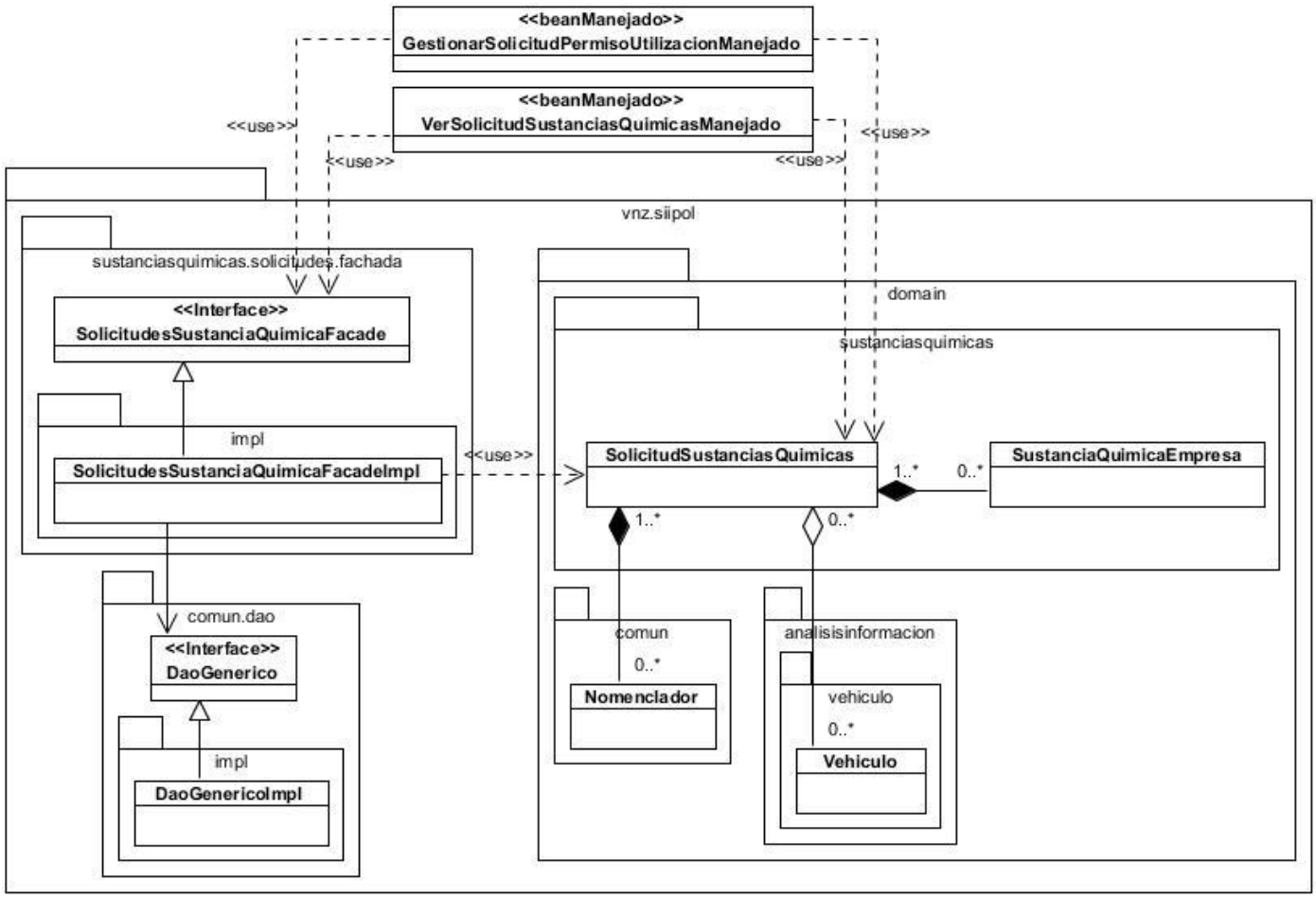


Figura. 11 Diagrama de Clases. CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas - Vista de negocio y acceso a datos.

2.3.2.2 REALIZACIONES DE CASOS DE USO

Una realización de caso de uso - diseño es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de clases de diseño y sus objetos. (…) Una realización de caso de uso-diseño tiene una descripción de flujo de eventos textual, diagramas de clases que muestra sus clases de diseño participantes, y diagramas de interacción que muestran la realización de un flujo o escenario concreto de un caso de uso en términos de interacción entre objetos del diseño. Si fuera necesario, los diagramas pueden mostrar también los subsistemas e interfaces implicados

en la realización de casos de uso (es decir, los subsistemas que contienen las clases participantes del diseño). (12)

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si consideramos el "interior" del sistema, tendremos algún objeto de diseño que recibe el mensaje del actor. Después el objeto de diseño llama a algún otro objeto, y de esta manera los objetos implicados interactúan para realizar y llevar a cabo el caso de uso. En el diseño, es preferible representar esto con diagramas de secuencia ya que nuestro centro de atención principal es el encontrar secuencias de interacciones detalladas y ordenadas en el tiempo. (12)

Para minimizar la complejidad que representaba realizar los diagramas de secuencia mediante la interacción entre objetos, la dirección del proyecto decidió realizar estos diagramas mediante la interacción entre subsistemas, a lo cual denominaron “Diagrama de contrato entre paquetes”, siendo estos artefactos del proyecto.

Para entender mejor en que se basan estos diagramas, se cita textualmente del libro Proceso Unificado de Desarrollo el siguiente fragmento:

“En los diagramas de secuencia, mostramos las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. Cuando decimos que un subsistema “recibe” un mensaje, queremos decir en realidad que es un objeto de una clase del subsistema el que recibe el mensaje. Cuando un subsistema envía un mensaje, realmente es un objeto de una clase del subsistema el que envía el mensaje.” (12)

Seguidamente se presenta el diagrama de secuencia del caso de uso seleccionado para la realización de la presente investigación.

CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas.

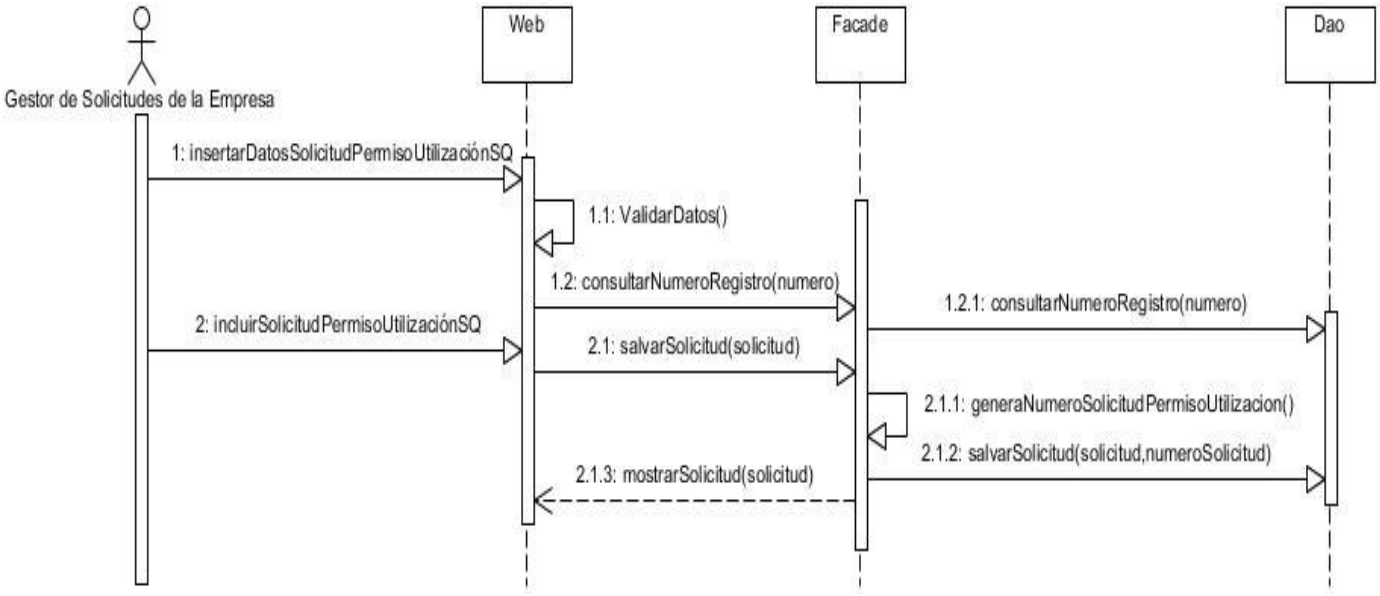


Figura. 12 Diagrama de Contratos entre Paquetes. CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas. Escenario Incluir.

2.3.2.3 CLASES SIGNIFICATIVAS PROPIAS DE LA SOLUCIÓN

A continuación se describen algunas clases de importancia en el diseño de la solución, relacionadas al caso de uso modelado anteriormente.

Nombre: SolicitudSustanciasQuimicas	
Tipo de Clase: Entidad Persistente	
Atributo	**Tipo**
sustanciaQuimicasAsociadas	Set<SustanciaQuimicaEmpresa>
actividadesComerciales	Set<Nomenclador>
vehiculosAsociados	Set<Vehiculo>

Nombre: SustanciaQuimicaEmpresa	
Tipo de Clase: Entidad Persistente	
Atributo	**Tipo**
tipoSustancia	Nomenclador
solicitudSustanciaQuimicas	Documento

cantidad	Float
unidadMedidaCantidad	Nomenclador
proveedoresInternacionales	Set<ProveedorInternacional>
proveedoresNacionales	Set<ProveedorNacionalSustancia>

Nombre: GestionarSolicitudPermisoUtilizacionManejado	
Tipo de Clase: Controladora	
Atributo	**Tipo**
empresa	Empresa
numeroRegistro	String
solicitudPermisoUtilizacionSQ	SolicitudSustanciasQuimicas
sustanciasQuimicasDeLaSolicitud	List<SustanciaQuimicaEmpresa>
tipoSustanciasQuimicasDeLaSolicitud	List<Nomenclador>
sustanciaQuimicaSolicitud	SustanciaQuimicaEmpresa
actividadComercial	Nomenclador
noRegistroValido	boolean
modificar	boolean
representanteLegalInv	boolean
iAsociarEnteExternoManejado	IAsociarEnteExternoManejado
solicitudesSustanciaQuimicaFacade	SolicitudesSustanciaQuimicaFacade
nomencladorFacade	NomencladorFacade
experticiasSustanciasQuimicasFacade	ExperticiasSustanciasQuimicasFacade
administracionSistemaFacade	AdministracionSistemaFacade
Para cada responsabilidad:	
Nombre: incluirSolicitud	
Descripción: Incluye una Solicitud de Permiso de Utilización.	
Nombre: modificarSolicitud	
Descripción: Modifica una solicitud de Permiso de Utilización.	
Nombre: verEmpresa	
Descripción: Permite ver los datos de la Empresa Asociada.	
Nombre: incluirSustanciaQuimica	
Descripción: Incluye una sustancia química a la Solicitud.	
Nombre: eliminarSustanciaQuimica	
Descripción: Permite eliminar una sustancia química de la Solicitud.	
Nombre: incluirActividadComercial	
Descripción: Adiciona una Actividad Comercial a la Solicitud.	
Nombre: eliminarActividadesComerciales	
Descripción: Elimina una Actividades Comerciales de la Solicitud.	
Nombre: vistaPrevia	
Descripción: Permite ver una vista previa de la Solicitud de Permiso de Utilización hasta el momento.	
Nombre: asociarEmpresa	
Descripción: Asocia una Empresa a la Solicitud.	

Nombre: cambiarEmpresa
Descripción: Cambia la Empresa asociada a la Solicitud.
Nombre: validarCambiarNoRegistro
Descripción: Valida que el número de registro tenga el formato correcto y que no se repita.

Nombre: SolicitudesSustanciaQuimicaFacadeImpl	
Tipo de Clase: Controladora	
Atributo	**Tipo**
generadorNumeroSolicitudPermisoUtilizacion	GeneradorNumeroSolicitudPermisoUtilizacionImpl
generadorNumeroSolicitudRenovacion	GeneradorNumeroSolicitudRenovacionImpl
generadorNumeroSolicitudAmpliacion	GeneradorNumeroSolicitudAmpliacionImpl
generadorNumeroSolicitudExtension	GeneradorNumeroSolicitudExtensionImpl
generadorNumeroPermisoProvisional	GeneradorNumeroPermisoProvisionalImpl
generadorNumeroSolicitudInspeccion	GeneradorNumeroSolicitudInspeccionImpl
Para cada responsabilidad:	
Nombre: salvarSolicitudSustanciaQuimica	
Descripción: Salva la Solicitud.	
Nombre: obtenerSolicitudSustanciaQuimicaPorId	
Descripción: Dado un id obtiene la Solicitud.	
Nombre: obtenerPermisosProvisionalesAsociadosPersona	
Descripción: Dada una persona obtiene la lista de Permisos Provisionales asociados a la misma.	
Nombre: obtenerPermisosProvisionalesAsociadosEmpresa	
Descripción: Dada una empresa obtiene la lista de Permisos Provisionales asociados a la misma.	
Nombre: obtenerPermisoProvisionalPorId	
Descripción: Dado un id obtiene el Permiso Provisional.	
Nombre: salvarPermisoProvisional	
Descripción: Salva el Permiso Provisional.	
Nombre: salvarSolicitudInspeccion	
Descripción: Salva la Solicitud de Inspección.	

2.3.3 MODELO DE DATOS

"Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base)". (55)

2.3.3.1 DIAGRAMA DE CLASES PERSISTENTES

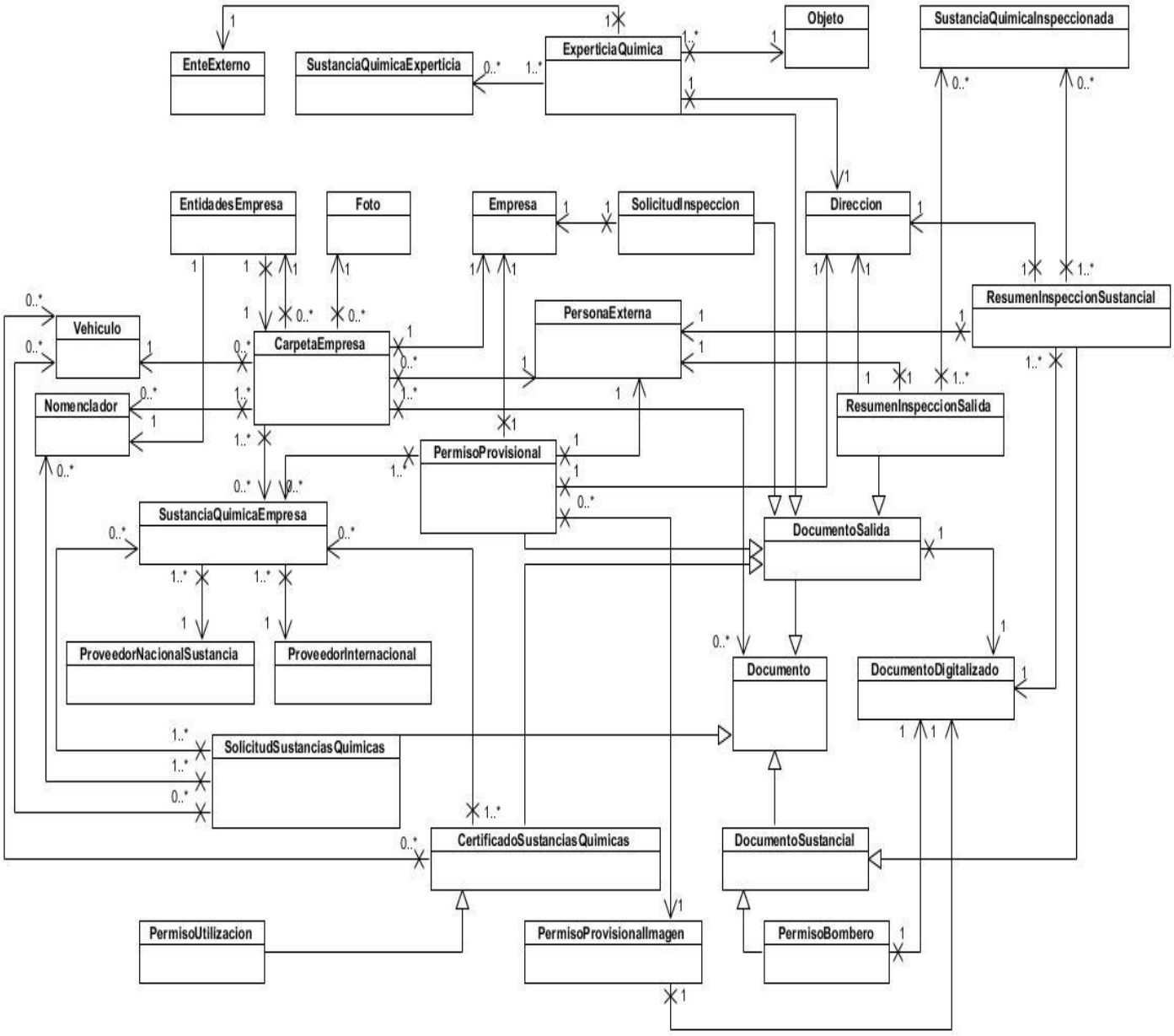


Figura. 13 Diagrama de clases persistentes.

2.3.3.2 DIAGRAMAS DE TABLAS DEL MODELO RELACIONAL

El diagrama que se presenta a continuación, muestra el diagrama de tablas del modelo relacional de manera reducida; sólo se representan las tablas y sus relaciones. Para más detalle de las mismas ver Anexo. 1.

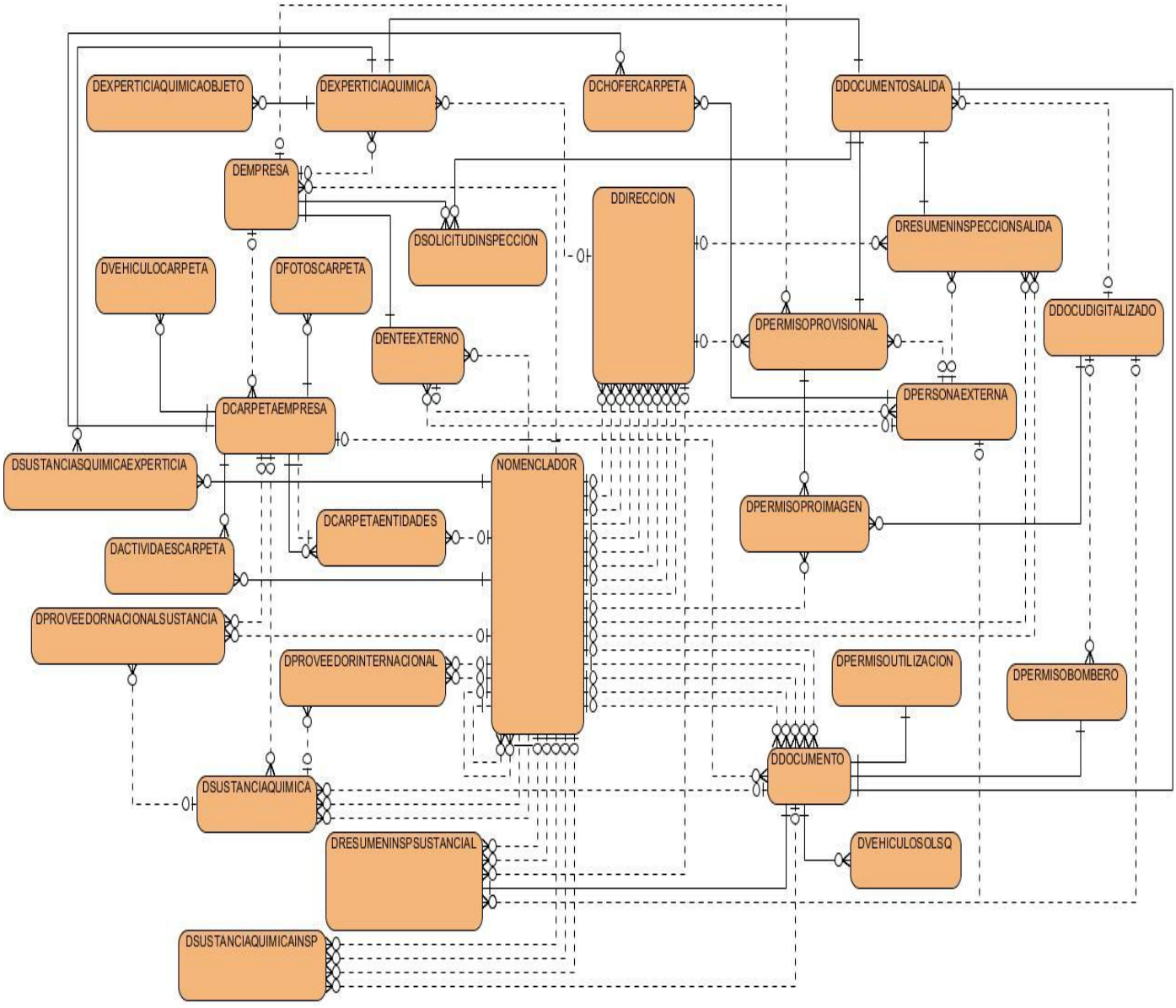


Figura. 14 Diagramas de tablas del modelo relacional.

2.4 IMPLEMENTACIÓN

En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir ficheros de código fuente, scripts, ficheros de códigos binario, ejecutables, (…). (12)

Los propósitos de la implementación son (12):

- Planificar las integraciones de sistema necesarias en cada iteración. Se sigue para ello un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue. Esto se basa fundamentalmente en las clases activas encontradas durante el diseño.
- Implementar las clases y subsistemas encontrados durante el diseño. En particular, las clases se implementan como componentes de fichero que contienen código fuente.
- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones de sistema.

2.4.1 MODELO DE IMPLEMENTACIÓN

El modelo de implementación describe cómo los elementos del modelo de diseño, cómo las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (12)

2.4.1.1 DIAGRAMAS DE SUBSISTEMAS DE IMPLEMENTACIÓN

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas (recursivamente). Además un subsistema puede implementar -y así proporcionar - las interfaces que representan la funcionalidad que exportan en forma de operaciones. (12)

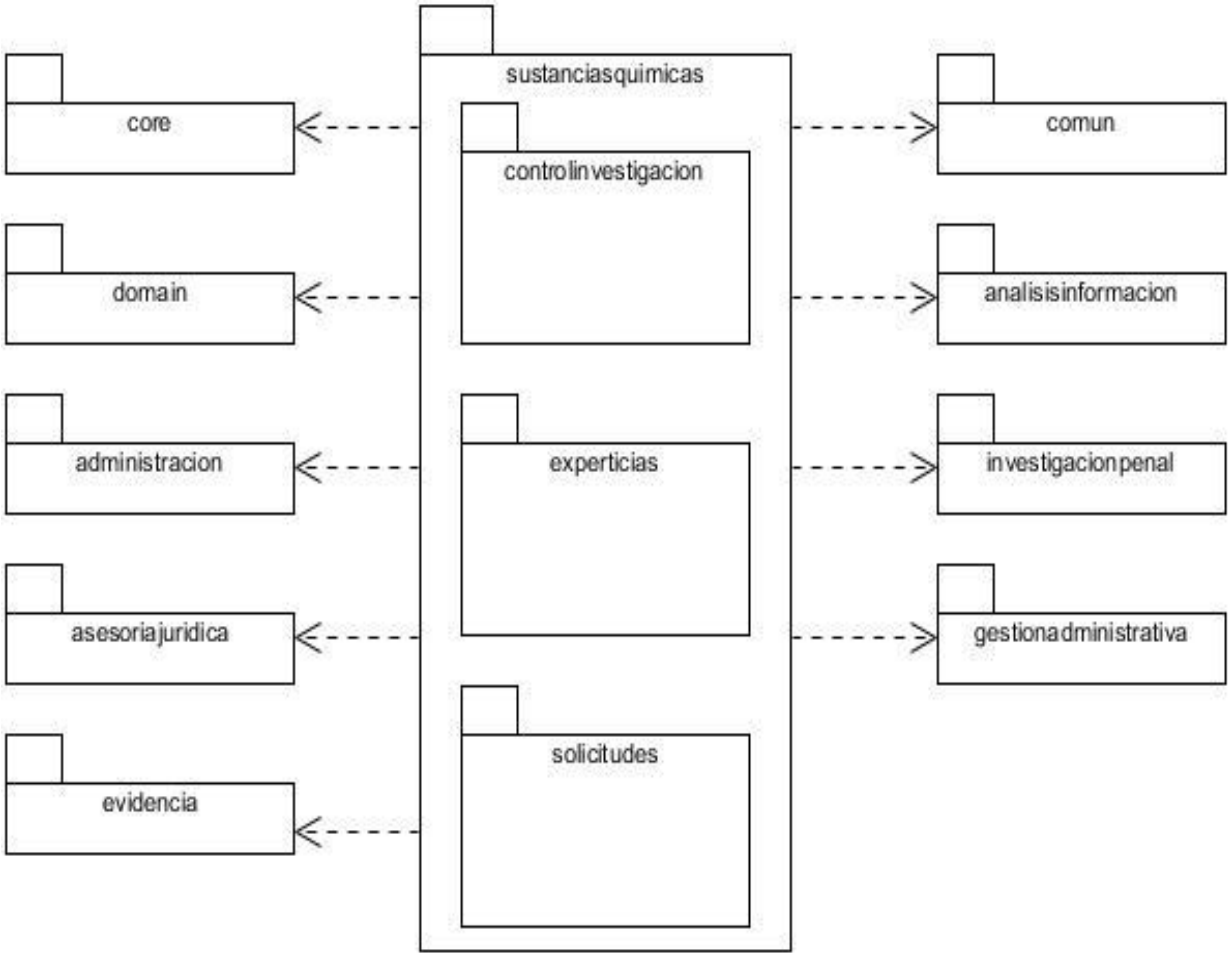


Figura. 15 Implementación. Diagramas de Subsistemas de Implementación.

2.4.1.2 DIAGRAMA DE COMPONENTES

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. (12)

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones.

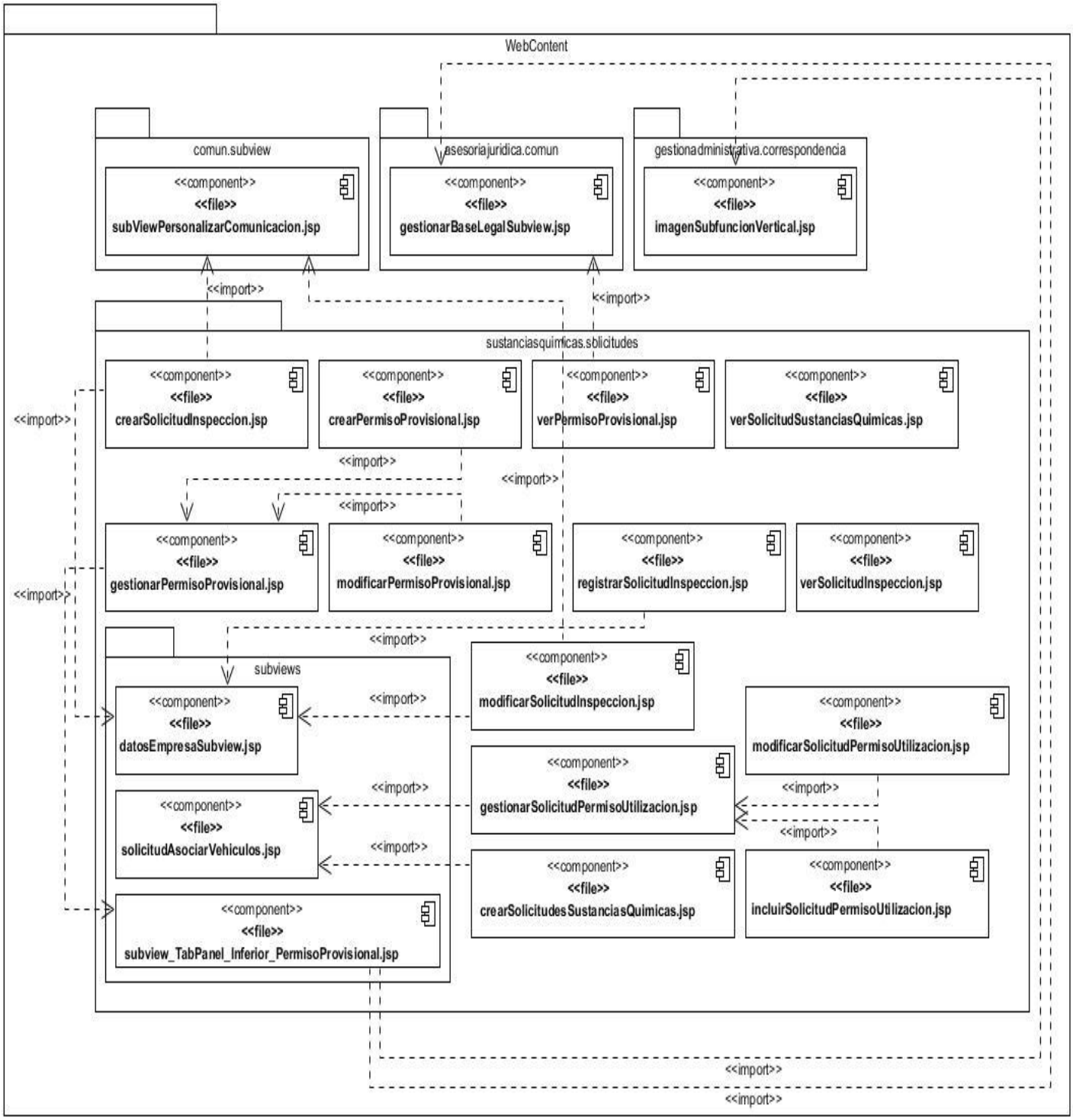


Figura. 16 Diagrama de componentes. Submódulo Solicitudes. Vista de la capa de presentación. Páginas JSP.

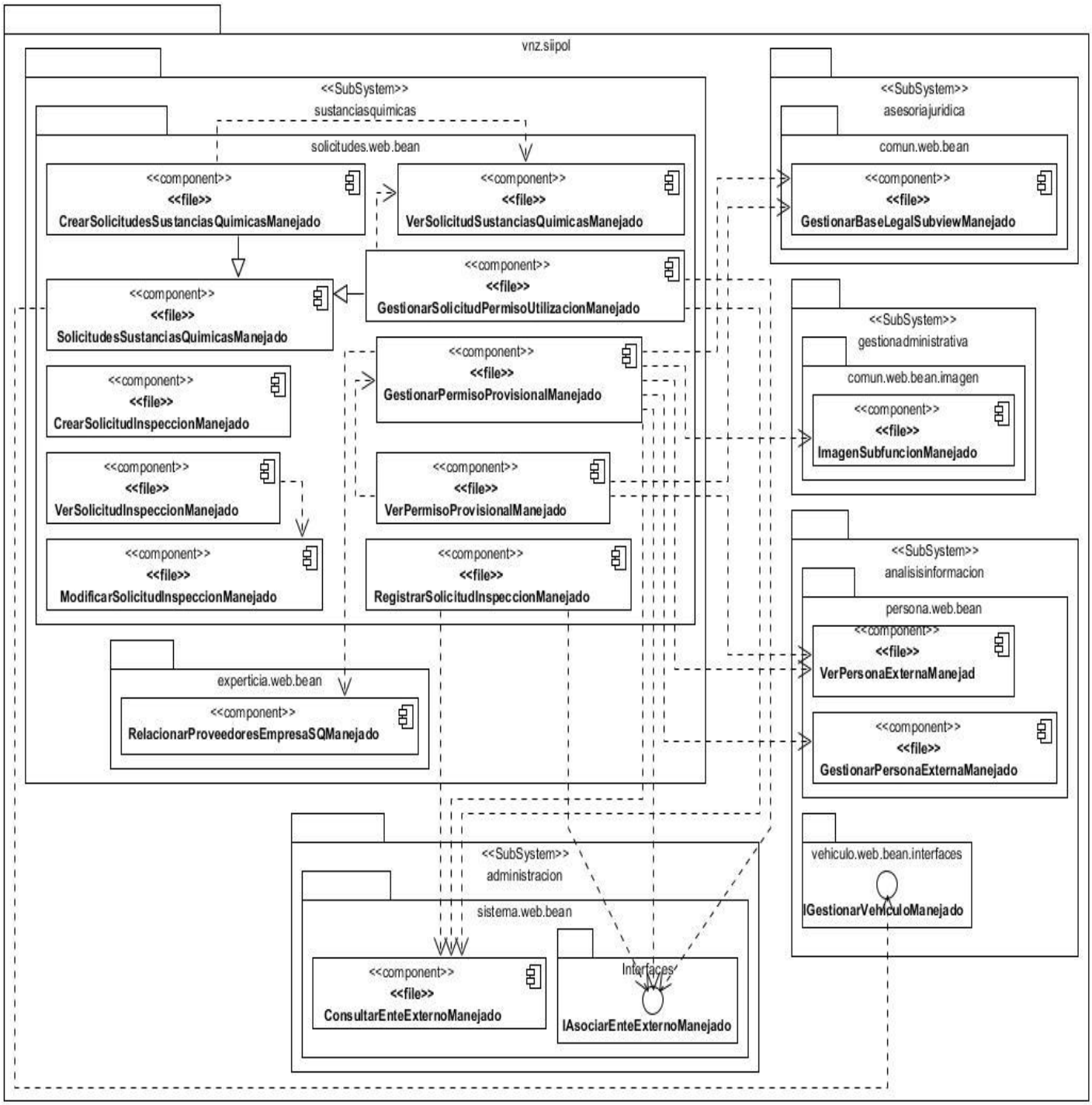


Figura. 17 Diagrama de componentes. Submódulo Solicitudes. Vista de la capa de presentación. Beans Manejados.

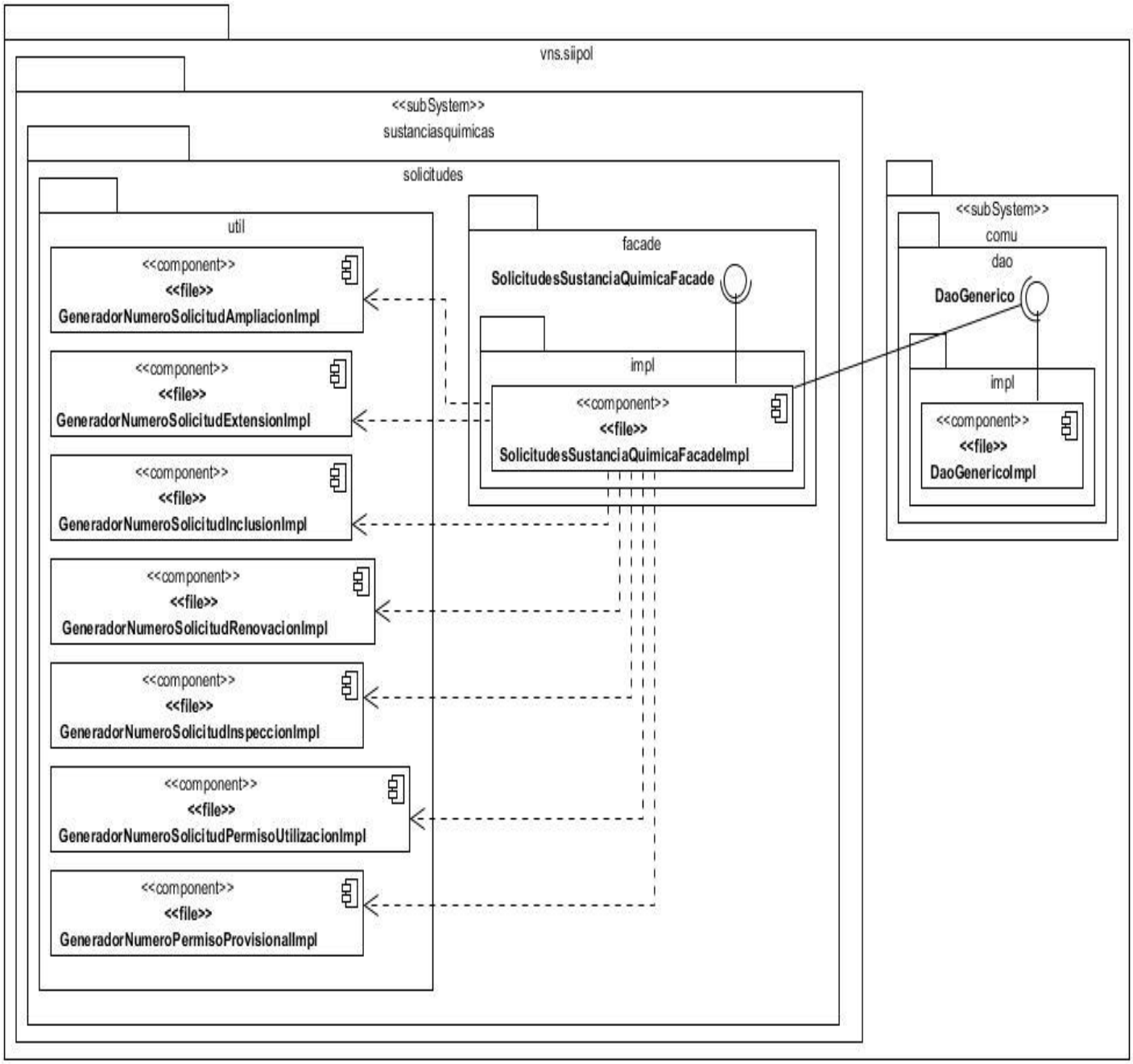


Figura. 18 Diagrama de componentes. Submódulo Solicitudes. Vista de negocio y acceso a datos.

2.5 CONCLUSIONES

Con el término de este capítulo, se le dio cumplimiento a los objetivos planteados referentes al diseño e implementación de la solución propuesta. Inicialmente se realizó un análisis de los artefactos generados en la Ingeniería de Requerimientos, con el objetivo de fortalecer el entendimiento del sistema y preparar las condiciones para la creación con éxito del diseño e implementación de la solución. También, se hizo un estudio de la arquitectura establecida por la dirección del proyecto, haciendo énfasis en los principales patrones de diseño implementados, para un mayor entendimiento de los elementos de la solución. Además fueron expuestos los diagramas generados durante los flujos de trabajos: diseño e implementación, conformando así el modelo de diseño y el modelo de implementación. Modelos que brindan diferentes perspectivas del módulo a desarrollar y son de gran importancia a la hora de la creación del producto en cuestión.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN

3.1 INTRODUCCIÓN

El presente capítulo se enfoca en la verificación de los requisitos funcionales y no funcionales asociados a la solución propuesta, siguiendo la metodología seleccionada para el desarrollo de la solución se exponen los resultados obtenidos por diferentes tipos de pruebas realizadas al módulo Sustancias Químicas. El desarrollo de las pruebas se basa en dos métodos de pruebas (Caja Blanca y Caja Negra), además está enfocado a 3 niveles (pruebas de unidad, pruebas de integración y pruebas de sitema), los cuales, son fundamentales y aportan resultados significativos para la validación del módulo.

3.2 ESTRATEGIA DE PRUEBA

"*Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuando se deben planificar y realizar esos pasos, y cuanto esfuerzo, tiempo y recursos van a requerir. Por tanto cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.*" (52)

La estrategia de prueba elaborada para llevar a cabo la verificación de la solución está basada en el Modelo en V (ver figura 19). Inicialmente se analizan los requisitos del software, dicho análisis permite planificar los métodos y niveles de prueba necesarios para realizar las pruebas de una forma eficiente. Los niveles de prueba se dividen en dos grupos, Alto nivel y Bajo nivel. Dentro de este último se encuentran las pruebas de unidad, las cuales se centran en verificar cada unidad tal como está implementada en código fuente. Luego de haber concluido las pruebas unitarias se realizan las de integración cuyo objetivo principal es probar el diseño y la construcción de la arquitectura. Con la culminación de las pruebas de integración se avanza hacia el Alto nivel, en el cual entra a jugar su papel las pruebas del sistema, las cuales se encargan de probar la solución como un todo. Finalmente se procede a las pruebas de aceptación, en las que el cliente es el encargado validar todos los requisitos.

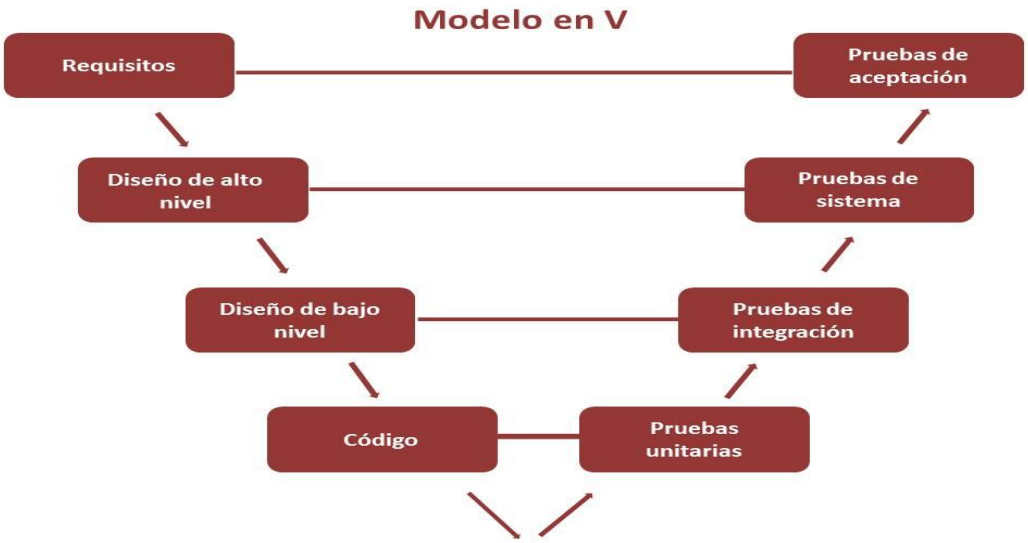


Figura. 19 Pruebas. Modelo en V.

3.3 MÉTODOS DE PRUEBA

Los métodos de prueba definen qué estrategia seguir en cuanto a la verificación y validación del sistema ya que están diseñados para descubrir fallos. Los métodos más significativos son las pruebas de caja blanca y las pruebas de caja negra; las primeras, pruebas orientadas a la estructura y las segundas al comportamiento del software.

3.3.1 PRUEBA DE CAJA BLANCA

"La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba." (52)

Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas. Dichas pruebas están dirigidas a las funciones internas del módulo; permiten ejercitar y validar los caminos de cada módulo, las condiciones lógicas, los bucles y sus límites, así como también las estructuras de datos. Las pruebas de caja blanca son

ejecutadas por los miembros del equipo de desarrollo, donde los propios programadores buscan errores cometidos y verifican la obtencion del resultado esperado.

3.3.2 PRUEBA DE CAJA NEGRA

"Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa." (52)

La prueba de caja negra intenta encontrar errores de las siguientes categorías (52):

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

3.4 NIVELES DE PRUEBAS

Un nivel de prueba permite especificar el alcance de la prueba de software que se realiza, se pueden identificar principalmente dos niveles (Bajo Nivel y Alto Nivel).

3.4.1 BAJO NIVEL

En este nivel se encuentran aquellas pruebas que se realizan a componentes individuales de un programa. Dentro de este nivel se encuentran las Pruebas de Unidad y las Pruebas de Integración.

3.4.1.1 PRUEBAS DE UNIDAD

"*Las pruebas unitarias son un procedimiento para validar que una porción del código del sistema funciona apropiadamente de manera aislada. Rob Johnson define las pruebas unitarias como el nivel más fino de granularidad en las pruebas, que verifican una simple unidad de funcionalidad y deben probar que cada método de una clase satisface su contrato documentado."* (56)

Para un mejor entendimiento de como se realizaron las pruebas de unidad en el presente trabajo se cita textualmente del documento: Descripción De La Arquitectura De Software SIIPOL 5.0.

El uso de las pruebas unitarias automatizadas ha desaparecido totalmente de la base de código. La razón es organizativa y tecnológica al mismo tiempo: La introducción del mecanismo conversacional, unido a la salida masiva de personal calificado y la introducción de personal sin experiencia dentro de un sistema cuya arquitectura ya está consolidada y no cambiará al ritmo visto anteriormente, hace que el tiempo utilizado en el desarrollo de las pruebas ya no reporte el mismo beneficio que antes. La necesidad institucional de probar todos los casos de uso antes de cada liberación del producto, y las frecuentes iteraciones de calidad interna sustituyen en cierta forma el papel jugado por las pruebas de unidad anteriormente, aunque no cumplan con los mismos objetivos ni metodología de trabajo. No obstante, el desarrollador antes de liberar un Caso de Uso realiza sus propias pruebas unitarias y de integración antes de entregarle dicho Caso de Uso al equipo de calidad el cual le tiene que dar el visto bueno antes de su liberación parcial o total. (19)

3.4.1.2 PRUEBAS DE INTEGRACIÓN

"La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño". (52)

Luego de realizadas las pruebas unitarias se llevaron a cabo las pruebas de integración, las cuales prueban el sistema como un todo. Dichas pruebas, realizadas por los desarrolladores del módulo estuvieron guiadas a probar la dependencia entre el módulo Sustancias Químicas y otros módulos del sistema. Tal es el caso de la integración con los módulos:

- Análisis de Información: para asociar vehículos a la solicitud de permiso de utilización en caso de que la empresa realice la actividad comercial Transportista Importadora.
- Administración: para asociar las empresas a las solicitudes de permiso de utilización.
- Asesoría Jurídica: se integra a la hora de incluir la subvista base legal a los certificados y permisos otorgados a las empresas.

3.4.2 ALTO NIVEL

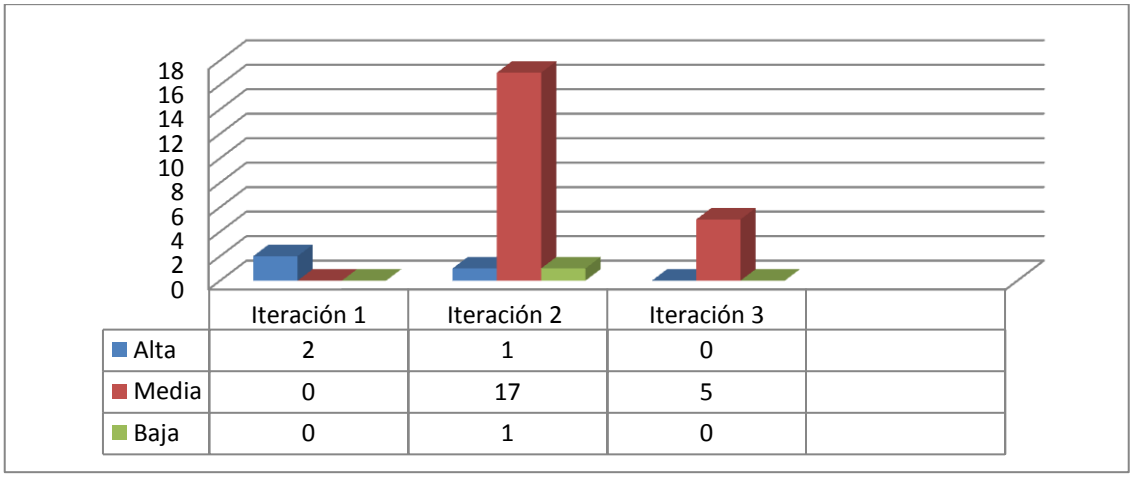
En este nivel las pruebas están orientadas a verificar que las funcionalidades de la solución cumplan con los requisitos del cliente, es decir un producto completo, integrado en un todo.

3.4.2.1 PRUEBAS DEL SISTEMA

La prueba del sistema, realmente, está constituida por una serie de pruebas diferentes, cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. (52)

Las pruebas del sistema realizadas al módulo fueron las pruebas de calidad interna, las pruebas cruzadas y las pruebas de liberación.

Pruebas Cruzadas: Consisten en la realización de pruebas al sistema por los demás equipos de desarrollo del proyecto. Estas estuvieron centradas en encontrar la mayor cantidad de errores en cuanto a validaciones, formato de los campos. Los resultados de estas pruebas se pueden apreciar en la figura 20.

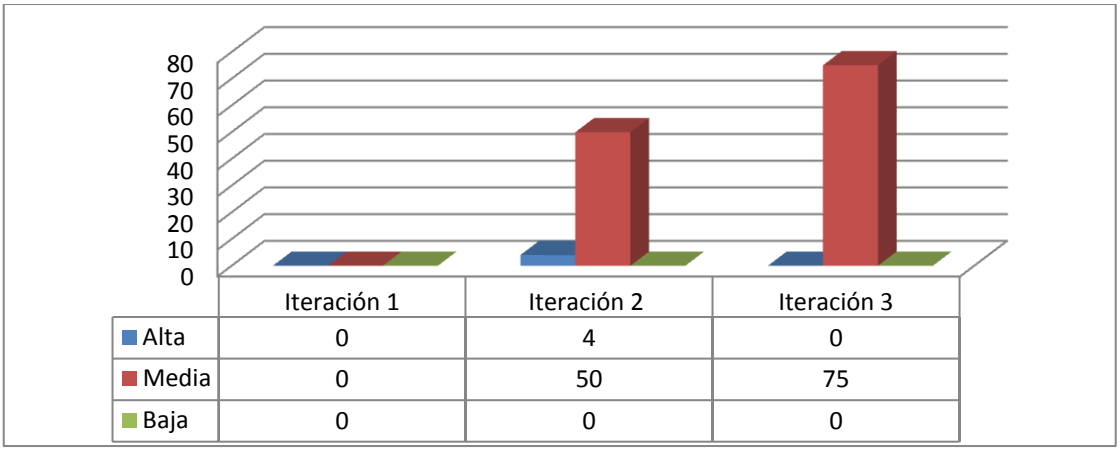


	Iteración 1	Iteración 2	Iteración 3	
Alta	2	1	0	
Media	0	17	5	
Baja	0	1	0	

Figura. 20 Pruebas. Resultados de las pruebas cruzadas.

Pruebas de Calidad Interna: Estas pruebas fueron realizadas por el equipo de calidad interna del proyecto y enmarcadas en el método de caja negra. Estuvieron centradas en el cumplimiento de las

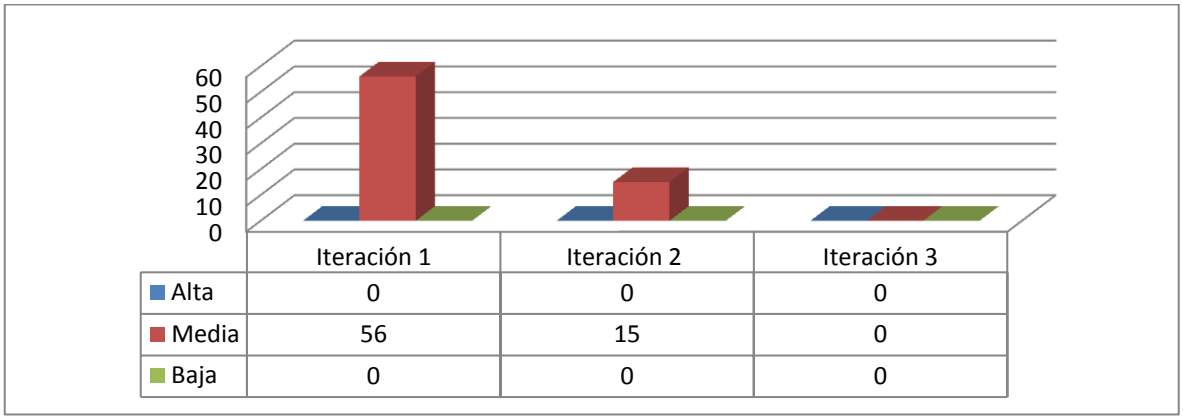
funcionalidades descritas en los requerimientos y casos de uso. Los resultados de las pruebas de calidad interna se muestran en la figura 21. Las (no conformidades)[13] registradas se clasifican en altas (errores en la interpretación de los procesos de la entidad y de funcionalidad), medias (errores de terminología y de diseño de interface) y bajas (errores de redacción y ortografía).



	Iteración 1	Iteración 2	Iteración 3
Alta	0	4	0
Media	0	50	75
Baja	0	0	0

Figura. 21 Pruebas. Resultados de las pruebas de calidad interna.

Pruebas de Liberación: Dichas pruebas son realizadas por un tercero, en este caso Calisoft[14]. A continuación en la figura 22 se presentan los resultados obtenidos en estas pruebas para el módulo Sustancias Químicas.



	Iteración 1	Iteración 2	Iteración 3
Alta	0	0	0
Media	56	15	0
Baja	0	0	0

Figura. 22 Pruebas. Resultados de las pruebas de Calisoft.

[13] De acuerdo a la definición en la norma NC ISO 9000: 2005 (3.1.2), una no conformidad es el incumplimiento de un requisito.
[14] Empresa especializada en la prestación de servicios y soporte técnico en el área de sistemas.

3.5 RESULTADO DE LAS PRUEBAS

Luego de haber concluido las pruebas cruzadas, calidad interna y liberación, mencionadas anteriormente, se puede apreciar en la figura 23 el resumen de los resultados obtenidos en las mismas. Las pruebas cruzadas arrojaron un total de 26 no conformidades, mientras que las de calidad interna y liberación 129 y 71 respectivamente. De acuerdo a los resultados alcanzados, se puede concluir que las pruebas cruzadas no se realizaron con la calidad requerida, no siendo así en las pruebas de calidad interna donde fueron registradas una cantidad significativa de no conformidades, lo que posibilitó que se llegara a las pruebas de liberación con la menor cantidad de errores posibles.

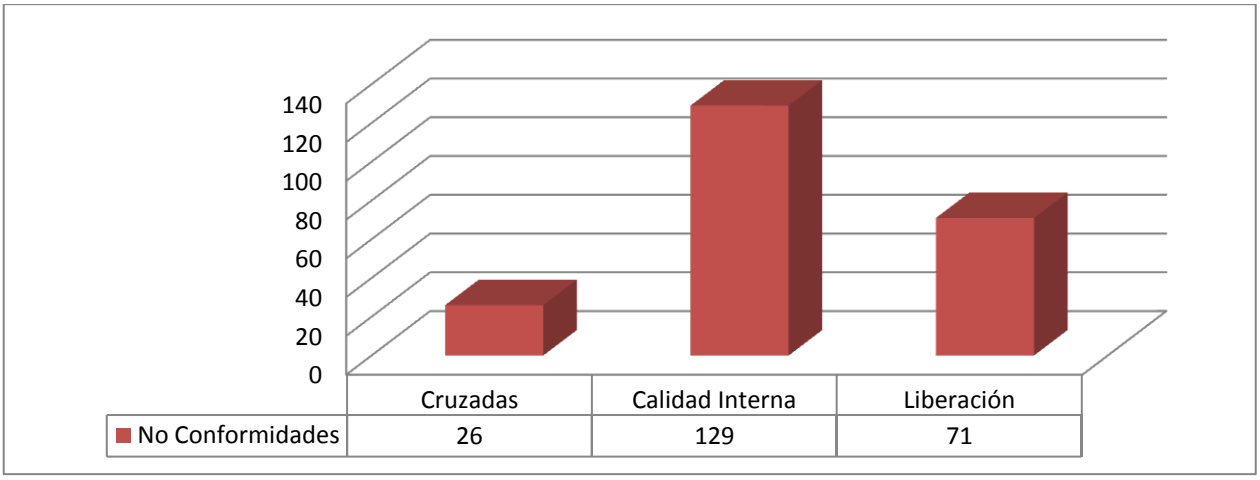


Figura. 23 Resultados de las pruebas.

3.6 CONCLUSIONES

Una actividad indispensable en todo proceso de desarrollo de software, es la prueba, si esta actividad no se llevara a cabo, se tendría un desconocimiento total sobre la calidad del producto. Luego de haber realizado la verificación de la solución, basada en diferentes tipos de pruebas, y haber corregido las No Conformidades encontradas, se puede decir que el producto final cumple con los requisitos especificados para el módulo Sustancias Químicas. En fin, se está en presencia de un producto listo para presentarlo al cliente para su aceptación.

CONCLUSIONES

El presente documento describe el proceso de desarrollo del módulo Sustancias Químicas del SIIPOL, siendo este la respuesta al problema a resolver planteado. Para ello se realizó un estudio del marco teórico en el que se sustenta la investigación, enfatizando en las características que debe poseer un software de gestión de información policial y se analizaron algunos de los ya existentes, lo cual arrojó a la primera conclusión de la investigación: era necesario implementar un nuevo sistema, puesto que ninguno de los existentes cumplía con las necesidades del cliente.

Para materializar la propuesta de solución se generaron una serie de artefactos, basados en la metodología utilizada (RUP) orientados a elaborar el diseño e implementación del subsistema. Dichos artefactos son la prueba del cumplimiento del objetivo general de la presente investigación: diseñar e implementar el módulo Sustancias Químicas del Sistema de Investigación e Información Policial (SIIPOL).

Como elementos concluyentes de la presente investigación se tiene:

- Las herramientas y definiciones arquitectónicas puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido.
- El uso de RUP como metodología de desarrollo, guió todo el proceso de desarrollo y aseguró la calidad del producto según lo esperado.
- La verificación del software fue de gran importancia, pues se comprobó el adecuado funcionamiento de las funcionalidades del módulo, logrando una elevada calidad del producto final.
- Los resultados obtenidos con la implementación del módulo Sustancias Químicas fueron satisfactorios.

RECOMENDACIONES

Ampliar las funcionalidades del módulo según las necesidades del cliente.

Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas.

Utilización de la información generada y la documentación para proyectos futuros similares, siempre cumpliendo con las normas y políticas de confidencialidad requeridos.

BIBLIOGRAFÍA REFERENCIADA

1. Vision of Humanity. *Vision of Humanity.* [Online] [Cited: enero 17, 2011.] http://www.visionofhumanity.org/info-center/global-peace-index-2011/.

2. Cuerpo de Investigaciones Científicas, Penales y Criminalísticas. *Cuerpo de Investigaciones Científicas, Penales y Criminalísticas.* [Online] [Cited: septiembre 14, 2011.] http://www.cicpc.gov.ve/mision-vision.

3. **Cabrera Medina, Andy.** *Ingeniería de Requisitos aplicada al Módulo de Sustancias Químicas del Sistema de Investigación Policial del CICPC.* Ciudad de la Habana : s.n., 2010.

4. **Ruiz, Ramón.** *El Método Científico y sus Etapas.* México : s.n., 2007.

5. REAL ACADEMIA ESPAÑOLA. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [Online] [Cited: marzo 14, 2012.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=informaci%C3%B3n.

6. Information Management. [Online] [Cited: septiembre 13, 2011.] http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion/.

7. **Bartle, Philip.** CEC. *Community Empowerment Collective.* [Online] marzo 13, 2011. [Cited: septiembre 13, 2011.] http://cec.vcn.bc.ca/mpfc/modules/mon-miss.htm.

8. Municipalidad Distrital de Jangas. *Municipalidad Distrital de Jangas.* [Online] [Cited: marzo 15, 2012.] http://www.munijangas.gob.pe/?page_id=3535.

9. Vision Of Humanity. *Vision Of Humanity.* [Online] [Cited: marzo 15, 2012.] http://www.visionofhumanity.org/wp-content/uploads/2011/05/2011-GPI-Results-Report-Final.pdf.

10. **Montoya Castillo, Adonis .** *Análisis, Diseño e Implementación del módulo Transmisiones del Sistema de Investigación e Información Policial.* Ciudad de la Habana : s.n., 2010.

11. ¿Qué es la ingeniería de Software? [Online] 12 16, 2011. [Cited: 1 10, 2012.] www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html.

12. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Educacion. S.A., 2000.

13. **Jacobson, Ivar, Rumbaugh, James and Booch, Grady.** *El lenguaje Unificado de Modelado. Manual de Referencia.* Madrid : Addison Wesley, 2007.

14. ITESCAM. *Instituto Tecnológico Superior de Calkiní en el Estado de Campeche.* [Online] 2012. [Cited: marzo 20, 2012.] www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r19670.DOC.

15. **Visual Paradigm International Limited.** *Visual Paradigm for UML 8.0 Released.* Agosto 16, 2010.

16. Free Download Manager. *Free Download Manager.* [Online] [Cited: septiembre 4, 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

17. Oracle. *Oracle.* [Online] agosto 2010. [Cited: octubre 6, 2011.] http://download.oracle.com/javaee/5/firstcup/doc/.

18. Oracle. *Oracle.* [Online] septiembre 2010. [Cited: octubre 6, 2011.] http://download.oracle.com/javaee/5/tutorial/doc/.

19. **Ramos Rodríguez, Msc. Yadiel.** *Descripción De La Arquitectura De Software SIIPOL 5.0.* Ciudad de la Habana : s.n., 2011.

20. Java. *Java.* [Online] [Cited: octubre 4, 2011.] http://java.com/es/about/.

21. LuaUf. [Online] mayo 13, 2008. [Cited: octubre 13, 2011.] http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/.

22. *MANUAL DE TRANSFERENCIA TECNOLÓGICA. ARQUITECTURA DEL SISTEMA SIIPOL.* Cuidad de la Habana : ALBET, S.A. IP–SW–DE–042.

23. *The Java EE 6Tutorial.* Redwood City, CA 94065 U.S.A : Oracle Corporation, 2011. 821–1841–13.

24. **Mann, Kito D.** *Java Server Faces in Action.* s.l. : Manning Publications Co., 2005. 1-932394-11-7.

25. JBoos Community. *RichFaces Developer Guide.* [Online] mayo 05, 2010. [Cited: marzo 21, 2012.] http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html/.

26. SpringSource. *Spring Framework Reference Documentation 3.1.* [Online] 2011. [Cited: mayo 28, 2012.] http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/.

27. **Walls, Craig and Breidenbach, Ryan.** *Spring in Action Second Edition.* s.l. : Manning Publications Co., 2008. 1-933988-13-4.

28. Hibernate Community Documentation. *Hibernate Reference Documentation.* [Online] mayo 02, 2012. [Cited: mayo 28, 2012.] http://docs.jboss.org/hibernate/orm/4.1/manual/en-US/html_single/.

29. **Serrano Hernández, Arniel and Rodriguez Hernández, Armando Alejandro.** *Software de Migración de ADABAS a Oracle.* Ciudad de la Habana : s.n., 2008.

30. KDE Documentation. *Elementos de UML.* [Online] [Cited: febrero 7, 2012.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

31. *CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

32. *CU Gestionar Solicitudes de Renovación del Permiso de Utilización de Sustancias Químicas. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

33. *CU Gestionar Solicitudes de Extensión. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

34. *CU Gestionar Solicitudes de Inclusión. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

35. *CU Gestionar Solicitud de Ampliación de Información. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

36. *CU Gestionar Permiso Provisional. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

37. *CU Gestionar Solicitud de Inspección. Módulo Sustancias Químicas.* Submódulo Solicitudes : s.n.

38. *CU Gestionar Carpeta de la Empresa. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

39. *CU Relacionar Proveedores a la Empresa. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

40. *CU Gestionar Resumen de Inspección. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

41. *CU Gestionar Permiso de Utilización de Sustancias Químicas. Módulo Sustancias.* Químicas Submódulo Experticias : s.n.

42. *CU Gestionar Permiso de Bombero. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

43. *CU Gestionar Experticia Química. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

44. *CU Gestionar Certificados de Inclusión o Extensión. Módulo Sustancias Químicas.* Submódulo Experticias : s.n.

45. *CU Consultar Documentos de la Empresa. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

46. *CU Consultar Experticia Química. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

47. *CU Consultar Resumen de Inspección. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

48. *CU Consultar Permiso Provisional. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

49. *CU Consultar Carpetas de las Empresas. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

50. *CU Consultar Empresas con Permisos Vencidos. Módulo Sustancias Químicas.* Submódulo Control de Investigación : s.n.

51. **Gómez León, Miguel.** *Análisis, Diseño e Implementación del Módulo Acciones Especiales del Sistema de Investigación e Información Policial.* Ciudad de la Habana : s.n., 2010.

52. **Pressman, Roger S.** *Ingeniería de Software un enfoque práctico. Quinta Edición.* s.l. : McGraw-Hill Companies, Febrero 2002. 8448132149.

53. **Larman, Craig.** *UML y Patrones 2a Edición. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : PEARSON EDUCACION, 2002. 9788420534381.

54. Java. *Composite View.* [Online] [Cited: marzo 26, 2012.] http://java.sun.com/blueprints/patterns/CompositeView.html.

55. Definición.de. *Definición de modelo de datos.* [Online] [Cited: octubre 4, 2011.] http://definicion.de/modelo-de-datos/.

56. **Rivero Duharte, Franklin.** *Pruebas Unitarias de Software en la Plataforma J2EE .* Ciudad de La Habana : s.n., 2008.

BIBLIOGRAFÍA CONSULTADA

57. dmapas. *Mapas Digitales S.A.* [Online] [Cited: septiembre 14, 2011.] http://www.dmapas.cl/productos_stegpol.htm.

58. **TEPPER FISHER, PAUL and MURPHY, BRIAN D.** *Spring Persistence with Hibernate.* s.l. : Apress, 2010. 978-1-4302-2632-1.

59. **Larman, Craig.** *UML Y PATRONES INTRODUCCIÓN AL ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS.* México : PRENTICE HALL, 1999. 970-17-0261-1.

60. *Oracle® Database Concepts, 10g Release 2 (10.2).* s.l. : Oracle, October 2005. B14220-02.

61. **Ramirez Luján, Ing. Dariena.** *ARQUITECTURA DE INFORMACIÓN V2.0 SIIPOL.*

62. *PLAN DISTRITAL DE SEGURIDAD CIUDADANA DE SANTIAGO DE SURCO.* 2007.

63. **FACULTAD 2.** *Propuesta de Guía para la presentación del Trabajo de Diploma.* Curso 2008-2009.

64. **Gacía Córdoba, Fernando.** *La Tesis y el trabajo de tesis.* México : Limusa , 2004. 968-18-6235-X.

65. **Bezos, Javier.** *Bibliografías y su ortotipografía.* Madrid : s.n., 2011.

66. **Universidad Calos III de Madrid.** *Cómo citar bibliografía.* 2012.

67. **Alvarez de Zayas, Dr. Cs. Carlos.** *METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA.* SANTIAGO DE CUBA : CENTRO DE ESTUDIOS DE EDUCACION SUPERIOR "MANUEL F. GRAN", 1995.

68. **Hernández León, Rolando Alfredo and Coello González, Sayda.** *EL PROCESO DE INVESTIGACIÓN CIENTÍFICA.* Ciudad de La Habana : Editorial Universitaria, 2011. 978-959-16-1307-3.

69. **VELASCO ELIZONDO, PERLA INÉS.** *PRUEBA DE COMPONENTES DE SOFTWARE BASADAS EN EL MODELO DE JAVABEANS.* s.l. : APIZACO, TLAX, ABRIL DE 2001.

70. **Monagas Reyes, Ing. Miguel Angel.** *GLOSARIO DE PROCESOS ELEMENTALES DEL NEGOCIO.* 2007.

71. **Valdés Jiménez, Ing. Sasha.** *GLOSARIO DE TÉRMINOS.* 2007.

GLOSARIO DE TÉRMINOS

Adabas-Natural: Es una base de datos jerárquica de alto rendimiento creada por la empresa alemana Software AG, en el año 1969. Actualmente se sigue comercializando bajo la versión Adabas 2006.

AOP: Aspect Oriented Programming, traducido al español: Programación Orientada a Aspectos, paradigma de programación que separa la lógica del negocio de aspectos de servicios, tales como la seguridad, las transacciones, entre otros.

API: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Bean: En el lenguaje Java es un componente software reutilizable. Existe con la finalidad de ahorrar tiempo al programar.

Eclipse Public License: Es un tipo de licencia que acompaña a todos los programas desarrollados por la Fundación Eclipse, de forma que cualquier uso, reproducción o distribución del programa constituye la aceptación del usuario de dicho acuerdo.

JavaScript: JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas.

Java Server Pages (JSP): es una tecnología Java, que permite combinar HTML estático con HTML dinámico, también se puede generar contenido dinámico en otros formatos como, XML y otros.

Open Source: Representa el software de dominio público, esto significa sin licencia, cuyo código fuente está disponible y se le permite usar y modificar.

plug-in: (del inglés "enchufable"), add-on (agregado), complemento, conector o extensión, es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

ANEXOS

Anexo 1. Diagrama de tablas del modelo relacional.

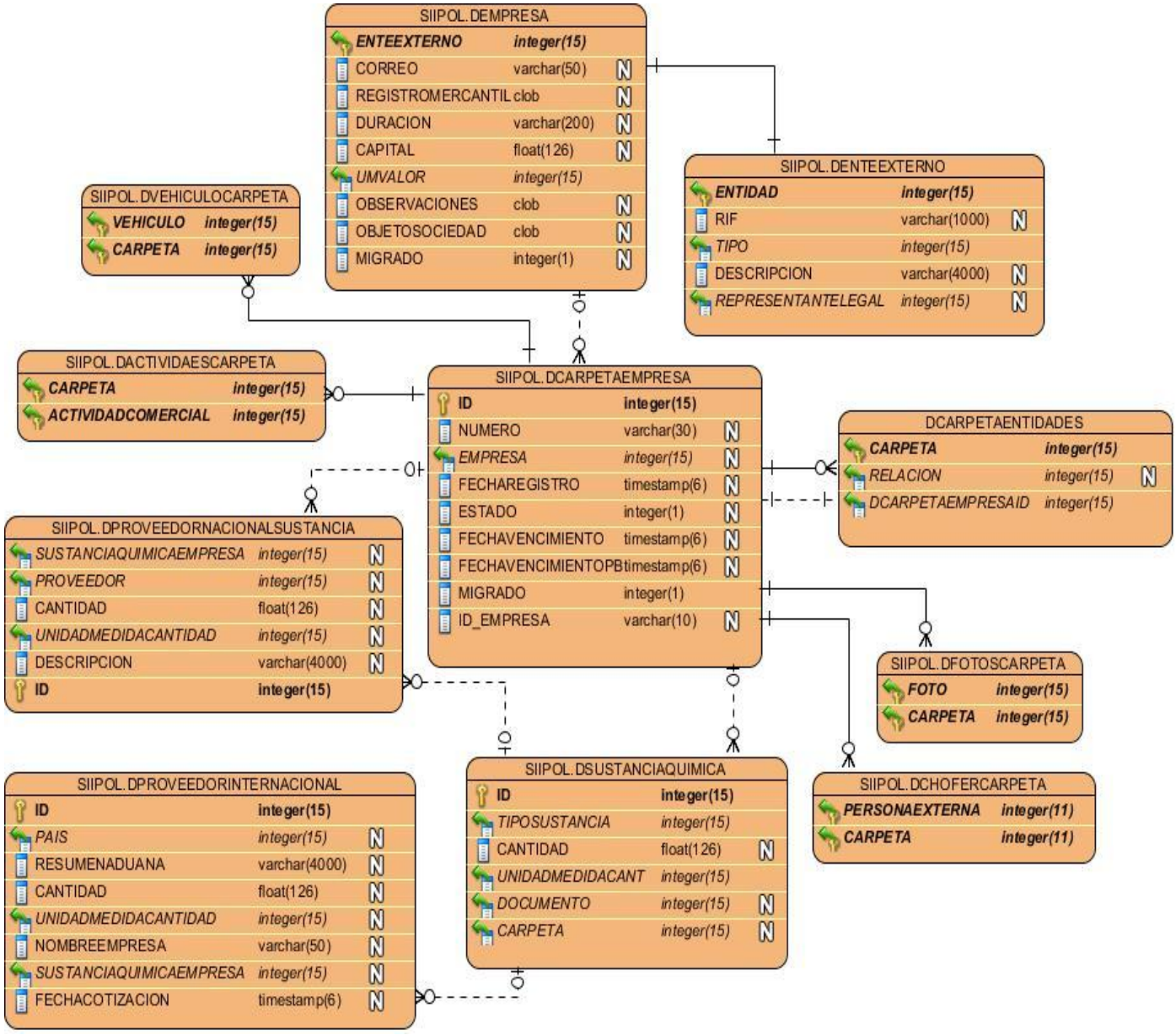


Figura. 24 Anexos. Diagrama de tablas del modelo relacional. Parte 1.

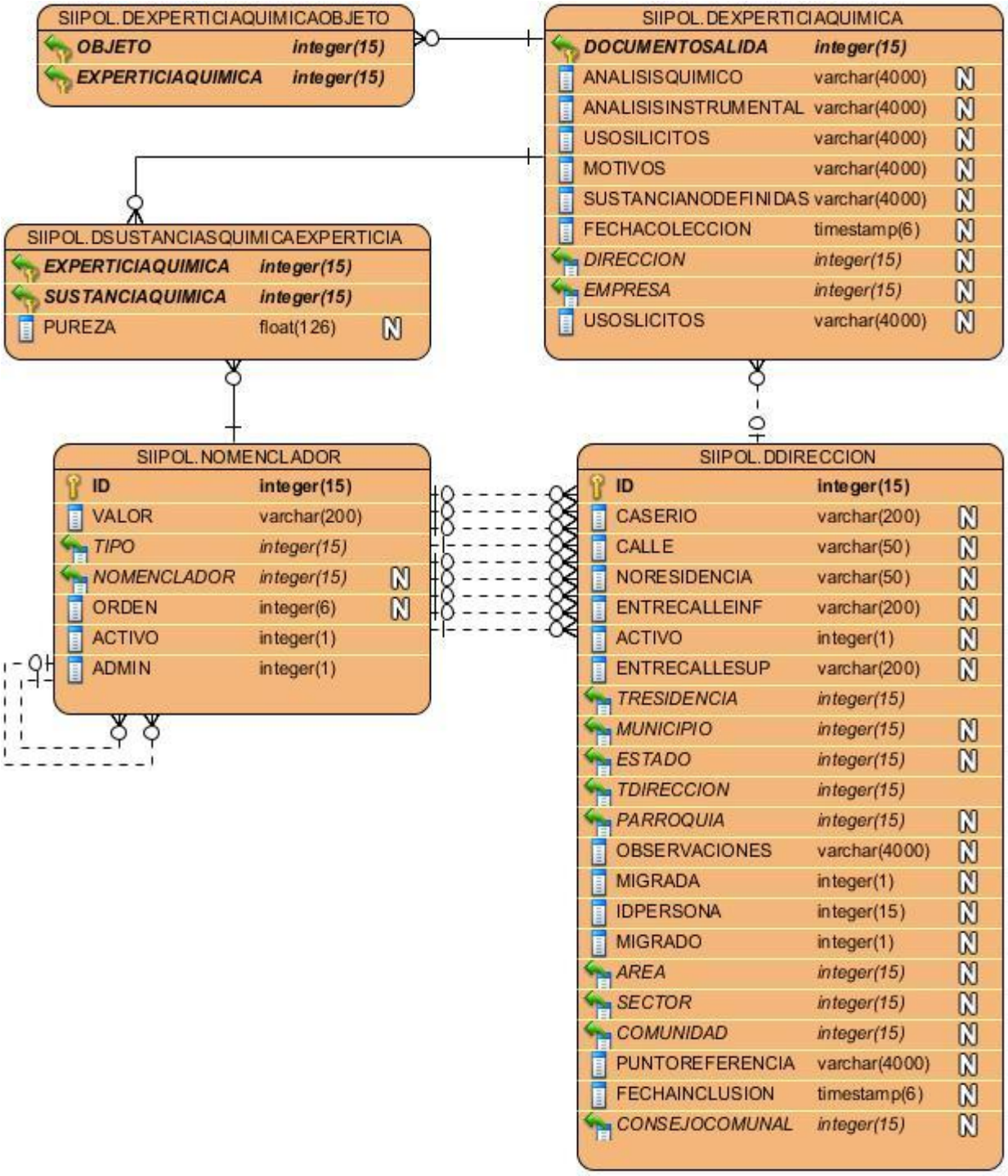


Figura. 25 Anexos. Diagrama de tablas del modelo relacional. Parte 2.

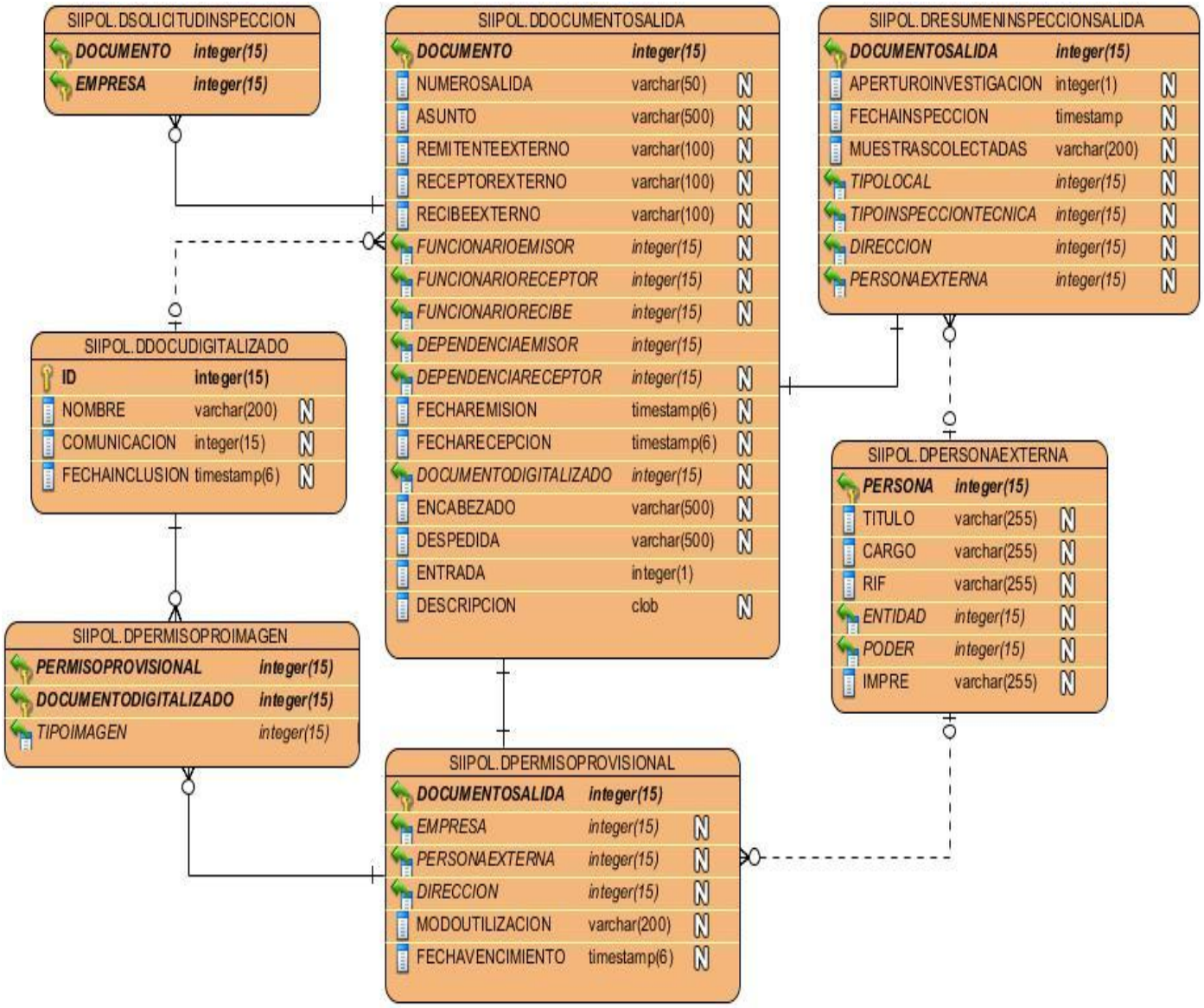


Figura. 26 Anexos. Diagrama de tablas del modelo relacional. Parte 3.

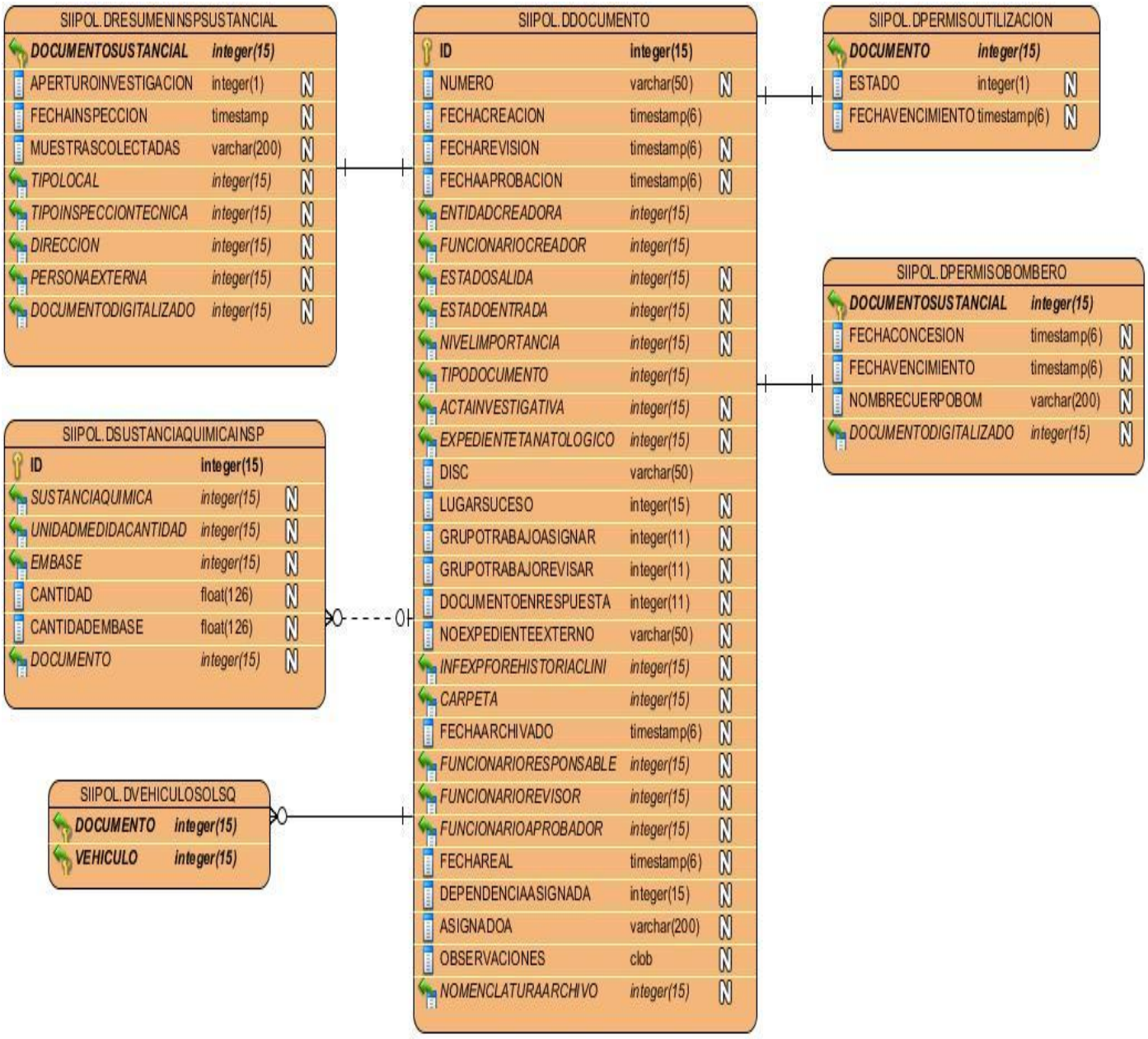


Figura. 27 Anexos. Diagrama de tablas del modelo relacional. Parte 4.

Anexo 2. Interfaz del CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas.

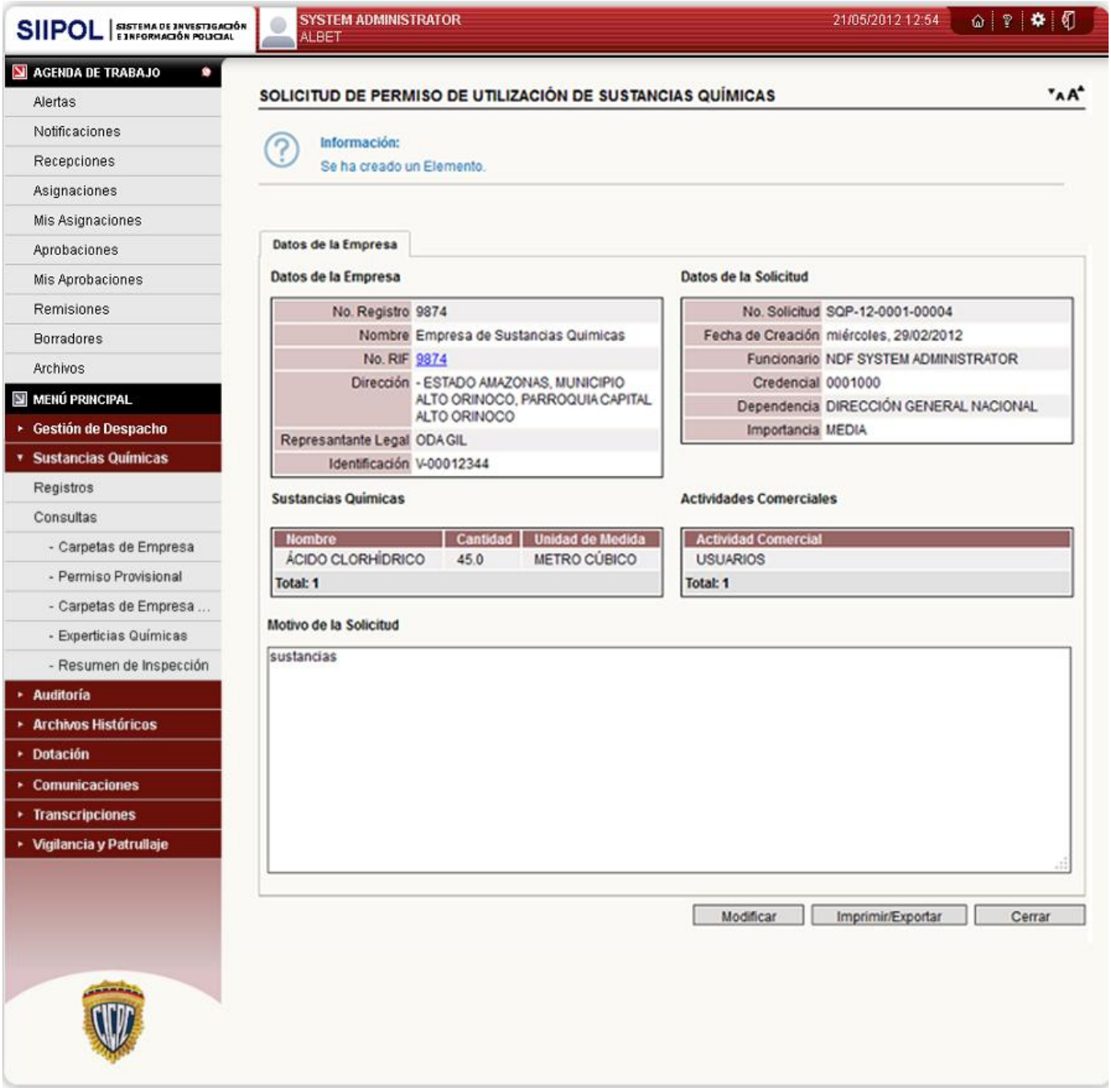


Figura. 28 Anexos. Interfaz. CU Gestionar Solicitud de Permiso de Utilización de Sustancias Químicas.