UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

Título: Sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos

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

Autores:

Pablo Enrique Reyes Gil

Yuliet Fernández López

Tutores:

Ing. Doris Medina Mustelier

Ing. Yuneimy Tellez Pérez

La Habana junio 2013

Año 55 de la Revolución

Declaración de Autoría

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

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

Pablo Enrique Reyes Gil	Doris Medina Mustelier
_____	_____
Firma del Autor	Firma del Tutor
Yuliet Fernández López	Yuneimy Tellez Pérez
_____	_____
Firma del Autor	Firma del Tutor

Datos de Contacto

Tutores:

Ing. Doris Medina Mustelier
Universidad de las Ciencias Informáticas, La Habana, Cuba
Email: dmedina@uci.cu

Ing. Yuneimy Tellez Pérez
Universidad de las Ciencias Informáticas, La Habana, Cuba
Email: ytellez@uci.cu

Agradecimientos

A mi hermanito y a mi mamá. Que sepan que el mérito de este trabajo es por ustedes.

A mi tía hay tiita que sería mi vida sin ti. Gracias por guiarme en la vida, que si hoy puedo ser alguien es por ti.

Hay pipo (Pablo) que voy a decir de ti, mi compañero, mi amigo, mi resabioso del alma, gracias por estar en todos los momentos hay conmigo. Gracias otra vez corazón.

A mi hermanita postiza Ainala, gracias por ser la amiga que eres, tú sabes que esto es de las dos.

Gorda (Diana) gracias por estar en todos los momentos presente.

Que voy a decir de eso suegros maravillosos que dios me ha dado (Sunilda y Eduardo). Gracias de corazón.

A mis profesores durante la carrera en especial a mis queridos Yindra, Yunier René y Neo que sin ustedes hoy no pudiera estar cumpliendo mi sueño. Esto se lo dedico a ustedes por confiar en mí.

Fifi (Rafaela) no me olvide de ti, gracias por ser esa madre, por tus consejos y por todo tu apoyo. Gracias desde el fondo de mi corazón.

En especial le quiero agradecer a mis tutoras Doris y Yune por su tiempo, su dedicación y no rendirse conmigo. ¡Lo logramos!

Yuliet.

Agradecimientos

A mis padres que son la fuerza y el ejemplo que me inspira a seguir adelante cada día gracias por siempre estar presente apoyarme en todo, empujarme hacia las mejores decisiones cuando no he elegido el camino correcto, gracias por convertirme en quien soy y ayudarme a realizar mi sueño, esto es para ustedes que son la luz de mi vida. Los quiero.

A mis abuelos y mi tía que tanto cariño me brindado siempre que estoy cerca, todo lo que hago es para que se sientan orgullosos de mí.

A mi novia casi mi esposa, que ha compartido conmigo estos 5 años, gracias por soportarme, quererme y apoyarme siempre que lo he necesitado, te quiero mucho.

A todos mis compañeros en especial al Grande, Darlon, Keimer, el Rene, Sandy, Trujillo, Dago, Tomás. Diana, Andry, gracias de corazón a todos.

A mis tutoras que tanto tiempo le han dedicado a este trabajo, de verdad muchísimas gracias, ojalá todos pudieran contar con tutoras como ustedes.

Pablo

Resumen

Las Tecnologías de la Información y las Comunicaciones (TICs) son un factor de vital importancia en la transformación de la sociedad, especialmente con la necesidad del almacenamiento de información que cada día tiene mayor auge. La presente investigación, se enmarca en la concepción e implementación de un sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos. Este sistema le permitirá a los especialistas del departamento Almacenes de Datos del centro DATEC de la Universidad de las Ciencias Informáticas (UCI), gestionar los metadatos de forma automatizada y segura. Producto a esto, surge como objetivo fundamental de la investigación desarrollar el sistema propuesto. Para su cumplimiento, se realizó un estudio de las tendencias actuales de desarrollo que facilitó una mejor comprensión del sistema a desarrollar. Posteriormente se procedió a fundamentar las herramientas, tecnologías y metodología a utilizar. El proceso estuvo guiado por la metodología de desarrollo Open Up y se utilizó como lenguaje de programación Java. De igual forma, se efectuó el análisis y diseño, dándole paso a la implementación y prueba del sistema. Como resultado final, se obtuvo un sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, cumpliéndose el objetivo propuesto de la investigación.

Palabras Claves: sistema, Almacenes de Datos, metadato, sistema de gestión y proceso de integración.

ÍNDICE DE CONTENIDO

Índice de Figuras

Introducción

La humanidad ha alcanzado un considerable desarrollo tecnológico a escala mundial, siendo la segunda mitad del siglo XX un período con abundante crecimiento en muchos sectores científicos. Es indiscutible que la informática ha tomado el control en cuanto a adelanto tecnológico se refiere, convirtiéndose en una ciencia clave para el progreso de la humanidad y de excepcional apoyo para otras ramas tecnológicas. Todo este desarrollo ha contribuido a que exista mayor intercambio de datos en el mundo. Dicho intercambio es realizado a través de sistemas de información que son cada vez más complejos. En la actualidad, para suplir estas necesidades de gestión, se desarrollan aplicaciones informáticas con diversas herramientas y métodos de desarrollo.

Cuba no ha quedado exenta de estos avances tecnológicos. En los últimos años, el país ha encaminado sus esfuerzos en informatizar diversas entidades con el objetivo de lograr un mayor nivel de desarrollo en determinados sectores. La UCI desde su fundación, juega un papel importante en el desarrollo informático de la sociedad cubana.

La UCI cuenta con diversos centros productivos, uno de ellos el Centro de Tecnologías de Gestión de Datos (DATEC), el cual posee varios departamentos. En el Departamento de Almacenes de Datos, las actividades son organizadas mediante grupos de trabajos, como el grupo de “Integración de Datos”. El mismo es responsable de desarrollar el proceso de integración de datos en la construcción de soluciones de almacenes de datos. A partir de este, se logran visiones únicas de sistemas de información y se generan disímiles metadatos, que permiten administrar y auditar el proceso. Estos metadatos tanto técnicos como de negocio, son almacenados en un sistema gestor de base de datos. Sin embargo, sería de apoyo poder consultarlos y gestionarlos a través de herramientas que provean a los administradores, realizar el proceso de integración de datos de forma automatizada.

Actualmente, cuando se necesitan insertar nuevos metadatos o realizar alguna operación en el modelo de datos, se debe tener un conocimiento total del mismo, así como de las dependencias entre las tablas para no violar las relaciones referenciales. Se debe además consultar la información a través de consultas

SQL, las cuales solo pueden ser generadas por especialistas de base de datos y no por el usuario final. Por otro lado, resulta necesario establecer reglas y políticas, para el acceso a los datos de los metadatos que se almacenan para dar soporte a los procesos de integración, debido a que constituyen una información valiosa.

De la situación anterior se puede definir como **problema de la investigación** ¿Cómo contribuir a la gestión de metadatos en el proceso de integración de datos en soluciones de almacenes de datos? Definiéndose como **objeto de estudio** de esta investigación la gestión de los metadatos y especificándose como **campo de acción,** los sistemas de gestión de metadatos para la auditoría y control del proceso de integración de datos en soluciones de almacenes de datos.

Proponiendo como **objetivo general** desarrollar un sistema de gestión de metadatos para la auditoría y el control del proceso de integración de datos en soluciones de almacenes de datos.

De acuerdo con el análisis realizado se desglosan los siguientes **objetivos específicos**:

- ✓ Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.
- ✓ Realizar el análisis y diseño del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.
- ✓ Realizar la implementación y pruebas al sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

Con el propósito de dar cumplimiento a los objetivos anteriormente planteados, se definen las siguientes **tareas de la investigación**:

- ✓ Realización de un estudio bibliográfico relacionado con los sistemas de gestión de metadatos para la auditoría y control del proceso de integración de datos desarrollados, que permita determinar el estado de las tendencias actuales.
- ✓ Realización de un estudio bibliográfico, que permita fundamentar la selección de las herramientas y tecnologías a utilizar en el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

- ✓ Levantamiento de los requisitos que permitan conocer las necesidades del negocio.
- ✓ Definición de la arquitectura que permita establecer el marco de trabajo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.
- ✓ Diseño del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, para guiar las actividades de implementación.
- ✓ Implementación del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, para dar cumplimiento a los requisitos definidos en el negocio.
- ✓ Definición de los Casos de Pruebas para guiar las pruebas del sistema.
- ✓ Realización de las pruebas unitarias y pruebas del sistema para probar el funcionamiento del sistema implementado, para que todo el software funcione correctamente.

Estructura del documento

El presente trabajo de diploma estará compuesto por resumen, introducción, tres capítulos, conclusiones generales, bibliografías, referencias bibliográficas y anexos. A continuación, se realiza una descripción de los principales elementos contenidos en cada acápite.

Capítulo 1: Fundamentación Teórica del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

En este primer capítulo se realiza un estudio de los conceptos fundamentales relacionados con el tema de la investigación. Se describen los elementos asociados con el proceso de integración datos y la gestión de los metadatos. Se definen las tecnologías, herramientas y metodología de desarrollo de software a utilizar para dar cumplimiento al objetivo propuesto.

Capítulo 2: Análisis y diseño del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

En este segundo capítulo se realiza un estudio preliminar del negocio, con el fin de obtener los requisitos funcionales y no funcionales que el sistema debe cumplir. Se elaboran los diagramas de clases, los

diagramas de iteración para los casos de uso arquitectónicamente más significativos, el diagrama de despliegue para dar una visión de cómo quedará el despliegue de la solución. Se define además la arquitectura del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

Capítulo 3: Implementación y Prueba del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

En este tercer capítulo se presentan los artefactos de las actividades de implementación y pruebas necesarias para cumplir los objetivos del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos. Se explican los estándares de codificación utilizados en las clases principales. Además se muestran los resultados de las pruebas desarrolladas, siendo estas las pruebas unitarias y pruebas del sistema.

Capítulo 1: Fundamentación Teórica

Introducción

En el presente capítulo se realiza un estudio bibliográfico de los sistemas de gestión de metadatos desarrollados, que permite determinar el estado de las tendencias actuales. Se describen los principales conceptos asociados al dominio del problema, haciendo énfasis en sus características y aspectos más relevantes. Se definen las actividades del proceso de integración de datos, abordando la definición de los metadatos, sus distintos tipos y se describe la metodología, las herramientas y tecnologías a utilizar para resolver la problemática.

1.1. Proceso de integración de datos

El término *integración de datos* está asociado a la combinación de datos procedentes de distintas fuentes heterogéneas (1). El objetivo fundamental de este proceso es de proporcionar una visión única y global de los datos, permitiendo el acceso a los mismos, reformatearlos, limpiarlos, moverlos desde múltiples fuentes y cargarlos en otro sistema para su posterior análisis. Este proceso surge producto a la necesidad de interconectar las distintas aplicaciones que puede tener una organización y así unir la información fragmentada en distintos repositorios ((2) (3)). Permite además, la entrega de información y el conocimiento del negocio. En la Figura 1 se muestra el proceso básico de integración de datos e información.

Figura 1: Proceso de integración de datos e información

Los procesos de integración cuentan con disímiles técnicas (Véase,

Figura 2) que se establecen en dependencia del objetivo de la técnica en sí. A continuación se explica en qué consisten cada una de ella y las ventajas de utilizarlas.

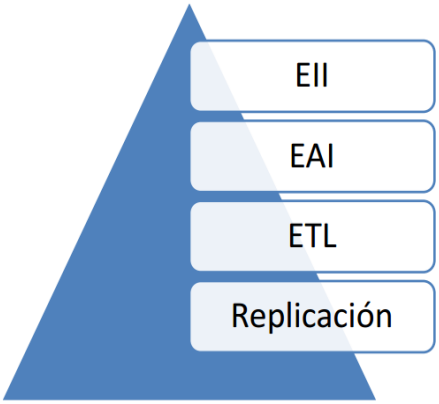


Figura 2: Técnicas de integración de datos e información

Replicación: la replicación es un conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de una base de datos a otra, para luego sincronizar ambas bases de datos con el fin de mantener su coherencia. Es útil dividir la replicación en dos amplias categorías: replicación de datos, en un servidor

para un entorno de servidor y replicación de datos entre un servidor y los clientes. La utilización de esta técnica presenta dentro de sus principales ventajas, no tener impacto en la aplicación y que se ejecuta prácticamente en tiempo real ((1) (4)).

ETL por sus siglas en inglés Extracción, Transformación, y Carga (Extract, Transform, Load): esta técnica permite mover datos desde múltiples fuentes, reformatearlos, limpiarlos, y cargarlos en otra base de datos, en un data mart (mercado de datos), o data warehouse (almacén de datos) para su posterior análisis. Particularmente la extracción consiste en sustraer los datos desde las fuentes de origen ubicados en bases de datos relacionales, NoSQL[1], ficheros planos y demás estructuras de datos ((5) (1)). Sin embargo, la transformación se efectúa una vez terminada la fase de extracción, realizando un chequeo que verifique si los datos cumplen con las pautas estipuladas. En caso de que los datos no cumplan, se aplican una serie de trasformaciones para que estos queden listos y puedan ser cargados (6). La fase de carga interactúa directamente con la base de datos destino y es la encargada de almacenar finalmente los datos que se ajustan con las necesidades del negocio (1).

La técnica ETL está diseñada para proporcionar una plataforma que permita mejorar la productividad mediante la reutilización de objetos, se sustenta sobre la gestión de metadatos y realiza la carga de almacenes de datos para migrar o combinar datos ((1) (4)).

EAI por sus siglas en inglés Integración de Información Empresarial (Enterprise Application Integration): engloba las metodologías, procesos, herramientas y tecnologías usadas para conectar sistemas, datos y procesos de una entidad o de un conjunto de entidades. La misma está orientada a transacciones de punto a punto, es decir, son conjuntos de adaptadores y enrutadores que mueve transacciones de negocio en forma de mensajes entre las aplicaciones interconectadas. Entre las ventajas se destaca que permite una solución alternativa simple (1).

[1] Es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS), entre sus características más notables es que no usan SQL como el principal lenguaje de consultas, los datos almacenados no requieren estructuras fijas como tablas y normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, coherencia, aislamiento y durabilidad).

EII por sus siglas en inglés Integración de Información Empresarial (Enterprise Information Integration): es la integración de datos a partir de múltiples sistemas en un formato de representación unificada, coherente y precisa orientada a la manipulación de datos y la navegación. Los datos son agregados, reestructurados, re-etiquetados (si es necesario) y presentados al usuario. Por lo general, el resultado de este enfoque es prácticamente un sistema distribuido heterogéneo integrado de información. Entre sus ventajas se destaca, que se puede acceder directamente a los datos desde los sistemas de información involucrados sin moverlos de su base de datos origen (7).

Para el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, la presente investigación debe interactuar con la técnica de ETL. La misma se utiliza en el Departamento de Almacenes de Datos en el cual está enmarcado el problema de la investigación.

1.2. Metadatos

Los metadatos son definidos como la información sobre los datos que posee una empresa o que utiliza un sistema de información. Suelen referirse a la estructura, significado o procedencia de los datos. Muchos especialistas llaman a los metadatos, datos acerca de los datos. Para el caso particular de los almacenes de datos, los metadatos son similares a una enciclopedia. Se refieren a la estructura, el contenido y las relaciones que existen entre los componentes del almacén. Los metadatos son información descriptiva sobre los datos, los procesos que lo manipulan y otras estructuras como objetos y reglas de negocio. Los mismos son agrupados en 3 categorías en dependencia de lo que describen ((8) (9) (10)):

Metadatos técnicos: enfocado a los diseñadores, desarrolladores y administradores durante la construcción, el mantenimiento y la gestión del ambiente de desarrollo utilizado. Es el punto técnico que agrupa las herramientas, aplicaciones y sistemas, para que juntos construyan la solución. Un ejemplo de la utilización de los metadatos técnicos se puede apreciar en el diseño del esquema de un almacén de datos de un repositorio, en el cual los metadatos son utilizados para generar el script que contiene las tablas del almacén.

Metadatos de negocio: muestra una imagen del ambiente de trabajo a los usuarios finales, así como los resultados del proceso de integración de datos en función de las métricas y requerimientos del negocio. A través de este tipo de metadato se puede obtener una descripción de las terminologías del negocio u otros datos que brinden información del proceso de integración.

Metadatos de proceso: es la presentación de las estadísticas sobre los resultados de la ejecución del propio proceso de integración de datos, incluyendo medidas tales como filas cargadas con éxito, filas rechazadas, y la cantidad de tiempo de carga.

El presente trabajo estará enfocado en gestionar los metadatos de negocio básicos en un proyecto de integración de datos. En el modelo a gestionar se incluyen algunos metadatos de proceso y metadatos técnicos, aunque estos no son los que prevalecen.

1.3. Sistemas de gestión de metadatos

Los sistemas de gestión son mecanismos que permiten adquirir, ordenar y emplear adecuadamente los recursos en función de un objetivo dado. Apoyan el cumplimiento de las metas de una organización mediante una serie de estrategias que incluyen la optimización de los procesos, enfoques centrados en la gestión y el pensamiento disciplinado. Se utilizan para gestionar información generada en escuelas, empresas, institutos de investigación, órganos de seguridad interior, entre otros; lo que posibilita aprovechar y desarrollar el potencial existente en cualquiera de estos sectores.

En la actualidad existen en el mundo una amplia gama de software dedicado a la integración de datos. Los mismos obtienen e integran los datos, aplicaciones e información de prácticamente cualquier sistema de negocio y en cualquier formato (11). Entre los software que se utiliza para la integración de datos es el Oracle Warehouse Builder, que es una herramienta para la administración completa del ciclo de vida de los datos y de los metadatos. El mismo provee una alta calidad de información sobre los datos y completa la integración relacional y modelamiento dimensional. Provee una interfaz amigable y permite el diseño de los procesos ETL, donde intervienen los almacenes de datos, las fuentes de almacenamiento y los usuarios finales. Otra aplicación destinada al proceso de integración es Informática Data Integration, la cual se caracteriza por poseer una arquitectura de integración de datos basada en metadatos y servicios.

Contiene un sólido entorno de desarrollo y herramientas de productividad, acceso universal a los datos, y está optimizado para cada entorno y aplicación (11).

Las herramientas mencionadas anteriormente poseen algunas desventajas como son el ser software privativo, con elevados precios en el mercado. Además de que realizan la gestión de los metadatos de forma genérica, al estar mayormente enfocadas a la implementación de las técnicas de integración de datos e información y no a la gestión de los metadatos generados en dichas técnicas.

El sistema propuesto en la presente investigación no pretende sustituir estas herramientas ya existentes en el mercado sino que permitirá complementarlas para potenciar la gestión y auditoría de los metadatos de negocio.

1.4. Metodologías de desarrollo

Una metodología de desarrollo de software define quién está haciendo qué, cuándo, y cómo alcanzar un objetivo específico, proporcionando normas para el desarrollo eficiente del software con calidad. Actualmente existen numerosas metodologías tradicionales y ágiles. En la presente investigación se realiza un estudio de algunas de ellas, con el fin de seleccionar la adecuada para el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos (12).

1.4.1. Metodología Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Desarrollo de Software (RUP[2]) se caracteriza por tomar los mejores elementos de metodologías anteriores permitiéndole hacer determinadas variaciones en dependencia del tipo de sistema a desarrollar lo que hace que sea flexible ante cambios. Se encuentra preparada para desarrollar complejos proyectos y describe detalladamente todas las actividades, roles, responsabilidades y artefactos. Está orientada a objetos y utiliza UML como notación para la representación visual. Sus principales características son (13):

[2] Por sus siglas en inglés Rational Unified Process

- ✓ **Dirigido por casos de uso:** los casos de uso se utilizan para capturar los requisitos funcionales. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, entre otras.
- ✓ **Centrado en la arquitectura:** la arquitectura muestra una visión completa del sistema que se desarrolla, por lo que le ofrece tanto a los clientes como a los desarrolladores una idea clara de la misma. Se toman decisiones concernientes a cómo debe ser construido el sistema y en qué orden. Incluye elementos de calidad, rendimiento, reutilización y capacidad de evolución.
- ✓ **Iterativo e incremental:** las iteraciones se refieren a pasos en cada flujo de trabajo y sus incrementos. La idea básica es desarrollar el sistema siguiendo etapas incrementales, caracterizadas por la generación de sucesivas versiones que van abarcando requerimientos hasta completar el sistema.

RUP agrupa sus actividades en 9 disciplinas principales. En la Figura 3 se muestran gráficamente las disciplinas de trabajo y las fases donde intervienen.

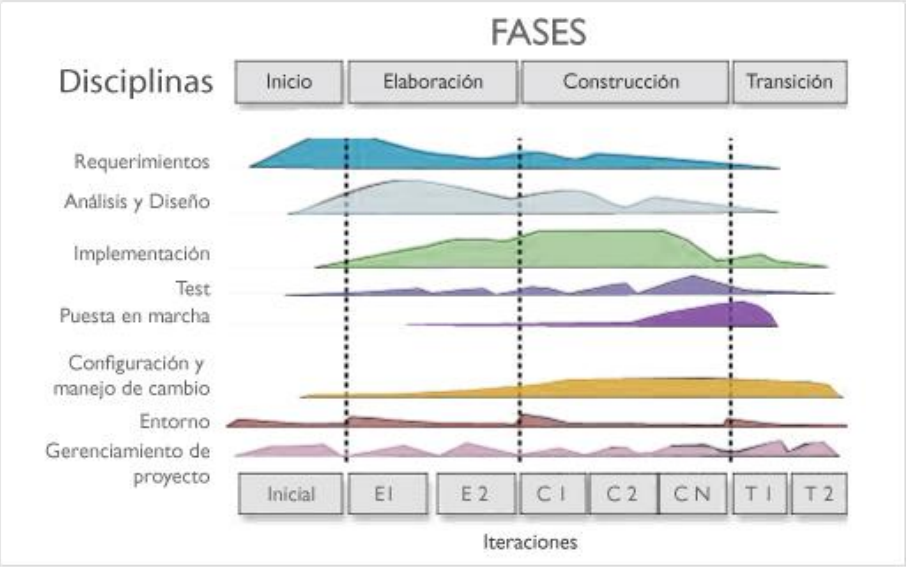


Figura 3: Fases de RUP (13)

1.4.2. Metodología Proceso Unificado Abierto

La metodología de Proceso Unificado Abierto (Open Up[3]) provee un ciclo de retroalimentación relativamente corto, que permite flexibilidad y mejor adaptación a las decisiones tomadas dentro de cada iteración (14). Divide el proyecto en iteraciones que son planeadas sobre intervalos de tiempo definido en semanas. Dichas iteraciones se centran en el cumplimiento de los objetivos definidos previamente en el plan para cada una, por parte del equipo de desarrollo, donde cada ciclo iterativo debe concebir como resultado un demo o un ejecutable con funcionalidades específicas (15). Open Up en cada iteración del ciclo de vida, incrementa progresivamente los objetivos de las iteraciones anteriores, añadiendo nuevas funcionalidades a las versiones estables del software que se tiene hasta el momento. Dentro del ciclo de vida tiene 4 fases (Véase, Figura 4):

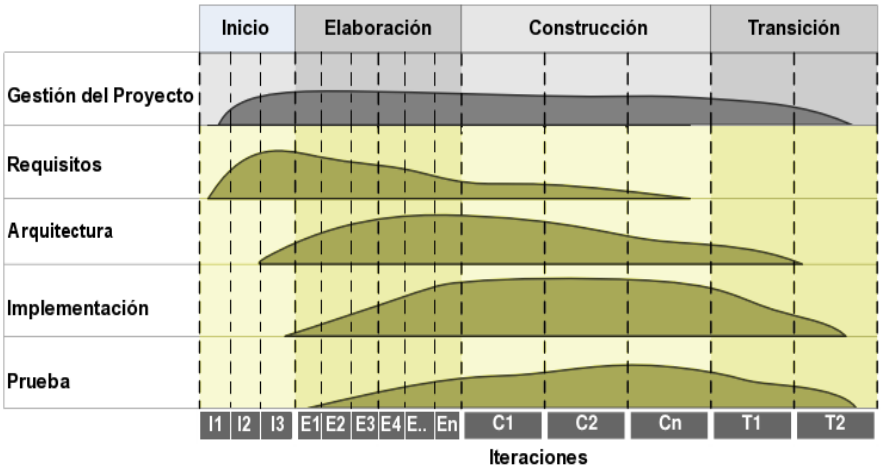


Figura 4: Fases de Open Up

Esta metodología preserva las características fundamentales de RUP, que incluye un desarrollo iterativo, casos de uso y escenarios, administración de riesgos, entre otros elementos. Excluye la mayoría de las partes opcionales de RUP e incluye varios elementos nuevos. Es ágil, ligera y promueve el desarrollo del software sobre las buenas prácticas, haciéndola una metodología pequeña, extensible y si es necesario incluye partes de otros modelos. Open Up se adapta a condiciones específicas de proyectos ágiles y

[3] Por sus siglas en inglés Open Unified Process

pequeños. Permite entregar un software con calidad, pues constituye un proceso unificado de corta duración, aplicado de manera iterativa e incremental dentro de un ciclo de vida. El mismo está estructurado en tres capas: micro incrementos, ciclo de vida por iteración y ciclo de vida por el proyecto (16). Open Up se centra en el proceso colaborativo de desarrollo del software que puede ser aplicado a variedades de proyectos en diferentes direcciones.

Entre los elementos fundamentales que permitieron la selección de la metodología a utilizar en el desarrollo del sistema de gestión para la auditoría y control del proceso de integración de datos están, la cantidad de desarrolladores, el limitado tiempo para realizar la solución propuesta, la complejidad y su innecesario volumen de documentación dado el número de desarrolladores. Estas características propias de la solución, permitieron determinar a Open Up como metodología de desarrollo.

1.5. Herramienta CASE[4] Visual Paradigm

Con el fin de desarrollar programas utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas, existen hoy en día diversas herramientas que han sido creadas para el desarrollo de la Ingeniería de Software. Las herramientas constituyen un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (17), una de ellas es Visual Paradigm. Esta herramienta soporta el ciclo de vida completo del proceso de desarrollo del software a través de la representación de diferentes diagramas y fue desarrollada con el fin de trabajar sobre el lenguaje de modelado UML, el cual es un lenguaje estándar que permite describir el sistema en forma de modelo. Incluye aspectos conceptuales tales como procesos de negocios, funciones del sistema y aspectos concretos como componentes de software reutilizables y expresiones de lenguajes de programación. Visual Paradigm fue creado para una amplia gama de usuarios interesados en la construcción de sistemas de software utilizando un enfoque orientado a objetos (18).

[4] CASE (Computer Aided Software Engineering, Ingeniería de Software Assisted port compotators)

Dentro de sus principales características se pueden encontrar:

- ✓ Disponibilidad en múltiples plataformas (Windows, Linux).
- ✓ Diseño centrado en casos de uso y enfocado al negocio.
- ✓ Uso de un lenguaje estándar que facilita la comunicación para todo el equipo de desarrollo.
- ✓ Provee dos tipos de licencia (gratuita y comercial).
- ✓ Soporta el diseño de aplicaciones web.
- ✓ Generación de código para java y exportación como HTML.
- ✓ Soporte de UML versión 2.1.
- ✓ Ingeniería inversa de bases de datos desde sistemas gestores de bases de datos existentes a diagramas de Entidad-Relación.
- ✓ Distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas UML.

Se selecciona Visual Paradigm para UML por ser una herramienta profesional, fácil de usar que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite modelar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

1.6. Gestores de base de datos

Los Sistemas de Gestores de Base de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que lo utilizan. Se componen de los lenguajes de definición de datos, manipulación de datos y de consulta (19). En la actualidad existen numerosos SGBD, entre ellos Microsoft Access, Oracle, MySQL, SQL Server y PostgreSQL, entre otros.

Con el fin de lograr una mejor comprensión del funcionamiento de estos sistemas de gestión, se realizará a continuación una caracterización de algunos de los más utilizados en la actualidad (20).

1.6.1. Oracle

Es un manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar el aprovechamiento al máximo de toda la información. Este gestor también cuenta con un grupo de ventajas que se mencionan a continuación (21).

- ✓ Posee igual interacción en todas las plataformas (Windows, Unix, Macintosh y Mainframes).
- ✓ Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- ✓ Tiene una amplia gama de herramientas para operar con la base de datos, dichas herramientas facilitan el trabajo tanto de los usuarios como de los administradores.

A pesar de todas las ventajas que ofrece presenta algunas desventajas, entre las que se encuentran su elevado precio comercial en el mercado y que requiere recursos y prestaciones específicas en el equipo donde se encuentre instalado el gestor de base de datos.

1.6.2. PostgreSQL

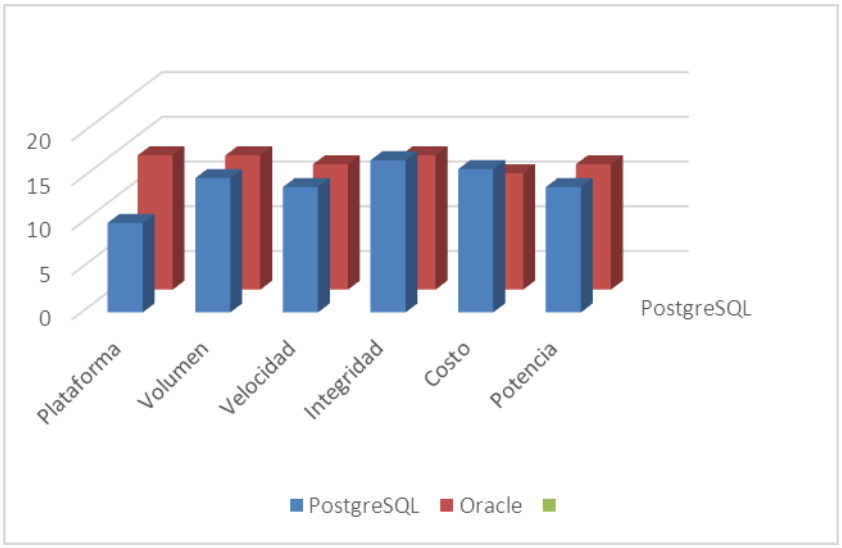
PostgreSQL es un gestor de bases de datos orientado a objetos, es la alternativa a MySQL cuando se necesitan características avanzadas como transacciones, procedimientos almacenados y vistas (22). Tiene la capacidad de almacenar procedimientos en la propia base de datos. Además, permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de ser bloqueados. Al igual que los demás gestores de base de datos posee un grupo de prestaciones (23):

- ✓ El API[5] de acceso al SGBD se encuentra disponible en C, C++, Java, Perl, PHP, Python y TCL, entre otros.

[5] (Application Programmer Interface)

- ✓ Cuenta con disímiles tipos de datos, además se pueden definir nuevos operadores programados por el usuario.
- ✓ Su administración se basa en usuarios y privilegios.

A modo de conclusión en la se establece una comparación entre los diferentes SGBD antes mencionados.



Gráfica 1: Comparación entre los SGBD

Dadas las características anteriores, el SGBD que se escogió para desarrollar la aplicación fue PostgreSQL. El mismo es un sistema de base de datos relacional orientado a objetos, es un software gratuito y libre de comercializar. Debido a ello se va mejorando constantemente por los propios usuarios. En su versión más actual, no presenta grandes diferencias en comparación con otros sistemas gestores de base de datos existentes, además este sistema gestor forma parte de la tecnología que usa la entidad para el cual se va a realizar la aplicación.

1.7. Entorno de desarrollo integrado

Un entorno de desarrollo integrado es una herramienta de software dedicada al desarrollo de programas informáticos, brindando una serie de complementos que facilitan el desarrollo ágil de software. Para el desarrollo de aplicaciones en el lenguaje de programación Java, se utilizan fundamentalmente dos IDEs: Eclipse y NetBeans, sin descartar a IntelliJ, pero este por ser comercial y de elevado costo no posee un buen respaldo en la comunidad (24).

1.7.1. NetBeans

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito para Java, pero puede servir para cualquier otro lenguaje de programación (25). Existe además un número importante de módulos para extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso (26). Presenta un grupo de características que se mencionan a continuación:

- ✓ Los proyectos desarrollados no dejan de ser multiplataforma y poseen lanzadores para cada plataforma.
- ✓ Sistema de ventanas práctico para desarrollar las interfaces de usuario.
- ✓ Su licencia permite construir aplicaciones de código abierto como comerciales.

A pesar de las características antes mencionadas el mismo también presenta desventajas como son la poca existencia de plug-ins para esta plataforma y que no posee un editor de código HTML (27).

1.7.2. Eclipse

Eclipse es una plataforma de programación, desarrollada originalmente por IBM. Actualmente sigue siendo desarrollada por la Fundación Eclipse, la cual constituye una organización independiente de código abierto. Eclipse basa su funcionamiento en módulos para proporcionar todas sus funcionalidades, esto lo hace muy versátil y portable a diferencia de otros IDE (28). Este mecanismo de módulos es una plataforma ligera para componentes de software, lo que permite a Eclipse extenderse a varios lenguajes de programación como son Java, C/C++, PHP, Cobol, Python entre otros. Tiene soporte para la gestión de

la configuración y el control de versiones, incluye plug-ins para realizar pruebas de unidad entre otras características (28).

Se decidió utilizar Eclipse, ya que en la práctica resultó ser más económico en la utilización de los recursos de hardware y por lo tanto muestra mayor rendimiento. Tiene además, los beneficios de ser una herramienta de código abierto y poder soportar las herramientas que manipulan los diferentes tipos de lenguajes. Se pueden sumar como características la capacidad de soportar distintas arquitecturas, permitiendo la integración con herramientas CASE.

1.8. Plataforma de desarrollo Java

Se entiende por plataforma, el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones, sobre el cual un programa puede ejecutarse (29).

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de la compañía Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones (30). Hoy en día esta plataforma ha evolucionado en concordancia con el avance tecnológico y se ha convertido en una de las plataformas de programación más usadas por los desarrolladores. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que esté operando. Es una tecnología orientada al desarrollo de software con el cual se puede realizar cualquier tipo de programa. Ha cobrado mucha importancia en el ámbito web con su plataforma (31).

1.8.1. Lenguaje de programación Java

Java es un lenguaje simple, orientado a objetos, distribuido, intérprete, robusto, seguro, de arquitectura neutral, portátil, de alto desempeño, de hilos múltiples y dinámico ((31) (32)). A continuación se presentará una descripción de esas características:

Orientado a objetos: esto significa que se debe poner especial atención a los datos de la aplicación y a los métodos que manipulan los datos, y que no se debe pensar estrictamente en términos de procedimientos (33).

I**ntérprete**: el compilador genera bytecode para la máquina virtual de Java, en vez de código nativo de máquina. Para que se ejecute realmente un programa, se usa el intérprete de Java para ejecutar los bytecode compilados (34).

Arquitectura neutral y portátil: como los programas de Java se compilan en un formato de bytecode de arquitectura neutral, una aplicación de Java se puede ejecutar en cualquier sistema, siempre y cuando dicho sistema instrumente la máquina virtual de Java y se podrá ejecutar sobre cualquier plataforma. (35).
Dinámico y distribuido: una clase de Java se puede cargar en todo momento en un intérprete de Java en ejecución, a esta se le puede denominar lenguaje distribuido. Esto significa, que proporciona un soporte de alto nivel para redes (36).

Robusto: Java se diseñó para escribir software robusto, pues no elimina la necesidad de asegurar la calidad del software, ya que todavía es posible escribir software defectuoso en Java. Sin embargo, sí se eliminan ciertos tipos de errores de programación. Permite además una verificación exhaustiva en tiempo de compilación (37).

Seguro: es una característica muy importante en java ya que se han implementado barreras de seguridad en el lenguaje y en el sistema de ejecución de tiempo real (31).

Para llevar a cabo el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, es objeto de la investigación, escoger el lenguaje de programación. Para ello se escoge como lenguaje de programación al lenguaje Java, por las ventajas y potencialidades que el mismo posee en el desarrollo de aplicaciones web en conjunto con el framework de desarrollo spring, permitiendo obtener productos de excelente calidad y en un corto período de tiempo.

1.9. Frameworks

Un framework es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En términos informales, un framework se puede considerar como una

aplicación genérica incompleta y configurable, a la que se le pueden añadir elementos para construir una aplicación concreta (38). Los objetivos principales que persigue un framework son acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. La Plataforma Java 2 Enterprise Edición (J2EE) cuenta con diferentes marcos de trabajo para desarrollar aplicaciones web, como son Struts, Java Server Faces (JSF) y Spring Framework.

1.9.1. Struts

Es un marco de trabajo para aplicaciones J2EE que implementa el patrón arquitectónico Modelo Vista Controlador (MVC). Fue uno de los primeros framework MVC de desarrollo el cual es de código abierto y está liberado bajo la licencia Apache 2.0. Proporciona un conjunto de etiquetas que cooperan con el controlador para ser usadas en la capa de visualización, en la que puede usarse tecnologías como JSP[6], FreeMarker [7]y otras. Puede integrarse con otros marcos de trabajo como Hibernate y Spring. Presenta desventajas de fallos de diseño. De una u otra manera los creadores de Spring se han valido de la experiencia adquirida al utilizar Struts, para no cometer los mismos errores al diseñar Spring MVC (39).

1.9.2. Spring

Spring es un framework de código abierto encaminado al desarrollo de aplicaciones para la plataforma Java. Cuenta con un conjunto de librerías en las que se pueden escoger aquellas que faciliten el desarrollo de una aplicación. Entre sus posibilidades más potentes está su contenedor de Inversión de Control, también llamado Inyección de Dependencias, el cual es una técnica alternativa a las clásicas búsquedas de recursos vía JNDI[8]. Spring permite configurar las clases en un archivo XML y definir en él las dependencias. De esta forma la aplicación se vuelve modular y a la vez no adquiere dependencias con Spring (39).

Este framework posibilita que los diferentes componentes que forman una aplicación trabajen entre sí; pero no establece importantes dependencias consigo mismo. Sería posible retirarlo sin prácticamente

[6] Java Server Pages (JSP) es una tecnología para crear páginas web dinámicas basadas en HTML, XML entre otros tipos de documentos.

[7] Es un motor de plantillas basado en Java que utiliza el patrón MVC

[8] Java Naming and Directory Interface (Interfaz de Nombrado y Directorio Java) es una Interfaz de Programación de Aplicaciones (API) de Java para servicios de directorio.

cambiar líneas de código. Lo único necesario es, lógicamente, añadir la funcionalidad que provee, ya sea con otro framework similar o mediante el código de la aplicación.

En la presente investigación se utilizará Spring, ya que implementa toda su estructura mediante interfaces. Además, todas las partes del framework son configurables vía plug-ins en la interfaz. Spring provee clases concretas como opción de implementación y garantiza una seguridad al software.

1.10. Servidor de Aplicaciones Tomcat

Las funciones de todos los servidores de aplicaciones son similares pero no brindan sus servicios con la misma velocidad, seguridad y confiabilidad.

Tomcat es un contenedor de Servlets con un entorno JSP. Este puede funcionar como servidor web por sí mismo. Es usado mundialmente como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Es un servidor Web es desarrollado actualmente por la Apache Software Foundation (40). Como los servidores en los cuales estará desplegado finalmente el sistema, son de pocas prestaciones, se selecciona el Tomcat como servidor de aplicaciones, debido a que el mismo necesita poco potencial de hardware (41).

1.11. Conclusiones del capítulo

El estudio del estado del arte sobre los sistemas de gestión de metadatos permitió determinar sus características y tendencias actuales. Además contribuyo a seleccionar como metodología de desarrollo Open Up reflejándose la relación entre las características de la solución y la metodología utilizada. Así como las herramientas a utilizar entre las que se encuentra Visual Paradigm en su versión 8.0 como herramienta de modelado, PostgreSQL en su versión 9.1 como sistema gestor de base de datos, utilizando el lenguaje de programación Java en su versión 6.0 y apoyándose a través del IDE de desarrollo Eclipse en su versión 3.3 con el framework Spring MVC para implementación de la seguridad.

Capítulo 2: Análisis y Diseño

Introducción

En el presente capítulo se abordarán aspectos referentes al levantamiento de requisitos, con el fin de obtener los requisitos funcionales y no funcionales del sistema a implementar. Se elaboran diagramas de caso de uso (CU) del sistema, así como sus correspondientes descripciones. Además, se define la arquitectura del sistema, se realiza el modelado de diagramas de clases del diseño, los diagramas de interacción y diagrama de despliegue para el diseño seleccionado.

2.1. Análisis y diseño

Para la realización del análisis del sistema se efectuaron entrevistas con el cliente, para conocer las necesidades del mismo. Se definieron los objetivos que persigue la organización y se obtuvo una visión del sistema a desarrollar. Además, se precisaron los requisitos para llevar a cabo el modelado del diagrama de caso de uso del sistema, así como una descripción detallada de los CU.

2.2. Modelo dominio

El modelo de dominio se utiliza para descomponer el problema en conceptos u objetos individuales. Para el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, un paso fundamental fue descomponer el problema como se muestra en la Figura 5.

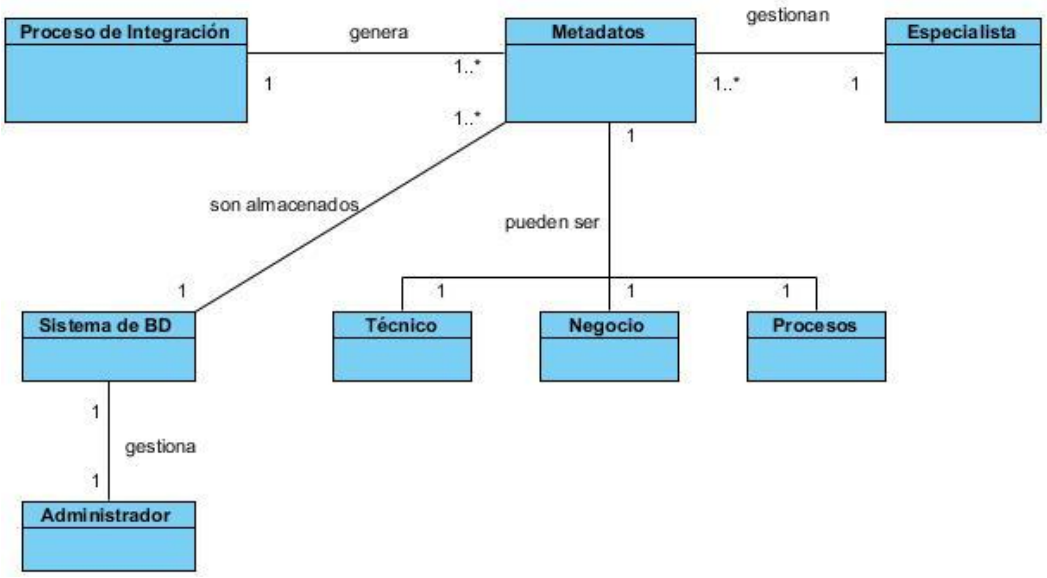


Figura 5: Diagrama de Modelo de dominio

Para una mejor comprensión, a continuación se describen cada uno de los conceptos que interactúan en el diagrama mostrado.

- ✓ **Administrador:** es el rol que representa a la persona capacitada para administrar los usuarios del sistema.
- ✓ **Especialista**: es el rol que representa a la persona con conocimientos básicos en el área de Almacenes de Datos, capacitada para interactuar con el sistema.
- ✓ **Metadatos:** son datos estructurados sobre la información, o expresado de forma más simple, datos sobre datos.
- ✓ **Metadatos técnicos:** representan el diseño del esquema de un almacén de datos en un repositorio, el cual puede ser utilizado para generar un script que construya las tablas del almacén.
- ✓ **Metadatos de negocio:** brindan imágenes del ambiente de trabajo a los usuarios finales, así como los resultados del proceso de integración de datos en función de las métricas y requerimientos del negocio.

- ✓ **Metadatos de proceso:** es la presentación de las estadísticas sobre los resultados de la ejecución del propio proceso de integración de datos, incluyendo medidas tales como filas cargadas con éxito, filas rechazadas, y la cantidad de tiempo de carga.
- ✓ **Sistema de Base de Datos:** es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos.

2.3. Especificación de requisitos del software

La base de comunicación entre los clientes y el equipo de desarrollo reside en los requisitos de software. Estos se clasifican en dos tipos, requisitos funcionales y requisitos no funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) de un sistema describen su funcionalidad y los servicios que de él se esperan. Expresan una especificación detallada de las responsabilidades del sistema en cuestión y permiten determinar, de una manera clara, lo que debe hacer el sistema. Se realizaron reuniones con el cliente para conocer sus necesidades, capturándose los siguientes requisitos funcionales:

- ✓ **RF 1:** Autenticar Usuario
- ✓ **RF 2:** Gestionar Usuario

  RF 2.1: Insertar Usuario

  RF 2.2: Modificar Usuario.

  RF 2.3: Eliminar Usuario.
- ✓ **RF 3:** Administrar selección de componentes para ser desplegados.

  RF 3.1: Seleccionar componentes para su despliegue.

  RF 3.2: Visualizar la selección de componentes para su despliegue.

- ✓ **RF 4:** Gestionar metadatos.

  RF 4.1: Insertar metadatos.

  RF 5.2: Modificar metadatos.

  RF 4.3: Eliminar metadatos.

- ✓ **RF 5:** Restringir conexiones al sistema.

  RF 5.1: Restringir conexiones por rango de IP.

- ✓ **RF 6:** Visualizar reportes.

  RF 6.1: Visualizar gráfica de resultados estadísticos.

  RF 6.2: Visualizar conceptos de una fuente de datos.

  RF 6.3: Visualizar variables de una fuente de datos.

  RF 6.4: Visualizar estadísticas de calidad dado una fuente de datos o una variable.

  RF 6.5: Visualizar registro del estado de la última carga realizada.

  RF 6.6: Visualizar los registros de una carga de datos realizada específicamente.

  RF 6.7: Visualizar diccionario de correspondencia de una fuente, concepto o variable determinada.

- ✓ **RF 7:** Visualizar Información de trazas.

RF 8: Cargar información a partir de varios formatos.

RF 8.1: Cargar información a partir de documentos con extensiones .xls.

2.3.2 Patrones arquitectónicos

Un avance importante dentro de la construcción del software ha sido el desarrollo de la arquitectura de software, que permite representar la estructura de un sistema, a un nivel mayor que el dado por la programación, o incluso por el diseño. La arquitectura de software define la estructura de un sistema. Esta estructura se constituye de componentes, módulos o piezas de código que nacen de la noción de abstracción, cumpliendo funciones específicas e interactuando entre sí con un comportamiento definido. Los componentes se organizan de acuerdo a ciertos criterios, que representan decisiones de diseño. En los últimos años la arquitectura de software se ha denominado patrón arquitectónico (44).

Entre los patrones arquitectónicos se encuentra el patrón **Modelo Vista Controlador (MVC)** que trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Para aplicaciones J2EE, la arquitectura MVC satisface las necesidades del sistema. El patrón indica que se deben establecer 3 componentes o capas en la arquitectura, y que cada una de estas, sólo se comunica con la adyacente (Ver, Figura 6). Además un modelo puede tener diversas vistas, cada una con su correspondiente controlador (44).

Vista: contiene las clases dedicadas puramente a la creación de objetos de presentación. Para trabajar con las vistas se empleó el uso de plantillas.

Controlador: procesa las peticiones que vienen desde la capa de modelo, en la cual está encapsulada la lógica del negocio.

Modelo: representa específicamente el dominio de la información sobre la cual funciona el sistema. El modelo encapsula los datos y las funcionalidades.

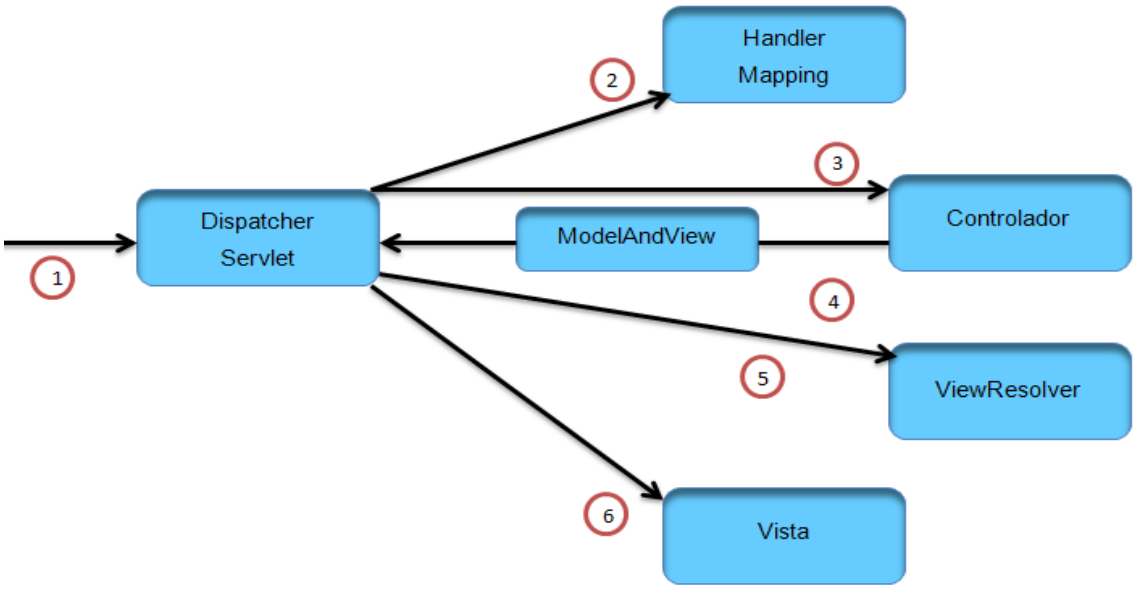


Figura 6: Patrón arquitectónico Modelo Vista Controlador

2.3.3 Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Dentro de sus clasificaciones se encuentran las siguientes:

Usabilidad

Las funcionalidades del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos brindarán la posibilidad de poder gestionar los metadatos.

- ✓ Se debe tener un conocimiento básico sobre el proceso de integración de datos para así poder gestionar los metadatos y poder entender los resultados de la aplicación.

Portabilidad:

El sistema será multiplataforma, razón por la cual podrá ser utilizado en cualquier sistema operativo y arquitectura de hardware.

Soporte

El sistema de gestión brindará al usuario final un manual de usuario que indica cómo proceder para la utilización de las funcionalidades brindadas y cómo conducirse en el entorno del sistema, para que el equipo de desarrollo pueda darle mantenimiento, en el menor tiempo posible a la aplicación.

Software

En los ordenadores de los usuarios debe estar instalado:

- ✓ Máquina Virtual de Java en su versión 6.0.
- ✓ Navegador web con complementos para visualizar applets[9] de Java.

En el servidor de servicios de aplicaciones se deben instalar:

- ✓ Servidor de aplicaciones Apache Tomcat en su versión 7.0.27.
- ✓ Gestor de base de datos PostgreSQL en su versión 9.1.

Hardware

Para el desarrollo y puesta en práctica del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos se requieren de una PC cliente, servidor Web y servidor de base de datos, los cuales deben de cumplir como mínimo las siguientes especificaciones:

El servidor Web:

- ✓ Mínimo un procesador Pentium 3 o superior.
- ✓ Mínimo 512 GB de memoria RAM.
- ✓ Mínimo 4 GB de espacio disponible en el disco duro.

El servidor de base de datos:

- ✓ Mínimo un procesador Pentium 3 o superior.

[9] Componentes implementados en Java que pueden mostrarse a través de una página HTML.

- ✓ Mínimo 512 MB de memoria RAM.
- ✓ Mínimo 2 GB de espacio de espacio disponible en el disco duro.

Seguridad

Autenticación

El sistema debe garantizar que las contraseñas cumplan con los requerimientos de complejidad definidos a continuación:

- ✓ La longitud mínima de las contraseñas debe ser de 8 caracteres.
- ✓ Las contraseñas deben de contener combinaciones de letras minúsculas, mayúsculas, números y caracteres especiales.
- ✓ El sistema debe controlar el historial de las contraseñas con vista a que el usuario no repita contraseñas usadas anteriormente que incluye las 5 últimas ocurrencias.
- ✓ El usuario deberá cambiar la contraseña la primera vez que se autentique.
- ✓ Las contraseñas pueden ser cambiadas por los usuarios del sistema.

Autorización

Para la autorización de los usuarios al sistema se utilizarán los mecanismos de manejo de sesiones, lo que implicará:

- ✓ Las cuentas de usuario no serán compartidas, una cuenta de usuario puede tener una sola sesión activa en el servidor.
- ✓ Las sesiones expirarán dado un tiempo que debe ser configurable mediante la aplicación que no excederá los 5 minutos.

Alarma

El sistema implementará un mecanismo de alarmas para informar a los administradores ante los usos indebidos de la aplicación, la misma será configurable con la siguiente información:

- ✓ Número definido de intentos de inicio de sesión con contraseñas incorrectas.
- ✓ Repetición de usuario inexistente desde una misma dirección IP.
- ✓ Intento de inicio de sesión con usuario que posee sesión activa en el servidor.

Confidencialidad e integridad de los datos.

- ✓ El sistema empleará protocolos seguros en la transmisión de las credenciales de autenticación entre los clientes y el servidor.
- ✓ Las acciones de la aplicación sobre la base de datos constarán con mínimo privilegio.
- ✓ El sistema garantizará los mecanismos para proteger las cadenas de conexión y las contraseñas para conectarse a la base de datos.
- ✓ La información almacenada en los metadatos de conexión se almacenará de forma cifrada.

2.4. Diagrama de caso de uso del sistema.

Los CU describen las interacciones entre uno o más usuarios y el sistema, con el fin de proporcionar un resultado de valor observable para el actor. El diagrama de CU del sistema se puede apreciar en la Figura 7.

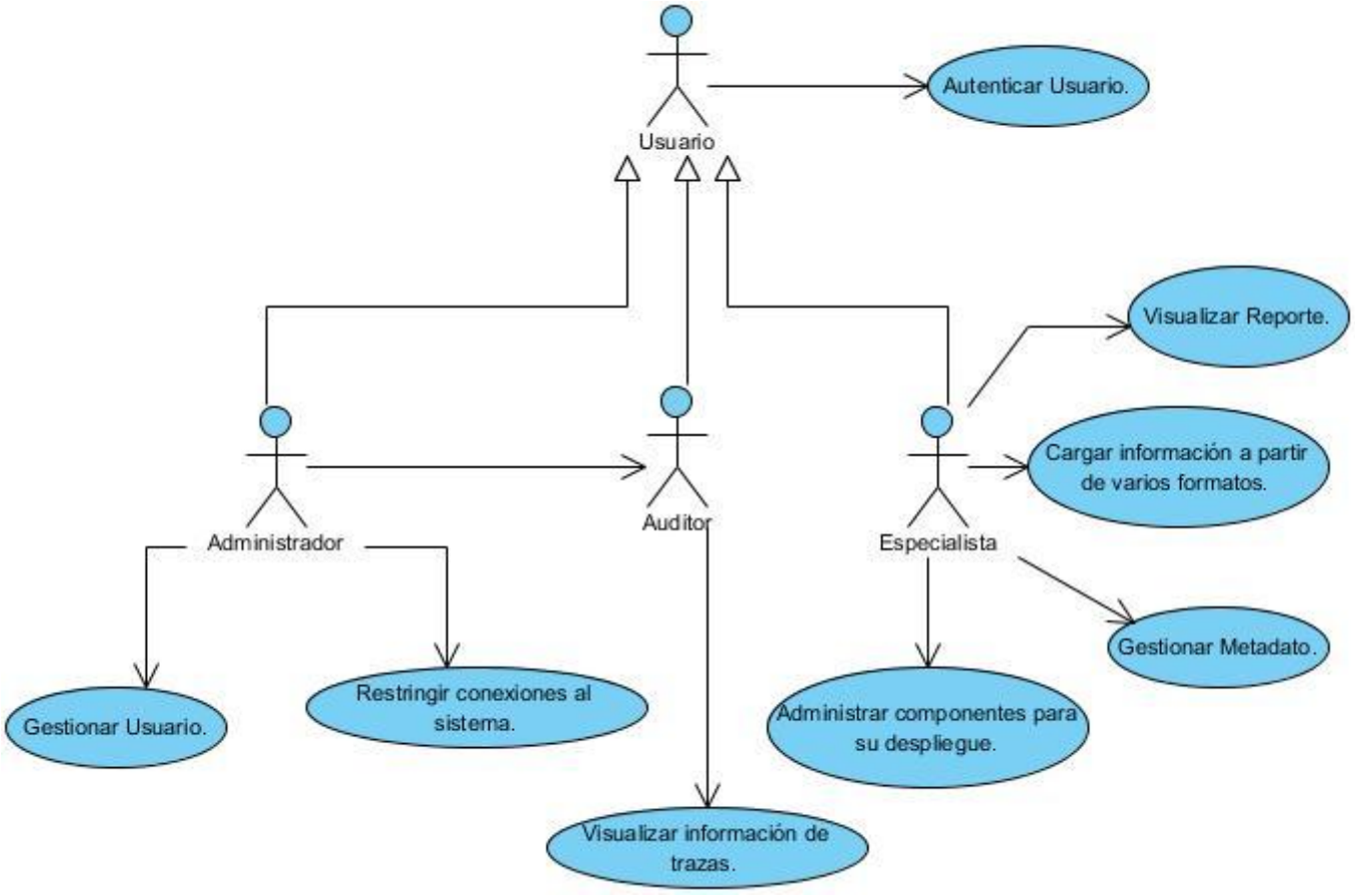


Figura 7: Diagrama de caso de uso del sistema

2.5. Especificación de caso de uso.

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario, o con otro sistema, para conseguir un objetivo específico (42). A continuación, se describen los casos de uso más significativos del sistema "*Gestionar metadato"*.

Tabla 1: Descripción del CU Gestionar Metadato

Objetivo	Gestionar Metadato.	
Actores	Especialista: (Inicia) Gestionar Metadato.	
Resumen	El CU inicia cuando el actor desea insertar, eliminar o modificar un metadato en el sistema. Se insertan los datos de un nuevo metadato al sistema, se elimina uno ya existente o se modifica en dependencia de la acción seleccionada. El CU finaliza una vez que el actor haya insertado, eliminado o modificado algún metadato en el sistema.	
Complejidad	Baja	
Prioridad	Media	
Precondiciones	El actor tiene que estar autenticado.	
Pos - condiciones	El metadato queda insertado, eliminado o modificado.	
Flujo de eventos		
Flujo básico Gestionar Usuario.		
	Actor	**Sistema**
1.	Si el actor realiza una de las siguientes acciones: a) Insertar metadato. Ir a la Sección: **"Insertar Metadato".** b) Eliminar metadato. Ir a la Sección: **"Eliminar Metadato".** c) Modificar metadato. Ir a la Sección: **"Modificar** Metadato**".**	
2.		Muestra la interfaz correspondiente a la acción que va a realizar el actor.
Sección1: "Insertar Metadato"		

Flujo básico Gestionar Metadato.		
	Actor	**Sistema**
1.		El sistema muestra un formulario para poder insertar el tipo de metadato seleccionado.
2.	2.1 El especialista introduce los datos del nuevo metadato.	
3.		3.1 El sistema verifica que los campos del formulario insertar metadato estén llenos. 3.2 El sistema valida los datos insertados y si son correctos, introduce los datos del metadato en la base de datos, finalizando así el CU.
Flujos alternos		
3.1 Existen campos vacíos		
		Se muestra un mensaje de error, para que se llenen los campos y regresa a la acción 2.1.
3.2 Datos incorrectos		
		Se muestra un mensaje de error: los datos insertados no son los correctos y regresa a la acción 2.1
Sección2: "Eliminar Metadato"		
Flujo básico Gestionar Metadato.		
	Actor	**Sistema**
4.		El sistema muestra los datos del tipo de metadato seleccionado.

5.	5.1 El especialista escoge el metadato que se desea eliminar.	
6.		El sistema le envía un mensaje de confirmación para eliminar el metadato.
7.	7.1 El especialista confirma o cancela el metadato a eliminar.	
8.		El sistema elimina el metadato de la base de datos, finalizando así el caso de uso.
Flujos alternos		
7.1 Cancelar opción		
		El sistema muestra el formulario eliminar metadato y regresa a la acción 5.1
Sección3: "Modificar Metadato"		
Flujo básico Gestionar Metadato		
	Actor	**Sistema**
9.	9.1 El especialista escoge el metadato que desea modificar.	
10.		El sistema muestra un formulario con los datos del metadato seleccionado para que el mismo sea modificado.
11.	11.1 El especialista modifica los datos del metadato seleccionado.	
12.		12.1 El sistema verifica que los campos del formulario modificar metadato estén llenos. 12.3 El sistema valida los datos insertados y si son correctos,

		introduce los datos del metadato en la base de datos, finalizando así el CU.
Flujos alternos		
12.1 Existen campos vacío		
		Se muestra un mensaje de error, para que se llenen los campos y regresa a la acción 11.1.
12.2 Datos incorrectos		
		Se muestra un mensaje de error: los datos insertados no son los correctos y regresa a la acción 11.1
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica

2.6. Arquitectura de software del sistema

La arquitectura de software define la estructura de un sistema. Esta estructura se constituye de componentes, módulos o piezas de código que nacen de la noción de abstracción, cumpliendo funciones específicas e interactuando entre sí con un comportamiento definido. Los componentes se organizan de acuerdo a ciertos criterios, que representan decisiones de diseño. En los últimos años la arquitectura de software se ha debatido en dos corrientes fundamentales: los patrones de diseño y los patrones arquitectónicos (43).

2.6.1 Patrones de diseño

Al momento de elaborar los diagramas de clases del diseño es importante asignar correctamente las responsabilidades. Para ello, una buena práctica es auxiliarse de los patrones de diseño, entre los que se pueden mencionar los patrones GRASP, más conocidos como patrones de asignación de responsabilidades. Los patrones GRASP describen los principios fundamentales de la asignación de

responsabilidades a objetos, expresados en forma de patrones (44). A continuación se realiza un breve resumen de los patrones de diseño.

Creador: identifica quién debe ser el responsable de la creación o instancia de nuevos objetos o clases. Brinda un soporte al bajo acoplamiento. En el sistema a desarrollar, se utilizó este patrón en el diseño de las clases que se encuentran en el paquete de objeto de acceso a datos. Como por ejemplo la clase ServicioImplementado.java es la encargada hacer una instancia del objeto DAO (Ver, Figura 8).

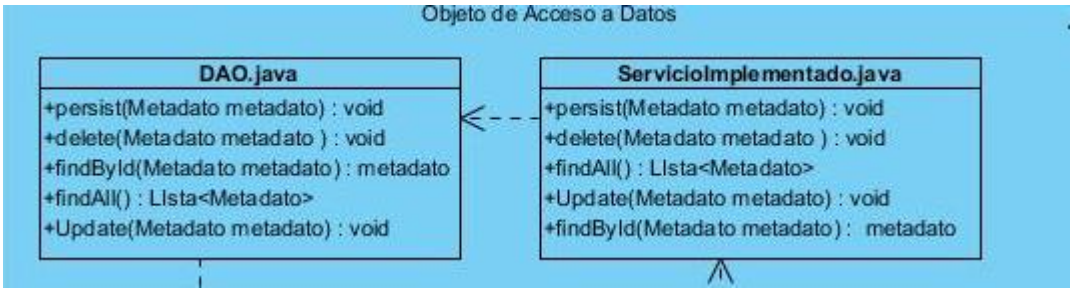


Figura 8: Patrón GRASP Creador

Alta Cohesión: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. En el sistema a desarrollar las clases que se encuentran en el diagrama de clases del diseño, son un ejemplo de la aplicación de este patrón. En el diseño de la solución, se encuentran varias clases SP, donde cada una de ellas es responsable de controlar la configuración de toda la información que gestiona una determinada clase, evidenciando de este modo que las clases SP representan el patrón (Ver, figura 8).

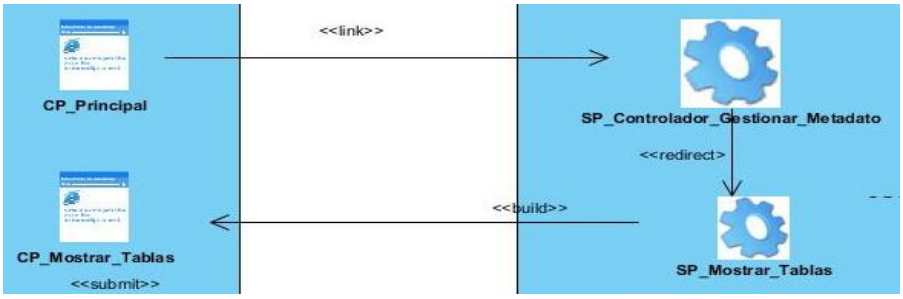


Figura 9: Patrón GRASP Alta Cohesión.

2.7. Diagramas de clases del diseño

El diagrama de clases describe, la estructura del sistema mostrando sus clases, atributos y relaciones entre ellos. Un diagrama de clases del diseño a diferencia del modelo conceptual, contiene las definiciones de las entidades del software en vez de conceptos del mundo real.

En el diagrama de clases del diseño para el CU "*Gestionar metadato"* (Ver, Figura 10),se identifican 3 paquetes: vista, controlador y objeto de acceso a datos, los que corresponden a las capas del patrón arquitectónico MVC. Además, se puede apreciar la distribución de los objetos dentro de estos paquetes, separando el sistema en varias capas. Se muestran las páginas clientes (CP) en el paquete vista, las páginas servidoras (SP) en el paquete controlador y el paquete encargado de gestionar la base de datos en el paquete de objeto de acceso a datos.

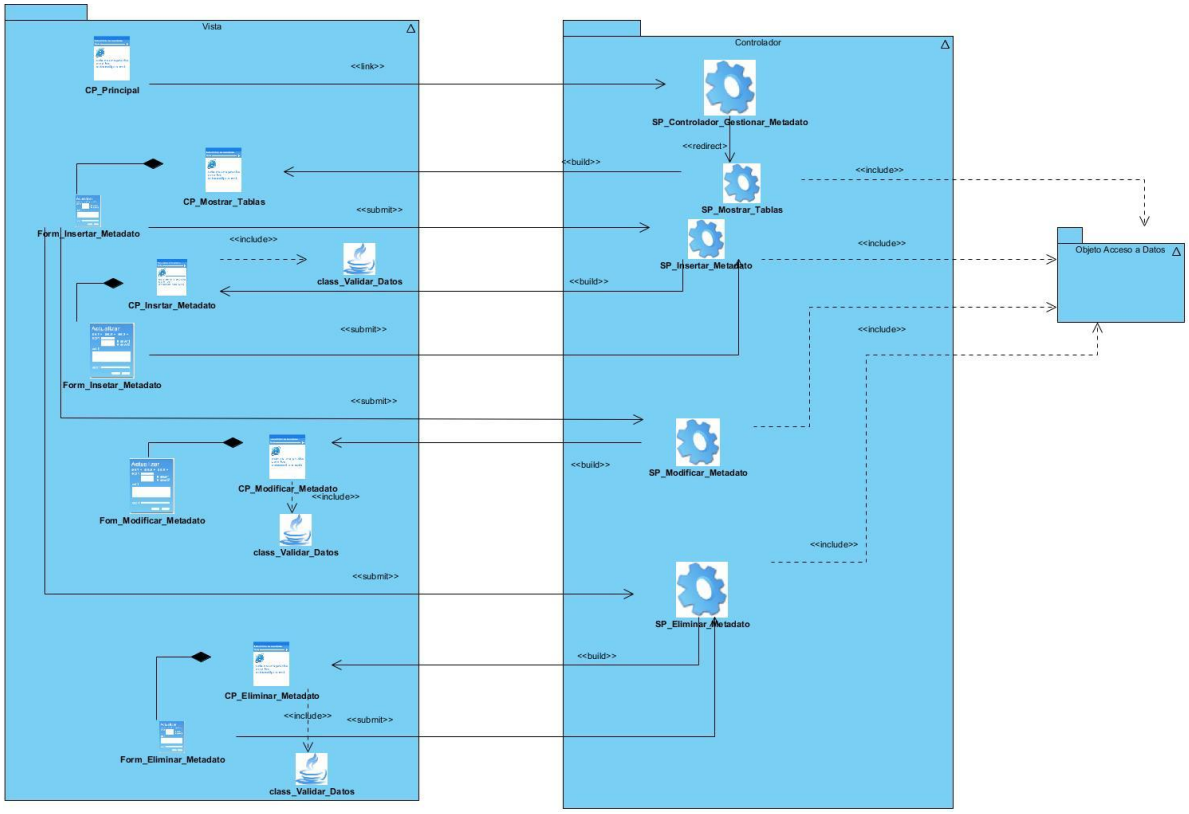


Figura 10: Diagrama de clases del diseño para CU "Gestionar metadato"

En el diagrama de clases del diseño correspondiente al paquete de objeto de acceso a datos (Ver, Figura 11) se define la clase DAO.java que es la interfaz donde se encuentran todos los métodos necesarios para gestionar la base de datos. La clase DAOImplementado.java es la que implementa las funciones que permiten acceder a la base de datos, Servicio.java y ServicioImplementado.java son capas que existen entre los DAO y los controladores.

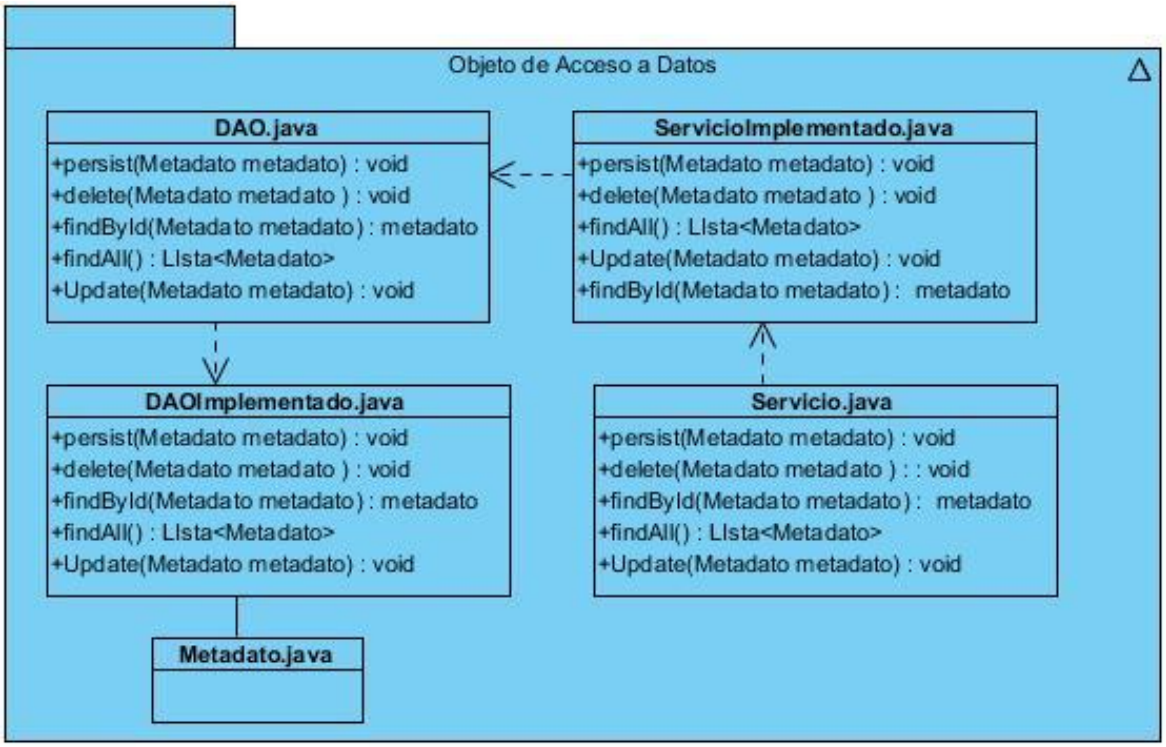


Figura 11: Diagrama de clases del diseño para el paquete Objeto de Acceso a Datos

2.8. Diseño de la base de dato modelo físico

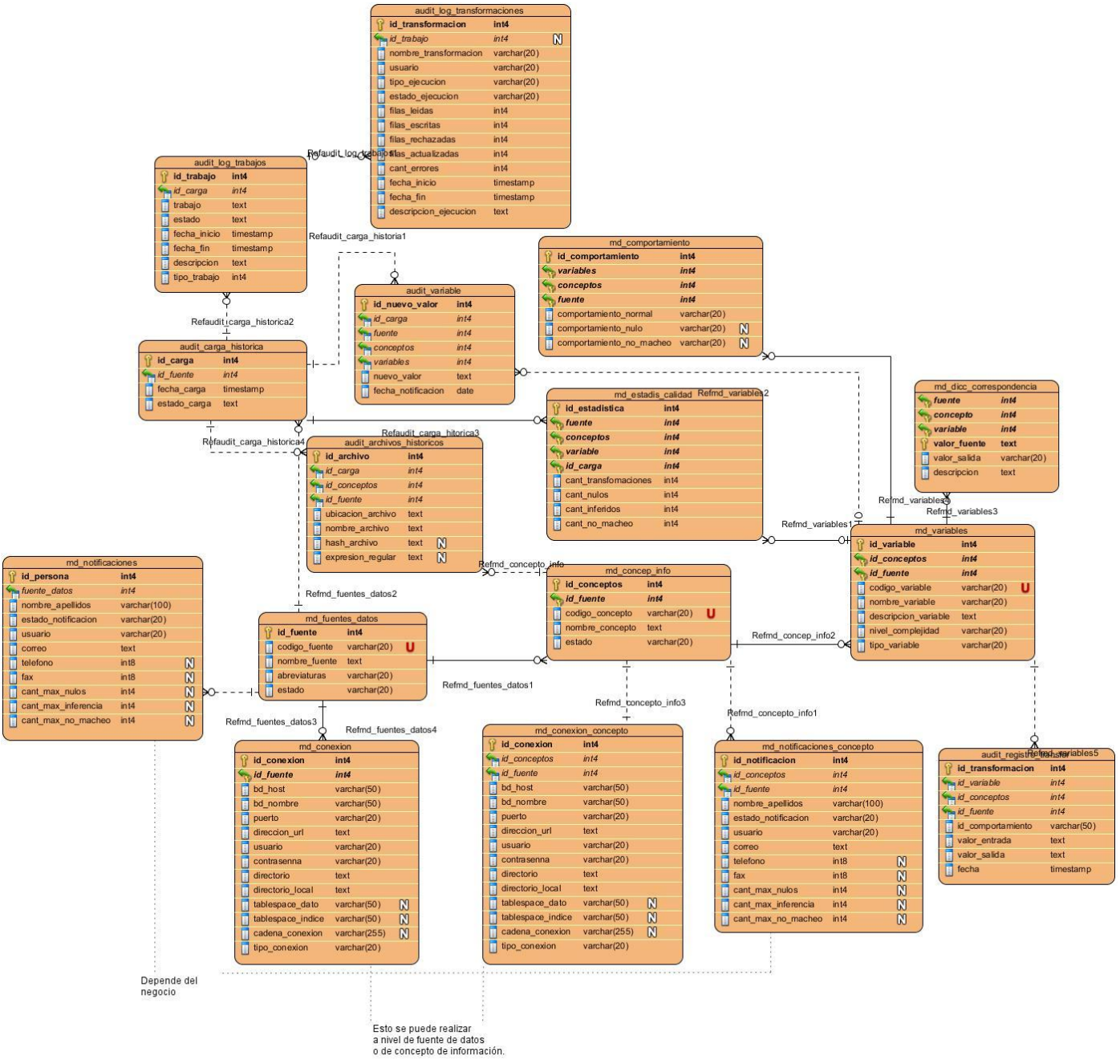


Figura 12: Diagrama Entidad-Relación

Diagrama de interacción

Los diagramas de interacción describen secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Estos diagramas, muestran el flujo de control a través de muchos objetos.

Los diagramas de interacción se exhiben en 2 diagramas centrados en distintos aspectos pero complementarios.

Los diagramas de secuencia para el diseño del sistema propuesto muestran la secuencia de mensajes entre objetos durante un escenario concreto (Ver, figura 13 y Anexo 1).

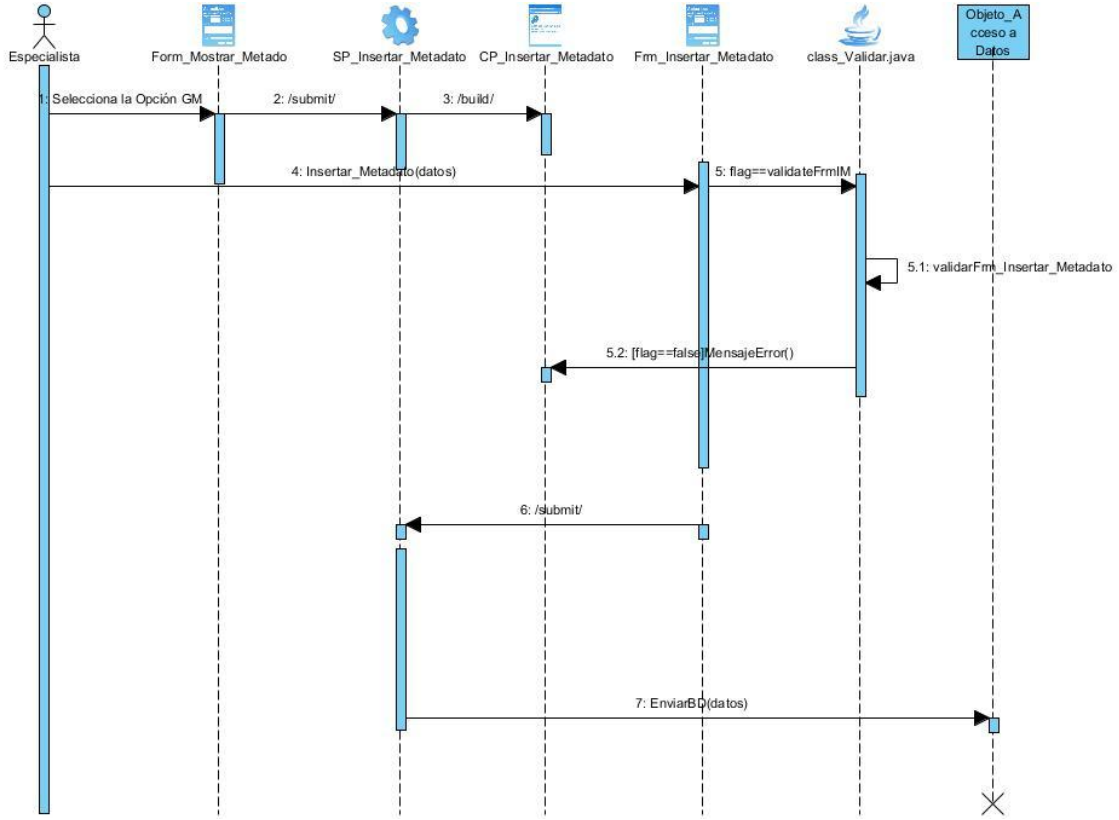


Figura 13: Diagrama de secuencia "Insertar Metadato"

2.9. Diagrama de despliegue

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellas. Los diagramas de despliegue representan los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red y conexiones (45).

En la Figura 14: Diagrama de despliegue se muestra el diagrama de despliegue del sistema. Para el sistema a realizar se requiere de al menos una PC cliente donde el especialista accede al Servidor Web mediante el protocolo HTTPS y se conecta al nodo Servidor DB, que es donde se encuentra la base de datos del sistema. A esta base de datos se accede a través del componente acceso a datos que se encuentra en el nodo usando el protocolo TCP/IP.

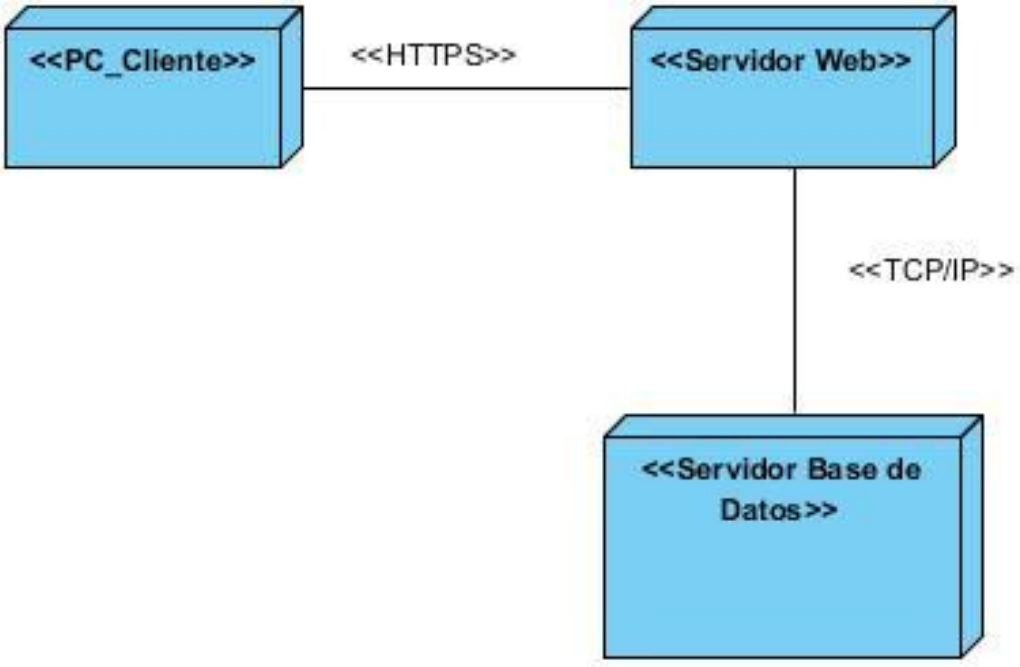


Figura 14: Diagrama de despliegue

2.10. Conclusiones del capítulo

El levantamiento de requisitos realizado permitió conocer las necesidades del negocio, para de esta forma elaborar el diagrama de CU del sistema describiéndose solo el CU más significativo “Gestionar Metadatos”.

Además la aplicación de los patrones MVC y GRASP permitió guiar la elaboración de las clases del diseño en el sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos, que estos a su vez fueron el punto de partida para realizar los diagramas de interacción, secuencia y despliegue.

Capítulo 3: Implementación y Prueba

Introducción

Una vez concluido el modelo del diseño, se dispone de la información necesaria para proceder a la construcción del sistema. En el presente capítulo se construye el modelo de implementación correspondiente al sistema de gestión para la auditoría y control del proceso de integración de datos, desglosándolo en los diagramas de componentes de los requisitos más importantes. De la misma forma se determinan los niveles de pruebas y métodos de prueba, con el fin de facilitar la implementación del sistema.

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes, ficheros de código fuente, ejecutables, datos, entre otros. Describe también cómo se organizan los componentes de acuerdo a los mecanismos de estructuración disponibles en el entorno de implementación y lenguajes de implementación empleados, y cómo dependen los componentes unos de otros (46). Esta descripción es de utilidad a la hora de implementar el sistema, debido a que facilita la organización del trabajo y lo hace más entendible para los desarrolladores.

3.1.1. Diagrama de componentes

Los diagramas de componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas entre otros. El diagrama de componentes muestra un conjunto de ficheros relacionados entre sí, para lograr una completa funcionalidad del sistema. En la Figura 15 se muestra el diagrama de componentes del CU “*Gestionar metadato*”.

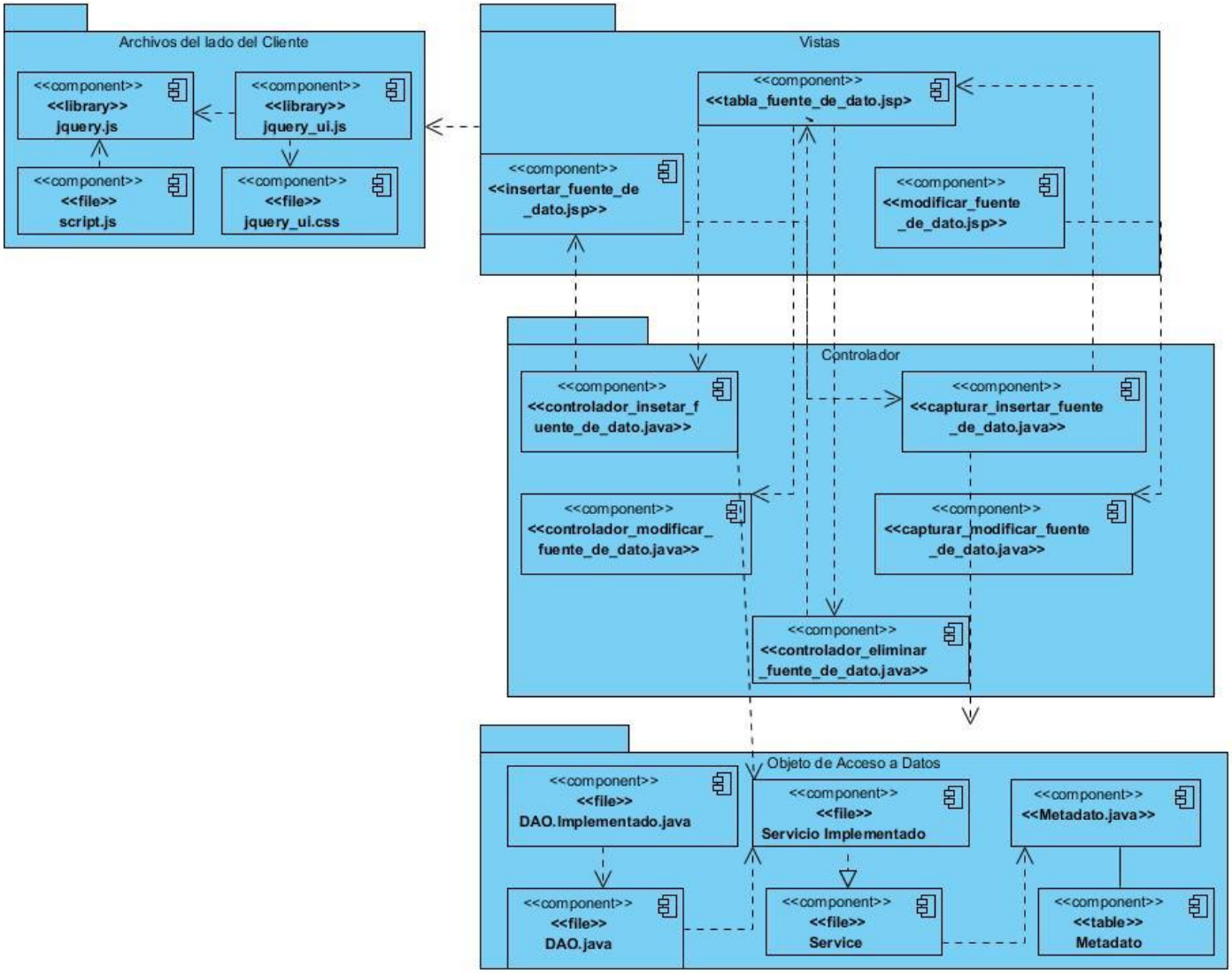


Figura 15: Diagrama de componentes del CU "Gestionar metadato"

Tabla 2: Descripción de los componentes del CU "Gestionar metadato"

Componente	Propósito
Archivos del lado del Cliente	Contiene los estilos CSS y los archivos JavaScript que se ejecutan en el navegador del cliente.
DAO.java	Clase que declara los métodos necesarios para acceder a la base de datos.
DAOImplementado.java	Clase que implementa las funciones que permiten acceder a la base de datos.
Metadato	Tabla de la base de datos que representa los metadatos del sistema.
controlador_insertar_fuente_de_dato.java	Controlador encargado de mostrar la vista, insertar fuente de dato y enviar los objetos necesarios para llevar a cabo esta funcionalidad.
controlador_eliminar_fuente_de_dato	Controlador encargado de obtener el id de la fuente de dato a eliminar, eliminarlo y mostrar la vista actualizada.
capturar_insertar_fuente_de_dato	Es el encargado de recibir los datos entrados por el usuario, insertarlo en la BD y mostrar la tabla actualizada.
controlador_modificar_fuente_de_dato	Controlador encargado de mostrar la vista de modificar fuente de dato y enviar los datos necesarios para llevar a cabo esta funcionalidad.
capturar_modificar_fuente_de_dato	Es el encargado de recibir los datos introducidos por los usuarios, modificarlos en la BD y mostrar la tabla actualizada.

3.2. Mapa de navegación

El siguiente mapa de navegación (Ver, Figura 16) se elaboró con el objetivo de una mejor comprensión de la estructura del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

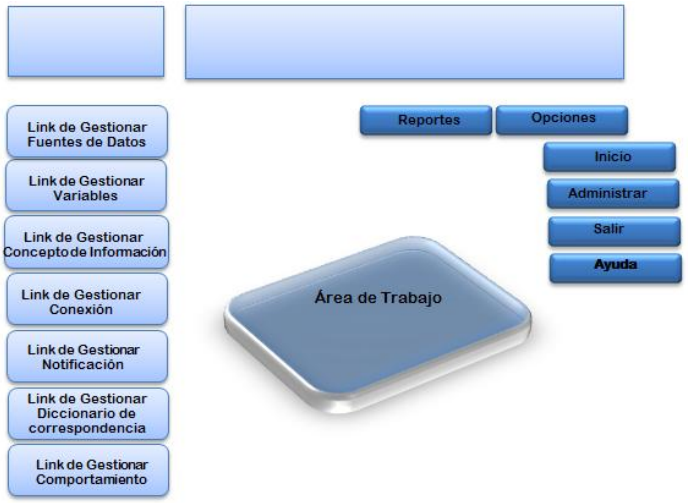


Figura 16: Mapa de navegación del sistema de gestión para la auditoría y control del proceso de integración de datos

3.3. Prueba de calidad de software

La prueba de software es un proceso que se desarrolla para identificar posibles fallos de implementación, calidad, o usabilidad de un software. Todo sistema debe ser revisado con el objetivo de establecer el nivel de calidad requerido. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas (47).

Las pruebas definen el grado de aceptación del sistema que pueden tener los clientes del mismo, e incluso determinar si lo aceptan o no. Para ser más eficaces, las pruebas deberían ser realizadas independientes al equipo de desarrollo. Esto garantiza pruebas con alta probabilidad de encontrar errores, que es el objetivo principal. Son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se definen los siguientes tipos de niveles de prueba:

3.3.1. Prueba de unitarias

Esta prueba es una forma de probar el correcto funcionamiento de un módulo de código. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera.

3.3.2. Prueba del sistema

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo y estas ocurren tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

3.3.3. Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

- ✓ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada sistema, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplica al software, logrando como resultado que disminuya el número de errores existentes en los sistemas (42).

La **prueba del camino básico** es una técnica de prueba de caja blanca que determina la complejidad ciclomática de una porción de código. La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa (47). Cuando se usa el camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar.

3.3.4. Casos de pruebas caja negra

La prueba de caja negra se centra principalmente en los requisitos funcionales del software. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien, se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los que identifican los métodos de caja blanca (47). La prueba de caja negra intenta identificar errores de las siguientes categorías:

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

Existen varias técnicas de pruebas de caja negra dentro de las que se puede mencionar la de partición de equivalencia. La misma divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta técnica se dirige a la definición de los casos de usos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

A modo de conclusión se arrojaron los siguientes pasos para realizar las pruebas al sistema de gestión para la auditoría y control del proceso de integración de datos (Ver, Figura 17).

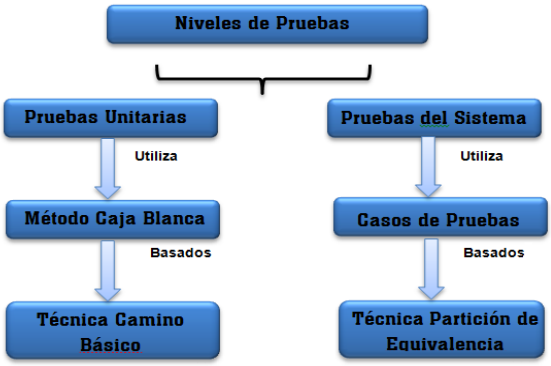


Figura 17: Proceso de prueba para el sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

3.4. Aplicación de pruebas de caja blanca

Como se mencionó anteriormente, uno de los métodos de prueba de caja blanca es la del camino básico. Se determinó aplicar este método a la función InsertarFuenteDato.java.

Para la función “OnSubmit” se identificaron los bloques de ejecución y se enumeraron Figura 18. Se obtuvieron 6 bloques y se determinó el camino básico ilustrado en la Figura 19, identificando en cada sentencia condicional un nodo predicado del cual se derivan más de un camino a seguir, tal es el caso de los nodos 2 y 3.

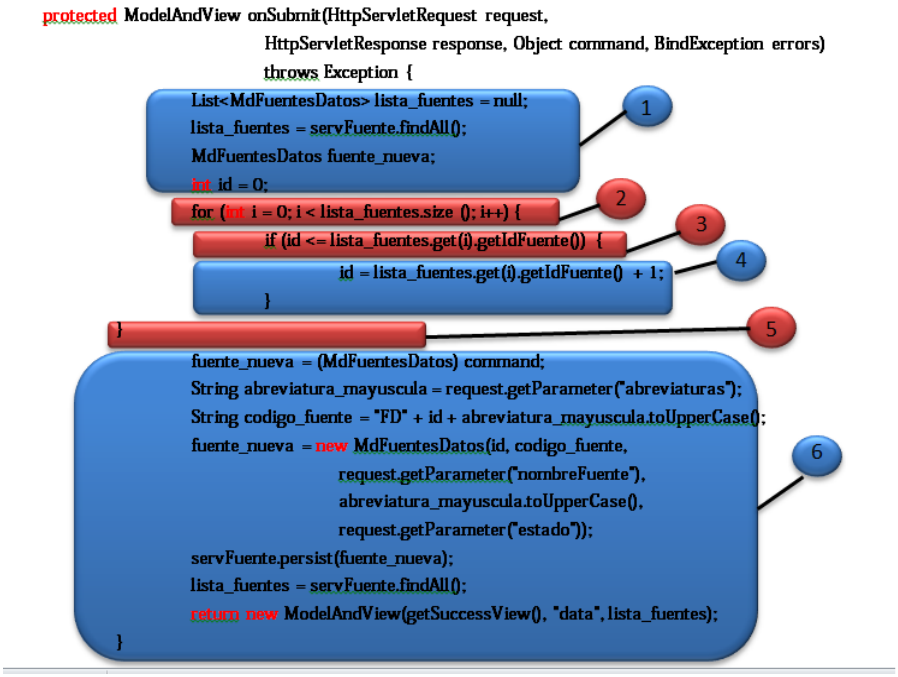


Figura 18: Función que permite insertar una fuente de datos a la BD

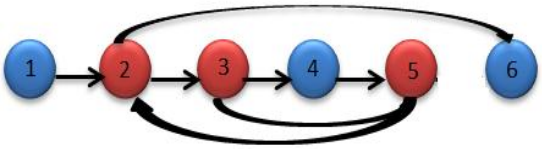


Figura 19: Camino básico de la función "OnSubmit"

Con el camino básico determinada, se aplica una de las tres formas para calcular la complejidad dicromática, se utilizó la fórmula V (G)= A-N+2, para la cual se obtuvo 7 aristas y 6 nodos, por lo tanto: V (G)= 7-6+2, quedando V (G)=3. De la misma forma se pueden comprobar que las otras variantes de calcular la complejidad ciclomática arriban al mismo resultado. La complejidad ciclomática indica los posibles casos de ejecución para la función, lo que es muy útil a la hora de diseñar los casos de prueba de caja negra.

3.4.1. Casos de pruebas caja negra

Para aplicar las pruebas al sistema se determinaron los casos de pruebas, un conjunto de entradas, condiciones de ejecución y resultados esperados, con tal de verificar el cumplimiento de un objetivo en particular.

Antes de ejecutar un caso de prueba es necesario definir una serie de variables que serán utilizadas, para el CU: "*Gestionar Metadato"*, las variables usadas para dicho caso de prueba se resumen en la Tabla 3.

Tabla 3: Variables para el caso de prueba "Insertar Fuente de Datos"

No	Nombre del campo.	Clasificación	Valor Nulo	Descripción
1	Id_ fuente	Campo numérico	No	Número mayor que 0.
2	código_fuente	Campo de texto	No	fd+id+abreviatura cantidad de caracteres menor que 20.
3	nombre_fuente	Campo de texto	No	Secuencia de caracteres alfanumérico
4	abreviatura	Campo de texto	No	Secuencia de caracteres alfanumérico cantidad de caracteres menor que 20

Tabla 4: Caso de prueba 1: Insertar Fuente de Datos

Escenario	Descripción	id	código_fuente	nombre_fuente	abreviatura	estado	Respuesta del sistema	Flujo Central
EC1.Insertar fuentes de datos.	Se insertan los campos correctamente con los	V [10]	V	V	V	V	El sistema muestra una tabla con los valores	1. Autenticarse 2. Gestionar fuentes de

[10] Entrada válida

	nuevos valores entrados.						actualizados.	datos. 3. Insertar fuentes de datos. 4. Aceptar.
EC1.Insertar campos vacíos.	Se desea insertar una fuente de dato dejando campos vacíos.	I[11] V V V V	V I V V V	V V I V V	V V V I V	V V V V I	Muestra un mensaje de error en el campo vacío.	1. Autenticarse 2. Gestionar fuentes de datos. 3. Insertar fuentes de datos. 4. Aceptar.

El caso de prueba de la Tabla 4 fue aplicado al CU “Insertar Fuente de Datos”. De la misma forma se definió un escenario de prueba para el CU “Insertar Concepto de Información”, las variables usadas para dicho caso de prueba se resumen en la

[11] Entrada inválida

Tabla 5

Tabla 5: Variables para el caso de prueba "Insertar Concepto de Información"

No	Nombre del campo.	Clasificación	Valor Nulo	Descripción
1	Id_ concepto	Campo numérico	No	Valor que identifica al concepto y no puede estar repetido.
2	Id_ fuente	Campo numérico	No	Valor que identifica a la fuente y no puede estar repetido.
3	código_ concepto	Campo de texto	No	ci+id+abreviatura cantidad de caracteres menor que 20.
4	nombre_ concepto	Campo de texto	No	Secuencia de caracteres alfanumérico
5	abreviatura	Campo de texto	No	Secuencia de caracteres alfanumérico cantidad de caracteres menor que 20
6	estado	Campo de texto	NO	Activado \ Desactivado

Tabla 6: Caso de prueba 2: Insertar Concepto de Información

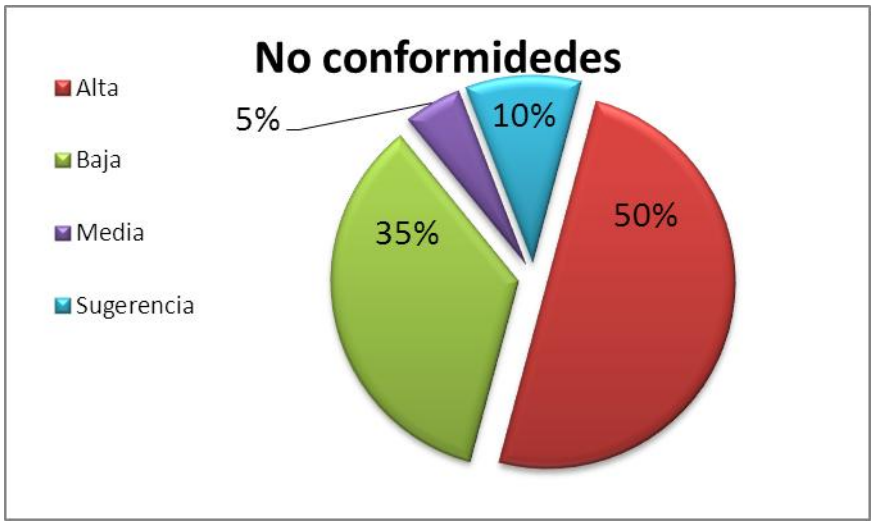
Escenario	Descripción	id	código _fuente	nombre _fuente	abreviatura	estado	Respuesta del sistema	Flujo Central
EC1.Insertar concepto de información.	Se insertan los campos correctamente con los nuevos valores entrados.	V	V	V	V	V	El sistema muestra una tabla con los valores actualizados.	1. Autenticarse. 2. Gestionar concepto de información. 3. Insertar concepto de información. 4. Aceptar.

EC1.Insertar campos vacíos.	Se desea insertar un concepto de información dejando campos vacíos.	I V V V V	V I V V V	V V I V V	V V V I V	V V V V I	Muestra un mensaje de error en el campo vacío.	1.Autenticarse 2. Gestionar fuentes de datos. 3. Insertar concepto de información. 4. Aceptar.

La aplicación de los casos de prueba permitió verificar el cumplimiento de los requisitos funcionales del sistema, de igual forma corroboró la robustez del sistema en el manejo de errores y situaciones anormales.

3.5. Resultado de las pruebas

De las pruebas realizadas al sistema, se detectaron en la 1ra iteración 20 no conformidades de ellas el 50% son altas, 35% bajas, 5% son medias y 10% sugerencias (Ver Gráfica 2).



Gráfica 2: No conformidades

Después de corregir las no conformidades detectadas, estas fueron nuevamente revisadas por un especialista del departamento de Almacenes de Datos del centro DATEC y no se detectaron nuevas no conformidades por lo que se aprobó la liberación del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.

3.6. Conclusiones del capítulo

La realización de los diagramas de componentes guiaron las tareas de implementación correspondiente al sistema de gestión para la auditoría y control del proceso de integración de datos, permitiendo dar cumplimiento a los requisitos definidos en el negocio. Además la aplicación de las pruebas unitarias y pruebas del sistema a través de los métodos caja blanca y los casos de prueba de caja negra permitieron probar el funcionamiento del sistema implementado, arrojando no conformidades que fueron solventadas mediantes las iteraciones de pruebas realizadas al software que contribuyeron al correcto funcionamiento del sistema.

Conclusiones generales

Una vez concluida la investigación se puede afirmar que se desarrolló el sistema de gestión para la auditoría y control del proceso de integración de datos. Se concluye que:

- ✓ El estudio bibliográfico relacionado con los sistemas de gestión de metadatos, permitió la selección de las herramientas, tecnologías y metodología a utilizar en el desarrollo del sistema de gestión de metadatos para la auditoría y control del proceso de integración de datos.
- ✓ A través de la definición de los requisitos del sistema se puedo conocer las necesidades del negocio, lo que permitió elaborar el diagrama de CU del sistema, que dio paso a las clases del diseño guiado por el patrón arquitectónico MVC, obteniendo un sistema separado en capas, altamente escalable y menos vulnerable al cambio.
- ✓ A partir de los diagramas de componentes se guiaron las tareas de implementación permitiendo dar cumplimiento a los requisitos definidos en el negocio. Además la realización de las pruebas unitarias y del sistema permitieron probar el funcionamiento de la aplicación implementada, obteniéndose un producto capaz de darle solución a las necesidades del cliente.

Recomendaciones

Con el desarrollo de esta investigación se cumplieron los objetivos propuestos, sin embargo con vistas a lograr una mayor efectividad en el funcionamiento del sistema se recomienda:

- ✓ Incorporar nuevos metadatos técnicos y de proceso que permitan aumentar la capacidad de análisis del sistema.
- ✓ Integrar el sistema de gestión de metadatos con otras aplicaciones desarrolladas dentro del departamento de Almacenes de Datos en el centro DATEC para utilizar en el desarrollo de soluciones de almacenes de datos
- ✓ Agregar nuevos reportes que permitan elevar las potencialidades del sistema y contribuir a la toma de decisiones sobre el proceso de integración de datos.

Referencias bibliográficas

1. **SYNTEL.** Integration, EAI vs. ETL: Drawing Boundaries for Data. *S y n t e l.* [En línea] S y n t e l, 2011. [Citado el: 10 de 5 de 2012.] http://w w w . s y n t e l i n c . c o m. ISBN 1-59904-365-3.

2. *EII - ETL - EAI What, Why, and How.* **Corporation, IBM.** 2, Taiwan : IBM, 28 de 10 de 2005, Vol. I. ISBN 8448132149.

3. *CWM, A Data Quality Metamodel Extension to.* **Pedro Gomes, José Farinha, Maria José Trigueiros.** Portugal : Wiley Publishing, Inc, 2007. Vol. 67. SBN: 0-764-57923-1.

4. **Ralph Kimball, Joe Caserta.** *The data Warahause ETL Tookit.* Canada : Wiley Publishing, Inc, 2004. SBN: 0-764-57923-1.

5. **Kimball, Ralph.** Sub Sistemas de ETL. *BIllage.* [En línea] BIllage, 17 de 9 de 2009. [Citado el: 5 de 10 de 2012.] http://bi.social.uoc.edu/smc/blog/34-subsistemas-etl-de-kimball. SBN: 0-764-57923-1.

6. **Inmon, W. H.** *Building the Data Warehouse,Fourth Edition.* Canada : Fourth Edition, 2005. Vol. I. ISBN-13: 978-0-7645-9944-6.

7. **Rangel, Wilfredo.** *ETL_CRM_BPM_ERP_BI.* Caracas : Centro CISI , 2009. ISSN 1316-6239 .

8. *PROPUESTA DE UN REPOSITORIO DE METADATOS PARA LA GESTIÓN Y AUDITORÍA DE PROCESOS DE INTEGRACIÓN DE DATOS.* **Doris Medina Mustelier, Yuneimy Téllez Pérez, Yonelbys Iznaga González, Mabel Medina Rodríguez.** 1, La Habana : s.n., 2013, Vol. I.

9. **Inmon, William H.** *Building the Data Warehouse.* Canada : Fourth Edition, 2005. ISSN 1316-6239 .

10. **Reeves, Laura, y otros, y otros.** *The data warehause lifecycle toolkit(Expert Methods for Designing, Developing and Depliying Data warehauses) .* United States of America : Integrated Book Technology, 2007. ISBN 1-59904-364-5.

11. ORACLE DATABASE 11G ORACLE WAREHOUSE BUILDER DATA QUALITY OPTION. *oracle warehouse builder.* [En línea] 14 de 1 de 2006. [Citado el: 4 de 11 de 2012.] http://www.oracle.com/technology/products/warehouse/11gr1/datasheets/warehouse-builder-11g-data-quality-datasheet.pdf.

12. *Ingeniería de Software un enfoque práctico. McGraw-Hill Companies S.l.* **Pressman, Roger S.** s.l. : Quinta edición, 2002, Vol. III. ISBN: 8448132149.

13. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico Parte1.* Madrid : Quinta Edicción, 2005. ISBN-10: 0-7645-9944-5.

14. **Sanderson, S.** Sitio de Metodología de Desarrollo. *Ayuda extendida de OpenUp.* [En línea] Sanderson, S, 20 de 10 de 2010. [Citado el: 14 de 10 de 2012.] http://10.34.20.5:5800/OpenUP..

15. **Management, Business.** Business Management. [En línea] 4 de 8 de 2010. [Citado el: 15 de 10 de 2012.] http://www.busmanagement.com//article/Open-Source--the-Next-Wave-for-Project-and-Portfolio-Management-Applications/.

16. **Garcilaso, Jordana.** Introducción a Open UP. [En línea] 2008-. [Citado el: 14 de 10 de 2012.] http://www.eclipse.org/epf/general/OpenUP.pdf.

17. *Herramientas IDE - CASE - Presentation Transcript.* **Valencia, Luis.** 5, Mexico : 2, 2011, Vol. II. ISBN-13: 978-0-7645-9944-6.

18. **Sierra, Daniel.** [En línea] 5 de 4 de 2009. [Citado el: 20 de 11 de 2012.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml..

19. *Entonación de Bases de Datos .* **Mota.** 4, Mexico : s.n., 2005, Vol. I.

20. **Base de Datos.** netpecos. *netpecos.* [En línea] 8 de 6 de 2007. [Citado el: 13 de 13 de 2012.] http://www.netpecos.org/docs/mysql_postgres/x57.html..

21. *The Java EE 5 Tutorial. E.E. U.U.* **Corporation, Oracle.** 2, Lima : Dawn Speth White, 2010, Vol. I. ISBN 0-13-748880-7.

22. **Rig, Krosing H.** Postgresql. *Postgresql.* [En línea] 18 de 7 de 2007. [Citado el: 14 de 12 de 2012.] http://www.computerworld.com.au/article/62894/postgresql_affiliates_org_domain..

23. *PostgreSQL 9. Administration Cookbook.* **Rig, Krosing H.** 2, E.E. U.U : Birmingham Packt, 2009, Vol. I.

24. **En linea.** Informática. [En línea] 15 de 3 de 2006. [Citado el: 15 de 1 de 2013.] http://www.informatica.com/es/products/enterprise-data-integration/.. ISBN.

25. **NetBeans.** NetBeans. *NetBeans.* [En línea] 1 de 9 de 2008. [Citado el: 15 de 1 de 2013.] http://www.netbeans.org/index_es.html..

26. **IBM Corporation.** NetBeans vs Eclipse. [En línea] 09 de 12 de 2003. [Citado el: 15 de 1 de 2013.] http://www.juanjonavarro.com/masquecodigo/netbeans-frente-a-eclipse..

27. **En linea.** caracteristicas-la-plataforma-netbeans. *caracteristicas-la-plataforma-netbeans.* [En línea] 8 de 7 de 2012. [Citado el: 15 de 1 de 2013.] http://netbeansaccesible.blogspot.com/.

28. **Eclipse.** Qué es Eclipse. *Qué es Eclipse.* [En línea] 14 de 9 de 2009. [Citado el: 16 de 1 de 2013.] http://plataformaeclipse.com/faq..

29. **Java.** Plataform Eclipse. *Plataform Eclipse.* [En línea] 4 de 5 de 2009. [Citado el: 16 de 1 de 2013.] http://plataformaeclipse.com/faq.

30. **D, Flanagan.** Java en pocas palabras . [aut. libro] Flanagan D. México. : s.n., 2009.

31. **Java.** Tutorial de Java. Introducción al AWT. *Tutorial de Java. Introducción al AWT.* [En línea] 1 de 3 de 2005. [Citado el: 16 de 1 de 2013.] http://www.ac.uma.es/localres/manuals/JavaTut-Froufe/Cap4/awt.html#Intro.

32. **Calderón , Marcela y Davis , Swing Emilio.** La solución actual de Java. 2009, Vol. I, 2.

33. *Web Development with Java.* **Downey, Tim.** 5, EE UU : s.n., 2009, Vol. III.

34. *Programación Java Server con J2EE.* **S, Allamaraju y C, Beust.** 7, EE UU : s.n., 16 de 9 de 2011, Vol. I.

35. **E., Burns, C, Schalk y N, Griffin.** *JavaServer Faces 2.0:.* Mexico : s.n., 2009.

36. **j, The jQuery Project.** *Query is a new kind of JavaScript Library.* EE UU : s.n., EE UU.

37. **Java.** Que es Java. *Que es Java.* [En línea] Ciberuela, 11 de 5 de 2005. [Citado el: 17 de 1 de 2013.] http://java.ciberaula.com/articulo/que_es_java..

38. **Jesus, Gorostiaga.** Frameworks. [En línea] 3 de 2011. [Citado el: 16 de 1 de 2013.] ttp://es.scribd.com/doc/76348225/Frameworks-Php-y-Codeigniter.. ISBN.

39. *Spring 2.* **Minter, Dave.** New York : Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, , 2008, Vol. II. ISBM.

40. **Tomcat.** Tutorial de programación. [En línea] 1 de 1 de 2006. [Citado el: 16 de 1 de 2013.] http://www.programacion.com/tutorial/tomcatintro/1/#1_intro.. ISBN.

41. **RICH SEZOV, JR.** *Liferay in action. Shelter Island.* Lima : s.n., 20011.

42. **Pressman.** Ingeneria de Software. [aut. libro] Pressman S. *Ingeneria de Software.* Mexico : Quinta Edición, 2000.

43. **C, Larman.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* [ed.] Pablo Eduardo Roig Vkquez. México : David Fayerman Aragón, 1999. ISBN 970-1 7-0261-1 .

44. —. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : David Fayerman Aragón, 2003. ISBN 789-547-247.

45. Ecured. [En línea] [Citado el: 5 de marzo de 2013.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

46. **H.W, Jung, S.G, Kim y C.S., Chung.** *Measuring software product quality.* Mexico : s.n., 2006. ISBN.

47. **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico.* España : edicion de 1996. , 2001.

Bibliografía

1. **SYNTEL.** Integration, EAI vs. ETL: Drawing Boundaries for Data. *S y n t e l.* [En línea] S y n t e l, 2011. [Citado el: 10 de 5 de 2012.] http://w w w . s y n t e l i n c . c o m. ISBN 1-59904-365-3.

2. *EII - ETL - EAI What, Why, and How.* **Corporation, IBM.** 2, Taiwan : IBM, 28 de 10 de 2005, Vol. I. ISBN 8448132149.

3. *CWM, A Data Quality Metamodel Extension to.* **Pedro Gomes, José Farinha, Maria José Trigueiros.** Portugal : Wiley Publishing, Inc, 2007. Vol. 67. SBN: 0-764-57923-1.

4. **Ralph Kimball, Joe Caserta.** *The data Warahause ETL Tookit.* Canada : Wiley Publishing, Inc, 2004. SBN: 0-764-57923-1.

5. **Kimball, Ralph.** Sub Sistemas de ETL. *BIllage.* [En línea] BIllage, 17 de 9 de 2009. [Citado el: 5 de 10 de 2012.] http://bi.social.uoc.edu/smc/blog/34-subsistemas-etl-de-kimball. SBN: 0-764-57923-1.

6. **Inmon, W. H.** *Building the Data Warehouse,Fourth Edition.* Canada : Fourth Edition, 2005. Vol. I. ISBN-13: 978-0-7645-9944-6.

7. **Rangel, Wilfredo.** *ETL_CRM_BPM_ERP_BI.* Caracas : Centro CISI , 2009. ISSN 1316-6239 .

8. *PROPUESTA DE UN REPOSITORIO DE METADATOS PARA LA GESTIÓN Y AUDITORÍA DE PROCESOS DE INTEGRACIÓN DE DATOS.* **Doris Medina Mustelier, Yuneimy Téllez Pérez, Yonelbys Iznaga González, Mabel Medina Rodríguez.** 1, La Habana : s.n., 2013, Vol. I.

9. **Inmon, William H.** *Building the Data Warehouse.* Canada : Fourth Edition, 2005. ISSN 1316-6239 .

10. **Reeves, Laura, y otros, y otros.** *The data warehause lifecycle toolkit(Expert Methods for Designing, Developing and Depliying Data warehauses) .* United States of America : Integrated Book Technology, 2007. ISBN 1-59904-364-5.

11. ORACLE DATABASE 11G ORACLE WAREHOUSE BUILDER DATA QUALITY OPTION. *oracle warehouse builder.* [En línea] 14 de 1 de 2006. [Citado el: 4 de 11 de 2012.] http://www.oracle.com/technology/products/warehouse/11gr1/datasheets/warehouse-builder-11g-data-quality-datasheet.pdf.

12. *Ingeniería de Software un enfoque práctico. McGraw-Hill Companies S.l.* **Pressman, Roger S.** s.l. : Quinta edición, 2002, Vol. III. ISBN: 8448132149.

13. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico Parte1.* Madrid : Quinta Edicción, 2005. ISBN-10: 0-7645-9944-5.

14. **Sanderson, S.** Sitio de Metodología de Desarrollo. *Ayuda extendida de OpenUp.* [En línea] Sanderson, S, 20 de 10 de 2010. [Citado el: 14 de 10 de 2012.] http://10.34.20.5:5800/OpenUP..

15. **Management, Business.** Business Management. [En línea] 4 de 8 de 2010. [Citado el: 15 de 10 de 2012.] http://www.busmanagement.com//article/Open-Source--the-Next-Wave-for-Project-and-Portfolio-Management-Applications/.

16. **Garcilaso, Jordana.** Introducción a Open UP. [En línea] 2008-. [Citado el: 14 de 10 de 2012.] http://www.eclipse.org/epf/general/OpenUP.pdf.

17. *Herramientas IDE - CASE - Presentation Transcript.* **Valencia, Luis.** 5, Mexico : 2, 2011, Vol. II. ISBN-13: 978-0-7645-9944-6.

18. **Sierra, Daniel.** [En línea] 5 de 4 de 2009. [Citado el: 20 de 11 de 2012.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml..

19. *Entonación de Bases de Datos .* **Mota.** 4, Mexico : s.n., 2005, Vol. I.

20. **Base de Datos.** netpecos. *netpecos.* [En línea] 8 de 6 de 2007. [Citado el: 13 de 13 de 2012.] http://www.netpecos.org/docs/mysql_postgres/x57.html..

21. *The Java EE 5 Tutorial. E.E. U.U.* **Corporation, Oracle.** 2, Lima : Dawn Speth White, 2010, Vol. I. ISBN 0-13-748880-7.

22. **Rig, Krosing H.** Postgresql. *Postgresql.* [En línea] 18 de 7 de 2007. [Citado el: 14 de 12 de 2012.] http://www.computerworld.com.au/article/62894/postgresql_affiliates_org_domain..

23. *PostgreSQL 9. Administration Cookbook.* **Rig, Krosing H.** 2, E.E. U.U : Birmingham Packt, 2009, Vol. I.

24. **En linea.** Informática. [En línea] 15 de 3 de 2006. [Citado el: 15 de 1 de 2013.] http://www.informatica.com/es/products/enterprise-data-integration/.. ISBN.

25. **NetBeans.** NetBeans. *NetBeans.* [En línea] 1 de 9 de 2008. [Citado el: 15 de 1 de 2013.] http://www.netbeans.org/index_es.html..

26. **IBM Corporation.** NetBeans vs Eclipse. [En línea] 09 de 12 de 2003. [Citado el: 15 de 1 de 2013.] http://www.juanjonavarro.com/masquecodigo/netbeans-frente-a-eclipse..

27. **En linea.** caracteristicas-la-plataforma-netbeans. *caracteristicas-la-plataforma-netbeans.* [En línea] 8 de 7 de 2012. [Citado el: 15 de 1 de 2013.] http://netbeansaccesible.blogspot.com/.

28. **Eclipse.** Qué es Eclipse. *Qué es Eclipse.* [En línea] 14 de 9 de 2009. [Citado el: 16 de 1 de 2013.] http://plataformaeclipse.com/faq..

29. **Java.** Plataform Eclipse. *Plataform Eclipse.* [En línea] 4 de 5 de 2009. [Citado el: 16 de 1 de 2013.] http://plataformaeclipse.com/faq.

30. **D, Flanagan.** Java en pocas palabras . [aut. libro] Flanagan D. México. : s.n., 2009.

31. **Java.** Tutorial de Java. Introducción al AWT. *Tutorial de Java. Introducción al AWT.* [En línea] 1 de 3 de 2005. [Citado el: 16 de 1 de 2013.] http://www.ac.uma.es/localres/manuals/JavaTut-Froufe/Cap4/awt.html#Intro.

32. **Calderón , Marcela y Davis , Swing Emilio.** La solución actual de Java. 2009, Vol. I, 2.

33. *Web Development with Java.* **Downey, Tim.** 5, EE UU : s.n., 2009, Vol. III.

34. *Programación Java Server con J2EE.* **S, Allamaraju y C, Beust.** 7, EE UU : s.n., 16 de 9 de 2011, Vol. I.

35. **E., Burns, C, Schalk y N, Griffin.** *JavaServer Faces 2.0:.* Mexico : s.n., 2009.

36. **j, The jQuery Project.** *Query is a new kind of JavaScript Library.* EE UU : s.n., EE UU.

37. **Java.** Que es Java. *Que es Java.* [En línea] Ciberuela, 11 de 5 de 2005. [Citado el: 17 de 1 de 2013.] http://java.ciberaula.com/articulo/que_es_java..

38. **Jesus, Gorostiaga.** Frameworks. [En línea] 3 de 2011. [Citado el: 16 de 1 de 2013.] ttp://es.scribd.com/doc/76348225/Frameworks-Php-y-Codeigniter.. ISBN.

39. *Spring 2.* **Minter, Dave.** New York : Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, , 2008, Vol. II. ISBM.

40. **Tomcat.** Tutorial de programación. [En línea] 1 de 1 de 2006. [Citado el: 16 de 1 de 2013.] http://www.programacion.com/tutorial/tomcatintro/1/#1_intro.. ISBN.

41. **RICH SEZOV, JR.** *Liferay in action. Shelter Island.* Lima : s.n., 20011.

42. **Pressman.** Ingeneria de Software. [aut. libro] Pressman S. *Ingeneria de Software.* Mexico : Quinta Edición, 2000.

43. **C, Larman.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* [ed.] Pablo Eduardo Roig Vkquez. México : David Fayerman Aragón, 1999. ISBN 970-1 7-0261-1 .

44. —. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : David Fayerman Aragón, 2003. ISBN 789-547-247.

45. Ecured. [En línea] [Citado el: 5 de marzo de 2013.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

46. **H.W, Jung, S.G, Kim y C.S., Chung.** *Measuring software product quality.* Mexico : s.n., 2006. ISBN.

47. **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico.* España : edicion de 1996. , 2001.

48. **WALLS, CRAIG .** *Spring in Action.* United States of America : Manning Publications , 2008. ISBN 1-933988-13-4.

49. **Medina Mustelier, Doris.** *Técnicas de Extracción, Transformación y Carga de Datos del Sistema de Información Nacional de Seguridad Ciudadana en la Sistema de Información Nacional de Seguridad Ciudadana en la.* Cuba : s.n., 2009.

Anexos

Anexo 1: Diagramas de secuencias

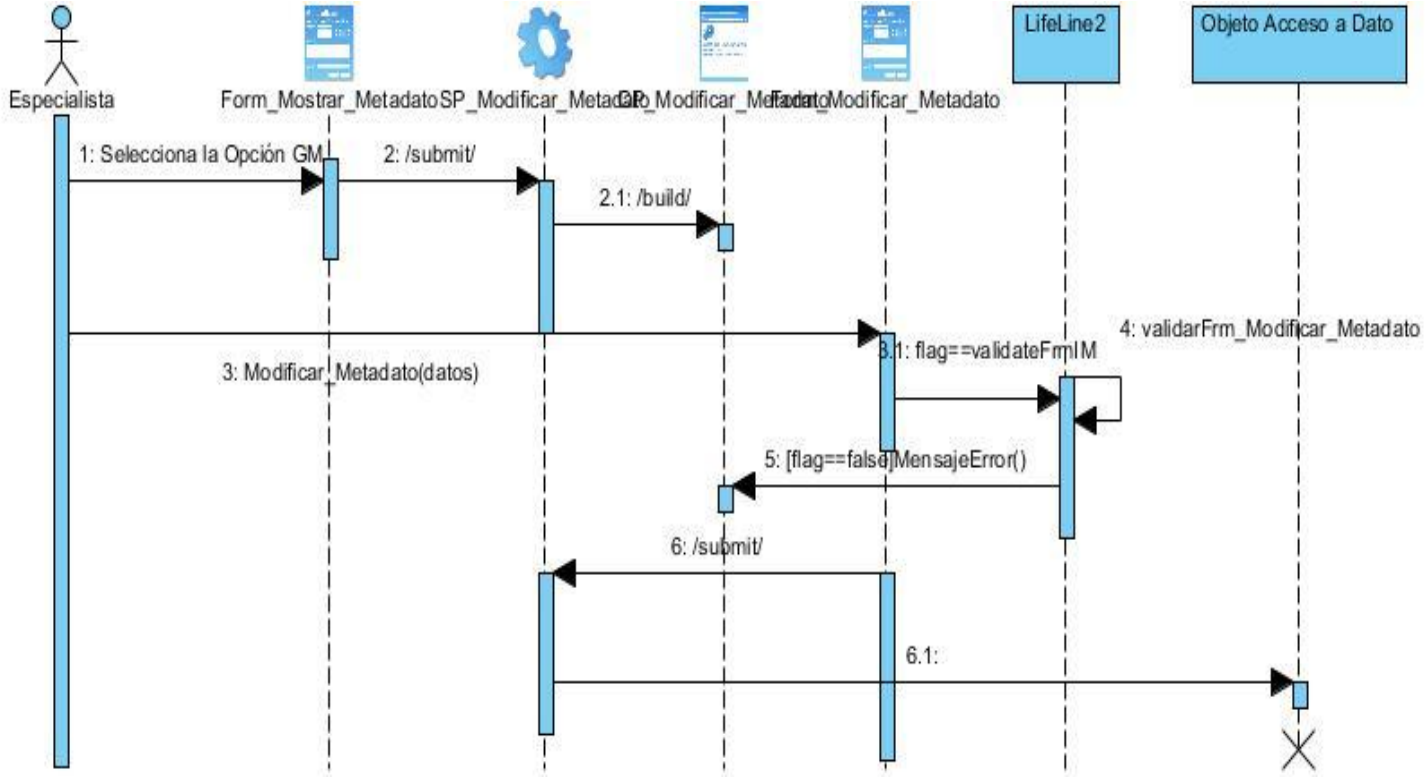


Figura 20: Diagrama de secuencia Modificar Metadato

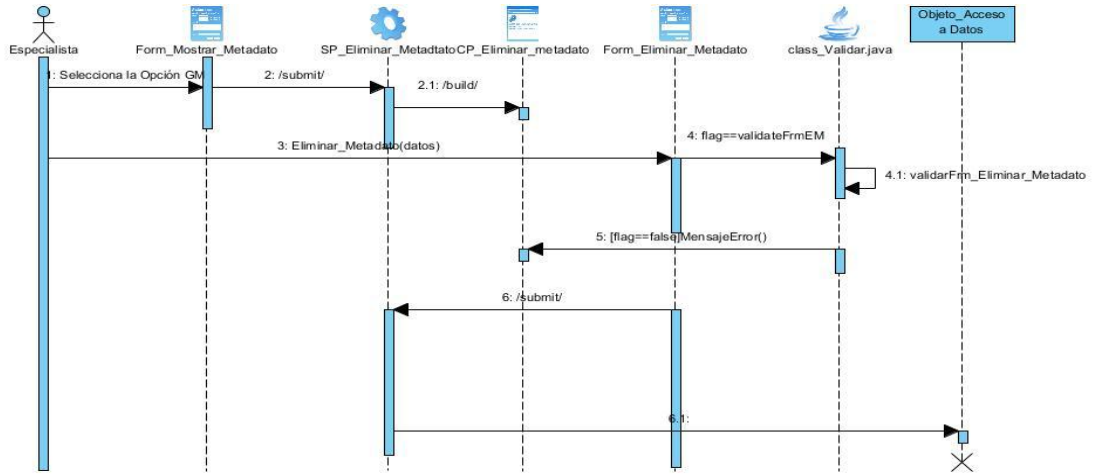


Figura 21: Diagrama de secuencia Eliminar Metadato

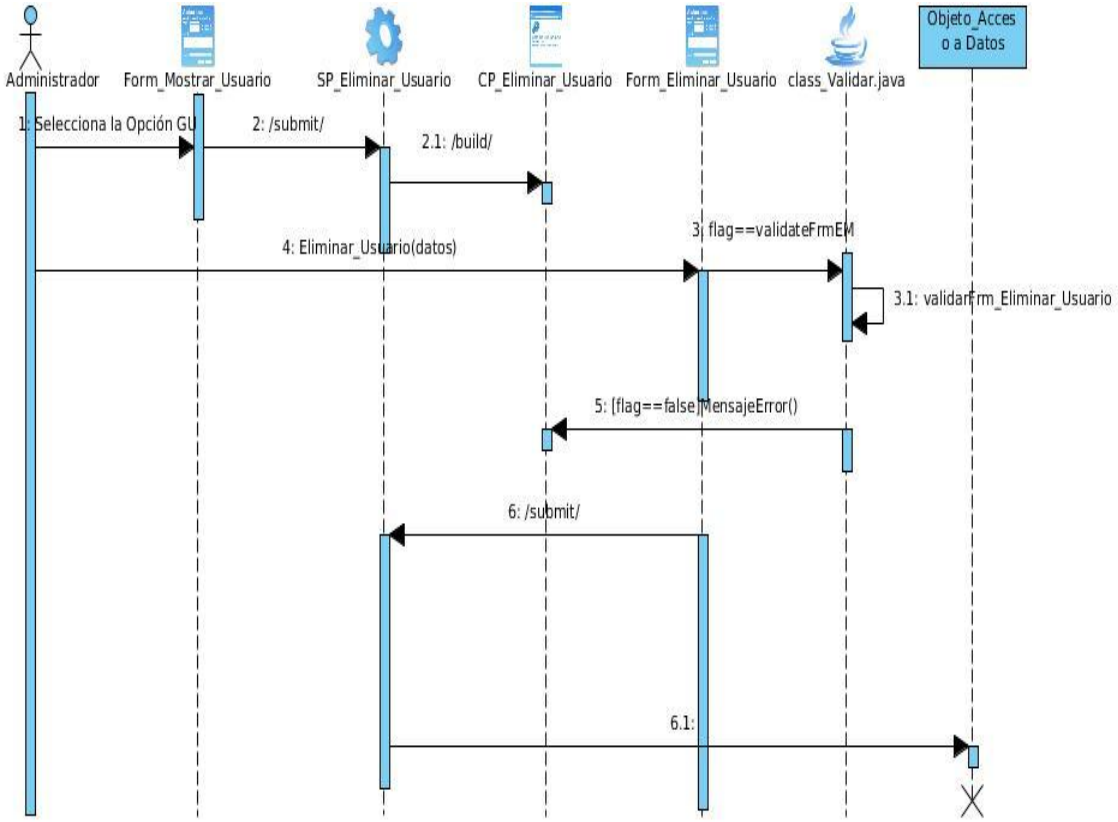


Figura 22: Diagrama de secuencia Eliminar Usuario

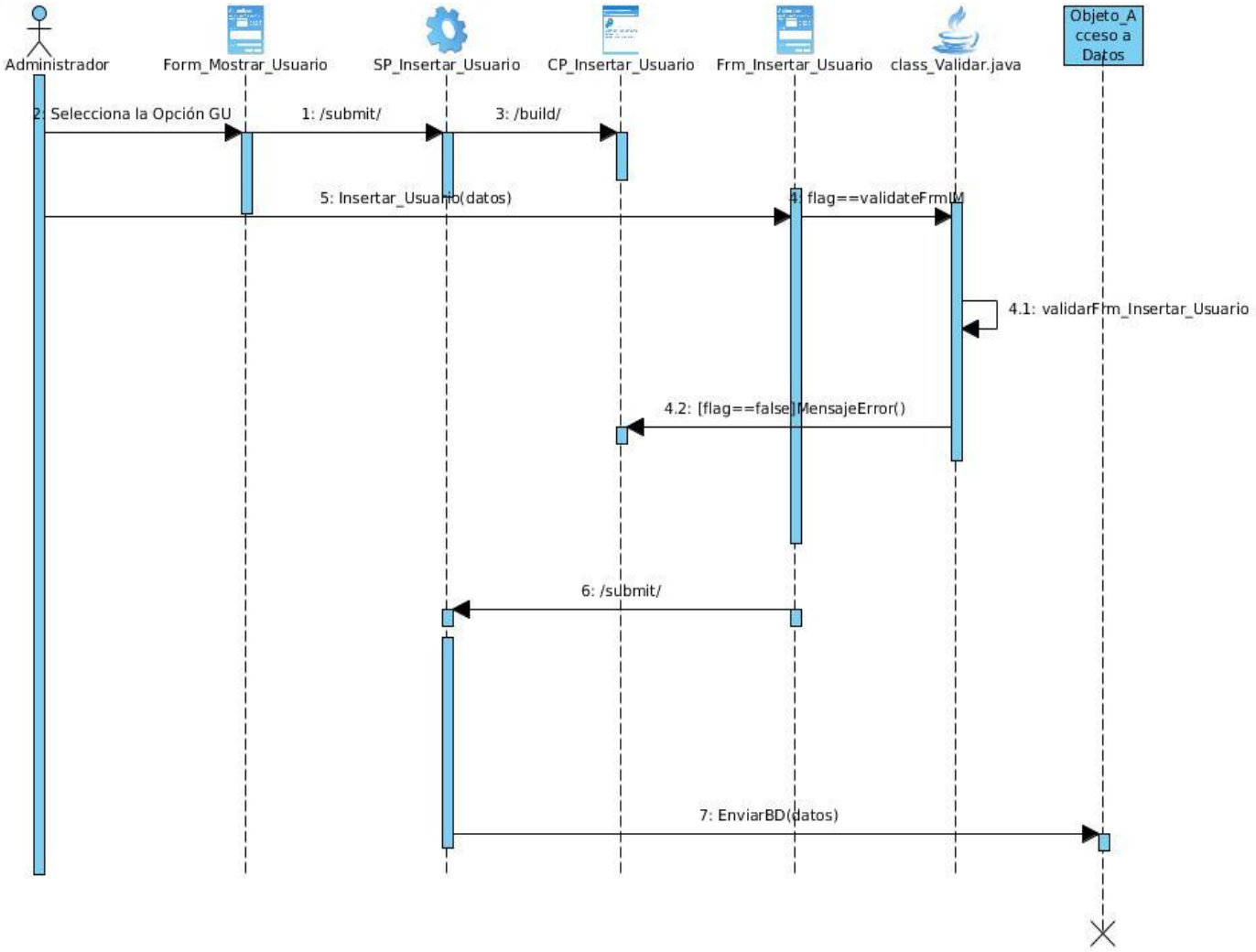


Figura 23: Diagrama de secuencia Insertar Usuario