Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos

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

Autor(es): Javier Ernesto Viguera Olmos
Yohan Carlos Sánchez Hernández
Tutor(es): Ing. Yosbel Rodríguez Rodríguez
Ing. Yonelbys Iznaga González

La Habana, Junio 2013
"Año 55 de la Revolución"

Leonardo da Vinci (Vinci, Toscana, 1452 - Amboise, Turena, 1519). Artista, pensador e investigador italiano que, por su insaciable curiosidad y su genio polifacético, representa el modelo más acabado del hombre del Renacimiento. La obra pictórica de Leonardo da Vinci, la que le ha hecho destacar como un personaje cumbre en la historia del arte, debido a una veintena de cuadros conservados, entre los cuales destacan La Gioconda o Mona Lisa, La Anunciación, La Virgen de las Rocas, La Santa Cena, La Virgen y Santa Ana, La Adoración de los Magos, el Retrato de Ginebra Benzi.

"El buen juicio nace de la buena inteligencia y la buena inteligencia deriva de la razón, sacada de las buenas reglas; y las buenas reglas son hijas de la buena experiencia: madre común de todas las ciencias y las artes".

Leonardo Da Vinci

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 _____.

Yohan Carlos Hernández Sánchez

_____

Firma del Autor

Ing. Yonelbys Iznaga González

_____

Firma del Tutor

Javier Ernesto Viguera Olmos

_____

Firma del Autor

Ing. Yosbel Rodríguez Rodríguez

_____

Firma del Tutor

Ing. Yonelbys Iznaga Gonzáles.
Universidad de las Ciencias Informáticas.
Especialista.
Instructor docente.
e-mail yiznaga@uci.cu

Ing. Yosbel Rodríguez Rodríguez
Universidad de las Ciencias Informáticas.
Especialista.
e-mail yrodriguezro@uci.cu

AGRADECIMIENTOS

Agradecemos a la Revolución Cubana y nuestro Comandante en Jefe Fidel Castro por ser el creador y guía de una institución tan especial como lo es esta universidad. A todos los profesores que han contribuido con nuestra formación profesional. A nuestros tutores por aportar su experiencia durante todo este curso. Infinitos agradecimientos a nuestros familiares que nos han brindado todo su apoyo a lo largo de nuestras vidas, por darnos todo su amor, por estar siempre en los buenos y malos momentos.

No existen palabras que describan lo que sentimos hacia nuestros padres, promotores de nuestras acciones diarias, ejemplos de sacrificio, honestidad, entrega y dignidad. Agradecemos eternamente a ellos, por darnos su confianza y amor, de todo corazón MUCHAS GRACIAS.

Yohan y Javier

DEDICATORIA

Le dedico esta tesis, primero que todo, a todas las personas que me quiere y me admira, que desde el comienzo de mi vida me han dado su apoyo y cariño, especialmente a mis padres Juan Carlos y Mercedes por darme todo el amor del mundo, a mis abuelos Adolfina y Modesto, a María por ser madre y abuela, a mis tíos Pipo, Nine, Viñales, Papito y EL Gordo, por darme confianza, a Lito aunque no esté hoy en día, por darme entusiasmo y motivarme a seleccionar la carrera, a mis primos Ernesto, Yosvany (Gucho), Juan Carlitos, José Carlos, por ser mis hermanos, a mi tía Dinora por brindarme apoyo, ya Marlén por acogerme en su casa como un hijo más. A mi tía Delma del alma que me crió como su hijo y que la quiero mucho.

A mis amigos del barrio, Dalian, Eduardo, Carmen, Leudys, el Flaco, Yuset y Daney que son como mis hermanos, los cuales hemos compartido juntos todos estos años de felicidad y alegría. A mis amigos de siempre Alejandro, Ronniel, Roly, Yasnier, Eduardo y Darien por haber compartido momentos inolvidables en el IPI.

A mi hermano y compañero de tesis Javier, por estar juntos en los momentos buenos y difíciles de la carrera, a mis compañeros de cuarto José Carlos, Keimer, Carlos, Julio. A mis compañeros de grupo, especialmente Odalys, Martha, Ernesto Antonio, Darien, Antonio Miguel, Antonio Garcés, Carlos (el Carly) y Yasmany por ser buenos amigos. A los amigos del edificio. En fin a todas la personas cercanas que de una forma u otra hicieron este sueño realidad.

Principalmente le dedico esta tesis a la persona que más quiero en la vida, y por la cual me ha dado fuerzas para lograr lo que quiero en la vida, a mi hermanita Yumercy, que aunque no esté presente, siempre la llevo en el corazón y nunca la olvidaré.

Yohan

El resultado de estos largos años de estudio se resume en este magnífico logro, por lo que no podría pasar por alto a las personas que han compartido buenos y malos momentos estos 23 años de vida, a mis amigos de la infancia, especialmente Antonio que lo considero mi hermano, a mi compañeros de la universidad José Carlos, Leonel, Darién, Ernesto, Carlos, Julio, Keimer, Antonio Miguel, Marcos, Popy, Osman, Suly, Poty, Eudel, Odalys, Mirita, Carmen, Marta, los buenos compañeros del edificio, a todos ellos le dedico este trabajo, A Janette y sus padres compartir una de las mejores etapas de mi vida, a Ivet por regalarme una parte de su colección de buenos momentos.

A quien tiene el 50% de este trabajo, mi hermanito y compañero Yohan, el cual estuvo desde el comienzo dando lo mejor de sí mismo.

A mis familiares que nunca dudaron de mi cuando tome le decisión de estudiar esta carrera, a mi padrastro Luis(El Chino)por alegrar con su buen humor los momentos que compartimos, a mis tíos Rangel y Raúl, a mis tías Sandra, Amelia, Elsa y Sonia, a todos mis primos fundamentalmente a David y Sandra, a mis hermano Emmanuel y Javier Eduardo el cual nunca me dejara de sorprender con sus buenos actos, demostraciones de respeto, cariño y admiración, a mis hermanas Carla, Estela y Gabriela la cual amo sin condición, a mi abuelo Ever, por enseñarme el significado de la palabra HOMBRE, por darme ese patrón a seguir en la vida, por todo su legado a él le agradezco de todo corazón, a la persona más noble que he conocido en el mundo, la que me mimo desde pequeño, esa que ve por mis ojos y se pone sentimental siempre que me despido de ella, a ti mi abuelita Vida, como te suelo llamar siempre, te dedico este trabajo y mi corazón.

Esta dedicatoria especial es para la mujer que más amo en este mundo, a la que le he dedicado cada meta superada, cada logro, la que diariamente me da fuerzas para ser una mejor persona, la que sé que nunca me dejara solo pase lo que pase, la que me ha demostrado amor incondicional, a ti mamita de mi alma (Lilian) te dedico mi VIDA.

Javier

RESUMEN

El Sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos, coordinado con el departamento de Almacenes de Datos y con el grupo de desarrollo del mismo departamento, tiene como objetivo desarrollar aplicaciones con el fin de mejorar la consola administrativa del *Business Intelligent Server.* Dicho sistema permite facilitar el despliegue y control de las rutinas de integración generadas por la herramienta *Pentaho Data Integration* utilizada por el grupo de desarrollo de ETL, por lo que se realiza un análisis de otros sistemas con características similares y se definen las tecnologías y herramientas para desarrollar el sistema. El ciclo de vida del proyecto estuvo guiado por la metodología OpenUP, se utilizó el lenguaje de programación Java. Como resultado se obtuvo un sistema capaz de controlar las soluciones realizadas en el departamento de Almacenes de Datos, un producto con un alto grado de seguridad, además del cumplimiento de los requisitos funcionales propuestos por el cliente.

PALABRAS CLAVE

Almacenes de Datos, Consola, rutinas de integración, sistema.

TABLA DE CONTENIDO

TABLA DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde siempre, el hombre ha tenido la necesidad de nutrirse de conocimientos, haciendo dela indagación de aspectos informativos un área de su interés. La creación, búsqueda y obtención de información (según (1), conjunto organizado de datos, que constituye un mensaje sobre un cierto fenómeno o ente), son acciones inherentes a la naturaleza humana. La búsqueda constante de soluciones a los problemas, así como de los datos relacionados con los mismos, y de maneras de gestionar estas bases de conocimientos, ha constituido unos de los impulsos que ha logrado la instauración en el mundo de instrumentos cada día más poderosos y veloces en el proceso informativo, dando paso al surgimiento de la informática, ciencia encargada de la gestión automática de la información (2). Esta ciencia, a diferencia de otras, es capaz de cambiar la concepción de soluciones de problemas con proposiciones de métodos y estrategias, en un corto periodo de tiempo, llegando a jugar un papel protagónico en casi todas las esferas de la sociedad.

La creciente tendencia de acceder a grandes cúmulos de información y de la utilización de métodos más ágiles para consultarla, trajo como consecuencia la aparición de la industria del software, que hoy en día, contribuye a la toma de decisiones en las más importantes empresas, aportando un gran avance en su desarrollo tecnológico (3). En aras de aprovechar los beneficios aportados por la industria del software, y de la utilización de las nuevas tecnologías y conocimientos que resultan rasgos fundamentales de la revolución tecnológica, el Estado Cubano se propuso la creación de una infraestructura científica y técnica que permitiese contar con un conjunto de centros de investigación capaces de ofrecer soluciones que tributasen a la informatización de la sociedad cubana. Por esta razón, en septiembre del 2002, al calor de la Batalla de Ideas, surge la Universidad de las Ciencias Informáticas, casa de altos estudios que vincula la docencia y la investigación con la producción de software.

La Universidad de las Ciencias Informáticas cuenta con disímiles sedes productivas dedicadas a la búsqueda de soluciones informáticas. El Centro de Tecnologías de Gestión de Datos (DATEC), perteneciente a la Facultad 6, está formado, entre otros, por el departamento de Almacenes de Datos, que utiliza para el desarrollo de las soluciones de Inteligencia de Negocio la suite de Pentaho en su versión comunitaria. Específicamente dentro de la línea de ETL se utiliza la herramienta *Pentaho Data Integration*, sin embargo, debido a las experiencias de los especialistas de dicha línea se han detectado una serie de inconvenientes:

- Actualmente el despliegue y ejecución de las soluciones de integración de datos se realiza desde la consola del sistema operativo, por lo cual dicho proceso resulta engorroso y exige que los especialistas posean conocimientos sobre el sistema operativo.
- Una vez desplegadas las soluciones de integración de datos, resulta necesario administrar sus procesos. Sin embargo, para llevar a cabo actividades como verificar el estado de ejecución o simplemente modificar la planificación del mismo, resulta necesario en muchas ocasiones detener los procesos o realizar cambios en la implementación de las rutinas de integración. Producto de esta situación, se hace muy difícil controlar la ejecución de dichas rutinas, corriendo el riesgo de provocar inconsistencias en los datos que están siendo procesados o en el mejor de los casos, sería necesario implementar directamente dentro de las rutinas de integración los mecanismos necesarios.

Luego de un análisis de lo anteriormente descrito se identifica como **problema de la investigación**: ¿Cómo gestionar los procesos de integración de datos implementados para la herramienta *Pentaho Data Integration*?

La investigación en curso enmarca como **objeto de estudio**: los procesos de integración de datos, delimitando como **campo de acción**: la gestión de los procesos de integración de datos de la herramienta *Pentaho Data Integration*.

Para dar solución al problema de la investigación se ha definido como **objetivo general**: desarrollar un sistema que permita gestionar los procesos de integración de datos de la herramienta *Pentaho Data Integration*.

Se desglosan del objetivo general los siguientes **objetivos específicos**:

- Fundamentar la metodología, herramientas y tecnologías a utilizar para el desarrollo de la solución.
- Realizar análisis y diseño del sistema.
- Implementar el sistema.
- Realizar pruebas al sistema.

Para dar cumplimiento a los objetivos específicos se definen las siguientes **tareas de la investigación**:

- Caracterización de la metodología, herramientas y tecnologías a utilizar en el desarrollo de la solución.
- Realización del análisis del sistema.

- Obtención de los diagramas de clases del diseño.
- Elaboración de los diagramas de componentes.
- Implementación del sistema.
- Aplicación de pruebas al sistema.

La investigación refleja como posible resultado un sistema para la gestión de los procesos de Integración de Datos para la herramienta *Pentaho Data Integration.*

El Trabajo de Diploma está estructurado de la siguiente forma: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

Capítulo 1: Fundamentación Teórica.
Este capítulo abarca las principales características y conceptos de la investigación, se realiza un análisis del estado del arte del objeto de estudio, y se caracteriza la metodología, herramientas y tecnologías a utilizar.

Capítulo 2: Análisis y Diseño del sistema.
Se realiza un levantamiento de los requisitos funcionales y no funcionales, la identificación de los actores del sistema que intervienen. Se hace una representación de los casos de uso del sistema y los patrones utilizados. Se modelan los artefactos correspondientes, los diagramas de clases del diseño por cada caso de uso crítico, los diagramas de secuencias, el cual brinda con mayor claridad la infraestructura desarrollada.

Capítulo 3: Implementación y Prueba.
Se describen cómo serán implementados los elementos correspondientes al modelo del diseño y se realizan pruebas de caja blanca y casos de prueba de caja negra al sistema para controlar su correcto funcionamiento. Además se refleja el resultado de la implementación con las vistas del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

El presente capítulo comprende un breve estudio de los procesos de integración de datos en las soluciones de almacenes de datos, haciendo énfasis además en aspectos fundamentales para el desarrollo del sistema, como son las herramientas, metodología y tecnologías a utilizar.

1.2 Metodología de Desarrollo de Software.

Las metodologías de desarrollo de software surgieron a raíz de la necesidad de controlar, guiar, y documentar proyectos cada vez más complejos, impulsadas principalmente por instituciones económicamente importantes y con requisitos de seguridad y fiabilidad en sus sistemas sumamente estrictos. (ACUÑA, 2006)

La experiencia acumulada a lo largo de los años de ejecutar proyectos, indica que los proyectos exitosos son aquellos que son administrativos, siguiendo una serie de procesos que permiten organizar y luego controlar el proyecto. Según esta visión los proyectos que no sigan estos lineamientos corren un alto riesgo de fracasar. Para esto es necesario el uso de metodologías de desarrollo.

Las metodologías de desarrollo de software son un conjunto de técnicas, procedimientos y ayudas a la documentación que permiten estructurar, controlar y planificar el proceso de desarrollo de software. Estas metodologías poseen dos grandes enfoques: robustas y ágiles, las primeras, pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto, mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente (4).

1.2.1 Metodologías Tradicionales o Robustas.

Se enfocan en la definición detallada de los procesos, tareas a realizar y herramientas a utilizar. Estas requieren de una extensa documentación, pues pretenden prever todo de antemano. Las metodologías tradicionales son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear (5).

Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

Dentro de las principales metodologías tradicionales y las más usadas, se encuentra la metodología RUP (Proceso Racional Unificado), que centra una documentación extensiva y compleja de todo el proyecto.

1.2.1.1 Proceso Racional Unificado (RUP).

La metodología RUP (5) es una metodología de desarrollo de software iterativo, creado por Rational Software Corporation. RUP resultó de la combinación de varias metodologías y se vio influenciado por métodos previos como el modelo en espiral. Entra en la clasificación de las metodologías robustas, como también es de las más antiguas y usadas.

Las cuatro fases (Figura 1) en las que se divide el proceso de desarrollo del software son las listadas a continuación:

Inicio: determinar la visión del proyecto.

Elaboración: determinar la arquitectura óptima.

Construcción: obtener la capacidad operacional inicial.

Transmisión: obtener el realce del proyecto.

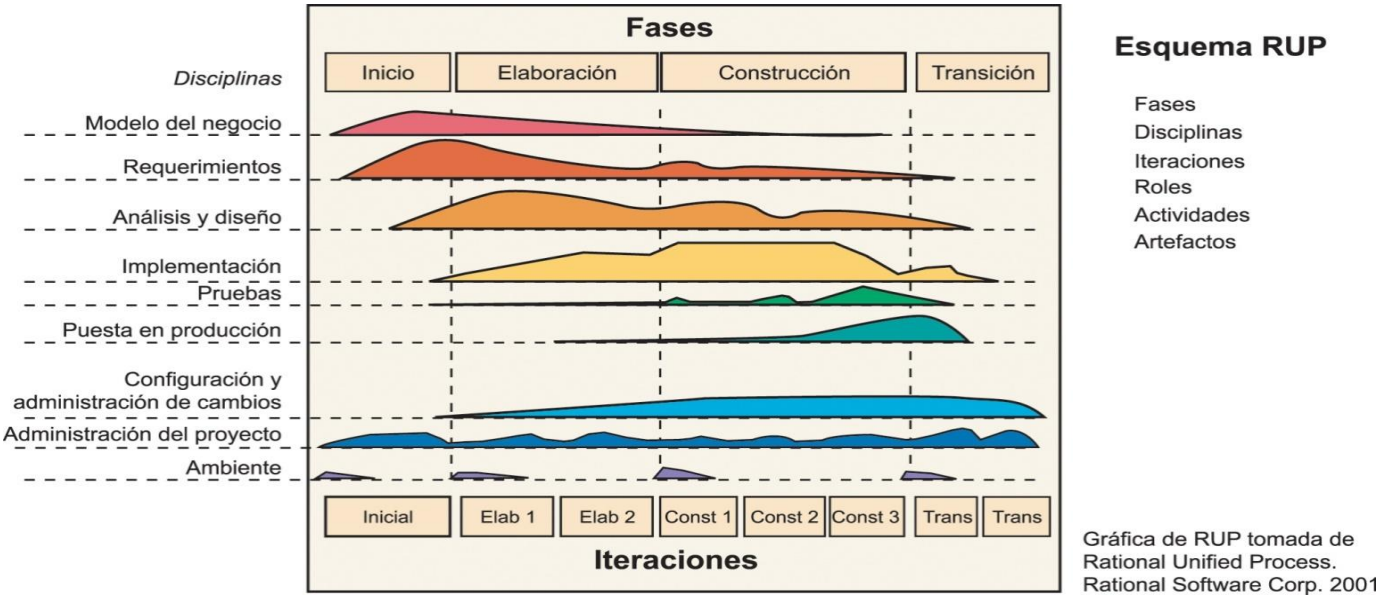


Figura 1. Fases de RUP

Cada etapa se desarrolla mediante el ciclo de iteraciones, consistiendo esto en reproducir el ciclo de vida en cascada a menor escala, dándole esto la característica de ser Iterativo-incremental. El ciclo de vida desarrollado en cada iteración es llevado bajo dos disciplinas que son:

Disciplina de Desarrollo

- ✓ **Modelado del Negocio**: analizar y entender las necesidades del negocio para el cual se está desarrollando el software.
- ✓ **Requisitos**: proveer una base para estimar los costos y tiempo de desarrollo del sistema.
- ✓ **Análisis y diseño:** trasladar los requisitos analizados anteriormente a un sistema automatizado y desarrollar una arquitectura para el sistema.
- ✓ **Implementación:** crear un software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
- ✓ **Pruebas:** asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
- ✓ **Despliegue:** producir distribuciones del producto y distribuirlo a los usuarios.

Disciplina de Soporte

- ✓ **Configuración y administración del cambio:** guardar todas las versiones del proyecto.
- ✓ **Administración del proyecto:** administrar los horarios y recursos que se deben de emplear.
- ✓ **Entorno:** administrar el entorno de desarrollo del software.

Las características principales del proceso son:

- ✓ Dirigidos por Casos de Uso.
- ✓ Centrado en la Arquitectura.
- ✓ Iterativo e Incremental.

RUP implementa las siguientes mejores prácticas al proceso de Ingeniería de Software:

- ✓ Desarrollo Iterativo.

- ✓ Manejo de los Requerimientos.
- ✓ Uso de una Arquitectura basada en componentes.
- ✓ Modelización Visual.
- ✓ Verificación Continua de la Calidad.
- ✓ Manejo de los Cambios. (PRESSMAN, 2005)

1.2.2 Metodologías Ágiles.

Las metodologías ágiles, tienen como común denominador un modelo de desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos, y predisposición para el campo y la adaptación continúa; según sea la conformidad o no de lo producido, y las modificaciones propuestas por los usuarios. Estas metodologías por lo general se centran en desarrollar productos funcionales más que en conseguir una buena documentación.

Se basan en dos aspectos fundamentales: retrasar las decisiones y la planificación adaptativa. Establecen su fundamento en la adaptabilidad de los procesos de desarrollo. El desarrollo de software es incremental, cooperativo, sencillo y adaptado. Una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos, encuentran estas metodologías bastante agradables (5).

1.2.2.1 SCRUM.

La metodología ágil SCRUM se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. Es un soporte de proceso que incluye un conjunto de prácticas y roles predefinidos (5).

Dentro de esta metodología existen varios roles como los que se muestran a continuación:

Scrum Master: mantiene los procesos y trabajos junto con el jefe de proyecto.

Product Owner: representa a las personas implicadas en el negocio.

Team: incluye a los desarrolladores.

SCRUM permite la creación y organización de grupos de trabajos con el objetivo de que todos los miembros del equipo se encuentren en un mismo local. Promueve la comunicación verbal y las disciplinas implicadas en el proyecto. Una de las mayores ventajas de SCRUM es que es muy fácil de entender y requiere poco esfuerzo para comenzar a usarse.

1.2.2.2 Extreme Programming (XP).

La programación extrema es un modelo de programación, basada en una serie de metodologías de desarrollo de software, en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación (GÓMEZ, 2006).

Se enfoca en incrementar las relaciones entre las personas del proyecto como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, toma en cuenta la adquisición de conocimientos de los desarrolladores, favoreciendo un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, buen entendimiento entre los involucrados, facilidad en las soluciones implementadas y audacia para asumir los cambios. (5)También se utiliza en proyectos con requisitos indefinidos y muy cambiantes donde existe un alto riesgo técnico.

XP está compuesto por una serie de roles que se muestran a continuación:

- ✓ Programador
- ✓ Cliente
- ✓ Encargado de pruebas (tester)
- ✓ Encargado de seguimiento (tracker)
- ✓ Entrenador (coach)
- ✓ Consultor
- ✓ Gestor (bigboss)

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del coste del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione.

1.2.2.3 Open Unified Process (OpenUP).

OpenUP es una metodología de desarrollo de software de código abierto. Esta metodología de desarrollo unificado está basada en RUP, del cual posee características como, desarrollo iterativo incremental, casos de uso y escenarios de conducción de desarrollo, la gestión de riesgo y el enfoque centrado en la arquitectura (6).

Es reconocida mundialmente como una de las metodologías de desarrollo de software de mayor calidad, basándose en los principios de “adaptación”, “importancia a los involucrados e interesados en los resultados del proyecto”; “colaboración”, “valor a la iteración”; y “calidad continua”.

OpenUP es aplicable a proyectos pequeños, con grupos de dos a seis personas interesadas en el desarrollo rápido e interactivo. Cumple con tres características esenciales, es mínimo, completo y extensible. Es extensible, porque puede ser utilizada como base para agregar o adaptar según las necesidades. Es un proceso ejemplar y extensible para una gama de los procesos del desarrollo y de la gerencia del software que apoya el desarrollo iterativo, ágil, e incremental y es aplicable a un amplio sistema de plataformas y de usos del desarrollo. Mínimo, porque solo se incluye lo fundamental y completo, abarcando las disciplinas esenciales para el ciclo de vida de desarrollo del software, que guían al equipo hasta la obtención de un producto.

A continuación (Figura 2.) se muestran las cuatro fases de OpenUP así como sus disciplinas de desarrollo.

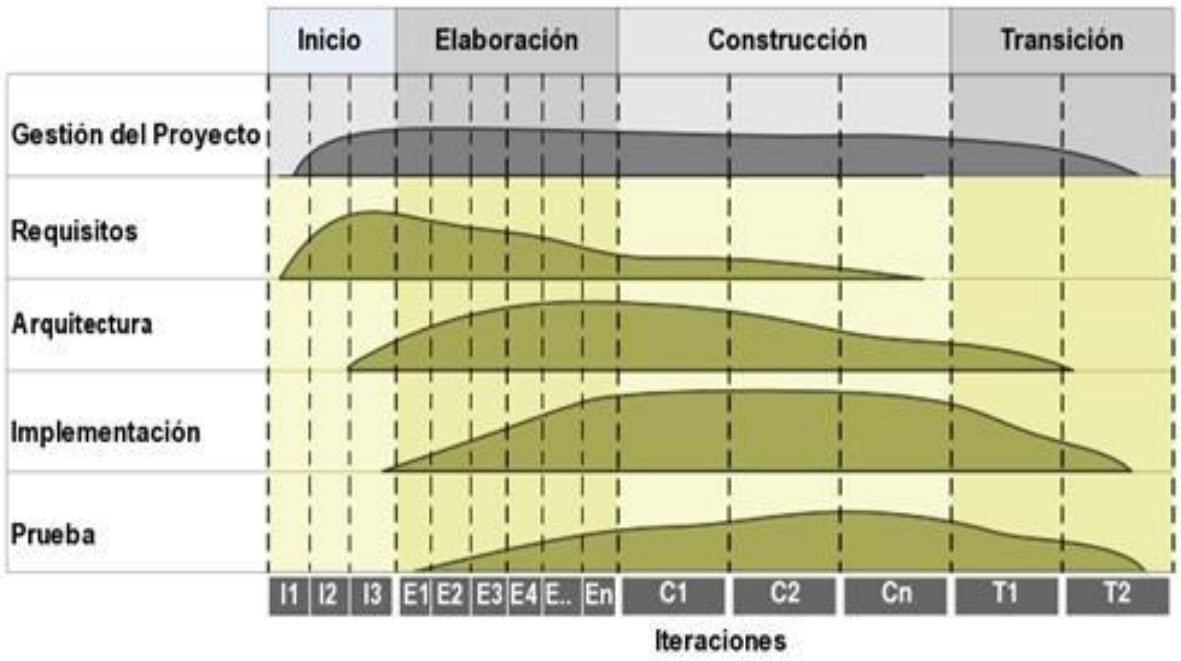


Figura 2. Fases de OpenUP

1.3 ¿Por qué OpenUP?

Por la necesidad de utilizar poco personal y alcanzar el desarrollo del Sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos en un corto periodo de tiempo, se decide utilizar la metodología OpenUP. Una de sus características principales es que se emplea en proyectos de corta duración, tomando las bases fundamentales de RUP: centrado en la arquitectura, dirigido por caso de uso e iterativo incremental.

1.4 Lenguaje de Modelado Unificado (UML: Unified Modeling Language).

El Lenguaje Unificado de Modelado establece un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación (7).

UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Además una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos.

Su utilización es independiente del lenguaje de programación y de las características del proyecto, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto como informáticos o de arquitectura, o de cualquier otra rama. Permite la modificación de todos sus miembros mediantes estereotipos y restricciones. Un estereotipo permite indicar especificaciones del lenguaje al que se refiere el diagrama UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción se fuerza el comportamiento que debe tener el objeto al que se le aplica (8).

1.5 Herramientas CASE

Las herramientas CASE (en inglés: *Computer Aided Software Engineering,* en español: Ingeniería de Software Asistida por Computadora.) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de un sistema (9). En ellas se integran el análisis de datos y procesos integrados mediante un repositorio, generación de interfaces entre el análisis y el diseño, generación del código a partir del diseño, y control de mantenimiento.

Existen disímiles herramientas CASE entre ellas se encuentran Microsoft Project, Rational Rose, JDeveloper, Magic Draw, Microsoft Visio, BoUML. Además existe la herramienta Visual Paradigm que se utiliza en el proyecto, la cual posee licencia en la Universidad de las Ciencias Informáticas.

1.5.1 Visual Paradigm para UML v8.0.

Visual Paradigm es una herramienta CASE profesional, que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, además proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (10).

También, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto y permite control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad (DELGADO PORRES, REYES MATOS, 2011).

Esta herramienta presenta numerosas características, que resultan útiles para elaborar una aplicación con gran calidad:

- ✓ Soporta aplicaciones Web.
- ✓ Varios idiomas.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.

1.6 Lenguajes de Programación y Tecnologías.

Por la experiencia de los desarrolladores, para la implementación del sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos, es necesario utilizar el lenguaje de programación Java y la tecnología *JavaServer Pages* (JSP). Conjuntamente a esto, existen en la actualidad gran cantidad de herramientas libres que permiten su desarrollo ágil y eficiente. Además de la gran compatibilidad entre el lenguaje y la tecnología seleccionada.

1.6.1 Java.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principio de los años 90, diseñado por James Gosling y basado en C++, con el objetivo de desarrollar un lenguaje de programación moderno y potente. En sus inicios se le llamó Oak por un roble que había fuera de la oficina de Gosling y después al ver que Oak era una marca registrada se le nombró Green y finalmente Java (11).

Las aplicaciones Java están compiladas en un código intermedio llamado bytecode (fichero binario que tiene un programa ejecutable), que es interpretado por una máquina genérica llamada Máquina Virtual de Java (JVM). Java posee una amplia gama de aplicaciones, siendo utilizada en dispositivos móviles, como teléfonos y pequeños electrodomésticos.

Entre sus características principales posee un lenguaje simple, el cual procede de la misma estructura de C y C++. Es orientado a objetos en su mayoría, facilitando su manipulación. Permite abrir sockets, establecer y aceptar conexiones con los servidores y clientes remotos por lo que es distribuido. Es a su vez robusto porque proporciona numerosas comprobaciones en compilación y en tiempo de ejecución, es seguro, siendo uno de las características fundamentales, puesto que se han implementado barreras de seguridad en el sistema de ejecución; además es de alto rendimiento, siendo muy veloz en el momento de levantar programas y ahorrando muchas líneas de código.

1.6.2 Java Server Pages (JSP).

Java Server Pages es la tecnología de la plataforma Java que permite transmitir contenido dinámico a los clientes web de una manera portátil y seguro, como HTML, DHTML, XHTML y XML. La especificación Java Server Pages extiende el Java Servlet API para ofrecer a los desarrolladores de aplicaciones web un marco sólido en el contenido de la web usando HTML, XML, plantillas y código Java (12).

Las páginas JSP, pueden ser trasladadas fácilmente a varias plataformas y entre servidores web, sin la necesidad de realizarle cambios de ningún tipo, permitiendo combinar el poder de la tecnología Java por el lado del servidor, con las ventajas de las páginas HTML estáticas.

1.7 Herramientas de Desarrollo.

Las herramientas de desarrollo son aquellos programas encargados de desarrollar un programa o aplicación, el cual puede ser un ensamblador, compilador, editor o un Entorno de Desarrollo Integrado (IDE) (13). Los IDEs son un conjunto de herramientas para el programador, que suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores e integración con sistemas controladores de versiones o repositorios (LUCIANO, 2010).

1.7.1 Eclipse.

Eclipse es un Entorno de Desarrollo Integrado de código abierto y extensible, desarrollado por la fundación Eclipse, que es una organización sin ánimo de lucro que creó la promoción y evolución de la

plataforma Eclipse. Esta se divide en proyectos capaces de proporcionar una plataforma robusta, escalable y de calidad (14).

La plataforma Eclipse entre sus características principales se encuentra el resaltado de sintaxis donde posee un editor de texto. Además dispone de módulo para realizar pruebas unitarias JUnit, Sistema Control de Versiones (CVS, Control Version System) e integración con Ant (15) y es compatible con los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX.

Sin embargo esta plataforma posee una arquitectura basada en plug-ins, permitiendo agregar editores, lenguajes de programación y visores de forma fácil.

1.7.2 NetBeans.

NetBeans al igual que Eclipse es un Entorno de Desarrollo Integrado, nació como un proyecto estudiantil en 1996 en la Republica Checa, nombrado originalmente Xelfi, el cual fue el primer IDE para Java escrito en el mismo lenguaje.

Este IDE soporta Java SE, Java EE, Java ME, además de tener una gran cantidad de módulos de terceros o plugins. En Junio de 2000 pasa a ser una herramienta de código abierto, donde se le incrementa en gran medida su comunidad y se le unen mucho más desarrolladores. Con este potente IDE se pueden desarrollar aplicaciones completas para el cliente, estas pueden ser de escritorio, web, móvil y de empresa, además de crear ventanas, menús, barras de herramientas y otras acciones fácilmente. NetBeans corre sobre los sistemas operativos Windows, Linux, Open Solaris y Mac OS, destacando que es 100% Java. Entre sus características fundamentales resaltan las listadas a continuación: (16)

- ✓ Soporte para Ruby, JRuby, y Ruby on Rails.
- ✓ Instalación y actualización más simple.
- ✓ Enlazar datos con el Swing GUI.
- ✓ Características visuales para el desarrollo web.
- ✓ Creador gráfico de juegos para celulares.
- ✓ Mejoras para SOA y UML.
- ✓ Soporte para PHP.

Para el desarrollo del proyecto se utiliza el IDE de desarrollo Eclipse, el cual brindaresaltado de sintaxis donde posee un editor de texto. Además tiene una enorme variedad de plugins, el despliegue de capacidades que no están necesariamente disponibles con NetBeans.

1.8 Herramientas de Integración de Datos.

Las herramientas de integración de datos son aquellas que proporcionan de forma general una serie de funcionalidades, como por ejemplo: (17)

- ✓ Control de la extracción de los datos y su automatización, disminuyendo el tiempo empleado en el descubrimiento de procesos no documentados, minimizando el margen de error y permitiendo mayor flexibilidad.
- ✓ Acceso a diferentes tecnologías, haciendo un uso efectivo del hardware, software, datos y recursos humanos existentes.
- ✓ Proporcionar la gestión integrada del Almacén de datos y los Mercados de datos existentes, integrando la extracción, transformación y carga para la construcción del Almacén de datos corporativo y de los Mercados de datos.
- ✓ Uso de la arquitectura de metadatos, facilitando la definición de los objetos de negocio y las reglas de consolidación.
- ✓ Acceso a una gran variedad de fuentes de datos diferentes.
- ✓ Manejo de excepciones.
- ✓ Planificación, registros, interfaces a programadores de terceros, que permitirán llevar una gestión de la planificación de todos los procesos necesarios para la carga del Almacén de datos.
- ✓ Interfaz independiente de hardware.
- ✓ Soporte en la explotación del Almacén de datos.

1.8.1 Pentaho Data Integration (PDI o Kettle).

La suite de Pentaho proporciona un espectro completo de herramientas de inteligencia de negocio, reportes, análisis, dashboards, minería de datos e integración de datos. Ofrece además, una serie de servicios críticos entre los que están la autentificación, programación de tareas, seguridad y servicios web. Este conjunto de herramientas y servicios forman una plataforma integral de inteligencia de negocio, convirtiendo a *Pentaho* en una plataforma potente de Inteligencia del Negocio (*Business Intelligence (BI)*) de código abierto.

Pentaho Data Integration es un herramienta de integración de datos de código abierto que se encarga de la extracción, transformación y carga (ETL) de los procesos de integración de datos (18). Permite desarrollar y desplegar poderosas aplicaciones de *Business Intelligence* iterativas con la participación de desarrolladores y usuarios finales, combinando el desarrollo de soluciones complejas en un solo proceso, con un ahorro considerable de tiempo.

Pentaho Data Integration está conformado por varias herramientas con un propósito en específico, las cuales son:

Spoon: herramienta principal de trabajo que permite el diseño de las transformaciones y los trabajos (Jobs).

Pan: herramienta que permite ejecutar las transformaciones desarrolladas en el Spoon y permite ejecutar scripts desde la línea de comandos.

Kitchen: permite ejecutar los Jobs diseñados en el Spoon.

Carte: servidor web para ejecutar remotamente las transformaciones y los trabajos.

1.8.2 Talend Open Studio.

Talend (19) es un proyecto de la empresa *Talend Open Data Solutions,* posee una interfaz de usuario bastante atractiva basada en la plataforma Eclipse, es una herramienta de integración de datos de código abierto y multiplataforma. Esta herramienta de ETL basa su diseño en tres niveles los cuales se listan a continuación:

- ✓ Modelado de Negocio: este nivel está diseñado para modelar de manera teórica la aplicación.
- ✓ Diseño de Trabajos: en este nivel se diseña el trabajo en sí, el código que se ejecutará posteriormente.
- ✓ Contextos: es el nivel que contiene los contextos, los cuales pueden ser definidos como variables globales de ejecución del programa, como la carpeta donde se ejecutará la aplicación final o variables iniciales de entrada (20).

Se decide utilizar para el proyecto en curso la herramienta *Pentaho Data Integration*, teniendo en cuenta que esta es la herramienta principal utilizada en el departamento de Almacenes de Datos. Además es de código abierto y desarrollado en java, lo cual facilita la integración con el sistema a desarrollar.

1.9 Frameworks.

Un Framework es un marco de trabajo, un conjunto de herramientas, librerías y buenas prácticas que simplifican tareas repetitivas, o lo comúnmente llamado código reutilizable. Tiene que cumplir la condición de poder ser utilizado en cualquier ámbito de trabajo y soportar aplicaciones de diferentes tamaños, donde se pueda utilizar sus componentes en toda la aplicación, además de ser extensible lo que significa que se pueda definir parámetros para crear componentes que se integren fácilmente con el resto de las funcionalidades del Framework (21).

1.9.1 Framework Spring.

Según (22) Framework Spring es una plataforma que proporciona una infraestructura que actúa de soporte para desarrollar aplicaciones Java.

Este es uno de los marcos de trabajo más populares de la comunidad de Java. Como parte de la gran gama de Framework existentes en el mundo incorpora una serie de librerías que facilitan el trabajo en el desarrollo de aplicaciones. Framework Spring implementa el patrón Modelo Vista Controlador (MVC) donde se define y separa muy bien la interfaz de usuario, los controladores y la capa de acceso a datos, además de poder utilizarse con otros marcos de trabajo como lo son JSF y STRATUS. También permite el trabajo con herramientas ORM (Object Relational Mapping) como Hibernate y con motores de plantillas para interfaces de usuarios como Tiles y Velocity.

1.9.2 Spring Security.

Spring Security es un Framework de seguridad para aplicaciones Web, fundamentadas en su totalidad en Framework Spring. El mecanismo de seguridad que utiliza es declarativo e independiente del entorno donde se despliega la aplicación; el mismo proporciona una solución completa para los dos requisitos fundamentales de seguridad, autenticación y autorización (23).

1.9.3 Spring Web Modelo Vista Controlador (Spring Web MVC).

Spring Web MVC en su versión 2.0, es un Framework moderno que ofrece una separación clara de los modelos, las vistas y controladores, además de ser muy flexible debido a que implementa su estructura mediante interfaces. Las partes de Spring MVC se pueden probar fácilmente, puesto que no obliga a la herencia de clases y una dependencia directa del controller del Servlet que despacha las peticiones. Además el mismo ofrece una interfaz bien definida para la capa de negocio (24).

El framework tiene un conjunto de interfaces que después se implementan para proporcionar la funcionalidad correspondiente. Las interfaces están acopladas claramente al Servlet Api.
La clase Dispatcher Servlet está en el controlador frontal y es responsable de delegar y coordinar el control entre varias interfaces en la fase de ejecución durante una petición http.
Las interfaces más importantes definidas en Spring MVC, y sus responsabilidades, son las siguientes: (24)

- ✓ Handler Mapping: permite manejar peticiones de entrada.
- ✓ Handler Adapter: ejecución de objetos que permiten manejar las peticiones entrantes.
- ✓ Controller: está entre el modelo y la vista, y permite manejar peticiones entrantes y redirigirlas a la respuesta adecuada.
- ✓ Vista: responsable de retornar una respuesta al cliente.
- ✓ View Resolver: selecciona una vista basada en un nombre lógico de la vista.
- ✓ Handler Interceptor: intercepta las peticiones entrantes, es comparable pero no igual a los filtros de Servlet.
- ✓ Locale Resolver: resuelve y opcionalmente salva el local de un usuario individual.
- ✓ Multipart Resolver: facilita trabajar con ficheros de subida encapsulando peticiones de entrada.

1.9.4 Framework Apache Struts.

Apache Struts (25) es un Framework de software libre orientado al desarrollo de aplicaciones web. Struts se ajusta al patrón Modelo Vista Controlador, que establece una separación de la capa de lógica de negocio, la capa de presentación y la capa de control.
Proporciona un desarrollo de aplicaciones basado en “capas”, bajo un entorno más robusto y escalable que el método basado exclusivamente en JSP. Struts se basa en tecnología Java y en estándares de la comunidad SUN para proporcionar estas características.
Un aspecto fundamental del entorno Struts lo constituye la forma en que extiende el flujo petición respuesta de una aplicación web convencional. El controlador del modelo Struts gestiona los caminos que la aplicación web recorre, ayuda en la recolección de los datos necesarios e incluso puede proporcionar información en varios idiomas (internacionalización) (26).

1.9.5 Struts Modelo Vista Controlador.

Struts Modelo Vista Controlador (27) es un patrón de diseño aportado originariamente por el lenguaje SmallTalk a la Ingeniería del Software. El paradigma MVC consiste en dividir las aplicaciones en tres partes:

- ✓ La capa de **Modelo** comprende los objetos de negocio (típicamente representan objetos almacenados en nuestras fuentes de datos), la lógica de negocio y los accesos a las fuentes de datos (bien sean bases de datos o sistemas de ficheros, por ejemplo). Es la parte que siempre hay que desarrollar de forma diferente, y es también la parte en la que menos aporta Struts.
- ✓ La capa de **Vista**, es decir, la capa de presentación. Típicamente las aplicaciones web tienen tres tipos de presentación: las pantallas de información, las pantallas en las que un usuario introduce datos, y las pantallas donde se presentan los resultados solicitados. Tanto la captación de los datos introducidos como la presentación de los resultados se realiza mediante formularios (implementados por clases *ActionForm*) de forma automatizada por Struts. Struts escribe en el formulario los datos introducidos en las pantallas de captación de datos, y lee los datos del formulario que contiene los resultados en las pantallas de presentación. Para eliminar el código Java en las páginas JSP y para automatizar tareas, Struts facilita sus taglibs (html, bean y logic).
- ✓ La capa de **Controlador**. Esta capa decide qué acciones se ejecutarán y cuál será la página de destino de estos resultados. Struts proporciona la clase ActionServlet que automatiza todo este trabajo y permite su configuración (struts-config.xml) de una forma muy sencilla, eligiendo la acción que se ejecutará (implementada por clases Action); esta acción elegirá un destino posible (ActionForward) (25).

A continuación (Figura 3) se introduce las líneas generales del comportamiento del patrón Modelo-Vista-Controlador. La capa Modelo es la encargada de la gestión de los datos. La capa Vista tiene la función de presentar los datos del modelo al usuario y recoger las peticiones de éste para enviarlas a la capa del Controlador, que responderá a las acciones del usuario y realizarán cambios tantos en el modelo como en la vista. Se puede resumir su esquema de funcionamiento de la siguiente forma:

- El cliente manda una petición HTTP que tiene como destino un Servlet (Controlador).

- Al recibir la petición HTTP el Servlet, delega el proceso de los datos al Modelo.
- El Modelo ejecuta los procesos necesarios y obtiene unos resultados.
- El Controlador redirige el conjunto de procesos de la petición a la Vista.
- La Vista recupera los datos generados por el Servlet y genera una respuesta HTTP.
- La respuesta HTTP es enviada hacia el cliente.

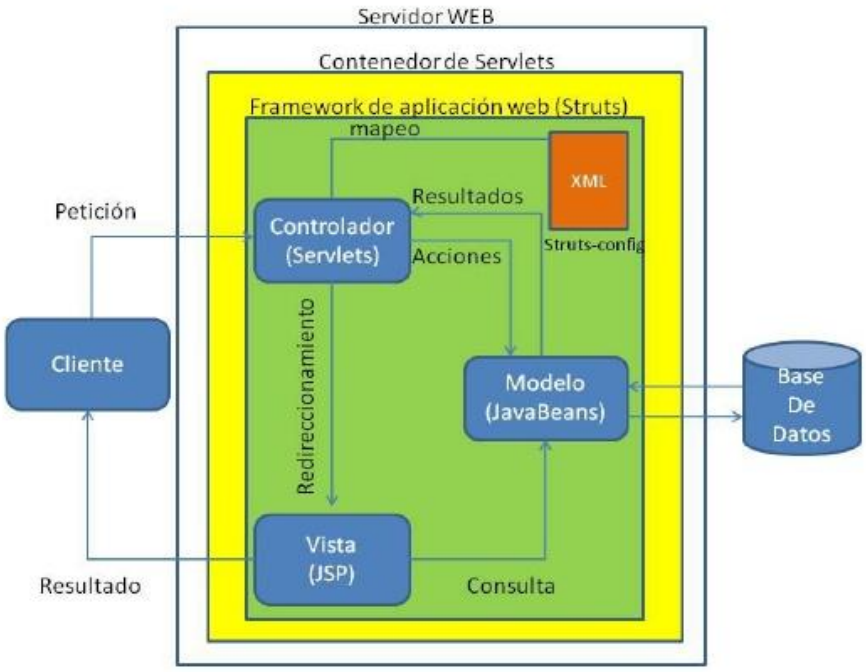


Figura 3. Funcionamiento de Struts

1.9.6 Spring MVC vs Struts MVC.

A continuación de muestran un listado con las principales ventajas y desventajas entre el Modelo Vista controlador Spring y el Modelo Vista Controlador Struts:(25)

- ✓ Spring MVC ofrece una división limpia entre Controladores, Modelos (JavaBeans) y Vistas.
- ✓ Spring MVC es muy flexible, ya que implementa toda su estructura mediante interfaces, no como Struts que obliga a heredar de clases concretas tanto en sus Actions como en sus Forms. Además,

todas las partes del Framework son configurables vía plugins en la interface, aunque Spring provee clases concretas como opción de implementación.

- ✓ Spring MVC provee interceptores también como controladores que permiten factorizar el comportamiento común en el manejo de múltiples requests.
- ✓ Spring MVC no obliga a utilizar JSP, permite utilizar XLST, Velocity o implementar tu propio lenguaje para integrarlo en la Vista de la aplicación.
- ✓ Los controladores de Spring MVC se configuran mediante registros como los demás objetos, lo cual los hace fácilmente testeables e integrables con otros objetos que estén en el contexto de Spring, y por tanto sean manejables por éste.
- ✓ Las partes de Spring MVC son más fácilmente testeables que las de Struts, debido a que evita la herencia de una clase de manera forzosa y una dependencia directa en el controlador del servlet (controller-servlet.xml) que despacha las peticiones.
- ✓ La capa Web de Spring es una pequeña parte en lo alto de la capa de negocio de Spring, lo cual parece una buena práctica. Struts y otros Frameworks web dejan a tu elección la implementación de los objetos de negocio, mientras que Spring ofrece un Framework para todas las capas de la aplicación.
- ✓ Más cantidad de código testeable, las validaciones no dependen de la Api de servlets.
- ✓ Struts obliga a extender la clase Action, mientras que Spring MVC no, aunque proporciona una serie de implementaciones de *Controllers* para que el usuario los utilice.
- ✓ Spring tiene una interfaz bien definida para la capa de negocio.

Se decide utilizar como Framework, Spring MVC, puesto que ha aprendido de la experiencia adquirida de usar Struts para no cometer errores similares. Además porque no existen ActionForms, se enlaza directamente con los beans del negocio y porque ofrece una división limpia entre Controladores, Modelos (JavaBeans) y Vistas.

1.10 Servidor Web.

Un servidor Web es un programa que se ejecuta ininterrumpidamente en una computadora, en espera de peticiones que hagan los usuarios de la red. Estas peticiones son respondidas a través del navegador de forma adecuada en correspondencia con la solicitud (28).

Siguiendo las características del sistema para la gestión de los procesos de integración de datos en las soluciones de almacenes de datos se utilizará como servidor web el *Apache Tomcat*, el cual se caracteriza por su fácil uso y ser muy ligero, permitiendo una solución potente del problema en cuestión y contribuyendo en gran parte con el producto final.

1.10.1 Jetty Web Server.

Jetty resalta como características principales: posee licencia *Open Source*, escrito en Java y muy ligero. Esto permite embeberlo dentro de las grandes aplicaciones Java. Además las facilidades de su tamaño son muy eficientes para ofrecer Servicios Web en aplicaciones independientes.

Posee una colección de conectores relacionados entre sí, que son los encargados de responder las conexiones HTTP. Generalmente permite que las aplicaciones de escritorio sean accedidas desde un cliente de servicios web ahorrando una gran cantidad de tiempo y trabajo. Por su capacidad de ejecutar servlets y utilizar JSP se usa para tener una interfaz de administración que pueda comunicarse con un servicio Windows o proceso ejecutable (29).

1.10.2 Apache Tomcat V6.0.

Apache Tomcat, también conocido como Yakarta Tomcat, es un servidor web multiplataforma que funciona como contenedor de servlets y que se desarrolla bajo el proyecto denominado Yakarta, perteneciente a la *Apache Software Foundation* (30), bajo la licencia Apache 2.0 y que implementa las especificaciones de los servlets y de Java Server Pages o JSP (31) de Sun *Microsystem*. Dicho servidor es mantenido y desarrollado por miembros de la fundación y voluntarios independientes, los cuales tienen libre acceso al código fuente bajo los términos establecidos por la *Apache Software Foundation.*

Tomcat no es un servidor de aplicaciones y dicho producto fue desarrollado en Java. Puede ejecutarse sobre cualquier sistema operativo, previa instalación de la Máquina Virtual de Java aunque, también se puede usar con servidores de aplicaciones como: MAMPP (Mac OS X), LAMPP (GNU/Linux), WAMPP (Windows) o XAMPP. Además, puede funcionar como servidor web por sí mismo sin embargo, en sus inicios se pensaba que dicho servidor era recomendable usarse en entornos de desarrollo con requisitos mínimos de velocidad. En la actualidad no existe esta percepción y por esto, es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Apache Tomcat en su versión 6.0 posee soporte para *Unified Expression Language* 2.1. Fue Implementado de Servlet 2.5 y JSP 2.1 y también fue diseñado para funcionar en Java SE 5.0 y posteriores.

1.11 Conclusiones del capítulo.

En el presente capítulo se realizó un estudio profundo de algunos de los aspectos y conceptos relacionados con la integración de datos, además de las metodologías, herramientas y lenguajes existentes en el mundo para desarrollo de sistemas similares, lo cual aportará en gran medida con la construcción del proyecto en curso. Partiendo de la base teórica adquirida con la investigación realizada, se decide utilizar la metodología OpenUP para guiar el desarrollo del sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos. Además se utiliza el lenguaje de modelado UML en conjunto con la herramienta Visual Paradigm en su versión 8.0, para la implementación el lenguaje Java y el IDE Eclipse en su versión 3.5 y como servidor de aplicaciones Apache Tomcat en su versión 6.0.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA.

2.1 Introducción.

En este capítulo se describen los procesos de integración de datos, así como sus características, tendencias y el modelo del dominio. Se identifican también los requisitos funcionales y no funcionales. Se describen cada uno de los casos de uso existentes, el patrón arquitectónico a emplear y los patrones de diseño, así como los diagramas de clases del diseño y de secuencia respectivamente.

2.2 Procesos de integración de datos.

El término de integración de datos se refiere al proceso de combinación de datos procedentes de distintas fuentes heterogéneas. Posee la esencia de proporcionar una visión única y global de dichos datos combinados, donde el principal componente es el ETL (32). El proceso de ETL consiste en leer los datos de las diferentes fuentes existentes, para limpiarlos y transformarlos, añadiendo contextos y significados. Además con la creación de agregaciones y tablas resúmenes se efectúa la carga de los datos dándole salida a la fuente destino.

A continuación se listan las fases del proceso explicado anteriormente:

Extracción: proceso consistente en la adquisición de datos desde una o varias fuentes.

Transformación: proceso de acondicionamiento de la forma y/o contenido de los datos para encajarlos en la estructura de la(s) fuente(s) destino.

Carga: proceso de carga de los datos en la(s) fuente(s) destino.

2.3 Breve descripción del sistema.

El sistema para la gestión de los procesos de integración de datos en las soluciones de Almacenes de Datos, le proporcionará al grupo de trabajo de la línea de ETL ejecutar, detener, reiniciar y pausar de forma visual las rutinas de integración de la herramienta *Pentaho Data Integration*. Además de mostrar y modificar la planificación de dichas rutinas, permitiendo también desplegar nuevas soluciones y eliminar las rutinas obsoletas.

2.4 Especificación de los requisitos del sistema.

2.4.1 Requisitos funcionales.

Define las capacidades y funciones que un sistema debe ser capaz de realizar con éxito y especifican cómo debe comportarse el sistema en situaciones particulares (33).

A continuación se listan los requisitos funcionales identificados durante el proceso de análisis de la investigación en curso:

RF1- Autenticar Usuario.

RF2- Controlar Rutinas de Integración.

- **RF2.1-** Desplegar Rutinas de Integración.
- **RF2.2-** Ejecutar Rutinas de Integración.
- **RF2.3-** Detener Rutinas de Integración
- **RF2.4-** Pausar Rutinas de Integración.
- **RF2.5-** Reiniciar Rutinas de Integración.
- **RF2.6-** Eliminar Rutinas de Integración desplegadas.

RF3- Mostrar estado de ejecución de las Rutinas de Integración.

RF4- Mostrar historial de las Rutinas de Integración.

RF5- Planificar ejecución de las Rutinas de Integración.

Hasta aquí se identificaron diez requisitos funcionales donde uno de ellos está basado en el patrón de agrupamiento CRUD completo.

2.4.2 Requisitos no funcionales.

Los requisitos no funcionales (RnF) definen propiedades y restricciones del sistema, cualidades que el sistema debe tener aunque no formen parte de este (33).

Usabilidad

RnF1. Las interfaces del sistema deben estar en el idioma español y con un manual de usuario que ayude al buen entendimiento.

Soporte

RnF2. Una vez concluida la aplicación, se realizarán varias pruebas para comprobar su funcionalidad. Con la puesta en práctica de la aplicación deben quedar satisfechas las necesidades de los usuarios, mediante actualizaciones y mejoras a la misma.

Restricciones del Diseño

RNF3. Utilizar los lenguajes de programación definidos durante la investigación: Como lenguaje para la implementación del sistema se empleará Java, como tecnología web JSP. Como Framework se utilizará Spring y como servidor web Apache Tomcat en su versión 6.0.

Interfaces de Hardware

RNF4. La máquina donde radicará el servidor del sistema debe tener como mínimo 512 MB de RAM, Microprocesador Pentium IV 2.8 GHz o superior. Además debe contar con espacio libre en disco duro de 5 GB.

Interfaces de Software

RnF5. Se requiere para el funcionamiento de la aplicación, disponer en el servidor de la Máquina Virtual de Java 6.0. Los usuarios de la aplicación deberán contar con un navegador Internet Explorer 6+, Mozilla Firefox 2.0+.

Requisitos legales

RnF6. Las tecnologías y herramientas que se utilicen para desarrollar la aplicación web deben estar bajo la licencia de software libre.

Estándares Aplicables

RnF7. La aplicación permite ejecutarse en dos de los sistemas operativos existentes en la actualidad, Windows y Linux.

2.5 Actores y casos de uso del sistema.

Los actores son la representación de personas, software o hardware externos al sistema que interactúan con él. En el sistema que se describe se identificó el siguiente actor.

Tabla 1. Actor del sistema

Actor	Objetivo
Administrador.	El administrador es el usuario que tendrá la posibilidad de interactuar con el sistema, teniendo control total del mismo donde podrá utilizar las funcionalidades prestadas.

2.6 Definición de Caso de Uso (CU) del Sistema.

Un caso de uso es una secuencia de acciones realizadas por un sistema en respuesta a la interacción del actor y el mismo sistema. Estos casos de uso sirven para explicar mejor como responde un programa mediante la interacción del usuario y otros sistemas. Los casos de uso explicitan los requisitos funcionales del sistema relativos a uno de los objetivos del usuario (34).

2.6.1 Patrón de Caso de Uso utilizado.

CRUD

El patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

CRUD Completo

El patrón CRUD Completo (35) consiste en un caso de uso para administrar la información (CRUD Información), permite modelar las diferentes operaciones para administrar una entidad de información, tales como creación, lectura, actualización y eliminación.

A continuación se muestra un ejemplo de la utilización de CRUD completo con el caso de uso Controlar Rutinas de Integración.

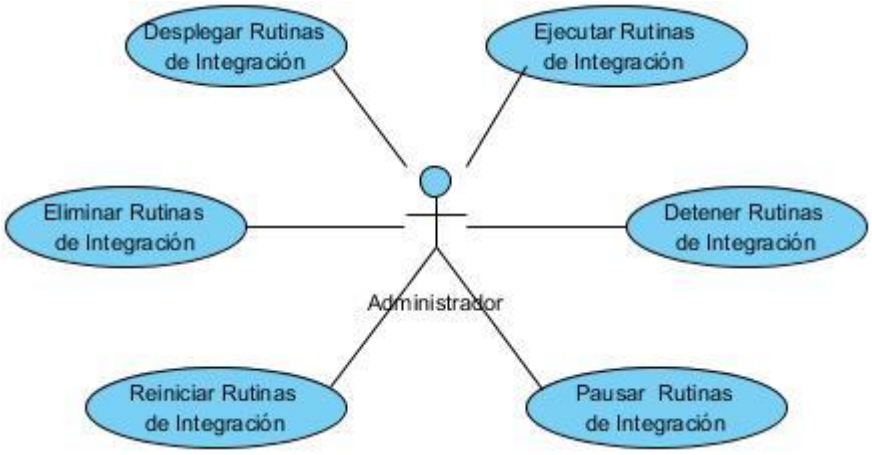


Figura 4. DCU Controlar Rutinas (antes de aplicar el patrón CRUD)

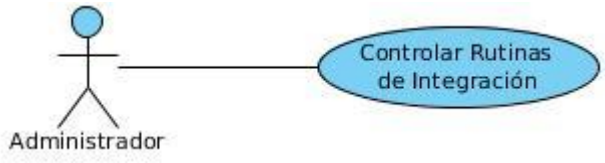


Figura 5. DCU Controlar Rutinas (después de aplicar el patrón CRUD)

2.6.2 Diagramas de Casos de Uso del Sistema.

Los diagramas de casos de uso (36) documentan el comportamiento de un sistema desde el punto de vista del usuario. Es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones, es decir, describe lo que hace un sistema desde el punto de vista de un observador externo. Estos diagramas muestran los distintos requisitos funcionales que se esperan de una aplicación o sistema y como se relaciona con su entorno (usuarios u otras aplicaciones).

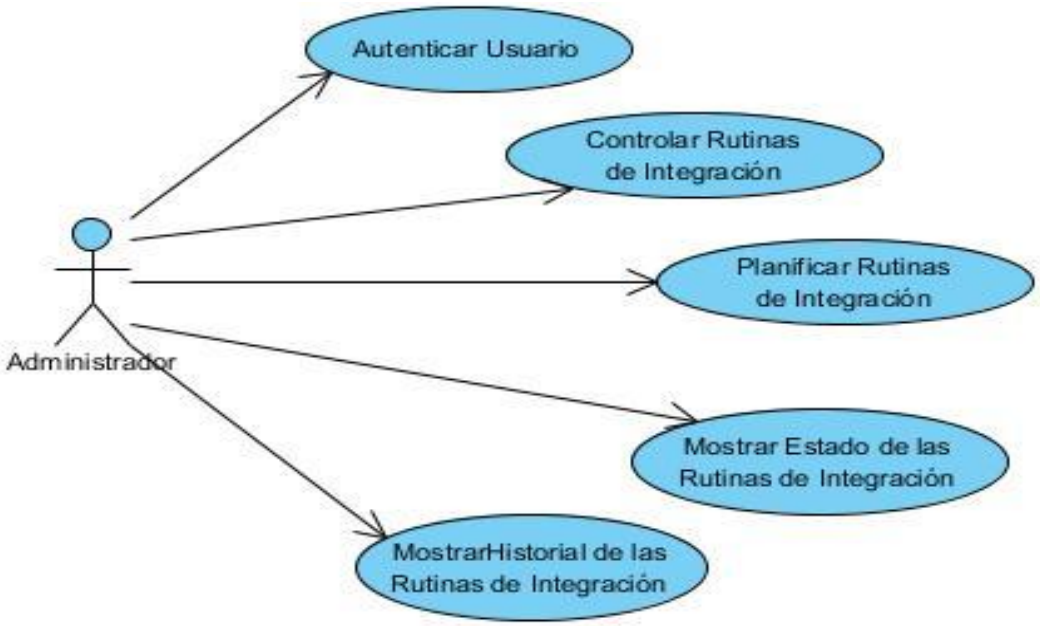


Figura 6. Diagrama de Caso de Uso del Sistema

2.6.3 Descripción de los Casos de Uso del Sistema.

Tabla 2. Caso de Uso < Controlar Rutinas de Integración>

Objetivo	El administrador debe ser capaz de desplegar, ejecutar, detener, reiniciar, pausar y eliminar las rutinas de integración.	
Actores	Administrador.	
Resumen	El caso de uso inicia cuando el administrador selecciona la solución desplegada.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario debe ser autenticado como Administrador.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Controlar rutinas de integración>		
	Actor	**Sistema**

		1.1 El sistema muestra una interfaz, con las opciones que el administrador puede realizar.
	El administrador selecciona la solución y elije la opción a realizar.	**2.1** El sistema muestra una respuesta con información asociada a la opción seleccionada.
		3.1 Termina el caso de uso
Flujos alternos		
	Actor	**Sistema**
4.	No existen rutinas en el sistema, el administrador tiene que desplegar una solución.	**4.1**El sistema muestra una interfaz con las opciones posibles a realizar. Regresa al punto 2
Prototipo de interfaz		
No aplica		

Las demás descripciones de los casos de uso críticos se encuentran en los anexos.

2.7 Patrones arquitectónicos.

Los patrones arquitectónicos son los que definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema (37).

2.7.1 Modelo Vista Controlador (MVC).

El patrón de arquitectura de software Modelo Vista Controlador separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos (Modelo, Vista y Controlador) (38).El Modelo, que son los objetos de la aplicación, también conocida como lógica de negocio, o lógica de aplicación. La Vista especifica la visualización de los datos, algunas veces conocida como lógica de presentación. El controlador es el coordinador entre estos dos últimos, es decir, define la forma en que la interfaz de usuario reacciona ante la entrada de usuario.

A continuación se explica con mayor profundidad las responsabilidades de cada elemento:

Modelo: comprende las funcionalidades para el acceso a la capa de almacenamiento de los datos, facilita la creación de las reglas de negocio, el registro de las vistas y controladores del sistema, además posibilita la notificación a las vistas de los cambios que en los datos produce un agente.

Controlador: responsable de captar los eventos de entrada, es decir un clic, cambio en un campo de texto, etc. Contiene una serie de elementos o reglas para la gestión de estos eventos. Estas operaciones pueden admitir peticiones al modelo o a la vista.

Vista: responsable de la visualización de los datos recibidos desde el modelo, contienen además un registro de su controlador asociado. Esta brinda el servicio de actualización, para que sea invocado por el controlador.

2.8 Patrones de diseño.

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (39).

2.8.1 Patrones de Software para la Asignación General de Responsabilidad.

Para la asignación de responsabilidades existen los llamados Patrones de Software para la Asignación General de Responsabilidad (GRASP, *Greedy Randomized. Adaptive Search Procedures*). Dentro de esta clasificación se encuentran los patrones Experto, Creador, Controlador, Alta cohesión, Bajo acoplamiento, Polimorfismo entre otros.

A continuación se explican algunos de los empleados en el desarrollo del sistema.

Experto: Consiste en asignar la responsabilidad de ejecutar una tarea a la clase que posee la información necesaria.

Creador: Consiste en crear una nueva instancia por la clase que, tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase, contiene o agrega la clase.

Alta cohesión: Consiste en asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase.

2.9 Diagramas de clases de Diseño.

Para cada uno de los casos de uso se construyó un diagrama de clases del diseño, donde se muestran las relaciones entre las clases del diseño. A continuación se muestra el CU Controlar Rutinas de Integración. Para ver los restantes CU críticos consultar el Anexo.

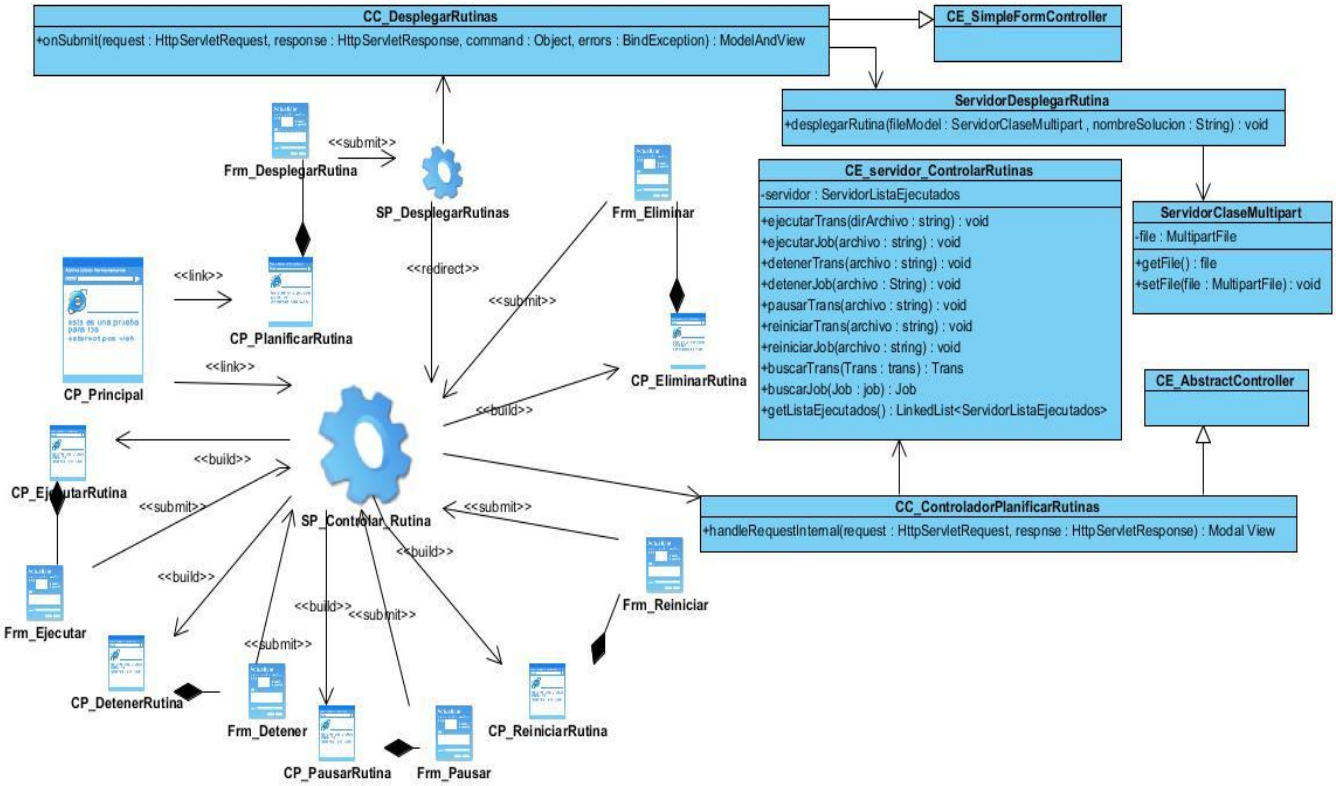


Figura 7. Diagrama de Clases del Diseño. CU Controlar Rutina

Los demás diagramas de clases del diseño se encuentran en los anexos.

2.10 Diagramas de interacción.

Los diagramas de interacción son utilizados para el modelado de los aspectos dinámicos de un sistema. Existen dos tipos de diagrama de interacción:

- ✓ Diagramas de secuencia.
- ✓ Diagramas de colaboración.

El diagrama de secuencia muestra las interacciones entre un conjunto de objetos y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad prevista. Mientras que el diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes (40).

Para cada uno de los escenarios de cada caso de uso se construyó un diagrama de secuencia. A continuación se muestra el escenario controlar rutinas. Para ver otros escenarios de los restantes CU consultar el Anexo.

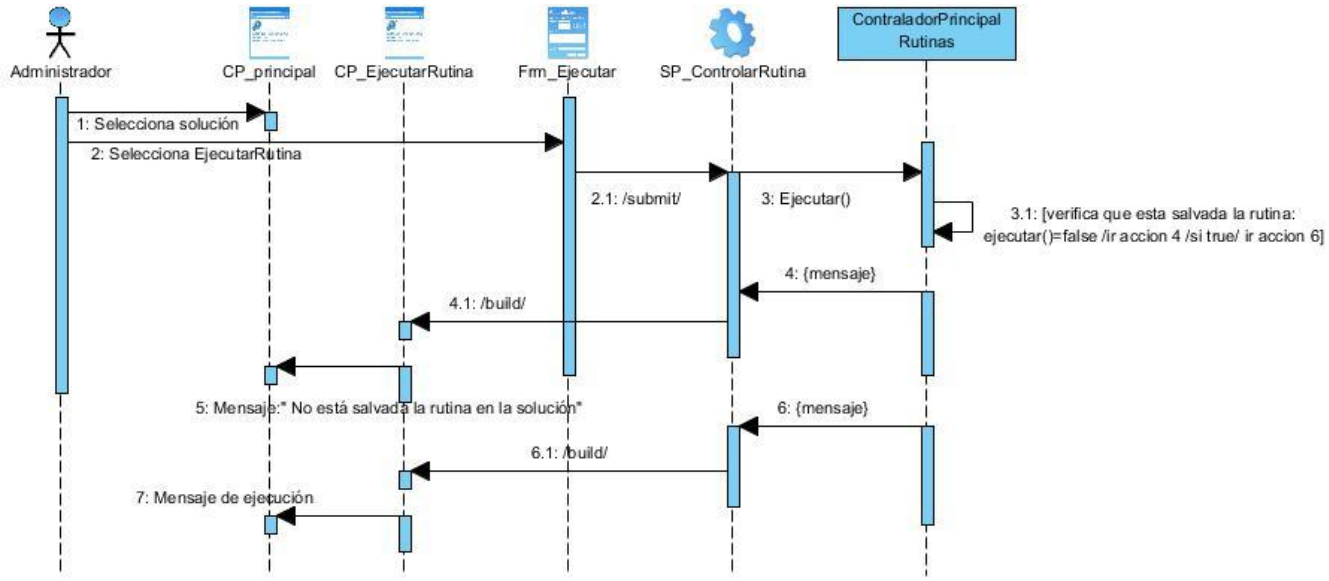


Figura 8. Diagrama Secuencia (CU Controlar Rutina, Escenario Ejecutar Rutina)

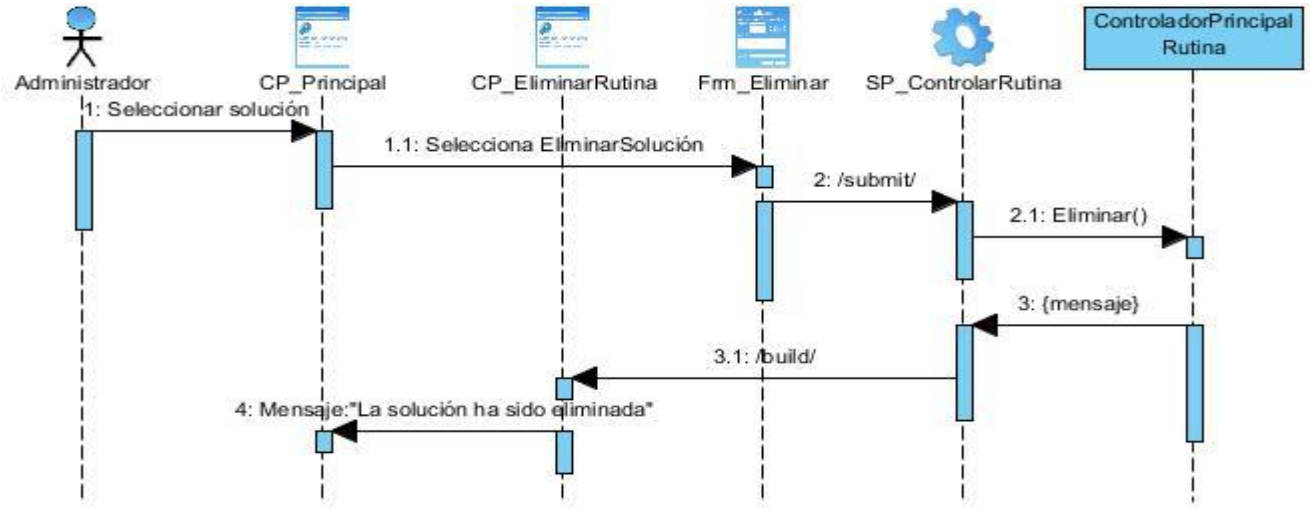


Figura 9. Diagrama Secuencia (CU Controlar Rutina, Escenario Eliminar Rutina)

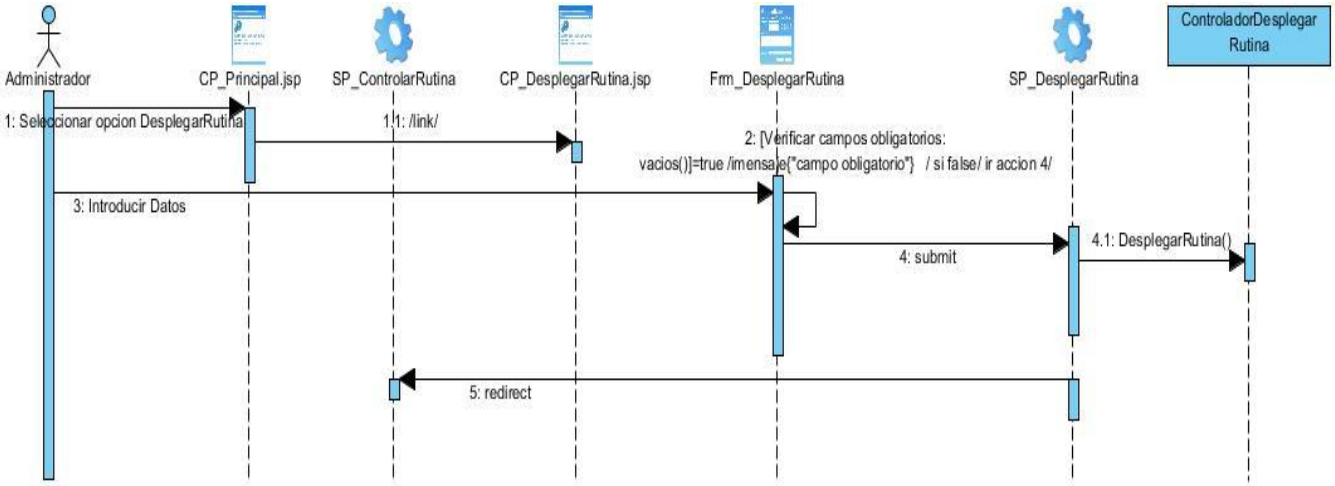


Figura 10. Diagrama Secuencia (Controlar Rutina, Escenario Desplegar Rutina)

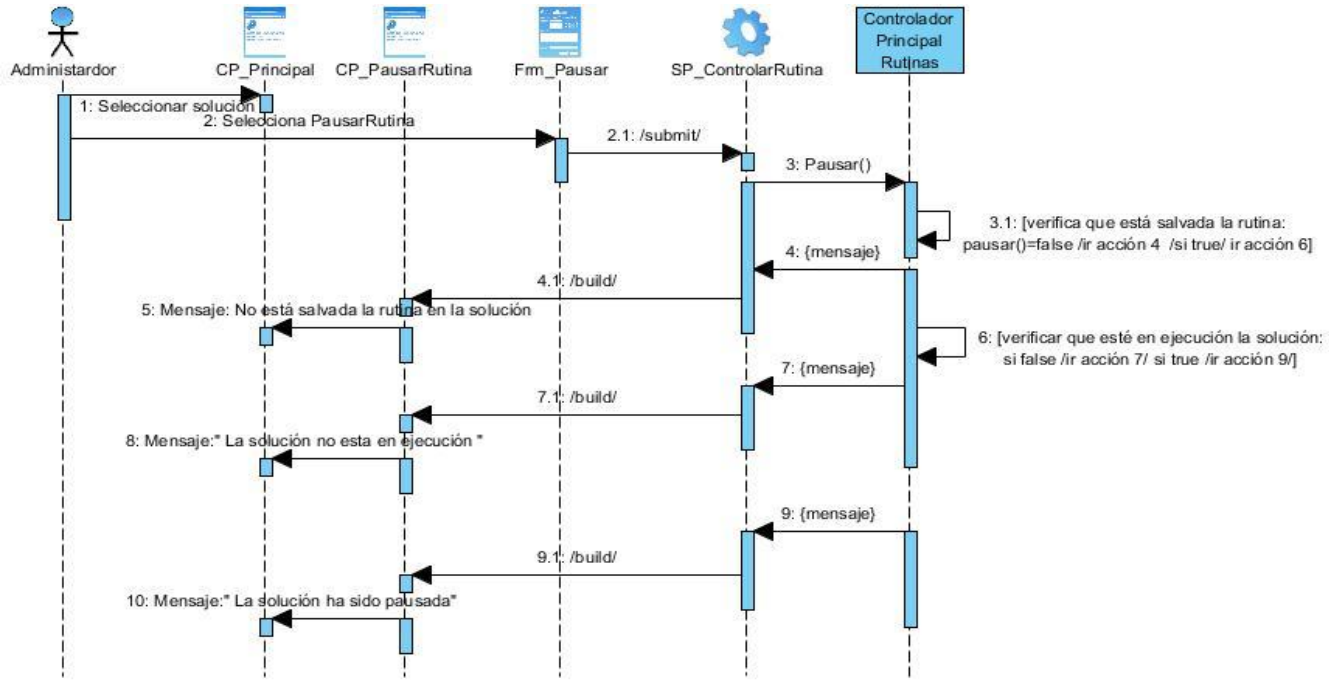


Figura 11. Diagrama Secuencia (CU Controlar Rutina, Escenario Pausar Rutina)

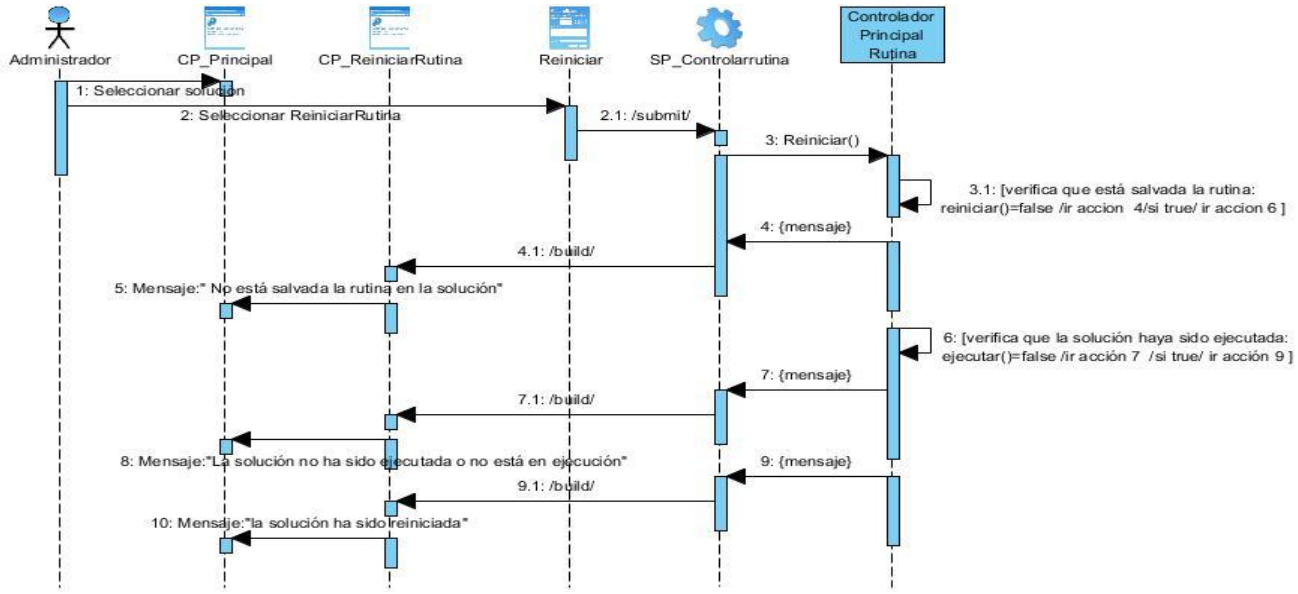


Figura 12. Diagrama Secuencia (CU Controlar Rutina, Escenario Reiniciar Rutina)

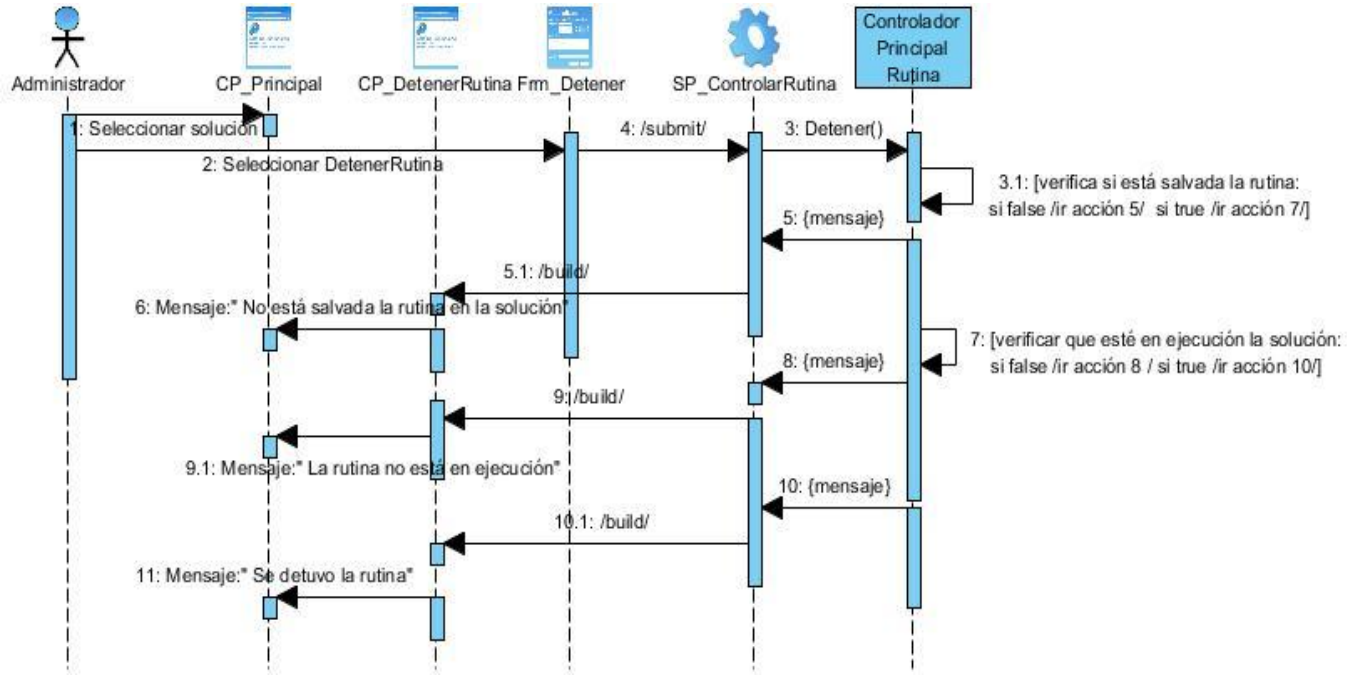


Figura 13. Diagrama Secuencia (CU Controlar Rutina, Escenario Detener Rutina)

Los demás diagramas de secuencia se encuentran en los anexos.

2.11 Conclusiones del capítulo.

En el presente capítulo, se realizó un exhaustivo análisis del sistema, el cual dio pie al correcto diseño del mismo, proporcionando a los desarrolladores una base fundamental para lograr el cumplimiento del objetivo principal del problema en cuestión. El análisis realizado también permitió la identificación de los patrones de diseño, los cuales estructuran la composición de las clases, así como las funcionalidades que tendrán que realizar.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción.

Una vez terminado el análisis y el modelo del diseño se consta con los detalles necesarios para la construcción del sistema, para luego proceder con las pruebas necesarias para la liberación del mismo. En este capítulo se realiza el modelo de implementación del módulo básico, separados en los diagramas de componentes de los requisitos más importantes, se determinan los tipos de prueba que se realizan, así como los casos de pruebas que se le aplican al sistema.

3.2 Modelo de Despliegue.

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe como se implementan los componentes, agrupándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre estos. Para representar los diagramas del Modelo de Implementación se puede emplear el diagrama de UML de Componentes.

3.2.1 Diagrama de Componentes.

Un diagrama de componentes muestra las dependencias lógicas entre componentes software, sean estos componentes fuentes, binarios o ejecutables. Los componentes software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución. El diagrama de componentes muestra un conjunto de ficheros relacionados entre sí para lograr una completa funcionalidad del sistema.

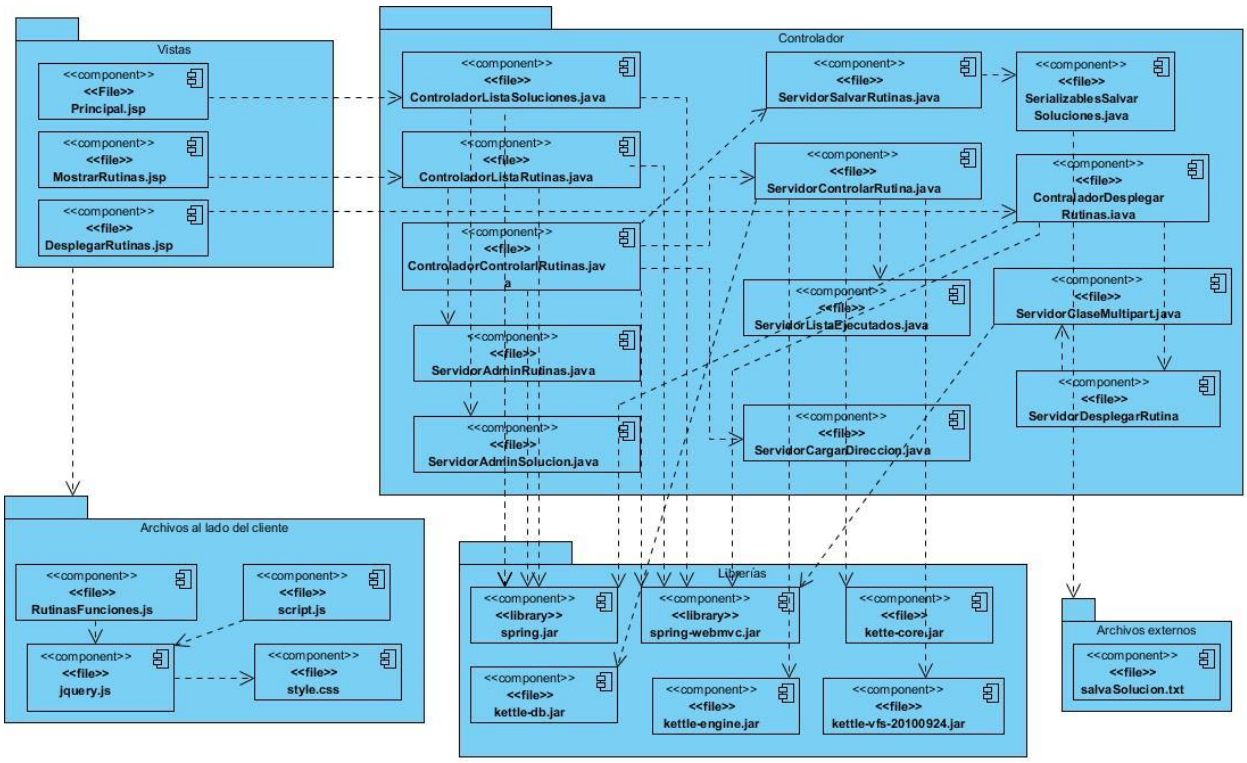


Figura 14. Diagrama de Componentes. CU Controlar Rutina

3.3 Diagrama de Despliegue.

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

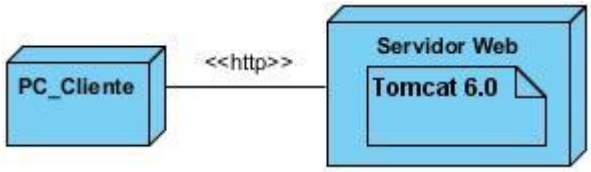


Figura 15. Diagrama de despliegue

A continuación se explica el uso de cada nodo.

- ✓ La **PC Cliente** a través del protocolo HTTP es la encargada de acceder al servidor web mediante un navegador.
- ✓ El **Servidor Web** Apache Tomcat 6.0 es el encargado de darle respuesta a las peticiones del cliente y es donde se encuentra la aplicación.

3.4 Framework de Desarrollo.

3.4.1 JQuery

JQuery es una biblioteca JavaScript rápido, pequeño y rico en funciones (41). Permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web. Además soporta extensiones, posee varias utilidades como obtener información del navegador, opera con objetos y vectores, es compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 9+ y Google Chrome (42).

3.5 Estándares de Codificación.

Los estándares de codificación no son más que normas a tener en cuenta cuando se escribe el código fuente, de tal forma que se facilite el entendimiento a otros programadores de dicho código.

En particular el de Java, tiene reglas como: Las clases inician en mayúsculas, los atributos y métodos inician con minúsculas, las constantes son todas en mayúsculas, entre otros. Para el desarrollo del sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos se tuvieron en consideración muchos de estos, a continuación se evidencia una muestra de ellos.

Comentarios.

Estos permiten describir el código fuente con un lenguaje común para un mejor entendimiento. Así como deshabilitar código que no se utilice. Estos se muestran de la siguiente forma (// o /**/).

```
// Para comentar una sola línea.

/* Para comentar varias líneas. */
```

Declaración de variables.

Las variables serán declaradas en una línea y su primera letra será minúscula, si es compuesta la primera palabra inicia con letra minúscula y la segunda con letra mayúscula.

```
public int contador;

int contadorTrans;
```

Declaración de clases.

Las clases tendrán la particularidad de que comiencen con el nombre del paquete al cual pertenecen seguido de un guion bajo y la próxima letra en mayúscula.

```
public class ServidorControlarRutinas {

}

public class ServidorTareaEjecutar extends ServidorControlarRutinas {

}
```

Declaración de métodos.

Los métodos comenzarán siendo compuestos o no con letra minúscula.

```
privateint contarRutinas(){

}
```

```
Int controlarRutinas(){

}
```

Declaración de atributos.

Los atributos se declararán siendo compuestos o no, comenzando con minúscula la primera letra.

```
private int contador;

int contadorRutina;
```

3.6 Pruebas de Software.

Cuando un sistema está terminado, se hace necesario determinar el número de errores que el mismo presenta, es por eso que se diseñan pruebas para corregir la mayor cantidad posible de los inconvenientes encontrados. Con el fin de entregar al cliente un producto de alta calidad, se diseñan una serie de casos de pruebas que tengan una gran probabilidad de encontrar errores. Para la elaboración de estos casos de pruebas, existen un conjunto de técnicas, las cuales brindan una guía para comprobar la lógica interna de los componentes y que verifiquen los datos de entrada y salida del sistema, con el propósito de detectar errores y comprobar el rendimiento del mismo (PRESSMAN, 2005). Las pruebas realizadas son las encargadas de determinar el grado de conformidad de los clientes, por lo que es recomendable que estas sean ejecutadas por un conjunto de especialistas, para determinar el mayor número de errores, que en un final es el principal objetivo. Existen dos tipos de pruebas fundamentales que son las de Caja Blanca y las de Caja Negra.

3.6.1 Niveles de Prueba.

Los niveles de prueba son diferentes ángulos de verificar y validar un producto de software. Existen diferentes niveles que se le aplican al producto, entre ellas se mencionan algunas:

Pruebas Unitarias: Ese tipo de pruebas se le realiza a pequeñas unidades o fragmentos de código, que no son más que los métodos o funciones del sistema. Estas pruebas revela al igual que otras la presencia de errores en el sistema, pero no informa de errores en la integración de las partes, ni errores de rendimiento o problemas derivados del sistema sobre el que está ejecutándose el programa. El objetivo de

las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Pruebas de sistema: Este tipo de pruebas se realiza cuando el probador no tiene acceso al código fuente, que es lo más frecuente. Estas pruebas se realizan en muchos casos para estudiar el comportamiento del sistema bajo determinada condiciones.

Pruebas de integración: En este caso se prueba el sistema por módulos con el objetivo de verificar si su integración es correcta y funciona según lo planeado. Este tipo de prueba tiene la peculiaridad de ser incremental puesto que se incrementa un nuevo módulo al terminar de comprobar el anterior.

Pruebas de aceptación: Las pruebas de aceptación están muy relacionadas con el cliente, ya que este es el que da el visto bueno del sistema, emitiendo un criterio del producto lo cual ofrece un indicador de su satisfacción.
Para el proceso de software que validará la herramienta se realizarán pruebas unitarias y pruebas de sistema.

3.6.2 Métodos de prueba de Caja Blanca.

Son pruebas estructurales. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones. (PRESSMAN, 2005)Uno de los métodos de prueba de caja blanca es el camino básico, la cual consiste en calcular la complejidad ciclomática de un fragmento de código. 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 facilita un límite superior para el número de pruebas que se deben realizar. Esta complejidad se puede calcular de tres formas:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V (G), de un grafo de flujo G, se define como: V (G)=A-N+2.
  Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V(G) = P +

Donde P es el número de nodos predicado5contenidos en el grafo de flujo G.

3.6.3 Métodos de prueba de Caja Negra.

Las pruebas de Caja Negra son pruebas funcionales. Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro (Caja negra). Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación (PRESSMAN, 2005).

Una de las técnicas más usadas en pruebas de caja negra es la de partición equivalente, la cual divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

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

3.6.4 Casos de prueba.

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados obtenidos son observados, registrados y finalmente se realiza una evaluación (PRESSMAN, 2005).

3.6.5 Aplicando pruebas de caja blanca.

Como se mencionó en el epígrafe 3.6.1 unas de las técnicas de caja blanca es la del camino básico, que se le aplica a un fragmento de código de la aplicación. En este caso se seleccionó la funcionalidad que lista las soluciones en el sistema.

Para la funcionalidad “listaSoluciones”, se identificaron los bloques de ejecución y se enumeraron, como se muestra en la figura 16. Se obtuvo siete bloques identificados por nodos, así como el camino básico

mostrado en la figura 17, donde se observa a simple vista los nodos predicados 2,4, y 5 en bordes blancos, que son aquellos que se derivan en más de un camino.

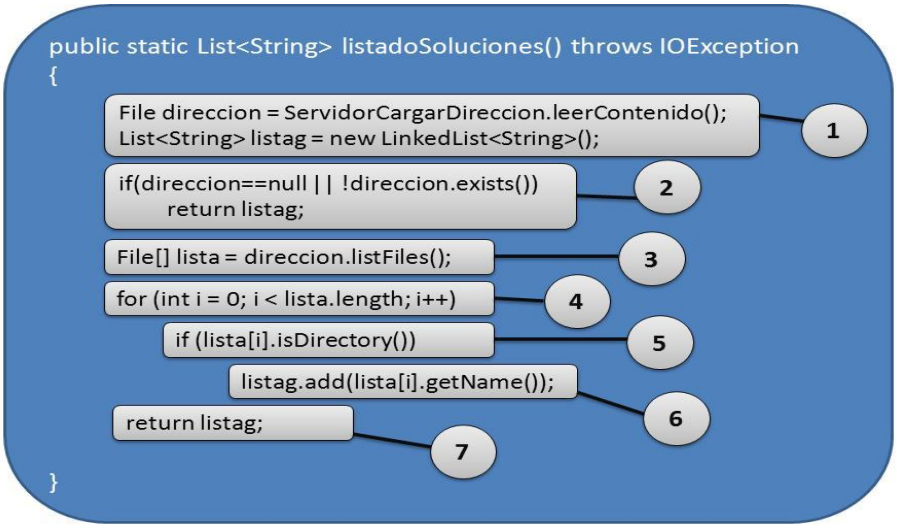


Figura 16. Función que lista las soluciones existentes en el sistema. Está dividido por bloques enumerados que representan los nodos del camino básico.

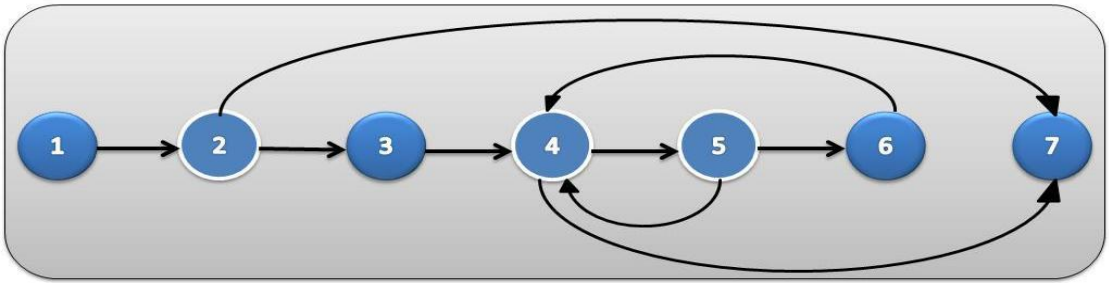


Figura 17. Camino básico obtenido a partir de la figura 16. Los nodos con bordes son los nodos predicados.

Una vez obtenido el camino básico se procede a determinar la complejidad ciclomática con una de las tres formas existentes, se utilizó V (G) = P + 1, para la cual se obtuvo tres predicados, por tanto:

V (G) = 3+1, quedando V (G) = 4. De la misma manera se puede comprobar que las otras estrategias de complejidad arriban al mismo resultado.

3.6.6 Aplicando casos de prueba de caja negra.

Para aplicar casos de pruebas de caja negra al sistema se determinaron las entradas, condiciones de ejecución y las salidas con los resultados esperados.

Primeramente se definen las variables para comenzar el caso de prueba, a continuación se muestran dos variables para el caso de prueba Desplegar Rutina.

Nota: Consultar los anexos para los demás casos de pruebas del caso de uso Controlar Rutinas de Integración.

Tabla 3. Variables para el caso de prueba Desplegar Rutina.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	nombreSolucion	string	No	El usuario introduce el nombre de la solución que se va a desplegar en el sistema.
2	subirRutina	string	No	El usuario introduce la dirección del fichero que se incluirá en la solución.

Tabla 4. Caso de prueba 1: Desplegar Rutina

Escenario	Descripción	nombreSolucion	subirRutina	Respuesta del sistema	Flujo central
EC 1.1 Subir solución al servidor correctamente.	En este escenario el administrador sube la solución al servidor correctamente.	V	V	El sistema muestra una interfaz para introducir el nombre de la solución y el directorio donde se encuentra la solución.	El administrador se autentica, selecciona en la parte superior izquierda de la aplicación la opción: Subir solución al servidor. Introduce el nombre de la solución correctamente y selecciona el archivo con la extensión correcta.

EC 1.2 Subir solución al servidor incorrectamente	En este escenario el administrador sube la solución al servidor incorrectamente.	I	V	El sistema muestra una interfaz para introducir el nombre de la solución y el directorio donde se encuentra la solución. El sistema muestra un mensaje de error ("Dejó campos vacíos") o ("El nombre de la solución no permite: /:<>\?|").	El administrador se autentica, selecciona en la parte superior izquierda de la aplicación la opción: Subir solución al servidor. Introduce el nombre de la solución incorrectamente o lo deja vacío y selecciona el archivo con la extensión correcta.
EC 1.3Subir solución al servidor incorrectamente	En este escenario el administrador sube la solución al servidor incorrectamente.	V	I	El sistema muestra una interfaz para introducir el nombre de la solución y el directorio donde se encuentra la solución. El sistema muestra un mensaje de error ("Dejó campos vacíos") o ("El fichero debe tener extensión: ktr, kjb o zip").	El administrador se autentica, selecciona en la parte superior izquierda de la aplicación la opción: Subir solución al servidor. Introduce el nombre de la solución correctamente y selecciona el archivo con la extensión incorrecta o deja el campo vacío.

La aplicación de los casos de prueba permitió verificar el cumplimiento de los requisitos funcionales del sistema, donde se identificaron once no conformidades, de ellas cuatro de complejidad baja, seis de complejidad media y una de complejidad alta, las cuales fueron corregidas, permitiendo una mayor robustez del sistema en el manejo de errores.

3.7 Vistas del Sistema.

Figura 18. Interfaz Principal.

En la pantalla principal el usuario cuenta con varias opciones, las que le permitirán controlar las rutinas de integración, desplegar nuevas soluciones, planificarlas, entre otras, en la barra superior se ubican los botones que ejecutan las acciones antes mencionadas, en el área izquierda se muestra un listado de las soluciones desplegadas en el sistema.

Figura 19. Interfaz para desplegar una solución.

El usuario cuenta con una pequeña ventana la cual le permite desplegar una nueva solución en el sistema, donde los campos a llenar son obligatorios.

Figura 20. Interfaz para planificar una solución.

El usuario cuenta con una vista de planificaciones, la cual permite tres tipos de planificaciones, diariamente, semanal o en una fecha determinada. En esta ventana todos los campos son obligatorios, con la peculiaridad de que en la planificación por fecha esta no puede ser menor que la actual.

3.8 Conclusiones del Capítulo.

En este capítulo se plasma el proceso de implementación del sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos. El uso de estándares de codificación como CamelCase, permitió contar con un código legible y entendible para otros desarrolladores ya que es un estilo de escritura muy usado en el mundo de la programación Java. Con el estudio de los tipos de pruebas se pudo elegir el más adecuado para aplicar al sistema y con su aplicación se logró obtener el mayor cúmulo de información en cuanto a rendimiento del mismo así como detectar el mayor número posible de errores, que es el objetivo fundamental de la etapa de pruebas.

CONCLUSIONES

Ya culminada la investigación se puede afirmar que se concluyó la construcción del sistema para la gestión de los procesos de integración en las soluciones de Almacenes de Datos. Para su desarrollo:

- ✓ Se definió la implementación de una interfaz web del servidor encargado del control de las rutinas de integración, lo cual facilita el acceso al sistema ya que no tiene que estar desplegado en la PC del usuario.

- ✓ Se definieron los requisitos funcionales del sistema y se agruparon en cinco casos de uso, donde se identificó como crítico el CU *Controlar rutinas de integración,* empleando para la confección de la arquitectura del sistema varios patrones como MVC, que fueron aplicados a los diagramas de interacción y del modelo de diseño.

- ✓ Se implementaron los requisitos funcionales definidos por el cliente, logrando un producto capaz de desplegar, ejecutar, planificar las rutinas de integración, además de mostrar la información asociada al estado de las rutinas desplegadas en el sistema.

- ✓ Se comprobó el correcto funcionamiento de los requisitos a través de las pruebas aplicadas al sistema, las cuales permitieron encontrar inconformidades que condujeron al mejoramiento del mismo.

RECOMENDACIONES

- ✓ En próximas versiones de la aplicación, integrarlo a otros sistemas existentes en el departamento de Almacenes de Datos.
- ✓ Realizar el control de las soluciones desplegadas en un repositorio.
- ✓ Mejorar el tiempo de respuesta de la ejecución de la primera solución.
- ✓ Visualizar el estado y la planificación de las soluciones en la misma área de trabajo.

REFERENCIAS BIBLIOGRÁFICAS

1. *Diccionario de Direccion de Empresas y Marketing.* **Castillo, Miguel Angel Sastre.** 3-28015, Madrid, España : Editorial del economista, 2009, Vol. Volumen 8.

2. **Héctor, A. García.** Proyecto Salón Hogar. *Proyecto Salón Hogar.* [En línea] [Citado el: 4 de Diciembre de 2012.] http://www.proyectosalonhogar.com/Tecnologia/La_informatica.htm..

3. *Comisión Económica para América Latina y el Caribe (CEPAL). Desafíos y oportunidades de la industria del software en América Latina.* 2009. ISBN: 978-958-8307-56-5.

4. **Rafael Menéndez, Barzanallana Asencio.** Metodologías de Desarrollo. *Metodologías de Desarrollo.* [En línea] 13 de octubre de 2011. [Citado el: 20 de enero de 2013.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

5. *Ingeniería del Software: Metodologías y Ciclos de Vida. Laboratorio Nacional de Calidad del Software del Instituto Nacional de Tecnologías de la Comunicación (INTECO). .* 2009.

6. *Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.* **Lianet Cabrera González, Enrique Roberto Pompa Torres.** 10, La Habana : Serie Científica de la Universidad de las Ciencias Informáticas, 2012, Vol. V. ISSN/RNPS.

7. *Modelado de Sistemas com UML. Popkin Software and Systems.* 2001, Vol. 4. ISSN/RNPS.

8. *Herramienta para la migración de datos del Sistema de Inventario Participativo al ASSETS.* **López, Meisbel Daudinot.** La Habana : s.n, 2004. ISSN/RNPS.

9. Alipso. *Alipso.* [En línea] 1 de diciembre de 2012. [Citado el: 20 de diciembre de 2012.] http://www.alipso.com/monografias/desarrollo_de_sistemas_de_informacion.

10. Software. *Software.* [En línea] 2 de enero de 2011. [Citado el: 3 de febrero de 2013.] http://www.software.com.ar/visual-paradigm-para-uml.html.

11. codigoprogramacion. *codigoprogramacion.* [En línea] 112 de agosto de 2011. [Citado el: 12 de diciembre de 2012.] http://codigoprogramacion.com/java/47-introjava.html.

12. Oracle. *Oracle.* [En línea] 20 de julio de 2011. [Citado el: 10 de diciembre de 2012.] http://www.oracle.com/technetwork/java/javaee/jsp/index.html.

13. *Herramientas de Desarrollo de Software: Hacia la Construccion de una Ontologia.* **Lornel A. Rivas, María Pérez, Luis E. Mendoza y Anna Grimán.** Caracas, Venezuela : s.n.

14. eclipse. *eclipse.* [En línea] 25 de enero de 2011. [Citado el: 5 de diciembre de 2012.] http://www.eclipse.org/org.

15. **Facultad de Ingenieria. Departamento de Ingenieria de Sistemas y Computacion.** Proyecto Cupi2. *Proyecto Cupi2.* [En línea] [Citado el: 8 de enero de 2013].

16. Netbeans. *Netbeans.* [En línea] 25 de diciembre de 2012. [Citado el: 5 de enero de 2013.] http://www.netbeans.org.

17. DataPrix. *DataPrix.* [En línea] 5 de enero de 2010. [Citado el: 23 de noviembre de 2012.] http://www.dataprix.com/blogs/respinosamilla/herramientas-etl-que-son-para-que-valen-productos-mas-conocidos-etl-s-open-sour.

18. El Rincon del BI . *El Rincon del BI.* [En línea] [Citado el: 10 de diciembre de 2012.] http://churriwifi.wordpress.com/2010/05/10/16-3-construccion-procesos-etl-utilizando-kettle-pentaho-data-integration.

19. **Talen Open Data Solutions.** Talend. *Talend.* [En línea] [Citado el: 29 de enero de 2013.] http://www.talend.com.

20. El Rincón del BI. *El Rincón del BI.* [En línea] [Citado el: 25 de enero de 2013.] http://churriwifi.wordpress.com/category/talend.

21. CodeBox. *CodeBox .* [En línea] [Citado el: 15 de septiembre de 2010.] http://www.codebox.es/glosario.

22. Genbeta:dev. *Genbeta:dev.* [En línea] 15 de junio de 2011. [Citado el: 9 de diciembre de 2012.] http://www.genbetadev.com/java-j2ee/spring-framework-introduccion.

23. SpringSource. *SpringSource.* [En línea] 12 de marzo de 2009. [Citado el: 12 de diciembre de 2012.] http://static.springsource.org/spring-security/site.

24. SpringSource. *SpringSource.* [En línea] 12 de enero de 2012. [Citado el: 2 de marzo de 2013.] http://static.springsource.org/spring/docs/2.0.x/reference/mvc.htm.

25. **The Apache Software Foundation.** Struts. *Struts.* [En línea] 10 de enero de 2013. [Citado el: 7 de febrero de 2013.] http://struts.apache.org.

26. *Aplicaciíon Web para la Gestión de Explotaciones Agrícolas.* **Gallardo, Antonio Ramírez.** Málaga : s.n, 2010, Vol. 4.

27. *Java Struts Framework.* **Hervella, Juan F. Rodríguez.** Madrid : s.n, 2010.

28. duplika . *duplika.* [En línea] 2 de marzo de 2012. [Citado el: 11 de diciembre de 2012.] http://www.duplika.com/blog/que-son-los-servidores-web-y-por-que-son-necesarios.

29. El Club del Programador. *El Club del Programador.* [En línea] [Citado el: 8 de diciembre de 2012.] http://www.elclubdelprogramador.com/2012/02/16/jetty-jetty-un-servidor-ligero-y-embebido-en-tus-aplicaciones.

30. Apache Tomcat. *Apache Tomcat.* [En línea] 2 de enero de 2011. [Citado el: 12 de diciembre de 2013.]http://tomcat.apache.org.

31. Oracle. *Oracle.* [En línea] 20 de julio de 2011. [Citado el: 10 de diciembre de 2012.] http://www.oracle.com/technetwork/java/javaee/jsp/index.html.

32. *Mejora de las aplicaciones de business intelligence con MicroStrategy Desktop. MicroStrategy.* 09450720, s.l. : s.n, 2002, Vol. 3.

33. *MetodologíaRUP(Rational UnifiedProcess).* **Vilma Quispe Carita, Dante Harry Huamantuco Solorsano , Jose Luis Vargas Yupanqui.** Puno, Peru : s.n, 2011.

34. **Santiago, Ceria.***Casos de Uso. Un Método Práctico para Explorar Requerimientos. .*

35. SG. *SG.* [En línea] 2013. [Citado el: 30 de febrero de 2013.] http://sg.com.mx/content/view/510.

36. *Diagramas de Caso de Uso.* **Tello, Jesús Cáceres.** Madrid : s.n, 2013.

37. Comusoft. *Comusoft.* [En línea] 5 de febrero de 2011. [Citado el: 25 de enero de 2013.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

38. **Erica Camacho, Fabio Cardeso, Gabriel Nuñez FABIO, Cardeso Gabriel Nuñez.** *ArquitecturasdeSoftware .* 2004.

39. *Patrones de diseño: ejemplo de aplicación en los Objetos de Aprendizaje Generativos.* **Tello, Jesús Cáceres.** s.l. : s.n, 2011.

40. Ecured. *Ecured.* [En línea] 8 de marzo de 2011. [Citado el: 25 de marzo de 2013.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

41. jquery. *jquery.* [En línea] 25 de marzo de 2012. [Citado el: 10 de febrero de 2013.] http://www.jquery.com.

42. Ecured. *Ecured.* [En línea] 2 de marzo de 2011. [Citado el: 10 de febrero de 2013.] http://www.ecured.cu/index.php/JQuery.

BIBLIOGRAFÍA

- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software. Madrid:* Addison Wesley, 2000. ISBN: 8478290362.
- **Larman, Craig.** *UML y Patrones.* México: Prentice Hall, 1999. ISBN: 9701702611.
- **Hall, Marty y Brown, Larry.** *Core Servlets and JavaServer Pages.* s.l: Prentice Hall, 004. ISBN: 3827266459.
- **Walls, Craig y Breidenbach, Ryan.** *Spring in Action.* Segunda Edición. s.l: Manning Publications, 2008. ISBN: 1933988134.
- **Pressman, Roger S. 2005.** Ingienería del Software. Un enfoque Práctico. 2005.

GLOSARIO DE TÉRMINOS

ETL: Extracción, Transformación, Carga.

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

dashboards: cuadro de mando digital utilizado en la capa de visualización.

Business Intelligence: Inteligencia de Negocio.

Object Relational Mapping: Mapeo Relacional de Objetos.

JSF (JavaServer Faces): tecnología java que se emplean en aplicaciones web.

XLST: Transformaciones XLS, permite transformar documentos XML.

Velocity: proyecto de la fundación Apache, encargada de la creación y mantenimiento de software de código abierto.

STRATUS: tecnología java.

SOA: Arquitectura Orientada a Servicios.

Ruby: lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla.

Ruby on Rails: es un Framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).

JRuby: es una implementación del lenguaje de programación Ruby, pero para la JVM. Se puede decir que es un intérprete de Ruby escrito en Java, que se ejecuta sobre la máquina virtual.

CamelCase: estándar de codificación.

PHP: Preprocesador de Hipertexto, es un lenguaje de código abierto, originalmente diseñado para ser usado en el desarrollo de aplicaciones web.

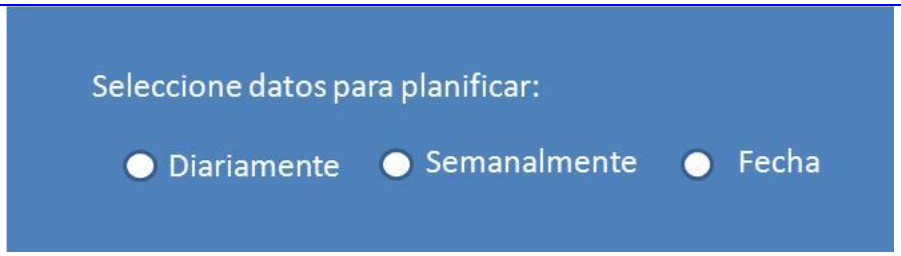
Swing GUI: librería de interfaz, proporciona una paleta de componentes para el trabajo con formularios y ventanas.

Unified Expression Language: (Expresión Unificada del Lenguaje) es un lenguaje de programación de propósito especial que se utiliza sobre todo en aplicaciones web de Java para incrustar expresiones en páginas web.

ANEXOS

Tabla 5. Especificación de Casos de Uso < Planificar Rutina de integración >

Objetivo	Este caso de uso le permite al administrador del sistema, hacer una planificación de las rutinas de integración.	
Actores	Administrador.	
Resumen	El caso de uso inicia cuando el administrador selecciona la solución desplegada.	
Complejidad	Alta.	
Prioridad	Crítico.	
Precondiciones	El usuario se ha autenticado.	
Postcondiciones	Existen soluciones desplegadas en el sistema.	
Flujo de eventos		
Flujo básico <Planificar rutinas de integración>		
	Actor	**Sistema**
1.	El administrador selecciona la solución que desea planificar.	**1.1** El sistema muestra una interfaz, con los tipos de planificación existentes.
2.	El usuario selecciona el tipo de planificación que desea realizar.	**2.1** El sistema muestra un interfaz con los campos a llenar.
3.	El usuario llena los campos correspondientes y presiona el botón aceptar.	**3.1** El sistema muestra un mensaje de confirmación de la planificación.
4.		**4.1** Termina el caso de uso.
Flujos alternos		
	Actor	**Sistema**

5.	El administrador selecciona una solución que no está configurada.	**5.1** El sistema muestra un mensaje de error y regresa al punto 1.
6.	El administrador selecciona una solución que ya está planificada.	**6.1** El sistema muestra una interfaz con los datos te la planificación existente para que el usuario modifique si desea los datos y regresa al punto 3.
7.	El usuario deja campos vacíos, o introduce datos incorrectos.	**7.1** El sistema muestra un mensaje de error y regresa al punto 3.
Prototipo de Interfaz		
		

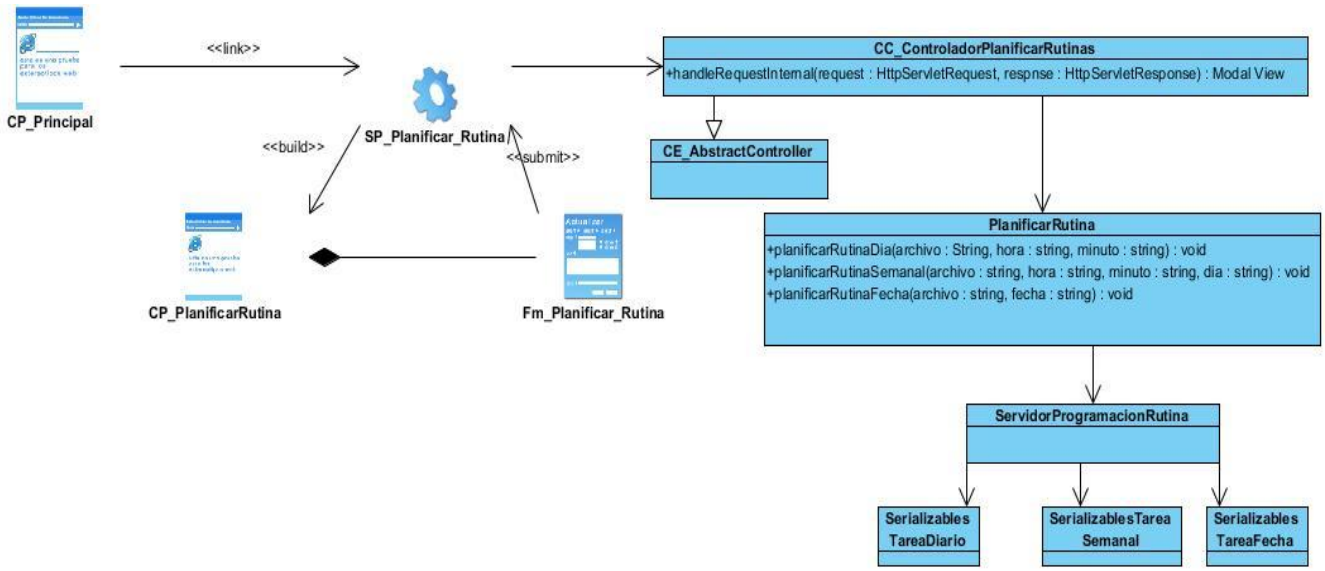


Figura 21. Diagrama de clases del diseño. CU Planificar Rutina.

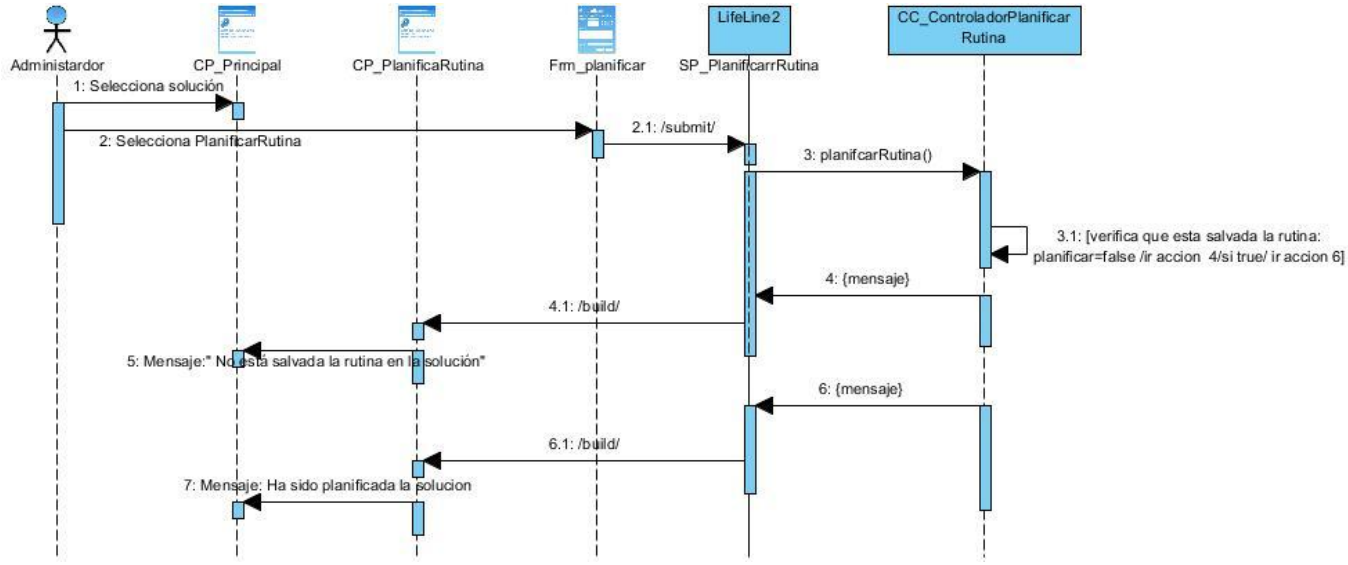


Figura 22. Diagrama Secuencia CU. Planificar Rutina

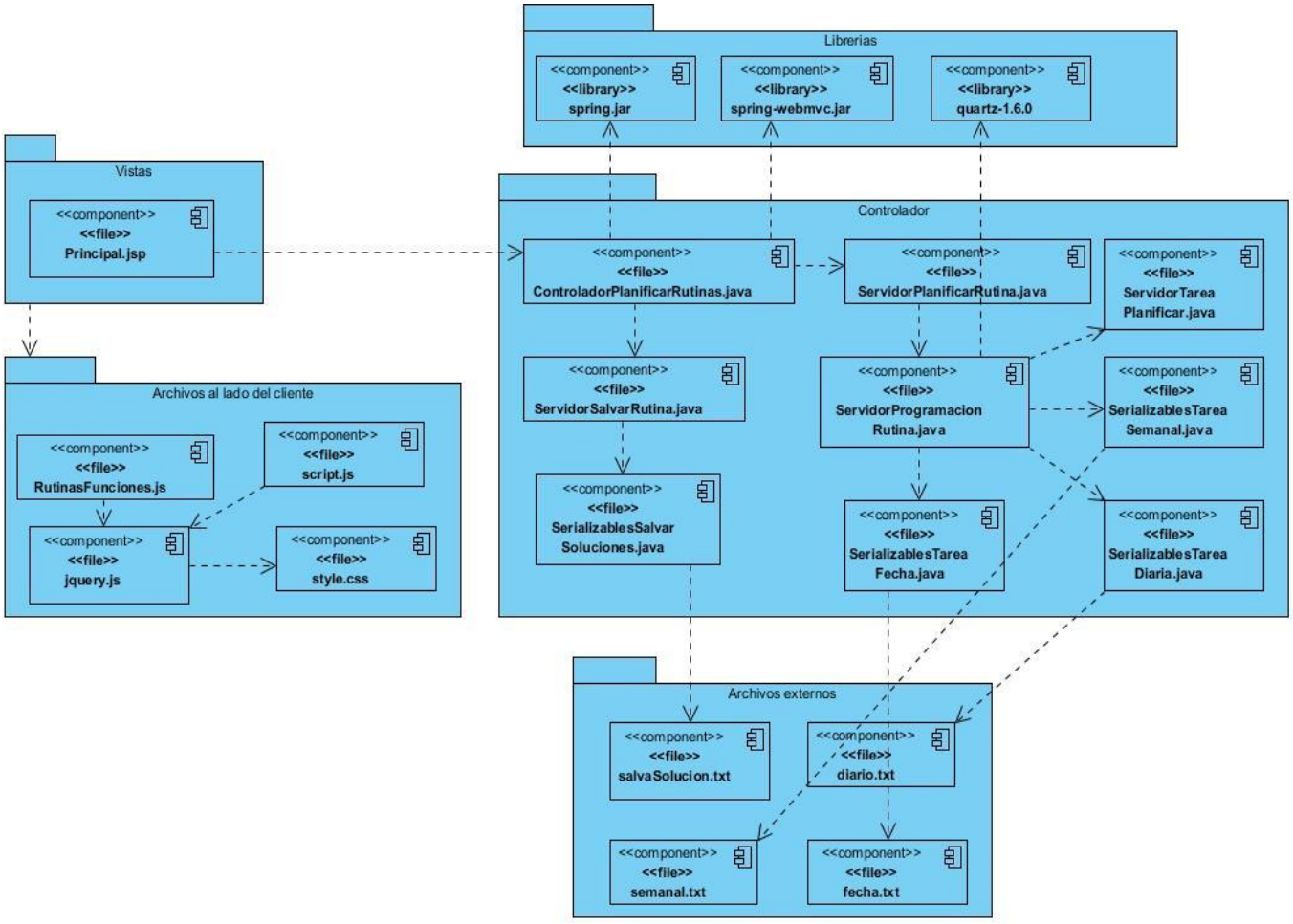


Figura 23. Diagrama de Componentes. CU Planificar Rutina.

Casos de Prueba

Tabla 6. Variables para el caso de prueba Ejecutar, Detener, Pausar, Reiniciar y Eliminar Rutina.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	seleccionarSolucion	boolean	No	Selecciona el nombre de la solución a ejecutar.

Tabla 7. Caso de prueba 2: Ejecutar Rutina

Escenario	Descripción	seleccionarSolucion	Respuesta del sistema	Flujo central
EC 2.1 Ejecutar Rutinas correctamente.	En este escenario el administrador ejecuta correctamente la solución.	V	El sistema muestra un mensaje ("La solución fue ejecutada satisfactoriamente").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la primera opción: Ejecutar solución.
EC 2.2 Ejecutar Rutinas incorrectamente.	En este escenario el administrador ejecuta incorrectamente la solución.	I	El sistema muestra un mensaje de error("Debe configurar la solución y salvar la rutina principal")	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la primera opción: Ejecutar solución y la solución no se encuentra configurada.
			El sistema muestra un mensaje de error("Debe seleccionar la solución")	El administrador se autentica, no selecciona la solución en la parte izquierda y selecciona en la parte superior central la primera opción: Ejecutar solución.

Tabla 8. Caso de prueba 3: Pausar Rutina

Escenario	Descripción	seleccionar Solucion	Respuesta del sistema	Flujo central
EC 3,1 Pausar Rutinas correctamente.	En este escenario el administrador pausa correctamente la solución.	V	El sistema muestra un mensaje ("La solución ha sido pausada").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción: Pausar solución.
EC 3,2 Pausar Rutinas incorrectamente.	En este escenario el administrador pausa incorrectamente la solución.	I	El sistema muestra un mensaje de error ("Debe configurar la solución y salvar la rutina principal").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción Pausar solución. La solución no se encuentra configurada,
			El sistema muestra un mensaje de error ("la solución no ha sido ejecutada").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción Pausar solución. La solución no ha sido ejecutada,
			El sistema muestra un mensaje de error("Debe seleccionar la solución")	El administrador se autentica, no selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción Pausar solución.

Tabla 9. Caso de prueba 4: Detener Rutina

Escenario	Descripción	seleccionar Solucion	Respuesta del sistema	Flujo central
EC 4.1 Detener Solución correctamente.	En este escenario el administrador detiene correctamente la solución.	V	El sistema muestra un mensaje ("La solución ha sido detenida").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción: Detener solución.
EC 4.2 Detener Solución incorrectamente	En este escenario el administrador detiene incorrectamente la solución.	I	El sistema muestra un mensaje de error ("Debe configurar la solución y salvar la rutina principal").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción: Detener solución. La solución no se encuentra configurada,
			El sistema muestra un mensaje de error ("La solución no ha sido ejecutada").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción: Detener solución. La solución no ha sido ejecutada,
			El sistema muestra un mensaje de error("Debe seleccionar la solución")	El administrador se autentica, no selecciona la solución en la parte izquierda y selecciona en la parte superior central la segunda opción: Detener solución.

Tabla 10. Caso de prueba 5: Reiniciar Rutina

Escenario	Descripción	seleccionar Solucion	Respuesta del sistema	Flujo central
EC 5,1 Reiniciar Rutinas correctamente.	En este escenario el administrador reinicia correctamente la solución.	V	El sistema muestra un mensaje ("La solución ha sido reiniciada").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la cuarta opción: Reiniciar solución.
EC 5.2 Reiniciar Solución incorrectamente	En este escenario el administrador reinicia incorrectamente la solución.	I	El sistema muestra un mensaje de error ("Debe configurar la solución y salvar la rutina principal").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la cuarta opción Reiniciar solución. La solución no se encuentra configurada,
			El sistema muestra un mensaje de error ("La solución no ha sido ejecutada").	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la cuarta opción Reiniciar solución. La solución no ha sido ejecutada,
			El sistema muestra un mensaje de error("Debe seleccionar la solución")	El administrador se autentica, no selecciona la solución en la parte izquierda y selecciona en la parte superior central la cuarta opción Reiniciar solución.

Tabla 11. Caso de prueba 6: Eliminar Rutina

Escenario	Descripción	seleccionar Solucion	Respuesta del sistema	Flujo central
EC 6.1 Eliminar Solución correctamente.	En este escenario el administrador elimina correctamente la solución.	V	El sistema muestra un mensaje de confirmación ("Estás seguro que desea eliminar el elemento seleccionado"). El sistema muestra un mensaje("La solución fue eliminada con éxitos")	El administrador se autentica, selecciona la solución en la parte izquierda y selecciona en la parte superior central la quinta opción: Eliminar solución.
EC 6.2 Eliminar Solución incorrectamente	En este escenario el administrador elimina incorrectamente la solución.	I	El sistema muestra un mensaje de error("Debe seleccionar la solución")	El administrador se autentica, no selecciona la solución en la parte izquierda y selecciona en la parte superior central la quinta opción: Eliminar solución.