Universidad de las Ciencias Informáticas

Facultad 2

Componente de carga de datos para el Sistema de Información Hospitalaria del Centro de Informática Médica.

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

Autores: Francisco Ruiz Duanys

José Pedro Groero López

Tutora: Ing. Nadiezka Milan Cristo

Cotutores: Ing. Arianna Carbonell Tamayo

Ing. Alfredo José Chiong Zaldivar

La Habana, junio de 2015

"Año 57 de la Revolución"

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.
Para que así conste firmamos la presente a los __ días del mes de junio del año 2015.

Autores:

Francisco Ruiz Duanys

José Pedro Groero López

Tutora:

Ing. Nadiezka Milan Cristo

Cotutores:

Ing. Arianna Carbonell Tamayo

Ing. Alfredo José Chiong Zaldivar

Datos de contacto

Ing. Nadiezka Milan Cristo

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2007. Miembro del Departamento de Desarrollo de Aplicaciones del Centro de Informática Médica (CESIM). Actualmente labora en el Proyecto Personalización del Sistema XAVIA HIS desempeñando el rol de Jefe del departamento. Ha impartido las asignaturas Inteligencia Artificial, Práctica Profesional y Gestión de Software.

Correo Electrónico: nmilan@uci.cu

Ing. Arianna Carbonell Tamayo

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2014. Ha participado en varios proyectos de desarrollo vinculados al perfil de salud. Miembro del Departamento de Desarrollo de Aplicaciones del Centro de Informática Médica (CESIM). Actualmente labora en el Proyecto Personalización del Sistema XAVIA HIS desempeñando el rol de Analista.

Correo Electrónico: actamayo@uci.cu

Ing. Alfredo José Chiong Zaldivar

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2014. Miembro del Departamento de Desarrollo de Componentes del Centro de Informática Médica (CESIM). Actualmente labora en el Proyecto Personalización del Sistema XAVIA HIS desempeñando el rol de Arquitecto de *Software*.

Correo electrónico: chiong@uci.cu

"Para ser exitoso no tienes que hacer cosas extraordinarias. Haz cosas ordinarias extraordinariamente bien".

Jim Rohn

DE FRANCISCO:

A mis abuelos por el cariño tan grande y la comprensión que me han brindado desde muy pequeño.

A mi papá por ser además de mi padre ha sido mi profesor y mi amigo, por todo lo que he aprendido de él y por ser mi ejemplo a seguir.

A mi mamá y mi padrastro por todo el sacrificio que han hecho para que yo pueda cumplir mis sueños y por todo el cariño que me han dado a pesar de la distancia en estos últimos tiempos.

A todos mis tíos por estar siempre conmigo aunque estuvieran lejos y por el apoyo tan grande que he recibido.

A mis primos por quererme tanto y ser como hermanos para mí.

A Yordanka y Gisell por haber sido tan importantes en mi vida durante todo este tiempo maravilloso que he pasado en la universidad.

A todos mis amigos por el aporte que han hecho a esta persona que hoy se hace ingeniero gracias a muchos de ustedes.

A mis tutores un agradecimiento especial por el sacrificio, la comprensión y la atención que han tenido con nosotros, por lo profesionales que han demostrado ser y porque gracias a ustedes hoy estoy aquí.

DE JOSÉ PEDRO:

A mi viejuca, por estar ahí para mí siempre cuando la he necesitado.

A mi hermana Titi, por ser un ejemplo a seguir para mí.

A mi hermano Toni, por ser como mi segundo padre.

A mi hermano Roynie, por darme siempre su apoyo incondicional.

A mi primo Yubi, por apoyarme y cuidarme a la vieja cuando yo estaba lejos.

A los profesores que me han ayudado a ser mejor persona y mejor profesional, especialmente a las profesoras Niurys y Rosa que han sido como una madre para mí aquí en la escuela.

A todos mis compañeros de aula desde 1er año, especialmente a Nesty, Cesar y Jordan y a todos mis compañeros actuales.

A mis hermanos de escuela: Ivey, El Indio, El Riki, Mario, Alex y El Franko, que tanto los he jodido a lo largo de estos 5 años.

Y a alguien que lleva ya un tiempito aguantando mis malcriadeces, como la de supuestamente exponer el día de hoy sin traje, a mi Anyta.

DE FRANCISCO:

A mi abuelo Luis donde quiera que esté, que sepa que su nieto no le falló, a mi abuela Maito, mi mamá, mi papá y a mis tíos por ser las personas que más me motivaron a ser quien soy.

DE JOSÉ PEDRO:

A mi viejo, que este donde quiera que esté, quiero que sepa que seguí sus consejos y que nunca lo defraudaré. Ya su Macomeco es Ingeniero!!!

Resumen

El proceso de carga de datos de forma rápida y eficiente, en sistemas de gestión de la información, siempre ha constituido una necesidad puesto que los mismos manejan gran volumen de información. Actualmente estos sistemas hacen uso de mecanismos de extracción, transformación y carga para lograr una mejor integración de los datos que utilizan. El Sistema de Información Hospitalaria desarrollado en el Centro de Informática Médica maneja un gran cúmulo de información para su configuración una vez instalado. El proceso de carga de datos se realiza de forma manual por lo que se torna engorroso y puede dar lugar a la ocurrencia de errores. Existen varios mecanismos para la carga de datos que resuelven este tipo de problemas en sistemas específicos, pero el sistema no soporta su incorporación.

El objetivo de la presente investigación es desarrollar un componente de carga de datos que permita una mejor configuración del Sistema de Información Hospitalaria en las instituciones hospitalarias. La metodología de desarrollo de software utilizada es el Proceso Unificado de Desarrollo. Como herramienta de modelado se emplea el *Visual Paradigm* y *JBoss Developer Studio* como herramienta de desarrollo. Como servidor de aplicaciones *JBoss Server*, *PostgreSQL* como sistema gestor de base de datos y *Pentaho Data Integration* como herramienta ETL para el diseño de las transformaciones necesarias para la integración de datos. El componente propuesto permite la correcta carga de datos en el Sistema de Información Hospitalaria y minimiza el tiempo de despliegue del mismo.

Palabras clave: carga, datos, extracción, transformación.

Índice de contenidos

Introducción 1

Capítulo 1. Tendencias del proceso de extracción, transformación y carga de datos. 5

1.1 Conceptos fundamentales 5

1.2 Proceso de extracción, transformación y carga. 6

1.3 Componentes de carga de datos en sistemas de gestión de información existentes vinculados con el campo de acción. 7

1.4 Principales herramientas a tener en cuenta para la implementación de un proceso ETL. 10

1.5 Tecnologías a utilizar para el desarrollo del componente de carga de datos. 13

Capítulo 2. Componente de carga de datos. 20

2.1 Esquema general de carga de datos 20

2.2 Elementos esenciales del componente de carga de datos 21

2.3 Propuesta del componente de carga de datos. 21

2.4 Especificación de los requisitos de software 31

2.5 Modelo de casos de uso del sistema 33

Capítulo 3. Diseño del componente. 41

3.1 Descripción de la arquitectura. Fundamentación 41

3.2 Modelo de diseño 41

Capítulo 4. Aplicación y validación. 48

4.1 Diagrama de componentes 48

4.2 Aplicación del componente de carga de datos en el Sistema de Información Hospitalaria 48

4.3 Tratamiento de errores 54

4.4 Validación de tiempo 55

Conclusiones 56

Recomendaciones 57

Referencias bibliográficas 58

Bibliografía.....63

Anexos.....64

Glosario de términos.....67

Índices de tablas

Tabla 1. Comparación de los componentes de carga de datos en los sistemas de gestión de información estudiados 10

Tabla 2. Comparación de las herramientas ETL estudiadas 13

Tabla 3. Descripción del actor del sistema.....34

Tabla 4. Descripción del caso de uso: Adicionar transformación.....34

Tabla 5. Descripción del caso de uso: Buscar transformación 37

Tabla 7. Promedio de tiempos obtenidos tras 2 iteraciones del proceso de carga de datos.....55

Índice de figuras

Figura 1. Esquema general de carga de datos....20
Figura 2. Propuesta del componente de carga de datos....22
Figura 3. Estructuras generales en PDI....23
Figura 4. Datos de los servicios en el documento *Excel*....24
Figura 5. Entrada de datos a la transformación....24
Figura 6. Validación del nombre de la entidad....25
Figura 7. Unión de los flujos de datos por una clave....26
Figura 8. Validación y conversión de valores....27
Figura 9. Búsqueda de datos en la base de datos....28
Figura 10. Pasos de persistencia de datos....28
Figura 11. Libro *Excel* correspondiente al módulo Configuración....30
Figura 12. Diagrama de casos de uso....34
Figura 13. Diagrama paquetes....43
Figura 14. Diagrama clases del diseño....44
Figura 15. Clase Transformación....44
Figura 16. Clase TransformacionManager....45
Figura 17. Clase AdicionarTransformacion....45
Figura 18. Clase BuscarTransformacion....46
Figura 19. Clase ModificarTransformacion....46
Figura 20. Clase EjecutarTransformacion....47
Figura 21. Diagrama de componente....48
Figura 22. Modificación del módulo configuración....49
Figura 23. Componente de carga de datos....50
Figura 24. Método addListaT() de la funcionalidad Adicionar transformación....50
Figura 25. Método adicionarLista() de la funcionalidad Adicionar transformación....51

Figura 26. Método modificarTransf() de la clase ModificarTransformacion.....51

Figura 27. Métodos setForigen() y setCon() de la clase Transformacion 52

Figura 28. Método ejecutar*()* de la clase Transformacion.....53

Figura 29. Flujo básico para realizar la carga de datos al sistema 53

Figura 30. Validación de la transformación mod_conf_datos_entidad.ktr.....54

Figura 31. Bloque *try/catch* del método adicioarTransf() 54

Introducción

El desarrollo científico y tecnológico por el cual ha transitado el mundo desde el pasado siglo, ha tenido una trascendencia sustancial para la actividad científica y una influencia extraordinaria en la sociedad. El acelerado crecimiento de los volúmenes de datos que se obtienen de los procesos realizados por el hombre, dificulta el procesamiento, almacenamiento y centralización de la información resultante de su actividad. Para dar solución a estos problemas surgen los sistemas de gestión de la información (1) los cuales constituyen una poderosa herramienta para agilizar y optimizar numerosos procesos que responden a necesidades específicas de la sociedad.

Hoy en día los sistemas gestores de información manejan grandes cúmulos de datos los cuales suelen almacenarse en diversas fuentes y ser muy extensos, por lo que el procesamiento de esta información implica un alto costo en tiempo y recursos para estos sistemas. Además se dificulta el acceso a los datos para generar informes, realizar análisis, tomar decisiones o cualquier otra actividad de interés organizacional. Estos elementos han hecho necesario cargar la información dispersa al sistema que la gestiona para lograr una mayor disponibilidad y facilitar la gestión de la misma.

La carga de datos es el proceso de transferencia de información desde las fuentes de origen hasta el sistema destino. Esta es una de las tareas más críticas que se presentan durante la instauración de un sistema de gestión de la información, ya que algunos de estos no cumplen su objetivo tras la ausencia de datos. Por lo que dicha carga suele ser la primera actividad que se realiza luego de la instalación de un sistema, aunque se puede realizar en cualquier otro momento que se requiera. De la calidad de este proceso depende el funcionamiento y la eficiencia de los servicios que brinda el sistema, por lo que su correcta ejecución constituye una prioridad, lo que sustenta la aplicación de un mecanismo adecuado para lograrlo.

Este mecanismo es el encargado de incorporar todos los datos requeridos antes y durante la utilización del sistema, realizándole las modificaciones necesarias para obtener un correcto formato de los mismos. Una de las estrategias usadas para dar solución a este problema es el proceso Extracción, Transformación y Carga (ETL, por sus siglas en inglés) (2). El proceso ETL permite extraer grandes cantidades de datos de distintas fuentes, transformarlos y ubicarlos en el destino deseado para su posterior utilización. Este proceso está diseñado para el movimiento e integración de información entre varias aplicaciones, lo que posibilita obtener una visión única y global que permite a los analistas y directivos, tomar las decisiones estratégicas adecuadas.

En la actualidad los grandes sistemas gestores de información necesitan usar herramientas ETL para la integración de los datos que utilizan. Entre estos se encuentran los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés) (3), los cuales permiten gestionar información sensible asociada a los procesos administrativos y de atención médica, lo que implica un aumento en la calidad de atención a los pacientes. Debido a los tipos de servicios que se brindan y a la sensibilidad de la información que se maneja, la cual no puede ser eliminada, estos sistemas han tenido un progresivo aumento del volumen de datos que generan y utilizan a diario.

El Centro de Informática Médica (CESIM), perteneciente a la Universidad de las Ciencias Informáticas, desarrolla un Sistema de Información Hospitalaria compuesto por 17 módulos, los cuales se vinculan a cada una de las diferentes áreas de una institución hospitalaria. En el momento de desplegar este sistema, para su correcta puesta en marcha se hace necesario registrar grandes cúmulos de datos de acuerdo a las características de la entidad donde se va a implantar.

Ejemplo de esto lo constituye la información de todo el personal médico y no médico que interactúa con el sistema, así como de los distintos servicios brindados en la entidad y sus especialidades. Además de otros datos que son necesarios para la configuración general, estos varían en cantidad según el tamaño de la entidad y son almacenados en 19 tablas de la base de datos del sistema. También es necesario cargar la información referente a los productos y medicamentos existentes, la cual es distribuida en 20 tablas y pueden superar las 2500 tuplas. Por otra parte se cargan los datos referentes a cada uno de los módulos que sean instalados, tal es el caso de Laboratorio el cual utiliza información almacenada en más de 30 tablas, además de Bloque Quirúrgico, Banco de Sangre, entre otros.

Actualmente el proceso de carga de los datos se realiza a partir de un conjunto de documentos en formato *Excel* que son presentados al cliente, el cual se encarga de introducir en ellos los valores requeridos. Una vez instalado el sistema, un conjunto de especialistas del proyecto deben registrar la información recopilada, y realizar las modificaciones necesarias para lograr la compatibilidad entre los formatos origen y destino. Todo este procedimiento se realiza de forma manual, dando lugar a la ocurrencia de errores, tales como la perdida, alteración y duplicación de la información. Además no se cuenta con un mecanismo de control e información de errores que garantice el éxito del proceso, lo que puede provocar fallas futuras en el sistema, así como reiteradas e innecesarias ejecuciones de todo el procedimiento.

Realizar el proceso de esta forma una vez el sistema se haya instalado se torna lento y engorroso cuando la información a cargar es abundante. En este caso surge la necesidad de utilizar más personal y extender

el tiempo para culminar la configuración del sistema, poniendo en riesgo el cumplimiento del término de tiempo establecido para concluir el despliegue.

A partir de la situación descrita anteriormente se identifica el siguiente **problema científico:** el proceso de carga de datos en el Sistema de Información Hospitalaria del Centro de Informática Médica dificulta la configuración del mismo en las instituciones hospitalarias.

En correspondencia con el problema, el **objeto de estudio** lo constituye: el proceso de carga de datos en sistemas de gestión de información. El **campo de acción** se centra en: los mecanismos de carga de datos en el Sistema de Información Hospitalaria del Centro de Informática Médica.

En vista de proporcionar una solución al problema descrito se define como **objetivo general:** desarrollar un componente de carga de datos que permita una mejor configuración del Sistema de Información Hospitalaria del Centro de Informática Médica en las instituciones hospitalarias.

Para el cumplimiento del objetivo planteado, se determinan las siguientes **tareas de la investigación:**

1. Analizar tendencias para la implementación del proceso de extracción, transformación y carga de datos en sistemas de gestión de la información.
2. Asimilar la arquitectura y pautas definidas para el desarrollo de los artefactos, el diseño y las aplicaciones del Sistema de Información Hospitalaria del Centro de Informática Médica.
3. Diseñar la estructura de los ficheros *Excel* para realizar el proceso de carga de datos en el Sistema de Información Hospitalaria del Centro de Informática Médica.
4. Diseñar las transformaciones necesarias para llevar a cabo la carga de los datos al Sistema de Información Hospitalaria haciendo uso de la herramienta ETL seleccionada.
5. Obtener los artefactos correspondientes a los flujos de trabajo: Modelado de negocio, Gestión de requerimientos, Diseño e Implementación relacionados con los procesos de extracción, transformación y carga de datos, mediante el Proceso Unificado de Desarrollo.
6. Desarrollar un componente de carga de datos para el Sistema de Información Hospitalaria del Centro de Informática Médica.

Con el desarrollo de la solución propuesta se espera obtener los siguientes **beneficios:**

1. Reducir el esfuerzo requerido para introducir en el Sistema de Información Hospitalaria del CESIM los datos necesarios para su correcto funcionamiento una vez instalado.
2. Disminuir el tiempo necesario para la configuración del sistema.
3. Ejecutar en un entorno controlado, amigable y rápido las transformaciones necesarias para lograr compatibilidad de formatos entre el sistema y los datos que se cargan.

Los **métodos científicos** utilizados en la presente investigación son:

Métodos teóricos:

- **Analítico - sintético:** para el análisis de conceptos y definiciones descritas en las literaturas existentes relacionadas con el objetivo de la investigación.
- **Histórico - lógico:** para el estudio del proceso de carga de datos de sistemas existentes con el objetivo de conocer su funcionamiento.

Métodos empíricos:

- **Entrevistas no estructuradas:** para obtener información relacionada con el proceso de carga de datos durante el despliegue del HIS, así como la distribución de los datos requeridos para este proceso.

El documento se encuentra estructurado en 4 capítulos, de la siguiente manera:

Capítulo 1. Tendencias del proceso de extracción, transformación y carga de datos: describe los diferentes conceptos vinculados al objeto de estudio de la investigación. Se describen las características fundamentales de algunos sistemas de gestión de la información que hacen uso de mecanismos ETL para la integración de sus datos. Se fundamenta la selección de la tecnología a utilizar para el desarrollo de la propuesta.

Capítulo 2. Componente de carga de datos: describe cómo se realiza el proceso de carga de datos del Sistema de Información Hospitalaria del CESIM. Se describen los elementos esenciales de la propuesta de solución, así como los requisitos funcionales y no funcionales que se tuvieron en cuenta para el desarrollo de la misma. Se modelan los casos de uso del componente.

Capítulo 3. Diseño del componente: propone la arquitectura de desarrollo del componente de carga de datos. Se describen los patrones de diseño utilizados para la propuesta y se modelan los artefactos necesarios para el diseño.

Capítulo 4. Aplicación y validación: describe el proceso de aplicación del componente de carga de datos en el Sistema de Información Hospitalaria. Además se valida la disminución del tiempo de configuración del sistema al utilizar la propuesta.

Capítulo 1. Tendencias del proceso de extracción, transformación y carga de datos.

El presente capítulo describe las bases conceptuales para el proceso de carga de datos en sistemas de gestión de información. Se realiza un estudio de herramientas ETL con el objetivo de seleccionar la más factible para la solución del problema. Se analiza además el proceso de carga de datos en sistemas de gestión de la información. Por último se fundamenta la tecnología a tener en cuenta para el desarrollo de la propuesta.

1.1 Conceptos fundamentales

Para lograr un mejor entendimiento del objeto de estudio del presente trabajo, se detallan los conceptos fundamentales asociados al mismo, los cuáles serán explicados a continuación.

1.1.1 *Data Warehouse*

Un *Data Warehouse*, es un repositorio no volátil de información extraída de otros sistemas, ya sean sistemas transacciones, base de datos departamentales, la intranet o base de datos externas. Los datos en el *Data Warehouse* deben estar integrados, orientados a temas, consolidados, seguros, y limpios para que sea una fuente segura de soporte para la toma de decisiones y para las aplicaciones de la información. Dentro de sus principales características se encuentra que debe proveer la mayor cantidad de información a la mayor cantidad de personas dentro de la organización, monitorear el comportamiento de los clientes y mejorar la capacidad de respuesta a las cuestiones del negocio. (4)

1.1.2 *Data Mart*

Data Mart es concebido como un repositorio menos ambicioso que un *Data Warehouse*, teniendo un alcance de contenido limitado. Éste se usa para una parte más pequeña de determinada organización y/o para un problema particular de análisis dentro de la misma. La complejidad de un *Data Mart* es baja o mediana debido a que tiene menos usuarios, menos temas y provee una vista de una parte de las áreas funcionales de la organización. Por esta razón su construcción requiere mucho menos tiempo y costo que la construcción de un *Data Warehouse*. Por tanto, los *Data Marts* son considerados una serie de almacenes de datos que tienen una orientación y dominio específico y centralizado, nutriendo de esta forma a los *Data Warehouse*. (4)

1.1.3 Integración de datos

Los procesos de integración de datos, tienen como fin combinar los datos que residen en diversas fuentes para lograr obtener una vista unificada de los datos con los cuales se trabaja. Como principal situación a la

hora de realizar la integración, se tiene que los datos se encuentran distribuidos, dispersos, de forma heterogénea y en la mayoría de los casos no estandarizados. Por lo que la información se encuentra aislada e inconsistente, impidiendo la uniformidad de los datos en cuanto a términos, cantidades, unidades de medida, formatos, etc. En entidades generadoras de grandes cúmulos de datos, la tarea de integración se hace sumamente compleja y costosa debido a las inconsistencias existentes entre los datos. Es por ello que la integración de datos se hace necesaria para lograr la homogeneidad de los datos pertenecientes a múltiples fuentes de origen. (5)

1.2 Proceso de extracción, transformación y carga

Un proceso de Extracción, Transformación y Carga (ETL, por sus siglas en inglés) es un proceso que comprende la extracción de datos de fuentes externas, su transformación para adaptarse a las necesidades operativas existentes y la carga de estos en las fuentes de destino final. Los sistemas ETL se utilizan comúnmente para integrar datos de múltiples aplicaciones, normalmente desarrolladas por diferentes proveedores o alojado en *hardware* separado. La primera etapa de un proceso ETL implica la extracción de los datos de los sistemas de origen. Luego se realizan las transformaciones pertinentes según una serie de reglas o funciones establecidas, las cuales son aplicadas a los datos extraídos desde la fuente hasta obtener los datos que serán cargados en el destino final. (2)

1.2.1 Extracción

Extracción consiste en extraer los datos desde las fuentes de origen especificadas, estas pueden encontrarse sobre arquitecturas, estructuras o archivos planos heterogéneos. Los formatos de las fuentes normalmente se encuentran en bases de datos relacionales o ficheros planos, pero pueden incluir bases de datos no relacionales u otras estructuras diferentes. Una parte intrínseca del proceso de extracción es verificar los datos extraídos, de lo que resulta un chequeo que comprueba si los datos cumplen las pautas y la estructura deseada, de no ser así son rechazados. (2)

Un requisito importante que debe exigir la tarea de extracción es que ésta provoque un mínimo impacto en el sistema origen, pues si los datos a extraer son excesivos, el sistema de origen podría colapsar, provocando que no pueda utilizarse con normalidad para su uso cotidiano. Esta es la razón por la que en grandes sistemas las operaciones de extracción suelen programarse en horarios o días donde este impacto sea mínimo o nulo.

Las herramientas utilizadas en la extracción deben ser adaptables, extensibles y capaces de filtrar los datos relevantes a extraer de las fuentes, permitiendo la compresión, descompresión, y encriptación de

datos. Al concluir esta primera etapa del proceso los datos se encuentran listos para la siguiente etapa, la transformación.

1.2.2 Transformación

Transformación constituye la etapa central del proceso, donde los datos extraídos son convertidos de su estado original a un formato compatible con el destino, cumpliéndose las reglas de negocio y funciones establecidas. Para potenciar la eficacia de estas directrices, hay que asegurarse que sean declarativas, independientes, claras, inteligibles y con una finalidad útil para el negocio.

En esta etapa los datos son reformateados, limpiados y normalizados sin perder su exactitud o veracidad con respecto a las fuentes. También son tratados aquellos datos incompletos, que contienen errores o valores fuera de los límites y que pueden entrar en conflicto con el sistema de ser cargados sin modificarlos. Una vez limpiados los datos, se proceden a realizar las transformaciones, que pueden ir desde simples conversiones de formato, obtención de valores calculados, transposición de columnas, hasta las más complejas operaciones de integración. Al concluir esta etapa, se procede a ejecutar las acciones de la última etapa del proceso, la carga de los datos al destino deseado. (2)

1.2.3 Carga

Carga constituye la tercera y última etapa del proceso, donde los datos resultantes de la fase anterior son cargados al sistema de destino. Dependiendo de los requerimientos de la organización, este proceso puede abarcar una amplia variedad de acciones diferentes. En algunas bases de datos se sobrescribe la información antigua con nuevos datos, en los *Data Warehouse* se mantiene un historial de los registros, de manera que se pueda hacer una auditoría de los mismos y disponer de un rastro de toda la historia de un valor a lo largo del tiempo. Luego de tomar la decisión de realizar la carga de datos hacia el destino deseado, se aplican todas las restricciones y *triggers* (6) que se hayan definido, las cuales contribuyen a garantizar la calidad de los datos obtenidos una vez concluido el proceso ETL. (2)

1.3 Componentes de carga de datos en sistemas de gestión de información existentes vinculados con el campo de acción.

En la actualidad con el constante aumento de los volúmenes de datos que son generados en las instituciones, se hace necesario el uso de los mecanismos ETL en los sistemas gestores de la información. Con la utilización de estos mecanismos se satisfacen criterios como la sincronización, la conectividad y la actualización de los sistemas. Durante la investigación se realizó un estudio de sistemas que hacen uso de alguna herramienta o mecanismo ETL, destacando sus principales funcionalidades, ventajas y herramientas con las cuales fueron desarrollados.

1.3.1 Componente de carga de datos para el sistema SAP R/3

SAP R/3 es un sistema de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés de *Enterprise Resource Planning*) (7) desarrollado por la compañía alemana SAP (8). Mediante este sistema de gestión es posible realizar un control exhaustivo de todos los procesos y operaciones que se realizan dentro de una compañía, almacenando la información que puede ser utilizada por cualquier sector de la organización. Este fue implementado en ABAP (9) aunque en las últimas versiones ha tenido parte de su codificación en *Java*. Por otra parte opera utilizando el principio cliente/servidor aplicado a los tres niveles de la arquitectura de procesos con los que trabaja: bases de datos, servidor de aplicaciones y cliente. Además es compatible con los sistemas operativos *Windows, Macintosh, Red Hat de Linux, IBM OS/400, IBM S/390, OS/2, Citrix, HP-UX, AIX, MPE/iX* y *Open VMS*, así como con bases de datos *Microsoft SQL Server, Oracle, IBM DB/2, Adabas, Informix y Sybase ASE*. (10)

SAP R/3 está diseñado en módulos claramente diferenciados, que responden a las necesidades propias de cada una de las áreas de una empresa. Además cuenta con un componente dedicado a la carga inicial de datos referentes a los registros maestros de los activos fijos de la empresa. Este proceso se realiza a partir de la utilización de diferentes métodos como son: carga inicial automática mediante *batch input*, carga inicial automática mediante *input* directo, carga inicial automática mediante BAPI (11), transacción de carga inicial manual o carga inicial mediante documentos *Excel*. Este último es especialmente adecuado para transferir *sets* de datos pequeños, tales como unos pocos centenares de activos fijos. El mismo consiste en hacer una carga masiva de datos previamente insertados de forma manual en un documento *Excel*, para lo cual es necesario establecer un formato específico. (10)

1.3.2 Componente de carga de datos para el sistema SIGESIC

Sistema Integral de Gestión para las Industrias y el Comercio (SIGESIC) es un sistema web venezolano desarrollado en *Python* el cual establece conexión con su base de datos en *PostgreSQL* haciendo uso de la herramienta *Pentaho*. Este sistema permite el registro de la información de las unidades económicas públicas o privadas, entes gubernamentales, cámaras, organizaciones gremiales y personas naturales, con la finalidad de suministrar la información requerida por el Ministerio del Poder Popular para la Industria. Actualmente está activo solo el registro de las unidades económicas manufactureras y comercializadoras al por mayor. SIGESIC está compuesto por 15 módulos de los cuales en 6 de ellos se realiza carga masiva de datos. (12)

La carga masiva de datos se realiza mediante el uso de archivos con extensión (.csv), estos son descargados desde SIGESIC proporcionándosele un formato correcto en el momento de la descarga con el objetivo de garantizar el éxito del proceso. Los archivos deben ser llenados siguiendo las orientaciones

del manual de usuario establecido para realizar la carga masiva al sistema (12). Al concluir el llenado de los archivos, estos se cargan al sistema, el cual al finalizar el proceso envía un correo electrónico con información detallada de lo ocurrido durante el proceso de carga.

1.3.3 Componente de carga de datos para el Sistema de Registro de Contrato de Seguro Vida Ley

El Sistema de Registro de Contrato de Seguro Vida Ley (SRCSVL) es un sistema web desarrollado en Perú. El mismo permite registrar los datos de las pólizas, de los trabajadores y de los beneficiarios que forman parte de la póliza. Para agilizar y llevar a cabo la carga de datos de forma masiva este sistema cuenta con una funcionalidad denominada Carga Masiva. Esta funcionalidad realiza la carga masiva de datos al sistema haciendo uso de ficheros *Excel* con extensión (.xlsm) los cuales deben ser descargados con un formato obligatorio desde la página web del Ministerio de Trabajo y Promoción del Empleo (13). Luego de introducir los datos en los ficheros *Excel*, se deben ejecutar los macros existentes en estos en el orden especificado en la guía de usuario para la carga masiva, lo que genera los ficheros (.txt) que serán cargados posteriormente al sistema. (14)

1.3.4 Componente de carga de datos para el Sistema Integrado de Información Financiera (SIIF)

La solución SIIF Nación II es un desarrollo Web colombiano, diseñado con filosofía de arquitectura empresarial que utiliza como herramienta de desarrollo *.NET* y *SQL Server 2005* como motor de Base de Datos. Es una herramienta modular, transversal y transaccional, a través de la cual las entidades que hacen parte del Presupuesto General de la Nación realizan su gestión financiera pública, de manera estandarizada, segura, conforme a la norma y en línea y tiempo real, que permite generar información consistente, confiable y oportuna. (15)

El sistema cuenta con una funcionalidad denominada Cargas Masivas dedicada a la carga de la información presupuestal. El proceso se realiza a partir de documentos de texto plano y *Excel* que contienen la información a cargar en un formato específico. Además antes de ejecutar el proceso estos ficheros tienen que ser comprimidos y firmados digitalmente. (16)

1.3.5 Módulo de carga de datos *offline* del Sistema Integral de Análisis de Información

El Módulo de carga de datos *offline* del Sistema Integral de Análisis de Información (SIAI) es una solución desarrollada en la Universidad de las Ciencias Informáticas. El mismo está encaminado a garantizar que la información con caducidad mayor de un período de tiempo que se encuentra *offline*, en ficheros de texto, pueda ser cargada al sistema. Para esto posee un mecanismo de extracción, transformación y carga a la medida de sus necesidades, así como una plataforma para la administración de notificaciones de solicitud y realización de carga de datos.

El módulo permite que la información pueda ser utilizada en el proceso de investigación por parte de los analistas de fraude en ETECSA sobre el uso de los servicios de telefonía fija y móvil en Cuba. Además proporciona al usuario interfaces intuitivas para la configuración del manejo de datos, así como generar reportes. El desarrollo de este módulo fue guiado bajo el Proceso de Desarrollo Unificado, se usó *Python* 2.7 como lenguaje de programación y *Oracle* 11g como gestor de base de datos. (17)

Comparación de los componentes de carga de datos en los sistemas de gestión de información

A continuación se muestra en la Tabla 1 una comparación teniendo en cuenta las principales características de los componentes estudiados:

Tabla 1. Comparación de los componentes de carga de datos en los sistemas de gestión de información estudiados

Componente	Privativo	Tecnología	Conexión con *PostgreSQL*	Proceso ETL	Ficheros
SAP R/3	Si	ABAP, *Java*	No	Si	*Excel*
SIGESIC	Si	*Python*	Si	Si	CSV
SRCSVL	Si	*Java*	Si	No	*Excel*, Texto plano
SIIF Nación	Si	*Java, .Net*	No	Si	*Excel*, Texto plano
SIAI	No	*Python*	No	Si	Texto plano

Una vez analizadas las principales características de los componentes de carga de datos presentes en el estudio se obtuvieron las siguientes conclusiones:

- Solo SIAI no es privativo.
- La mayoría de los componentes están soportados por la tecnología *Java*.
- Solo SIGESIC y SRCSVL establecen conexión con *PostgreSQL*.
- Se ha generalizado el uso del proceso ETL para la implementación de los componentes.
- La mayoría de los componentes de carga de datos cuya fuente es un archivo *Excel* implementan un proceso ETL.

1.4 Principales herramientas a tener en cuenta para la implementación de un proceso ETL.

Hoy en día existe una gran variedad de herramientas ETL, las cuales son utilizadas de acorde a las necesidades de cada institución. Con su uso muchas de las tareas, cuya realización de forma manual se

tornaba engorrosa, pueden ahora ser ejecutadas más ágil y óptimamente. Un ejemplo de estas tareas lo conforma el movimiento de datos de una fuente a otra, la limpieza y el reformateo de los datos. A continuación se muestra una selección de las principales herramientas ETL.

1.4.1 *Jaspersoft ETL*

Jaspersoft ETL supera a muchos sistemas ETL de código abierto y propietario. Con su uso se pueden extraer datos de un sistema transaccional para crear un preciso almacén de datos o *Data Mart* para la presentación de informes y análisis. Incluye un modelador de negocios para acceder a una vista no técnica de la información, flujo de trabajo, mostrar y editar el proceso ETL con el diseñador de *Job*, una herramienta de edición gráfica, definir asignaciones complejas y transformaciones con *Tranformation Mapper*. Por otra parte se puede generar código portable *Perl* o *Java* que se puede ejecutar en cualquier máquina. También tiene la capacidad de seguimiento de las estadísticas de ETL de principio a fin con depuración en tiempo real, permite la salida simultánea desde una entrada a múltiples fuentes incluyendo archivos planos, archivos XML, bases de datos, servicios web, servidores FTP y POP con cientos de conectores disponibles y el uso de la consola de monitorización de actividades para supervisar los eventos de trabajo (éxitos, fracasos, advertencias, etc.), los tiempos de ejecución, y los volúmenes de datos. (18)

1.4.2 *Talend Open Studio*

Talend Open Studio es un conjunto versátil de productos de código abierto para el desarrollo, prueba, despliegue y administración de proyectos de integración de aplicaciones de gestión de datos. *Talend* ofrece una plataforma que facilita la gestión de datos y la integración de aplicaciones de forma más sencilla proporcionando un entorno unificado para la gestión del ciclo de vida de toda la empresa. En proyectos ETL, *Talend Open Studio* proporciona para la integración de datos un rico conjunto de características incluyendo un entorno de desarrollo integrado con una interfaz gráfica intuitiva basada en arrastrar y soltar el diseño del trabajo, y un sistema de repositorio unificado para almacenar y reutilizar metadatos. *Talend* posee más de 400 componentes integrados en los conectores que permiten salvar rápidamente entre bases de datos, sistemas de archivos, los servicios *web*, paquetes de aplicaciones empresariales, almacenes de datos, *software* como servicio y las aplicaciones basadas en la nube. (19)

1.4.3 *CloverETL*

CloverETL es una rápida solución de integración de datos de extremo a extremo. Es de fácil uso y cuenta con controles intuitivos, gran flexibilidad y velocidad de procesamiento. *CloverETL* es una herramienta de integración de datos basada en el *framework* de *Java* que se puede utilizar para transformar, mapear o manipular datos en varios formatos como CSV, FIXLEN, XML, XBASE, COBOL, *Lotus*. *CloverETL* se

puede utilizar independiente o incorporado (como una librería) y se conecta a RDBMS, JMS, SOAP, LDAP, S3, HTTP, FTP, ZIP y TAR. (20)

1.4.4 *Pentaho*

Pentaho es una gráfica e intuitiva herramienta liberada bajo licencia Apache, basada en estándares de arrastrar y soltar, tiene una herramienta de integración de datos de código abierto que ofrece varios niveles de funcionalidad ETL junto con herramientas para gestionar a nivel de metadatos. *Pentaho* ofrece una *suite* de productos de inteligencia de negocios, incluyendo una interfaz gráfica para diseñar, ejecutar y supervisar estos procesos (*Spoon*). *Pentaho Data Integration* (PDI) es uno de los componentes de esta *suite* que soporta la funcionalidad ETL.

PDI o *Kettle* como se le conoce en la comunidad de código abierto, puede ser embebido en una aplicación *Java*. Esto se puede lograr mediante el uso de un conjunto de librerías, las que permiten la creación de instancias de clases en la aplicación para crear el escenario ETL que se desee. Los escenarios incluyen: ejecución de transformaciones, ejecución de tareas, construcción de transformaciones, construcción de trabajos de forma dinámica y obtención de información de registro. Dentro de sus características se incluyen la migración de datos entre aplicaciones o bases de datos, exportar datos de bases de datos a archivos planos, carga masiva de datos a las bases de datos, limpieza de datos y la integración de aplicaciones. (21)

1.4.5 *Toolsverse Framework ETL*

Toolsverse Framework ETL es un motor ETL independiente escrito en *Java*. Incluye ejecutables para todas las plataformas y puede ser integrado en otras aplicaciones. Dentro de sus características claves se encuentra que es integrable, de código abierto y libre, utiliza funciones de las base de datos de destino para hacer transformaciones y cargas, posee mapeo manual y automático de datos y posibilita la transmisión de datos entre distintas fuentes. Además tiene una gran gama de *drivers*, permitiendo conectarse a todo el catálogo de metadatos de *Oracle*, *MS SQL Server, MySQL, Informix, DB2, Sybase ASE y PostgresSQL*. También es posible establecer conexión con archivos planos como *Excel ODBC, TXT, ODBC de Access, XML, JSON, Excel XLS y Excel XLSX*. El mismo es adaptable a las necesidades de cada institución, permitiendo dejar solamente los componentes indispensables para el trabajo a realizar. (22)

Comparación de las herramientas ETL estudiadas

A continuación se muestra en la Tabla 2 una comparación de las herramientas ETL presentes en el estudio, para lo que se tienen en cuenta los siguientes criterios:

- Tecnología: Lenguaje de programación o tecnología en la que se desarrolló la herramienta.
- Conexión con *PostgreSQL*: Permite o no conexión con el servidor *PostgreSQL*.
- Conexión con ficheros *Excel*: Permite o no extraer información de ficheros *Excel*.
- Motor independiente: Posee o no un motor independiente que pueda ser embebido en otras aplicaciones.
- Licencia: Licencia bajo la cual es distribuida la herramienta.

Tabla 2. Comparación de las herramientas ETL estudiadas

Herramienta	**Tecnología**	**Conexión con *PostgreSQL***	**Conexión con ficheros *Excel***	**Motor independiente**	**Licencia**
Jaspersoft ETL	*Java*	Si	Si	No	Comercial
Talend Open Studio	*Java*	Si	Si	No	Apache
CloverETL	*Java*	Si	Si	Si	Comercial
Pentaho Data Integration	*Java*	Si	Si	Si	Apache
Toolsverse Framework ETL	*Java*	Si	Si	Si	Comercial

Una vez realizado el análisis de las herramientas estudiadas según los criterios definidos se obtuvieron las siguientes conclusiones:

- Todas de las herramientas fueron desarrolladas con el uso de la tecnología *Java*.
- Generalmente establecen conexión con el servidor de base de datos *PostgreSQL* y permiten extraer datos desde ficheros *Excel*.
- *CloverETL, Pentaho Data Integration y Toolsverse Framework ETL* poseen un motor independiente que posibilita sea embebido en otras aplicaciones.
- Solo *Talend Open Studio* y *Pentaho Data Integration* están distribuidos bajo licencia Apache.

1.5 Tecnologías a utilizar para el desarrollo del componente de carga de datos.

A continuación se describe la tecnología a tener en cuenta para el desarrollo de la propuesta.

1.5.1 Metodología de desarrollo

Las metodologías de desarrollo de *software* son un conjunto de procedimientos y técnicas que ayudan a la documentación para el desarrollo de productos *software*. Con estas se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (23)

Proceso Unificado de Desarrollo

Proceso Unificado de Desarrollo (RUP por sus siglas en inglés) es un modelo que permite el desarrollo de *software* a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. RUP provee un enfoque para la asignación de tareas y responsabilidades dentro de un proceso de desarrollo. Su objetivo es asegurar la producción de *software* de alta calidad que satisfaga las necesidades de los usuarios finales, dentro de una planificación y presupuesto predecibles. El ciclo de vida de RUP se caracteriza por ser dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. (24)

1.5.2 Patrón de arquitectura

Los patrones arquitectónicos expresan un esquema organizativo estructural fundamental para sistemas de *software*, representando los diferentes elementos que los componen y las relaciones entre ellos. Estos proveen un conjunto de reglas y pautas que constituyen una guía para garantizar la estructura adecuada para soluciones a problemas de desarrollos de *software*, basado en soluciones comunes ya probadas y documentadas. (24)

Modelo-Vista-Controlador (MVC)

MVC es un patrón de arquitectura de *software* que separa los datos de una aplicación, la lógica y la interfaz de usuario en tres componentes distintos. Este patrón es frecuentemente visto en aplicaciones *Web*, donde el modelo lo constituyen los datos con los que trabaja la aplicación y la lógica de negocio, la vista es la página HTML (25) donde se encuentra el código que provee los datos dinámicos a la página. Mientras que el controlador es el responsable de recibir los eventos de entrada desde la vista. (26)

- Modelo: Es el componente encargado del acceso a datos, representando las estructuras de datos del sistema. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- Controlador: Responde a eventos y actúa como intermediario entre el modelo, la vista y cualquier otro recurso necesario para generar una página.

1.5.3 Arquitectura de sistema

La arquitectura de sistema es el diseño o conjunto de relaciones entre las diferentes partes que constituyen un sistema. Esta también es concebida como una representación de un sistema existente o de uno que se vaya a crear, y puede ser usada para transportar información abstracta sobre estos, sobre las relaciones entre sus elementos y sobre las reglas que rigen dichas relacione. (27)

Arquitectura Cliente Servidor

La Arquitectura Cliente Servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible, permitiendo simplificar las actualizaciones y mantenimiento del sistema. En este modelo, el papel del cliente es iniciar el diálogo, enviando peticiones al servidor conforme a algún protocolo asimétrico, pidiéndole que actúe, que le informe, o ambas cosas. El papel del servidor es entonces responder las solicitudes del cliente. (28)

Entre sus principales características se encuentran:

- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. Siendo los mensajes el mecanismo para la petición y entrega de solicitudes de servicios.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco.

1.5.4 Plataforma de desarrollo

***Java Enterprise Edition 6* (JEE 6)**

Java Enterprise Edition es el estándar de la industria para la computación empresarial de *Java*, el cual brinda soporte para desarrollar *software* para las empresas. Su gran éxito como plataforma para el desarrollo de aplicaciones se encuentra en esta especificación, que no es más que un conjunto de librerías que establecen un estándar para lograr un producto altamente calificado. Ofrece características tales como API de servicios de tecnología *JavaServer Faces* (JSF) (29) y *web*. (30)

Java EE 6 hace que la codificación sea más simple, pero al mismo tiempo mantiene el poder que ha establecido *Java EE* como la primera plataforma para servicios *web* y desarrollo de aplicaciones empresariales. Con el lanzamiento de esta edición, la especificación dio un gran paso hacia la excelencia, pues incorporó nuevas tecnologías de punta que no estaban contempladas anteriormente y que indiscutiblemente le incorporan a la plataforma gran simplicidad a la hora de trabajar, suficientes razones para no dudar en establecerla como base para el desarrollo.

1.5.5 Lenguaje de programación

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Cada lenguaje posee sus propias sintaxis y con su uso se pueden crear un conjunto de órdenes o instrucciones que resuelvan un determinado problema. (31)

Java

Java es un lenguaje de programación desarrollado por *Sun Microsystems* a principios de los años 90. Este se caracteriza por ser simple, orientado a objeto, distribuido, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, de multihilos y dinámico. *Java* incorpora parte de la sintaxis de C, C++, *Smalltalk*, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros. Tiene una arquitectura neutral, portátil y robusta. Posee las estructuras mínimas de un lenguaje de programación tradicional, sin añadir ninguna estructura más. Una de las características más importantes es que los programas ejecutables, creados por el compilador de *Java*, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. Proporciona numerosas comprobaciones en compilación y tiempo de ejecución y elimina la necesidad de liberación explicita de memoria, convirtiéndolo en un lenguaje robusto. (32)

1.5.6 Servidor de Base de Datos

Un servidor de base de datos es un programa que provee servicios de base de datos a otros programas o computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio. Estos son utilizados para almacenar, recuperar y administrar los datos de una base de datos, además de garantizar la seguridad e integridad de los mismos. (33)

PostgreSQL 8.4

PostgreSQL es en la actualidad uno de los sistemas gestores de bases de datos de código abierto más potente del mercado. *PostgreSQL* utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Dentro de sus características se encuentra que es una bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible y libre de costo. Por esta razón se puede caracterizar como extensible, ya que permite ser personalizado en caso de ser necesario. Otra de sus principales características lo constituye que es multiplataforma debido a su capacidad de poder ser ejecutado en *Linux* y *UNIX* en todas sus variantes *(AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64)* y en *Windows* 32/64bit. (34)

1.5.7 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como *C++, PHP, Python, Java, C#, Delphi, Visual Basic*, etc. (35)

JBoss Developer Studio 8.0.0

Red Hat JBoss Developer Studio es un conjunto de herramientas de desarrollo basadas en *Eclipse*. Contiene *plug-ins* que se integran con Eclipse para extender la funcionalidad existente del IDE. *JBoss Developer Studio* está diseñado para aumentar su productividad en el desarrollo de aplicaciones. El mismo integra y certifica tanto componentes de tiempo de ejecución como herramientas mediante la combinación de Eclipse. *JBoss Developer Studio* viene empaquetado con todas las dependencias necesarias y *plug-ins* de terceros para una instalación simplificada. (36)

1.5.8 Herramienta de modelado

Las herramientas de modelado permiten al desarrollador crear un modelo del sistema que se va a construir y también la evaluación de la validez y consistencia de este modelo. Proporcionan un grado de confianza en la representación del análisis y ayudan a eliminar errores con anticipación. (37)

***Visual Paradigm 8.0* para UML**

Se utiliza *Visual Paradigm 8.0* para UML ya que es una herramienta *CASE* que soporta el modelado mediante UML. Además proporciona asistencia a los analistas, ingenieros de *software* y desarrolladores,

durante todos los pasos del ciclo de vida de desarrollo de un *software*. La misma se caracteriza por facilitar el modelado UML, al proporcionar herramientas específicas para ello, permitiendo la estandarización de la documentación. Se integra con otras aplicaciones como herramientas ofimáticas, lo que aumenta su productividad. Posee corrección sintáctica y coherencia entre diagramas, controlando así que el modelado con UML sea correcto y permitiendo visualizar un mismo elemento en varios diagramas. Cuenta con generación de código lo que reduce los tiempos de desarrollo y evita errores en la codificación del *software*. También posee generación de informes a partir de la información introducida en la herramienta, soporta aplicaciones *web* y es fácil de instalar y actualizar. (38)

1.5.9 Servidor de aplicaciones

Los servidores de aplicaciones *web* tienen la capacidad de almacenar y gestionar aplicaciones *web*, las que constituyen servicios que pueden ser accedidos por los usuarios. Dichos servidores son los encargados de recibir las peticiones referidas a las páginas o elementos de la *web* a través del protocolo establecido. Los mismos son capaces de acceder a recursos de otros servidores y de entender instrucciones de lenguajes avanzados y traducirlas al lenguaje del navegador con el fin de dar una respuesta al usuario de forma transparente. (39)

JBoss Server 4.2.2 GA

Para el desarrollo de este trabajo se utilizó como servidor de aplicaciones *web JBoss Application Server* ya que es uno de los servidores de aplicaciones de código abierto más ampliamente desarrollado del mercado. También se tuvo en cuenta que posee Licencia Pública General Reducida de GNU (LGPL, por sus siglas en inglés) (40), por lo que puede ser distribuido o redistribuido y usarse en la implementación de cualquier aplicación comercial sin la implicación de costo alguno. El mismo soporta todas las funcionalidades de *Java EE*, incluyendo servicios adicionales como *clustering, caching* y persistencia. *JBoss Application Server* es ideal para aplicaciones *Java* y aplicaciones basadas en la *web*. Además soporta *Enterprise Java Beans* (EJB), y esto hace que el desarrollo de las aplicaciones sea mucho más fácil. (41)

Conclusiones parciales

- La tecnología asimilada en la investigación presenta validez a nivel internacional para el desarrollo de componentes de carga de datos de sistemas de gestión de información.
- Los componentes de carga de datos estudiados no cubren las necesidades del problema identificado, por los motivos siguientes:

  - La tecnología que da soporte a los mismos no es compatible con el Sistema de Información Hospitalaria del CESIM.
  - Están concebidos para solucionar problemas específicos.
  - Al ser herramientas propietarias trae como consecuencia un gasto monetario en su uso y mantenimiento.
- El *Pentaho Data Integration* es la herramienta más adecuada para el proceso de carga de datos en sistemas de gestión de información.

Capítulo 2. Componente de carga de datos.

En este capítulo se presentan los elementos esenciales que conforman la propuesta de solución a la situación problemática planteada. Se muestra el modelo de dominio, donde se abordan los conceptos fundamentales relacionados con el sistema. Se definen las principales características que poseerá la aplicación a desarrollar. Se realiza la especificación de los requisitos funcionales y no funcionales, y se muestran los artefactos correspondientes a partir de los cuales se representan los casos de uso, con el fin de estructurar el diagrama de casos de uso del componente.

2.1 Esquema general de carga de datos

A continuación se describen los conceptos asociados al dominio del problema en cuestión y las relaciones entre ellos.

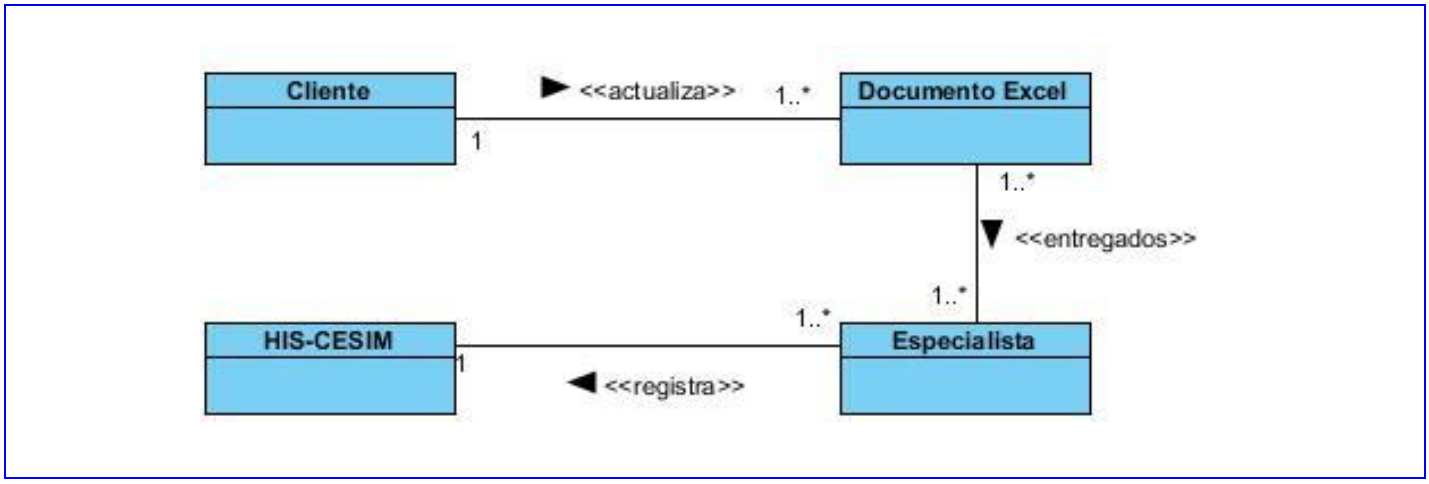


Figura 1. Esquema general de carga de datos

Fuente: Elaboración propia

El proceso de carga de datos del Sistema de Información Hospitalaria se realiza actualmente de la siguiente forma: un responsable por cada área de la institución donde se instalará el sistema, se encarga de registrar en documentos *Excel*, los datos necesarios para llevar a cabo la configuración del sistema. Luego estos documentos son entregados a los especialistas encargados de registrar manualmente los datos recopilados en el Sistema de Información Hospitalaria del CESIM.

A continuación se describen cada una de las clases mostradas en el esquema general de carga de datos:

Cliente: Es la persona encargada de registrar en documentos *Excel* los datos necesarios para llevar a cabo la configuración del sistema.

Documento *Excel*: Son los documentos con la estructura para recoger los datos necesarios para llevar a cabo la configuración del sistema.

Especialista: Es la persona encargada de introducir los datos proporcionados por el cliente hacia el sistema.

HIS-CESIM: Sistema de Información Hospitalaria del CESIM.

2.2 Elementos esenciales del componente de carga de datos

El componente de carga de datos propuesto se compone de cinco partes fundamentales:

- Las fuentes de origen de datos: las cuales están constituidas por los documentos *Excel* que contienen los datos a extraer, procesar y cargar al sistema.
- La herramienta *Pentaho Data Integration*: es la herramienta ETL seleccionada para diseñar el proceso por el cual transcurrirán los datos desde el origen hasta el destino.
- Las transformaciones: son los ficheros resultantes de los diseños realizados en la herramienta *Pentaho Data Integration*, los cuales contienen en su interior un conjunto de elementos que guían el proceso de extracción, transformación y carga de datos.
- El HIS: es el encargado de ejecutar las transformaciones diseñadas previamente en el *Pentaho Data Integration*.
- La base de datos: es el lugar donde son cargados los datos que necesita el sistema para su correcto funcionamiento.

Los datos extraídos de los documentos *Excel* son analizados y procesados durante todo el camino que recorren hasta ser cargados a la base de datos. En la Figura 1 se muestra la relación existente entre los elementos esenciales que conforman el componente de carga de datos.

2.3 Propuesta del componente de carga de datos.

Luego de concluir la investigación realizada y poner en práctica los conocimientos adquiridos, se obtiene un componente de carga de datos para el Sistema de Información Hospitalaria del CESIM.

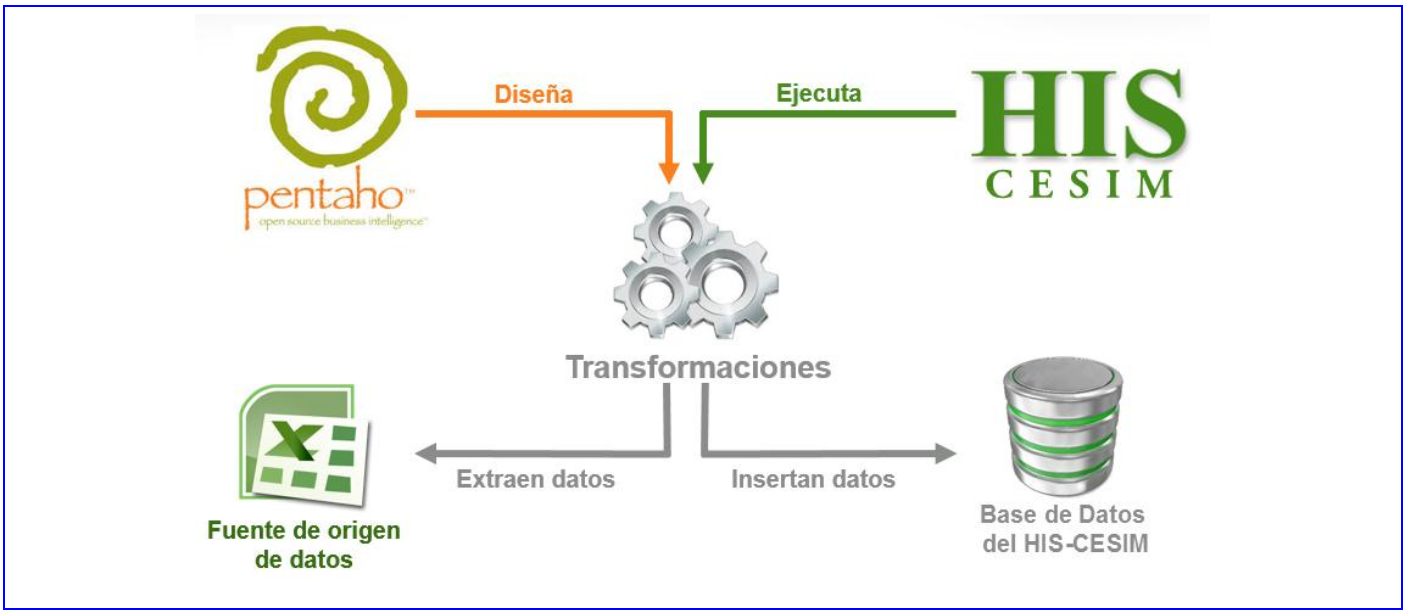


Figura 2. Propuesta del componente de carga de datos

Fuente: Elaboración propia

2.3.1 Pentaho Data Integration

La herramienta *Spoon* del *Pentaho Data Integration* contiene un conjunto de acciones llamadas **pasos,** cada uno diseñado para una función específica. Estos pasos son unidos por una representación gráfica del flujo llamada **saltos**, los cuales se encargan de trasportar datos de un paso a otro. (42)

Durante el desarrollo del componente de carga de datos se usan los pasos *Microsoft Excel Input* y *Get System Info* para la extracción de datos de los ficheros *Excel* y del sistema, como es el caso de la fecha actual. Los pasos *Database lookup* y *Database join* para realizar búsquedas sencillas y complejas en la base de datos del sistema. Para realizar transformaciones sobre los datos extraídos se usan los pasos *Split field to rows*, *Unique rows*, *Add constants*, *Sort rows* y *Select values* los que permiten dividir cadenas por un carácter, eliminar filas duplicadas, añadir nuevos valores, ordenar y eliminar o renombrar campos respectivamente.

Por otra parte se utilizan los pasos *Block this step until steps finish*, *Abort*, *Filter rows* y *Append streams* para manipular el flujo de datos, ya sea para detenerlo temporalmente en un paso determinado o para detener la ejecución si se cumple alguna condición, así como para separar y unir los flujos de datos. El paso *Merge Join* se usa para unir dos flujos extraídos de distintas fuentes, siempre que tengan un campo en común. Para notificar al usuario de los errores ocurridos durante la ejecución del proceso se utiliza el paso *Write to log*. El paso *Modified Java Script Value* se emplea para la validación, conversión y creación

de nuevos campos a través del lenguaje *JavaScript*. Finalmente los datos se persisten en la base de datos del sistema por el paso *Table output*.

2.3.2 Transformaciones

Una transformación permite realizar una variedad de tareas como movimiento de datos de un lugar a otro. Su rol principal es extraer datos de las fuentes de origen, transformarlos de una representación a otra y cargarlos en las fuentes de salida. La misma consiste en una secuencia de pasos relacionados por saltos como se muestra en la Figura 3. (42)

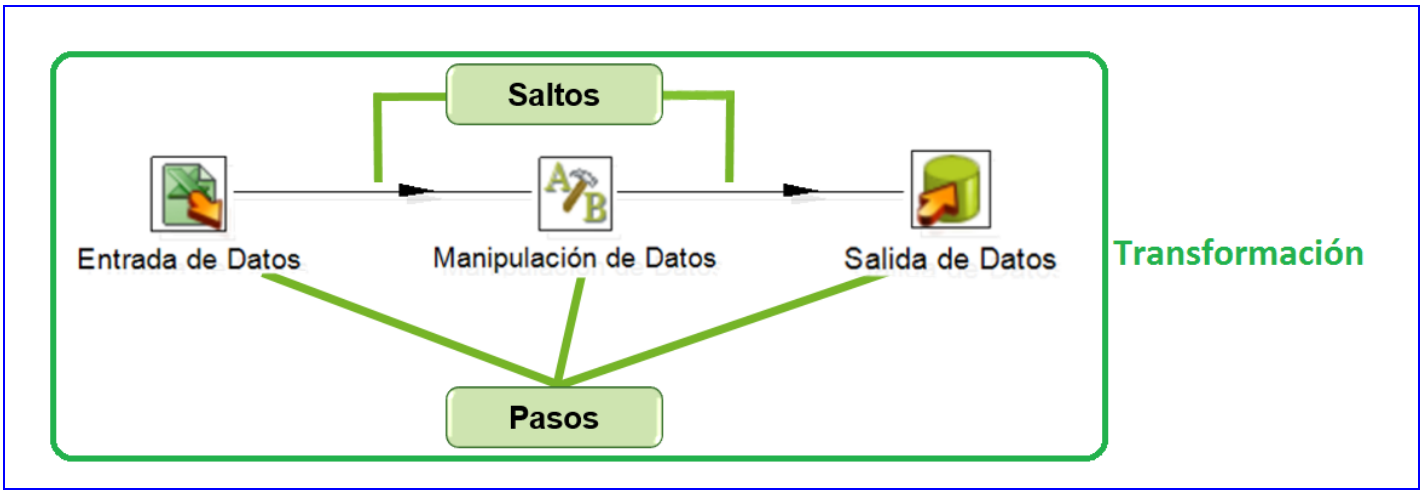


Figura 3. Estructuras generales en PDI

Fuente: Elaboración propia

A continuación se muestra de manera más detallada la elaboración de una de las transformaciones necesarias para el componente desarrollado, así como la descripción de cada uno de los pasos utilizados en el diseño de la misma.

La transformación mod_conf_servicio.ktr surge de la necesidad de cargar al sistema los datos correspondientes a los servicios con que cuenta una determinada entidad. De estos servicios se recoge el nombre del servicio, el departamento y servicio al que pertenece, si es físico o no y si tienen o no consulta externa, hospitalización y emergencias. Esta información se encuentra en la hoja Servicios del documento *Excel* HIS Módulo Configuración.xls como se observa en la Figura 4.

XAVIA HIS

SISTEMA HIS: MÓDULO CONFIGURACIÓN

Nombre de la entidad: Hospital Industrial San Tomé

Listado de servicios por departamento

No	Departamento	Nombre del servicio	Fisico	Consulta externa	Hospitalización	Emergencias	Servicio al que pertenece
1	Cirugía	Cirugía Maxilofacial	Si	Si	Si	Si	
2		Cirugía Cabeza Cuello	Si	Si	No	No	Cirugía Maxilofacial
3	Medicina	Cardiología	Si	No	Si	Si	
4		Neurología	Si	No	Si	Si	
5		Gastroenterología	Si	No	Si	Si	
6		Reumatología	Si	No	Si	Si	
7		Oncología	Si	No	Si	Si	
8	Pediatría	Pediatría	Si	Si	No	Si	
9		Neonatología	Si	Si	No	Si	
10	Ginecología y Obstetricia	Ginecología	Si	Si	No	Si	
11		Obstetricia	Si	Si	No	Si	
12	Equipos ambiente y auxiliares	Impresoras	Si	No	No	No	

... | Ubicaciones por entidad | Departamentos | **Servicios** | Especialidades | Asociar Referencias | Usuario tipc

Figura 4. Datos de los servicios en el documento *Excel*

Fuente: Elaboración propia

Para llevar los datos al sistema mediante la transformación lo primero es cargarlos, lo cual se hace mediante el paso *Excel Input* de la herramienta *Spoon* del *Pentaho*. En este caso se usan dos instancias del mismo, uno para extraer los datos de los servicios y otro para el nombre de la entidad, como se puede observar en la Figura 5.

Figura 5. Entrada de datos a la transformación

Fuente: Elaboración propia

Luego de tener los datos en la transformación es necesario validar que no falte alguno que sea de carácter obligatorio, empezando por el nombre de la entidad, que en caso de estar vacío este campo o no encontrarse registrado en el sistema alguna entidad con el nombre analizado, no se puede continuar el proceso. Para saber si el campo nombre de entidad es nulo se usa el paso *Filter rows* (entidad_nula), en caso positivo se detiene el proceso usando un paso *Abort* (valor_entidad_nulo). Si el valor no es nulo entonces este se busca en el sistema en el paso *Database lookup* (busqueda_entidad) y luego para saber si se encontró se usa el paso *Filter rows* (existe_entidad). Ver Figura 6.

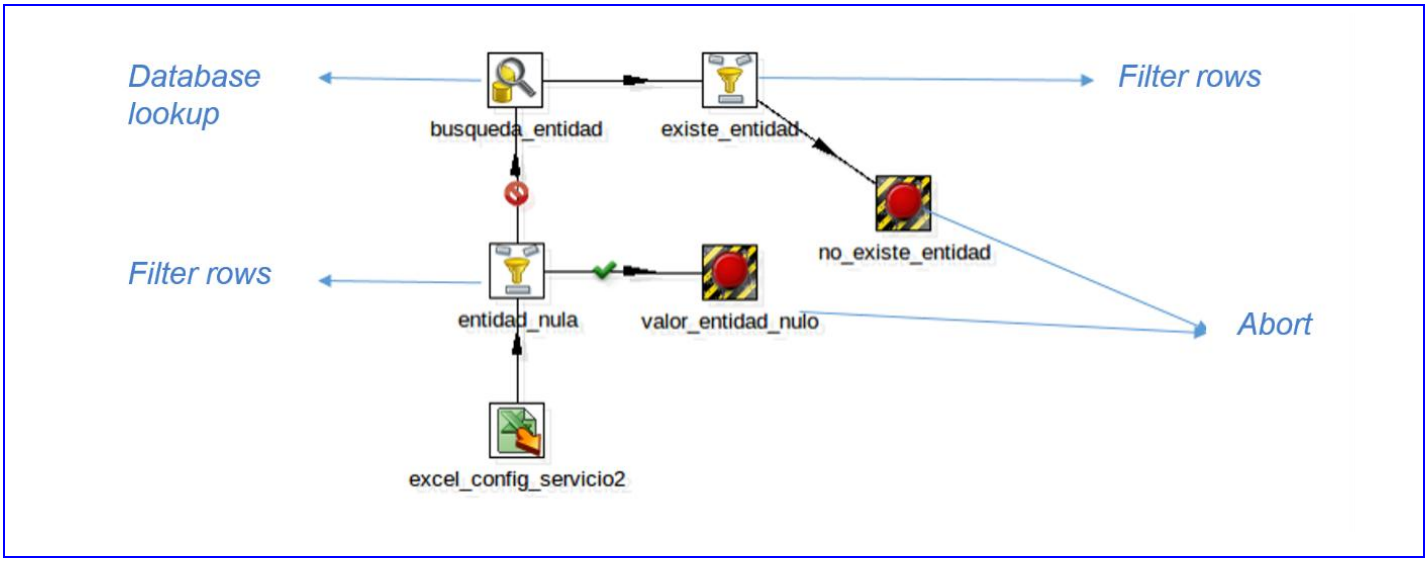


Figura 6. Validación del nombre de la entidad

Fuente: Elaboración propia

Una vez comprobado los casos anteriores, si todo es correcto es necesario unir la información extraída por cada uno de los pasos *Excel Input* para poder continuar el flujo. Para lo cual se usa el paso *Merge Join* (union_por_clave), este une dos flujos de datos por un campo clave con valores comunes en cada flujo, similar al JOIN en SQL, pero con la condición de que ambos flujos estén ordenados por la clave. Como en este caso no se cuenta con una clave común para cada flujo, se usan dos instancias del paso *Add constants:* constante_aux y constante_aux2, además de los correspondientes pasos de tipo *Sort rows:* ordenar_filas y ordenar_filas2 como se detalla en la Figura 7.

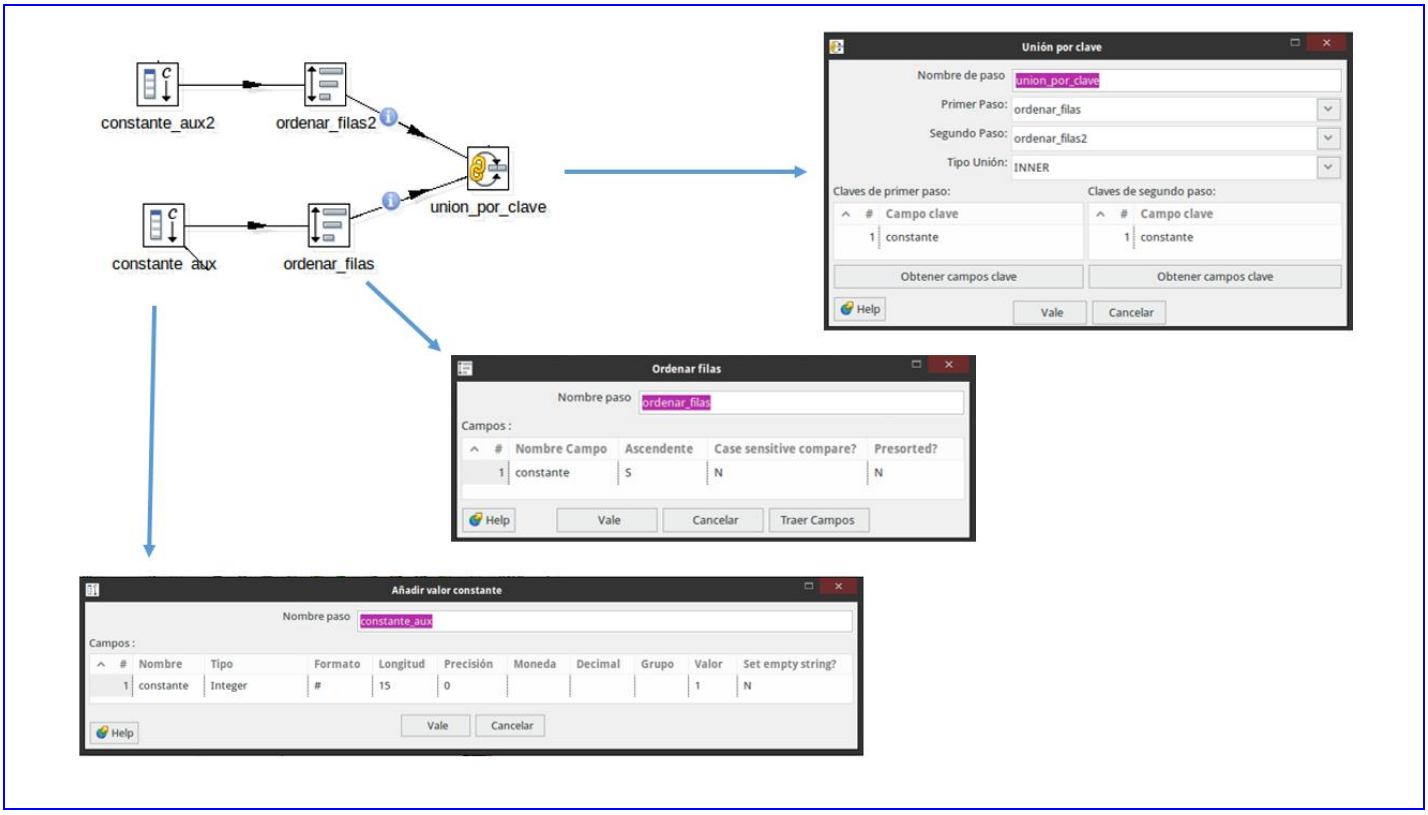


Figura 7. Unión de los flujos de datos por una clave

Fuente: Elaboración propia

Cuando estos pasos son ejecutados satisfactoriamente se obtiene el conjunto de los servicios asociados a la entidad junto a las dos contantes añadidas al flujo. Estas últimas no cumplen ningún objetivo durante el proceso restante por lo tanto son eliminadas en el paso *Select values* (seleccionar_valores_utiles). Luego de obtenidos los campos necesarios para continuar el flujo se valida que los de carácter obligatorio no sean nulos, similar a la validación del nombre de la entidad. La diferencia en este caso es que la transformación no es detenida, sino que se escribe un mensaje en el log de ejecución de la misma mediante el paso *Write to log* (valor_requerido_nulo). Además se eliminan las filas duplicadas mediante el paso *Unique rows* (serv_unicos) y se realiza la conversión de tipos de datos, así como la creación de nuevos campos a través del paso *Modified Java Script Value* (js_convertir_valores). Ver Figura 8.

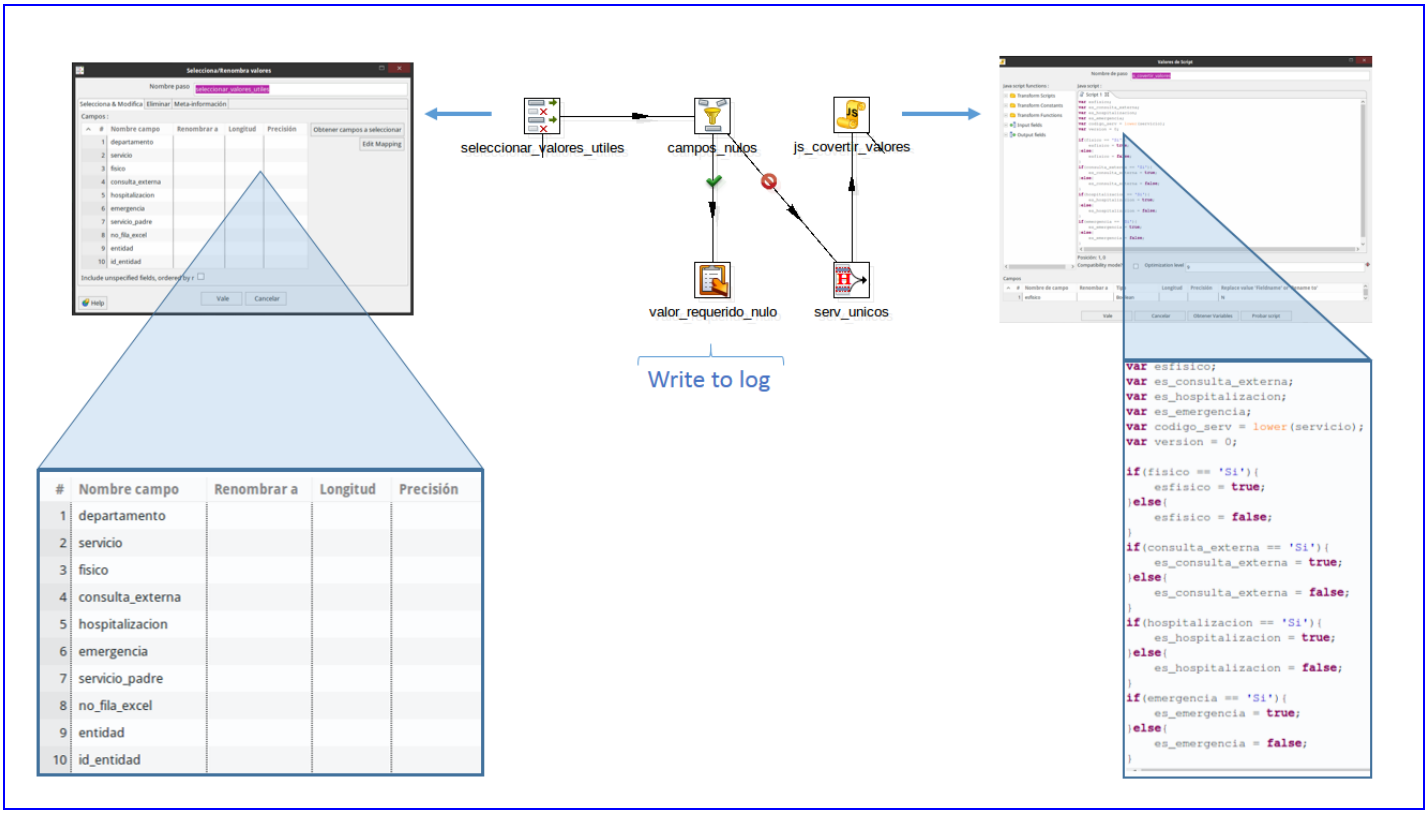


Figura 8. Validación y conversión de valores

Fuente: Elaboración propia

Una vez validados y obtenidos todos los valores a cargar en el sistema, teniendo en cuenta que estos no pueden ser insertados directamente a una tabla de la base de datos, sino que se necesita de algún valor almacenado en la misma, se procede a realizar su búsqueda. En este caso, para insertar un servicio se necesita el identificador del departamento al que pertenece, pero solo se cuenta con el nombre por lo que se realiza la búsqueda a través del paso *Merge Join* (buscar_departamento).

La salida obtenida de este paso solo contiene los datos de los servicios para los cuales se encontró el departamento, dada la necesidad de escribir en el *log* de ejecución los errores ocurridos, el flujo es dividido en dos (flujo 1 y flujo 2). Al pasar el flujo 1 por el paso buscar_departamento se añade un nuevo campo con el valor del identificador, este campo también tiene que ser añadido al flujo 2, mediante el paso constante_auxiliar, para que se puedan fusionar otra vez, con el uso del *Append streams* (unir_flujo_datos). De este paso se obtiene un solo flujo de datos pero con filas duplicadas, por tanto es necesario eliminarlas haciendo uso del paso eliminar_repetidos, ver Figura 9. Por último se comprueba la existencia de departamentos en el paso existe_departamento y de no existir alguno se escriben los detalles en el log de ejecución mediante paso no_departamento.

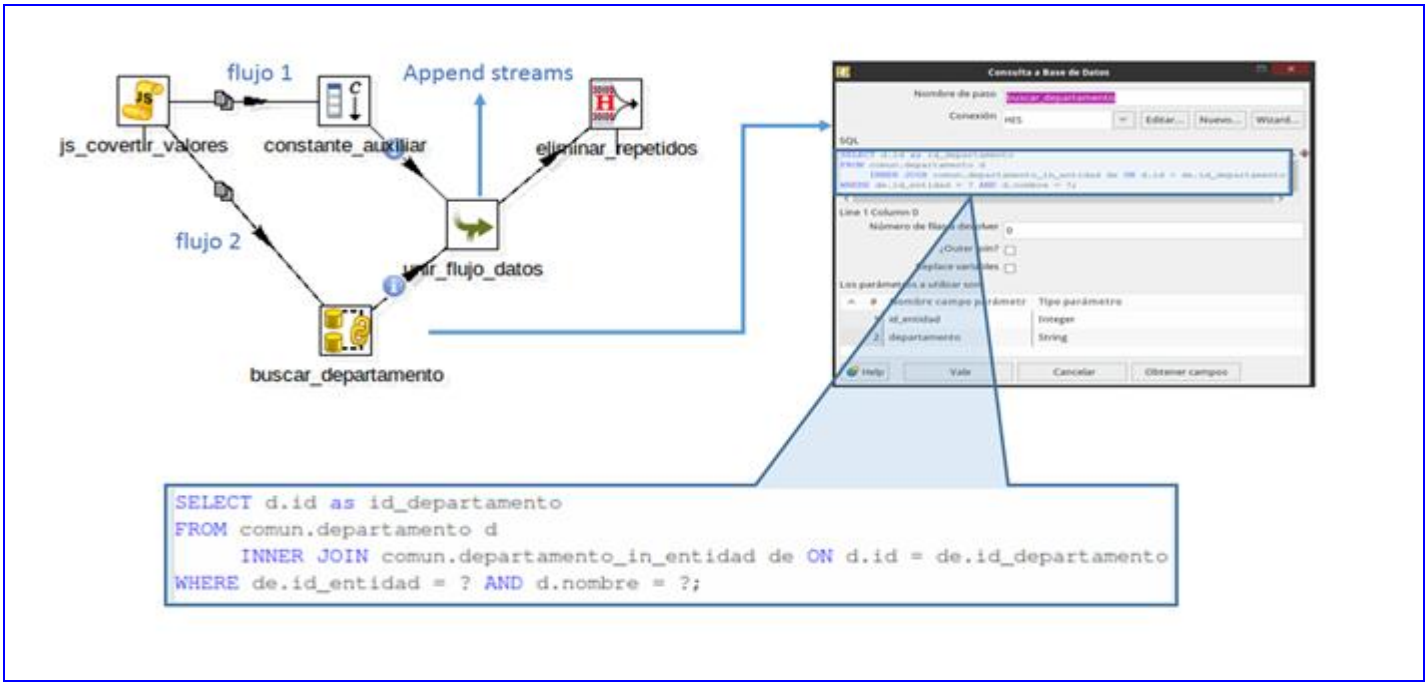


Figura 9. Búsqueda de datos en la base de datos

Fuente: Elaboración propia

Luego de este punto se comprueba si los servicios a cargar no están ya presentes en el sistema haciendo uso de los pasos buscar_servicio y existe_servicio, en el caso de los que no lo estén, se verifica si pertenecen a algún otro servicio mediante el paso no_es_hijo. En caso positivo es necesario buscar en la base de datos el identificador del servicio padre, ya que solo se tiene el nombre, para esto se usa el paso buscar_serv_padre, que al igual que para buscar el departamento, es necesario dividir los flujos y añadir constantes. Finalmente después de pasar por todos los pasos previos, el flujo de datos está listo para ser insertado en la base de datos mediante los pasos *Table output* (insertar_servicio, insertar_serv_in_ent, insertar_servicio_hijo e insertar_servicio_in_ent2). Ver Figura 10.

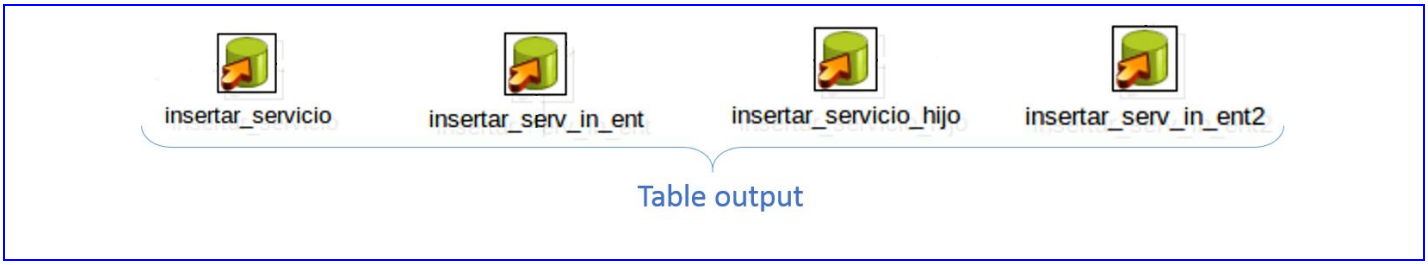


Figura 10. Pasos de persistencia de datos

Fuente: Elaboración propia

De esta manera queda lista la transformación para ser ejecutada desde el HIS (Anexo 1).

2.3.3 HIS-CESIM

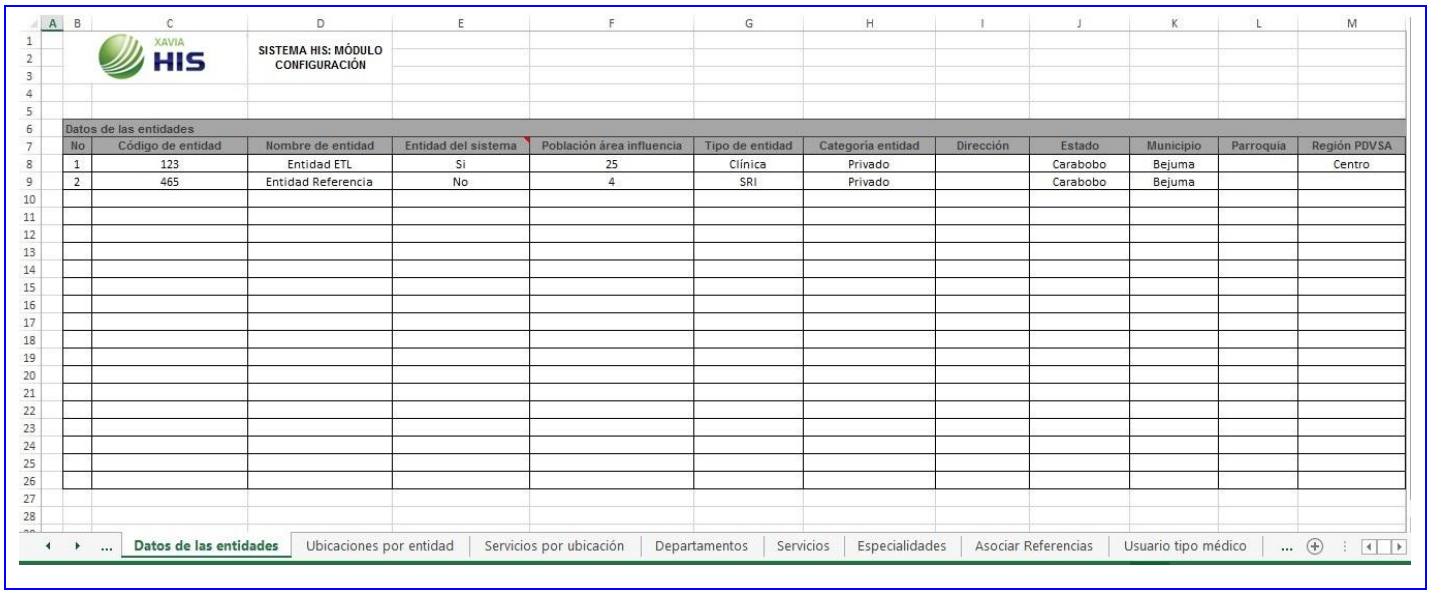
El HIS es el sistema de gestión de información hospitalaria que contiene un componente de carga de datos para llevar a cabo la ejecución de las transformaciones diseñadas con anterioridad. El mismo lleva a cabo la carga de datos al sistema haciendo uso de sus funcionalidades, las cuales permiten:

- Buscar de la lista de transformaciones diseñadas, aquella que se desee ejecutar en determinado momento.
- Adicionar un nuevo diseño de transformación que no se encuentre en la lista de transformaciones propuesta.
- Modificar los parámetros necesarios para establecer una conexión satisfactoria entre los archivos *Excel* y la Base de Datos.
- Eliminar de la lista de transformaciones aquellas que no se vayan a ejecutar.
- Ejecutar las transformaciones de forma lineal y acorde a las necesidades de cada institución.

A estas funcionalidades se logra acceder mediante una interfaz intuitiva y amigable, la cual permite proveerle al sistema los datos imprescindibles para su correcto funcionamiento

2.3.4 Fuentes de origen de datos

Las fuentes de origen de datos están compuestas por libros *Excel* de tipo Libro de *Excel* 97-2003 (*.xls) los cuales son creados para recoger datos de módulos específicos tales como: Configuración, Laboratorio, Almacén, Farmacia, Banco de Sangre y Bloque Quirúrgico. Cada uno de estos libros *Excel* contienen hojas especializadas en un área determinada del módulo elegido y diseñadas con un formato intuitivo para su mejor comprensión. Ver Figura 11.



No	Código de entidad	Nombre de entidad	Entidad del sistema	Población área influencia	Tipo de entidad	Categoría entidad	Dirección	Estado	Municipio	Parroquia	Región PDVSA
1	123	Entidad ETL	Si	25	Clínica	Privado		Carabobo	Bejuma		Centro
2	465	Entidad Referencia	No	4	SRI	Privado		Carabobo	Bejuma		

Figura 11. Libro *Excel* correspondiente al módulo Configuración

Fuente: Elaboración propia

HIS Módulo Configuración es el libro *Excel* encargado de recoger la información necesaria para el correcto funcionamiento del módulo Configuración. El mismo está compuesto por 14 hojas detalladas a continuación:

1. Presentación*:* Es la hoja principal del libro, en ella se registran las reglas de confiabilidad, el objetivo del documento, las referencias y el control del documento.
2. Datos de las entidades*:* Esta hoja recopila la información de todas las entidades que serán insertadas en el sistema.
3. Ubicaciones por entidad*:* En esta hoja se relaciona la entidad en cuestión con sus ubicaciones, de las cuales se recoge el tipo de ubicación y la ubicación a la que pertenece.
4. Servicios por ubicación*:* Permite relacionar las ubicaciones de la entidad en cuestión con los servicios que son prestados en estas.
5. Departamentos*:* Permite registrar la información concerniente a los departamentos de una determinada entidad y brinda la posibilidad de especificar si estos son clínicos o no.
6. Servicios*:* Permite registrar los datos de los servicios que brinda una determinada entidad.
7. Especialidades*:* Permite registrar el listado de especialidades por servicios.
8. Asociar referencias*:* Permite registrar el listado de entidades y servicios a los cuales se pueden emitir referencias.
9. Usuario tipo médico*:* Permite registrar los datos de los usuarios tipo médico.

10. Usuario tipo bioanalista*:* Permite registrar los datos de los usuarios tipo bioanalista.
11. Usuario tipo enfermero*:* Permite registrar los datos de los usuarios tipo enfermero.
12. Otros usuarios*:* Permite registrar los datos de otros tipos de usuarios.
13. Camas*:* Permite registrar el listado de camas de la entidad en cuestión de acorde a la ubicación en que se encuentran y al servicio que prestan.
14. Nomencladores*:* Esta hoja contiene los datos que son de carácter obligatorios, los cuales se utilizan en las demás hojas para facilitar el trabajo con ellas.

2.3.5 Base de datos del sistema

En la base de datos en donde se almacena toda la información necesaria para el funcionamiento del HIS. La cual es llenada inicialmente con los datos recogidos en los documentos *Excel*, mediante la ejecución de las transformaciones en el sistema.

2.4 Especificación de los requisitos de software

En este epígrafe se exponen los requisitos funcionales y no funcionales con los cuales tiene que cumplir la aplicación desarrollada.

2.4.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que este no debe hacer, aunque en la mayoría de las ocasiones describen lo que si debe hacer (43). Los requisitos funcionales que se identificaron a partir de los procesos de negocio estudiados y de las actividades a automatizar detectadas se muestran a continuación:

RF1: Buscar transformación.

RF2: Adicionar transformación.

RF3: Modificar transformación.

RF4: Eliminar transformación.

RF5: Ejecutar transformación.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que se refieren a las propiedades o cualidades que el producto debe tener, son restricciones de los servicios o funciones ofrecidas por este. Estos requisitos incluyen

restricciones de tiempo sobre el proceso de desarrollo y pueden ser aplicados al sistema en su totalidad. Estos se centran en responder a las necesidades que deben satisfacerse con la creación del producto a desarrollar, detallando sus características en cuanto a rendimiento del sistema, protección, disponibilidad, fiabilidad, tiempo de respuesta, capacidad de almacenamiento, entre otras propiedades emergentes (43). A continuación se muestran aquellos requisitos no funcionales correspondientes al componente de carga de datos del Sistema de Información Hospitalaria del CESIM:

- Usabilidad

El componente de carga de datos presenta una interfaz amigable que permite a los usuarios una fácil interacción con el mismo, logrando que estos puedan adquirir las habilidades necesarias para trabajar con él en un tiempo reducido. Además utiliza un lenguaje acorde al ambiente de trabajo donde se aplica o sea correspondiente con la terminología del Sistema de Información Hospitalaria.

- Viabilidad

El componente de carga de datos del Sistema de Información Hospitalaria del CESIM es viable, ya que durante el proceso de instalación del sistema es necesario cargar un gran volumen de información, lo cual resulta lento y engorroso realizarlo de forma manual. Con este componente se pretende automatizar la carga de información al sistema, para agilizar y lograr un mayor control del proceso.

- Eficiencia

El componente minimiza el tiempo necesario para realizar la carga de datos. El mismo respeta las buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos, además optimiza el trabajo con cadenas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

- Requisitos de portabilidad

Se puede utilizar desde cualquier sistema operativo garantizando su característica multiplataforma.

- Requisitos de interfaz de usuario

Las ventanas del sistema contienen los datos claros y bien estructurados, lo que permite la interpretación correcta de la información. La entrada de datos incorrecta es detectada claramente e informada al usuario. Todos los textos y mensajes en pantalla aparecen en el idioma de preferencia del usuario siempre que este sea soportado por la aplicación. El diseño de la interfaz del sistema responde a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.

- Requisitos de instalación

El componente forma parte del Sistema de Información Hospitalaria del CESIM, por lo que puede desplegarse en los sistemas operativos Windows 7 o superior y Linux, utilizando *Java Runtime Environment*, *JBoss Server* y *PostgreSQL*. El usuario debe disponer de un navegador *web*, el cual puede ser *Google Chrome, Internet Explorer 7.0, Firefox 4.0* o versiones superiores de estos.

- Requisitos de hardware

Estos requisitos de hardware se definieron por la dirección del CESIM atendiendo a experiencias anteriores de despliegue y se definen como requisitos mínimos en cada una las ofertas comerciales del mismo.

Estaciones de trabajo:

El componente puede utilizarse en clientes que posean como mínimo 1GB de memoria RAM y procesadores de 2 núcleos.

Servidores:

Los servidores, por su parte, deben tener alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

- ✓ Servidores de base de datos: Procesador 8 núcleos, 8 GB de memoria RAM, 80 GB de disco duro.
- ✓ Servidores de aplicaciones: Procesador 8 núcleos, 16 GB de memoria RAM, 80 GB de disco duro.

2.5 Modelo de casos de uso del sistema

El modelado de casos de uso es una técnica de modelado usada para describir el comportamiento de un sistema informático. Son concebidos como descripciones funcionales del sistema, las cuales detallan como los actores pueden hacer uso de este. Dentro de sus principales componente se encuentran los actores, los casos de uso y las relaciones entre estos. Los actores representan un tipo de usuario del sistema. Se entiende como usuario cualquier ente externo que interactúe con el sistema. No tiene por qué ser un ser humano, puede ser otro sistema informático, unidades organizativas o empresas. Al interactuar con el sistema dicho usuario debe tener algún tipo de relación con al menos un caso de uso. (44)

Un caso de uso se define como una secuencia de acciones que realiza el sistema y que conducen a un resultado observable, representando una funcionalidad completa. Los actores y los casos de uso son conectados mediante asociaciones denominadas líneas de comunicación, las cuales representan las

relaciones entre estos. Dentro de las ventajas de un modelado de casos de uso se encuentra la facilidad para ser interpretados, lo que hace que sean de gran utilidad en la comunicación con el cliente. (44)

2.5.1 Definición del actor del sistema

Tabla 3. Descripción del actor del sistema

Actor	Objetivos
Administrador	Es el encargado de: • Adicionar al sistema las transformaciones necesarias para cargar la información deseada. • Buscar del listado de transformaciones una determinada transformación. • Modificar los parámetros necesarios en las transformaciones para establecer una conexión satisfactoria. • Eliminar las transformaciones que no se utilizan. • Ejecutar las transformaciones en el orden correspondiente.

2.5.2 Diagrama de casos de uso

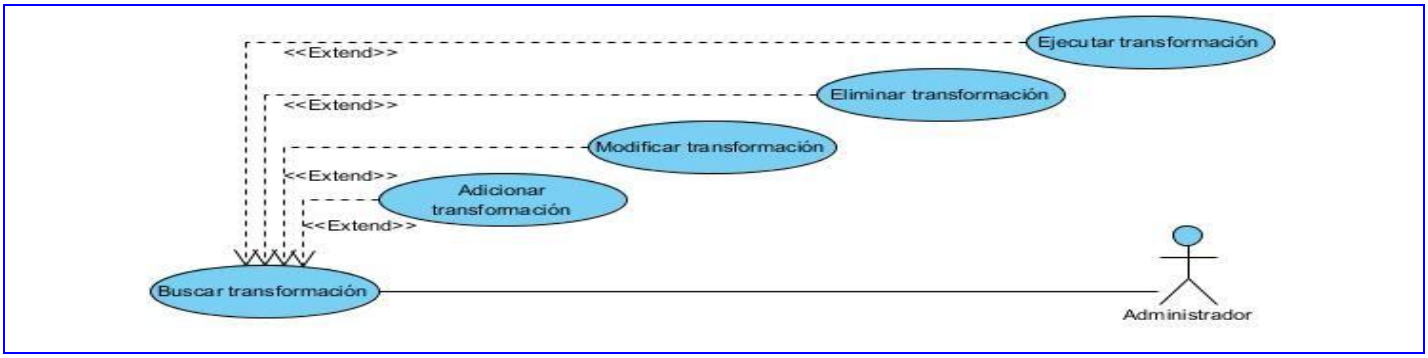


Figura 12. Diagrama de casos de uso

Fuente: Elaboración propia

2.5.3 Especificación de los casos de uso

CU1. Adicionar transformación

Tabla 4. Descripción del caso de uso: Adicionar transformación

Objetivo	Adicionar una transformación en el sistema.
Actores	Administrador

Resumen	El caso de uso inicia cuando el actor selecciona la opción Adicionar transformación, el sistema muestra una vista donde brinda la posibilidad de buscar la transformación a adicionar, el actor selecciona la trasformación deseada y accede a la opción Abrir, luego selecciona la opción Aceptar y el caso de uso termina.
Complejidad	Media
Prioridad	Auxiliar
Precondiciones	Para adicionar una transformación, esta debe haber sido creada.
Postcondiciones	Se adicionó una transformación por el actor.

Flujo de eventos		
Flujo básico Adicionar transformación		
	Actor	**Sistema**
1.	El caso de uso inicia cuando se accede a la opción: Adicionar transformación.	
2.		Permite: • Examinar Brinda la opción de navegar por los directorios del sistema para lograr encontrar el que contiene las transformaciones • Terminar: Ver **Evento 1**: Terminar
3.	Accede a la opción: Examinar.	
4.		Brinda la opción de navegar por los directorios del sistema para lograr encontrar el que contiene las transformaciones.
5.	Busca el directorio en donde se encuentran las transformaciones, selecciona la transformación a adicionar y accede a la opción: Abrir.	
6.		Muestra la transformación examinada en una lista temporal de transformaciones.
7.		Permite: • Aceptar

		Adiciona las transformaciones de la lista temporal al sistema. • Cancelar. Ver **Evento 2:** Cancelar operación.
8.		Verifica que se trate de una transformación correcta, en caso de estar corrupta muestra un mensaje brindando información acerca de lo ocurrido. Ver **Evento 3:** Transformación corrupta.
9.		Termina el caso de uso.
Flujos alternos		
Evento 1. Terminar.		
	Actor	**Sistema**
1.	Accede a la opción: Terminar	
2.		Regresa a la vista anterior
3.		Termina el caso de uso
Evento 2. Cancelar operación.		
1.	Accede a la opción: Cancelar	
2.		Elimina de la lista temporal de transformaciones la elegida
3.		Regresa al paso 2 del **Flujo Normal de Eventos**.
4.		Termina el caso de uso
Evento 3. Transformación corrupta		
1.		Muestra el mensaje: Transformación corrupta
2.	Accede a la opción: Aceptar.	
3.		Regresa al paso 2 del **Flujo Normal de Eventos**.
4.		Termina el caso de uso.
Relaciones	**CU incluidos**	No existen
	CU extendidos	No existen
Requisitos no funcionales		
Asuntos	No existen	

pendientes	

CU2. Buscar transformación

Tabla 5. Descripción del caso de uso: Buscar transformación

Objetivo	Buscar un conjunto de transformaciones en el sistema.
Actores	Administrador
Resumen	El caso de uso inicia cuando el actor accede a la opción Buscar transformación, el sistema brinda la posibilidad de introducir el criterio de búsqueda para localizar la transformación, el actor introduce los datos que considera como criterio para realizar una búsqueda, el sistema busca y muestra las transformaciones que cumplen con el criterio de búsqueda, el caso de uso termina.
Complejidad	Media
Prioridad	Secundario
Precondiciones	Deben estar creadas las transformaciones en el sistema.
Postcondiciones	Se buscaron las transformaciones correctamente.
Flujo de eventos	
Flujo básico Buscar transformación	

	Actor	**Sistema**
1.	El caso de uso inicia cuando se accede a la opción: Buscar transformación.	
2.		Muestra un listado de transformaciones con el siguiente atributo: • Nombre de la transformación Ordenado alfabéticamente por el nombre de la trasformación, mostrando la cantidad de elementos configurados a mostrar por página, permitiendo navegar por el resultado. Permite: • Ordenar el resultado por el atributo: Nombre de la transformación de manera ascendente o descendente. Ver **Evento 1:** Ordenar el resultado

		ascendente o descendentemente por un atributo. • Ordenar el resultado por el orden ejecución. Ver **Evento 2:** Ordenar resultado por el orden de ejecución. • Si no se encuentra ninguna transformación Ver **Evento 3:** No se encontró transformación.
3.		Permite: • Realizar una búsqueda simple. Ver **Evento 4:** Realizar una búsqueda simple. • Adicionar transformación. Ver **Evento 5:** Adicionar transformación. • Modificar transformación. Ver **Evento 6:** Modificar transformación. • Eliminar transformación. Ver **Evento 7:** Eliminar transformación. • Ejecutar transformación. Ver **Evento 8:** Ejecutar transformación.
4.		Termina el caso de uso.
Flujos alternos		
Evento 1. Ordenar el resultado ascendente o descendentemente por un atributo.		
	Actor	**Sistema**
1.	Selecciona un atributo del resultado para ordenarlo ascendente o descendentemente por el atributo seleccionado.	
2.		Reordena y muestra el resultado ascendente o descendentemente por el atributo seleccionado.
3.		Regresa al paso 2 del **Flujo Normal de Eventos.**
Evento 2. Ordenar el resultado por el orden de ejecución.		
1.	Selecciona un atributo del resultado para ordenarlo por el orden de ejecución.	

2.		Reordena y muestra el resultado ascendente o descendentemente por el atributo seleccionado
3.		Regresa al paso 2 del **Flujo Normal de Eventos.**
Evento 3. No se encontró transformación.		
1.		Muestra el mensaje de información: No se encontró ninguna transformación.
2.		Regresa al paso 2 del **Flujo Normal de Eventos.**
Evento 4. Realizar una búsqueda simple.		
1.		Brinda la posibilidad de introducir los criterios de búsqueda simple: • Nombre de la transformación
2.	Introduce los datos que considera como criterios para realizar una búsqueda simple y selecciona la opción: Buscar.	
3.		Busca los datos de las transformaciones que cumplen con el criterio de búsqueda.
4.		Regresa al paso 2 del **Flujo Normal de Eventos.**
Evento 5. Adicionar transformación.		
1.	Selecciona la opción: Adicionar transformación.	
2.		Permite adicionar transformación. Ver caso de uso: Adicionar transformación.
3.		Termina el caso de uso.
Evento 6. Modificar transformación.		
1.	Selecciona la opción: Modificar transformación.	
2.		Permite modificar transformación. Ver caso de uso: Modificar transformación.
3.		Termina el caso de uso.
Evento 7. Eliminar transformación.		

1.	Selecciona la opción: Eliminar transformación.	
2.		Permite eliminar la transformación. Ver caso de uso: Eliminar transformación.
3.		Termina el caso de uso.
Evento 8. Ejecutar transformación.		
1.	Selecciona la opción: Ejecutar transformación.	
2.		Permite ejecutar la transformación. Ver caso de uso: Ejecutar transformación.
3.		Termina el caso de uso.

Relaciones	**CU incluidos**	No existen
	CU extendidos	Adicionar transformación. Ver CU Adicionar transformación Modificar transformación. Ver CU Modificar transformación Eliminar transformación. Ver CU Eliminar transformación Ejecutar transformación. Ver CU Ejecutar transformación
Requisitos no funcionales	No existen	
Asuntos pendientes	No existen	

CU3. Modificar transformación (Ver artefacto: Especificación de casos de uso)

CU4. Eliminar transformación (Ver artefacto: Especificación de casos de uso)

CU5. Ejecutar transformación (Ver artefacto: Especificación de casos de uso)

Conclusiones parciales

- La descripción del proceso de carga de datos en el Sistema de Información Hospitalaria del CESIM a través del modelo conceptual, posibilita la rápida identificación de los requisitos funcionales y no funcionales que sustentan el desarrollo de la propuesta.

Capítulo 3. Diseño del componente.

En este capítulo se describe la arquitectura y los patrones de diseño definidos para el desarrollo del componente de carga de datos. Se muestra la composición por paquetes del modelo de diseño, así como el diagrama de clases de diseño. Además se brinda una descripción de las clases identificadas para su posterior implementación.

3.1 Descripción de la arquitectura. Fundamentación

La arquitectura de un sistema de *software* constituye la estructura del sistema la cual incluye los componentes de *software*, las propiedades visibles externas de esos componentes, y las relaciones entre ellos. Una arquitectura de *software* es un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del *software* para un sistema de información (45). Dicha arquitectura permite a los programadores, diseñadores y analistas trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

En la creación de las funcionalidades asociadas al componente de carga de datos se hace uso del patrón arquitectónico Modelo Vista Controlador (expuesto en el capítulo 1). Dicho patrón es usado en la creación de aplicaciones *web* por las ventajas que proporcionan sus tres componentes: el modelo, para la administración de los datos; la vista, para mostrar los datos del modelo al usuario y el controlador, encargado de gestionar las entradas del usuario. Con la utilización de este patrón se logra separar la vista del modelo, permitiendo así reutilizar los componentes usados en este último.

La capa de la vista o capa de presentación está compuesta por páginas XHTML, desarrolladas básicamente con JSF, utilizando las librerías *Ajax4JSF* y *RichFaces*, las cuales se complementan con la plataforma de integración *JBossSeam*. En la capa del modelo se realizan tareas como la carga, modificación y validación de los datos antes de ser persistidos en la base de datos. La realización de estas tareas se llevan a cabo utilizando las librerías del *Pentaho Data Integration* que han sido embebidas dentro del sistema. Por otra parte la capa del controlador o capa de negocio compuesta por las clases *Java* controladoras, define la lógica del negocio y se encarga de manipular, controlar y procesar la información.

3.2 Modelo de diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose gráficamente en la lógica de los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación tienen impacto en el sistema a considerar.

Este modelo sirve de abstracción de la implementación y es utilizado como entrada fundamental de las actividades de implementación. (46)

3.2.1 Patrones de diseño

Para el diseño del componente de carga de datos, se tienen en cuenta los patrones generales de *software* para asignar responsabilidades (GRAPS, por sus siglas en inglés) (47). La clase BuscarTransformacion fue creada con el objetivo de permitirle al usuario realizar una búsqueda simple de las trasformaciones deseadas. En esta se evidencia el uso del patrón **alta cohesión** al realizarse solamente en ella la función de búsqueda y designarse las demás funciones a otras clases. El patrón **creador** se manifiesta en la clase AdicionarTransformacion al ser la encargada de crear y guiar la asignación de responsabilidades relacionadas con la creación de objetos de tipo Transformacion.

Por otra parte el uso del patrón **experto** se ve reflejado en la clase EjecutarTransformacion, pues esta contiene las instrucciones y la información necesaria para ejecutar la transformación. Por último se hace uso del patrón **bajo acoplamiento** que soporta el diseño de clases más independientes y reduce el impacto de los cambios, pues si se quisiera modificar o ejecutar una trasformación solamente habría que llamar a las clases encargadas de estas funcionalidades.

3.2.2 Diagrama de paquetes

Durante el desarrollo de la presente investigación se diseña un conjunto de clases para la gestión de las transformaciones previamente diseñadas en la herramienta ETL seleccionada. En la Figura 13 se observa la estructura de paquetes utilizada para la realización del modelo de diseño.

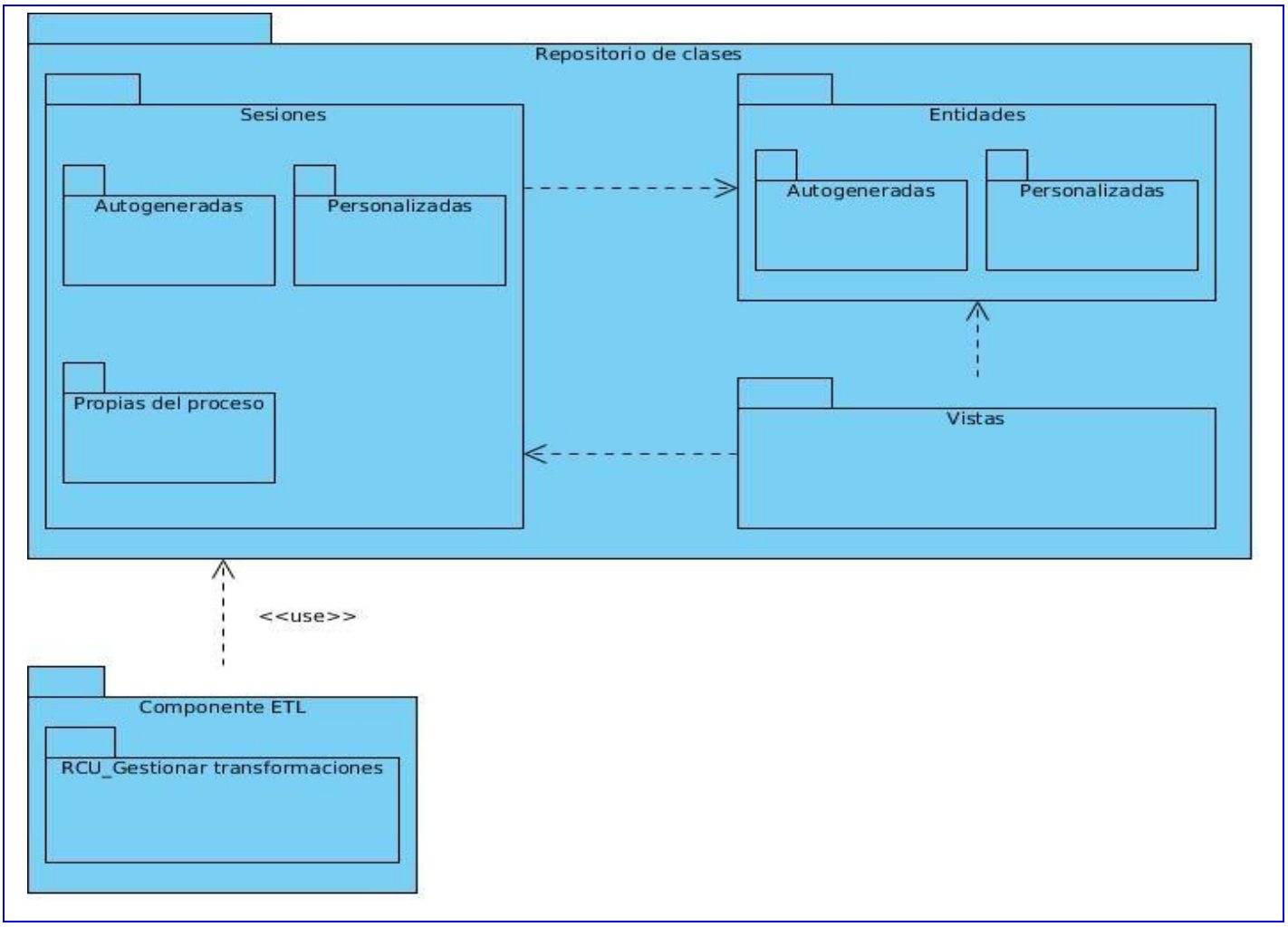


Figura 13. Diagrama paquetes

Fuente: Elaboración propia

3.2.3 Descripción de las clases de diseño

El paquete **Componente ETL** contiene las clases implementadas para gestionar las transformaciones diseñadas durante el desarrollo de la propuesta de solución. A continuación se muestran las clases contenidas en este paquete:

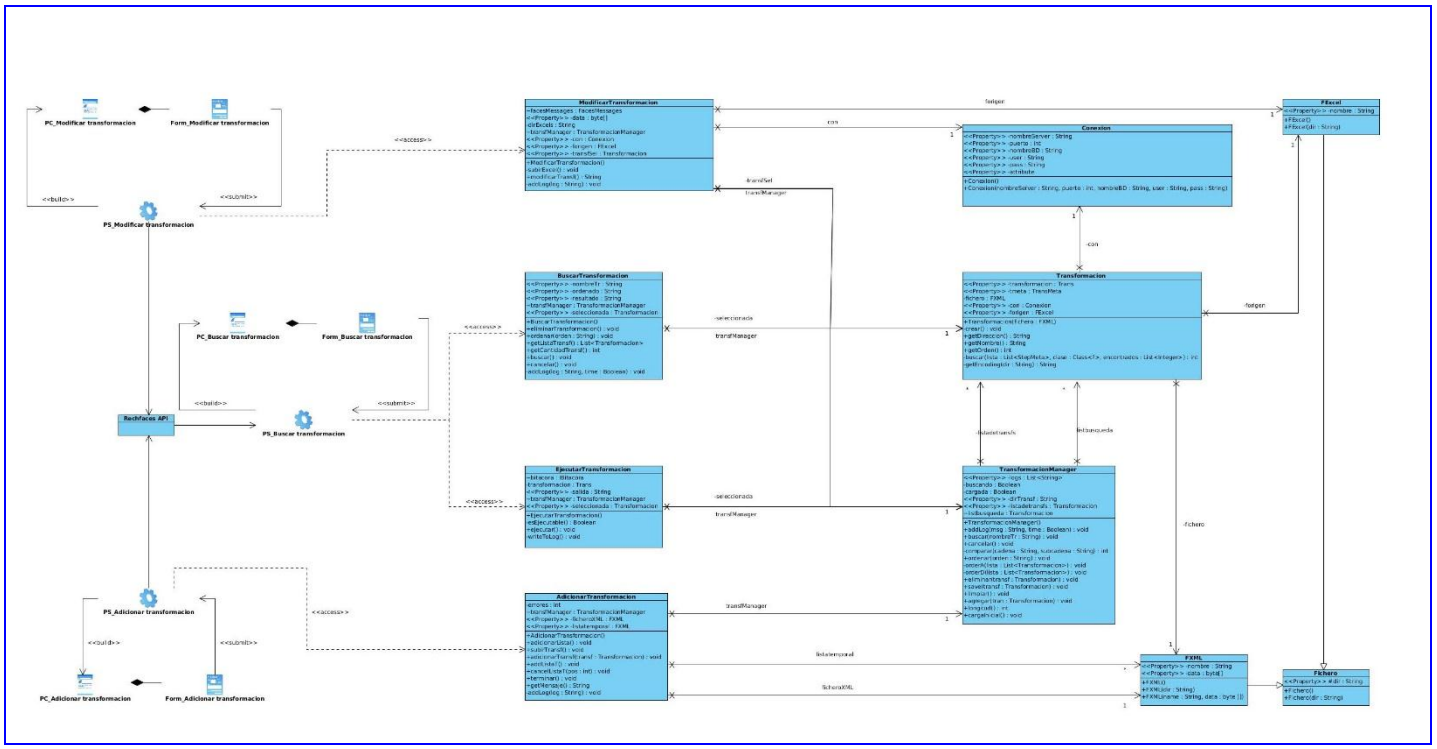


Figura 14. Diagrama clases del diseño

Fuente: Elaboración propia

Transformación

Propósito: Es una clase controladora personalizada que permite manejar una transformación.

Figura 15. Clase Transformación

Fuente: Elaboración propia

TransformacionManager

Propósito: Es una clase controladora personalizada que permite la gestión directa sobre la lista de transformaciones.

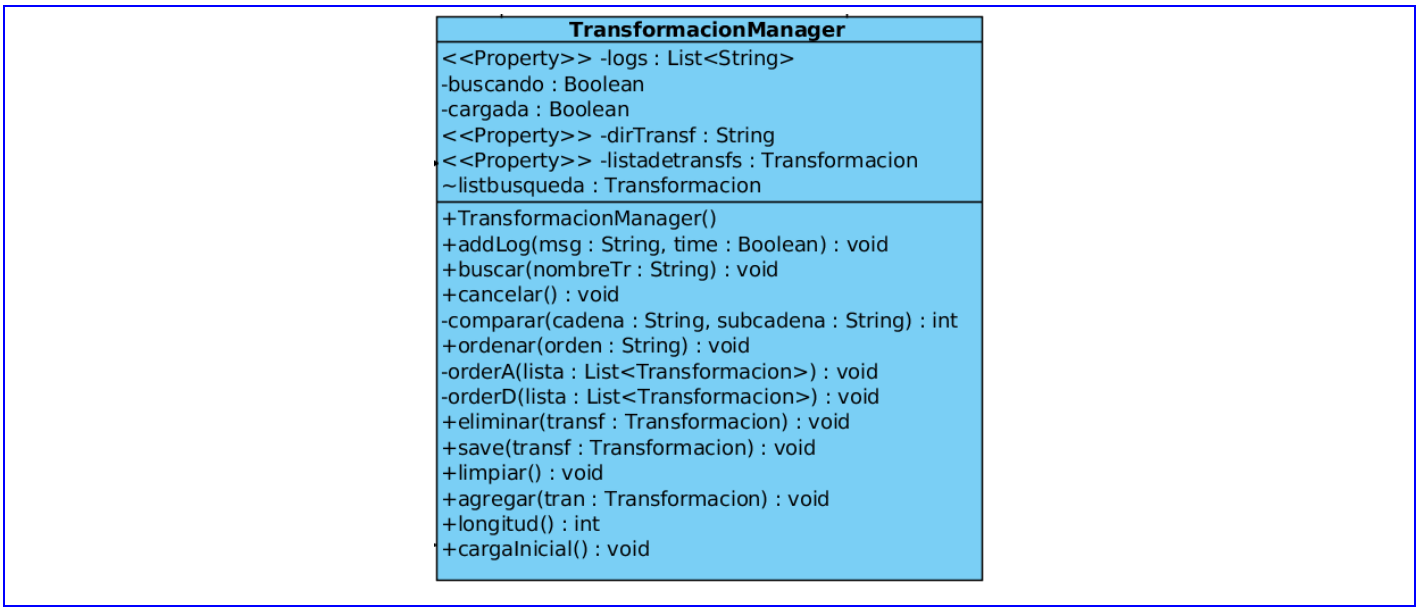


Figura 16. Clase TransformacionManager

Fuente: Elaboración propia

AdicionarTransformacion

Propósito: Es una clase controladora personalizada que permite adicionar transformaciones a lista existente en el sistema.

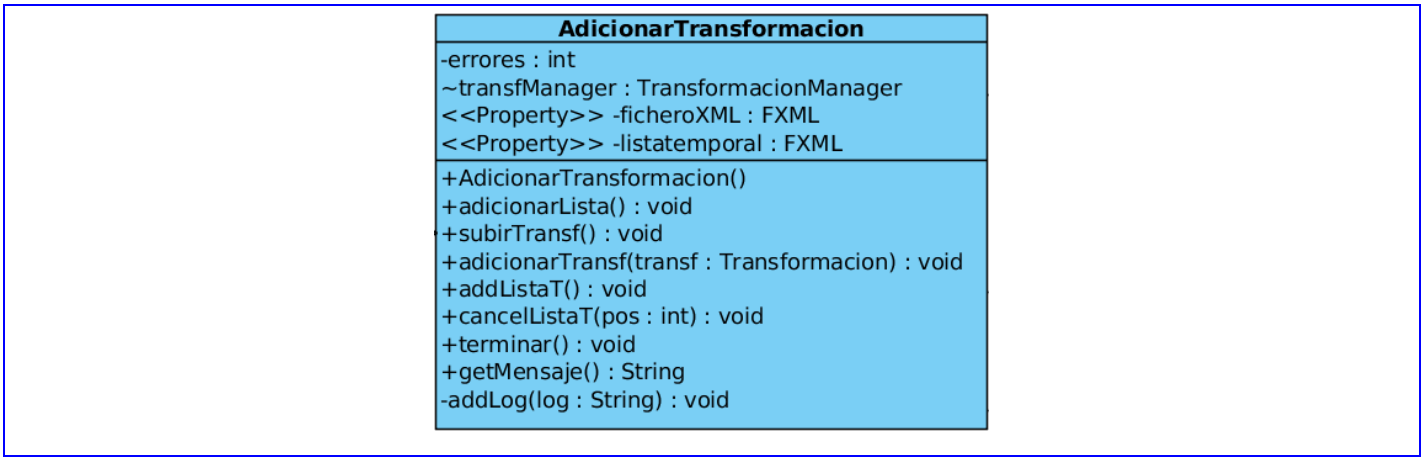


Figura 17. Clase AdicionarTransformacion

Fuente: Elaboración propia

BuscarTransformacion

Propósito: Es una clase controladora personalizada que permite buscar transformaciones dentro de la lista existente en el sistema, así como eliminarlas.

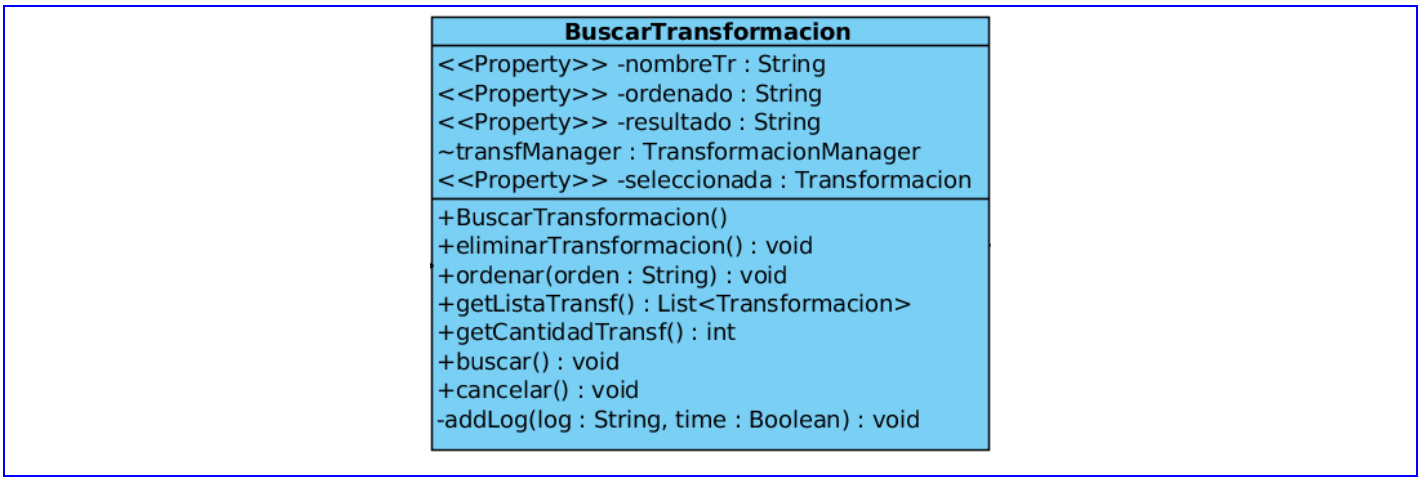


Figura 18. Clase BuscarTransformacion

Fuente: Elaboración propia

ModificarTransformacion

Propósito: Es una clase controladora personalizada que permite modificar las transformaciones existente en el sistema.

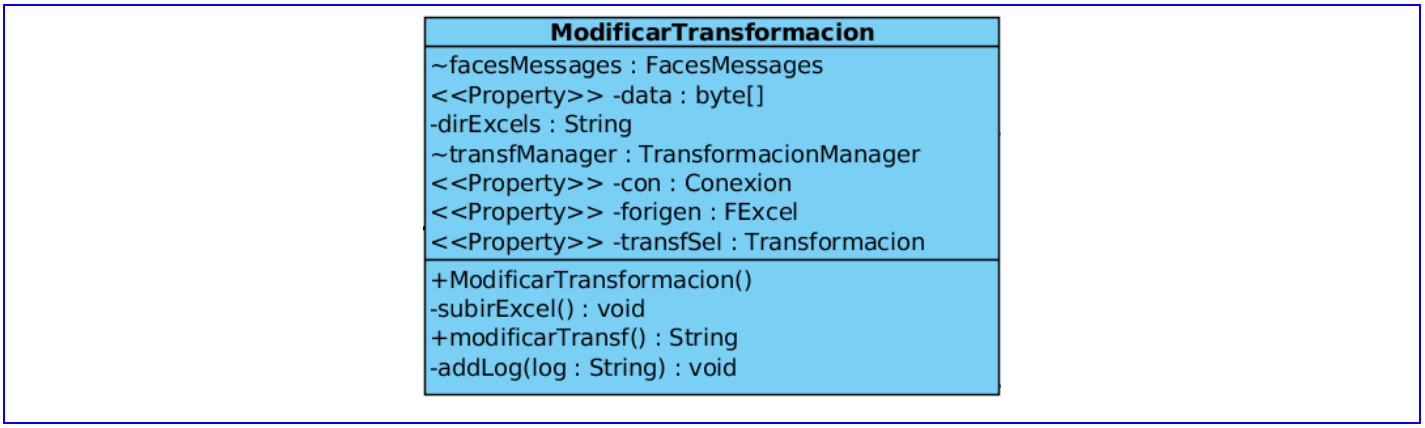


Figura 19. Clase ModificarTransformacion

Fuente: Elaboración propia

EjecutarTransformacion

Propósito: Es una clase controladora personalizada que permite ejecutar las transformaciones existente en el sistema.

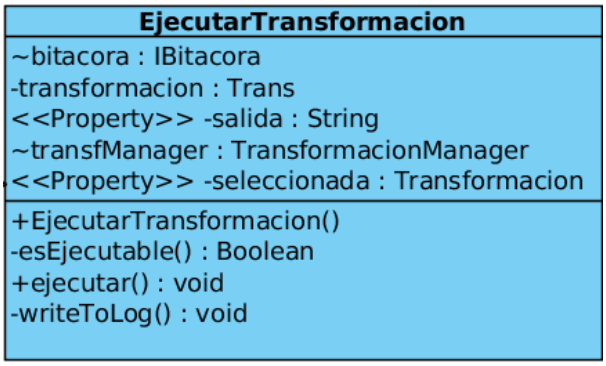


Figura 20. Clase EjecutarTransformacion

Fuente: Elaboración propia

Conclusiones parciales

Al finalizar la realización de este capítulo se arriban a las siguientes conclusiones:

- El empleo de los patrones Creador, Experto, Alta Cohesión y Bajo Acoplamiento propicia la asignación de responsabilidades y el diseño eficaz del componente.
- La confección del Diagrama de clases del diseño permite comprender fácilmente el funcionamiento del componente.

Capítulo 4. Aplicación y validación.

En este capítulo se describe el proceso de aplicación y validación del componente de carga de datos en el HIS del CESIM. Por otra parte se realizan pruebas de carga de datos en las que se comparan los resultados obtenidos al utilizar el componente con los obtenidos al realizar el proceso sin la utilización del mismo. Durante la realización de las pruebas se tiene en cuenta el tiempo incurrido en realizar la carga de los datos y la persistencia satisfactoria de estos al concluir el proceso.

4.1 Diagrama de componentes

Un diagrama de componentes muestra la organización y las dependencias lógicas entre un conjunto de componentes *software*, sean estos de código fuente, librerías, binarios o ejecutables. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando los subsistemas y sus dependencias a la hora de importar el código y organizando estos subsistemas en capas. El mismo es utilizado para modelar la vista estática y dinámica de un sistema, además de mostrar organización entre sus componentes. (48)

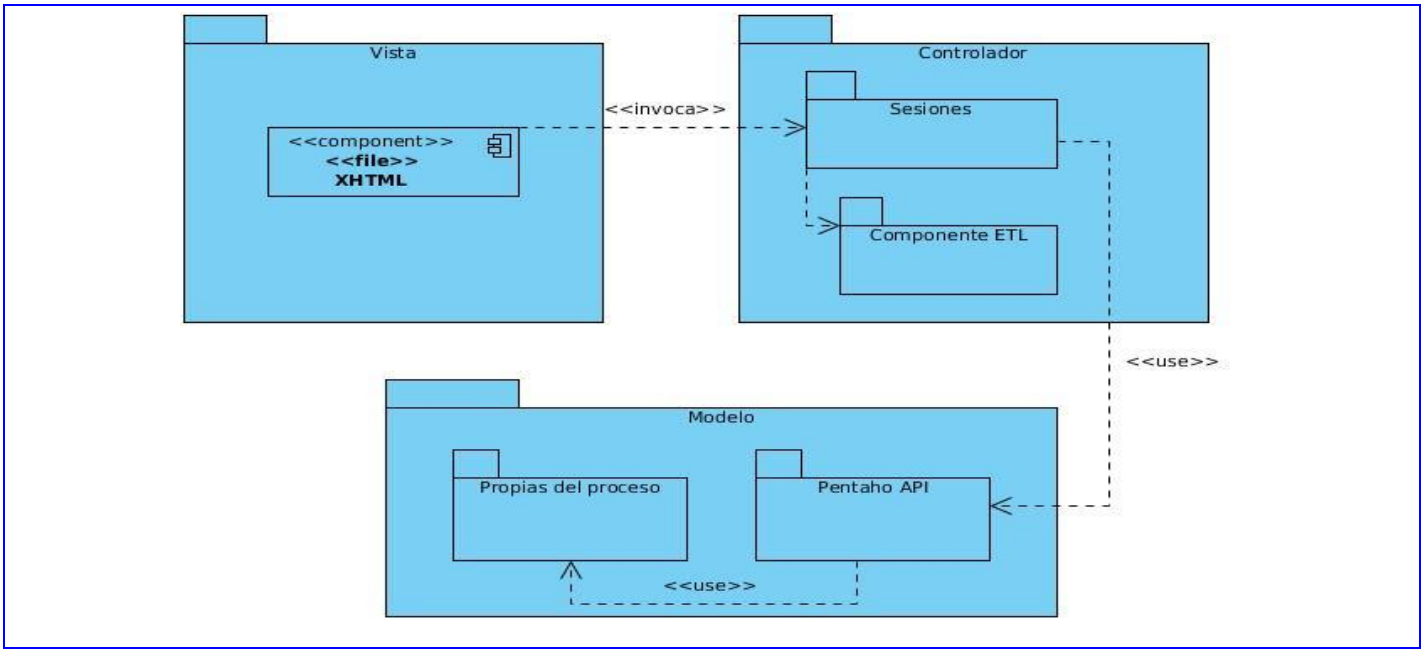


Figura 21. Diagrama de componente

Fuente: Elaboración propia

4.2 Aplicación del componente de carga de datos en el Sistema de Información Hospitalaria

Para ejecutar y comprobar el funcionamiento de las transformaciones diseñadas para la carga de datos, se agregó una nueva funcionalidad al módulo Configuración del Sistema de Información Hospitalaria del

CESIM. Se implementaron las clases Transformacion, AdicionarTransformacion, BuscarTransformacion, ModificarTransformacion, EjecutarTransformacion y TransformacionManager descritas en la sección 2.6 del capítulo 2, las cuales ofrecen las funcionalidades básicas del componente de carga de datos.

La nueva funcionalidad denominada “Carga de datos” (Figura 22) permite acceder a las funcionalidades del componente de carga de datos.

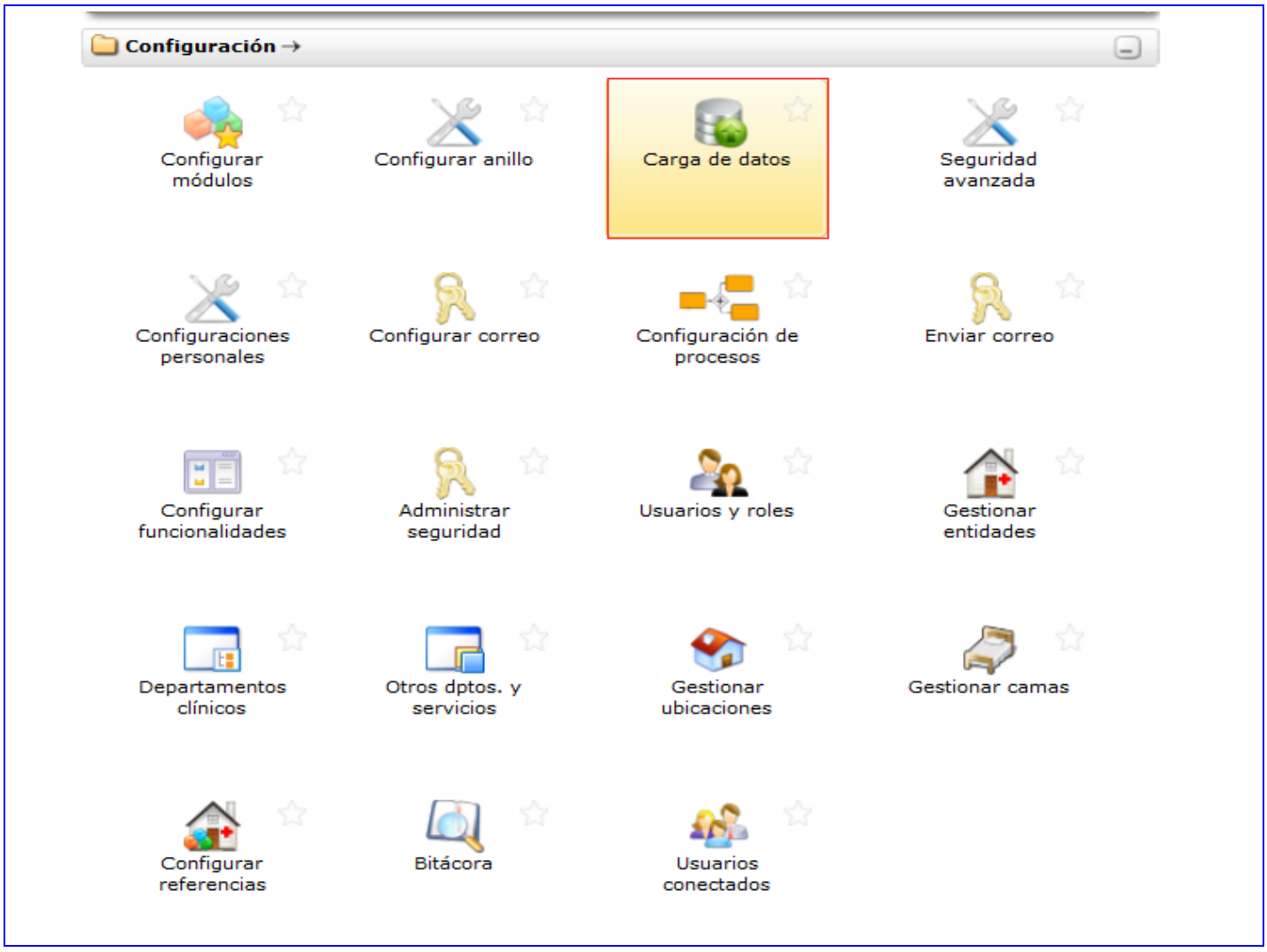


Figura 22. Modificación del módulo configuración

Fuente: Elaboración propia

El componente de carga de datos cuenta con un conjunto de funcionalidades para realizar la búsqueda, adición, modificación, eliminación y ejecución de una o varias transformaciones como se muestra en la Figura 23.

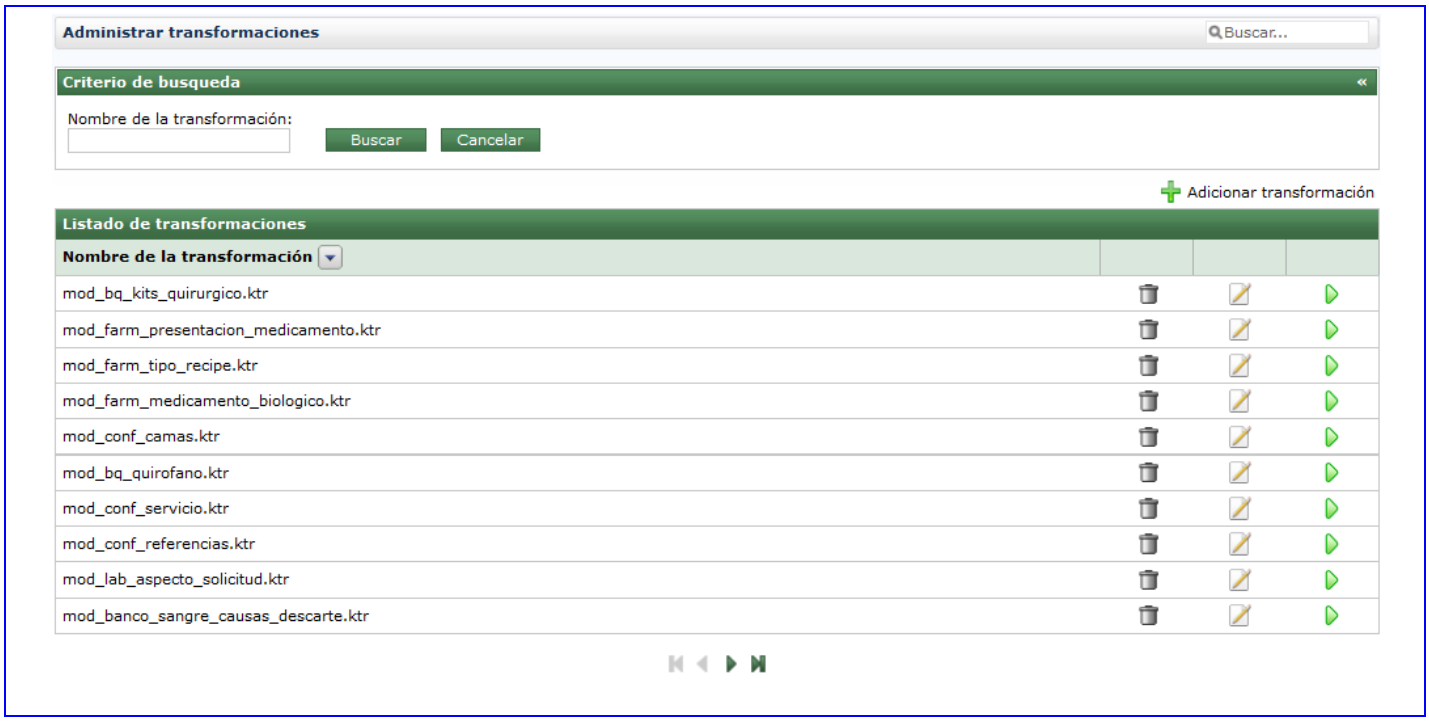


Figura 23. Componente de carga de datos

Fuente: Elaboración propia

4.2.1 Descripción de la funcionalidad adicionar transformación

Al acceder a la funcionalidad Adicionar transformación, se invocan los métodos addListaT(), subirTransf(), adicionarTransf() y adicionarLista() de la clase AdicionarTransformacion encargados de adicionar las transformaciones al sistema. El método addListaT() realiza el llenado de la lista temporal de transformaciones que serán cargadas posteriormente al sistema. Figura 24.

```
public void addListaT() {
    if (ficheroXML.getData() != null) {
        listatemporal.add(new FXML(ficheroXML.getNombre(), ficheroXML
                .getData()));
        ficheroXML = new FXML();
    }
}
```

Figura 24. Método addListaT() de la funcionalidad Adicionar transformación

Fuente: Elaboración propia

Luego el método adicionarLista() es el responsable de agregar al sistema todas las transformaciones existentes en la lista temporal. Este invoca repetidamente a los métodos subirTransf() y adicionarTransf()

mientras la lista temporal no este vacía, garantizando así que las trasformaciones sean copiadas hacia el servidor. Ver Figura 25.

```
public void adicionarLista() {
    while (!listatemporal.isEmpty()) {
        this.ficheroXML = listatemporal.get(0);
        Transformacion transf = subirTransf();
        if (transf != null) {
            try {
                adicionarTransf(transf);
            } catch (Exception e) {
                addLog(e.getMessage());
            }
        }
        cancelListaT(0);
    }
}
```

Figura 25. Método adicionarLista() de la funcionalidad Adicionar transformación

Fuente: Elaboración propia

4.2.2 Descripción de la funcionalidad modificar transformación

Al acceder a la funcionalidad Modificar transformación, se invoca el método modificarTransf() de la clase ModificarTransformacion. Ver Figura 26.

```
public String modificarTransf() {
    try {
        if(data.length == 0){
            facesMessages.addToControlFromResourceBundle("addExcel",Severity.ERROR,"Valor requerido");
            return null;
        }
        subirExcel();
        transfSel.setForigen(forigen);
        transfSel.setCon(con);
        transfManager.save(transfSel);
        addLog("La transformación " + transfSel.getNombre()
                + " ha sido modificada satisfactoriamente.");
        return "ok";
    } catch (Exception e) {
        addLog(e.getMessage());
        return null;
    }
}
```

Figura 26. Método modificarTransf() de la clase ModificarTransformacion

Fuente: Elaboración propia

Este método permite modificar los valores necesarios para lograr una correcta conexión con los ficheros origen mediante el método setForigen(), así como con la base de datos del sistema mediante el método

setCon(), ambos de la clase Transformacion. En este último se tienen en cuenta la dirección, el puerto del servidor, el nombre de la base de datos, el usuario y la contraseña para acceder a la misma. Figura 27.

```java
public void setForigen(FExcel forigen) throws Exception {
    this.forigen = forigen;
    List<StepMeta> lstep = tmeta.getSteps();
    List<Integer> encontrados = new ArrayList<Integer>();
    int pos = buscar(lstep, ExcelInputMeta.class, encontrados);

    do {
        encontrados.add(pos);
        StepMeta stepmeta = lstep.get(pos);
        ExcelInputMeta excel = (ExcelInputMeta) stepmeta
                .getStepMetaInterface();
        String[] dirs = { forigen.getDir() };
        excel.setFileName(dirs);
        excel.setEncoding(getEncoding(dirs[0]));
        stepmeta.setStepMetaInterface(excel);
        tmeta.setStep(pos, stepmeta);
        pos = buscar(lstep, ExcelInputMeta.class, encontrados);
    } while (pos != -1);
    transformacion.setTransMeta(tmeta);
}
```

```java
public void setCon(Conexion con) {
    this.con = con;
    List<DatabaseMeta> ldbs = tmeta.getDatabases();
    for (int i = 0; i < ldbs.size(); i++) {
        DatabaseMeta dbmeta = ldbs.get(i);
        dbmeta.setHostname(con.getNombreServer());
        dbmeta.setDBPort(con.getPuerto() + "");
        dbmeta.setDBName(con.getNombreBD());
        dbmeta.setUsername(con.getUser());
        dbmeta.setPassword(con.getPass());
        ldbs.set(i, dbmeta);
    }
    tmeta.setDatabases(ldbs);
    transformacion.setTransMeta(tmeta);
}
```

Figura 27. Métodos setForigen() y setCon() de la clase Transformacion

Fuente: Elaboración propia

4.2.3 Descripción de la funcionalidad ejecutar transformación

Al ejecutar las trasformaciones cargadas en el sistema se invoca el método ejecutar() de la clase EjecutarTransformacion, en el cual transformacion es un objeto de tipo Trans definido en la librería kettle-engine-5.0.1-stable.jar del *Pentaho*. Este objeto contiene toda la información correspondiente a una transformación diseñada en *Spoon* y mediante el método setLogLevel() establece el nivel de detalle del contenido que será escrito en el registro de eventos al ejecutarse el método writeToLog(). Finalmente el método ejecutar() ofrece el resultado final de la ejecución mediante la variable *salida.* Figura 28.

```
public void ejecutar(){
    try {
        if (esEjecutable()) {
            transfManager.addLog("Ejecutando la transformación '"
                    + seleccionada.getNombre() + "'... ", true);
            transformacion = seleccionada.getTransformacion();
            transformacion.setLogLevel(LogLevel.BASIC);
            transformacion.execute(new String[0]);
            transformacion.waitUntilFinished();
            Result resultado = transformacion.getResult();
            int errors = (int) (resultado.getNrErrors() + errores());
            salida = "La transformación "
                    + seleccionada.getNombre()
                    + " se ejecutó "
                    + (errors == 0 ? "satisfactoriamente." : "con "
                            + errors + " error(es).");

            writeToLog();
            transfManager.addLog(salida, true);
            bitacora.registrarInicioDeAccion("Ejecutando la transformación: "
                    + seleccionada.getNombre());
        } else
            salida = "No se puede ejecutar la transformación '"
                    + seleccionada.getNombre() + "'.";

    } catch (Exception e) {
        transfManager.addLog(e.getMessage(), true);
    }
}
```

Figura 28. Método ejecutar*()* de la clase Transformacion

Fuente: Elaboración propia

Para validar el funcionamiento de la carga de datos realizada por el componente se realizan consultas a la base de datos del sistema antes y después de la ejecución (Figura 30) de una transformación siguiendo el flujo básico de la Figura 29.

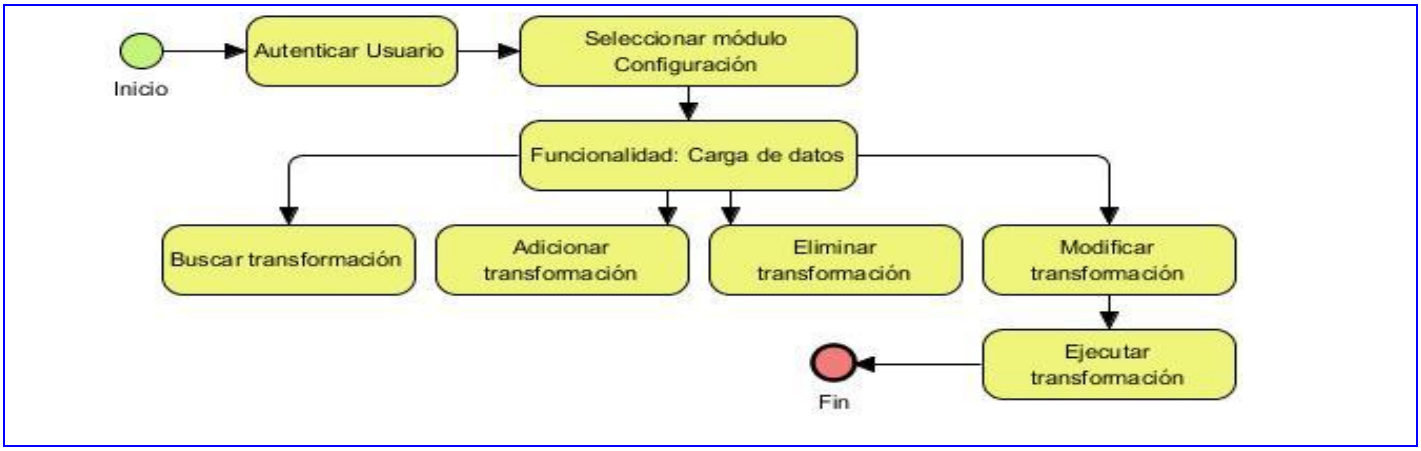


Figura 29. Flujo básico para realizar la carga de datos al sistema

Fuente: Elaboración propia

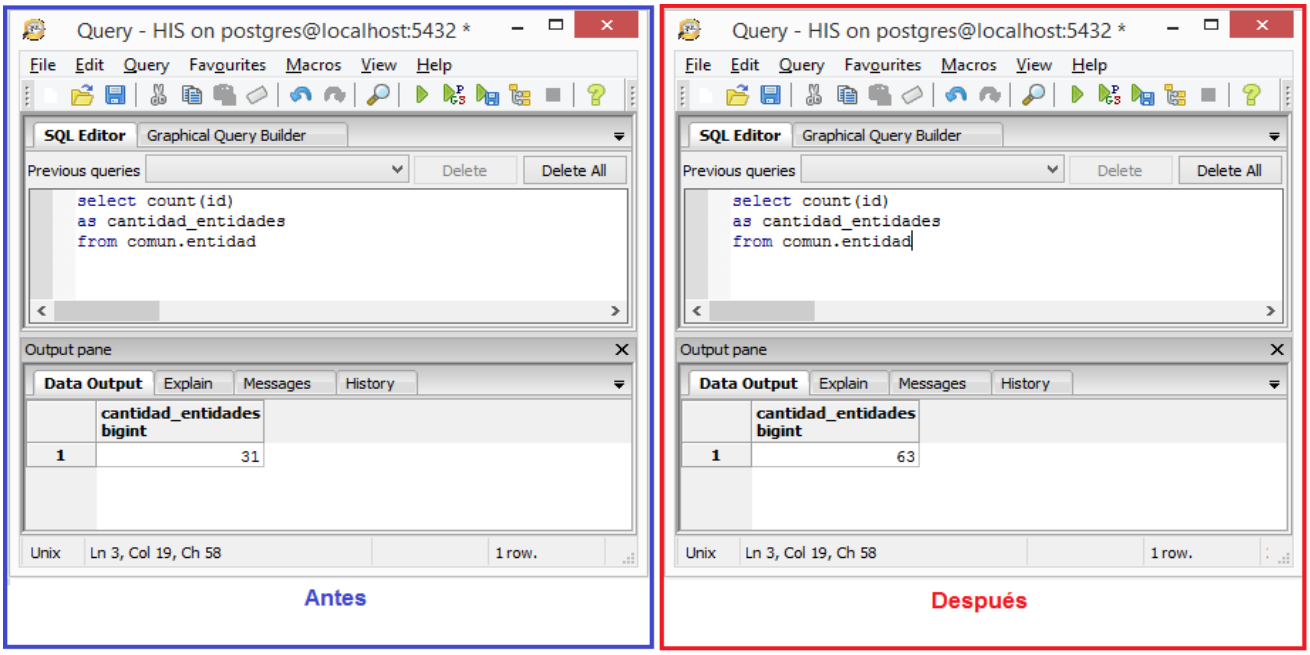


Figura 30. Validación de la transformación mod_conf_datos_entidad.ktr

Fuente: Elaboración propia

4.3 Tratamiento de errores

En la programación siempre se producen errores, más o menos graves, pero que hay que gestionar y tratar correctamente. Debido a esto el tratamiento de errores es uno de los aspectos más importante a tener en cuenta durante el desarrollo de cualquier sistema informático. Durante el desarrollo del componente de carga de datos, se realizó el tratamiento de errores haciendo uso de los bloques *try/catch* (49). A continuación se muestra un fragmento de código del método adicionarTransf() de la clase AdicioarTransformacion en el que se realiza el tratamiento de errores haciendo uso de un bloque *try/catch*. Ver Figura 31.

```
public void adicionarTransf(Transformacion transf) throws Exception {
    try {
        transfManager.agregar(transf);
        addLog("Se ha añadido satisfactoriamente la transformación '"
                + transf.getNombre() + "'.");
    } catch (Exception e) {
        throw e;
    }
}
```

Figura 31. Bloque *try/catch* del método adicioarTransf()

Fuente: Elaboración propia

4.4 Validación de tiempo

Para la validación de la solución al problema identificado en la investigación, se realiza una comparación entre los tiempos promedios de carga de datos conseguidos en dos iteraciones, donde se toma como muestra un conjunto de 10 sujetos. Para una primera iteración, se mide el tiempo consumido para realizar una carga manual de datos en el sistema, seguidamente en una segunda iteración se vuelve a medir el tiempo para una carga de los mismos datos pero con el uso del componente desarrollado (Anexo 2).

Tabla 7. Promedio de tiempos obtenidos tras 2 iteraciones del proceso de carga de datos

Iteraciones	Promedio de tiempo 10 tuplas	Promedio de tiempo 50 tuplas
Carga manual	5.6 minutos	24.2 minutos
Carga con el uso del componente	53.8 segundos	55.4 segundos

De los resultados obtenidos en el análisis anterior se concluye lo siguiente:

- El tiempo estimado para la carga manual de datos aumenta proporcionalmente al número de tuplas a cargar.
- El tiempo estimado para la carga de datos mediante el uso del componente, varía muy poco en comparación con la cantidad de tuplas a cargar.

Conclusiones parciales

Al finalizar las validaciones realizadas en este capítulo, se obtuvieron las siguientes conclusiones:

- La aplicación del componente desarrollado al HIS del CESIM permite la carga de datos al mismo, sin necesidad de utilizar una herramienta externa.
- La utilización de bloques de código try/catch permite el control de errores no esperados en el componente desarrollado.
- La comparación de los tiempos obtenidos al realizar la carga de datos antes y después de la integración del componente, evidencia la notable disminución de tiempo al utilizar dicho componente en el HIS del CESIM.

Conclusiones

Con el desarrollo del componente carga de datos para el Sistema de Información Hospitalaria del CESIM se concluye lo siguiente:

- El estudio realizado acerca de sistemas de gestión de información que poseen un componente de carga de datos, permite identificar ETL como mecanismo para realizar la carga de datos al Sistema de Información Hospitalaria del CESIM.
- El componente de carga de datos para el Sistema de Información Hospitalaria del CESIM, evita el engorroso trabajo manual para la entrada de datos al sistema durante la configuración del mismo en las instituciones hospitalarias.
- La incorporación del componente de carga de datos al Sistema de Información Hospitalaria del CESIM disminuye el tiempo de registro de la información necesaria para su correcto funcionamiento.

Recomendaciones

Luego de haber cumplido los objetivos propuestos se recomienda confeccionar o modificar los ficheros de captura de datos para los módulos restantes del Sistema de Información Hospitalaria del Centro de Informática Médica. Además diseñar las transformaciones necesarias para la carga de datos correspondientes a estos módulos.

Referencias bibliográficas

1. Sousa, Ken y Oz, Effy. *Management information systems.* s.l. : Cengage Learning, 2014.

2. Santos Rodríguez, Carlos y Ana, Aday Martínez Teresa. Revista Cubana de Ingeniería - Cujae. Ciudad de la Habana : s.n., 2012, Vol. III, 1, págs. 7-12.

3. Lara, Florina Gatica y Fernández Puerto, Fernando J. Departamento de Servicios de Cómputo, Universidad Nacional Autónoma de México. Mexico : s.n., 2003.

4. Date, C. J. *Introduccion a los sistemas de bases de datos.* [ed.] José Luis Vázques. [trad.] Antonio Núñez Ramos. Séptima. 2001. págs. 709-712. ISBN: 968-444-419-2.

5. Rios, Angel. Oracle Data Integrator. [En línea] Febrero de 2009. [Citado el: 13 de Febrero de 2015.] http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/317498_esa.pdf.

6. Departamento de Lenguajes y Sistemas Informáticos. [En línea] 2008.1.1, Enero de 2008. http://www.lsi.us.es/docencia/get.php?id=3173.

7. INTELLEGO. ERP – Enterprise Resource Planning. [En línea] 6 de Abril de 2015. [Citado el: 17 de Abril de 2015.] http://www.grupointellego.com/erp/.

8. SAP SE. SAP. [En línea] SAP SE, 21 de Enero de 2010. [Citado el: 2 de Febrero de 2015.] http://www.sap.com.

9. SAP4 PROGRAMMING. Documentación sobre SAP y ABAP. [En línea] 29 de Noviembre de 2013. [Citado el: 16 de Febrero de 2015.] http://wiki.sap4.com/wiki/index.php/ABAP.

10. Sap Help Portal. The Best-Run Businesses Run SAP. [En línea] SAP R/3 Enterprise, Marzo de 2004. [Citado el: 16 de Febrero de 2015.] http://help.sap.com/r3.

11. Help SAP. Help SAP Portal. *The Best-Run Businesses Run SAP.* [En línea] Marzo de 2003. [Citado el: 2 de Febrero de 2015.] http://help.sap.com/saphelp_46c/helpdata/en/a5/3ec8464ac011d1894e0000e829fbbd/content.htm.

12. Ministerio del Poder Popular para Ciencia, Tecnología e Innovación; Fundación Centro Nacional de Desarrollo e Investigación en Tecnologías Libres. SIGESIC. [En línea] 22 de Marzo de 2012. [Citado el: 18 de Febrero de 2015.] https://sigesic.mppi.gob.ve/sistema/media/manuales/instructivoUsuariosCargaMasivaSIGESIC.pdf.

13. Ministerio de Trabajo y Promoción del Empleo. Ministerio de Trabajo y Promoción del Empleo. [En línea] 26 de Agosto de 2014. [Citado el: 2015 de Febrero de 6.] http://www.mintra.gob.pe/.

14. Ministerio de Trabajo y Pormoción del Empleo del Perú; Instituto de Salud y Trabajo. Manual de uso. [En línea] Junio de 2011. [Citado el: 11 de Febrero de 2015.] www.mintra.gob.pe/archivos/file/manualsegurovida.pdf.

15. Ministerio de Hacienda y Crédito Público. Ministerio de Hacienda y Crédito Público. [En línea] 21 de Abril de 2015. [Citado el: 2015 de Febrero de 18.] http://www.minhacienda.gov.co/HomeMinhacienda/siif/ObjetivoContenido.

16. Uriel Rodriguez, Jesús y Mejia Triana, Claribel. Ministerio de Hacienda y Crédito Público. [En línea] 1.0, 13 de Julio de 2009. [Citado el: 23 de Febrero de 2015.] http://www.minhacienda.gov.co/portal/page/portal/C6129F1A503A12DCE040090A20000F18.

17. Carbonell Figueredo, Pedro A. Repositorio Institucional de la UCI . [En línea] Junio de 2013. [Citado el: 24 de Febrero de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8242/1/TD_06750_13.pdf.

18. Jaspersoft Corporation. Jaspersoft ETL. The Open Source Data Integration Platform. [En línea] Jaspersoft Community, 1 de Julio de 2013. [Citado el: 17 de Febrero de 2015.] http://community.jaspersoft.com/project/jaspersoft-etl.

19. Talend Inc. Talend. [En línea] 17 de Diciembre de 2014. [Citado el: 17 de Febrero de 2015.] http://www.talend.com/.

20. CloverETL Worldwide. CloverETL-Rapid Data Integration. [En línea] 16 de Enero de 2015. [Citado el: 17 de Febrero de 2015.] http://www.cloveretl.com/.

21. Pentaho Community. Pentaho. A Hitachi Data System Company. [En línea] 4 de Enero de 2015. [Citado el: 7 de Marzo de 2015.] http://community.pentaho.com/projects/data-integration/.

22. Toolverse. Toolverse ETL Framework. [En línea] 3 de Enero de 2015. [Citado el: 17 de Febrero de 2015.] http://www.toolsverse.com/products/etl-framework/.

23. Carrier, Bernard y Tarum, Gerry. *Software Engineering Methodology. A goal-oriented approach to Software Engineering.* Bloomington : AuthorHouse, 2010. págs. 49-50. 978-1-4520-2223-9(sc).

24. Jacobson, Ivar , Booch, Grady y Rumbaugh, James . *El proceso unificado de desarrollo de software.* [ed.] Pearson Educación. 2000. ISBN 8478290362, 9788478290369.

25. Castro, Luis. [En línea] [Citado el: 6 de Marzo de 2015.] http://aprenderinternet.about.com/od/Glosario/g/Que-Es-Html.htm.

26. *Patrón Modelo-Vista-Controlador.* Fernández Romero, Yenisleidy y Díaz González, Yanette . 1, La Habana : s.n., Enero-Abril de 2012, Telem@tica, Vol. 11, págs. 47-57. ISSN 1729-3804.

27. Pressman, Roger S. Software Engineering. A Practitioner's Approach. Séptima. 2010, págs. 243-245.

28. Rondón Cardozo, Mirlina del Valle. [En línea] 2009. [Citado el: 17 de Febrero de 2015.] http://ri.biblioteca.udo.edu.ve/bitstream/123456789/365/1/TESIS_MR.pdf.

29. Vohra, Deepak. *JavaServer Faces 2.0: Essential Guide for Developers.* s.l. : Cengage Learning, 2014.

30. Ball, Jennifer, y otros. Java Platform, Enterprise Edition. [En línea] 20 de Diciembre de 2012. [Citado el: 17 de Febrero de 2015.] http://www.oracle.com/technetwork/java/javaee/overview/index.html.

31. Rodríguez Sala, Jesús Javier, y otros. *Introducción a la programación. Teoría y práctica.* San Vicente : Editorial Club Universitario, 2003. págs. 4-7. ISBN 84-8454-274-2.

32. Arnold, Ken, Gosling, James y Holmes, David. *El lenguaje de programación Java.* [trad.] Ana Isabel Torres Suárez. Tercera. s.l. : Pearson Educación, 2001. ISBN 9788478290451.

33. Date, Chris J. *Introducción a los sistemas de bases de datos.* s.l. : Pearson Educación, 2001.

34. Martinez, Rafael. PosgreSQL-es. [En línea] 2 de Octubre de 2010. [Citado el: 17 de Febrero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

35. Guhr, Jerry T. y Picone, Joseph . *"Integrated development environment for high speed transaction processing www applications on heterogeneous computer systems.* U.S. : s.n., 2003.

36. Red Hat, Inc. [En línea] 2014. [Citado el: 18 de Febrero de 2015.] https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Developer_Studio/8.0/pdf/8.0.0_Release_Notes/Red_Hat_JBoss_Developer_Studio-8.0-8.0.0_Release_Notes-en-US.pdf.

37. *Un estudio comparativo de herramientas para el modelado con UML.* Quintero, Juan Bernardo, y otros. 137, 2005, Universidad EAFIT, Vol. 41, págs. 60-76.

38. Company Headquarters. Visual Paradigm. [En línea] [Citado el: 18 de Febrero de 2015.] http://www.visual-paradigm.com/features/uml-and-sysml-modeling/.

39. Sánchez, Jorge. *Servidores de Aplicaciones Web.* s.l. : Autoedición, 2011.

40. Free Software Foundation. El Sistema Operativo GNU. [En línea] 5 de Junio de 2015. [Citado el: 15 de Junio de 2015.] URL: http://www.gnu.org/licenses/licenses.es.html.

41. JBoss Community. JBoss Aplication Server. [En línea] [Citado el: 18 de Febrero de 2015.] http://jbossas.jboss.org/history.

42. Carina Roldán, Maria. Pentaho Data Integration (Kettle) Tutorial. [En línea] Pentaho Community. [Citado el: 3 de Febrero de 2015.] http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+%28Kettle%29+Tutorial.

43. Sommerville, Ian . *Ingenieria de software.* [ed.] Miguel Martín-Romo. [trad.] María Isabel Alfonso Galipienso. Séptima. Madrid : Pearso Educatión, 2005. págs. 105-150. ISBN 10:84-7829-274-5/ISBN 13: 978-84-7829-074-1.

44. Fowler, Martin y Scott, Kendall. *UML gota a gota.* [ed.] Pablo Eduardo Roig Vázquez. Primera. s.l. : Addison Wesley Logman, 1999. págs. 49-60. ISBN 968-444-364-1.

45. Zea Restrepo, Claudia Maria. *Hacia una comunidad educativa interactiva.* s.l. : Universidad Eafit, 2007. págs. 97-102. ISBN 978-958-8281-63-6.

46. Pressman, Roger S. *Software Engineering. A Practitioner's Approach.* s.l. : Mc Graw Hill, Higher Education, 2010. págs. 215-227. ISBN 978-0-07-337597-7.

47. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado.* [trad.] Begoña Moros Valle. Segunda. s.l. : Pearson Educación, 2003. págs. 184-215. ISBN 9788420534381.

48. Campderrich Falgueras, Benet. *Diagrama de Componentes.* Primera. Barcelona : Editorial UOC, 2002. págs. 102-105. ISBN 84-8429-997-6.

49. Osmosis Latina. *Sintaxis del bloque try/catch.* [En línea] 2010-2011. http://javabasico.osmosislatina.com/curso/progavanza2/trycatch.htm.

50. Vassiliadis, Panos y Simitsis, Alkis. *EXTRACTION, TRANSFORMATION, AND LOADING.*

51. [En línea] www.mintra.gob.pe/archivos/file/GUIAUSUARIO_CARGAMASIVA.pdf.

52. Verónica Saladrigas, María. El sistema de clasificación ATC de sustancias farmacéuticas para uso humano. [En línea] 2004 de Marzo. [Citado el: 13 de Febrero de 2015.] www.medtrad.org/panacea/IndiceGeneral/n15_tribuna-Saladrigas.pdf.

53. Storkel, Scott. O'Reilly. Java.com. [En línea] O'Reilly Media, Inc, 12 de Noviembre de 2002. [Citado el: 18 de Febrero de 2015.] http://onjava.com/pub/a/onjava/2002/12/11/eclipse.html.

54. Rolando, Rizzardi. Introducción SAP R/3. [En línea] Junio de 2003. http://www.manualespdf.es/manual-sap-r3-enterprise/.

55. Reidl, Lucy y Peresmitré, Gilda Gómez. *METODOLOGÍA DE INVESTIGACIÓN EN CIENCIAS SOCIALES.* s.l. : U.N.A.M., 2013.

56. Ramos, Jorge Rivera. [En línea] [Citado el: 3 de Febrero de 2015.] http://correo.ryrhermanos.com/home/jrivera@ryrhermanos.com/Publico/PDI_Parte_1.pdf.

57. Nepo Linares, Edgardo. La CIE 10 en la Gestión de Servicios de Salud. [En línea] Junio de 2008. [Citado el: 13 de Febrero de 2015.] http://www.sunasa.gob.pe/educacion/foros/files_aya/11%20CIE%2010%20-%20Enepo.pdf.

58. Mora, David . Maromas Digitales. [En línea] 28 de Mayo de 2010. [Citado el: 3 de Marzo de 2015.] http://maromasdigitales.net/2010/05/patron-mvvm-explicado/.

59. Calderón Corail, Marcela. cognus. [En línea] 3 de Septiembre de 2008. [Citado el: 6 de Marzo de 2015.] http://www.cognus.biz/pentaho-open-bi/.

60. Alegsa, Leandro. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. *Definición de Excel.* [En línea] [Citado el: 14 de Febrero de 2015.] http://www.alegsa.com.ar/Dic/excel.php.

61. *Mercado de datos para una dirección de salud en Cuba.* Acosta Méndez, Geidy y Jorge Chacón, Disnayle. La Habana : s.n., 2013. ISBN: 978-959-7213-02-4.

62. JESEALM. [En línea] [Citado el: 18 de Febrero de 2015.] http://es.scribd.com/doc/92191689/SIGESIC#scribd.

63. Pentaho Javadoc. [En línea] http://community.pentaho.com/javadoc/.

64. Guia Ubuntu. [En línea] 10 de Marzo de 2008. [Citado el: Febrero de 19 de 2015.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

65. Guia de usuario. [En línea] www.mintra.gob.pe/archivos/file/GUIAUSUARIO_CARGAMASIVA.pdf.

66. Exist. Exist. [En línea] 16 de Marzo de 2015. [Citado el: 10 de junio de 2014.] http://exsis.com.co/conoce-las-arquitecturas-de-software-mas-comunes/.

67. Informática-Hoy. [En línea] 2012. [Citado el: 16 de Febrero de 2015.] http://www.informatica-hoy.com.ar/sap/Las-ventajas-de-SAP-R3.php.

Bibliografía

- ApS, Airflake. [En línea] [Citado el: 18 de Febrero de 2015.] http://www.osalt.com/jboss.
- Ball, Jennifer, y otros. Java Platform, Enterprise Edition. [En línea] 20 de Diciembre de 2012. [Citado el: 17 de Febrero de 2015.] http://www.oracle.com/technetwork/java/javaee/overview/index.html.
- Barzanallana Asensio, Rafael Menéndez . [En línea] 13 de Octubre de 2011. [Citado el: 24 de Febrero de 2015.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.
- Calderón Corail, Marcela. cognus. [En línea] 3 de Septiembre de 2008. [Citado el: 6 de Marzo de 2015.] http://www.cognus.biz/pentaho-open-bi/.
- Carbonell Figueredo, Pedro A. [En línea] Junio de 2013. [Citado el: 24 de Febrero de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8242/1/TD_06750_13.pdf.
- Castro, Luis. [En línea] [Citado el: 6 de Marzo de 2015.] http://aprenderinternet.about.com/od/Glosario/g/Que-Es-Html.htm.
- CloverETL-Rapid Data Integration. [En línea] [Citado el: 17 de Febrero de 2015.] http://www.cloveretl.com/.
- Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla. [En línea] 2008.1.1, Enero de 2008. http://www.lsi.us.es/docencia/get.php?id=3173.
- Help SAP. Help SAP Portal. [En línea] 2003. [Citado el: 2 de Febrero de 2015.] http://help.sap.com/saphelp_46c/helpdata/en/a5/3ec8464ac011d1894e0000e829fbbd/content.htm.
- Historia y definición de los lenguajes de programación. [En línea] 16 de Diciembre de 2013. [Citado el: 20 de Febrero de 2015.] http://www.monografias.com/trabajos99/historia-y-definicion-lenguajes-programacion/historia-y-definicion-lenguajes-programacion.shtml.
- Informática-Hoy. [En línea] 2012. [Citado el: 16 de Febrero de 2015.] http://www.informatica-hoy.com.ar/sap/Las-ventajas-de-SAP-R3.php.
- INTELLEGO. [En línea] 2014. [Citado el: 2 de Febrero de 2015.] http://www.intellego.com.mx/es/erp-planificacion-de-recursos-empresariales.
- Jaspersoft Corporation. [En línea] [Citado el: 17 de Febrero de 2015.] http://community.jaspersoft.com/project/jaspersoft-etl.
- Java. [En línea] [Citado el: 20 de Febrero de 2015.] http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html.

Anexos

Anexo 1: Diseño de la transformación Servicios por departamento

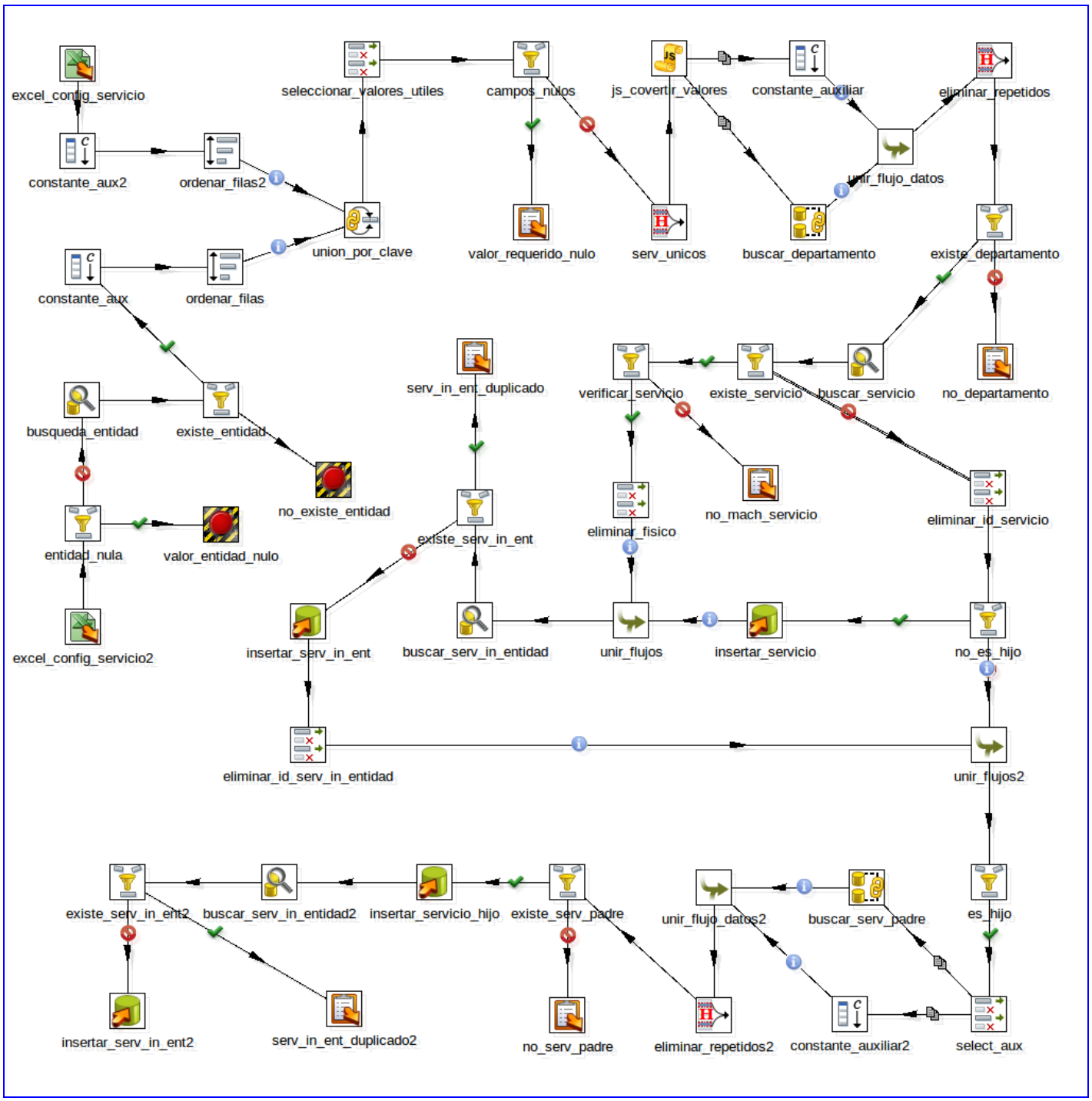


Figura A 1. Diseño de la transformación Servicios por departamento

Fuente: Elaboración propia

Anexo 2: Comparación del consumo de tiempo en el proceso de carga de datos en el Sistema de Información Hospitalaria del CESIM con el uso del componente y de forma manual.

A continuación se muestra la Tabla 8 con la información referente a los tiempos consumidos por cada una de las personas al realizar el proceso de carga de datos:

Tabla A 1. Relación de tiempos obtenidos tras las 2 iteraciones del proceso de carga de datos para 10 tuplas

Persona	Carga manual	Carga con el uso del componente
1	5.2 min	52.4 s
2	6.2 min	55.2 s
3	4.9 min	53.0 s
4	5.0 min	56.1 s
5	5.8 min	51.7 s
6	4.7 min	50.3 s
7	7.3 min	53.1 s
8	6.5 min	58.2 s
9	5.8 min	56.8 s
10	5.2 min	52.0 s
Promedio	*5.6 min*	*53.8 s*

Tabla A 2. Relación de tiempos obtenidos tras las 2 iteraciones del proceso de carga de datos para 50 tuplas

Persona	Carga manual	Carga con el uso del componente
1	25.1 min	53.2 s
2	29.0 min	54.1 s
3	21.9 min	54.7 s
4	24.2 min	57.3 s
5	23.5 min	54.6 s
6	19.4 min	51.3 s

7	29.6 min	59.0 s
8	27.5 min	1.2 min
9	22.4 min	57.7 s
10	21.2 min	53.1 s
Promedio	*24.2 min*	*55.4 s*

Teniendo en cuenta que el tiempo estimado para realizar el proceso de carga de datos de forma manual aumenta proporcionalmente al número de tuplas de datos a cargar, para un conjunto de 1000 filas una persona necesitaría aproximadamente 93 horas, lo que equivale a 12 días si se toma como horario laborable 8 horas diarias. En cambio haciendo uso del componente esta misma persona cargaría los datos en solo 3.1 minutos aproximadamente.

En el gráfico de la Figura A 2 se describe el comportamiento del tiempo (en minutos) necesario para cargar la información referente a las secciones de laboratorio en una entidad de forma manual y con el uso del componente. El eje de las abscisas representa las cantidades de tuplas a cargar, mientras que el eje de las ordenadas simboliza el tiempo estimado.

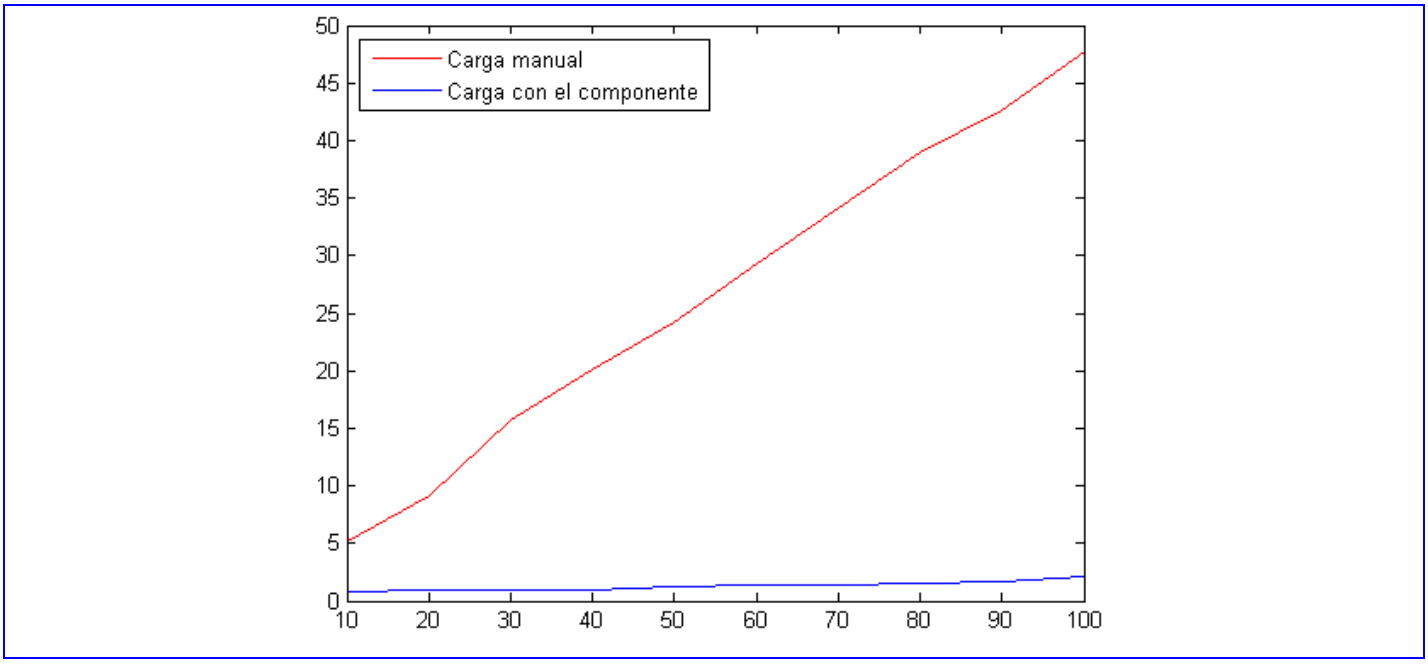


Figura A 2. Tiempos estimados para la carga de datos de las secciones de laboratorio en una entidad

Fuente: Elaboración propia

Glosario de términos

Aplicación: En informática, una aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos.

Arquitectura: Se define como un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción de *software* para un sistema informático. La arquitectura de *software* establece los fundamentos para que los analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema informático.

Código: Cifras, clave. En informática se utiliza para referirse a un conjunto de instrucciones en un lenguaje de programación.

Componente: Parte modular de un sistema de *software*. Se refiere a la agrupación de varias funcionalidades y estructuras que toman sentido en un contexto determinado.

Framework: Estructura predefinida para la creación de aplicaciones. Puede estar formado por un conjunto de librerías y clases o por una arquitectura que facilita el desarrollo de *software*.

Herramientas: Son programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo.

Patrones: Los patrones son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. Definen una estructura común debido al aprendizaje pasado.

Plug-ins: Programas que deben instalarse en los ordenadores clientes para que otros puedan reconocer y procesar determinados tipos de archivo y realizar determinadas funcionalidades.

Proceso: Un proceso es una secuencia de pasos con algún tipo de lógica, que se enfocan en lograr algún resultado específico.

***Spoon*:** *Spoon* es el diseñador gráfico de transformaciones y trabajos asociados con el sistema de ETL *Pentaho Data Integration*, también conocido como *Kettle*.

Tupla: Es un término usado en programación y se interpreta como un conjunto de elementos que se guardan de forma consecutiva en memoria. Una tupla es un objeto que bien puede tener diversos datos u objetos. En el lenguaje de bases de datos, una tupla se define como una función finita que asocia unívocamente los nombres con sus valores.