Universidad de las Ciencias Informáticas

Facultad 2

Desarrollo de la versión 2.0 del importador de datos del producto auditoría CEDRUX

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Juan José Rosales Rodríguez

Tutor: Ing. Yusnier Matos Arias

Co-tutor: Ing. Eliecer Cabrera Casas

La Habana, Cuba, 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Juan José Rosales Rodríguez

_____

Ing. Yusnier Matos Arias

_____

DATOS DE CONTACTO

Ing. Yusnier Matos Arias. (ymarias@uci.cu). Instructor. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Se desempeña como jefe de la línea de desarrollo de productos horizontales en el Programa ERP-Cuba.

AGRADECIMIENTOS

*…A nuestro **Comandante en Jefe Fidel Castro**, por ayudarme a realizar mi sueño.*

*…A la **Universidad de las Ciencias Informáticas** (UCI) por formarme como un futuro profesional.*

*…A **Yanel** por siempre estar a mi lado.*

*…A mis tutores **Matos**, **Eliecer**, **Cristian**, **Eduardo** por guiarme, ayudarme y apoyarme en cada situación que se me presentó.*

*…A la profe **Laritza**, por su ayuda.*

…Al equipo de trabajo de la Línea de productos horizontales por el apoyo y solidaridad mostrado

*…A la profesora **Yunet** por su ayuda en los requisitos. .*

*…A **Omarito** por su ayuda en la solución, mil gracias.*

*…A **Manuel** por ser mi compañero de lucha.*

*…A **Yisel** y su equipo por exigir lo mejor de mí.*

JJ

DEDICATORIA

*…A mi abuela **Anita** por ser más que abuela, mi madre.*

…A mi papá y mamá.

…A mi novia.

…A mis amigos por confiar en mí en todo momento.

JJ

RESUMEN

Como parte de los productos del Centro de Informatización de la Gestión de Entidades (CEIGE) se desarrolla el módulo de Auditoría y control. Este, tiene como objetivo principal auditar los datos pertenecientes a los diferentes sistemas instalados en las entidades cubanas. Estos datos pueden estar almacenados en diferentes gestores de bases de datos, por lo que se desarrolló una herramienta para importar los mismos hacia un entorno controlado para ser analizados. La versión 1.0 de dicha herramienta es capaz de importar datos alojados solamente en servidores PostgreSQL, siendo necesario en la actualizad trabajar con otros orígenes de datos.

En este trabajo de diploma se describe el análisis, diseño e implementación de la versión 2.0 del importador del módulo de Auditoría. En esta versión, la herramienta es capaz de importar datos desde diferentes fuentes de datos como: PostgresSQL, MySQL, MsSQL y ficheros CSV. Su diseño está orientado a facilitar el mantenimiento y la escalabilidad. Para la implementación de este diseño se utilizaron herramientas y tecnologías de la plataforma php que en su totalidad son libres y de código abierto. La nueva solución permite futuras extensiones para la importación de otros orígenes de datos no soportados.

PALABRAS CLAVE: auditoría, importación, control, datos, PostgresSQL, MySQL, MsSQL, CSV

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde los inicios del desarrollo de la humanidad el hombre de todas las generaciones ha tenido la necesidad de almacenar información con el propósito de que se transmita de generación en generación. En la actualidad, con el desarrollo de las tecnologías de la informática y las comunicaciones aparecen nuevas formas más eficientes para procesar y almacenar la información, las cuales han sido necesarias y a su vez imprescindibles para el desarrollo de cualquier país. Cuba nunca ha estado ajena a este proceso, por tal razón la información almacenadas en las entidades cubanas deben estar sujeta a frecuentes y rigurosos controles con el fin de garantizar una mayor fiabilidad de la misma. Una forma de garantizar esta fiabilidad en las entidades es a través de la auditoría y el control sistemático, actividades que permiten examinar y verificar información, registros y procesos, con el objetivo de expresar una opinión sobre su fiabilidad e integridad.

En cualquier empresa la auditoría es la actividad motora que define una situación real del comportamiento económico y financiero de la misma, dando plena seguridad de que los datos registrados son verdaderos y confiables. Evalúa el grado de eficiencia y eficacia con que se desarrollan todas las tareas administrativas y el cumplimiento de los planes u orientaciones dentro de un período.

La auditoría sin el empleo de las técnicas computacionales es un proceso que puede proporcionar un análisis con un alto grado de errores, producto a que la información consultada generalmente está almacenada en grandes volúmenes. El desarrollo de las tecnologías ha permitido que el almacenamiento sea más eficiente y seguro en cuanto a pérdidas de la información, pero está expuesta a otros factores de riesgos, como son el fácil acceso, las vulnerabilidades, etc. Con la automatización se proporcionará un mejor trabajo de los procesos auditables, aumenta la eficiencia, por ser más rápidos, seguros y exactos. El empleo de estas nuevas técnicas, de conjunto con la utilización de un software, facilita que se puedan detectar infracciones, delitos y fraudes cometidos en el manejo de la información.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla un software para realizar auditorías como parte del programa ERP-Cuba, este software es capaz de funcionar independientemente del mismo. Este producto proporciona un conjunto de funcionalidades necesarias para agilizar el proceso de auditoría a información contenida en las bases de datos de las entidades cubanas o extranjeras.

INTRODUCCIÓN

La importación[1] de datos de fuentes externas al ambiente propio del software de auditoría es una funcionalidad esencial. La herramienta de importación utilizada actualmente en el ERP-Cuba para importar la información a ser auditada es capaz de importar datos desde un ambiente externo, pero está limitada a importar datos únicamente del gestor de base de datos PostgreSQL

La importación únicamente desde el servidor PostgreSQL limita la usabilidad y el campo de aplicación que podría tener el software. Hoy en día existen muchas fuentes de datos tanto en gestores de bases de datos relacionales como otras basadas en diferentes tipos de ficheros, por lo tanto la aplicación debe ser capaz de importar varias fuentes de datos presentes en las entidades. Otro elemento a tener en cuenta de la diversidad de las fuentes de información, es el hecho de que tampoco se almacena la información de forma estándar. Cada entidad posee aspectos únicos, por lo que se necesita una herramienta capaz de reconocer la estructura en que se almacena la información.

Teniendo en cuenta la problemática descrita, se planteó la necesidad de poder trabajar con fuentes de datos provenientes de gestores diferentes de PostgreSQL, incluidos los basados en ficheros.

Partiendo de lo anterior, se formula el siguiente problema a resolver: ¿Cómo lograr la compatibilidad del módulo de Auditoría con otras fuentes de datos para que pueda ser usado en todas las entidades del país?

Partiendo del problema planteado, el objeto de estudio se enfocará hacia el proceso de importación de datos entre Sistemas Gestores de Base de Datos (SGBD) y ficheros.

El objetivo general que se persigue es: desarrollar la herramienta importadora de datos, con soporte para otros Gestores de Bases de datos, tales como: Pgsql, MySQL, MsSQL y ficheros: CSV.

Del objetivo general se desglosan los siguientes objetivos específicos:

- Establecer el marco teórico relativo a la importación de datos entre las fuentes señaladas en el objetivo de la investigación para establecer un punto de partida para la solución del problema planteado.

[1] Es el proceso que consiste en ingresar la información en un sistema desde una fuente de datos externa que la contiene.

- Realizar el análisis y diseño de la herramienta de Importación de datos 2.0 para establecer un punto de partida hacia la implementación de la solución.
- Desarrollar las funcionalidades de importación desde cada uno de las fuentes especificadas para dar cumplimiento al problema formulado en la investigación.
- Realizar pruebas al importador para validar los resultados de la investigación.

El campo de acción lo constituye el proceso de importación desde los diferentes gestores de base de datos Pgsql, MySQL, MssQL y ficheros: CSV, hacia el gestor de la aplicación PostgreSQL.

Para el cumplimiento de los objetivos específicos propuestos se definieron las siguientes tareas:

- Evaluación y análisis de la bibliografía consultada con el objetivo de fundamentar la investigación.
- Evaluación y análisis de las funcionalidades presentes actualmente en el subsistema de importación con el fin de identificar posibles reutilizaciones.
- Documentación de los nuevos requisitos identificados a incluir en el sistema.
- Diseño de la herramienta de importación.
- Documentación del diseño presentado para la implementación de las nuevas funcionalidades.
- Modelación del diseño de clases.
- Configuración de los diferentes escenarios para probar las soluciones.
- Implementación de las nuevas funcionalidades especificadas.
- Realización de las pruebas de caja negra y métricas de validación del diseño para validar la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1. INTRODUCCIÓN

Dentro de los objetivos del presente capítulo se encuentra sentar las bases teóricas para la investigación. Los conceptos relacionados con la investigación ya sea de manera directa o indirecta se enunciarán o se citarán, estos, serán en un principio el hilo conductor de la investigación.

Dentro del capítulo se realizará un estudio del estado del arte que recoge diferentes herramientas que poseen funcionalidades similares o que realizan importación de datos, con el objetivo de obtener información actualizada y tomar referencias para el futuro desarrollo, dando una valoración de las mismas. A lo largo del presente trabajo se utilizarán varias herramientas y tecnologías, por lo que se realizará una descripción de las mismas con el objetivo de conocerlas y brindar una panorámica de la solución. Para el desarrollo del software se utilizará un modelo de desarrollo propio, definido en el Centro de Informatización de la Gestión de Entidades (CEIGE), de acuerdo a las necesidades de los proyectos del mismo, este modelo es importante conocerlo por lo que se abordará en este capítulo.

1.1. Conceptos básicos

Software de Auditoría

Los *"software de auditoría"* consisten en programas de computadora usados por el auditor, como parte de sus procedimientos de auditoría, para procesar datos de importancia de auditoría principalmente del sistema de contabilidad de la entidad. Puede consistir en programas de paquete, programas escritos para un propósito, programas de utilería o programas de administración del sistema. Independientemente de la fuente de los programas, el auditor deberá verificar su validez para fines de auditoría antes de su uso. (1)

Características de un software de auditoría: (2)

- Independiente de la plataforma de origen de datos en que se encuentra la aplicación a analizar.
- Puede analizar en conjunto datos de múltiples aplicaciones o base de datos sin limitaciones.
- Tiene funcionalidades específicas de Auditoría y posibilidades de realizar pruebas muy extensas no disponibles en otras herramientas.
- Permite lectura de reportes.
- Plataforma independiente y ajena a posibles vicios o errores del sistema principal.
- Independencia del área de Informática.

- Gran potencial de análisis y sin límite de ficheros o transacciones.

Los software de auditorías presentan funcionalidades propias de auditoría y análisis de la información las cuales varían de acuerdo al fabricante, la mayoría de los mismos poseen una herramienta que tiene como objetivo importar la información hacia el ambiente del software para su posterior análisis, esta es una de las funcionalidades vitales e imprescindibles.

Para un software de contabilidad es de vital importancia almacenar la información, por lo que todos usan mecanismos de almacenamiento de datos. Algunos utilizan sistemas gestores de base de datos y otros sistemas de ficheros.

Ficheros

Se le llama fichero a un conjunto de información clasificada y almacenada de diversas formas para su conservación y fácil acceso en cualquier momento. En informática, un archivo o fichero también es un conjunto de información que se almacena en forma virtual para ser leído y/o accedido por medio de una computadora. Las posibilidades de almacenamiento y clasificación son mucho más ricas en un sistema informático, ya que la información no ocupa un espacio físico y, por ende, es posible conservar millones de datos en un dispositivo muy pequeño. Inclusive, se puede guardar información de texto, audio o video en un mismo lugar sin inconveniente alguno.

Tipos de ficheros a estudiar

Ficheros CSV: "Los ficheros CSV (del inglés comma-separated values) son un tipo de documento sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles. Se puede usar cualquier editor de textos para generar este tipo de documentos, incluso también otros programas de hojas de cálculo (excel, open office[2]) Además, como norma se usa como delimitador el carácter coma, aunque se puede usar otro como separador. (2)

Sistema Gestores de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de

[2] Paquete ofimático libre que se utiliza para el manejo de hojas de cálculo, documentos, presentaciones, etc.

datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc. (3)

Principales gestores de base de datos:

- MsSQL: Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. (4)
- MySQL: Es un sistema de gestión de base de datos relacional, multi-hilo y multiusuario. MySQL AB. Desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009 desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. (5)

1.2. Soluciones existentes

Existen varios software que poseen herramientas que realizan funciones similares, por lo que el estado del arte se enfocará al estudio de las mismas.

En el mundo existen dos software líderes en el tema de auditoría: IDEA y ACL los cuales son potentes herramientas de auditoría, las cuales gozan de preferencia internacionalmente por el gran número de funciones que realizan. (2)

ACL

ACL es una de las herramientas de software de auditoría preferidas por la comunidad de auditoría internacional, para la extracción y el análisis de datos, la detección de fraudes y el control continuo. Proporciona una exclusiva y eficiente combinación de acceso a los datos, análisis y elaboración integrada de reportes, ACL permite transformar los datos en información significativa y asistirlo en el logro de objetivos comerciales para agregar valor a su organización. (6)

Cuenta con una herramienta importadora de datos encargada de llevar los datos a un ambiente propio del software, la cual tiene las siguientes particularidades:

ACL puede importar los datos accediendo según su tipo, de los siguientes ficheros:

- AS/400
- Microsoft Access, Excel, Word
- SAP
- DB2
- HTML
- COBOL

IDEA

IDEA es un software de PC muy fácil de usar, que permite que el analista de datos o auditor de negocios acceda virtualmente a cualquier archivo de datos de cualquier entorno y analice 100% de miles o millones de transacciones en segundos detectando la totalidad de las excepciones y construyendo las propias bases de datos de Análisis de datos o Auditoría con datos completamente flexibles y de entornos diversos. (7)

Este software fue la base de desarrollo del software de auditoría existente en nuestros días en ERP-Cuba, fue objeto de estudio de cada una de sus funcionalidades para tomar las más importantes y así desarrollar una primera versión de la aplicación.

El mismo también presenta una herramienta importadora de datos encargada de llevar los datos a un ambiente propio del software la cual tiene las siguientes particularidades:

El proceso de importación se realiza de cuatro formas diferentes:

- Cuando es un fichero conocido estándar de PC(AS 400, ASCII, dBASE, Microsoft Access, EBCDIC, Microsoft Excel y otros).

  Cuando es un fichero conocido la aplicación es capaz de reconocer el contenido fácilmente, la herramienta mapea el fichero mostrando en una pantalla su información y los diferentes delimitadores de los campos, tales como punto y coma, coma, dos puntos, tabs, espacio, etc. En la pantalla también se muestra para elegir el tipo de fichero, esta opción sirve por si no se muestra el formato legible, un ejemplo de esto es cuando cargamos un Excel, no se muestra la información

legible, se muestran caracteres extraños, no entendibles por una persona o dice que la información no puede ser visualizada, por lo que hay que especificarle, qué tipo de fichero es.

- Mediante el ODBC[3] para los diferentes gestores de base de datos.

  Mediante ODBC la herramienta es capaz de importar los datos registrados en los orígenes de datos del sistema operativo, de esta forma se puede acceder a servidores relaciones como SQLServer, entre otros.

- Cuando es un fichero desconocido por la herramienta.

  La herramienta es capaz de mostrar el contenido del fichero, de esta forma se podrá definir los futuros campos que tendrá la tabla después de importado el mismo, esta funcionalidad se realiza de una manera simple manejando unas líneas delimitando cada campo.

- Cuando es un archivo de un informe impreso.

  Al igual que el desconocido se muestra el informe en una pantalla brindando la posibilidad de edición de los diferentes campos que se quieran tener para su posterior análisis.

IDEA importa desde un amplio rango de tipos de ficheros, puede importar tantos archivos *.mdb como *.accdb, además de:

- AS 400.
- ASCII o EBCDIC(Texto)
- Dbase
- Microsoft Access
- Microsoft Excel
- Adobe PDF
- SAP

Herramienta de importación 1.0

[3] Es un estándar de acceso a bases de datos que utilizan los sistemas Microsoft. Las siglas significan Open DataBase Connectivity. A través de ODBC, en un sistema Windows se puede conectar con cualquier base de datos.

La herramienta importadora de datos en su versión 1.0 es capaz de importar los datos desde el SGBD PostgreSQL hacia el ambiente propio del Software de Auditoría.

Para la realización del proceso de importación presentan tres interfaces de usuario[4]:

- En la primera interfaz se configura la conexión con el servidor origen, dentro de los parámetros a configurar están la dirección IP, puerto, usuario y contraseña. Después de llenar los campos se oprime el botón *"conectar"* para probar la conexión, si la misma fue realizada con éxito se cargará un Combobox [5] mostrando las bases de datos existentes en el servidor.
- En la segunda interfaz es capaz de mostrar todas las tablas contenidas en la base de datos, brindando la posibilidad de seleccionarlas e importarlas, además presenta otra funcionalidad que permite seleccionar automáticamente las tablas que tengan dependencias con la tabla seleccionada.
- En la tercera interfaz brinda un reporte detallado del proceso de importación, mostrando la cantidad de tablas que tenía el servidor, cuántas se importaron correctamente y cuántas presentaron problemas.

En esta versión el proceso de importación se puede ver afectado en algunos casos:

- El usuario puede conocer o no el puerto por defecto del servidor, de ser el caso de no conocerlo, esto sería un problema a la hora de realizar la importación.
- Si el origen de datos a importar posee una estructura organizativa como es el caso del gestor PostgreSQL, la herramienta no brinda la funcionalidad de mostrar la información referente a sus esquemas en forma de árbol, por tanto se hace un poco engorroso conocer a que esquema pertenece una tabla determinada.
- Otra dificultad que presenta esta versión es que permite importar tablas vacías ya que no muestra la cantidad de registros que tienen las mismas.
- Esta versión tampoco permite que el usuario personalice el nombre de las tablas a importar, por lo que el componente [6] le pone el nombre original por defecto.

[4] La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina.

[5] Componente visual que permite seleccionar un elemento en una lista desplegable.

[6] Estructura organizativa para organizar la implementación.

1.3. Análisis de las soluciones existentes

Existen herramientas de auditorías con gran calidad y eficiencia en la realización de los procesos de importación de datos como el ACL y el IDEA, de dichas herramientas se puede apreciar el uso de una capa abstracta encargada de importar los datos (ODBC), por tanto se deduce la necesidad de contar con una capa de abstracción en la herramienta de importación que permita importar datos de múltiples orígenes de datos. A su vez existe la versión 1.0 de la actual herramienta de importación, la cual no cuenta con un correcto diseño arquitectónico para lograr extenderla a otras fuentes de datos permitiendo solamente la importación de bases de datos PostgreSQL.

Por todo lo antes expuesto se decide desarrollar una herramienta de importación 2.0 con una capa de abstracción que soporte importar datos de múltiples orígenes de datos y a su vez extensible usando las tecnologías y herramientas definidas por el subsistema de tecnología del CEIGE.

1.4. Modelo de desarrollo

El Modelo de desarrollo fue elaborado por el equipo de producción del proyecto ERP-Cuba, de acuerdo a las necesidades presentadas por las líneas de desarrollo, teniendo en cuenta principales riesgos del proyecto. Este modelo también es conocido como modelo de desarrollo del CEIGE. Está basado en metodologías RUP y XP, dos de las metodologías más usadas en la construcción de software, tomando lo más notable y útil de cada una de acuerdo a las necesidades existentes en el proyecto ERP-Cuba. (8)

Características del Modelo de Desarrollo:

- Centrado en la arquitectura:

  La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

- Orientado a componentes:

  Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

- Iterativo e incremental:

  Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

- Ágil y adaptable al cambio:

  El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales [7] están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.5. Tecnologías y herramientas propuestas para el desarrollo del sistema

Para la realización de la solución se utilizaron las tecnologías y herramientas orientadas por la línea de tecnología del proyecto las cuales forman parte de las directrices arquitectónicas, a continuación se detalla más sobre ellas.

Lenguaje de modelado

Hoy día UML (Unified Modeling Languaje) está consolidado como el lenguaje estándar en el análisis y diseño de sistemas informáticos. Mediante el mismo es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representan la arquitectura del proyecto. (9)

UML: Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. (10)

Herramienta de Modelado

[7] Profesionales especializados en el negocio.

Visual Paradigm Suite for UML 6.4: Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones, mejores y a un menor coste. Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

PHP 5.2.6

PHP (acrónimo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. (11)

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (12)

Zend Framework 1.7

Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios Web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. (13)

Doctrine 1.2

El framework Doctrine es un sistema para el mapeo de objeto-relacional (ORM: Object Relational Mapper, mapeador relacional de objetos) para PHP 5.2 ó superior, este cuenta con una capa de abstracción escrita sobre PDO, de las que se destacan el diccionario de datos para los diferentes gestores de bases de datos. (14)

ExtJS 2.2

ExtJS Framework: Es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Incluye un alto rendimiento, interfaces de usuario personalizables, bien diseñado y extensible modelo de componentes, una interfaz intuitiva y fácil de utilizar con licencias de código abierto y comercial. (15)

Brinda soporte para:

- Construir interfaces gráficas complejas y dinámicas.

- Comunicar datos de forma asíncrona con el servidor.
- Diversos navegadores como: Internet Explorer, Firefox, Safari y Opera.

PostgreSQL 8.3

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre. Tiene una arquitectura probada por lo que ha ganado una sólida reputación para la fiabilidad, integridad y exactitud de los datos. Funciona en todos los principales sistemas operativos. Incluye los mejores tipos de datos. También soporta el almacenamiento de grandes objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de programación nativo de C/C++, Java, Net, Perl, Python, Ruby, Tcl,etc. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OS X, Solaris, BSD, y otros más.

PostgreSQL soporta ACID (es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.), o lo que es lo mismo, la realización de transacciones seguras; también, vistas, uniones, claves extranjeras, procedimientos almacenados, triggers[8], etcétera. (17)

Incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99, como: entero, numérico, booleano, char, varchar, fecha, interval o timestamp. (17)

Otras características interesantes de PostgreSQL son las siguientes: (17)

- Alta concurrencia, que evita tener que bloquear una tabla cuando se está escribiendo en ella.
- Copias de seguridad en línea.
- Replicación asíncrona.
- Transacciones anidadas.
- Optimizador de consultas.

NetBeans IDE 6.9

NetBeans IDE es un entorno de desarrollo, una herramienta práctica para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero es adaptable para cualquier otro lenguaje de programación, es un producto libre y gratuito sin restricciones de uso.

[8]Un ***trigger*** es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. (17)

1.6. Arquitectura Cliente/Servidor

La arquitectura Cliente/Servidor tiene como objetivo optimizar el uso tanto del hardware como del software, a través de la separación de funciones: el cliente, quien inicia una determinada petición y el servidor, dedicado a responder dichas peticiones, en la Figura 1 se representa dicha arquitectura.

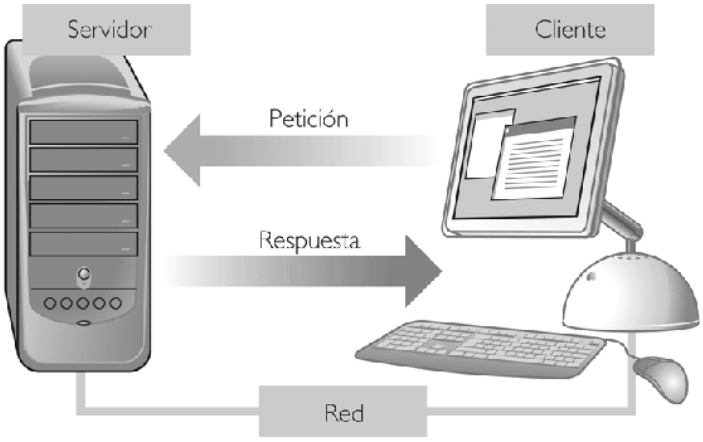


Figura 1. Arquitectura Cliente/Servidor

Puede presentarse como uno o varios clientes y servidores, junto con un sistema operativo y una plataforma de comunicación para formar un sistema cooperativo que permita la computación distribuida, el análisis y la presentación de datos. Un único servidor típicamente sirve a una multitud de clientes, ahorrando a cada uno de ellos el problema de tener la información almacenada localmente.

Características de la arquitectura Cliente/Servidor: (19)

- El servidor presenta una interfaz única y bien definida a todos sus clientes.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

- 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 no afectan al cliente.

1.7. Patrones

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son basadas en la experiencia y que se ha demostrado que funcionan. (18)

En el mundo actual, cada vez más diseñadores de software usan patrones para realizar de forma más eficiente soluciones de arquitectura, pues su uso permite el desarrollo de software altamente flexible, disminuyendo el tiempo invertido en la generación de código y fomentando en gran medida la reutilización del mismo, lo que aumenta la depuración del software. Existen infinidad de patrones encaminados a resolver disimiles situaciones, en este trabajo se propone el uso de varios de ellos, los cuales serán explicados a continuación para facilitar su entendimiento.

Patrones de diseño:

- Modelo-Vista-Controlador (MVC): Es un patrón de diseño que simplifica el desarrollo y mantenimiento de aplicaciones. Esto se logra mediante la separación de la aplicación en tres componentes lógicos:
  - **Modelo:** La capa del modelo es responsable de la lógica de negocio de una aplicación. Se va a encapsular el acceso a los almacenes de datos y proporcionará una biblioteca de clases reutilizables.
  - **Vista:** La capa de la vista es generalmente lo que se considera el diseño web o plantillas. Controla la apariencia de los datos y proporcionan funcionalidades para recoger datos del usuario. Tecnologías que se encuentran exclusivamente en la vista son HTML, CSS y JavaScript.
  - **Controlador:** La capa del controlador es la encargada de unir la capa de las vistas con la del modelo, Es el responsable de recibir la información recibida por la vista y manejar el flujo de ejecución de la aplicación, es la encargada de llamar a las funcionalidades del modelo y retornar estos datos a una vista. (21)

- **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. (18)
- **Factory:** Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al cliente la casuística para elegir el subtipo que crear. Este patrón aumenta la flexibilidad en tiempo de ejecución, a veces es imposible elegir de antemano cuál objeto específico debe ser instanciado, ya que la elección de los objetos a utilizar puede depender de algo en el entorno de ejecución (18)

  Patrones de asignación de responsabilidades:

  - Patrones GRASP (General Responsibility Asignment Software Patterns, Patrones Generales de Software de Asignación de Responsabilidades) (19)
  - **Experto:** Consiste en asignar una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para cumplir con la responsabilidad.(19)
  - **Controlador:** Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa.(19)
  - **Bajo Acoplamiento:** Consiste en crear pocas dependencias entre las clases con el objetivo que un cambio en alguna de ellas no afecte al resto o dicha afectación tenga una mínima repercusión en el resto de las clases potenciando la reutilización y disminuyendo la dependencia.(19)
  - **Alta Cohesión:** Nos dice que la información que almacena una clase debe de ser coherente y estar en la mayor medida de lo posible relacionada con la clase de modo que cada elemento de nuestro diseño debe realizar una labor única dentro del sistema. Una clase con mucha cohesión es útil porque es fácil de interpretar y de darle mantenimiento de esta forma se proporciona un aumento en la capacidad de reutilización. Por lo antes planteado recomendamos la aplicación de este patrón desde los inicios del desarrollo. (19)

1.8. Diseño de clases

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de

conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP en este caso). (23)

1.9. Métricas del diseño

A continuación se detallan los atributos que se miden en la validación de la solución:

- Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad del diseño: Consiste en la complejidad que posee una estructura de diseño de clases.
- Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización: Consiste en el grado de reutilización que presente una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para realizar una corrección, una mejora o una rectificación de algún error del diseño del software. Puede influir de manera indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.
- Eficiencia: Se refiere al esfuerzo que un usuario tiene que hacer para conseguir un objetivo.
- Efectividad: Variables que permiten medir la exactitud y la plenitud con la que se alcanzan los objetivos de una tarea concreta.

Las métricas aplicadas al diseño para la evaluación de la solución propuesta son las siguientes:

- Tamaño operacional de la clase (TOC): Está dado por el número de métodos asignados una clase.

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 1. Tamaño operacional de la clase (TOC)

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otras.

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Tabla 2. Relaciones entre clases (RC)

CONCLUSIONES

En este capítulo se realizó un análisis crítico de varias aplicaciones importadoras de datos, de las cuales se tomaron algunas ideas para aplicarlas en el desarrollo de la versión 2.0 del importador de datos del subsistema de Auditoría de Cedrux. Se expusieron las tecnologías y herramientas a ser usadas para el desarrollo del importador. Con este capítulo se da cumplimiento al primer objetivo específico planteado: Establecer el marco teórico relativo a la importación de datos entre las fuentes señaladas en el objetivo de la investigación para establecer un punto de partida para la solución del problema planteado, y se sientan las bases para el desarrollo de la solución.

CAPÍTULO 2. ANÁLISIS Y DISEÑO.

2. INTRODUCCIÓN.

En el presente capítulo se describe el flujo actual de los procesos relacionados con el proceso de importación de datos hacia el gestor PostgreSQL. Se modela el negocio identificado, detallando cada uno de los procesos involucrados para dar solución al problema planteado. Se realiza la captura de requisitos atendiendo a los criterios de los especialistas. Se describen los requisitos no funcionales del sistema. En este capítulo se realiza además el diseño del sistema y la validación de dicho diseño, También se abordan elementos como el uso de patrones en la solución y el diseño del modelo de datos.

2.1. Propuesta de la versión 2.0 del importador de datos

En este trabajo, se desarrolla la versión 2.0 del importador de datos del módulo de Auditoría, se pretende mejorar el diseño de la versión anterior, con el fin de reducir los problemas de usabilidad que presenta la misma. En la nueva versión el usuario podrá importar datos de diferentes fuentes. La misma brindará además, la posibilidad de mostrar la estructura correspondiente a cada una de estas fuentes de datos, por ejemplo mostrará un árbol de esquemas en el caso de PostgreSQL, el usuario podrá determinar qué tabla desea importar y con qué nombre. Si la tabla ya existe en la base de datos del software de auditoría la podrá remplazar o simplemente crear una copia nueva, esta versión visualizará la cantidad de registros que posee cada tabla origen, también se incorpora la posibilidad de importar datos desde ficheros CSV.

Para implementar la herramienta se hará uso de un diseño de clases basado en patrones, más acorde a la nueva solución, y se incorporará una capa extensible para que en caso de incorporar una nueva fuente de datos no sea necesario implementar nuevamente la solución.

2.2. Mapa de procesos de negocio auditoría

El mapa de procesos del negocio del módulo auditoría tiene como propósito brindar una breve descripción de cada uno de los procesos para la mejor comprensión de los mismos, el cual cuenta con 6 subprocesos:

- Importar contenedores de información.
- Analizador de información.
- Analizador de vistas.
- Analizador de tablas de las BD.

- Realizar editor de ecuaciones.
- Generar gráficos.

A continuación en la Figura 2 se presenta el mapa de procesos del software de auditoría:

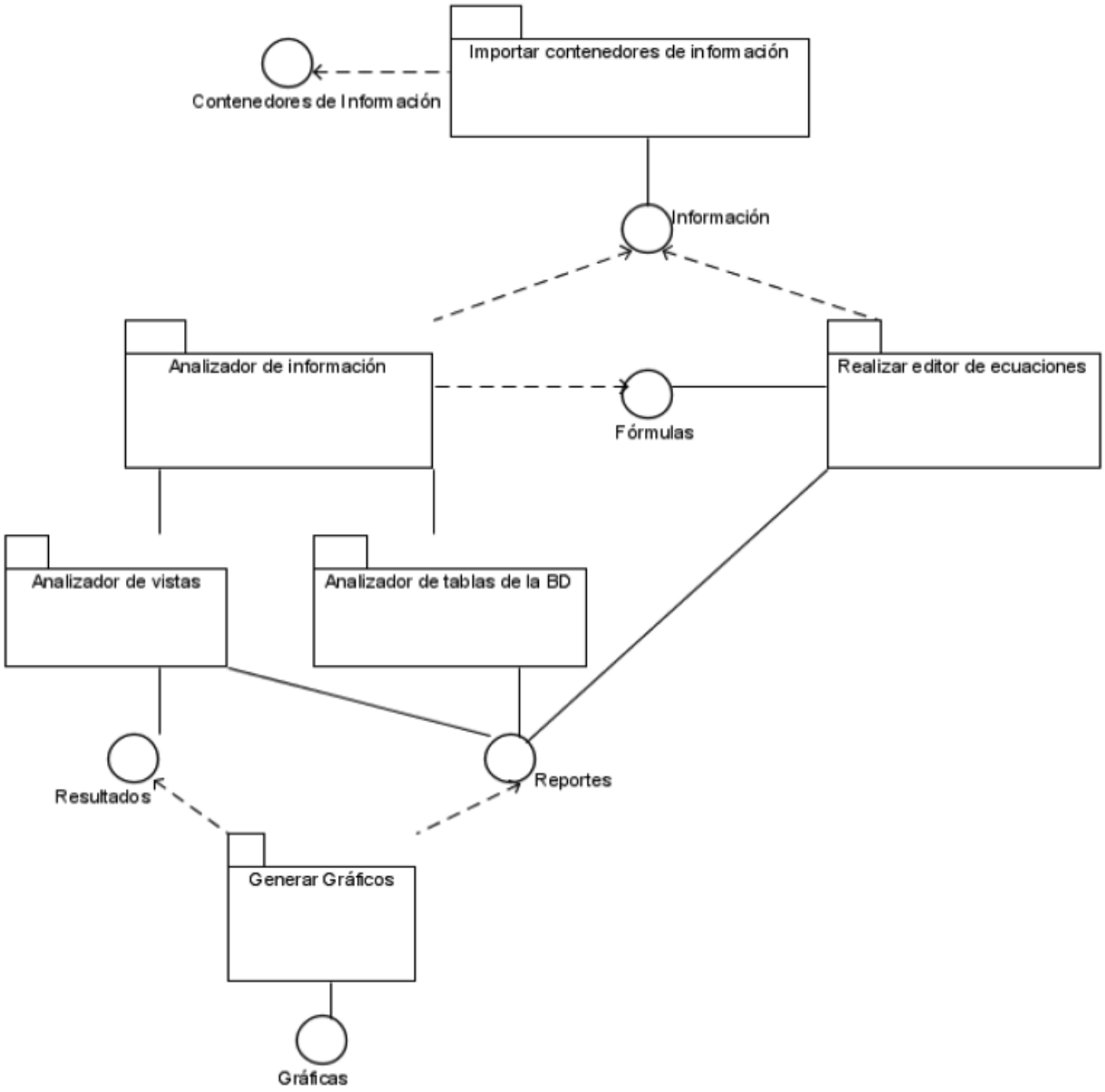


Figura 2. Mapa de procesos de negocio de la línea de auditoría

CAPÍTULO 2 ANÁLISIS Y DISEÑO

El análisis, diseño e implementación del proceso Importar contenedores de información será el objetivo fundamental de la investigación. El mismo queda representado en el mapa de proceso de software de auditoría con el fin de tener conocimientos exactos respecto a la interacción que este tiene con el resto de los procesos.

2.3. Descripción del proceso de negocio: Importar contenedores de información.

Objetivos	Mostrar la estructura de la fuente de datos. Importar los datos. Generar reporte.
Evento(s) que lo generan	Se debe haber realizado la configuración para acceder a la fuente de datos.
Precondiciones	Se debe acceder a la fuente de datos.
Post-condiciones	Se obtiene la información importada en la base de datos de la herramienta.
Marco jurídico	No procede.
Clientes internos	Analizador de información.
Clientes externos	No aplicable.
Entradas	Información.
Salidas	Tablas en la aplicación.

Tabla 3. Descripción del proceso de negocio importar contenedores de información

Mediante este proceso se permite acceder a las diferentes fuentes de datos a auditar. El auditor seleccionará el servidor y el gestor de base de datos, luego se autenticará en dicho servidor escogerá los datos que desea auditar y los importará hacia el gestor propio de la aplicación para realizar el análisis de la misma, en esta versión de la solución se importarán datos desde varios gestores como Mysql, PostgreSQL, MsSQL Server, de la misma forma el auditor podrá subir el fichero CSV para realizar la importación de sus datos.

Diagrama del proceso:

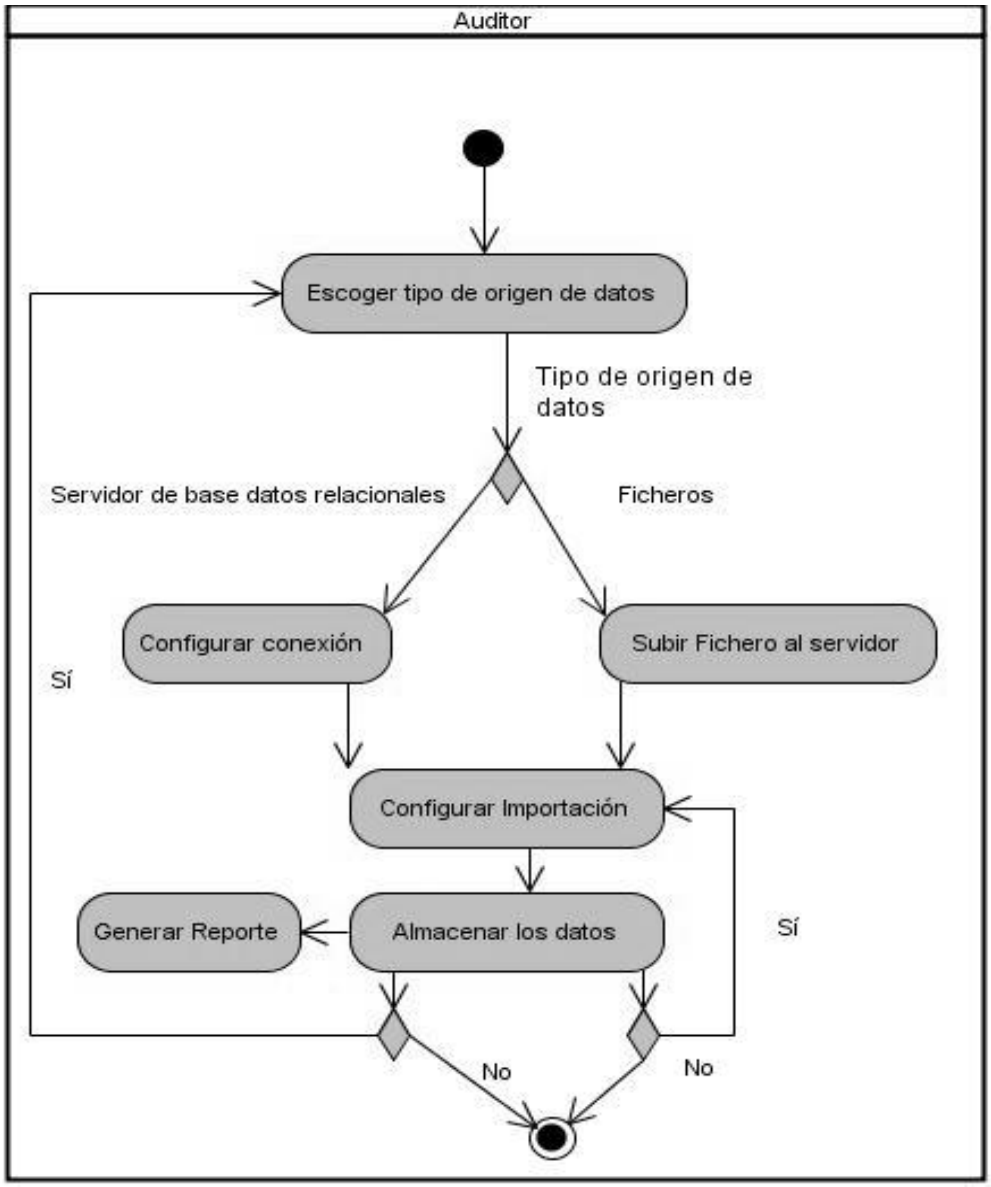


Figura 3. Diagrama del proceso Importar contenedores de información

Descripción del flujo básico.

Escoger el origen de los datos	El auditor selecciona la importación para realizar el posterior análisis de la misma.

Configurar la importación:	El auditor configura la importación en dependencia del tipo de origen de datos.
Almacenar los datos:	La información importada es almacena en la base de datos del servidor de la herramienta de auditoría y así de esta forma se genera el reporte.

Tabla 4. Descripción del flujo básico del proceso importar

Descripción del flujo paralelo.

Configurar conexión:	El auditor se conecta al servidor y escoge la base de datos a importar.
Ficheros:	El auditor escoge el tipo de fichero a importar.
Subir fichero al servidor:	El auditor sube el fichero a importar al servidor.

Tabla 5. Descripción de los flujos paralelos del proceso importar

El proceso Importar contenedores de información es el encargado de realizar la importación tanto como de servidores relacionales como con ficheros, este proceso genera tablas con la información importada hacia el software de auditoría.

2.4. Requisitos

Las técnicas utilizadas para el Levantamiento de Requisitos de la nueva versión de la herramienta de importación fueron las entrevistas realizadas a los especialistas funcionales de la Contraloría General de la República, también se realizaron talleres, ámbito que fue propicio para desencadenar lluvias de ideas. De las mismas surgieron los requisitos que se muestran y detallan a continuación.

Requisitos funcionales del Software

La herramienta de importación cuenta con 2 requisitos funcionales:

- Importar bases de datos relacionales. ***Ver Anexo 3.***
- Importar datos de ficheros CVS. ***Ver Anexo 4***

Requisitos no funcionales

Los requisitos no funcionales de la aplicación han sido agrupados por categorías y se muestran a continuación.

Software:

Servidores	Especificaciones
Servidor Aplicaciones Web	• Sistema Operativo: Ubuntu Server • Servidor Web: Apache 2.0 • Librerías Adicionales: PHP 5.2.6
Servidor de Base de Datos	• Sistema Operativo: Ubuntu Server • Sistema Gestor de Base de Datos: PostgreSQL 8.3.8
Servidor de Clientes Ligeros	• Sistema Operativo: Nova Server • Navegador Web: Mozilla Firefox 2.2 ó superior. • Herramientas Ofimáticas • Visualizador de ficheros pdf. • Herramienta de administración de clientes ligeros.

Tabla 6. Requisitos no funcionales de software en los servidores

Clientes	Especificaciones
PC Cliente	• Sistema Operativo: Linux o Windows • Navegador Web: Mozilla Firefox v2.2 ó superior. • Herramientas Ofimáticas • Visualizador de ficheros pdf.

Cliente Ligero	• Todo se instala en el servidor de clientes ligeros.

Tabla 7. Requisitos no funcionales de software en las estaciones clientes

Hardware:

Servidores	Especificaciones
Servidor Aplicaciones Web	• Procesador: 3.00 GHZ • RAM: 1GB • Disco duro: 160 GB • UPS: 1 • Lector de CD: 1 • Tarjeta de Red: 1
Servidor de Base de Datos	• Procesador:3.00 GHZ • RAM: 1GB • Disco duro: 160 GB • UPS: 1 • Lector de CD: 1 • Tarjeta de Red: 1
Servidor de Clientes Ligeros	• Procesador: 3.00 GHZ • RAM: 1GB • Disco duro: 160 GB • UPS: 1 • Lector de CD: 1 • Tarjeta de Red: 1

Tabla 8. Requisitos no funcionales de hardware en los servidores

Clientes	Especificaciones

PC Cliente	• Procesador: 1.40 GHZ • RAM: 256 MB (recomendado 512 Mb) • Tarjeta de Red: 1
Cliente Ligero	• Procesador: 2.0 GHZ • RAM: 256 MB (recomendado 512 Mb) • Tarjeta de Red: 1

Tabla 9. Requisitos no funcionales de hardware en las estaciones clientes

Apariencia o interfaz de usuario

El sistema debe poseer una interfaz amigable al usuario, basada en web, brindando facilidades que permitan interactuar y navegar con el sistema, de forma fácil y rápida. El sistema debe cumplir con los estándares de diseño establecidos en el proyecto ERP Cuba según se especifica en ERP-ARQ Estándar para el diseño de interfaces v1.1. (UCID, 2008).

Seguridad

El servidor de bases de datos y el servidor de aplicación estarán instalados en locaciones diferentes. Cada usuario realizará operaciones en la aplicación en dependencia de sus privilegios o niveles de acceso.

Usabilidad

El sistema debe poder satisfacer las peticiones del cliente de la forma más sencilla y fácil posible sin complicación ni complejidad. Podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Soporte

Se debe brindar mantenimiento a la aplicación en los entornos de despliegue.

2.5. Diseño de la solución

A continuación se mostrara el diseño de clases de la solución.

Diagramas de Clases del Diseño Importar bases de datos relacionales

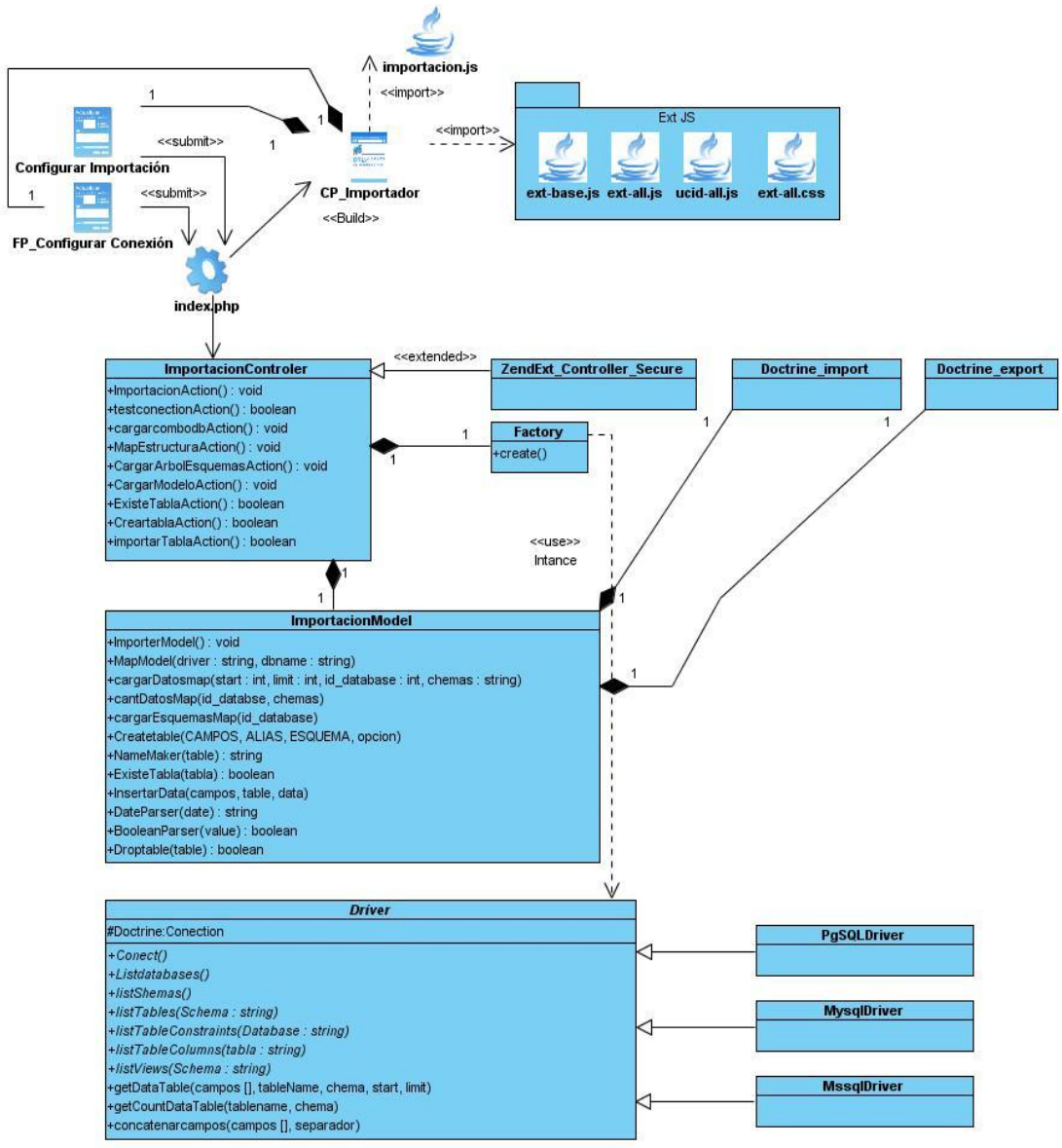


Figura 4. Diagrama de clases del proceso Importar bases de datos relacionales

Descripción de las Clases

Tabla 10. Descripción de Clases: ImportacionController

Nombre: ImportacionController	
Tipo de clase: (controladora)	
Para cada responsabilidad:	
Nombre:	**function** init()
Descripción:	Constructor de la clase ImportacionController.
Nombre:	**function** ImportacionAction()
Descripción:	Carga la vista correspondiente al importador controlador.
Nombre:	**function** testconectionAction()
Descripción:	Realiza una prueba de conexión con el origen de datos.
Nombre:	**function** cargarcombodbAction()
Descripción:	Carga las bases de datos del origen de datos.
Nombre:	**function** MapEstructuraAction()
Descripción:	Persiste la estructura del origen de datos.
Nombre:	**function** CargarArbolEsquemasAction()
Descripción	Carga el árbol con la información referente a la estructura del origen de datos.
Nombre:	**function** CargarModeloAction()
Descripción	Carga las tablas que tienen la estructura seleccionada y la cantidad de registros que tiene cada tabla.
Nombre:	**function** ExisteTablaAction()
Descripción	Devuelve si existe la tabla o no.
Nombre:	**function** CreartablaAction()

Descripción	Se encarga de crear la tabla a importar.
Nombre:	**function** importarTablaAction()
Descripción	Se encarga de invocar la lectura de los datos en origen de datos para insertarlo.
Nombre:	**function** RegistrarTablaEnelSistemaAction
Descripción	Se encarga de registrar en la aplicación las tablas que fueron importadas.
Nombre:	**function** cargarReporteAction()
Descripción:	Carga el reporte de la importación.

Tabla 11. Descripción de Clases: ImportacionModel

Nombre: ImportacionModel	
Tipo de clase: (Modelo)	
Atributo	Tipo
$cnx	obj. PDO conection
$chema_dest	String con el esquema destino.
Para cada responsabilidad:	
Nombre:	**function** MapModel($driver,$dbname)
Descripción:	Función que mapea la base de datos para persistir su estructura en las tablas del importador.
Nombre:	**function** cargarDatosmap($START,$LIMIT,$ID_DATABASE,$Chemas)
Descripción:	Función que carga los datos para mostrarlo en el grid con los nombres de las tablas y la cantidad de registros que tiene la misma.
Nombre:	**Function** cantDatosMap ($ID_DATABASE,$Chemas)

Descripción:	Esta función devuelve la cantidad de tablas que tiene la base de datos con id entrado por parámetro en la tabla mod_importador_config.dat_tables
Nombre:	**function** cargarEsquemasMap ($ID_DATABASE)
Descripción:	Carga los esquemas que tenga esa base de datos en el caso que los tenga.
Nombre:	**function** Createtable($CAMPOS,$ALIAS,$ESQUEMA,$opcion)
Descripción:	Esta función se encarga de crear o remplazar la tabla a importar en la base de datos del software de auditoría.
Nombre:	**function** NameMaker($table)
Descripción	Determina el nombre de la tabla en caso de que exista una tabla con el mismo nombre.
Nombre:	**function** ExisteTabla($table)
Descripción	Determina si la tabla ya existe una tabla con ese mismo nombre en la base de datos.
Nombre:	**function** InsertarData($campos,$table,$data)
Descripción	Inserta la información obtenida del origen de datos a la tabla creada.
Nombre:	**function** DateParser($bad_date_string,$format='d-m-Y')
Descripción	Se encarga de migrar las fechas al formato estándar del software de auditoría.
Nombre:	**function** BooleanParser($bad_bool)
Descripción	Se encarga de migrar el formato de la cadena entrada al formato correcto que maneja el software de auditoría.
Nombre:	**function** Droptable($table_Dest)
Descripción	Función que se encarga de eliminar la tabla cuyo nombre es entrado por parámetro.
Nombre:	**function** ClearMapById($iddb)
Descripción:	Esta función elimina los datos temporales creados durante el proceso de

	importación
Nombre:	**función** CantidadDeTablasSinImportadas($iddb)
Descripción:	Devuelve la cantidad de datos que no se importaron del origen de datos.
Nombre:	**function** RegistrarTablaInmportada($iddb,$alias,$dbw,$entidad,$Schema_name)
Descripción:	Registra la tabla importada en los catálogos del software de auditoría.

Tabla 12. Descripción de Clases: Driver

Nombre: Driver	
Tipo de clase: (Modelo)	
Atributo	Tipo
$DoctCon	Conexión de doctrine.
Para cada responsabilidad:	
Nombre:	**function** Connect($dbu,$dbpas,$dbhos,$dbna,$mot,$port,$sid)
Descripción:	Crea una conexión con PDO para usarla en la conexión de doctrine.
Nombre:	**abstract function** listDatabases()
Descripción:	Devuelve la lista de bases de datos que tiene el origen de datos.
Nombre:	**abstract function** listShemas()
Descripción:	Devuelve los esquemas que tiene una base de datos.
Nombre:	**abstract function** getDataTable($campos,$tableName,$chema,$start,$limit)
Descripción:	Devuelve los datos de la tabla de los campos pasados por parámetros.
Nombre:	**function** concatenarcampos($campos,$separator)
Descripción:	Devuelve una cadena de caracteres con el nombre de los campos separados por el separador entrado por parámetro.

Nombre:	**function** getCountDataTable($tableName,$chema)
Descripción	Devuelve la cantidad de registros que tiene la tabla.
Nombre:	**abstract function** listTableColumns($tableName,$chema=null)
Descripción	Devuelve las columnas de una tabla entrada por parámetro de un esquema pasado por parámetro en el caso de que el gestor sea postgres.
Nombre:	**abstract function** listTables($dbName,$chema);
Descripción	Devuelve un listado con las tablas de una base de datos entrada por parámetros y de un esquema dado.

Tabla 13. Descripción de Clases: factory

Nombre: factory	
Tipo de clase: (Modelo)	
Para cada responsabilidad:	
Nombre:	**static function** create($driver)
Descripción:	Retorna una instancia del driver pasado por parámetro.

Tabla 14. Descripción de Clases: PgSQLDriver

Nombre: PgSQLDriver	
Tipo de clase: (Modelo)	
Descripción	Implementación de la clase Driver para PostgreSQL

Tabla 15. Descripción de Clases: MysqlDriver

Nombre: MysqlDriver	
Tipo de clase: (Modelo)	
Descripción	Implementación de la clase Driver para MySQL

Figura 14 Descripción de Clases: MssqlDriver

Nombre: MssqlDriver	
Tipo de clase: (Modelo)	
Descripción	Implementación de la clase Driver para Microsoft SQL Server

Diagrama de clases del diseño importar ficheros

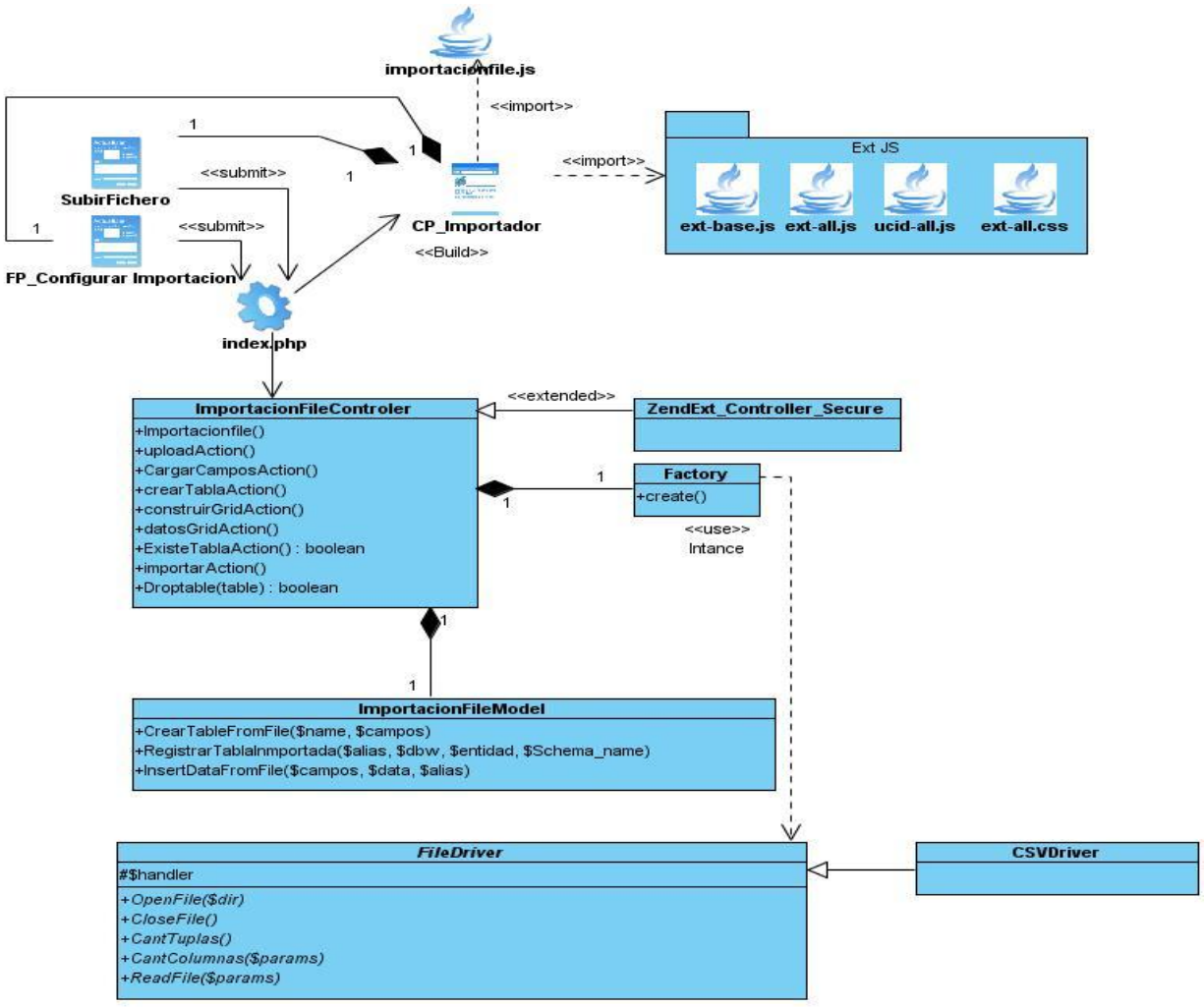


Figura 5. Diagrama de clases del flujo importar ficheros

Descripción de las clases

Tabla 16. Descripción de Clases: importacionFileController

Nombre: importacionFileController	
Tipo de clase: (Controlador)	
Para cada responsabilidad:	
Nombre:	**function** importacionfileAction()
Descripción:	Carga la vista correspondiente para importar ficheros.
Nombre:	**function** UploadAction()
Descripción:	Recibe el fichero enviado desde el formulario para procesamiento, se encarga de salvar el fichero en el servidor.
Nombre:	**function** CargarCamposAction()
Descripción:	Muestra la lista de campos (Columnas) devueltos por el driver especifico que tiene el fichero a importar.
Nombre:	**function** CrearTablaAction()
Descripción:	Recibe un arreglo de campos enviados desde la vista y ordena al importerfilemodel a crear la tabla con dicha configuración de campos.
Nombre:	**function** ConstruirGridAction()
Descripción:	Devuelve la configuración del columModel para crear el grid dinámico de la vista previa.
Nombre:	**function** DatosGridAction()
Descripción:	Devuelve a la vista la información de la vista previa para visualizarla en el grid dinámico.
Nombre:	**function** ExisteTablaAction()
Descripción:	Devuelve a la presentación si en la base de datos existe una tabla con el nombre del fichero a importar.

Nombre:	**function** ImportarAction()
Descripción:	Le solicita al Modelo ImporterFileModel que realice la importación de los datos devueltos por el driver hacia la tabla de la base de datos.
Nombre:	**function** Droptable()
Descripción:	Si ocurre algún error en la importación este método se encarga de eliminar la tabla creada durante este proceso erróneo.

Tabla 17. Descripción de Clases: importacionFileModel

Nombre: importacionFileModel	
Tipo de clase: (Modelo)	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**function** CreateTabbleFromFile($name,$campos)
Descripción:	Crea una tabla en la base de datos con el nombre y campo especificados.
Nombre:	**function** RegistrarTablaInmportada($alias, $dbw, $entidad, $Schema_name)
Descripción:	Registra el fichero importado como una tabla en los catálogos de la herramienta de auditoría.
Nombre:	**function** InsertDataFromFile($campos, $data, $alias)
Descripción:	Inserta la información en la tabla creada anteriormente.

Tabla 18. Descripción de Clases: FileDriver

Nombre: FileDriver

Tipo de clase: (Modelo)	
Atributo	Tipo
$handler	Objeto de php para manejar ficheros.
Para cada responsabilidad:	
Nombre:	**function** OpenFile($dir)
Descripción:	Abre el fichero de la dirección especificada y asigna al $handler su valor.
Nombre:	**function** CloseFile()
Descripción:	Cierra el fichero.
Nombre:	**function** CantTuplas()
Descripción:	Devuelve la cantidad de registros contenidos en el fichero.
Nombre:	**function** ReadFile($params)
Descripción:	Lee el contenido del fichero abierto.

Tabla 19. Descripción de Clases: CSVDriver

Nombre: CSVDriver	
Tipo de clase: (Modelo)	
Descripción	Implementación de la clase Driver para Microsoft SQL Server

2.6. Patrones de diseño empleados

Patrones de asignación de responsabilidades

Experto: Se puso en práctica en el componente de importación, con el uso de clases que poseen responsabilidades específicas a cumplir de acuerdo con la información que manejan, el componente cuenta con clases controladoras, modelos y entidades que poseen funciones concretas de acuerdo con la información que gestiona.

Controlador: Se utilizan cuando la aplicación es muy extensa, de esta forma, en el módulo en vez de tener un solo controlador y saturarlo, se tienen clases ***Controllers***, que son controladores más pequeños especializados en las funcionalidades de cada componente en nuestro caso se tienen dos controladores, ***ImportacionController*** y ***ImportacionFileController***.

Bajo acoplamiento: Las clases del componente de importación fueron diseñadas bajo este principio, para que solamente se establezcan las relaciones necesarias entre ellas.

Alta cohesión: Existe afinidad entre cada clase y los métodos que implementan, estas poseen responsabilidades vinculadas acordes a la información que controlan; y colaboran con otros objetos para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización.

Patrones de diseño

Modelo-Vista-Controlador (MVC): Dentro de las tecnologías utilizadas se encuentra el ZendFramework, el mismo proporciona un avanzado Model-View-Controller (MVC) el cual se utiliza para crear la estructura de nuestra solución.

Singleton: Mediante el uso de este patrón se accede a los elementos de configuración almacenados en el XML **modulesconfig.xml** para de esta forma obtener la configuración de la conexión de la base de datos del subsistema de auditoría, en la Figura 6 se muestra un ejemplo de uso.

```
$config = Zend_Registry::getInstance ()->config->bd->importar;
$dns = $config->gestor . ':' . 'dbname = ' . $config->bd . ' host = ' . $config->host;
$this->cnx = new PDO ( $dns, $config->usuario, $config->password );
```

Figura 6. Uso del singleton para obtener la configuración del subsistema

La implementación de este método ***GetInstance(),*** muestra que se invoca a un **singleton,** ver Figura 7

```
public static function getInstance()
{
    if (self::$_registry === null) {
        self::init();
    }
    return self::$_registry;
}
```

Figura 7. Implementacion del Singleton de Zend_Registry

Factory: Este patrón es uno de los más importantes de la solución. Es el encargado de crear el driver específico para cada origen de datos a la hora de importar los datos, en la Figura 8 se muestra la implementación del mismo dentro de la solución.

```
class FactoryModel extends ZendExt_Model{

    static function create($type)
    {
      $drivers_oportados= array(0 => 'PgsqlDriver', 1 => 'MysqlPgsqlDriver', 2 => 'MssQLPgsqlDriver');
       if(array_search($type, $drivers_oportados)!=false)
       {
            $driverclass =$type."Driver";
            return new $driverclass;
       }
       throw new ZendExt_Exception('AUD002');
    }
}
```

Figura 8 Implementación del patrón factory en la solución

2.7. Resultados de las Métricas

Resultados del instrumento de evaluación de las métrica Tamaño operacional de la clase (TOC) y (RC) para el requisito funcional principal, Importar servidor relacional.

Tamaño Operacional de la Clase (TOC)

La Figura 9 muestra los resultados de la métrica Tamaño Operacional de la Clase (TOC), donde en la gráfica se mide el porciento de cantidad de procedimientos que contiene una clase. Como se puede apreciar existe casi un 36% de clases que contienen menos de 6 procedimientos y un 55 % que presentan menos de 10 operaciones, lo que significa un valor aceptable y satisfactorio para la métrica.

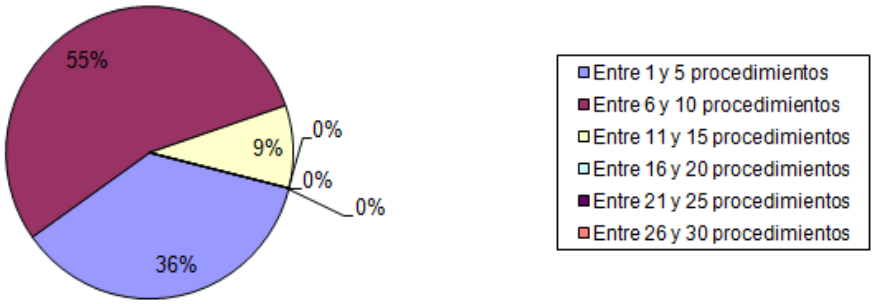


Figura 9. Representación del número de clases por procesos de la métrica TOC

A continuación en la Figura 10 se muestra la gráfica que mide el atributo responsabilidad de la métrica, obteniéndose resultados satisfactorios mostrando una responsabilidad baja con un valor de 100%.

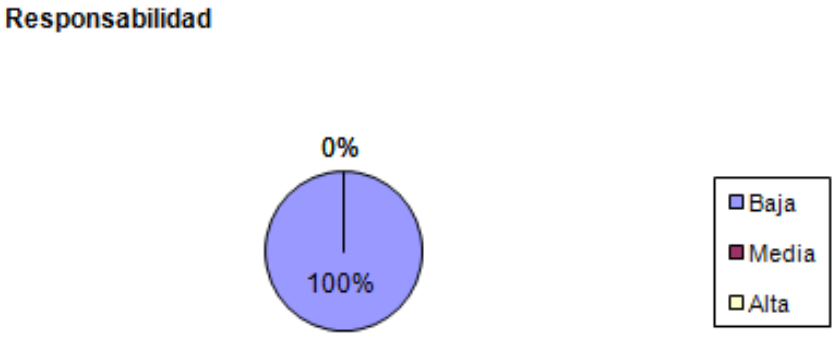


Figura 10. Representación de responsabilidad de la métrica TOC

El atributo que se mide en la gráfica que se muestra la Figura 11 después de haberse realizado la medición de la métrica, arrojó también resultados positivos puesto que la complejidad de las clases es un 100% baja.

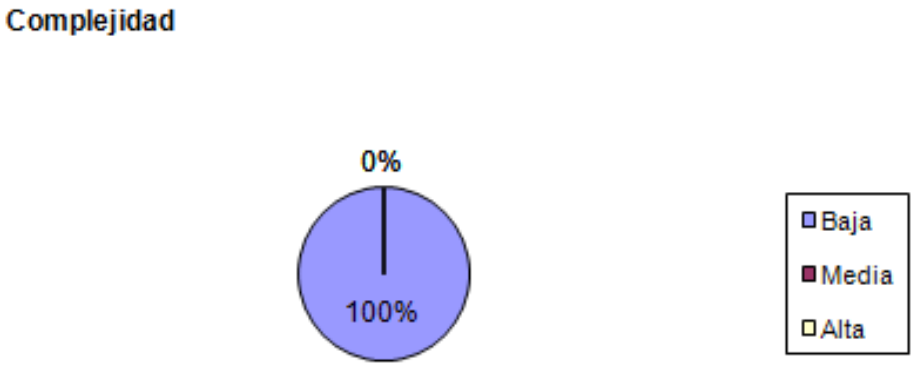


Figura 11. Representación de complejidad de la métrica TOC

En la Figura 12 se muestra que al igual que el atributo complejidad la reutilización del código presenta resultados satisfactorios al brindar valores por encima del 50% en una reutilización alta.

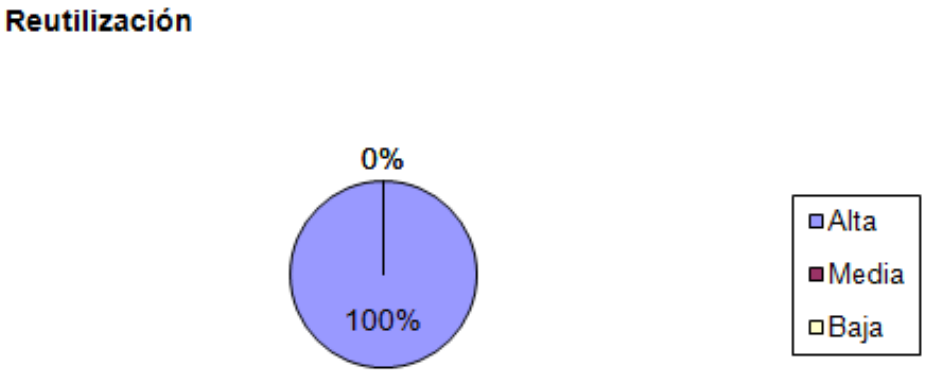


Figura 12. Representación de Reutilización de la métrica TOC

Relaciones entre las clases (RC)

La Figura 13 muestra el resultado de la métrica Relación entre Clases (RC), donde se mide el porciento de cantidad de dependencias que contiene una clase. Como se puede apreciar existe más de un 50% de clases que no contienen dependencias lo que significa un valor aceptable y satisfactorio para esta métrica.

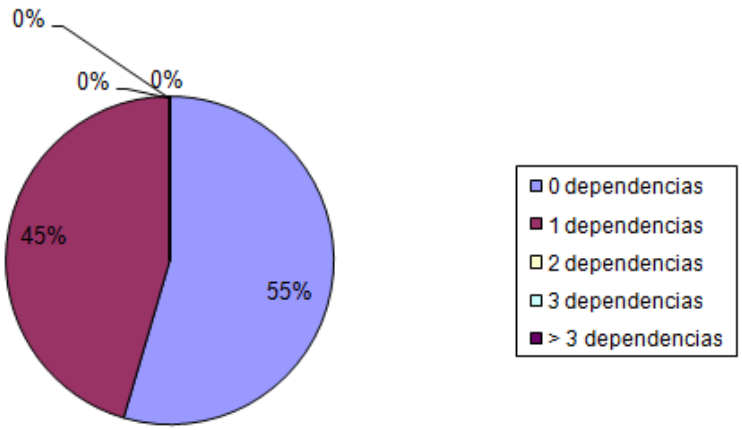


Figura 13. Representación del número de dependencias de las clases de la métrica RC

La Figura 14 muestra el resultado del atributo acoplamiento entre las clases del sistema, el cual presenta un 62% con valor ninguno, lo que quiere decir que las clases presentan una dependencia pobre, lo que significa que la medición de la métrica está dando frutos satisfactorios.

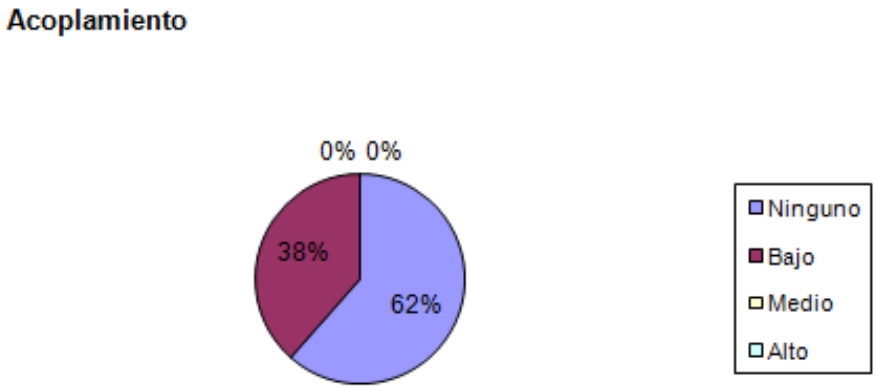


Figura 14. Representación del acoplamiento de la métrica RC

Por otra parte los resultados del atributo complejidad de mantenimiento son bastantes satisfactorios, con un 62% de baja complejidad, lo que significa fácil soporte y poca complejidad para mantener el código la Figura 15 muestra lo antes afirmado.

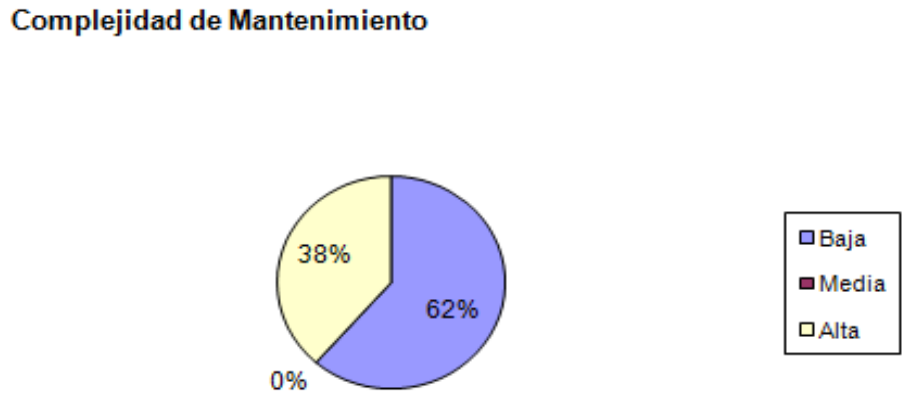


Figura 15. Representación de la complejidad del mantenimiento de la métrica RC

Luego de haber analizado varios parámetros, se llevó a cabo la medición de la variable cantidad de pruebas, que brindó resultados satisfactorios al presentar más de un 62% de bajo esfuerzo a la hora de realizar pruebas al diseño, en la Figura 16 se muestra dicho resultado.

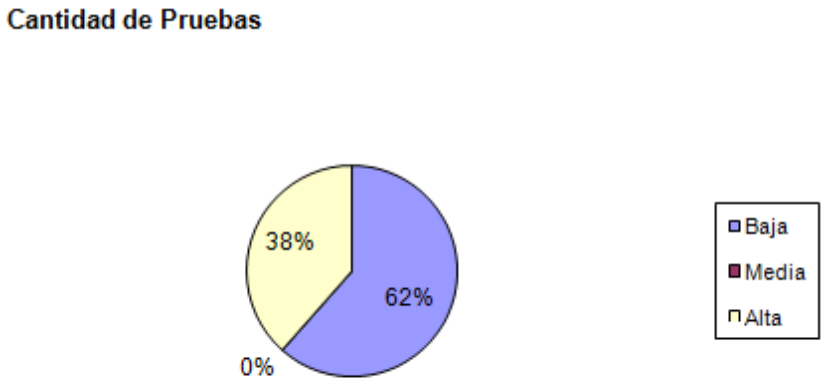


Figura 16. Representación de la cantidad de pruebas de la métrica RC

Finalmente se realizó la medición del atributo reutilización, para comprobar el nivel de reutilización del diseño de cases, la métrica generó resultados positivos al informar que el diseño de la solución posee más del 62% de reutilización de sus clases, este resultado es reflejado en la Figura 17.

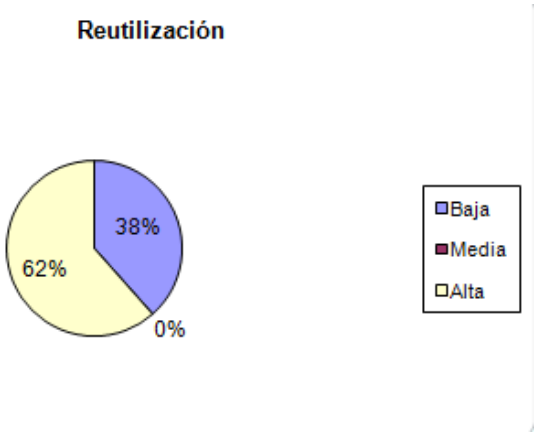


Figura 17. Representación de la reutilización de la métrica RC

Valoración de los resultados obtenidos

En conclusión, la evaluación de los resultados obtenidos durante la aplicación del instrumento de medición de la métrica Relación entre Clases y Tamaño Operacional de la Clase, demuestran el uso de un buen diseño de software en el diseño de la nueva versión de la herramienta de importación.

2.8. Modelo de datos

Un modelo de datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí. (101)

Para cumplir con las exigencias propuestas para el correcto desarrollo del conjunto de funcionalidades que debe presentar la versión a desarrollar se ha realizado el siguiente Diagrama Entidad-Relación el cual está compuesto por 6 entidades:

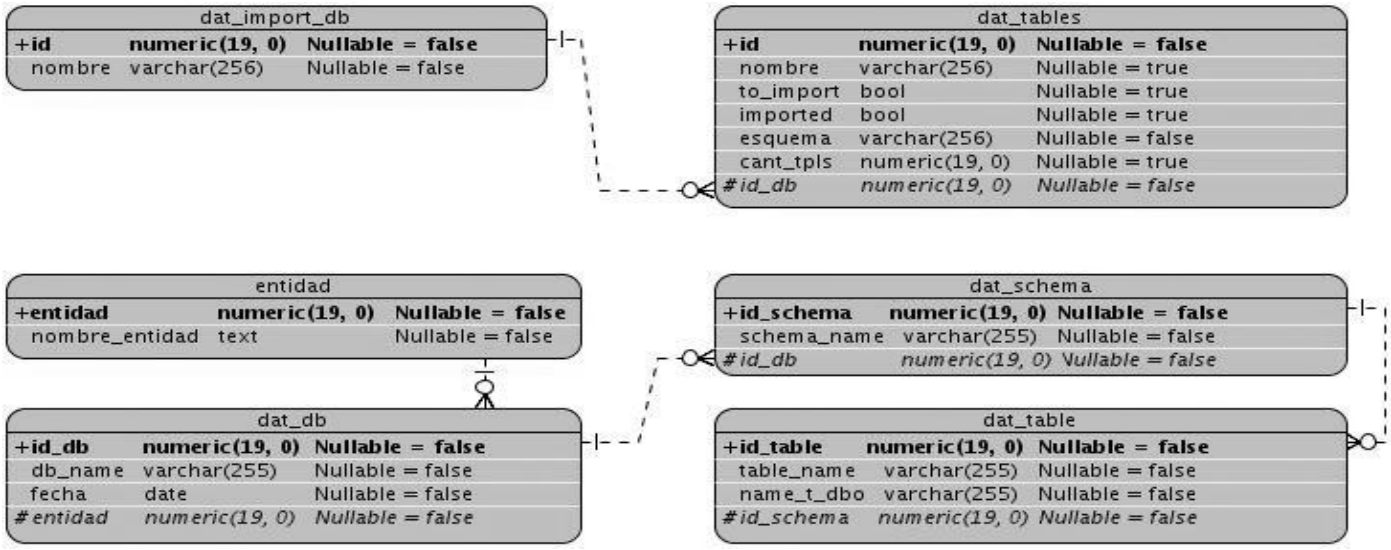


Figura 18. Modelo de datos del Componente importación

Descripción de las tablas del modelo de datos

Tabla 20. Descripción de las Tablas: dat_import_db

Nombre: dat_import_db		
Descripción: Tabla que almacena las bases de datos que están siendo importadas.		
Atributo	Tipo	Descripción
id	numérico	Es el identificador de la base de datos.
nombre	varchar	Almacena el nombre de la base de datos a importar.

Tabla 21. Descripción de las Tablas: dat_tables

Nombre: dat_tables		
Descripción: Almacena las tablas que están siendo importadas.		
Atributo	Tipo	Descripción
id	numérico	Es el identificador de la tabla

nombre	varchar	Nombre de la tabla.
to_import	booleano	Determina si la tabla va a ser importada o no.
Imported	booleano	Representa si la tabla ya fue importada o no.
Esquema	varchar	Almacena el nombre del esquema en caso de pertenecer a alguno.
cant_tpls	Numérico	Almacena la cantidad de registros que tiene la tabla
id_db	Numérico	Representa el identificador de la base de datos al cual pertenece la tabla.

Tabla 22. Descripción de las Tablas: entidad

Nombre: entidad		
Descripción: Almacena la relación de las entidades de las cuales se han importado datos en la herramienta de importación.		
Atributo	Tipo	Descripción
entidad	numérico	Es el identificador de la entidad.
nombre	varchar	Nombre de la entidad.

Tabla 23. Descripción de las Tablas: dat_db

Nombre: dat_db		
Descripción: Almacena la relación de las bases de datos pertenecientes a una entidad de las cuales se han importado datos en la herramienta de importación.		
Atributo	Tipo	Descripción
id_db	numérico	Es el identificador de la base de datos.
db_name	varchar	Nombre de la base de datos.
fecha	date	Representa la fecha en la cual fue realizada la importación de la base de datos.

entidad	numérico	Representa el identificador de la entidad a la cual pertenece la base de datos.

Tabla 24. Descripción de las Tablas: dat_schema

Nombre: dat_schema		
Descripción: Almacena la relación de los esquemas pertenecientes a una base de datos importada.		
Atributo	Tipo	Descripción
id_chema	numérico	Es el identificador del esquema.
schema_name	varchar	Nombre del esquema.
id_db	Numérico	Almacena el identificador de la base de datos a la cual pertenece el esquema.

Tabla 25. Descripción de las Tablas: dat_schema

Nombre: dat_table		
Descripción: Almacena la relación de las tablas pertenecientes a un esquema.		
Atributo	Tipo	Descripción
id_table	numérico	Es el identificador de la tabla.
table_name	varchar	Nombre de la tabla.
name_t_dbo	varchar	Almacena el nombre original en la fuente de datos.
id_schema	Numérico	Representa el identificador de esquema al cual pertenece la tabla.

CONCLUSIONES

En este capítulo se abordaron elementos del proceso de importación de datos en el subsistema de auditoría, se realizó la especificación de los requisitos asociados a la importación de datos. Se realizó el diseño de clases y la validación de la calidad del diseño propuesto, especificándose el uso de los patrones de diseño estudiados. Con el análisis de los resultados obtenidos mediante la aplicación de métricas al diseño se concluye que este cuenta con una calidad aceptable. Se realizó además el diseño del modelo de datos y la descripción del mismo. De esta manera se da cumplimiento a uno de los objetivos específicos planteados para la investigación: Realizar el análisis y diseño de la herramienta de Importación de datos 2.0 para establecer un punto de partida hacia la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3. INTRODUCCIÓN

En el presente capítulo se aborda la estructura del marco de trabajo donde se implementó el sistema. Se representa el diagrama de despliegue en conjunto con el diagrama de componentes. En el mismo se abordan los algoritmos no triviales de la solución, y el cálculo de su complejidad ciclomática. También se desarrolla la etapa de pruebas para garantizar la calidad del producto desarrollado.

3.1. Estructura del Marco de Trabajo

Cumpliendo con las decisiones arquitectónicas del equipo de arquitectura se define una estructura donde van a estar ubicados cada uno de los subsistemas implementados dentro de la aplicación, de manera que facilite la organización y claridad durante el desarrollo y mantenimiento, la especificación que a continuación se presenta está enfocada al componente de importación del subsistema de auditoría.

La carpeta denominada ***apps*** es donde se almacenan los controladores y modelos de cada uno de los componentes correspondientes a los subsistemas. En la Figura 19 se muestra dicho contenido para el componente y el subsistema en cuestión.

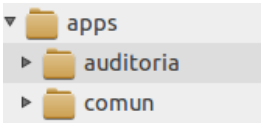


Figura 19. Carpeta de aplicación correspondiente al Subsistema auditoria

comun: Comprende la configuración específica para el Subsistema, contiene la carpeta recursos y dentro de esta una denominada xml. Esta última incluye a los ficheros: ***ioc***, ***validator***, ***exception*** que serán explicados a continuación.

ioc: Es donde se publican los servicios que brinda cada uno de los componentes del Subsistema en cuanto a nombre de clases, funciones y tipo de resultado.

validator: Chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.

exception: Se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

El componente importar va a contener un conjunto de paquetes que serán especificados a continuación:

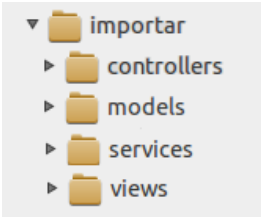


Figura 20. Paquete importar correspondiente a la carpeta de aplicación

En el paquete ***controllers*** se encontrarán las clases controladoras encargadas de gestionar las funcionalidades del sistema.

El paquete ***models*** contiene dos paquetes para agrupar clases, el paquete ***bussines*** contendrá las clases necesarias para acceder a los datos que persisten en la base de datos, el paquete ***domain*** debe contener las clases generadas por el ORM Doctrine a partir de cada una de las tablas existentes en la base de datos. Cada una de estas clases heredará de una clase generada igualmente por el Doctrine las cuales se ubican en otro paquete dentro de este llamado ***generated***, en nuestro caso estos paquetes permanecerán vacío ya que no se realiza mapeo con el ORM Doctrine.

El paquete ***models*** estará estructurado de la siguiente forma:

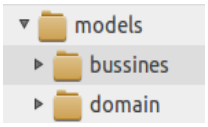


Figura 21. Paquete modelos correspondiente a la carpeta de aplicación

El paquete ***views*** contendrá los paquetes idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente, ver Figura 22.

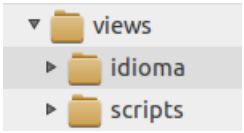


Figura 22. Paquete view correspondiente a la carpeta de aplicación

Al mismo nivel de la carpeta ***apps*** mencionada anteriormente debe existir además una carpeta que contiene las vistas de los subsistemas y componentes. Dicha carpeta se denomina ***web***, ver Figura 23.

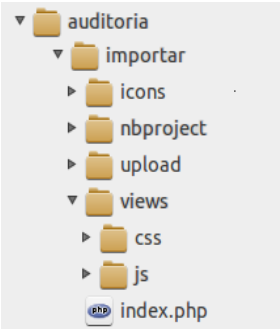


Figura 23. Carpeta de diseño correspondiente al componente importar

Index.php: Este fichero incluye la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código permanece igual para todos los componentes.

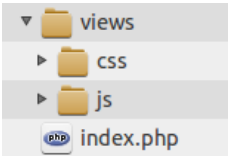


Figura 24.Paquete view correspondiente a la carpeta de diseño

La carpeta ***view*** contendrá los ***css*** y los ***js*** que se explicarán a continuación.

El paquete ***css*** incluirá las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido.

El paquete ***js*** comprenderá las clases JavaScript necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesto por paquetes con los nombres de las funcionalidades del componente.

3.1.1. Nomenclatura

Clases vistas:

Se definió para las clases que se encuentran dentro de Views que el nombre se escribirá sin ningún sufijo o prefijo. Los nombres se escribirán en notación camello (Camel Case). Ejemplo: ***GestionarSubordinacion.js.***

Clases Controladoras.

Se definió para las clases controladoras que después del nombre se les colocará como sufijo la palabra:”Controller”. Ejemplo: ImportacionController.php

Clases en los modelos.

Business (Negocio)

Se definió para las clases que se encuentran dentro de Business que después del nombre se les colocará como sufijo la palabra:”Model” Ejemplo: ImportacionModel.php

Domain (Dominio) y Bases del Dominio (Generated).

Se definió para las clases que se encuentran dentro de Domain que el nombre se escribirá sin ningún sufijo. Ejemplo: PgSql.php.

Nomenclatura de las funciones.

Se definió que el nombre de las funciones se escribe con la primera palabra en minúscula, para el caso, específico de que sea un nombre compuesto se empleará la notación CamelCasing (notación camello) y de forma tal que al leerlo se conozca el objetivo de la misma. Ejemplo: insertarDatos(). En caso particular que la función sea una acción de la clase controladora u otra se escribe después del nombre de la función la palabra “Action” como sufijo. Ejemplo: testContectionAction().

Nomenclatura de las variables.

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guión bajo. Ejemplo:

variable

variable_compuesta

Nomenclatura de los comentarios.

Se definió el uso de comentarios claros y precisos que ayudarán a una mejor comprensión del código implementado para que se entiendan sus objetivos específicos.

En las clases:

En el código fuente de una clase se escribe una pequeña descripción donde se expliquen los propósitos de la misma de forma general, así como su autor y sistema al que pertenece. Dicha descripción se escribe como se muestra en la Figura 25.

```
/**
* Nombre de la clase *
* Descripción *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche)
*/
```

Figura 25 Descripción de las clases

Quedando en la clase controladora de la manera representada en la Figura 26.

```
/**
 * Nombre: ImportacionController
 * Descripción : Clase controladora del componente importador
 * @author Juan  José  Rosales Rodriguez
 * @package Auditoría
 * @subpackage Importador
 * @copyright ERP
 * @version 2.0
 */
```

Figura 26. Ejemplo de comentario de clase

Comentarios por líneas

Se utilizan también los comentarios para líneas de código en específico que ayudan a aclarar la complejidad de algunos algoritmos. Para ellos se utiliza //, y seguidamente el comentario que describe la línea de código, tal y como se representa en la Figura 27.

```
//SE CREA LA INSTANCIA DEL DRIBER EN LA FABRICA
$driver = FactoryModel::create($motorS);
$driver->Connect($dbuser,$dbpass, $dbhost,$dbname,$port);
```

Figura 27. Ejemplo de comentario por líneas de código

3.1.2. Tratamiento de errores

Los errores se tratan en dependencia de la capa correspondiente:

Interfaces de usuario.

En el caso de las interfaces de usuario las validaciones se realizan médiate la librería de presentación EXT JS del marco de trabajo la misma presenta validaciones para campos vacío, numéricos, de fecha permite el uso de expresiones regulares para validaciones de campos por ejemplo campos alfa numéricos etc., Otro tipo de validación en el cliente es mostrar un mensaje cuando ocurre un evento determinado o no se cumplen todos los parámetros para realizar la petición al servidor. El ejemplo de la Figura 28 muestra una validación para manejar la respuesta del servidor.

```
fpAsistentedeImportacion1.getForm().submit({
    url     :'testconection',
    method  :'POST',
    waitMsg :perfil.etiquetas.lbcnetando,
    params:{},
    success: function (form,action)
    {
       conected=  true;
       stDataB.reload({params:{dbgestor:dbgestor,dbuser:dbuser,dbpass:dbpass,dbport:dbportf}});
       mostrarMensaje(1,perfil.etiquetas.lbconectadook);
       atnpadreSistemaGuardar.setText(dbgestor+" en "+dbhost);
    },
    failure: function (form, action)
    {
       mostrarMensaje(3,perfil.etiquetas.lbconetafalse);
       stDataB.removeAll();
       cmbbd.setValue('');
       conected=false;
    }
```

Figura 28. Tratamiento de errores en la interfaz

Tratamiento de errores a nivel de controlador o negocio.

En el servidor se validan nuevamente los datos enviados desde el cliente, también se realiza el tratamiento de errores tratando las excepciones lanzadas por el negocio o por errores del marco de trabajo para ello se usa el componente de excepciones del marco de trabajo ZendExt_Exception, a continuación en la Figura 29 se muestra un ejemplo de su uso:

```
try
{
   $driver = FactoryModel::create($motorS);
   $driver->Connect($dbuser,$dbpass, $dbhost,NULL,$port,$sid);
}
catch (DoctrineException $aa)
{
  throw new ZendExt_Exception('AUD1',$aa);
}
```

Figura 29. Tratamiento de errores a nivel de controlador o negocio

3.2. Descripción de los algoritmos no triviales

Durante la implementación del componente, la clase **ImporterModel** fue una de las que mayor complejidad presentó, obteniéndose durante su desarrollo algoritmos no triviales cuya dificultad de resolución provocó analizarle diferenciadamente. Ejemplo de ello fue la función **Createtable**($**CAMPOS, $ALIAS, $ESQUEMA, $OPCION)**, que se encarga de crear la tabla donde se almacenaran los datos, su complejidad reside en lograr la migración de los tipos de datos de la tabla externa para el gestor de la aplicación.

Análisis de la complejidad ciclomática

La complejidad ciclomática es una métrica del software, propuesta por Thomas McCabe en 1976, que proporciona una medición cuantitativa de la complejidad lógica de un programa. El análisis de la complejidad ciclomática de un algoritmo define el número de caminos independientes del conjunto básico del programa y proporciona el número mínimo de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

McCabe establece una concordancia entre los valores resultantes de aplicar la métrica y el nivel de riesgo que implicaría probar, entender y modificar el código.

Complejidad ciclomática	Evaluación de riesgo
1-10	Programa simple, sin mucho riesgo
11-20	Más Complejo, riesgo moderado
21-50	Complejo, programa de alto riesgo
50	Programa no testeable, muy alto riesgo

Figura 30. Complejidad ciclomática vs evaluación de riesgo

El cálculo de la complejidad ciclomática para la función en cuestión, **Createtable**, brinda una medida cuantitativa de su complejidad lógica, conociendo así el riesgo que implica de acuerdo con los valores expuestos en la tabla. Primeramente se enumeran las instrucciones del código de la función que constituyen diferentes alternativas del flujo básico. (Ver Anexo 5) Luego se representa el grafo de flujo asociado al código antes presentado atendiendo los siguientes componentes:

Nodo: Círculos del grafo de flujo que representan una o más secuencias procedimentales.

Aristas: Saetas que unen los nodos del grafo, representan el flujo de control.

Regiones: Son las áreas delimitadas por las aristas y nodos; el área exterior del grafo también se cuenta como una región más.

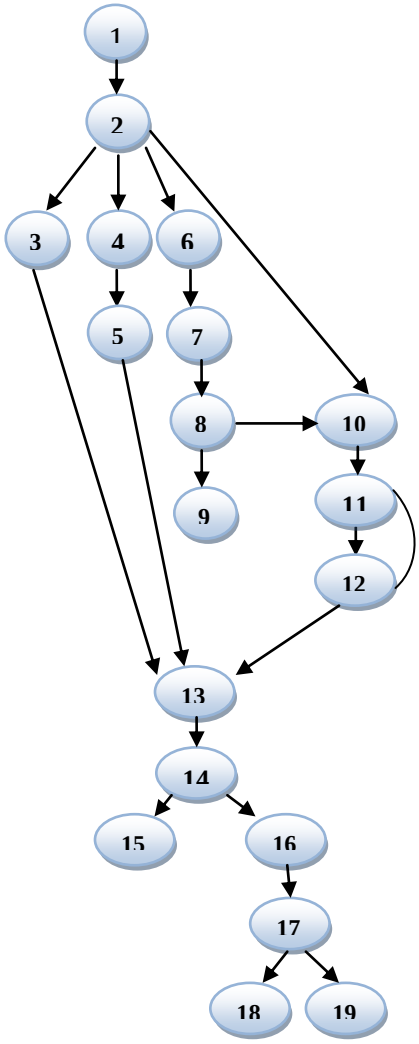


Figura 31. Grafo de flujo asociado al algoritmo Createtable

Concluida la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código mediante las siguientes vías:

V (G) = A – N + 2, donde A es el número de aristas en el grafo, N es el número de nodos.

V (G) = (22 – 19) + 2

V (G) = 5

V (G) = P + 1, siendo “P” la cantidad total de nodos predicados (nodos de los cuales parten dos o más aristas).

V (G) = 4 +1

V (G) = 5

V (G) = R, donde “R” es la cantidad total de regiones.

V (G)= 5 complejidad ciclomática

De acuerdo con el resultado obtenido, la función analizada se clasifica como un algoritmo simple sin mucho riesgo para el sistema.

3.3. Diagrama de Despliegue

Los diagramas de despliegue se utilizan para modelar la vista de despliegue estática. Se muestran tres nodos: un servidor de base de datos, un servidor web y una máquina cliente, se reflejan los canales de comunicación entre ellos.

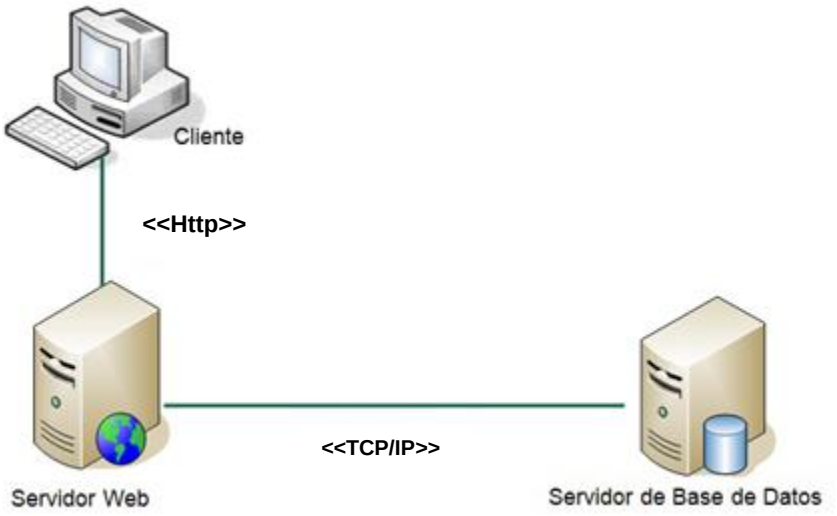


Figura 32. Diagrama de Despliegue

3.4. Diagrama de componentes

Es un diagrama que se utiliza para modelar la vista estática del sistema, para ello, puede mostrar un conjunto de elementos tales como componentes, subsistemas de implementación e interfaces. Los componentes pueden ser ejecutables, código fuente, librerías, bases de datos físicas, documentos, etc. A

continuación en la Figura 33 se muestra el diagrama de componentes del subsistema de auditoría, se resalta el componente desarrollado en la investigación.

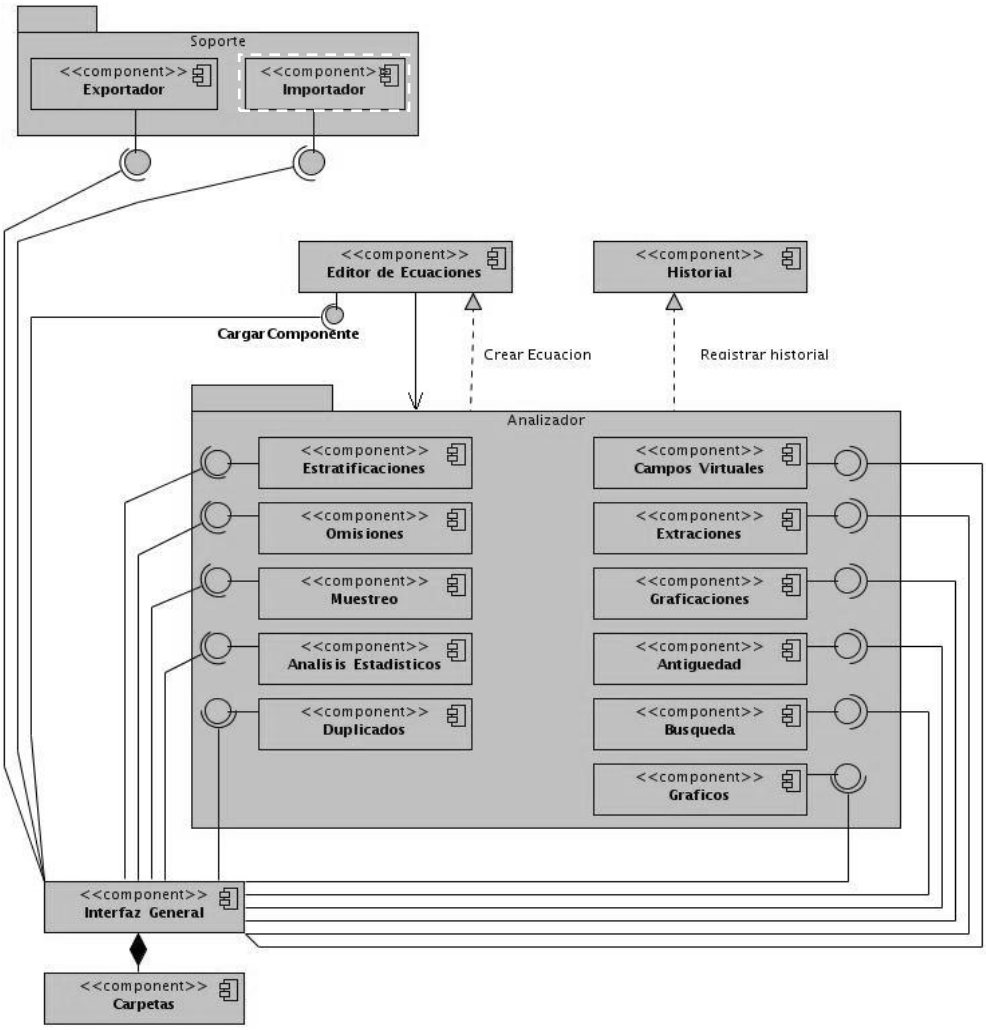


Figura 33. Diagrama de componentes de la herramienta de auditoría.

3.5. Pruebas

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. El objetivo de la prueba es diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio. (PRESSMAN 2005) Cualquier proceso de ingeniería puede ser probado de una de dos formas:

Se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.

Se permiten desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

La primera aproximación se denomina prueba de la caja negra y la segunda prueba de caja blanca.

3.5.1. Prueba de Caja Negra

"Las pruebas de caja negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero de software derivar conjuntos de condicione de entradas que ejercitaran por completo todos los funcionales de un programa." (Presman 14.2)

Las pruebas se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías:

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

3.5.2. Diseños de Casos de Prueba

Los casos de pruebas se diseñaron para que fuera probada la herramienta obtenida en el proceso de desarrollo, se le realizaran pruebas por el colectivo de Calidad del Proyecto ERP-CUBA.

Las especificaciones para llevar a cabo una correcta revisión de la herramienta se detallan a continuación. Condiciones de Ejecución.

- Debe seleccionar el menú Subsistemas/Auditoría/Gestor de auditorías/importar
- Es necesario para la utilización del Asistente de importación que conozca los parámetros de configuración de conexión de un servidor de bases de datos para importar sus datos.

Caso de prueba **Importar bases de datos relacionales** (Ver Anexo 1)

Caso de prueba **Importar datos de ficheros** CVS (Ver Anexo 2)

3.5.3. Resultados de las pruebas

La Tabla 26. Muestra los resultados de los procesos de revisión por parte de calidad interna del CEIGE.

Tabla 26 Resultados de los procesos de revisión.

Entregas	N/C Significativas	N/C No significativas	Total
1ra Entrega 10/05/2011	4	1	5
2da Entrega 31/05/2011	3	0	3

Todas las no conformidades detectadas fueron resueltas, quedando liberada por calidad interna para pasar a las siguientes etapas de pruebas por parte del equipo de calidad de la universidad **Calisoft**.

CONCLUSIONES

Se finaliza la etapa de Implementación del sistema, habiéndose obtenido el modelo de despliegue. Se concluye con la etapa de prueba del sistema. Se describieron los casos de pruebas del sistema definiendo entradas válidas y no válidas, obteniendo como resultado que el sistema cumple con las funcionalidades descritas. Se desarrollaron pruebas por el equipo de Calidad del Proyecto siendo liberada la herramienta como estable y lista para su explotación.

CONCLUSIONES

CONCLUSIONES GENERALES

Con la realización de este trabajo de diploma se logró cumplir con los objetivos trazados en el proceso de investigación y desarrollo, pues se logró dar una solución eficiente y amigable mediante una aplicación Web a la problemática existente. Se obtuvo una herramienta en su versión 2.0 capaz de importar los datos desde diferentes fuentes como PostgreSQL, Mysql, Mssql y CSV hacia el ambiente propio de la herramienta de auditoría, esta herramienta está avalada por el colectivo de Calidad del CEIGE, permitiendo dejar definida una solución práctica de una herramienta de importación de datos.

RECOMENDACIONES

Para un posterior desarrollo y perfeccionamiento del sistema implementado se recomienda:

- Extender los drives de la herramienta para que sea capaz de importar datos de otros gestores como: Oracle y SQLlite.
- Extender los drives de ficheros para permitir la importación de otros tipos de ficheros como Excel, DBF y TXT.

REFERENCIA BIBLIOGRAFÍCA

1. **SAP.** *Statement of Auditing Practice 1009, Epígrafe 26.*

2. **López Fernando, Castaño Pedro.** *Software de auditoria.* 2009.

3. **Páez, Fransisco Javier Martínez.** Adictos al trabajo. [En línea] 05 de 11 de 2006. [Citado el: 05 de 01 de 2011.] http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=CSV.

4. **Alvarez, Sara.** Desarrollo Web. [En línea] 31 de 06 de 2007. [Citado el: 05 de 01 de 2011.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

5. **Corporation, Microsoft.** Microsof SQL Server. [En línea] 11 de 11 de 2005. [Citado el: 06 de 01 de 2011.] http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx.

6. **Oracle.** mysql. [En línea] 2010. [Citado el: 04 de 01 de 2010.] http://dev.mysql.com/doc/refman/5.0/es/introduction.html.

7. **ACL.** ACL. [En línea] ACL, 2011. [Citado el: 11 de 01 de 2011.] http://www.acl.com/products/desktop.aspx?language=es.

8. **CaseWare.** IDEA-CASE WARE. [En línea] CaseWare . [Citado el: 04 20, 2011.] http://www.caseware.com/products/idea.

9. *Definición del ciclo de vida del proyecto.* **Vega, Yanet.** Habana : s.n., 2009.

10. **Jenndy Sanchez Cid, Rogelio Elias Guerrero.** *Desarrollo de un Editor de Ecuaciones para un Software de Auditoría.* La Habana : s.n., 2009.

11. *El Lenguaje Unificado de modelado.* **James Rumbaugh, Ivar Jacobson, Grady booch.** Madrid : Addison-Wesley, 1999.

12. Que es PHP. [En línea] PHP, 20 de 05 de 2011. [Citado el: 22 de 05 de 2011.] http://www.php.net/manual/es/intro-whatis.php.

13. **Valdés, Damián Pérez.** Maestros del Web. [En línea] 01 de 05 de 2010. [Citado el: 04 de 02 de 2011.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

14. **Zend.** Zend Framework. [En línea] Zend, 2011. [Citado el: 01 de 03 de 2011.] http://framework.zend.com/.

15. Doctrine. [En línea] Sensio Labs, 06 de 01 de 2011. [Citado el: 22 de 05 de 2011.] http://www.doctrine-project.org/.

16. ExtJS 2.2 . [En línea] Sencha, 04 de 08 de 2008. [Citado el: 05 de 04 de 2011.] http://www.sencha.com/blog/ext-22-released/.

17. PostgreSQL. [En línea] 2011. [Citado el: 15 de 05 de 2011.] http://www.postgresql.org/.

18. **ORACLE Corporation.** Netbeans. [En línea] ORACLE, 2011. [Citado el: 10 de 05 de 2011.] http://netbeans.org/index_es.html.

19. **Valle, Jose del.** Monografias. *Monografias.* [En línea] 23 de 10 de 2008. [Citado el: 04 de 06 de 2011.] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.

20. **Gracia, Por Joaquin.** IngenieroSoftware-Diseño de Software Orientado a Objetos. [En línea] 27 de 05 de 2005. [Citado el: 23 de 05 de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

21. **MCArthur, Kevin.** *Pro PHP Patterns,Framework, Testing and more.* 2008. ISBN-13 (pbk): 978-1-59059-819-1.

22. **Mora, Roberto Canales.** Adictos al Trabajo. [En línea] 22 de 12 de 2003. [Citado el: 12 de 04 de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

23. Wikipedia. [Online] 05 10, 2011. [Cited: 05 2011, 20.] http://es.wikipedia.org/wiki/Diagrama_de_clases.

BIBLIOGRAFÍA

1 **PRESSMAN**, R. S. Ingeniería de software. Un enfoque practico. Mc Graw Hill, 1998.

2 **Extjs**. http://extjs.com/. [En línea] 2009. http://extjs.com/.

3 PHP en Castellano, 2009 [Disponible en: http://www.programacion.com/php/.

4 MySQL Hispano. 2009 [Disponible en: http://www.mysql-hispano.org/.

5 ACL, 2009 [Disponible en: http://www.acl.com/es/.

6 Software para Auditoría Interna, Auditoría de Sistemas y Calidad de Datos. 2009 [Disponible en: http://www.eniac.com/productos/acl.htm.

7 Visual Paradigm, 2009 [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/].

8 **James Rumbaugh**, Ivar Jacobson, Grady booch. El Lenguaje Unificado de modelado. Madrid: Addison-Wesley, 1999.

9 Visual Paradigm International. [En línea] 2005. http://www.visual-paradigm.com/product/vpuml/communityedition.jsp.

10 **Kioskera**. Kioskera. [En línea] 2008. http://es.kioskea.net/forum/affich-51592-consulta-de-registros-en-bd-de-mysqlcon-php.

11 **Microsoft**. Microsoft. [En línea] 2007. http://msdn.microsoft.com

12 **Martin, Robert C.** Design Principles and Design Patterns

13 **Matt Zandstra.** PHP Objects,Patterns, and Practice, 2008

14 **Christopher Jones, Alison Holloway.** The Underground PHP and Oracle Manual , 2008

15 **Shea Frederick, Colin Ramsay.** Learnig Ext JS, 2008

16 **Jesus D. Garcia. Jr.** Ext JS in Action ,2009

GLOSARIO DE TÉRMINOS

ACL: Business Assurance Analytics. Software para Auditoría financiera y análisis de datos.

AJAX: Asynchronous JavaScript and XML. Javascript Asíncrono y XML. Es una técnica de desarrollo Web para crear aplicaciones interactivas, en las que se puede enviar y recibir información de los servidores sin necesidad de recargar las páginas completamente.

Apache: En informática es el nombre de un software servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.

Aplicaciones Web: Software que se utiliza en una red accediendo a servidores de aplicaciones que brindan determinados servicios.

Arquitectura Cliente-Servidor: Consiste básicamente en que un programa -el cliente- realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora, es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Arquitectura de software: Disciplina dedicada a definir los parámetros estructurales del software.

AS/400: (Antes llamado OS/400) es un sistema operativo de IBM. Muy usado en informática de gestión.

ASCII (American Standard Code for Information Interchange): Es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales. Norma de codificación de caracteres en 7 bits, sin recomendación de paridad, empleada para el intercambio de datos entre sistemas de información y sistemas de comunicación de datos.

Componente: En informática son aplicaciones que no funcionan por si solas sino que interactúa con otra aplicación para aportarle una función o utilidad específica.

ERP: Sistema de planificación empresarial.

Framework: En informática es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

GLOSARIO DE TERMINOS

HTML (Lenguaje de Marcado de Hipertexto): Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP: HyperText Transfer Protocol. Protocolo de transferencia de hipertexto. Protocolo utilizado en la Web.

IDE: Integrated Development Environment. Entorno de Desarrollo Integrado. Son un tipo de editores de texto realizados especialmente para programar en los distintos lenguajes, que proveen de herramientas de compilación, completamiento de código, y otros valores agregados.

IDEA: Software Auditoria y análisis de dato, inicialmente Interactive Data Extraction and Analysis.

Interfaz: Término utilizado en la informática para referirse a un elemento que sirve de conexión a otros dos independientes.

Lenguaje de programación C: Es uno de los lenguajes de programación más populares para crear software de sistemas, ya que con él se pueden lograr funcionalidades de sistema a bajo nivel.

Lenguaje de programación Perl: Lenguaje de programación de scripts multiplataforma muy utilizado para realizar actividades de procesamiento de texto.

Metodología: Una metodología es un conjunto ordenados de pasos a seguir para cumplir un objetivo. Dentro de la Ingeniería de Software, el objetivo es el desarrollo de software de alta calidad que cumpla con las necesidades del usuario (cliente).

Microsoft Excel: Es una aplicación para manejar hojas de cálculos. Este programa fue y sigue siendo desarrollado y distribuido por Microsoft, y es utilizado normalmente en tareas financieras, y actualmente integrado en el paquete Microsoft Office.

Microsoft Word: Es un procesador de texto creado por Microsoft, y actualmente integrado en el paquete Microsoft Office.

MVC: Modelo Vista Controlador

MySQL: es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.

GLOSARIO DE TERMINOS

Open Source: Código abierto. Término con el que se conoce al software distribuido y desarrollado libremente. Fue utilizado por algunos usuarios de la comunidad del software libre, como reemplazo al ambiguo nombre original en inglés del software libre, free software.

Oracle: Es una herramienta cliente/servidor para la gestión de Bases de Datos.

PHP (Hypertext Preprocessor): Lenguaje de script diseñado para la creación de páginas Web activas, muy popular en Linux, destaca por su capacidad de ser embebido en el código HTML.

PostgreSQL: Es un Sistema de Gestión de bases de datos Objeto-Relacionales (ORDBMS) libre. **Proceso:** Secuencia de actividades invocadas para producir un producto de software.

Servidor de aplicaciones: En Informática se denomina un servidor en una red de computadoras que ejecuta ciertas aplicaciones, como lo son los servidores de Apache, Internet Information Server.

Sistema Gestor de Bases de Datos (SGBD): Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Software: Conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

SQL: Lenguaje de consulta estructurado (SQL Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. T

AAC (Técnicas de auditoría asistida por computadora) Cualquier técnica de auditoría automatizada, tal como el software general de auditoría, generadores de datos de prueba, programas de auditoría computadorizados, y sistemas expertos en auditoría.

Tecnología: Conjunto de los conocimientos propios de una técnica. **TIC:** Tecnología de la Información y las Comunicaciones.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje Unificado de Modelado

GLOSARIO DE TERMINOS

UNIX: Sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Versión: Número que indica el nivel de desarrollo de un programa.

Vistas: En informática se emplea para referirse a una consulta SQL de una o varias tablas, también se les considera una tabla virtual. **XML** (Extensible Markup Language): Es un metalenguaje extensible de etiquetas, permite definir la gramática de lenguajes específicos.

ANEXOS

Anexo 1. Diseño de caso de prueba para el requisito **Importar bases de datos relacionales.**

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
1 – Importar bases de datos relacionales.	Accede al servidor de base de datos donde se encuentran los datos necesarios para realizar la auditoría. Luego, se procede a importar los datos seleccionados para la base de datos local de la herramienta de auditoría.	EP 1.1 Se selecciona importar\Importar bases de datos relacionales.	➢ Se especifica el servidor, gestor, usuario y contraseña donde se encuentra el contenedor de información a importar. ➢ Se presiona el botón Conectar. ➢ Se selecciona la base de datos a importar. ➢ Se selecciona la estructura de la base de datos a importar. ➢ Se presiona el botón Importar.
		EP 1.2: No se especificó correctamente el usuario de acceso.	➢ Se emite un mensaje de error “Por favor verifique nuevamente que hay campos con valores incorrectos”

		EP 1.3: No se especificó correctamente la contraseña de acceso.	➢ Se emite un mensaje de error “Falló la conexión revise los datos”
		EP 1.4 No se especificó correctamente el servidor donde se encuentra el contenedor a importar.	➢ Se emite un mensaje de error “Por favor verifique nuevamente que hay campos con valores incorrectos”
		EP 1.5 No se especifica correctamente el puerto para la conexión.	➢ Se emite un mensaje de error “Por favor verifique nuevamente que hay campos con valores incorrectos”

Descripción de las variables:

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción
1	Dirección servidor	Campo de texto	No	Se especifica la dirección del servidor para importar. Los datos a introducir sólo pueden ser números (1…9) y con el formato específico de direcciones IP.
2	Gestor	ComboBox	No	Se selecciona el gestor de bases de datos al cual se va a conectar.
3	Puerto	Numérico	No	Se escribe el puerto por el cual se va a realizar la conexión.
4	Usuario	Campo de texto	No	Se introduce el usuario. Pueden ser números y letras. Es validado.
5	Contraseña	Campo de texto	No	Se introduce la contraseña. Pueden ser números y letras. Es validado.

Juegos de datos a probar:

ID Esc.	Escenario	Variable Servidor	Variable Gestor	Variable Puerto	Variable Usuario	Variable Contraseña	Respuesta del sistema.
EP1	Seleccionar asistente de importación e importar bases de datos relacionales	V(10.7.17. 228)	V(pgsql)	V(5432)	V(auditoria)	V(8805224850 3)	Estadística: Total de tablas: Para importar: Importadas: Con conflicto:
EP2	No se especificó correctamente el usuario de acceso.	V(10.7.17.2 28)	V(pgsql)	V (asd)	V(auditoria)	V(88052248503)	“Por favor verifique nuevamente que hay campos con valores incorrectos”
EP3	No se especificó correctamente la contraseña de acceso.	V(10.7.17.2 28)	V(pgsql)	V (5432)	I(gkjugfksj,124 bj,nvd)	V(asd)	Falló la conexión revise los datos.

EP4	No se especificó correctamente el servidor donde se encuentra el contenedor a importar.	V(10.7.17.1)	V(pgsql)	V (5432)	V(auditoria)	V(88052248503)	Falló la conexión revise los datos.
EP5	EP 1.5 No se especifica correctamente el puerto para la conexión.	V(10.7.17.1)	V(pgsql)	V (3306)	V(auditoria)	V(88052248503)	Falló la conexión revise los datos.

Anexo 2. Diseño de caso de prueba para el requisito **Importar ficheros CSV.**

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
1 – Importar ficheros CSV.	Subir el fichero para el servidor, luego configurar las opciones de importación, importar los datos del fichero hacia el gestor de la aplicación.	EP 1.1 Se selecciona importar\Ficheros.	➢ Se selecciona el tipo de fichero. ➢ Se busca el fichero a importar. ➢ Se presiona el botón subir al servidor. ➢ Se presiona el botón siguiente. ➢ Se configura la importación. ➢ Se presiona el botón importar.
		EP 1.2: No se seleccionó correctamente el fichero.	➢ Se emite un mensaje de error “El fichero no es correcto.”
		EP 1.3: No se seleccionó un tipo de fichero.	➢ Se emite un mensaje de error “Se debe seleccionar un tipo de fichero.”

Descripción de las variables:

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción
1	A partir de.	Campo de numérico	Si	Se especifica a partir de que registro se comenzará a importar los datos.
2	Delimitador	Campo de texto	No	Se especifica el delimitador por el cual se separarán los campos.
3	Quote	Campo de Texto	No	Este campo especifica la configuración de los valores.

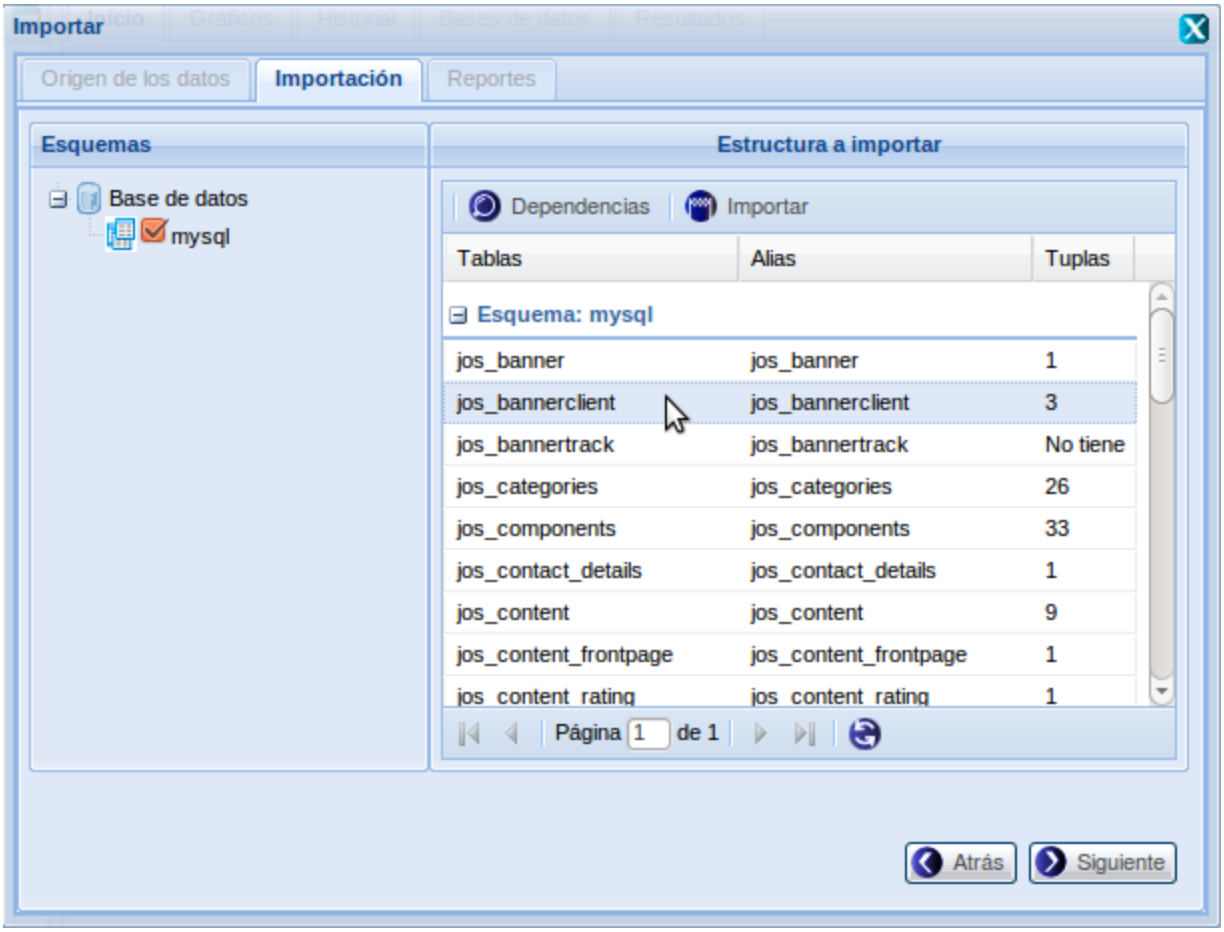
Anexo 3 Especificación de requisito **Importar bases de datos relacionales**

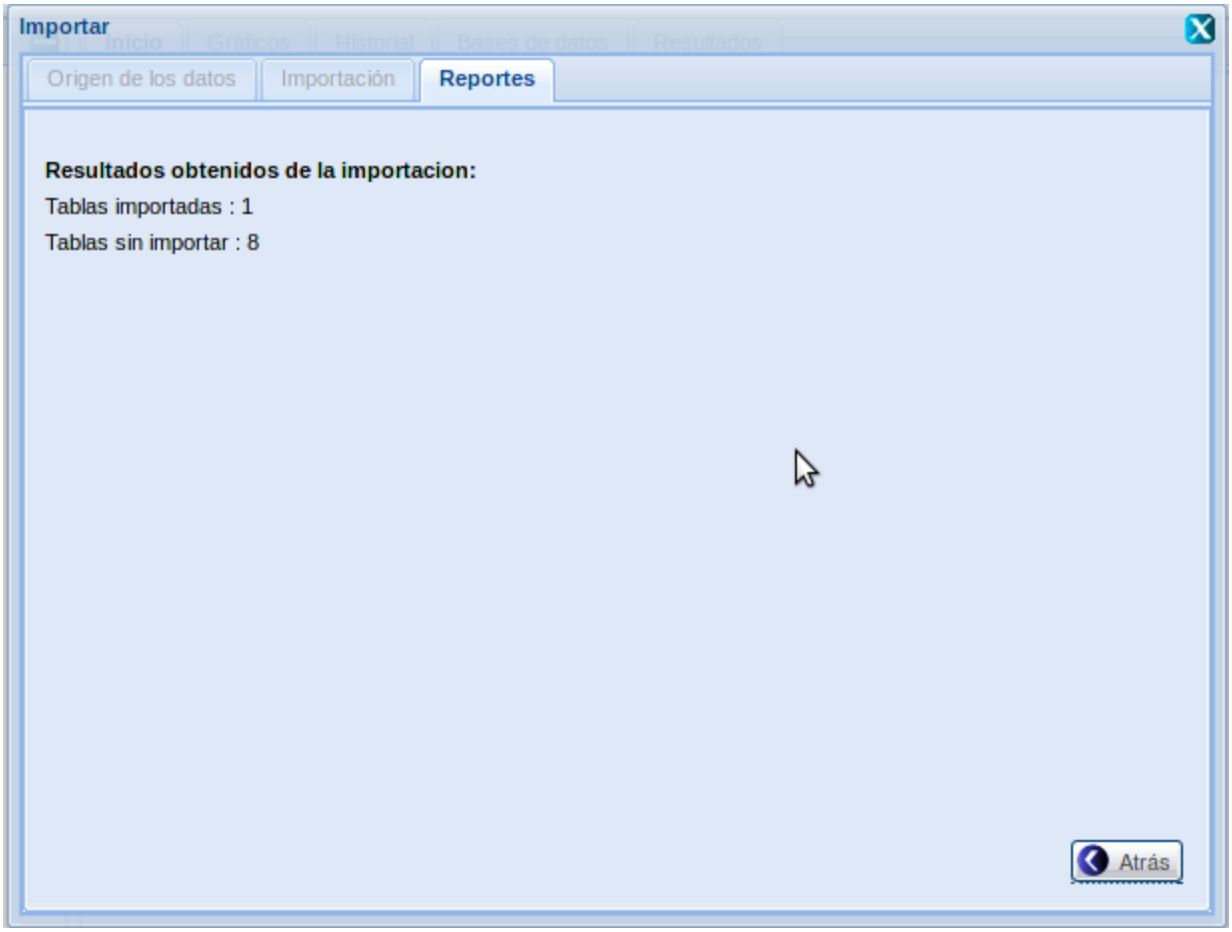
Descripción textual del requisito

Precondiciones	N/A
Flujo de eventos	
Flujo básico	
1	El sistema posibilita especificar los datos que permiten el acceso al contenedor de información que se importará. Los datos a especificar son: Dirección del servidor Gestor: (Mssql, Pgsql,Mysql,Oci,) Puerto Usuario Contraseña
2	Al presionar el botón **Conectar**, El sistema valida (ver validación 1) que los datos sean correctos.
3	Si los datos son correctos el sistema procede a realizar la conexión al servidor.
4	Si no existen fallas de conexión el sistema muestra un mensaje confirmación.
5	El sistema permite seleccionar el nombre de una base de datos para importar.
6	El usuario selecciona la base de datos que desea importar.
7	El sistema permite seleccionar los esquemas donde se encuentran las tablas que se van a importar.
8	El usuario selecciona el o los esquemas a importar.
9	El sistema muestra un listado con las tablas que contienen los esquemas seleccionados y permite seleccionar las tablas que se desean importar. (ver validación 2)
10	El usuario selecciona la o las tablas a importar y presiona la opción **Importar.**
11	El sistema valida (ver validaciones 3) la selección de tablas realizada.
12	Si la selección es correcta el sistema valida (ver validación 4) si ya existe una tabla con ese nombre.
13	Si no existe una tabla con el mismo nombre el sistema importa la tabla seleccionada.
14	El sistema muestra un mensaje confirmando la acción realizada.
15	Concluye el requisito.
Pos-condiciones	
1	Se ha importado el contenedor de información, listo para ser auditado.
2	Los datos importados se almacenan en una base de datos local.
Flujos alternativos	
Flujo alternativo 3.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 2 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 3.b Información incompleta	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 2 del flujo básico.

Pos-condiciones		
1	N/A	
Flujo alternativo 4.a No se puede acceder al servidor		
1	El sistema muestra un mensaje de error, informando un fallo de conexión.	
2	Volver al paso 1 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 6.a El usuario no selecciona una base de datos para realizar la importación		
1	El sistema muestra un mensaje de error y permite que el usuario seleccione la base de datos.	
2	El usuario selecciona la base de datos que desea importar.	
3	Volver al paso 7 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 12.a La tabla seleccionada no tiene registros		
1	El sistema muestra un mensaje de error informando que la tabla seleccionada no tiene registros y permite al usuario seleccionar una nueva tabla.	
2	Volver al paso 10 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 13.a Ya existe una tabla con el mismo nombre de la tabla seleccionada.		
1	El sistema permite al usuario reemplazar la tabla existente con la nueva tabla o crear una copia.	
2	Si el usuario selecciona reemplazar, se sobrescribe la tabla; y si selecciona hacer una copia el sistema genera una copia con un nombre aleatorio.	
3	Seguir al paso 14 del flujo básico.	
Pos-condiciones		
1	N/A	
Validaciones		
1	Si se selecciona Oci (Oracle) se debe activar el campo sid.	
2	Solo se puede seleccionar una tabla a la vez.	
3	La tabla seleccionada debe contener al menos un registro.	
4	No pueden existir dos tablas importadas con el mismo nombre.	
Relaciones	**Requisitos Incluidos**	N/A.
	Extensiones	N/A.
Conceptos	**Contenedor de información**	Visibles en la interfaz: Nombre Formato
Requisitos especiales	Se utilizará un asistente para la importación que permita en todo momento la navegabilidad en ambas direcciones durante la ejecución de la funcionalidad.	
Asuntos pendientes	N/A.	

Prototipo elemental de interfaz gráfica de usuario


Importar
Origen de los datos
Importación
Reportes
Esquemas
Base de datos
mysql
Estructura a importar
Dependencias
Importar
Tablas
Alias
Tuplas
Esquema: mysql
jos_banner
jos_banner
1
jos_bannerclient
jos_bannerclient
3
jos_bannertrack
jos_bannertrack
No tiene
jos_categories
jos_categories
26
jos_components
jos_components
33
jos_contact_details
jos_contact_details
1
jos_content
jos_content
9
jos_content_frontpage
jos_content_frontpage
1
jos_content_rating
jos_content_rating
1
Página 1 de 1
Atrás
Siguiente


Importar
Origen de los datos
Importación
Reportes
Resultados obtenidos de la importacion:
Tablas importadas : 1
Tablas sin importar : 8
Atrás

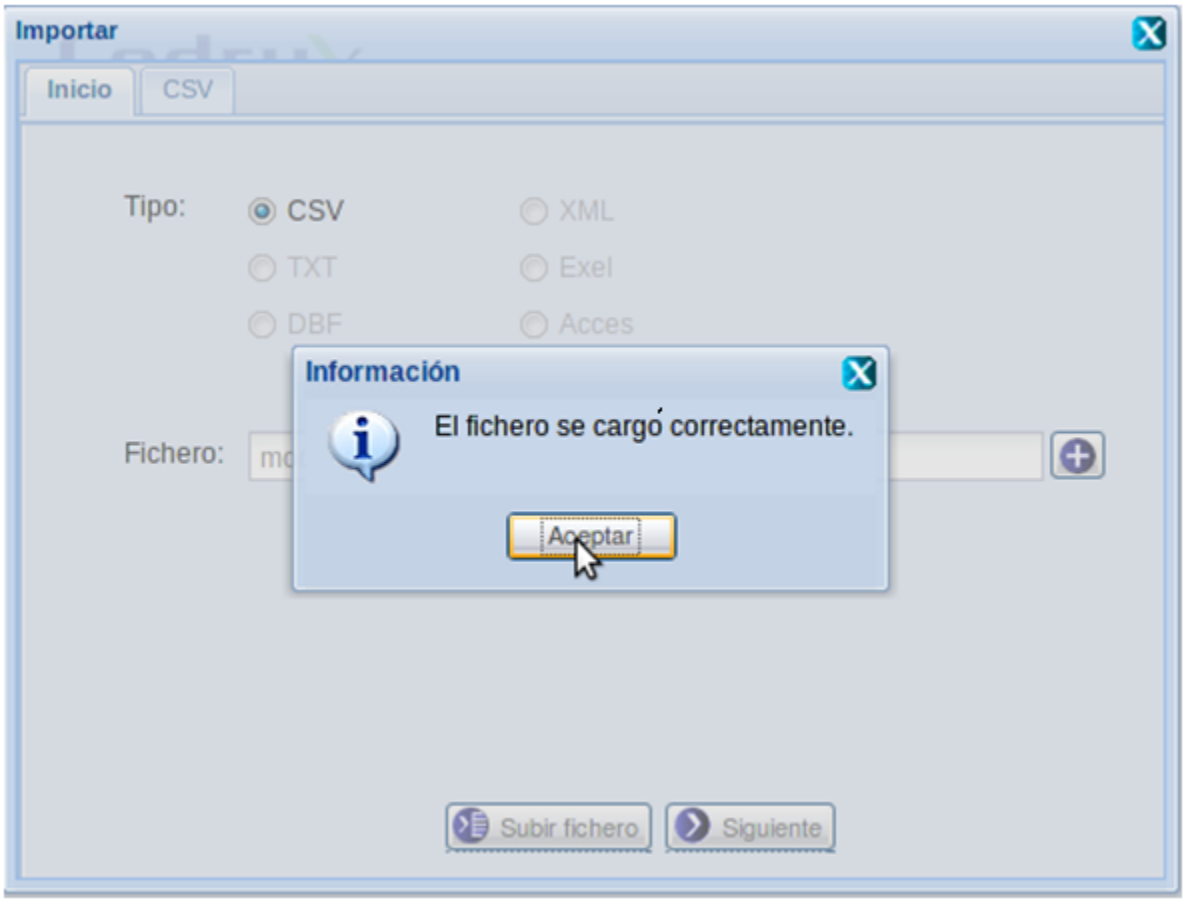
Anexo 4 Especificación de requisito: **Importar datos de ficheros CVS**

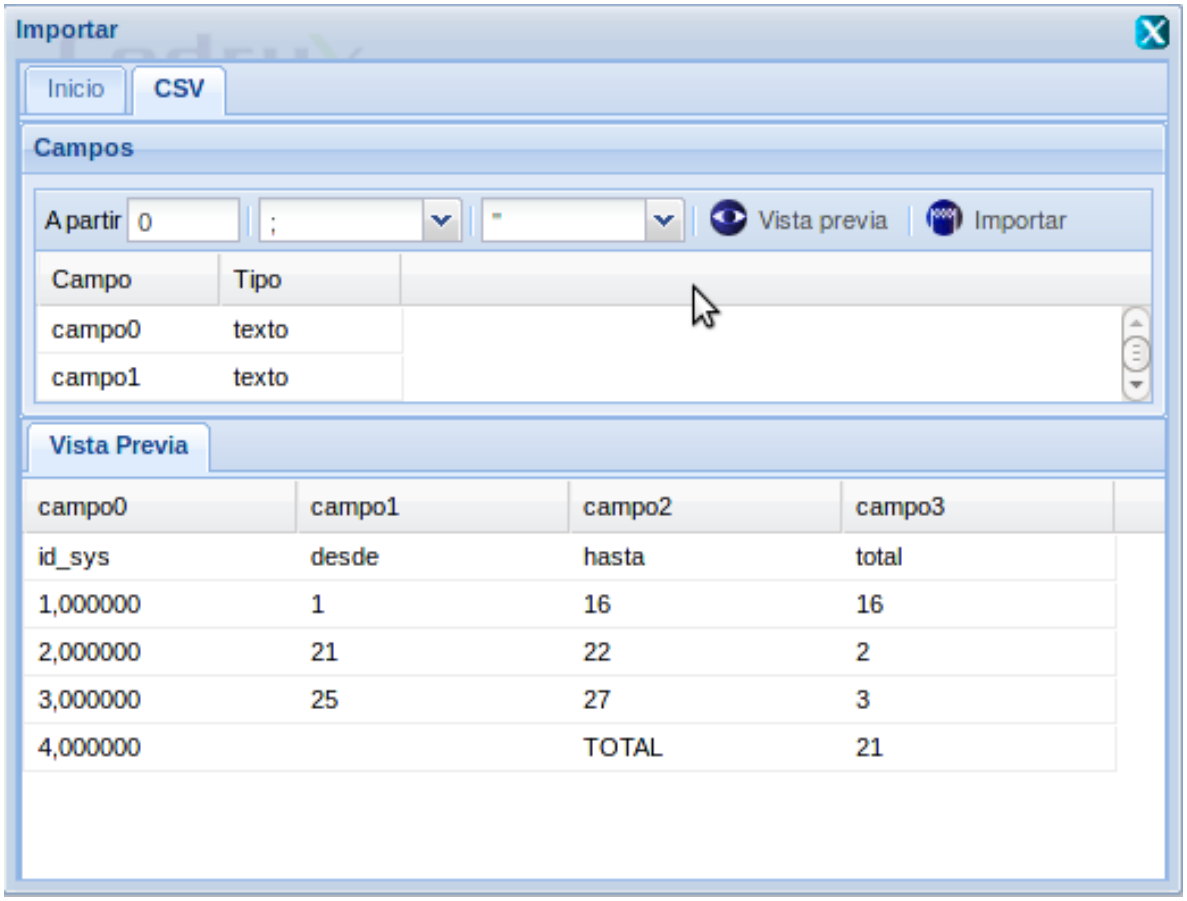
Descripción textual del requisito:

Precondiciones	N/A
Flujo de eventos	
Flujo básico	
4	El sistema posibilita seleccionar la extensión del fichero que se desea importar.
5	El sistema permite seleccionar la ubicación del fichero que se desea importar.
6	El usuario indica la ubicación del fichero y presiona el botón **Subir fichero.**
7	El sistema valida (ver validaciones 1 y 2) que se haya seleccionado una ruta válida y la extensión del fichero que se va a importar.
8	Si los datos son correctos el sistema sube el fichero seleccionado al servidor.
9	El usuario presiona el botón **Siguiente.**
10	El sistema permite configurar la los datos que se van a importar del fichero.
11	El usuario realiza la configuración de los datos que se van a importar del fichero y presiona el botón **Importar.**
12	El sistema valida (ver validación 3) que la configuración sea correcta.
13	Si la configuración es correcta el sistema importa los datos configurados y muestra un mensaje de confirmación de la acción.
14	Concluye el requisito.
Pos-condiciones	
1	Se ha importado el contenedor de información, listo para ser auditado.
2	Los datos importados se almacenan en una base de datos local.
Flujos alternativos	
Flujo alternativo 5.a La extensión del fichero seleccionado no es CSV	
3	El sistema muestra un mensaje informando que el fichero no es válido y permite seleccionar un nuevo fichero.
4	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 3.b No se selecciona una ubicación del fichero	
2	El sistema muestra un error indicando que debe especificar una ruta para el fichero que se va a importar y permite especificar la ruta.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
2	N/A
Flujo alternativo 10.a No se muestran datos en la configuración del fichero.	
4	El sistema muestra un mensaje de error, indicando que en la configuración realizada no hay datos para importar y permite realizar una nueva configuración.
5	Volver al paso 8 del flujo básico.
Pos-condiciones	
16	N/A
Validaciones	
1	Se validan los datos según lo establecido en el modelo conceptual CIG-N-AUD-i2101
2	La extensión del fichero seleccionado para importar tiene que ser CSV.

3	La configuración de un fichero para la importación debe contener al menos un registro de datos.	
Relaciones	**Requisitos Incluidos**	N/A.
	Extensiones	N/A.
Conceptos	**Fichero**	Visibles en la interfaz: Nombre Formato
Requisitos especiales	Se utilizará un asistente para la importación que permita la navegabilidad la ejecución de la funcionalidad.	
Asuntos pendientes	N/A.	

Prototipo elemental de interfaz gráfica de usuario




Importar
Inicio
CSV
Campos
A partir 0
;
"
Vista previa
Importar
Campo
Tipo
campo0
texto
campo1
texto
Vista Previa
campo0
campo1
campo2
campo3
id_sys
desde
hasta
total
1,000000
1
16
16
2,000000
21
22
2
3,000000
25
27
3
4,000000
TOTAL
21

Anexo 5. Algoritmo **Createtable**

```
public function Createtable($CAMPOS, $ALIAS, $ESQUEMA, $OPCION) {
    $temp = array();                                                                    //1
    if ($opcion == 'remplazar')                                                         //2
    {
        $this->Droptable($ALIAS);                                                       //3
        $this->DropSequence($ALIAS);                                                    //3
    }else if($opcion == 'copia')                                                        //4
    {
        $ALIAS = $this->NameMaker($ALIAS);                                              //5
    } else {                                                                            //6
      $name_Seq = $this->chema_dest . '.' . $ALIAS . '_seq';                            //7
      $seq = "                                                                          //7
        CREATE SEQUENCE $name_Seq
        INCREMENT 1
        MINVALUE 1
        MAXVALUE 9223372036854775807
        START 1
        CACHE 1;";
      $prepare = $this->cnx->prepare($seq, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));  //7
      if ($prepare->execute() == false)                                                 //8
        return false;                                                                   //9
      $id_sys = array(                                                                  //10
        'name' => 'id_sys',
        'type' => 'integer'
    );
    $temp['id_sys'] = $id_sys;                                                          //10
    foreach ($CAMPOS as $value) {                                                       //11
        $value['default'] = null;                                                       //12
        $temp['"' . $value['name'] . '"'] = $value;                                     //12
    }
    $temp_con = Doctrine_Manager::connection($this->cnx);                               //13
    $sql = $temp_con->export->createTableSql("$this->chema_dest.$ALIAS", $temp);        //13
    $sql = $sql[0];                                                                     //13
    $sql = str_replace('DEFAULT', "", $sql);                                            //13
    $sql = str_replace('NOT NULL', "", $sql);                                           //13
    $long = strlen($sql);                                                               //13
    $sql[$long - 1] = ',';                                                              //13
    $sql.= 'PRIMARY KEY("id_sys"))';                                                    //13
    $prepare = $this->cnx->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));  //13
```

```
    if ($prepare->execute() == false) {                                                    //14
        $model->DropSequence($ALIAS);                                                      //15
        return false;                                                                      //15
    }
    $alter = "ALTER TABLE $this->chema_dest.$ALIAS
              ALTER COLUMN id_sys SET DEFAULT nextval('$name_Seq'::regclass);";           //16
    $prepare = $this->cnx->prepare($alter, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY)); //16
    if ($prepare->execute() == false)                                                      //17
        return false;                                                                      //18
     return $ALIAS;                                                                        //19
    }
}
```

1