Universidad de las Ciencias Informáticas

Facultad 4

Título: Sistema de Log de Auditoría

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

Autores:

Daylin Real Madrigal

Reinaldo Alain Hernández Valdés

Tutor: Ing. Saumel Tejeda Díaz.

Co-Tutor: Ing. Yuniel Eliades Proenza Arias

La Habana, julio x, 2008

Año 50 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al MINFAR y a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Reinaldo Alain Hernández Valdés

_____

Daylin Real Madrigal

_____

Saumel Tejeda.

_____

"Si avanzo, seguidme, si me detengo, empujadme, si retrocedo, matadme".
Ernesto Che Guevara

Datos de Contacto

Tutor: Ing. Saumel Tejeda Díaz.

Graduado en el 2005 en la Carrera de Ingeniería Informática en el Instituto Técnico Militar José Martí en Ciudad de la Habana. Se ha desempeñado como Automatizador en varias unidades y ha participado en varios proyectos. Actualmente cursa la maestría en Gestión de Proyectos Informáticos y es el Jefe de la Línea de Planificación en el Centro de Compatibilización, integración y desarrollo de productos informáticos para la defensa (UCID)[1].

Co-Tutor: Ing. Yuniel Eliades Proenza Arias

Graduado en el 2006 de la carrera de de Ingeniería Informática en la Universidad de Holguín. Profesor de la Universidad de las Ciencias Informáticas. Opta por la Categoría Docente de Instructor. Se ha desempeñado como profesor en las disciplinas de Ingeniería y Gestión de Software y Técnicas de Programación. Actualmente cursa la maestría de Gestión de Proyectos Informáticos.

Agradecimientos

Daylin

Gracias a este proyecto de la Revolución por permitirme ser hoy una profesional.
Gracias a Fidel Castro Ruz, nuestro Comandante en Jefe por esta bella idea de hacer nuestra UCI.
Gracias a mis amigos por darme siempre el ánimo y apoyo que necesitaba en cada momento y enseñarme muchas de las cosas que hoy sé.
Gracias a todos los que me ayudaron con su apoyo incondicional a ampliar mis conocimientos y estar más cerca de mis metas profesionales.
Gracias a mi familia, en especial a mi mami por estar conmigo en los momentos más difíciles de la carrera.
Gracias a todos por atreverse a confiar en mí.

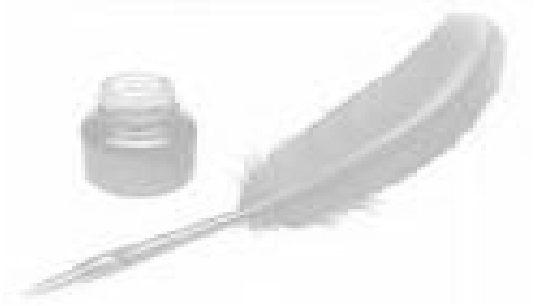
Reinaldo

Muchas gracias a nuestro Comandante en Jefe por la ingeniosa idea de construir esta maravillosa escuela en medio de esta bella de ideas.
Muchas gracias a Raúl Castro Ruz por permitirnos estudiar como cadetes insertados en la UCI.
Muchas gracias a mis padres por haberme guiado por el camino correcto y enseñarme desde pequeño a elegir por mi mismo, por requerirme cuando lo necesitaba y alentarme en los momentos difíciles,
por confiar en mi aún cuando yo mismo no estaba muy seguro, por todo el cariño y comprensión que me han dado de por vida. Muchas gracias a todos los profesores y profesoras que me han enseñado a lo largo de mi vida estudiantil.
Muchas gracias a todas mis amistades por haberme ayudado en los momentos difíciles y que de una forma u otra influyeron sobre mí para poder llegar hoy hasta aquí.
Muchas gracias a todos los que me ayudaron en la tesis. Y muchas gracias a mis novias que también hicieron posible este sueño.
De seguro no logré ser todo lo bueno que han soñado pero he sido lo mejor que he podido.
Gracias a todos por estar a mi lado.

Dedicatoria

Daylin

Dedico este triunfo de mi vida a todas las personas que quiero: a mi hermanita Michi, a mis abus Tatín y Luis, a mi tío Oscar y a papi Ñico por ser como unos padre para mí, a mis amigas y amigos de la UCI por ser mi familia durante estos 5 años, a todas las amistades que he conocido a lo largo de mi carrera estudiantil y en especial a la mujer más maravillosa del mundo, la base de mi vida, mi mejor amiga, a ti mamita.



Reinaldo

Le dedico este título a mi padre por haber sido siempre mi ídolo y mi guía, por enseñarme todo lo que se hoy de esta vida, por mostrarme el camino correcto en el momento adecuado y con las palabras justas, por aceptarme como soy y enseñarme a ser mejor. A mi madre por todo su apoyo y dedicación en la vida de este hijo, por todas las noches de desvelo a mi lado cuando estaba enfermo y por esos besos que tanto extraño cuando no estoy a su lado. A mi hermanita por su firmeza y su aventón cuando casi renuncio a esta batalla. A toda mi familia por quererme y apoyarme en cada paso de mi carrera. A mis profesores. Y a todos los que quieran graduarse de alguna especialidad sin importar el titulo de lo que sea que tengan siempre presente que:

“En el difícil arte de ascender la montaña es la voluntad quien determina llegar a la cumbre”. Mi mamá.

Resumen

El Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR)[2] está estructurado para su funcionamiento interno por departamentos que abarcan todas las esferas del ciclo logístico empresarial. Dentro de estos departamentos se encuentra el de Planificación Material y Financiera, en el cual se efectúan un sin número de procesos y se trabaja con grandes cantidades de información. Por esto, es de vital importancia lograr un control y una supervisión en las operaciones que se realizan sobre los procesos fundamentales. El siguiente trabajo tiene como propósito explicar el desarrollo de una aplicación de gestión Web[3], la cual debe elevar el nivel de informatización en el proceso de Auditoría mediante el Sistema de Log de Auditoría (SILOGA)[4] en la gestión de la información que se maneja en el departamento de Planificación del MINFAR. El desarrollo de este sistema se sostiene de la aplicación de metodologías y el uso de herramientas actuales que aseguran que su resultado sea un producto de software confiable, con un alto grado de calidad.

PALABRAS CLAVE

(MINFAR, Planificación, Material, Financiera, procesos, control, supervisión, SILOGA, sistema, herramientas, metodologías, software)

Índice

Índice de Tablas

Índice de Figuras

Introducción

El MINFAR en nuestro país realiza un proceso de previsión, estructuración y asignación de recursos a los diferentes organismos vinculados al mismo, para lograr una organización y optimización de los recursos materiales y financieros con el fin de cumplir los objetivos de trabajo y las metas trazadas en un tiempo determinado, proceso al que define como Planificación Material y Financiera.

Esta Planificación se inicia en el órgano Consumidor (OC)[5] el mismo confecciona su Plan de Demandas y lo envía a los respectivos Centros de Balance (CB)[6]. Una vez que el CB tenga en su poder el Plan de Demandas, balancea lo demandado por el OC y le agrega o retira elementos justificando siempre las razones. Finalmente el Plan es enviado a la Dirección de Economía (DE)[7] para su aprobación, consolidación y presentación a los Organismos de la Administración Central del Estado (OACE)[8] y al Ministerio de Economía y Planificación (MEP)[9].

Este trabajo requiere de una mayor seguridad para lograr el control sobre los registros que se llevan a cabo en el proceso de la Planificación dando lugar a la siguiente **situación problémica**: la gran cantidad de datos que genera el proceso de Planificación y la gran variedad de accesos a la información son muy difíciles de controlar, supervisar y salvaguardar.

Por todo lo antes expuesto nos surge el siguiente **problema** a resolver ¿Cómo garantizar el control sobre los procesos en el Sistema de Planificación?

El **objeto de estudio** lo constituye el Sistema de Planificación Material y Financiera del MINFAR.

Se deriva por tanto que el **campo de acción** se enmarca en los procesos de Registro de Auditoría del Sistema de Planificación Material y Financiera del MINFAR.

Para resolver el problema planteado se propone como **objetivo general:** Analizar, Diseñar e Implementar un sistema informático para el proceso de registro de Auditoría sobre el Sistema de Planificación Material y Financiera del MINFAR.

Para guiar la investigación se plantea la siguiente **hipótesis**: Si se cuenta con una aplicación para el proceso de Registro de Auditoría en el Sistema de Planificación entonces se logrará obtener un control y supervisión de la información.

Para dar cumplimiento al objetivo anteriormente planteado y resolver la situación problémica planteada se definen las siguientes **tareas**:

1- Identificar los procesos de Planificación Material y Financiera del MINFAR para (programar, Modelar) las funcionalidades principales del sistema.

2- Realizar un estudio general de los procesos de registro de Auditoría a un Sistema de Información para obtener el estado del arte.

3- Investigar las tecnologías y tendencias actuales a utilizar en el desarrollo de la aplicación para la selección de las herramientas a usar.

4- Desarrollar el análisis del SILOGA.

5- Desarrollar el diseño del SILOGA.

6- Implementar la aplicación para lograr el Sistema de Log de Auditoría.

En el Departamento de Planificación Material y Financiera se realizan un sin número de procesos y se trabaja con grandes cantidades de información, debido a ello la realización de este trabajo es importante pues con el lograremos tener un control y una supervisión en las operaciones que se realizan sobre los procesos fundamentales.

La investigación está estructurada por los siguientes capítulos:

-Capítulo 1: Fundamentación teórica

Se hace un estudio del arte sobre el objeto de estudio y campo de acción de la propuesta, evidenciando la importancia del desarrollo del sistema y la selección de la plataforma, el lenguaje, las herramientas y la metodología con las que se llevará a cabo la tarea.

-Capítulo 2: Descripción de la Propuesta Solución

Se describen las características del sistema y los requisitos con los que debe cumplir, obteniendo como artefacto fundamental su diagrama de caso de usos (CUs)[10].

-Capítulo 3: Análisis y Diseño del sistema

Se comienza la construcción de la propuesta, determinando las clases y la base de datos (BD)[11] que soportarán las funcionalidades del mismo.

-Capítulo 4: Implementación del sistema

Se concluye la construcción de la propuesta, determinando cómo se distribuirá el sistema en términos de hardware y los componentes que serán necesarios implementar.

Capítulo 1

Fundamentación teórica

Introducción

En el presente capítulo se ofrece una visión de los principales aspectos de la Auditoría Informática, los Log de Auditoría, los *Enterprise Resource Planning* (ERP)[12], así como algunas características del Sistema de Planificación Material y Financiera del MINFAR, además los principales problemas que afectan el campo de acción de este trabajo y la incidencia que puede tener la propuesta de solución. También se abordan importantes conceptos que son necesarios conocer para la creación de una aplicación de gestión Web y así entender correctamente en qué consiste. Se hace referencia a las principales características y funcionalidades de algunas tecnologías empleadas en la construcción de la misma.

1.1. Definición de auditoría informática

Es el proceso de recoger, agrupar y evaluar evidencias para determinar si un Sistema de Información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización y utiliza eficientemente los recursos.(Sanger, 2008)

La palabra auditoría viene del latín “auditorius” y de esta proviene auditor, que tiene la virtud de oír y revisar cuentas, pero debe estar encaminado a un objetivo específico que es el de evaluar la eficiencia y eficacia con que se está operando para que, por medio del señalamiento de cursos alternativos de acción, se tomen decisiones que permitan corregir los errores, en caso de que existan, o bien mejorar la forma de actuación. (Wikilearning, 2007)

1.1.1. Definición de auditoría de los Sistemas de información

Se define como cualquier auditoría que abarca la revisión y evaluación de todos los aspectos (o de cualquier porción de ellos) de los sistemas automáticos de procesamiento de la información, incluidos los procedimientos no automáticos relacionados con ellos y las interfaces correspondientes.

Se encarga de llevar a cabo la evaluación de normas, controles, técnicas y procedimientos que se tienen establecidos en una empresa para lograr confiabilidad, oportunidad, seguridad y confidencialidad de la información que se procesa a través

de los sistemas de información. La auditoría de sistemas es una rama especializada de la auditoría que promueve y aplica conceptos de auditoría en el área de sistemas de información. (France Telecom España, S.A, 1999).

1.1.1.1. Tipos de auditoría de los Sistemas de información

Existen 3 tipos de auditoría, auditoría a nivel de aplicación, auditoría a nivel de base de datos y auditoría a nivel de servidor.

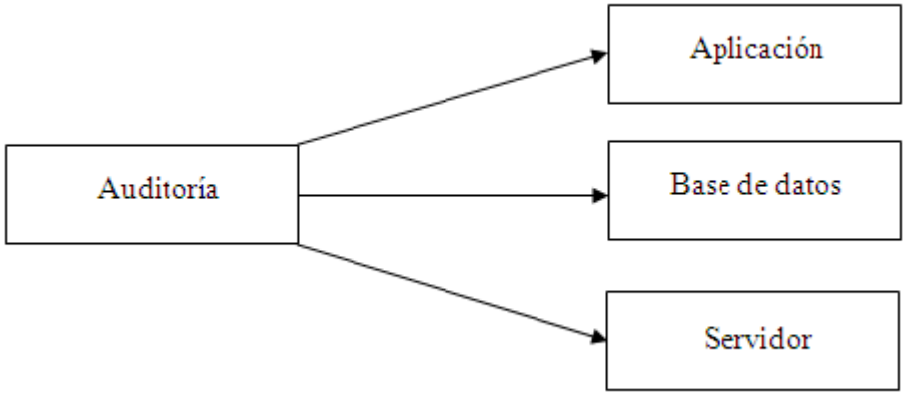


Figura. 1 Tipos de auditoría

1.2. Definición de un log.

Es un registro oficial de eventos durante un período de tiempo en particular. Para los profesionales en seguridad informática, un log es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (*who, what, when, where y why*) un evento ocurre para un dispositivo en particular o aplicación. (Wikimedia Foundation, Inc., 2008)

1.2.1. Definición de un log de Auditoría

Son los ficheros informáticos conteniendo detalles de cambios o alteraciones a registros de datos, que pueden usarse en el caso de que se precise la recuperación del sistema. Activar esta capacidad normalmente incurre en cierta carga añadida al sistema, pero permite la revisión de toda o parte de la actividad producida por o en el sistema. (Application LifeCycle Solutions, S.L., 2006)

1.3. Ejemplos de sistemas de registro

Para sustentar el desarrollo de este trabajo se realizó el estudio de algunos software a nivel Internacional y Nacional relacionado con los Sistemas de registro con el objetivo de encontrar alguno que diera solución al problema existente. Como resultado se obtuvo:

- ***Cyberprinter*:** El software que le ayudará a llevar registro de todas las impresiones en su empresa. Registra datos como nombre de documento, número de páginas impresas, copias, nombre de la impresora, fecha, hora, nombre de la PC, nombre de usuario, color de la impresión, tamaño de la hoja. (Serpol Software, 2002)
- ***Weblog Expert:*** Es un analizador de *logs* de servidor Web multi-característico. Te dará información de los visitantes en tu sitio, estadística de actividad, de acceso a archivos, de recorridos dentro del sitio, información de páginas de referencia, de buscadores Web, de navegadores, sistemas operativos, errores y más. Filtros flexibles que le ayudaran a realizar investigaciones completas Otras características incluyen la búsqueda multi-canal de DNS, programador incluido y mapeo IP. (Free Download Manage, 2008)
- **El registro del sistema, o registro de Windows**: Es una base de datos que almacena las configuraciones y opciones del sistema operativo *Microsoft Windows* en sus versiones de 32 bits, 64 bits y *Windows Mobile*. Contiene información y configuraciones de todo el *hardware*, *software*, usuarios, y preferencias del PC. Si un usuario hace cambios en las configuraciones del "Panel de control", en las asociaciones de ficheros, en las políticas del sistema o en el software instalado, los cambios son reflejados y almacenados en el registro.(Wikimedia Foundation, Inc., 2008)

Después de terminada la búsqueda de estos sistemas y ratificar que no dan solución al problema planteado anteriormente se decidió desarrollar el SILOGA.

1.4. Desarrollo histórico de la auditoría en el mundo

La contaduría publica como auditoría o revisión de cuentas tiene algunos antecedentes muy remotos, como el caso descrito en un papiro de Zenón, que refiere que en el año 254 (a.n.e) Apolonios, Ministro de finanzas del rey Filadelfo, de la dinastía de los Ptolomeos de Egipto, contraído por haberse pagado de su caja siete talentos de plata sin su autorización, ordeno fuesen comprobadas las cuentas de Aristeos, uno de los tesoreros y las del mayordomo Artemidoro esta orden la hizo extensiva a Zenón, administrador de todos sus intereses y jefe de Contabilidad para que preparase sus cuentas para ser inspeccionadas por Pythen, banquero del estado, a quien deberían entregar sus fondos que tuviesen en su poder y le serian devueltos más tarde.

En la Europa Feudal esta profesión comenzó a precisarse más, llegando a identificarse las funciones con el cargo y así nació el auditor.

El nombre del ´auditor´ debe su origen a la forma en que se recibían las liquidaciones de las cuentas.

De esta época existen algunos antecedentes, principalmente en Inglaterra de los siglos XIII y XIV que permiten establecer las causas que dieron origen a esta profesión, principalmente las siguientes:

1. La necesidad de comprobar la honestidad de aquellos que administraban los bienes y dinero de otros.
2. El deseo de los administradores de que su honradez quedase comprobada.
3. La falta de conocimientos en realidad, para rendir informes y cuentas de la gestión realizada.

A partir del siglo XVII, el feudalismo se debilitaba. Se desarrollaba la clase burguesa controlando la banca, el seguro, el tráfico marítimo, los mercados y la incipiente industria contraponiendo su poderío económico a la Hegemonía feudal terminando por derrotar al feudalismo. Comienza así una era de gran desarrollo en las actividades comerciales e industriales.

Entre las nuevas actividades que surgen encontramos la Contabilidad Pública. Según los antecedentes comúnmente aceptados por todos los tratistas de la contabilidad. George Watson fue el primer contador que ofreció al público sus servicios como auditor en el año 1645 en Escocia. Durante muchos años Watson desempeño cargos de tesorero, cajero y contador del banco de Escocia.

Es indudable que el desarrollo de la contaduría pública en el mundo, principalmente en Inglaterra tuvo una gran importancia las convulsiones económicas y financieras experimentadas por la humanidad en el siglo XIX.

En el año 1799 había varias firmas de contadores públicos ejerciendo en Inglaterra, lo que más tarde dio lugar a la creación de varias asociaciones de la nueva profesión, siendo la primera la formada en Escocia en el año 1854. En 1880 se organizó la de contadores certificado de Inglaterra y Gales. En 1885 se fundó la de contadores incorporados y auditores de Inglaterra. En 1896 se fundó la Asociación de contadores públicos de Estados Unidos.

En el año 1916 se comenzó la preparación de un programa mínimo de procedimientos a seguir en las auditorías quedando establecidas las primeras reglas que rigieron la contaduría pública.

Parejamente al desarrollo de las grandes Empresas se desarrolló la Contabilidad, haciéndose más profunda, más analista. Creándose especialidades, mecanizándose los sistemas contables, facilitando a las auditorías el mejoramiento de los métodos y procedimientos, a establecer reglas y principios, y a mantener una constante superación para no estancarse. (Monografias.com S.A. , 1997)

1.5. Definición de sistemas de planificación de recursos empresariales (ERP)

Son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

Por las ventajas que proporcionan y las necesidades que satisfacen los ERP, el MINFAR ha decidido implementar un Sistema Empresarial para lograr la integridad de todas sus áreas de trabajo y facilitar la toma de decisiones de los directivos; pues el ERP gestiona de manera integrada y eficiente la información, comunicando las diferentes áreas del negocio mediante procesos electrónicos. Una de esas áreas es la de Planificación de los Recursos Materiales y Financieros.

1.6. Solución a través de una aplicación de gestión Web

Una aplicación Web es aquella que los usuarios usan accediendo a un servidor Web a través de la red (**Internet** o **intranet)**. Las aplicaciones Web son populares debido a la practicidad del navegador como cliente ligero. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.

En una aplicación Web la navegación y la entrada de datos por parte de un usuario, afectan el estado de la **lógica del negocio**; empleando para ello tecnologías que generan contenidos dinámicos. Si no existe lógica del negocio en el servidor, el sistema no es considerado aplicación Web.

Las interfaces de las páginas Web poseen ciertas limitantes en la funcionalidad del cliente. Métodos comunes en las aplicaciones de escritorio como dibujar en la pantalla, arrastrar o soltar no están soportados por las tecnologías Web estándar. Los desarrolladores utilizan generalmente lenguajes interpretados que se ejecutan en el cliente Web para obtener una mayor funcionalidad, así como tecnologías que se ejecutan en el servidor para no tener que recargar la página en su totalidad, algo que molesta mucho a los usuarios. Se han desarrollado técnicas para coordinar estos lenguajes con tecnologías del lado del servidor, como por ejemplo *Personal Home Page Tools* (PHP)[13] y **A**synchronous **J**avaScript ***A**nd **X**ML* (AJAX)[14] que es una

técnica de desarrollo Web que usa una combinación de varias tecnologías como se verá más adelante.

1.6.1. Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es una forma de dividir las responsabilidades de un Sistema de Información separando la interfaz de usuario (Nivel de presentación) de la gestión de la información (Nivel de gestión de datos).

Esta arquitectura consiste básicamente en que un programa, el Cliente informático realiza peticiones a otro programa, el servidor, que les da respuesta.

Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema multiusuario distribuido a través de una red de computadoras.

1.7. Técnica y tecnologías del lado del cliente

Un cliente Web (navegador o browser) es una aplicación de software que permite al usuario recuperar y visualizar documentos de hipertexto. La parte cliente de las aplicaciones Web está formada por el código HTML que forma la página Web, con opción a código ejecutable mediante los lenguajes de scripting de los navegadores (JavaScript, Vbscript, etc.) o mediante pequeños programas en Java (applets). La programación del lado del cliente tiene como principal ventaja que la ejecución de la aplicación se delega al cliente, con lo cual se evita recargar al servidor de trabajo. Entre los navegadores más populares están, el Netscape, Internet Explorer, Mozilla. Los lenguajes más comunes que se ejecutan en el cliente son: *VBScript* y JavaScript**.**

1.7.1. Script del lado del cliente

Una práctica muy difundida en el mundo de la programación Web es emplear **lenguajes interpretados** del lado del cliente para añadir más funcionalidad a las aplicaciones, especialmente para crear una experiencia interactiva que no requiera recargar la página cada vez.

Javascript

Es un lenguaje interpretado, con una sintaxis semejante a la de los lenguajes Java y C. Su diseño no le permite considerarse un lenguaje puramente orientado a objetos como es el caso de Java. Fue desarrollado por la empresa *Netscape Communications.* Es muy utilizado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador Web así como para validar datos de entrada en las interfaces de las aplicaciones.

¿Por qué *JavaScript*?

Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Con *Javascript* se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones *Javascript* y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Para interactuar con una página Web se provee al lenguaje *JavaScript* de una implementación del Modelo de Objetos de Documento (DOM)[15], es una interfaz de programación de aplicaciones (API)[16] para documentos HTML[17] (paginas Web, *HyperText Markup Language* -Lenguaje de Marcado de Hipertexto) y XML[18] (*Extensible Markup Language*). Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento. En la especificación del DOM, el término "documento" se utiliza en un sentido amplio ya que es el documento el contenedor que soporta los demás elementos. A través del DOM los programadores pueden construir documentos, navegar por su estructura, añadir, modificar o eliminar elementos y contenido. Se puede acceder a cualquier elemento que se encuentre en un documento HTML o XML, y se puede modificar, eliminar o añadir usando DOM, salvo algunas excepciones.

ExtJS es una librería Javascript que hace de puente a las librerías de Yahoo!, jQuery y Prototype+Scriptaculous para ofrecernos de forma sencilla componentes *Graphical User Interface* (GUI)[19] en nuestras aplicaciones cliente. Entre los componentes que nos ofrece encontramos diálogos, menús, tablas, layouts, paneles, pestañas y mucho más. (Ext, LLC, 2006)

1.7.2. AJAX: Una técnica de desarrollo Web

JavaScript y XML Asíncronos, AJAX es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- XHTML (o HTML) y CSS[20] para el diseño que acompaña a la información.

- DOM accedido con un lenguaje de scripting por parte del usuario, como *JavaScript* y *JScript*, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto *XMLHttpRequest* para intercambiar datos asincrónicamente con el servidor Web.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano.

Propiedades.

- Peticiones y respuestas HTTP[21].

- Código de lado del cliente usando JavaScript.

- Programación de lado del servidor en PHP.

- Transferir y procesar datos mediante el uso de XML.

Libertad del refresco de página:

Las aplicaciones Ajax no necesitan refrescar la página completa para actualizar la información, pueden simplemente actualizar parte de la página en cualquier momento, dándoles a los usuarios una respuesta instantánea a sus ingresos y consultas. Esto les permite a los usuarios ver continuamente con lo que están trabajando y reaccionar a cualquier cambio, error, o actualización que la interfaz les notifique.

Chequeo y guardado instantáneo del campo:

En la Web, es fácil chequear los campos en el lado del cliente utilizando JavaScript. Esto produce un efecto inmediato, y replica el comportamiento de una aplicación de escritorio. Sin embargo, por razones de seguridad, es necesario chequear todos los campos también del lado del servidor. Afortunadamente, Ajax permite también que esto suceda.

Interfaz de pantalla única:

Una de las mayores razones para utilizar el abordaje basado en pantallas es la simplicidad de una interfaz de pantalla única. Una interfaz de pantalla única es más útil, brindando varias ventajas sobre las aplicaciones basadas en páginas.

Una de las ventajas de pantalla única es que el usuario puede ver el cuadro general de la aplicación, viendo todos los pasos necesarios para completar la aplicación. Esto le da al usuario una clara idea de lo que se espera de ellos durante una transacción.

La interfaz de pantalla única también le permite al usuario modificar y cambiar información en el orden que él elija.

Relativamente fácil de implementar:

Debido a que las aplicaciones Ajax están construidas utilizando nada más que estándares Web actuales, ellas son relativamente sencillas de construir.

Muchos diseñadores Web familiarizados con la construcción de aplicaciones basadas en páginas pueden migrar de una interfaz a Ajax más bien rápidamente.

También, desarrolladores emprendedores de Ajax crearon bloques de construcción fáciles de usar que les permiten a aquellos desarrolladores que no estén familiarizados con el abordaje migrar sus aplicaciones sin tener que escribir código desde cero.

Ajax: una buena alternativa:

Combinando la sofisticación de aplicaciones basadas en pantallas con la relativamente fácil de implementar de aplicaciones basadas en página, Ajax es una alternativa sólida para un nuevo desarrollo de interfaz.

A pesar de que nada construido con Ajax va a ser amigable para el usuario desde el comienzo, con cuidado las interfaces pueden sacar provecho de lo que se sabe y gusta de las aplicaciones de escritorio, y al mismo tiempo todavía sentir que se está usando la maravillosa Web.

1.8. Extensible Markup Language

XML es un metalenguaje, es decir, un lenguaje usado para hacer referencia a otros lenguajes que ofrece un formato para la descripción de datos estructurados. Esto facilita unas declaraciones de contenido más precisas y unos resultados de búsquedas más significativos en varias plataformas. Además, XML habilita una nueva generación de aplicaciones para ver y manipular datos basadas en la Web.

Se trata de una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

1.9. Tecnologías del lado del servidor

La parte del servidor está formada por un programa o script que es ejecutado por el servidor Web, y cuya salida se envía al navegador del cliente. Los principales

beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Programar del lado del servidor tiene como gran ventaja que cualquier acción puede ejecutarse sin tener en cuenta el tipo de cliente, ya que la aplicación se ejecuta en el servidor que es un ambiente controlado. Una vez ejecutada la aplicación, el resultado que se envía al cliente.

1.9.1. Módulo de los servidores Web

La tecnología más reciente para la ejecución de aplicaciones Web consiste en anexar a un servidor Web "módulos" que permiten al servidor interpretar un determinado lenguaje. De esta forma se logra eficiencia ya que el servidor no necesita crear un nuevo proceso por cada aplicación que ejecuta. Las aplicaciones son portables ya que son desarrolladas en un lenguaje estándar que no depende del servidor, y confiables ya que si bien pueden producir un error en el lenguaje en que están diseñadas si el módulo es sólido dichos errores no pueden comprometer al servidor Web. Por la potencialidad, rendimiento y seguridad que ofrece esta tecnología será la adoptada para el desarrollo de la aplicación que se propone como objetivo principal este trabajo.

1.9.2. Script del lado del servidor

Los lenguajes de scripts que se ejecutan en el servidor Web son interpretados por alguna aplicación alojada en dicho servidor y que se envían al cliente en un formato comprensible para él, son independientes del cliente por lo que son mucho más flexibles en relación al cambio de un navegador a otro o con respecto a las versiones del mismo. El cliente solo verá el código HTML terminado.

PHP

Es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. PHP es un acrónimo recurrente que significa "**P**HP ***H****ypertext* **P**re-*processor*" (inicialmente PHP Tools, o, *Personal Home Page Tools*), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web.

¿Por qué PHP?

Para el desarrollo de la aplicación Web se debe tener en cuenta cuestiones esenciales como: garantizar que el sistema sea multiplataforma y que pueda ser desarrollado con software libre por las características de nuestro país estar bloqueado y de las restricciones de las principales compañías de desarrollo informático. Analizando esto el más adecuado es el PHP por ser libre, multiplataforma, su flexibilidad de

comunicación con los principales gestores de bases de datos y por su potencialidad en funcionalidades y rapidez.

Dentro de las principales ventajas de PHP se encuentran:

- Es un lenguaje de programación multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL, PostgreSQL (soluciones libres).
- Lee y manipula datos desde diversas fuentes.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (extensiones).
- Se desarrolla activamente, cuenta con una amplia documentación en su página oficial.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos (POO)[22].
- Permite crear los formularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

1.10. Gestión de Datos

Un Sistema Gestor de Bases de Datos (SGBD)[23] o Sistema Gestor de Base de Datos Relacionales (RDBMS)[24] es un conjunto de programas que permite a los usuarios crear y mantener una BD, por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones así como la administración necesaria para mantenerlas operativas, mantener su integridad, consistencia, confidencialidad y seguridad.

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de información.

1.10.1. Lenguaje de Consulta Estructurado (*Structured Query Language*, SQL)[25]

Es un lenguaje de alto nivel, declarativo, optimizado y relacional ya que se emplea para manipular datos en BD relacionales, que permite realizar diversas operaciones sobre estas. En la actualidad el SQL se ha convertido en el estándar de la mayoría de

los SGBD. Sus funcionalidades van más allá de ser un simple lenguaje de consultas a BD, estas comprenden además: lenguaje definición de datos (DDL)[26], lenguaje de definición de vistas (VDL)[27], y lenguaje de manipulación de datos (DML)[28]. Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas.

El SQL trabaja con estructura cliente/servidor sobre una red de ordenadores, donde el ordenador cliente es el que inicia la consulta y el ordenador servidor es quien la atiende.

El SQL permite:

- Definir una BD mediante tablas.
- Almacenar información en tablas.
- Seleccionar la información que sea necesaria de la base de datos.
- Realizar cambios en la información y estructura de los datos.
- Combinar y calcular datos para conseguir la información necesaria.

1.10.2. Sistema Gestor de Base de Datos

¿Por qué PostgreSQL?

Porque es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle o DB2.

Soporta tanto la POO como la funcional. Las funciones tienen la particularidad de poder ejecutarse con los privilegios del usuario ejecutor o con los privilegios de un usuario definido previamente lo que supone un alto nivel de seguridad. Estas funciones son referidas en otros SGBD como procedimientos almacenados (stored procedures). PostgreSQL tiene la extraordinaria potencialidad de permitir que mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos esto es posible gracias a un sistema denominado **A**cceso **C**oncurrente **M**ultiversión (**MVCC** en ingles)[29]. Este gestor se identifica además por:

- Disparadores (triggers)[30].
- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

1.11. Proceso de Desarrollo de Software

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. Este es el encargado de guiar el desarrollo por cada uno de los pasos que garanticen un producto final de calidad. Debe brindar la posibilidad al equipo de desarrollo de documentar lo aspectos esenciales para en un futuro dar soporte al software.

1.11.1. Metodologías de desarrollo de software

Un proceso de software detallado y completo suele denominarse "Metodología". Sin embargo existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Vale destacar que todo desarrollo de software es riesgoso y difícil de controlar, por lo que es necesario aplicar una metodología que permita obtener como resultado un producto de calidad. Una definición acertada, en correspondencia con el objetivo de la investigación es la abordada por *Maddison* donde define metodología como "Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de Sistemas de Información".

Las metodologías guían el proceso de desarrollo y la clave del éxito de un proyecto de software es la elección correcta de la metodología, que puede conducir al programador a desarrollar un buen sistema de software, logrando el crecimiento de la calidad y la productividad del producto. La elección de la metodología adecuada es más importante que utilizar las mejores y más potentes herramientas.

Metodología para el desarrollo unificado, *Rational Unified Process* (RUP)[31]

El Proceso Unificado de Desarrollo (RUP) es un proceso de ingeniería de software que mejora la productividad del equipo de trabajo y entrega las mejores prácticas del software a todos los miembros del mismo. RUP, proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML)[32] forman una metodología estándar basada en un pequeño grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo; debe conocer hacia donde se dirige; debe documentar el proyecto de una manera perdurable y extensible.

RUP durante el paso por las diferentes etapas de desarrollo se van a "transformar los requerimientos de los usuarios en un sistema software". Esta metodología se divide en 4 fases el desarrollo del software.

- **Fase de Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.

Fase de Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.

Fase de Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.

Fase de Transición: El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Ver Figura 2

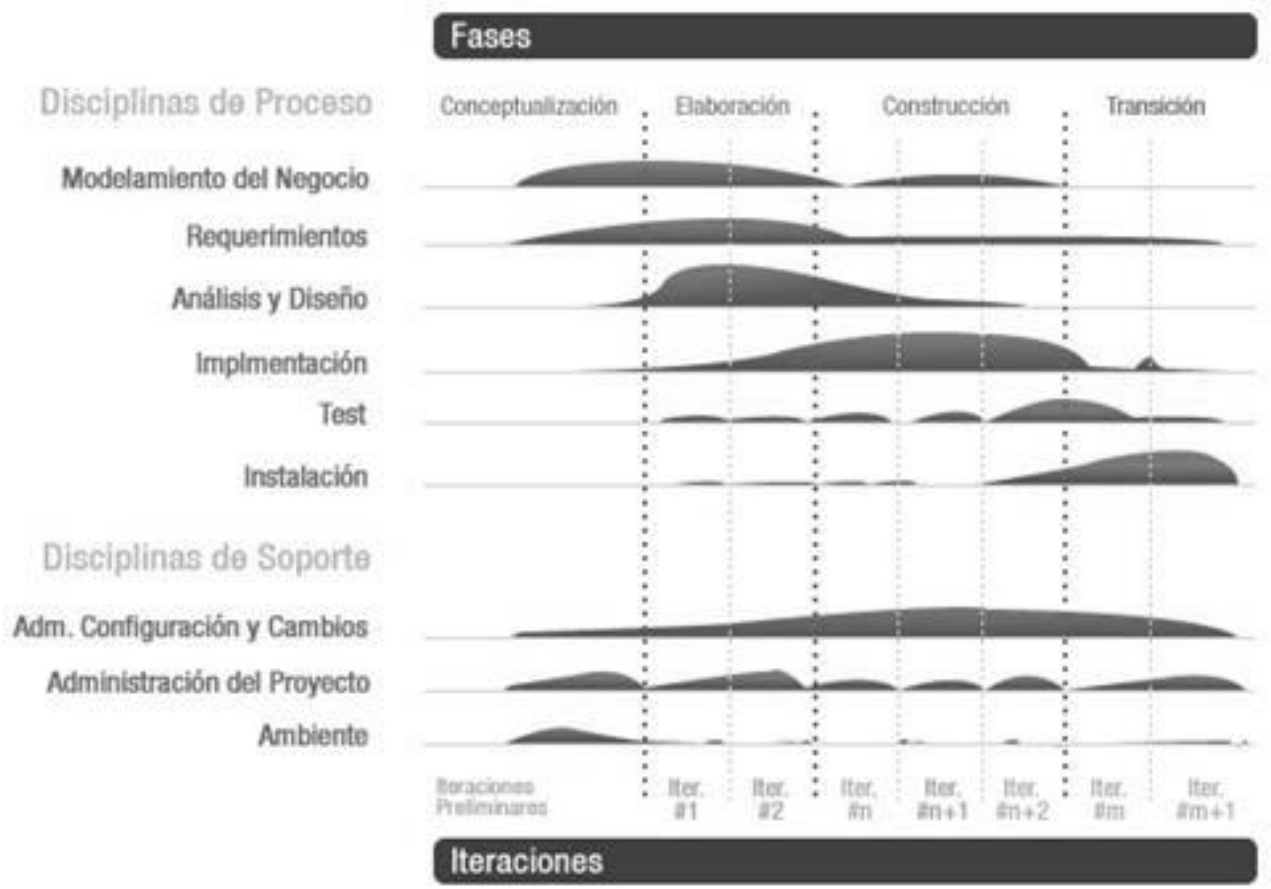


Figura. 2 Fases e Iteraciones de la Metodología.

Los elementos del RUP son:

Actividades, son los procesos que se llegan a determinar en cada iteración.

Trabajadores, vienen a ser las personas o entes involucrados en cada proceso.

Artefactos, un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Características del Proceso Unificado

RUP es un proceso de desarrollo de software de forma tal que se asignan tareas y responsabilidades cuyos objetivos son asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. El ciclo de vida de RUP se caracteriza por ser: Dirigido por los casos de uso, Centrado en la arquitectura, Iterativo e incremental.

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Representan requerimientos funcionales. Los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. La arquitectura se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por si sola.

Iterativo e Incremental: RUP propone dividir el trabajo en partes más pequeñas o mini proyectos, donde cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación las cuales están estrechamente relacionadas, cuyo resultado es una versión del software.

Beneficios de la iteración:

Reduce el coste del riesgo al coste de un solo incremento.

Menos riesgo de no sacar el producto al mercado en fecha.

Acelera el ritmo de desarrollo.

Las necesidades del usuario y correspondientes requisitos no pueden definirse completamente al principio. Se requieren iteraciones sucesivas.

1.11.2. Lenguaje de Modelado

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas:

1. Diagramas de Casos de Uso para modelar los procesos.
2. Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
3. Diagramas de Colaboración para modelar interacciones entre objetos.
4. Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
5. Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
6. Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
7. Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
8. Diagramas de Componentes para modelar componentes.
9. Diagramas de Implementación para modelar la distribución del sistema.

1.12. Herramientas CASE[33]

Las Herramientas **CASE** son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otros.

Visual Paradigm

El Visual Paradigm para UML es una herramienta CASE visual que ayuda a construir aplicaciones rápidamente, mejor y económicamente. Utiliza “UML”: como lenguaje de modelado, siguiendo el estándar UML 2.1. Esta herramienta tiene unas características gráficas muy cómodas que facilitan la realización de los diagramas de modelado que

sigue el estándar de UML, que son: Diagramas de clases, Casos de Uso, Comunicación, Secuencia, Estado, Actividad, Componentes, entre otros.

La herramienta Visual-Paradigm incorpora una funcionalidad de análisis textual, facilitando la interacción directa con un enunciado escrito en lenguaje natural, es decir, permite relacionar elementos presentes en un enunciado con los diagramas UML correspondientes, aunque no contempla el tratamiento directo en el texto de la relación entre los componentes del enunciado.

Beneficios:

1. Navegación intuitiva entre el modelo visual y el código.
2. Poderosa herramienta de generación de PDF[34] /HTML a partir de diagramas UML.
3. Sincronización entre el código fuente y el modelo en tiempo real o bajo demanda.
4. Entorno visual de modelado superior.
5. Soporte para toda la notación UML.
6. Sofisticados y automáticos diagramas de capas.
7. Análisis de textos.
8. Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
9. Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
10. Capacidades de ingeniería directa (versión profesional) e inversa.
11. Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
12. Disponibilidad de integrarse en los principales IDEs.
13. Disponibilidad en múltiples plataformas.

1.13. Herramientas de apoyo

Navegador

El navegador sobre el cual correrá la aplicación es el Mozilla Firefox este permite abrir por defecto las nuevas páginas Web en pestañas, cada una de esas pestañas tiene su propio botón de cerrado.

Firefox incorpora bloqueo de ventanas emergentes, marcadores dinámicos, soporte para estándares abiertos, y un mecanismo para añadir funcionalidades mediante extensiones.

Posee un corrector ortográfico para evitar que se cometan errores de ortografía en las entradas de información que se hagan. Tiene una sugerencia de búsqueda que se va desplazando a medida que se va introduciendo el texto que se desea buscar.

Mantiene a salvo a la aplicación de programas espías e impostores usando el poder de una comunidad de desarrollo que le da soporte. Estas y muchas otras posibilidades brindan este navegador del cual se hará uso para el sistema, aunque el sistema también correrá sobre el Internet Explorer.

Macromedia *Dreamweaver*

Dreamweaver es la herramienta de desarrollo Web líder del mercado y permite a sus usuarios diseñar, desarrollar y mantener de forma eficaz sitios y aplicaciones Web basadas en normas.

Con *Dreamweaver*, los desarrolladores Web lo abarcan todo, desde la creación y mantenimiento de sitios Web básicos hasta aplicaciones avanzadas compatibles con las mejores prácticas y las tecnologías más recientes.

Brinda múltiples herramientas visuales de diseño y un entorno de codificación adaptable a lenguajes de programación Web (PHP), trabaja con hojas de estilos CSS, permite la comparación de archivos para determinar que ha cambiado, permite el trabajo directo del lado del servidor, etc.

Conclusiones

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la comprensión del trabajo. Además se realizó un análisis para fundamentar las elecciones del lenguaje, del sistema gestor de bases de datos y de la metodología a utilizar a lo largo del desarrollo del sistema propuesto.

Capítulo 2

Descripción de la Propuesta Solución

Introducción

En la creación de un software es importante el entendimiento del negocio para llegar a una solución. En el presente capítulo se hará la descripción de la propuesta del trabajo, para lo que se determinarán los procesos de negocio que tienen que ver con el campo de acción, en base a lo cual se conformará un modelo de dominio. Además se enumerarán los requisitos funcionales y no funcionales que debe tener el sistema en proposición.

2.1. Descripción de los procesos de negocio

Para describir los procesos de negocio que se relacionan con el campo de acción de este trabajo es necesario enfocarse en el proceso de cómo se registra la Auditoría en el Sistema de Planificación Material y Financiera del MINFAR.

El proceso de registro se llevaría a cabo cuando un usuario accede a un sistema y el mismo realiza varias acciones sobre este sistema que constituyen eventos, los que generan información la cual es almacenada por el sistema.

Actualmente no se lleva a cabo ningún tipo de auditoría en el Sistema de Planificación Material y Financiera del MINFAR por lo que el SILOGA propone facilitar el registro de auditoría para un control y supervisión de los datos.

2.2. Propuesta del sistema

Para el desarrollo del proceso de Registro de Auditoría de la planificación Material y Financiera, el sistema debe registrar la auditoría a nivel de aplicación y de base datos, el cual te permitirá tener un registro de todas las acciones que se realicen sobre el sistema de Planificación, además de poder ver un Reporte según los criterios de búsqueda.

2.3 Modelo de Dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema.

El hecho de que los procesos del negocio tengan muy bajo nivel de estructuración; que sea difícil el establecimiento de reglas para la gestión como tal de los artefactos que se vayan definiendo en el proceso de desarrollo; que no existan flujos de información interconectados y bien definidos; inciden en que no se pueda llevar a cabo como tal un modelado de negocio, haciéndose necesario uno de dominio.

Tabla. 1 Definición de las entidades y conceptos principales

Concepto	Descripción
Sistema	Aplicación de software en funcionamiento que se utiliza para la gestión de la información.
Información	Conjunto organizado de datos, que puede ser que se encuentre en la aplicación o que se genere después de un evento.
Usuario	Persona que interactúa con los servicios que brindan los sistemas.
Acción	Actividad que realiza un usuario sobre un sistema.
Evento	Se registra cada vez que un usuario realiza una acción sobre el sistema.

2.4. Representación del modelo de dominio

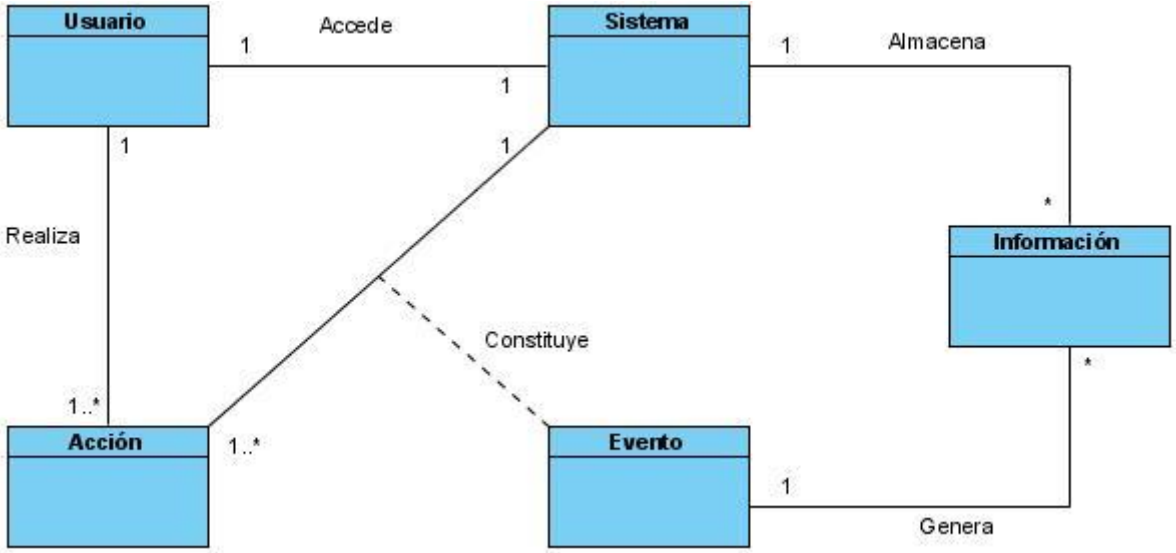


Figura. 3 Modelo de dominio

2.5. Requerimientos

Una de las principales tareas en el ciclo de desarrollo de un sistema es la de determinar los requerimientos del sistema de información. El propósito principal del

flujo de trabajo de RUP captura de requisitos es guiar el desarrollo hacia el sistema correcto donde se describan con claridad y sin ambigüedades el comportamiento del mismo. Así como lograr una comunicación efectiva entre el cliente (incluyendo a los usuarios) y a los desarrolladores sobre qué debe y qué no debe hacer el sistema.

¿Qué es un requerimiento?

Un requerimiento es la condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

Los requerimientos se pueden clasificar en: funcionales y no funcionales.

2.5.1. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Su objetivo principal es identificar y documentar las acciones que en realidad debe ejecutar el sistema para que cumpla con las metas planteadas al inicio de este trabajo.

Estas acciones se convierten en requisitos funcionales (RF)[35] y de acuerdo a los objetivos propuestos el sistema debe ser capaz de:

RF1. Registrar auditoría de aplicación.

RF2. Configurar auditoría de base datos.

2.1 Definir los eventos auditables en los esquemas.

2.2 Definir los eventos auditables en las tablas

2.3 Definir los eventos auditables en los campos

2.4 Guardar la configuración de la auditoría de base datos.

2.5 Generar Trigger.

RF3. Mostrar reportes de auditoría.

2.5.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades son las características que deben hacer al producto atractivo, usable, rápido y confiable. Forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto y de esta forma poder decir si el producto tiene la calidad requerida.

- **Apariencia o interfaz externa.**
  - ✓ Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
- **Usabilidad.**
  - ✓ El sistema será usado por personas que deben tener al menos, conocimientos básicos en el manejo de computadoras.
  - ✓ Siempre tendrá visible la opción de Ayuda, que brindará la posibilidad a los usuarios, de un mejor aprovechamiento de las funcionalidades del sistema.
- **Rendimiento.**
  - ✓ Tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información.
- **Soporte.**
  - ✓ Se requiere un servidor de bases de datos con las siguientes características:
    - o Soporte para grandes volúmenes de datos y velocidad de procesamiento.
    - o Tiempo de respuesta rápido en accesos concurrentes.
  - ✓ Versión de PHP 5.0.
  - ✓ Por parte del cliente se requiere un navegador capaz de interpretar JavaScript.
- **Portabilidad.**
  - ✓ Necesidad de que el sistema sea multiplataforma.
- **Seguridad**
  - ✓ Autenticación (contraseña de acceso)
  - ✓ Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que este activo.
  - ✓ Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
  - ✓ Verificación sobre acciones irreversibles (eliminaciones).

- **Políticos-culturales.**
  - ✓ El sistema solo podrá ser utilizado en territorio cubano y por las entidades autorizadas por el Ministerio de las FAR[36].
  - ✓ El producto no debe contener palabras en otros idiomas.
  - ✓ El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.
- **Legales.**
  - ✓ El sistema se basa en el manual de normas y principios establecidos por el MINFAR.
- **Confiabilidad.**
  - ✓ La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.
- **Hardware**
    - o Para las computadoras del cliente:
  - ✓ Se requiere tengan tarjeta de red.
  - ✓ Se requiere tengan al menos 64 MB de memoria RAM.
  - ✓ Se requiere al menos 100MB de disco duro.
  - ✓ Procesador 512 MHz como mínimo.
    - o Para los servidores:
  - ✓ Se requiere tarjeta de red.
  - ✓ Se requiere tenga al menos 256MB de RAM.
  - ✓ Se requiere al menos 1GB de disco duro.
  - ✓ Procesador 1.2 GHz como mínimo.
- **Software**
  - ✓ El sistema se desarrollará con tecnología PHP versión 5.0. Correrá sobre un servidor con el sistema operativo UNIX[37] (Linux)[38]. Se recurrirá a la tecnología Apache versión 2.0 o superior para el servidor Web. El sistema incluirá una base datos implementada en *PostgreSQL* versión 8.0 o superior.

- ✓ En las computadoras de los clientes debe estar instalado el navegador *Mozilla Firefox*. Versión 1.5 o superior. La comunicación de las computadoras clientes con el servidor será a través de conexiones de fibra óptica, a una velocidad constante de 100 Mbps o superior.

- **Ayuda y documentación en línea.**
  - ✓ Crear páginas de ayuda en cada interfaz y el Manual de usuario general referente a la aplicación.

2.6. Actores del sistema

Tabla. 2 Descripción de los actores del sistema

Actores del sistema	Justificación
Sistema	Aplicación en la cual se ejecuta una acción.
Admistrador del sistema	Persona encargada de la configuración del sistema.
Usuario	Persona que realizará una acción sobre el sistema.

2.7. Diagrama de Casos de Uso del Sistema

Un diagrama de Casos de Uso del sistema representa gráficamente a los procesos, representados en los casos de uso, y su interacción con los actores. Definen conjuntos de funcionalidades afines que el sistema debe cumplir para satisfacer todos los requerimientos que tiene a su cargo.

Esos conjuntos de funcionalidades son representados por los diferentes diagramas que darán solución a la aplicación. Para lograr una mejor compresión y facilitar el trabajo, se han dividido en 3 paquetes, un paquete contiene un CUS relacionado con el registro de log a nivel de aplicación, otro contiene los CUs relacionado con el registro de log a nivel de BDs y el otro contiene los caso de uso Cambiar contraseña y Autenticar usuario ya implementados en el sistema de seguridad que es el que brinda este servicio.

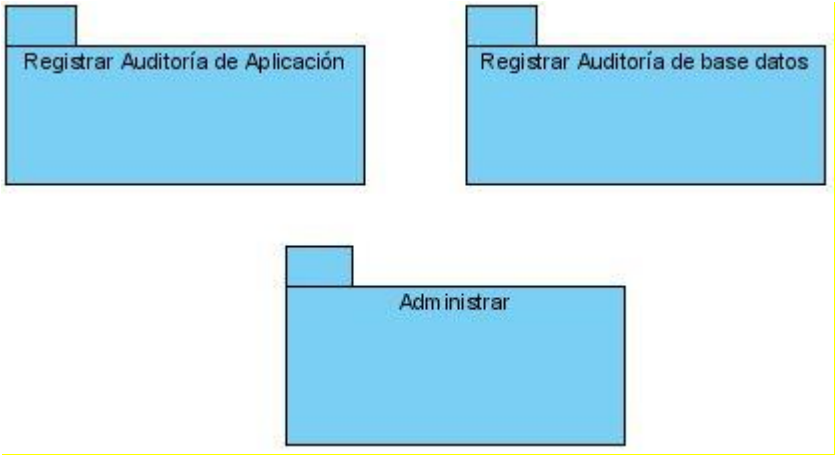


Figura. 4 Distribución de las funcionalidades por paquetes.

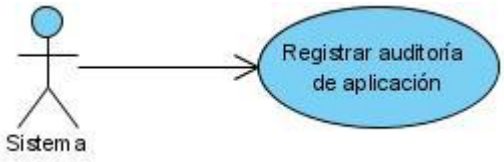


Figura. 5 Diagrama de casos de uso del sistema Registrar Auditoría de aplicación.

Figura. 6 Diagrama de casos de uso del sistema Registrar Auditoría de base de datos

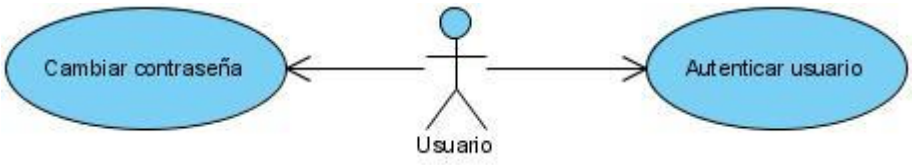


Figura. 7 Diagrama de casos de uso del sistema Administrar

2.8. Especificaciones de los casos de usos

Tabla. 3 Descripción breve del caso de uso Registrar Auditoría de aplicación

CU- 1	Registrar Auditoría de aplicación.
Actor	Sistema
Descripción	El caso de uso se inicia cuando el actor realiza una acción sobre el sistema de planificación. Termina cuando se ha almacenado toda la información referente a la Auditoría de aplicación.
Referencia	RF1
Prioridad	Crítico

Tabla. 4 Descripción breve del caso de uso Configurar auditoría de base datos

CU- 2	Configurar auditoría de base datos
Actor	Administrador del sistema
Descripción	El caso de uso se inicia cuando el Administrador del sistema define los eventos auditables en los esquemas, tablas, campos y se guarda esta configuración. Termina cuando los Triggers son generados.
Referencia	RF2,RF2.1, RF2.2, RF2.3, RF2.4, RF2.5, RF2.6
Prioridad	Crítico

Tabla. 5 Descripción breve del caso de uso Mostrar reportes de auditoría de base datos

CU- 4	Mostrar reportes de auditoría base datos
Actor	Administrador del sistema
Descripción	El caso de uso se inicia cuando el Administrador del sistema desea ver un reporte. Termina cuando el reporte es mostrado.
Referencia	RF3

Prioridad	Crítico

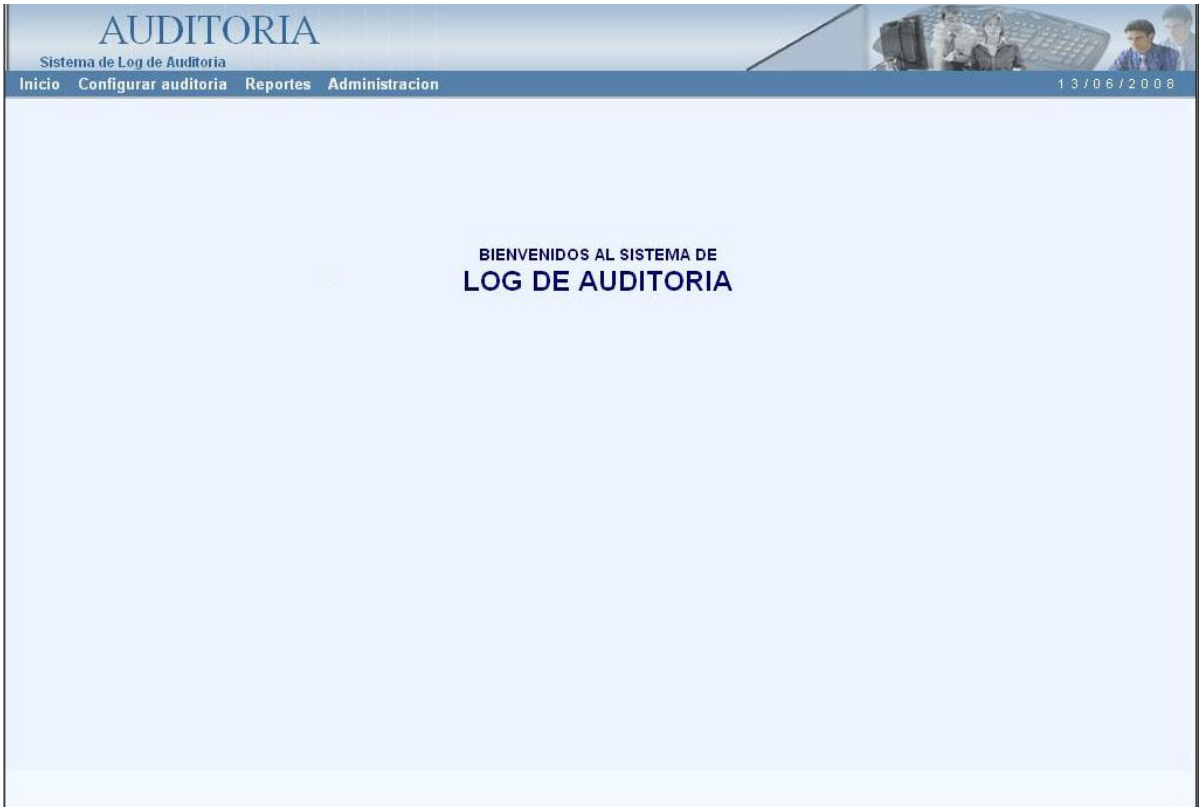
Tabla. 6 Descripción Textual del caso de uso Registrar Auditoría de aplicación.

Caso de uso:	**Registrar Auditoría de aplicación**
Actores:	Sistema (inicia).
Propósito:	Registrar Auditoría de aplicación.
Resumen	El caso de uso se inicia cuando el actor Sistema ejecuta una acción, se solicitan los eventos auditables al sistema de seguridad y auditoría, luego se auditan estos eventos y termina el caso de uso enviando esta auditoría al sistema de seguridad y auditoría.
Precondiciones:	Existencia de una acción.
Postcondiciones:	Quedan registrados los log de auditoría de aplicación.
Responsabilidades:	RF1
Curso normal de eventos para el caso de uso	
Acción del actor	**Respuesta del sistema**
1. El actor Sistema ejecuta una acción.	2. El sistema solicita eventos auditables al sistema de seguridad y auditoría.
	3. El sistema realiza auditoría si los eventos son auditables.
	4. Se envía la auditoría al sistema de seguridad y auditoría, quien la guarda, terminando el CU.
Curso alterno de eventos	

Acción del actor	Respuesta del sistema
	3. El sistema termina el CU si los eventos no son auditables.

Tabla. 7 Descripción Textual del caso de uso Configurar Auditoría de BD.

Caso de uso	Configurar auditoría de base datos
Actores:	Administrador del sistema (inicia).
Propósito:	Configurar la auditoría de base de datos
Resumen	Mediante este caso de uso el actor Administrador del sistema define los eventos auditables en los esquemas, tablas y campos, se guarda esta configuración. El CU termina Cuando son generados los Triggers.
Precondiciones:	El Administrador del sistema debe estar autenticado y la existencia de eventos.
Postcondiciones:	La configuración de la auditoría de base datos queda guardada.
Responsabilidades:	RF2, RF2.1, RF2.2, RF2.3, RF2.4, RF2.5
Casos de uso relacionados:	-



Interfaz 1

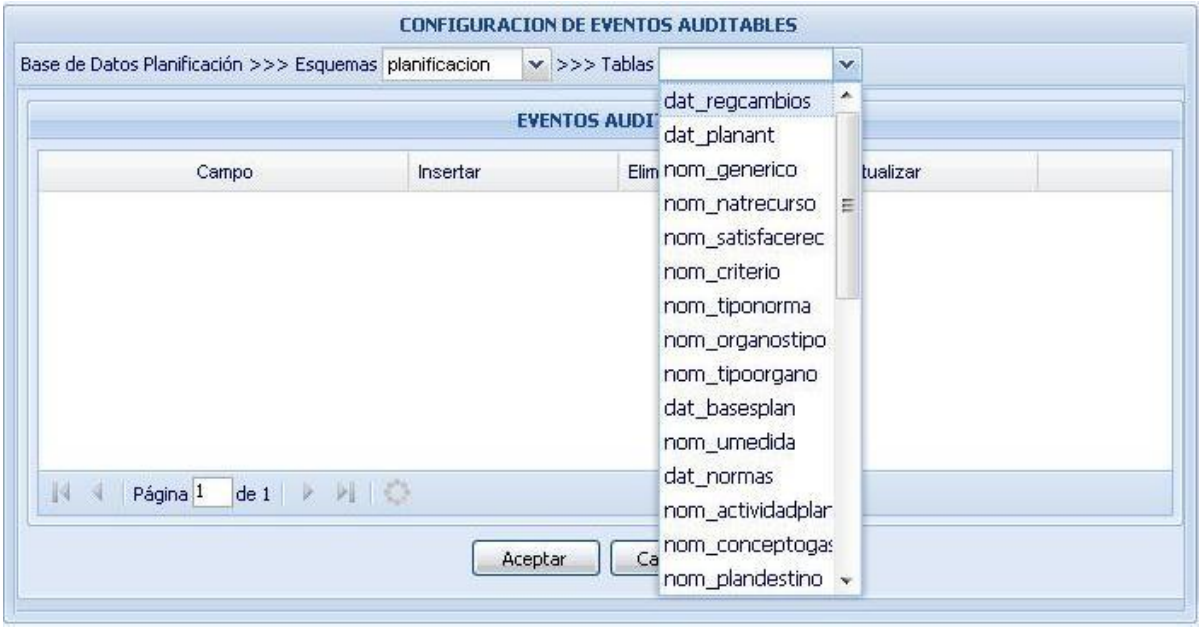
Curso normal de eventos para el caso de uso	
Acción del actor	**Respuesta del sistema**
1- El actor desea configurar la auditoría de base datos, para lo que da clic en el vínculo Configurar auditoría de la Interfaz 1.	2- El sistema muestra la Interfaz 2, en la que puede configurar la auditoría de base datos.

Interfaz 2

3- El actor selecciona un esquema, ver Interfaz 3.	4- El sistema carga las tablas de ese esquema, ver Interfaz 4.

Interfaz 3

5- El actor selecciona una tabla, ver Interfaz 4.	6- El sistema muestra los campos de esa tabla, ver Interfaz 5.



Interfaz 4

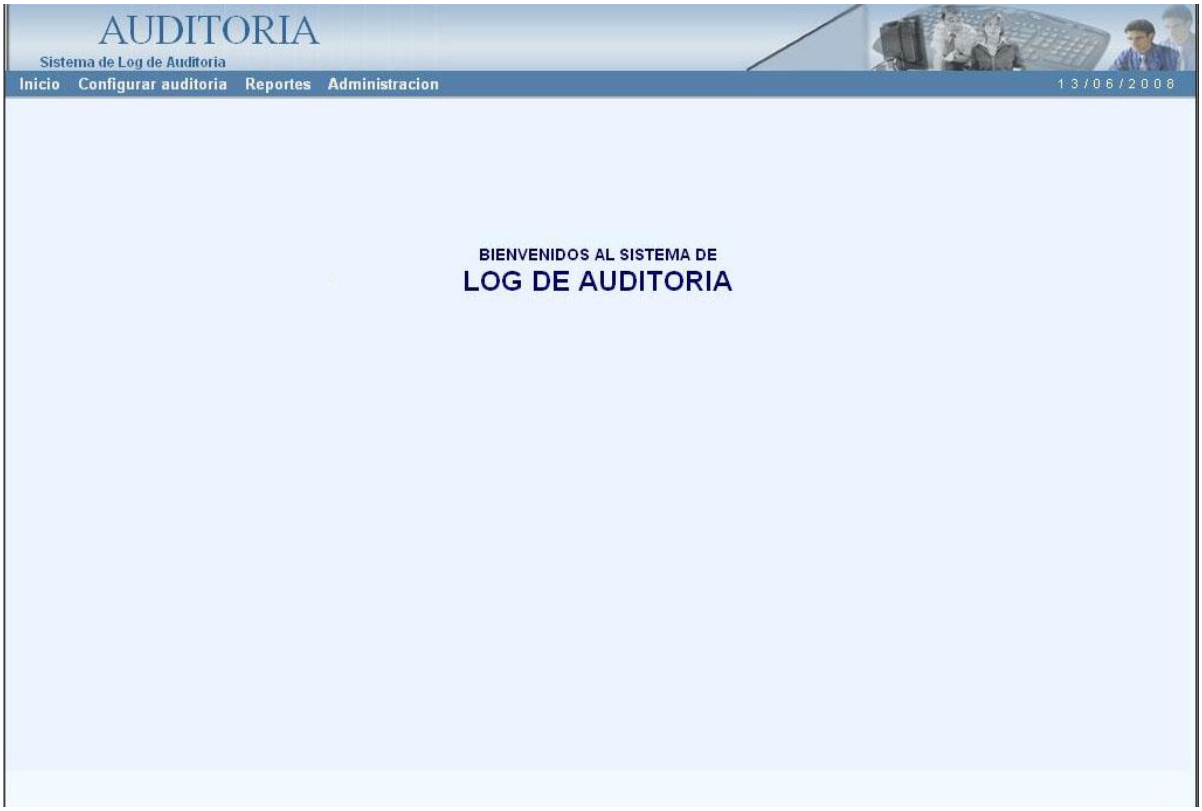
7- El actor selecciona los eventos auditables por campos, al terminar presiona el botón Aceptar, ver Interfaz 5.	8- El sistema guarda la configuración de la auditoría de base datos.

Interfaz 5

	9- El sistema genera los Trigger.

Tabla. 8 Descripción Textual del caso de uso Mostrar reportes de auditoría base datos

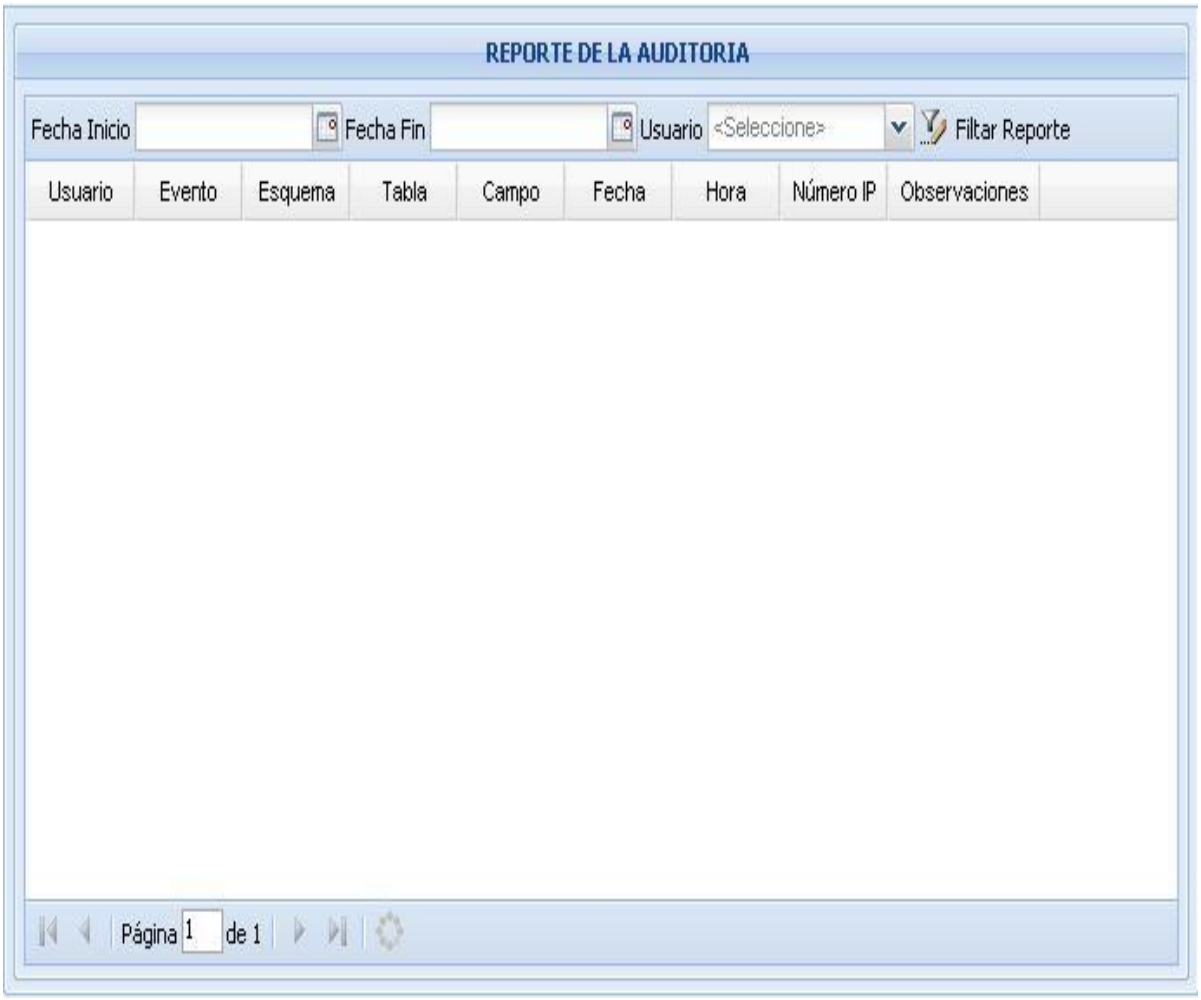
Caso de uso:	Mostrar reportes de auditoría base datos
Actores:	Administrador del sistema (inicia).
Propósito:	Poder ver todos los reportes de auditoría de base datos
Resumen	Mediante este caso de uso se muestran los reportes de auditoría de BDs cuando el Administrador del sistema desee verlos.
Precondiciones:	Deben estar guardados los logs de auditoría.
Postcondiciones:	Muestra el reporte solicitado.
Responsabilidades:	RF3
Casos de uso relacionados:	Configurar auditoría de base datos



Interfaz 1

Curso normal de eventos para el caso de uso

Acción del actor	Respuesta del sistema
1- El actor desea ver el reporte de la auditoría de BDs, para lo que da clic en Mostrar reporte, ver Interfaz 1.	2- El sistema muestra la Interfaz 2 donde se muestran todos los reportes que existen, Ver Interfaz 2.



Interfaz 2

3- El actor seleccionar un rango de fecha y un usuario del cual desea ver los reportes, da clic en el botón Filtrar Reporte, ver Interfaz 2.	4- El sistema busca en la BD todos los logs que cumplan con el criterio. Muestra el reporte, ver Interfaz 2.

Conclusiones

En este capítulo se modeló una propuesta de solución de software, a través del modelo conceptual, los requerimientos funcionales y no funcionales descritos detalladamente. Además de la descripción breve y la textual de todos los casos de usos.

Capítulo 3

Análisis y diseño del sistema

Introducción

En este capítulo se propone la construcción de la solución a través de los modelos de clases del análisis y del diseño, los diagramas de secuencia y la construcción y descripción de la BDs, partiendo del diagrama de CUs del sistema obtenido en el capítulo anterior.

3.1. Modelo del análisis

El modelo del análisis es un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema, pues el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

Aquí se mostrarán los diagramas de las clases del análisis donde estas pueden ser: clases de interfaz, clases controladoras y clases entidad.

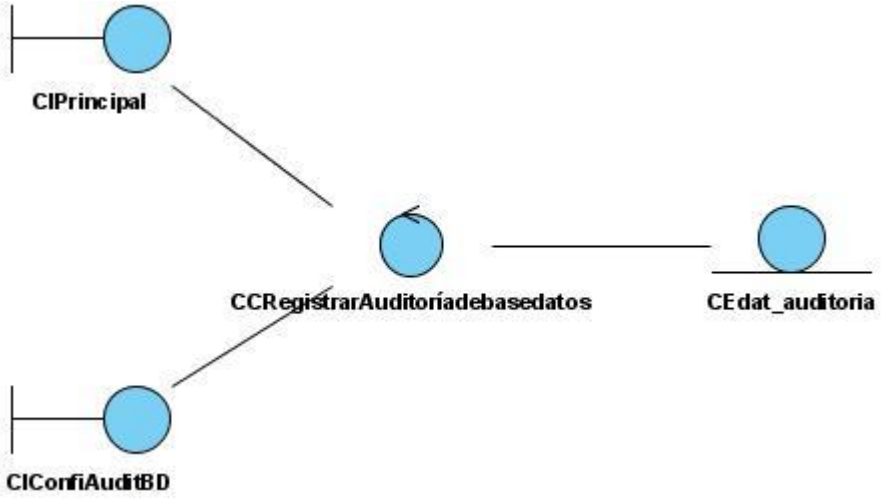


Figura. 8 Diagrama de clases del análisis del paquete Registrar Auditoría de base de datos, CU Configurar auditoría de base datos.

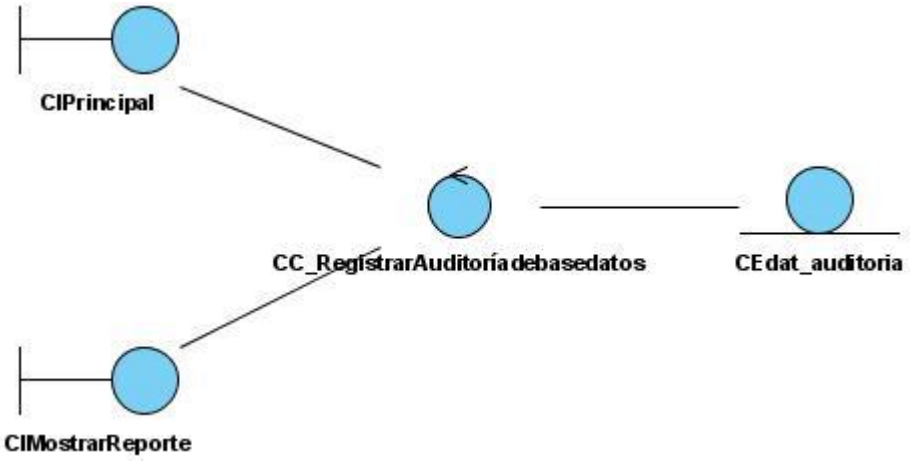


Figura. 9 Diagrama de clases del análisis del paquete Registrar Auditoría de base de datos, CU Mostrar reportes de auditoría de base datos.

3.2. Modelo del diseño

Una vez que se analizan y especifican los requisitos del software, el diseño es la primera de las tres actividades técnicas (diseño, generación de código y pruebas) que se requieren para construir y verificar el software.

3.2.1. ¿Qué es el diseño del software?

Es el proceso de establecer una implementación eficiente que realice las funciones y tenga la información del análisis de dominio, en las clases del diseño se tienen reflejados los atributos y métodos correspondientes, se detallan las abstracciones del análisis con características de implementación.

3.3. Arquitectura del sistema

La arquitectura es la representación para analizar la efectividad del diseño en la obtención de los requisitos fijados.

Dentro de la arquitectura de software, uno de los temas más tratados en los últimos años es el de los patrones, ya sean los patrones de diseño o los de arquitectura.

Un patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de una buena solución al problema, de forma que puede reutilizarse continuamente. Para el desarrollo de la aplicación se decidió utilizar el patrón de capas.

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia.

Se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

Descripción de la arquitectura que se propone:

El sistema se divide en tres capas:

• Capa de presentación: es la que ve el usuario, presenta la aplicación al mismo, le comunica la información y captura la información de él dando un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio.

• Capa de negocio: es donde residen los procesos que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio o de lógica del negocio pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación y con la capa de acceso a datos.

• Capa de acceso a datos: capa final en el flujo e intercambio entre todas las capas, aquí es donde se utilizan los componentes lógicos de acceso a los datos para obtener el acceso a los datos.

Estos componentes abstraen la semántica del almacén de datos subyacente y la tecnología de acceso a datos (como PDO) y proporcionan una interfaz simple de programación para la recuperación y realización de operaciones con datos.

Figura. 10 Arquitectura de tres capas.

3.4. Mecanismos de Diseño

Los mecanismos de diseño se modelan para comunicar la manera más óptima en que debe darse solución a problemas repetitivos en la aplicación. Aunque su desarrollo y

mantenimiento es opcional, se recomienda su uso en entornos de desarrollo complejos.

Con su elaboración, se modelaría un conocimiento que ayudará tanto al desarrollo de la aplicación actual como a futuras iteraciones, además de las labores de mantenimiento. En los mecanismos de diseño intervienen diversos elementos de la aplicación (clases, subsistemas).

Los mismos reportan beneficios como:

1. Mantener la homogeneidad en el diseño.
2. Reutilizar soluciones anteriormente probadas.
3. Reutilizar documentación.

Un ejemplo típico de mecanismos son los patrones de diseño.

3.4.1. Mecanismo de Diseño para Seguridad

Las aplicaciones Web permiten el acceso de usuarios a recursos centrales como el servidor Web y, a través de éste, a otros como los servidores de base de datos. Con la implementación correcta de medidas de seguridad, pueden protegerse los recursos con los que se cuenta, así como proporcionar un entorno seguro a los usuarios que trabajen con la aplicación.

Administrar centralmente la seguridad de la aplicación es un factor esencial para controlar el acceso a la misma, esto se logra a través del empleo de variables de sesión, en la cual se registra la identidad del usuario, una vez se autentica en el sistema.

Al constituir esta aplicación como módulo de un ERP, las variables de sesión necesarias se obtienen a través de otro módulo encargado de la seguridad, que brinda esta posibilidad por medio de un servicio Web, el cual proporciona una interfaz cControlacceso para la autenticación de los usuarios, esta interfaz contiene un método público llamado logueo, que recibe como parámetro el usuario, la contraseña y el módulo al que desea entrar, este método devolverá un mensaje de error en caso de que exista algún problema.

En caso de éxito en la autenticación el sistema devuelve además de las variables de sesión necesarias para tratar al usuario en el sistema, el menú y la barra de herramienta que será diferente para cada usuario en dependencia del rol. El rol del usuario es una de las variables de sesión de mayor importancia ya que esto indica al sistema los privilegios te tiene el usuario autenticado.

A continuación se muestra en la Figura 11 el mecanismo de diseño para la seguridad del sistema basado en el uso de servicios Web. .

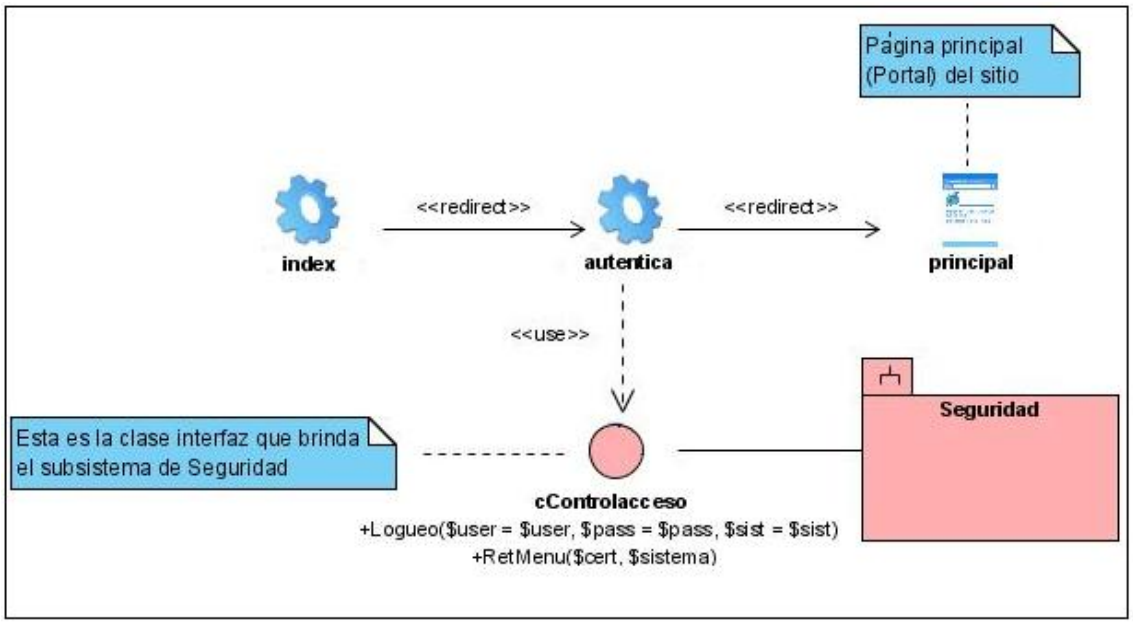


Figura. 11 Mecanismo de Diseño para Seguridad.

3.4.2. Mecanismo de Diseño para Acceso a Datos.

Este mecanismo surge con el objetivo de facilitar el acceso a datos del sistema, creándose un punto de referencia para los desarrolladores, permitiendo obtener diagramas más entendibles, que posibiliten una mayor comunicación con el equipo de desarrollo, trazando una línea común, una política a seguir, fomentando algo muy indispensable para lograr eficiencia, la reutilización.

La vista estática de este mecanismo de acceso a datos (Figura 12) visualiza un conjunto de clases que interactúan para dar acceso y manipulación de los datos de la persistencia desde el nivel más bajo, es decir, utilizando los objetos nativos brindados por el entorno de desarrollo PHP como son la extensión PDO que define una ligera interfaz, para el acceso a bases de datos., siguiendo así hasta la abstracción del acceso a datos a través de meBase de la cual heredan las clases particulares del sistema como Típicas y meSimples.

Para dar la responsabilidad a una clase que encapsulara las instancias de estos objetos se definió la clase FactoríaTípica, que está concebida para instanciar todas las típicas del sistema para utilizarlas de algún modo.

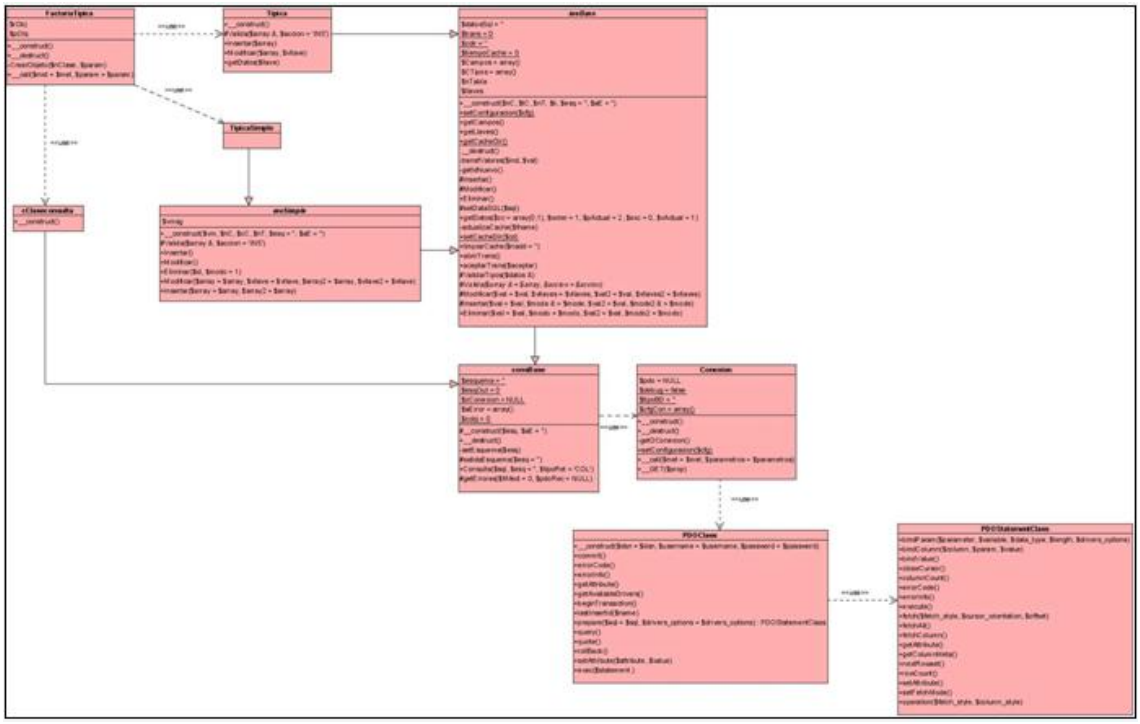


Figura. 12 Vista estática del mecanismo de diseño para acceso a datos.

A continuación se muestra una descripción más detallada de estas clases.

FactoriaTipica: clase que implementa la interfaz del modelo de persistencia con el resto de los subsistemas. A través de esta clase se crean y se manipulan los objetos de las típicas simples, los nomencladores y las demás típicas. Es una puerta entre la capa de Acceso a Datos y la capa de Lógica de Negocio, para su creación se tomó en cuenta lo dictado por el patrón de diseño Factoría, el cual centraliza en una clase controladora la creación de objetos de un tipo determinado.

Implementa un método de instanciación de clases típicas.

Típica: es una clase que representa a las clases típicas en general de la aplicación. Existe una típica para cada entidad de la base de datos. Para la implementación de esta clase se decidió aplicar el patrón de diseño Table Data Gateway, que consiste en crear una instancia por cada tabla existente en la BD. Sus métodos consisten en las operaciones básicas que se realizan sobre estas tablas, (INSERT, DELETE, UPDATE). Hereda de la clase abstracta meBase.

TípicaSimple: es una clase que representa a las clases típicas para nomencladores simples. Para la implementación de esta clase se decidió aplicar el patrón de diseño Table Data Gateway. Sus métodos consisten en las operaciones básicas que se realizan sobre estas tablas, (INSERT, DELETE, UPDATE). Hereda de la clase abstracta meSimple.

cClaseconsulta: es una clase que representa a las clases consultas en general de la aplicación. Existe una clase consulta para cada entidad de la base de datos. Hereda de la clase abstracta consBase.

meSimple: clase abstracta, base para la implementación de las típicas que responderán a los nomencladores simples** del modelo de persistencia dado. Redefine las operaciones básicas con la funcionalidad de Validación dada.

** Entidades cuya estructura responde al siguiente patrón: idALGO, ALGO, actual. Donde ALGO representa la descripción del atributo principal de los nomencladores clásicos.

meSimple define las operaciones básicas que pudieran realizarse a una entidad (INSERT, DELETE, UPDATE) para los nomencladores simples. Hereda de la clase abstracta meBase.

meBase: clase abstracta, base para el resto de las que implementen funcionalidades para el trabajo con las entidades del sistema a implementar. Implementa las operaciones básicas que pudieran realizarse a una entidad (INSERT, DELETE, UPDATE). Hereda de consBase la operación de CONSULTA.

consBase: esta clase es la base en toda la jerarquía de Acceso a Datos y es empleada para aportar contenido dinámico a las plantillas. Encapsula el objeto conexión. Implementa la operación de CONSULTA.

Conexion: clase encargada de establecer la conexión con el servidor de la BD a través de un objeto PDO de la librería de PHP. Se concibió aplicando el patrón Singleton el cual garantiza una única instancia para una clase y la creación de un mecanismo global (único) de acceso a dicha instancia.

PDO: es un modelo de acceso a bases de datos para PHP. PDO nos brinda una capa de abstracción para el acceso a bases de datos desde PHP.

3.5. Diagramas de clases del Diseño.

Para optimizar la modelación del diseño del SILOGA, se creó un diagrama genérico de clases del diseño (Figura 14), que tiene como misión la representación de los elementos comunes para todos los diagramas de clases a desarrollar, destacando en color azul el nombre de las clases que varían en dependencia del diagrama que se esté fabricando, por tanto los posteriores diagramas solamente contendrán las páginas clientes, servidoras, las lógicas de negocio, y en caso de tener formularios y ficheros java script también serán incluidos.

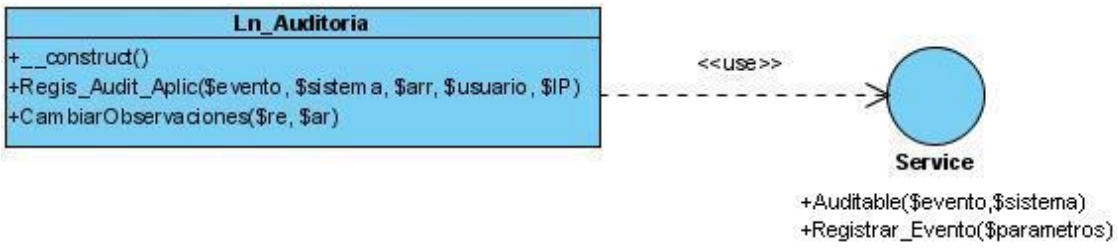


Figura. 13 Diagramas de clases de diseño de Registrar Auditoría de Aplicación

LnAuditoría: esta clase se encarga de realizar la auditoría en la aplicación, auditando todas las acciones que se realicen sobre el sistema, a través de un método que pide dos servicios al sistema de seguridad y auditoría para completar exitosamente la misma.

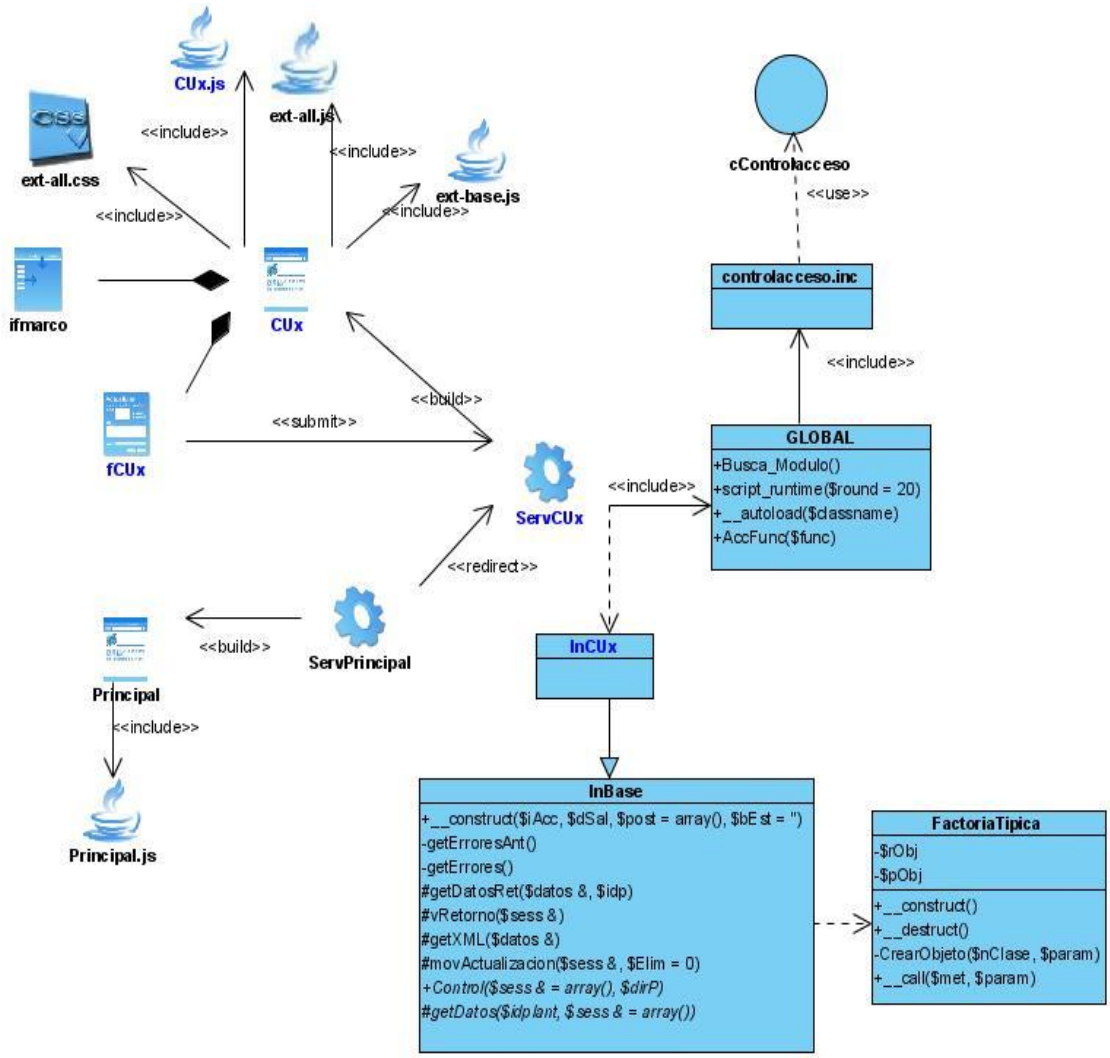


Figura. 14 Diagrama genérico de clases del diseño

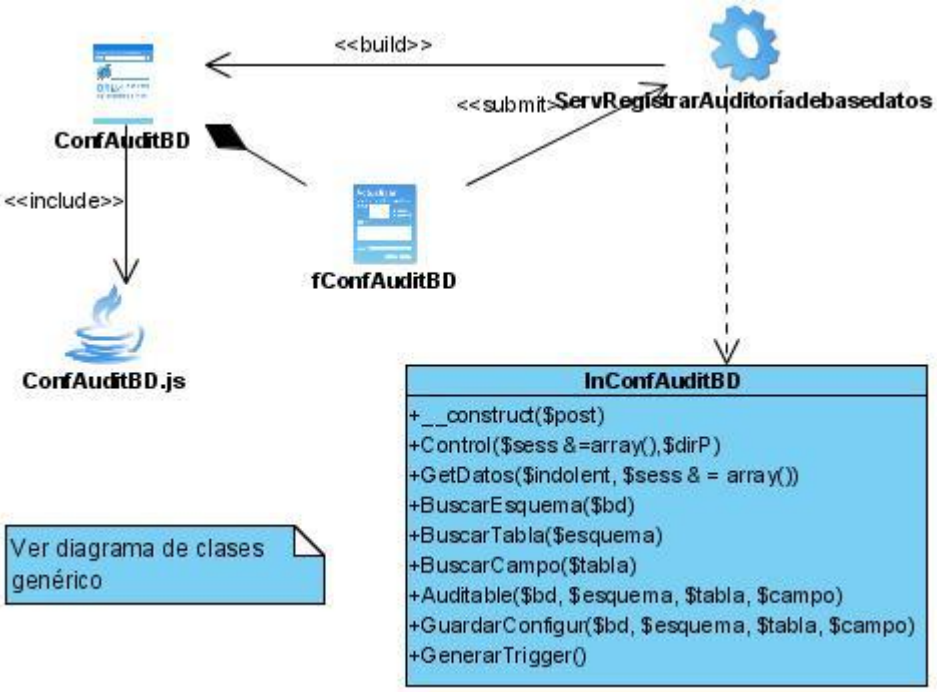


Figura. 15 Diagrama de clases del diseño del paquete Registrar Auditoría de base de datos, CU Configurar auditoría de base datos (configurar auditoría).

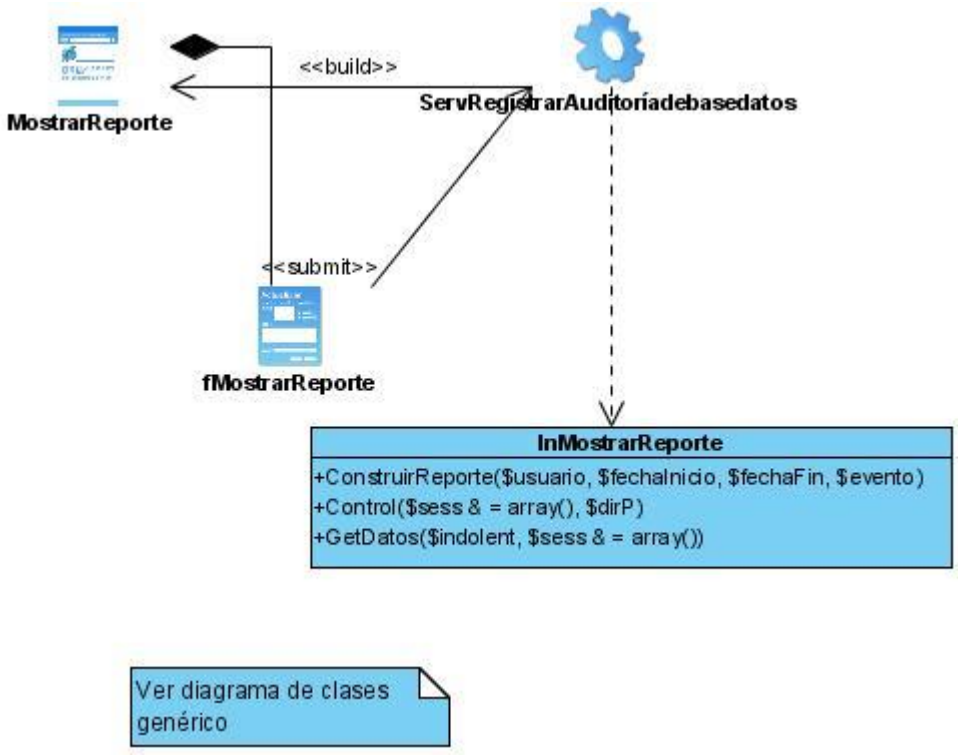


Figura. 16 Diagrama de clases del diseño del paquete Registrar Auditoría de base de datos, CU Mostrar reportes de auditoría de base datos.

3.6. Diagramas de interacción

Para organizar interacción entre objetos en un sistema uno de los diagramas más efectivos es el de secuencia, el que se modela para cada caso de uso y contiene detalles de implementación, incluyendo los objetos y clases que se usan para implementar el mismo, y mensajes pasados entre los objetos. Estos diagramas destacan el orden temporal de los mensajes, mostrando el intercambio de estos y poniendo especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

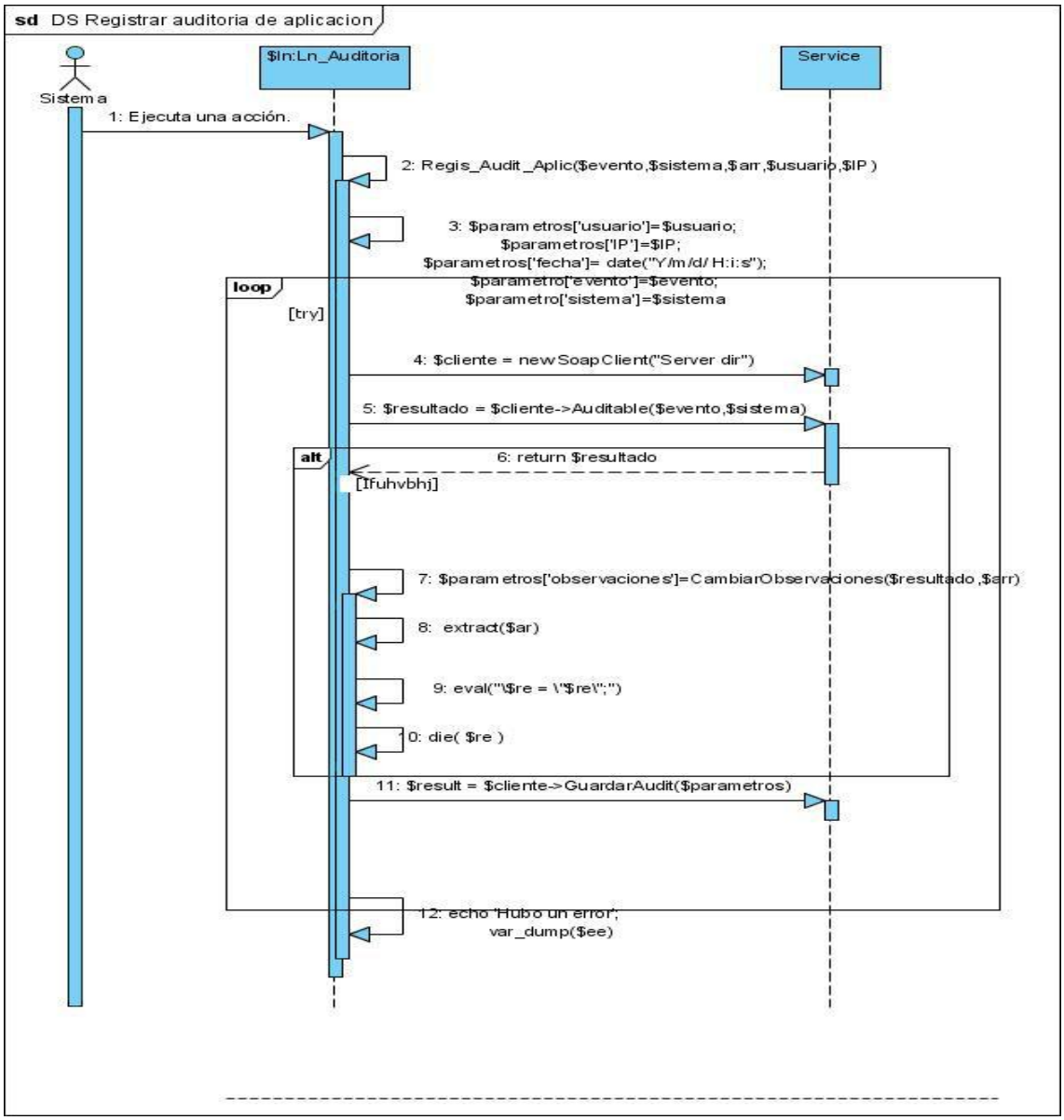


Figura. 17 Diagrama de Secuencia de Registrar Auditoría de Aplicación

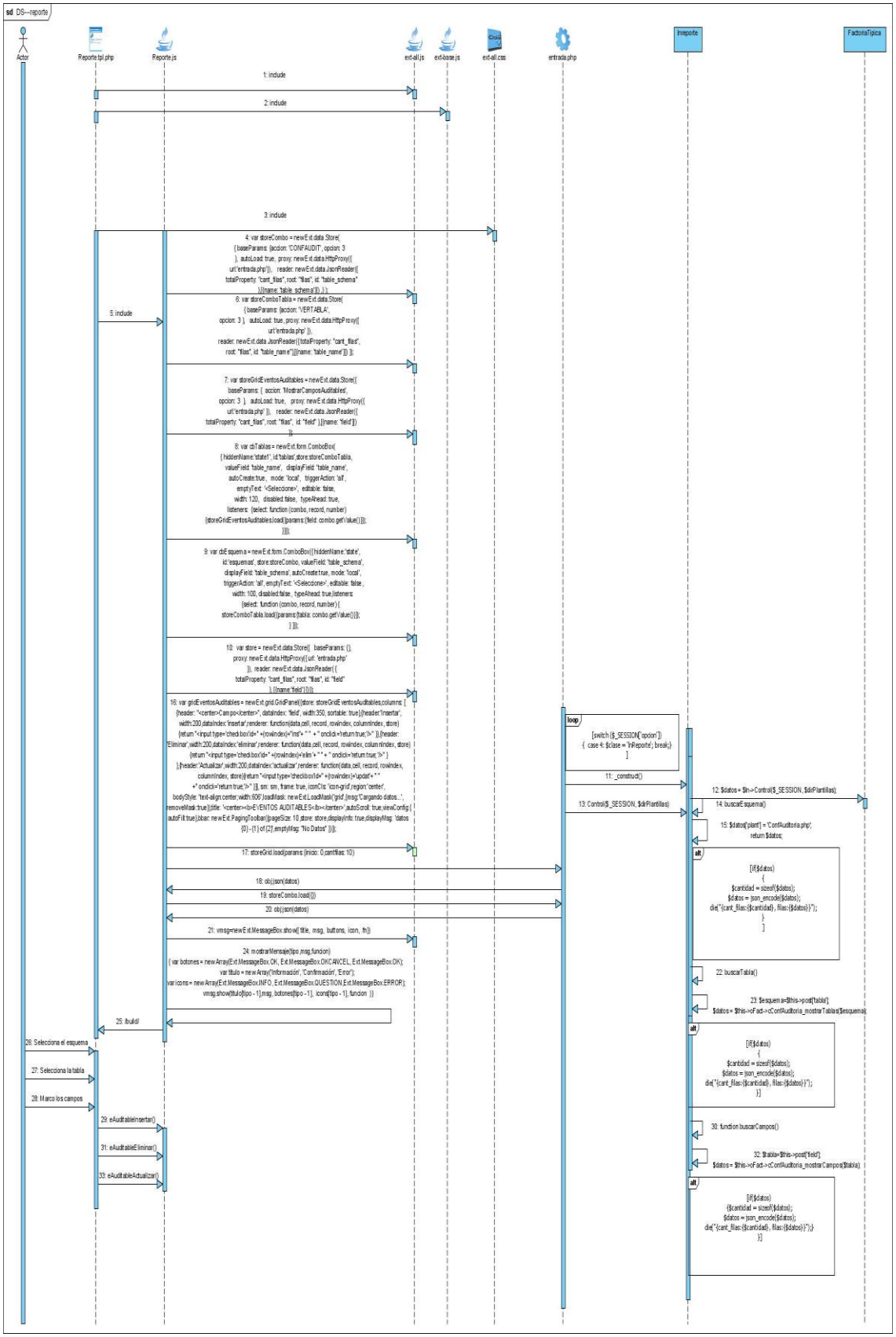


Figura. 18 Diagramas de secuencias configurar auditoría de bpd

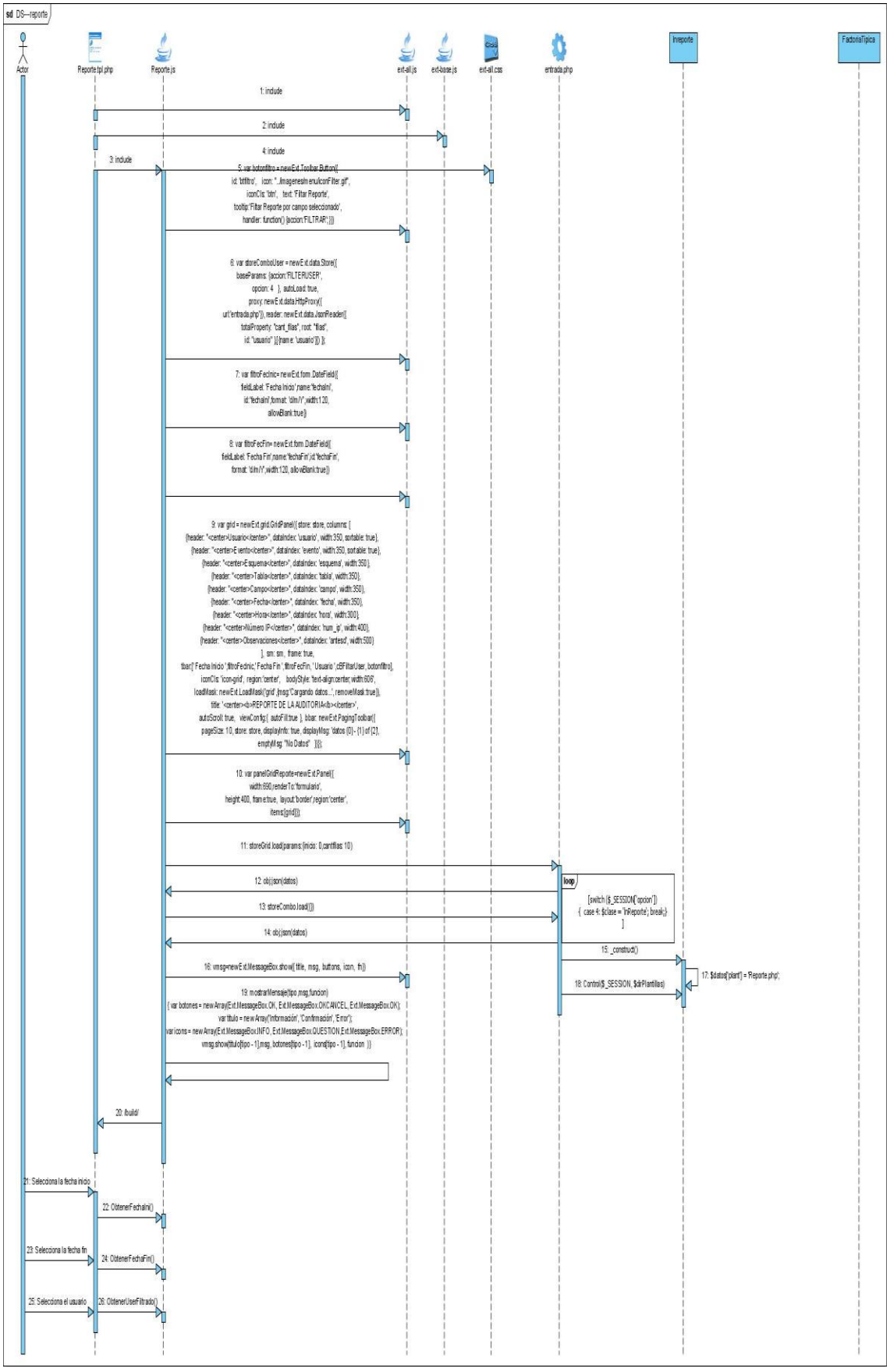


Figura. 19 Diagramas de secuencias mostrar reporte de auditoría de bd.

3.7. Modelo de datos

Los modelos entidad-relación (MER)[39] son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para dicho sistema, sus interrelaciones y propiedades.

Además en él se modela el tratamiento de la información con carácter persistente dentro del sistema.

Figura. 20 Diagrama Entidad Relación de la BD.

Tabla. 9 Descripción de la tabla que contiene la Base de Datos.

Nombre: dat_auditoría		
Descripción: Esta tabla contiene el usuario que se conecta a la base de datos, el evento auditable por esquema, tabla y campo, se guarda lo que había antes del evento realizado y después del mismo, fecha y hora en que se realizó el evento y número de ip desde donde se realizó.		
Atributo	**Tipo**	**Descripción**
Id_auditoría	Int4	Identificador de la auditoría
usuario	Varchar(0)	Usuario de la bd.
evento	Varchar(0)	Evento que se realiza.
esquema	Varchar(0)	Esquema que se audita.
tabla	Varchar(0)	Tabla que se audita.
campo	Varchar(0)	Campo que se audita.
antesD	Varchar(0)	Donde se guarda lo que había antes de la auditoría.
despuesD	Varchar(0)	Donde se guarda lo que había

		después de la auditoría.
Fecha	Date	Fecha en la que se audita.
Hora	Time(8)	Hora en la que se audita.
Num_IP	Varchar(0)	Número ip desde el cual se conecta a la bd.

3.8. Estándares de diseño

La página principal de la aplicación, se concibe como un portal, con un menú, que no debe exceder de 3 niveles de profundidad, donde se agrupan las funcionalidades del sistema.

Las páginas deben tener una cabecera (banner) representativa, un área de trabajo y una barra menú con las opciones, además tener una hoja de estilo en común para lograr la uniformidad.

3.9. Tratamiento de errores

Los errores se tratan en los diferentes niveles desde la entrada de datos de los usuarios hasta los más complicados, que se pueden generar en la actualización de los datos.

Para prevenir errores por parte del usuario, a la hora de efectuar cualquier operación se deshabilitan ciertos botones si el usuario no tiene que utilizarlos en ese momento.

Mediante una combinación de validación en el lado del cliente y en el lado del servidor, se garantiza que los datos suministrados por los usuarios, se almacenen íntegros y no existan inconsistencias.

Conclusiones

En este capítulo se mostraron varios diagramas para llevar a cabo el análisis y diseño del SILOGA, donde se explica la lógica del sistema , se diseñó el diagrama de clases persistentes que permitió hacer el modelo de datos para el diseño de la BD que se utilizará y así pasar a la próxima fase, la de implementación.

Capítulo 4

Implementación del Sistema

Introducción

Dando continuidad a este trabajo se propone en este capítulo los modelos de despliegue e implementación teniendo en cuenta el resultado del análisis y el diseño realizado en el capítulo anterior.

4.1. Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

El flujo de implementación esta fuertemente determinado por el lenguaje de programación.

Los **diagramas de despliegue** y **componentes** conforman lo que se conoce como un modelo de implementación al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará a aplicación.

4.2. Modelo de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución (los componentes que sólo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes). Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general un nodo será una unidad de computación de algún tipo, desde un sensor a un mainframe. Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz).

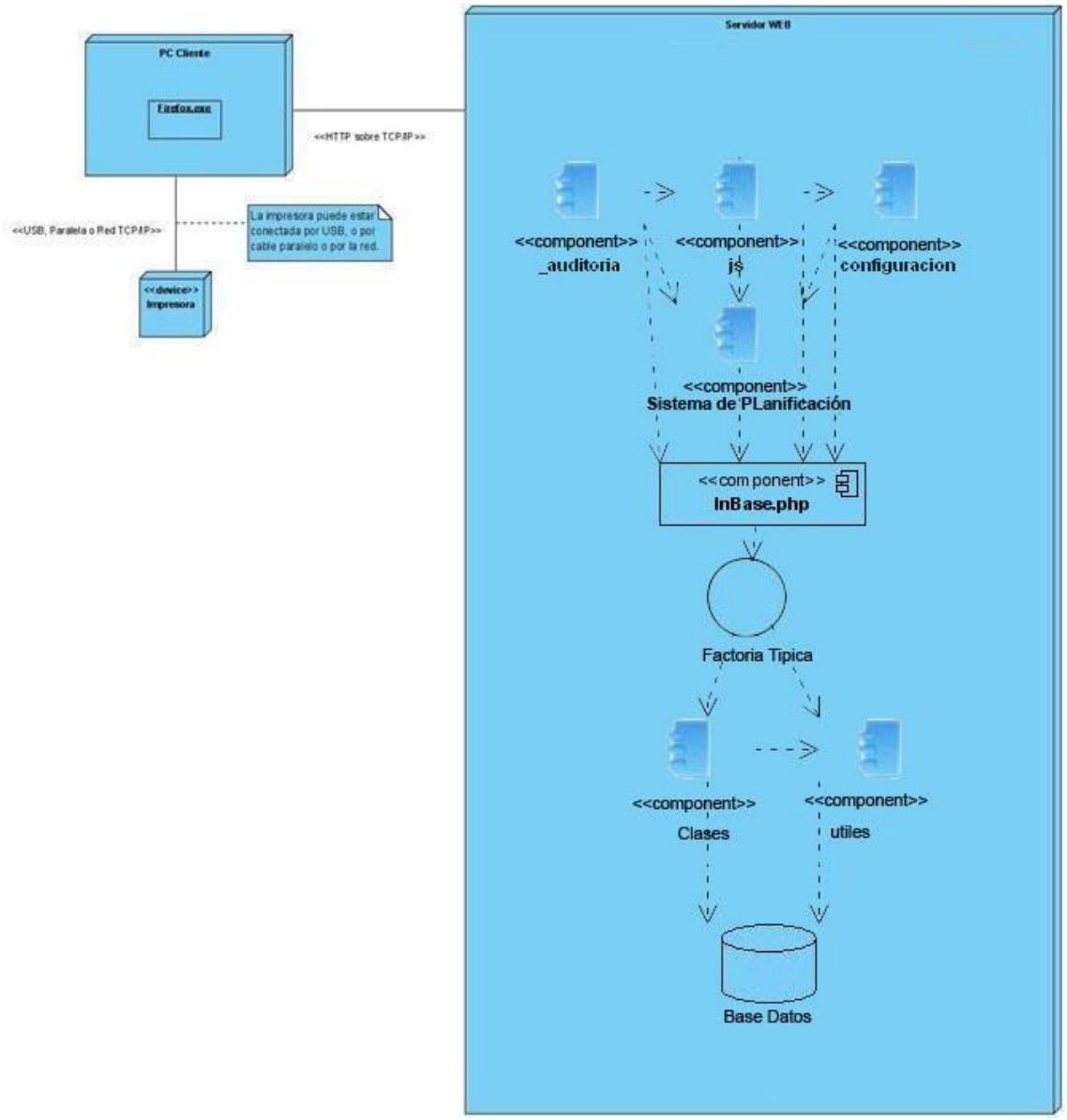


Figura. 21 Modelo de despliegue

4.3. Diagrama de componentes

Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten.

Es un diagrama que muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones.

Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que

un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Un ejemplo típico de una jerarquía en capas de este tipo es: Interfaz de usuario; Paquetes específicos de la aplicación; Paquetes reusables; Mecanismos claves; y Paquetes *hardware* y del sistema operativo.

Figura. 22 Diagrama de componente de Registrar Auditoría de Aplicación.

Figura. 23 Diagrama de componente del paquete Registrar Auditoría de Aplicación.

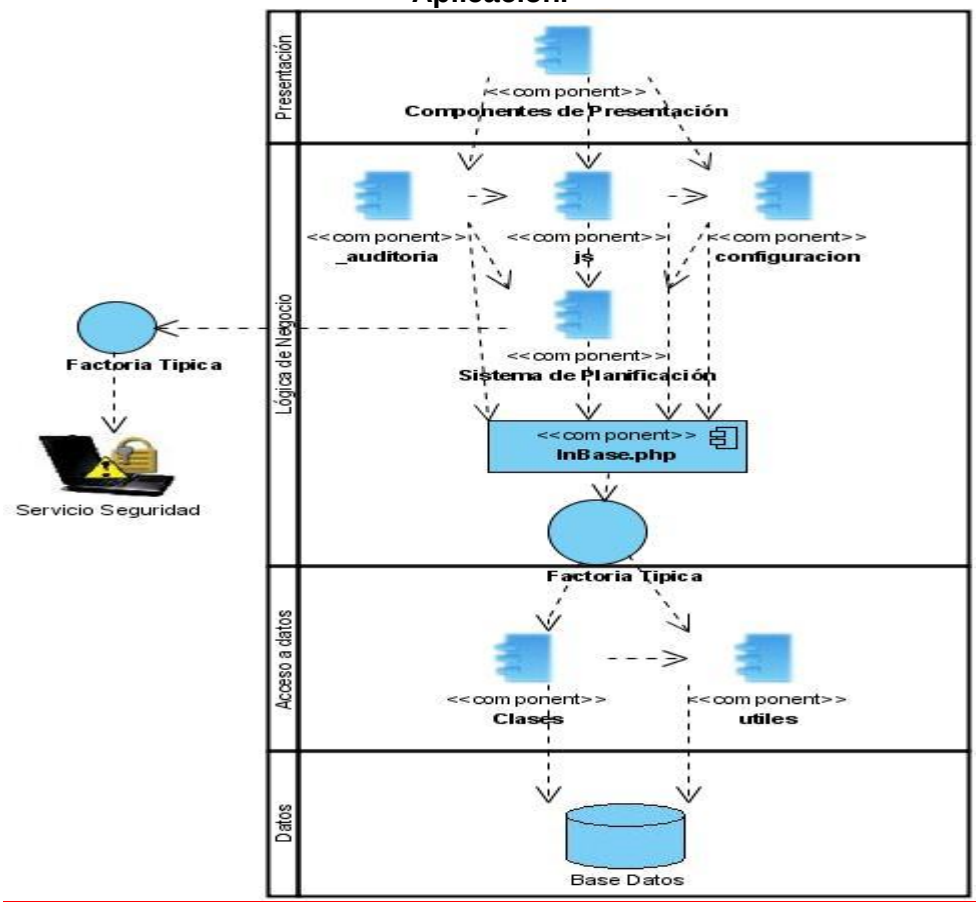


Figura. 24 Diagrama de componente para Registrar auditoría de BD.

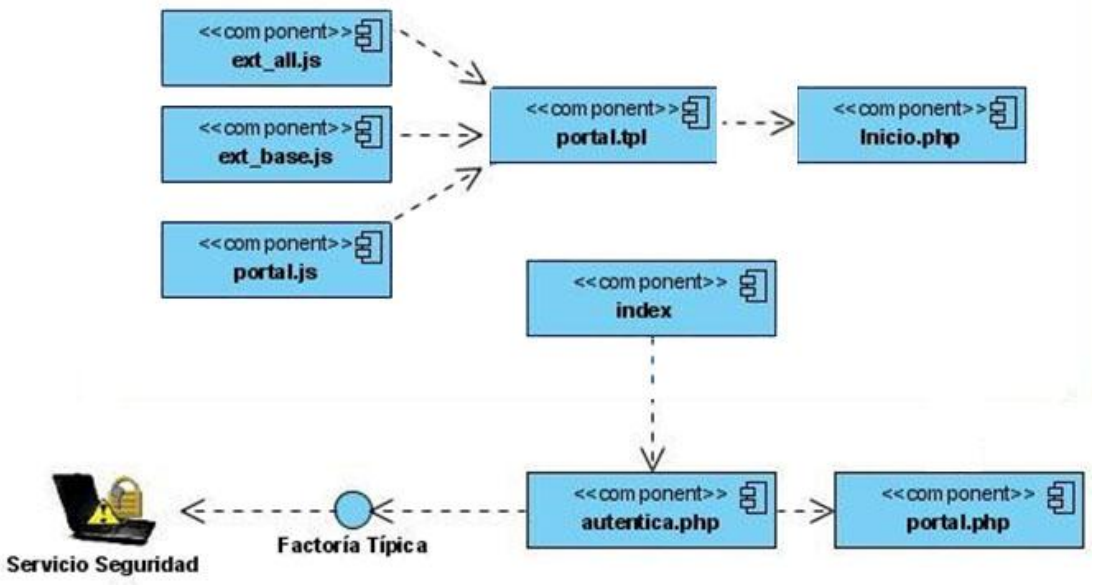


Figura. 25 Diagrama de componente Presentación.

Figura. 26 Diagrama de componente de _auditoría.

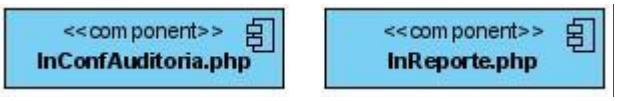


Figura. 27 Diagrama de componente de clases.

Figura. 28 Diagrama de componente de js.

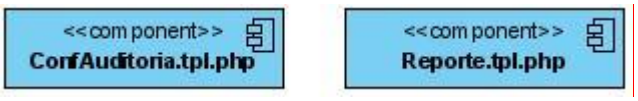


Figura. 29 Diagrama de componente de plantillas.

Figura. 30 Diagrama de componente js.

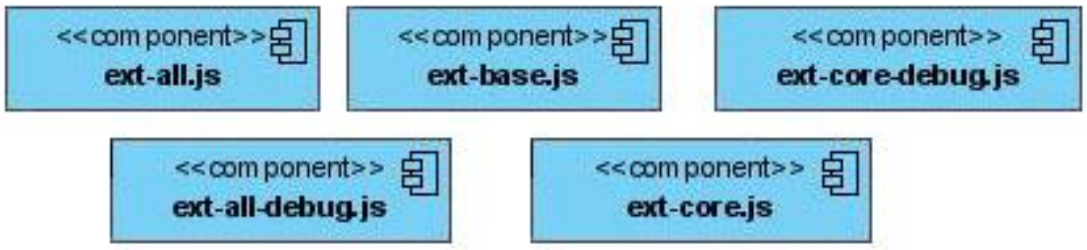


Figura. 31 Diagrama de componente yui2.

Figura. 32 Diagrama de componente de configuración.

Figura. 33 Diagrama de componente de clases.

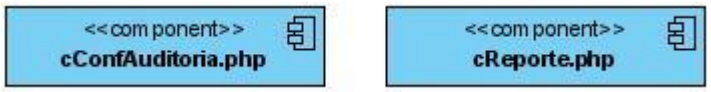


Figura. 34 Diagrama de componente de consultas.

Figura. 35 Diagrama de componente de Típicas.

Figura. 36 Diagrama de componente de útiles.

Conclusiones

En este capítulo se mostró como a través de la implementación, se produjo un refinamiento de la vista de la arquitectura del modelo de despliegue, donde los componentes ejecutables fueron asignados a nodos. Además cómo el modelo de implementación fue la entrada principal de las etapas de prueba que se realizan seguido de la implementación. Donde se verificó el resultado de esta probando cada construcción, incluyendo las versiones finales del sistema.

Se utilizaron diagramas de componentes para representar a través de un grafo los componentes de software unidos por medio de relaciones de dependencia; con los cuales se modeló la vista estática de un sistema. Además sirvieron para mostrar la organización y las dependencias lógicas entre un conjunto de componentes software. En este momento, ya se tiene el producto de software.

Conclusiones Generales

Al concluir el tránsito por las cuatro fases del desarrollo del software, transformando los requerimientos de los usuarios en un sistema software, se obtuvo un sistema (SILOGA), que permitirá el control y la supervisión de la gran cantidad de datos que genera el proceso de Planificación por la gran variedad de accesos a la información.

Con el estudio realizado e implementación del sistema propuesto se cumplió con el objetivo general de este trabajo, utilizando una metodología y lenguaje que responde a las nuevas ideas de la informatización en el Centro de Compatibilización, integración y desarrollo de productos informáticos para la defensa además de permitir una buena ejecución del proceso.

Recomendaciones

Con la realización del trabajo nos surgieron ideas las cuales recomendamos para otras iteraciones del SILOGA:

1. Brindar la posibilidad al usuario de exportar los reportes a otro formato.
2. Tener en cuenta este trabajo para otros gestores de base de datos.
3. Tener una BD para la auditoría a nivel de aplicación por si se cae la conexión poder guardar esa auditoría.

Bibliografía Citada

1. Sanger, Jimmy Wales y Larry. Wikimedia Foundation, Inc. *Wikimedia Foundation, Inc.* [Online] marzo 30, 2008. [Cited: febrero 16, 2008.] http://es.wikipedia.org/wiki/Auditor%C3%ADa_inform%C3%A1tica .

2. Wikilearning. *Wikilearning.* [Online] 27 9, 2007. [Cited: 2 16, 2008.] http://www.wikilearning.com/curso_gratis/la_auditoría-l_concepto_de_auditoría/12650-3 .

3. France Telecom España, S.A. *France Telecom España, S.A.* [Online] julio 11, 1999. [Cited: febrero 18, 2008.] http://html.rincondelvago.com/auditoría-de-los-sistemas-de-informacion.html .

4. Wikimedia Foundation, Inc. *Wikimedia Foundation, Inc.* [Online] mayo 22, 2008. [Cited: marzo 5, 2008.] http://es.wikipedia.org/wiki/Log_(registro) .

5. Application LifeCycle Solutions, S.L. *Application LifeCycle Solutions, S.L.* [Online] septiembre 2006. [Cited: marzo 6, 2008.] http://www.als-es.com/home.php?location=recursos/articulos/auditar-sistemas-informacion .

6. Serpol Software. *Serpol Software.* [Online] 2002. [Cited: marzo 8, 2008.] http://www.serpol.com/indexcp.html .

7. Free Download Manage. *Free Download Manager.* [Online] enero 7, 2008. [Cited: marzo 8, 2008.] http://www.freedownloadmanager.org/es/downloads/165_s/ .

8. Wikimedia Foundation, Inc. *Wikimedia Foundation, Inc.* [Online] abril 18, 2008. [Cited: mayo 6, 2008.] http://es.wikipedia.org/wiki/Registro_del_sistema .

9. Monografias.com S.A. . *Monografias.com S.A. .* [Online] 1997. [Cited: marzo 7, 2008.] http://www.monografias.com/trabajos13/auditor/auditor.shtml .

10. Monografias.com S.A. . *Monografias.com S.A. .* [Online] 1997. [Cited: abril 7, 2008.] http://www.monografias.com/trabajos13/auditor/auditor.shtml .

11. Ext, LLC. *Ext, LLC.* [Online] 2006. [Cited: mayo 9, 2008.] http://extjs.com/ .

Bibliografía Consultada

1. Cealys Alvarez Trujillo, Jiselle Almeida Berovides. *Sistema Informático para consolidar la Planificación Material y Financiera del MINFAR.* Ciudad de la Habana : UCI, 2007.

2. Ortiz, Yudisney Vázquez. *Análisis y diseño del sistema de información y gestión para la calidad del Software Educativo.* Ciudad de la Habana : UCI, 2007.

3. Ilieva Rodríguez Valido, Luis Manuel González Medel. *Sistema Informático de Gestión para la Planificación de los Órganos Consumidores del MINFAR.* Ciudad de la Habana : UCI, 2007.

4. Cepeda., Leevan Abon. *GENERADOR DE CÓDIGO PARA APLICACIONES EN PHP APLICADO AL ERPFAR.* Ciudad de la Habana : UCI, 2007.

5. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso unificado de desarrollo de software.* Ciudad de la Habana : Felix Varela, 2004. Vol. I. I-S-B-N.

6. —. *El Proceso unificado de desarrollo de software.* Ciudad de la Habana : Felix Varela, 2004. Vol. II. I-S-B-N.

7. Pressman, Roger S. *Ingeniería de software Un enfoque Práctico.* la Habana : Félix Varela, 2005. Vol. I.

8. —. *Ingeniería de software Un enfoque Práctico.* La Habana : Félix Varela, 2005. Vol. II.

Glosario de términos

[1] (UCID): Centro de Compatibilización, integración y desarrollo de productos informáticos para la defensa

[2] (MINFAR): Ministerio de las Fuerzas Armadas Revolucionarias.

[3] (WEB): Red Global Mundial.

[4] (SILOGA): Sistema de Log de Auditoría.

[5] (OC): Órgano Consumidor.

[6] (CB): Centros de Balance.

[7] (DE): Dirección de Economía.

[8] (OACE): Organismos de la Administración Central del Estado.

[9] (MEP): Ministerio de Economía y Planificación.

[10] (CUs): Caso de usos.

[11] (BD): Base de datos.

[12] (ERP): *Enterprise Resource Planning.*

[13] (PHP): *Personal Home Page Tools*.

[14] (AJAX): *Asynchronous JavaScript And XML*.

[15] (DOM): Modelo de Objetos de Documento.

[16] (API): Interfaz de programación de aplicaciones.

[17] (HTML): HyperText Markup Language.

[18] (XML): Extensible Markup Language.

[19] (GUI): *Graphical User Interface*.

[20] (CSS): Las hojas de estilo en cascada o *Cascading Style Sheets.*

[21] (HTTP): El protocolo de transferencia de hipertexto , HyperText Transfer Protocol.

[22] (POO): Programación Orientada a Objetos

[23] (SGBD): Sistema Gestor de Bases de Datos.

[24] (RDBMS): Sistema Gestor de Base de Datos Relacionales.

[25] (SQL): Lenguaje de Consulta Estructurado ,*Structured Query Language*.

[26] (DDL): Lenguaje definición de datos.

[27] (VDL): Lenguaje de definición de vistas.

[28] (DML): Lenguaje de manipulación de datos.

[29] (MVCC en ingles): Acceso Concurrente Multiversión.

[30](TRIGGERS): Es un disparador en una BD.

[31] (RUP): *Rational Unified Process*.

[32] (UML): Lenguaje Unificado de Modelado.

[33] (CASE): Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador.

[34] (PDF): El formato de documento portátil.

[35] (RF): Requisitos funcionales

[36] (FAR): Fuerzas Armadas Revolucionarias.

[37] (UNIX): Sistema operativo portable, multitarea y multiusuario.

[38] (LINUX): Sistema operativo UNIX.

[39] (MER): Modelos entidad-relación.