Universidad de las Ciencias Informáticas
"Facultad 4"

Título: "Determinación de Inconsistencias y Mantenimiento Automático de Base de Datos en Postgres".

Trabajo de Diploma para Optar por el Título de

Ingeniero en Ciencias Informáticas.

Autores: Yudaisy González Acosta (ygacosta@estudiantes.uci.cu).

Osmar Edel Fernández Sam (ofernandez@estudiantes.uci.cu).

Tutor: Ing. Alain Fernández Deronceré (afernandezd@uci.cu).

Ciudad de La Habana

Julio de 2007

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 _____.

Yudaisy González Acosta.

_____

Osmar Edel Fernández Sam

_____

Alain Fernández Deronceré

_____

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Determinación de Inconsistencias y Mantenimiento Automático de Base de Datos en Postgres.

Autores: Yudaisy González Acosta.
Osmar Edel Fernández Sam.

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de _____.

_____ _____

Firma Fecha

AGRADECIMIENTOS

“Es tan grande el placer que se experimenta al encontrar un hombre agradecido que vale la pena arriesgarse a no ser un ingrato”.

Sin la ayuda de ustedes esta victoria era difícil ganarla, por eso de esta manera queremos agradecerles a todas aquellas personas que hicieron posible la realización de este trabajo.

A nuestros padres, quienes nos guían siempre por el camino correcto, y que nos han enseñado a enfrentar la vida por muy dura que esta sea.

A nuestros familiares, que siempre nos están apoyando, llenándonos de amor y alegría para seguir adelante.

A mis suegros, por el gran apoyo que me han brindado durante todo este tiempo. (Yudaisy)

A nuestras parejas, por ser ellos tan pacientes, que a pesar de la distancia siempre estuvieron a nuestro lado brindándonos su amor y su energía positiva.

A nuestras amistades, por todo el cariño que nos han brindado durante estos años y por estar siempre en los momentos de flaqueza, sin ellos la vida en la universidad hubiese sido una flor sin encanto.

A nuestro tutor por depositar su confianza en nosotros y darnos la tarea de realizar este trabajo.

A nuestro eterno Fidel Castro y a la Revolución por darnos la oportunidad de convertirnos en profesionales y tener esa grandiosa idea de construir la UCI.

A todos aquellos que extendieron su mano cada vez que la necesitábamos, contribuyendo a que el presente trabajo llegara a su objetivo final.

A todos ustedes los tenemos muy dentro de nuestros corazones.

DEDICATORIA

A mi mamá Neysa, le dedico este gran triunfo de mi vida, por ser mi guía y mi faro, por trasmitirme la confianza de que en la vida todo se alcanza con esfuerzo y sacrificio.

A mi papá Alberto, también se lo dedico, por creer plenamente en que lograría este gran premio que la vida me ofrece, porque sé que está orgulloso de mí.

A mis abuelas Isabel y Dulce María, por ser siempre mis ángeles protectoras, y por apoyarme, dándome aliento para seguir adelante.

A mis hermanos Alberto y Erick, que a pesar de la distancia me han dado la oportunidad de demostrarles que la vida siempre se encarga de premiar a las personas que luchan por lo que desean.

Yudaisy.

Le dedico este triunfo a mi mamá Lourdes, a mi papá Ortelio que me ha inculcado que con amor y dedicación siempre se harán realidad mis sueños y son ejemplos a seguir.

A mi hermano Osbel y a mi tío Osmar que en todo momento me han dado su apoyo para seguir adelante.

A Cristina por darme siempre aliento en los momentos difíciles, por entenderme y por esperar pacientemente a que se cumpla este sueño hecho realidad.

A mi hija para que en el futuro le sirva de ejemplo.

A mi familia y amistades.

Osmar.

RESUMEN

Durante el proceso de gestión de datos en el proyecto de Planificación Material y Financiera del MINFAR (PMF-MINFAR) se genera un gran número de operaciones sobre la base de datos, llegando a existir incongruencias en la estabilidad de la misma, conllevando a demoras en la ejecución de procedimientos de gestión y consulta; y respuestas que difieren con la verdadera información que se encuentra almacenada, todo esto se debe a la falta de un proceso de mantenimiento periódico y regulado, dirigido a la optimización y mejora del desempeño de la base de datos de la PMF-MINFAR.

Actualmente el mantenimiento y la determinación de inconsistencias de la base de datos de dicho proyecto se realiza de una forma no automatizada, obligando a la persona responsable, tener que auxiliarse de una herramienta que le permita realizar estas operaciones de forma aislada, a la vez de que necesita acceder a dicha herramienta cada vez que vaya a ejecutar cualquiera de las tareas antes mencionadas.

Palabras Claves

incongruencias, estabilidad, gestión, consulta, mantenimiento, inconsistencias, automatizada

ÍNDICE

INTRODUCCION

La información es uno de los activos más importantes de las entidades, y de modo especial en algunos sectores de actividad. Es indudable que cada día las entidades dependen en mayor medida de la información y de la tecnología, y que los sistemas de información están más soportados por la tecnología, frente a la realidad de hace pocas décadas.

Por otra parte, hace unos años la protección era más fácil, con arquitecturas centralizadas y terminales no inteligentes, pero hoy día los entornos son realmente complejos, con diversidad de plataformas y proliferación de redes, no sólo internas sino también externas, incluso con enlaces internacionales.

Entre las plataformas físicas ("hardware") pueden estar: ordenadores grandes y medios, ordenadores departamentales y personales, solos o formando parte de redes, e incluso ordenadores portátiles. Esta diversidad acerca la información a los usuarios, si bien hace mucho más difícil proteger los datos, especialmente porque los equipos tienen filosofías y sistemas operativos diferentes, incluso a veces siendo del mismo fabricante.

Al hablar de seguridad se centra en la información misma, aunque a menudo se hable de seguridad informática, de seguridad de los sistemas de información o de seguridad de las tecnologías de la información.

En cualquier caso hay tres aspectos principales, como distintas vertientes de la seguridad:

La confidencialidad: Se cumple cuando sólo las personas autorizadas (en un sentido amplio se refiere también a sistemas) pueden conocer los datos o la información correspondiente. Se puede preguntar ¿qué ocurriría si un soporte magnético con los datos de los empleados o clientes fuera cedido a terceros? ¿cuál podría ser su uso final? ¿habría una cadena de cesiones o ventas incontroladas de esos datos, que podrían incluir datos como domicilios o perfil económico, o incluso datos médicos? ¿Y si alguien se hiciera con un "disquete" con lo que perciben los directivos de nuestra entidad?

La integridad: consiste en que sólo las personas autorizadas puedan variar (modificar o borrar) los datos. Además deben quedar pistas para control posterior y para auditoría. Pensar que alguien variara datos de forma que se pierda la información de determinadas deudas a cobrar (o que sin perderla se tuviera que recurrir a la información en papel), o que modificara la última parte de los domicilios de algunos clientes.

Algunas de estas acciones se podrían tardar en detectar, y tal vez las diferentes copias de seguridad hechas a lo largo del tiempo estarían "viciadas" (corruptas se dice a veces), lo que haría difícil la reconstrucción.

La disponibilidad: se cumple si las personas autorizadas pueden acceder a tiempo a la información. El disponer de la información después del momento necesario puede equivaler a la no disponibilidad.

Otro tema es disponer de la información a tiempo pero que ésta no sea correcta, e incluso que no se sepa, lo que puede originar la toma de decisiones erróneas.

Otro caso grave es la no disponibilidad absoluta, por haberse producido algún desastre. En ese caso a medida que pasa el tiempo el impacto será mayor, hasta llegar a suponer la no continuidad de la entidad, como ha pasado en muchos de los casos producidos (más de un 80% según las estadísticas), ya que las incidencias son frecuentes.

Todo lo anteriormente expuesto confirma que se necesita de un sistema que le proporcione un mantenimiento automático a las Bases de Datos; y así solucionar parte del problema relacionado con la pérdida de la información en las entidades, trabajar además con una información que no es real dentro de las mismas.

En el proceso de gestión de la información de la base de datos del proyecto PMF-MINFAR se encuentran muchas adversidades e irregularidades en el sistema. La generación de un considerable número de operaciones en la misma y la no existencia de un proceso de mantenimiento, hace que existan diferencias e inestabilidad en los datos, y debido a ello, el tiempo de ejecución de las consultas y gestión se hace más extenso, dicha respuesta de los procesos anteriores no coinciden con la información originalmente almacenada, es decir, existe problemas con la pérdida de la información dentro del servidor, provocado por la poca o ninguna ejecución de procesos de optimización de la base de datos.

El proyecto PMF-MINFAR se encuentra en una etapa donde se quiere automatizar una serie de operaciones y de servicios para así facilitarle y brindarle un trabajo más cómodo tanto a los administradores del sistema como a los usuarios del mismo. Una de las acciones a automatizar es el mantenimiento y la determinación de inconsistencias de la Base de Datos del proyecto PMF-MINFAR, acciones que actualmente se realizan de manera aislada, es decir, cada vez

que se va a ejecutar una de estas operaciones hay que abrir nuevamente la aplicación que permite realizarlas.

Logrando tal objetivo, se brinda al sistema una mayor facilidad para realizar dichas actividades en la Base de Datos, además de proporcionarle una fiabilidad extrema al mismo, a la vez que el usuario realice dichas funcionalidades de una forma automatizada, y ya así no depender de aquellas herramientas externas al sistema para poder ejercer las acciones mencionadas.

Teniendo como base todo lo anteriormente expuesto, el **problema a resolver** reside en: ¿Cómo mejorar el proceso de mantenimiento y determinación de inconsistencias de la base de datos de Planificación Material y Financiera del MINFAR?

El **objeto de estudio** del presente trabajo está enmarcado en los procesos que se realizan en Base de Datos sobre Postgres, definiendo como **campo de acción** el proceso de mantenimiento y determinación de inconsistencias de la Base de Datos de PMF-MINFAR.

Se trabaja en tal proceso con el **objetivo general** de automatizar el proceso de mantenimiento y determinación de inconsistencias en la Base de Datos del proyecto de PMF-MINFAR.

Definiéndose los siguientes **objetivos específicos**:

1) Obtener el modelo del dominio.
2) Obtener el análisis, diseño e implementación del sistema a realizar.
3) Conocer los resultados de las pruebas al sistema.
4) Obtener la documentación del trabajo de diploma.

Para darle cumplimiento a los objetivos específicos, se plantean las siguientes **tareas**:

1) Realizar un estudio de los sistemas que tengan funcionalidades de mantenimiento de Base de Datos en Postgres.
2) Modelar el dominio.
3) Desarrollar el análisis, diseño e implementación al sistema.
4) Realizar las pruebas necesarias al sistema.
5) Realizar la documentación del trabajo de diploma.

Para la realización del presente trabajo, se ha planteado la siguiente **hipótesis**: Realizando el proceso de mantenimiento y determinación de inconsistencias a través de un sistema automatizado, se logra una mejora en el rendimiento de la Base de Datos del proyecto PMF-MINFAR.

CAPITULO 1. FUNDAMENTACION TEORICA.

1.1 Introducción

En el capítulo se abordan algunos conceptos necesarios para el desarrollo de sistemas con un mantenimiento automático de Bases de Datos en Postgres tales como: qué es una Base de Batos, PostgreSQL. Se hace referencia además a los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS, s) y la importancia de estos al tener una buena calidad en los datos.

También se explica en qué consiste el Demonio, proceso que se lleva en los servidores Web, y se cita algunos derivados de dicho proceso.

Se hace alusión a los lenguajes de programación a utilizar en dicho trabajo, donde se mencionan sus características y ventajas que brindan al trabajar con los mismos.

Se da una explicación acerca del uso de la metodología para el análisis, diseño y desarrollo de sistemas informáticos y de ahí se propone la misma a seguir para el desarrollo de la aplicación del presente trabajo.

1.2 Base de datos.

Una base de datos o banco de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de base de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo en España, los datos personales se encuentran protegidos por la Ley Orgánica de Protección de Datos de Carácter Personal.((FSF), 2008)

1.3 PostgreSQL

PostgreSQL [1], es un proyecto open source, o de código abierto. Código Abierto por definición significa que se puede obtener el código fuente, usar el programa, y modificarlo libremente sin las limitaciones del software propietario. En el mundo de las bases de datos, código abierto significa que se tiene acceso real a números de **benchmarking** [2] y estadísticas de rendimiento, datos que otras compañías como Oracle no facilitan. Código abierto también significa que se es libre de modificar PostgreSQL para adaptarlo a las necesidades particulares.

Sin embargo, hay un malentendido en cuanto a que si bien el software de código abierto está libre de restricciones del distribuidor, sea siempre carente de costes para su compañía. Esto no es necesariamente el caso. Es cierto en cuanto a que se puede, sin coste alguno, descargar e instalar software de código abierto, pero siempre existirán costes asociados con el tiempo y la energía que la compañía dedique a soportar y adaptar la aplicación. Dicho esto, si no se dispone de esos recursos, existen varias distribuciones comerciales, así como consultores que trabajan específicamente con PostgreSQL.(WORSLEY y DRAKE, 2001)

PostgreSQL es sin duda el Sistema de Gestión de Bases de Datos de código abierto (gratuito y con código fuente disponible) más avanzado del mundo. Posee las características de los más potentes sistemas comerciales como Oracle o SQL Server. Entre ellas se pueden destacar:

- Completo soporte para transacciones. Una transacción está formada por un conjunto de acciones de forma que o se ejecutan todas ellas o bien ninguna. Utilizando transacciones aseguramos la consistencia de los datos.
- Soporte completo ACID (Atomicity Consistency Isolation Durability):
    - Es posible definir operaciones atómicas, es decir, formadas por comandos que se ejecutan todos o ninguno.

- Consistencia, que garantiza que la base de datos nunca se queda en un estado intermedio de una transacción (con parte de los comandos ejecutados y parte no).
- Aislamiento, que mantiene separadas las transacciones de usuarios distintos hasta que éstas han terminado.
- Durabilidad, garantiza que el servidor de datos guarda las actualizaciones pendientes de forma tal que pueda recuperarse de una terminación brusca tal como desenchufar la máquina. Esta característica se implementa mediante el log de transacciones, que almacena todas las transacciones que aún no han sido lanzadas (**commit** [3]).

- Procedimientos almacenados. Código ejecutable que se almacena compilado en el servidor. Entre otras cosas, permite optimizar las aplicaciones evitando transferencias innecesarias a través de la red (aplicaciones con parte cliente y parte servidor). Los procedimientos almacenados se pueden escribir con su propio lenguaje de programación (PL/pgSQL) o bien en Perl o TCL (**T**ool **C**ommand **L**anguage" o *lenguaje de herramientas de comando*).
- Soporte para construcciones SQL del tipo **subselect** [4].
- Triggers. Procedimientos almacenados que se lanzan automáticamente bajo determinadas circunstancias como actualizaciones de campos. Permite establecer reglas de integridad y consistencia a nivel del servidor de base de datos.
- Vistas. Conjunto de registros resultado de una consulta que se comportan como una tabla física para facilitar su manejo.
- Orientación a objetos con herencia de tablas.

Para hacerse una idea del prestigio y la solvencia de este sistema de bases de datos, basta decir que la empresa Afilias que gestiona los dominios .info y la parte técnica de los .org utiliza la versión estándar de PostgreSQL para almacenar todos los dominios .info y .org registrados a nivel mundial.

1.3.1. Gestión de bases de datos PostgreSQL

Las bases de datos de PostgreSQL no son archivos que se pueda subir a un sitio Web como los de Access, sino que residen en un servidor de datos separado. Por ello se debe utilizar algún programa cliente que permita conectarse al servidor de datos con el fin de crear las tablas, subir datos, editar registros, etc. Aunque PostgreSQL está en un servidor Linux, se puede gestionar las bases de datos desde ordenadores con cualquier sistema operativo utilizando las aplicaciones adecuadas.

1.4. Sistemas de mantenimiento de Bases de Datos relacionales tradicionales

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS, s) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos.

El modelo relacional sustituyó modelos previos en parte por su “simplicidad espartana”. Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema:

clases
herencia
tipos
funciones

Otras características aportan potencia y flexibilidad adicional:

Restricciones(Constraints)
Disparadores(triggers)
Reglas(rules)
Integridad transaccional

Estas características colocan a Postgres en la categoría de las Bases de Datos identificadas como objeto-relacionales. Nótese que éstas son diferentes de las referidas como orientadas a objetos, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales.

Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera.(POSTGRESQL, 2007)

1.5. Estado del Arte.

El mundo que vivimos actualmente es muy diverso, existe una gran variedad de cambios tecnológicos que son muy ventajosos y a la vez facilita a las personas que interactúan con estas tecnologías a un trabajo más cómodo y menos engorroso. Lo que antes se hacia manual, ahora existen sistemas informáticos muy avanzados que lo sustituye. Anteriormente, los datos de las empresas, personas, servicios, negocios, se archivaban en papeles que a la vez que se actualizaban necesariamente tenían que utilizar hojas nuevas para la información renovada, y así sucedía cuando se cometía errores; las reglas y las restricciones estaban en las mentes de las personas involucradas en dicho proyecto, esto causaba algunos problemas ya que las validaciones del mismo ya dependían de la conciencia de esas mismas personas. Actualmente, son muy pocas las empresas, negocios y servicios que no se encuentran automatizados, todo este proceso de automatización trae consigo la utilización de las Bases de Datos, que como su nombre lo indica, ahí se almacenan los datos del proyecto que se este llevando a cabo.

Estas mismas Bases de Datos necesitan actualizarse periódicamente y tener un mantenimiento automático para que a su vez sean más consistentes. En este Trabajo de Diploma se quiere realizar un sistema para darle un mantenimiento automático a toda Base de Datos, pero para ello se necesita realizar un profundo estudio sobre los procesos internos del sistema operativo para la realización de un gran numero de funciones en la Base de Datos, como son las salvas que se realizan automáticamente, en este caso se hace referencia al Demonio, y ya así con ello llegar al propósito de dicho Trabajo de Diploma.

1.5.1. Demonio

Un demonio, es un tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo). Este tipo de programas se ejecutan de forma continua (infinita), vale decir, que funciona sin tener relación con una terminal o consola y, consecuentemente, sin interactuar con un humano.

El origen de la palabra daemon (demonio) se encuentra en la antigua Grecia y la figura del daemon, un espíritu interior, equivalente a un "ángel protector" que guiaba y protegía a los hombres.

Suelen tener las siguientes características:

- No disponen de una interfaz directa con el usuario, ya sea gráfica o textual.
- No hacen uso de la entradas y salidas estándar para comunicar errores o registrar su funcionamiento, sino que usan archivos del sistema en zonas especiales (/var/log/ en los UNIX más modernos) o utilizan otros demonios especializados en dicho registro como el syslogd.

Por ejemplo, una máquina que alberga un servidor web utilizará un demonio httpd (HTTP Daemon) para ofrecer el servicio y que los visitantes a dicha web puedan acceder. Otro ejemplo son los demonios "cronológicos" como cron que realizan tareas programadas como mantenimiento del sistema en segundo plano.

1.5.1.1. El demonio syslogd

El demonio syslogd (*Syslog Daemon*) se lanza automáticamente al arrancar un sistema **Unix** [8], y es el encargado de guardar informes sobre el funcionamiento de la máquina. Recibe mensajes de las diferentes partes del sistema (núcleo, programas...) y los envía y/o almacena en diferentes localizaciones, tanto locales como remotas, siguiendo un criterio definido en el fichero de configuración /etc/syslog.conf, donde se especifican las reglas a seguir para gestionar el almacenamiento de mensajes del sistema. Las líneas de este archivo que comienzan por `#' son comentarios, con lo cual son ignoradas de la misma forma que las líneas en blanco; si ocurriera un error al interpretar una de las líneas del fichero, se ignoraría la línea completa. (HUERTA, 2006)

1.5.1.2. Cron (Unix)

Cron es el nombre del programa que permite a usuarios Linux/Unix ejecutar automáticamente comandos o scripts (grupos de comandos) a una hora o fecha específica. Es usado normalmente para comandos de tareas administrativas, como respaldos, pero puede ser usado para ejecutar cualquier cosa. Como se define en las páginas del manual de cron (#> man cron) es un demonio que ejecuta programas agendados.

Prácticamente todas las distribuciones de Linux usa la versión Vixie Cron, por la persona que la desarrolló, que es Paul Vixie, uno de los grandes **gurús** [9] de Unix, también creador, entre otros sistemas, de BIND que es uno de los servidores DNS más populares del mundo.

Cron es una utilidad de sistema que sirve para lanzar procesos con una periodicidad determinada, como por ejemplo copias de seguridad u otro tipo de procesos que deben ser lanzados de forma desatendida.

El paquete Cron provee dos utilidades, el *demonio cron* propiamente dicho y el editor de tareas, crontab, que es la herramienta más importante.(DURAN, 2005-2008)

1.6. 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. Java Script.

Es un lenguaje de Script, interpretado por el navegador y orientado a objeto. Aunque tiene menos capacidad que un lenguaje totalmente orientado a objetos como C++, Pascal, PHP y Java. Esta diseñado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador Web. Este fue el lenguaje seleccionado para validar los formularios en las páginas Web que conforman las Interfaces de la aplicación por su portabilidad y funcionalidad.

1.8. Ext 2.0.

Ext 2.0 es un framework completo y extremadamente avanzado. Este framework esta basado completamente a la programación Orientada a Objeto. Cada objeto contiene lo típico: propiedades, métodos, eventos, etc. Ext basa toda su funcionalidad en JS a través de librerías ya muy conocidas: YUI, jQuery y Prototype/Script.aculo.us. Así, en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de DOM. Ventanas, mensajes emergentes, grillas, date pickers y un sin numero de utilidades son todas creadas en tiempo de ejecución. Los datos son obtenidos con mucho AJAX a través de XML y/o JSON.

Ventajas:

- La orientación a objetos intensa hará modular todos los scripts.
- El diseño está completamente separado de la funcionalidad.
- Funciones comunes como validación, comboxes editables, ventanas arrastables (con minimizar y maximizar), grillas editables, son muy fáciles de implementar.
- Buena y amplia documentación, así como también su comunidad.

Desventajas:

- Crear un sistema serio con esta herramienta requiere un previo uso prolongado, ya que se pierde con muchos nuevos objetos en su extensa y bien documentada API.
- El tiempo de aprendizaje puede llegar a compararse con a aprender a programar en un lenguaje nuevo.

Al estar todo el sitio en JS, no podrá ser accesible para los buscadores, limitando su uso a sistemas y no sitios web.

1.9. PHP.

El lenguaje PHP es un lenguaje de programación de estilo clásico, quiere decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones. No es un lenguaje de marcas como podría ser **HTML** [10], **XML** [11]. Está más cercano a Java Script o a C.

Pero a diferencia de Java o Java Script que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo

podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una pagina XML.

Figura 1. Ejecución de PHP.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

PHP es el acrónimo de "PHP: Hipertexto Preprocessor", es decir, un "preprocesador del hipertexto" que se basa en una sintaxis parecida al C, al Java y al Perl, por supuesto con unos añadidos más. Nació en 1994 como proyecto "personal" y la primera versión se utilizó públicamente en 1995 con el nombre "Personal Home Page".

¿Por qué PHP?

Para el desarrollo de la aplicación Web se debe tener en cuenta cuestiones esenciales como: garantizar que el sistema 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** [12].
- 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. Metodologías.

El uso de una metodología para la elaboración de un producto informático, garantiza determinadas características del mismo, la calidad, es un factor clave tanto para el cliente como para el productor, el tiempo uno de los factores que puede afectar todo el producto, mucho más si no se ha hecho un buen estudio de la aplicación que se va a realizar, la cantidad de personal, muchas veces en exceso, otras en déficit, los sistemas de organización, los métodos de control, el dominio sobre el tema y sobre las herramientas de desarrollo por parte de los analistas y los programadores, la falta de conocimientos sobre asuntos informáticos por el lado de los clientes, son factores que pueden afectar todo el ciclo de desarrollo de una aplicación, todas estas actividades podemos resumirlas en el proceso de desarrollo del software, que no es más que el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software.

1.10.1. Metodología para el desarrollo unificado, Rational Unified Process (RUP)

Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos, esta basado en el desarrollo iterativo y el modelamiento visual UML para describir un sistema, lo cual permite incorporar al proceso de

desarrollo de software un mejor control de los requerimientos y cambios. Posibilita la distribución del trabajo en diversos frentes de forma simultánea.

A pesar de ser una metodología desarrollada directamente para el trabajo con clases y objetos brinda amplias posibilidades con el manejo eficiente del tiempo de diseño e implementación de aplicaciones Web.

Hay que destacar que el **RUP** [14] capacita a las organizaciones de muchas maneras, una de la más significativa es que proporciona la forma en la que el equipo de proyecto puede trabajar de una forma más conjunta con los clientes y demás implicados. Lo que favorece una mayor organización y entendimiento de lo que realmente el cliente necesita y una excelente proyección del proyecto.

RUP identifica las 6 mejores prácticas con las que define una forma efectiva de trabajar para los equipos de desarrollo de software, estas son:

• Gestión de los Requisitos.

• Desarrollos iterativos.

• Uso de arquitecturas basadas en componentes.

• Desarrollo Visual del Software (con UML).

• Verificación continúa de la calidad del software.

• Gestión de los Cambios.

Las fases que lleva la metodología RUP en el ciclo de vida de un proyecto son:

• Fase de concepción o inicio.

• Fase de elaboración.

• Fase de construcción.

• Fase de transición.

Características que identifica a *RUP*

• ***Manejado por Casos de Uso.***

Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Representan requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema. Los casos de uso guían la arquitectura del sistema, y la arquitectura del sistema influye en la selección de los casos de uso.

• ***Su centrado en la arquitectura.***

La arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción. Este concepto incluye los aspectos estáticos y dinámicos más significativos del sistema. Esta se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por si sola.

• ***Es iterativo e incremental.***

RUP propone dividir el trabajo en partes más pequeñas o miniproyectos, donde cada miniproyecto 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 miniproyectos.

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. (JACOBSON, BOOCH, & RUMBAUGH, 2000)

1.11. Lenguaje de Modelado.

El Lenguaje Unificado de Modelado **(UML**) [13] 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. Herramienta Case. Visual Paradigm

Las Herramientas **CASE** [16] 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.

El Visual Paradigm para UML es una herramienta CASE visual que ayuda a construir aplicaciones rápidamente, mejor y económicamente. Es una herramienta de desarrollo que se integra al **IDE** [15] (*Integrated Development Environment o Entorno de Desarrollo Integrado*) de Eclipse. Diseñada para desarrollar software con OOP (Programación Orientada a Objetos), busca reducir la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores.

Beneficios:

1. Navegación intuitiva entre el modelo visual y el código.

2. Poderosa herramienta de generación de PDF/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.

Rasgos:

• Generales**:**

1. Todos los diagramas UML.

2. Soporta UML 2.0.

3. Mecanismos de extensión de UML.

4. Soporte robusto para la notación de análisis.

5. Repositorio de clases: clases y paquetes pueden ser compartidos.

6. Chequeador de Modelos.

7. Diagramas de capas automáticos.

• Modelado de clases e ingeniería de código:

1. Control de visibilidades en los archivos compartidos.

2. Sincronización automática de código.

3. Generación de código.

4. Ingeniería inversa de código.

• Reversa instantánea:

Genera modelos VP-UML instantáneamente a partir de código binario .NET.

• Modelado básico:

1. Recursos del Visual Paradigm.

2. Colores para los elementos de los diagramas.

3. Conectores 'conectables' y configurables.

4. Estilos múltiples de movimiento del estado.

5. Hitos visuales para acciones válidas o inválidas del usuario.

6. Facilidades para copy paste de elementos.

7. Edición de texto en diagramas UML.

8. Edición de propiedades de diagramas fácilmente.

9. Los packages pueden contener elementos UML.

• Modelado Avanzado:

1. Generador de Modelos.

2. Generación automática de diagramas a partir de otros diagramas.

• Interfaz de Usuario:

1. Permite diferentes perspectivas del proyecto.

2. Explorador de proyecto.

• Análisis y modelado de Casos de Uso:

1. Análisis de texto a nivel de proyecto y a nivel de casos de uso.

2. Crea un modelo a partir de clases candidatas.

3. Descripción de casos de uso.

4. Flujo de eventos.

5. Planificador de Casos de Uso (manual).

6. Soporte para extensión de casos de uso.

• Generación de documentación:

1. Generación de HTML.

2. Generación de **PDF** [17].

• Exportaciones e importaciones:

1. Exporta diagrama a imagen.

2. Exporta proyecto a XMI e inversa.

3. Importa Racional Rose Project.

Usos en CBD (Central Business District)**:**

• En la etapa de definición de requerimientos:

1. Diagramas de clases para el Modelo Conceptual del Negocio.

2. Selección de clases candidatas para el Modelo Conceptual del Negocio.

3. Generación de Modelo Conceptual del Negocio a partir de clases candidatas.

4. Diagramas de actividad para describir los procesos de negocio.

5. Diagramas de casos de uso para los Modelos de Casos de Uso.

• En la etapa de especificación de componentes:

1. Ordenar por importancia los casos de uso.

2. Diagrama de clases para el Modelo de Escritura del Negocio, Especificación de

Interfaz y Diagramas de Responsabilidad.

3. Diagramas de colaboración para Diagramas de Iteración de Componentes.

4. Generación de diagramas a partir de otros.

5. Diagrama de componentes para los Diagramas de Especificación de

Componentes y Diagramas de Componentes de Arquitectura.

Ventajas:

1. Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.

2. Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.

Desventajas:

• No esta enfocado a CBD, por tanto herramientas como la generación automática de código no son aplicables. Cualquier diagrama generado automáticamente a partir de otro diagrama podría perder su utilidad.

Mecanismo de persistencia:

• Al igual que otros plugins permite exportar e importar archivos XMI que lo hacen portable con otras herramientas.

Comentarios de Testeo:

• Se encontraron problemas al testear la ingeniería inversa. Al querer pasar de código a diagrama de clases se perdieron algunas asociaciones.

• En el rankeo de procesos sólo facilita una tabla que se debe llenar con los procesos y el puntaje, no lo ordena automático al darle los puntajes.

• Ofrece la utilidad de colorear los diferentes elementos en los diagramas lo que facilitó la comprensión y legibilidad de los mismos.

• El plugin viene con un completo tutorial donde se muestra mediante screenshots como utilizar las herramientas del plugin. Fácil de utilizar y muy completo.

• Es un tanto lento al ejecutar.

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 8.

Dreamweaver 8 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 8, 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.

1.14. Conclusiones.

Ya se han visto las características de diferentes herramientas, para la creación de un software o aplicación así como algunas metodologías. Pero para el desarrollo de dicha aplicación Web, se debe enfocar hacia el desarrollo software libre, debido a las condiciones que enfrenta nuestro país por el bloqueo. Después de este análisis se quiere dejar plasmado que el lenguaje de programación utilizado fue el PHP por sus características antes mencionadas, así como el Javascript para el control del las diferentes funciones a realizar en el cliente, como gestor de base de datos se estableció el PostgreSQL.

En la selección de la metodología se inclina por la modelación del proceso unificado con *RUP*, por ser un proceso iterativo e incremental, que permite dirigir el proceso por roles y tener un mejor rectoreo de la calidad en cada etapa del proceso, la herramienta usada para ello es el VisualParadimg por las características antes mencionadas.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

En el campo del software también resulta útil la creación de modelos que organicen y presenten los detalles importantes de problemas reales que se vinculan con el sistema informático a construir.

Como se ha expresado anteriormente el objetivo del Proceso Unificado, es guiar a los desarrolladores de cualquier sistema software en la implementación y distribución eficiente de sistemas que se ajusten a las necesidades de los clientes.

En el presente capítulo, se dará una propuesta de solución de lo que funcionalmente debe realizar el sistema para erradicar lo problemas existentes anteriormente mencionados. Se transitará a lo largo de la fase de inicio por cada uno de los flujos de trabajo desde el negocio hasta el sistema. Todo esto se desarrollará por medio de un conjunto de artefactos y modelos resultantes de la aplicación de la metodología de desarrollo de software. Estos modelos deben cumplir una serie de propiedades, entre ellas la de ser coherentes y relacionados.

2.2. Propuesta del sistema.

Para ejecutar los procesos de mantenimiento y determinación de inconsistencias de una base de datos, el sistema debe permitir inicialmente al usuario, hacer una selección de la base datos y tablas, a las cuales se les va a aplicar los procesos anteriores, además el sistema permitirá que estas acciones previamente mencionadas puedan ser realizadas de forma automática, en una fecha y hora especificada.

El sistema será capaz de permitir la creación y aplicación de reglas por parte del usuario sobre la base de datos, dichas reglas pueden ser aplicadas solo una vez, ser guardadas por el sistema o simplemente insertadas dentro de la base de datos de forma permanente.

2.3. Modelo de Dominio.

¿Por qué Dominio y no Negocio?

La gestión que se hace en la Base de Datos del proyecto PMF-MINFAR, no presenta procesos de negocio definidos con fronteras bien establecidas, donde se logran ver claramente quiénes son las personas que los inician, los beneficiados con cada uno de estos procesos y quiénes

desarrollan las actividades en cada uno de ellos. Por las mencionadas razones se decidió crear un modelo de dominio que contenga los conceptos y definiciones que describen el negocio.

El modelo del dominio o modelo conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo, su objetivo es comprender y describir las clases más importantes dentro del contexto del sistema.

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software u objetos software con responsabilidades, el modelo del dominio podría considerarse como un diccionario visual de las abstracciones relevantes, vocabulario del dominio e información del dominio.

2.3.1. Diagrama de Clases del Dominio.

Aprovechando las bondades de los diagramas UML para representar cosas y conceptos, el diagrama del modelo del dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema analizado.

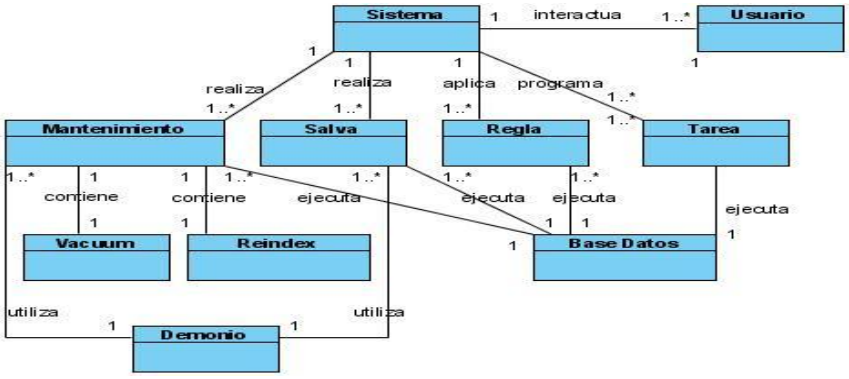


Figura 2: Diagrama de Clases del Dominio

2.3.2. Glosario de términos.

El glosario y el modelo del dominio ayudan a los usuarios, clientes, desarrolladores, y otros interesados a utilizar un vocabulario común. La terminología común es necesaria para compartir el conocimiento con los otros. Cuando abunda la confusión, el proceso de ingeniería se hace difícil. Para construir un sistema software de cualquier tamañoo, los ingenieros de hoy en día deben fundir el lenguaje de todos los participantes en uno solo consistente.

Sistema: Interfaz, a través de la cual, el usuario puede realizar acciones en la Base de Datos.

Base Datos: Lugar donde se encuentran almacenados los datos del proyecto.

Usuario: Trabajador del proyecto que tenga el rol de Administrador de la Base de Datos.

Mantenimiento: Proceso que se realiza a la Base de Datos para que funcione correctamente.

Salva: Proceso que se realiza a la Base de Datos para evitar la pérdida de la información.

Vacuum: Proceso de limpieza que se encarga de recuperar el espacio de disco ocupado por filas modificadas o borradas, actualizar las estadísticas usadas por el planificador, y evitar la pérdida de datos muy antiguos.

Reindex: Es una tarea, la cual se encarga de actualizar los index de las tablas de la Base de Datos.

Regla: Condición o restricción que debe cumplir un campo de una tabla en la Base de Datos.

Demonio: Proceso que se ejecuta en segundo plano, para realizar un mantenimiento a la Base de Datos, sin ser controlado por el usuario.

Tarea: Es una actividad donde se programan las operaciones seleccionadas por el usuario.

2.3.3. Modelo de objetos.

Un modelo de objetos es un modelo que describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo.

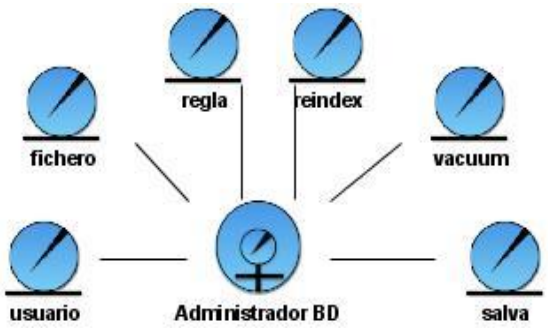


Figura 3: Modelo de Objetos

2.4. Requerimientos

Una de las principales tareas en el ciclo de desarrollo de un sistema es la de determinar los requerimientos del sistema de información (es decir, las condiciones o capacidades que el sistema debe cumplir). 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.

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

2.4.1. Requisitos Funcionales.

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

Todas estas acciones se convierten en requisitos funcionales y de acuerdo a los objetivos propuestos el sistema debe ser capaz de:

R1. Autenticar Usuario.

R2. Realizar Mantenimiento.

R 2.1. Realizar Vacuum.

R 2.2. Realizar Reindex.

R3. Realizar Salva.

R4. Gestionar Regla.

R 4.1. Crear Regla.

R 4.2. Modificar Regla.

R 4.3. Eliminar Regla.

R 4.4. Buscar Inconsistencia.

R5. Gestionar Tareas Vacuum.

R 5.1. Crear Tarea.

R 5.2. Modificar Tarea.

R 5.3. Eliminar Tarea.

R6. Gestionar Tareas Reindex.

R 6.1. Crear Tarea.

R 6.2. Modificar Tarea.

R 6.3. Eliminar Tarea.

R7. Gestionar Tareas Salvas.

R 7.1. Crear Tarea.

R 7.2. Modificar Tarea.

R 7.3. Eliminar Tarea.

2.4.2. Requisitos No Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales 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: La interfaz a implementar debe ser sencilla, de fácil uso y con rápida respuesta del sistema. Contendrá un menú dinámico para que actúe con el usuario en dependencia de sus necesidades. Ddebe estar concebido para simular una aplicación Web. Los colores a usar deben ser refrescantes para la vista, colaborando con los requerimientos medio ambientales y siguiendo los patrones de colores representativos de la entidad como: gris, blanco y azul.

Usabilidad: Debe tener una opción de ayuda sobre las principales operaciones que se realizan y sus iconos respectivos para lograr un menor tiempo de aprendizaje. Debe facilitar, principalmente, el ser manejado por usuarios que estén vinculados al proceso de Planificación Material y Financiera del MINFAR (PMF-MINFAR).

Rendimiento: Esta aplicación está concebida para un ambiente cliente/servidor así que los tiempos de respuestas deben ser generalmente rápidos, al igual que la velocidad de procesamiento de la información. El tiempo de búsqueda de información debe ser en el menor tiempo posible pues se deben generar pantallas dinámicas, implicando esto que el acceso a la base de datos debe ser lo más disponible, rápido y consistente posible.

Soporte: Para el servidor de aplicaciones: Se requiere que esté instalado un intérprete de ficheros PHP 5.x y con las últimas actualizaciones del lenguaje. Para el servidor de base de datos: Se requiere que esté instalado un gestor de base de datos que soporte grandes volúmenes de datos, maneje la concurrencia y transacciones. Para el cliente: Se requiere que esté instalado un navegador que interprete Javascript y versiones HTML 3.0 o superior.

Portabilidad: Requiere de un sistema operativo el cual soporte el navegador Mozilla Firefox.

Requerimientos de Hardware:

Para las computadoras del cliente:

- Se requiere que tengan tarjeta de red.
- Se requiere que tengan al menos 64 MB de memoria RAM.
- Se requiere al menos 100MB de disco duro.
- Procesador 512 MHz como mínimo.

Para los servidores:

- Se requiere tarjeta de red.
- Se requiere que tenga al menos 256MB de RAM.
- Se requiere al menos 1GB de disco duro.
- Procesador 1.2 GHz como mínimo.

Requerimientos de Software: El sistema se desarrollará con tecnología PHP versión 5.0. Correrá sobre un servidor con el sistema operativo UNIX (Linux). 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.

Seguridad: El sistema debe comunicarse usando un protocolo **HTTP** [18]. Chequear si el usuario que está accediendo al sistema esta autenticado y brindarle servicio de autenticación. Mostrar las operaciones de acuerdo al rol del usuario y no más. Mantener la integridad de la información, es decir que no se perderá durante su almacenamiento o transporte. Permitir que cuando se borre cualquier información pueda existir una opción de advertencia antes realizar la acción.

Disponibilidad: El sistema deberá estar disponible las 24 horas del día para el usuario con derecho a utilizarlo.

Confiabilidad: La información manejada por el sistema está protegida de acceso no autorizado y de divulgación.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción, de la misma forma será considerada igual a la fuente o autoridad de los datos.

Fiabilidad: La tasa de fallos del sistema no debe exceder 2 fallos por semana.

Legales: El sistema se basa en el manual de normas y principios establecidos por el MINFAR.

2.5. Modelación del Sistema.

El modelado del sistema representa la funcionalidad completa de un sistema mostrando su interacción con los agentes externos. Esta representación se hace a través de las relaciones entre los actores y los casos de uso dentro del sistema. Los diagramas de casos de uso 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.

2.6. Actores del Sistema.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Tabla 1: Actores del sistema y su justificación.

Actores del Sistema	**Justificación**
Administrador de la Base de Datos.	Es el encargado de ejecutar las salvas, realizar el mantenimiento y la determinación de inconsistencias sobre la Base de Datos del proyecto de Planificación Material y Financiera del MINFAR.

2.7. Casos de Uso.

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

2.7.1. Diagrama de casos de uso del sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

Figura 4: Diagrama Casos de Uso del Sistema

2.7.2. Especificaciones de los Casos de Usos.

Tabla 2: Especificación del caso de uso **Autenticar Administrador.**

CU- 1	**Autenticar Administrador.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador introduce los datos para acceder a la aplicación, es decir, su usuario y contraseña. El sistema encripta la contraseña, busca el usuario y compara la misma. En caso de ser correcto se le habilita la entrada, en caso de no existir, se envía un mensaje de aviso. Terminando así el caso de uso.
Referencia	R.1
Prioridad	Crítico.

Tabla 3: Especificación del caso de uso **Realizar Mantenimiento.**

CU- 2	**Realizar Mantenimiento.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea realizar mantenimiento a dicha Base de Datos. El sistema muestra la interfaz de realizar reindex y la de realizar vacuum, donde el administrador puede ejecutar vacuum y reindex a la misma. Finalmente se guardan dichas acciones. Terminando así el caso de uso.
Referencia	R.2, R 2.1, R 2.2
Prioridad	Crítico.

Tabla 4: Especificación del caso de uso **Realizar Salva.**

CU- 3	**Realizar Salva.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea realizar salvas a dicha Base de Datos. El sistema muestra la interfaz para realizar salvas a la base de datos, donde el administrador le da una ubicación donde guardar la misma. Finalmente se guardan dichas salvas. Terminando así el caso de uso.
Referencia	R.3
Prioridad	Crítico.

Tabla 5: Especificación del caso de uso **Gestionar Regla.**

CU- 4	**Gestionar Regla.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea gestionar las reglas de la Base de Datos. El sistema muestra la interfaz para crear, modificar, eliminar y buscar inconsistencias de las reglas de la base de datos. Finalmente se guardan dichas reglas. Terminando así el caso de uso.
Referencia	R.4, R 4.1, R 4.2, R 4.3
Prioridad	Crítico.

Tabla 6: Especificación del caso de uso **Gestionar Tareas Vacuum.**

CU- 5	**Gestionar Tareas Vacuum.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea gestionar las tareas vacuum que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan

	las tareas. Terminando así el caso de uso.
Referencia	R.5, R 5.1, R 5.2, R 5.3
Prioridad	Crítico.

Tabla 7: Especificación del caso de uso **Gestionar Tareas Reindex.**

CU- 6	**Gestionar Tareas Reindex.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea gestionar las tareas reindex que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan las tareas. Terminando así el caso de uso.
Referencia	R.6, R 6.1, R 6.2, R 6.3
Prioridad	Crítico.

Tabla 8: Especificación del caso de uso **Gestionar Tareas Salvas.**

CU- 7	**Gestionar Tareas Salvas.**
Actor	Administrador de la Base de Datos.
Descripción	El caso de uso se inicia cuando el administrador desea gestionar las tareas de salvas que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan las tareas. Terminando así el caso de uso.
Referencia	R.7, R 7.1, R 7.2, R 7.3
Prioridad	Crítico.

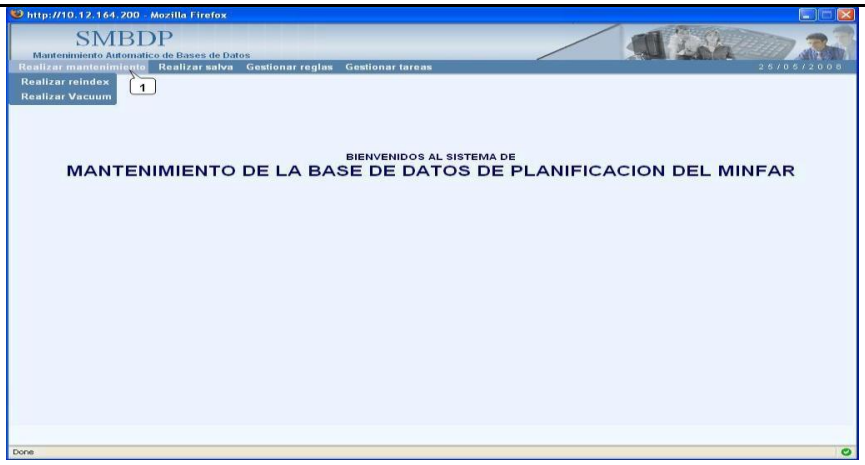
2.7.3. Casos de Uso expandidos.

La descripción puede ser elaborada de forma breve o extendida y debe ir acompañada del prototipo respectivo. El prototipo del sistema que se construye en este punto da una visión de las pantallas diseñadas para cada caso de uso, pero con comportamiento estático, que se presenta al usuario para verificar los requerimientos funcionales.

Ayuda a comprender los Casos de Uso del Sistema, además se hace referencia a los requerimientos consignados en el documento de Requerimientos, con los cuales tiene relación. Este sería el último paso en el análisis, para pasar a la construcción de la solución propuesta.

Tabla 9. Descripción Textual del caso de uso "Realizar Mantenimiento".

Caso de Uso	**Realizar Mantenimiento**
Actores	Administrador de la Base de Datos.
Propósito	El propósito de este caso de uso es que el administrador pueda realizar un mantenimiento automático a la base de datos.
Resumen	El caso de uso se inicia cuando el administrador desea realizar mantenimiento a dicha Base de Datos. El sistema muestra la interfaz de realizar reindex y la de realizar vacuum, donde el administrador puede ejecutar vacuum y reindex a la misma. Finalmente se guardan dichas acciones. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a realizar mantenimiento a la base de datos. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.2, R 2.1, R 2.2
Interfaz I	



(1) Submenú para seleccionar la acción que desea realizar.

Flujo normal de eventos

Acción del actor	**Respuesta del sistema**
1. El administrador desea realizar mantenimiento automático a la base de datos, que consiste en seleccionar la opción <Realizar Vacuum> y después la de <Realizar reindex> del menú principal **Interfaz I**	**2.** El sistema le muestra al usuario la **Interfaz II y** la **Interfaz III.**
3. El administrador decide: • Ejecutar vacuum (ver sección Realizar Vacuum). • Ejecutar reindex (ver sección Realizar Reindex).	

Sección Realizar Vacuum

Interfaz II

(1) RadioButton que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** FULL, **tipo:** string.

(2) RadioButton que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** FREEZE, **tipo:** string.

(3) Checkbox que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** ANALYZE, **tipo:** string.

(4) Botón que permite ejecutar el vacuum. **Nombre**: Aceptar, **tipo**: button.

(5) Botón para cerrar la ventana. **Nombre**: Cancelar, **tipo**: button.

Acción del actor	**Respuesta del sistema**
1. El administrador desea ejecutar un nuevo vacuum en la Base de Datos y selecciona la opción <Realizar Vacuum>.**Interfaz I**	**2.** El sistema le muestra al usuario la **Interfaz II** para dar la posibilidad de ejecutar un nuevo vacuum y muestra la opciones Aceptar y Cancelar activas.
3. El administrador selecciona el tipo de vacuum a realizar.	
4. El administrador selecciona <Aceptar> **Interfaz II.**	**5.** El sistema ejecuta el vacuum creado.

Sección Realizar Reindex

Interfaz III

(1) Checkbox que permite seleccionar la opción del tipo de reindex a realizar. **Nombre:** FORCE, **tipo:** string.

(2) Botón que permite ejecutar el reindex. **Nombre**: Aceptar, **tipo**: button.

(3) Botón para cerrar la ventana. **Nombre**: Cancelar, **tipo**: button.

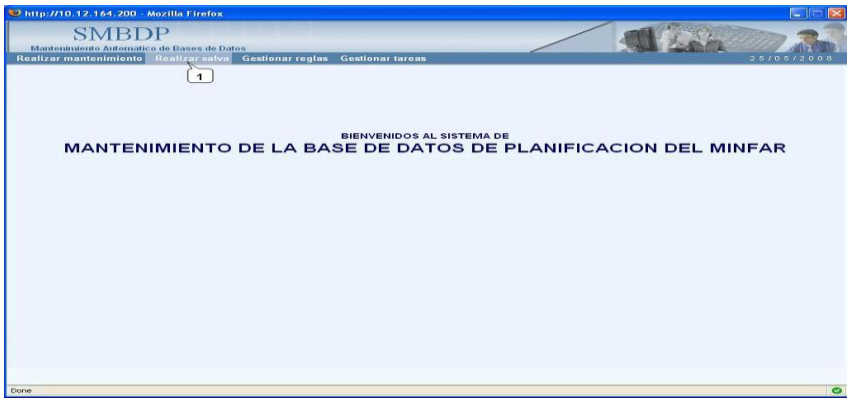
Acción del actor	**Respuesta del sistema**
1. El administrador desea ejecutar un nuevo reindex en la Base de Datos y selecciona la opción <Realizar reindex >.**Interfaz I**	**2.** El sistema le muestra al usuario la **Interfaz III** para dar la posibilidad de ejecutar un nuevo reindex y muestra las opciones Aceptar y Cancelar activas.
3. El administrador selecciona el reindex a realizar. **4.** El administrador selecciona <Aceptar> **Interfaz III**.	**5.** El sistema ejecuta el reindex creado.
Pos-condiciones	El Mantenimiento a la Base de Datos queda realizado.
Prioridad	Crítico

Tabla 10. Descripción Textual del caso de uso "Realizar Salva".

Caso de Uso	**Realizar Salva**
Actores	Administrador de la Base de Datos.

Propósito	El propósito de este caso de uso es que el administrador pueda realizar salvas a la base de datos.
Resumen	El caso de uso se inicia cuando el administrador desea realizar salvas a dicha Base de Datos. El sistema muestra la interfaz para realizar salvas a la base de datos, que le permite al usuario seleccionar una serie de características para ejecutar la misma. Finalmente se guardan dichas salvas. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a realizar salvas a la base de datos. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.3

Interfaz IV



(1) Submenú para seleccionar la acción de realizar salvas.

Interfaz V

(1) Edit Nombre de la salva a realizar. **Nombre:** Nombre, **tipo:** string.
(2) Combobox para seleccionar a que se le realizará la salva. **Nombre:** Backup a la**, tipo:** string. (3) Combobox que muestra las tablas de la base de datos. **Nombre:** Opciones, **tipo:** string. (4) Botón que permite ejecutar la salva. **Nombre**: Aceptar, **tipo**: button. (5) Botón para cerrar la ventana. **Nombre**: Cancelar, **tipo**: button.
Flujo normal de eventos

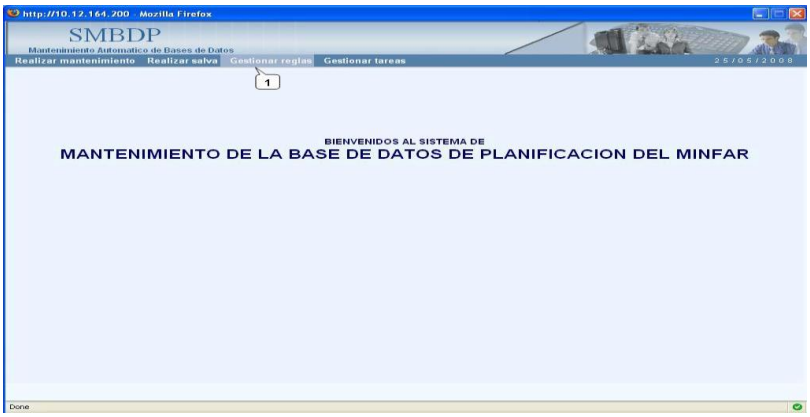
Acción del actor	**Respuesta del sistema**
1. El administrador de la base de datos desea realizar salvas a la misma y selecciona la opción <Realizar salva> del menú principal **Interfaz IV**	**2.** El sistema muestra la **Interfaz V.** **3.** El sistema le muestra el formulario (**Interfaz V**) en blanco para dar la posibilidad de ejecutar una nueva salva y muestra la opción Aceptar activa.
4. El administrador introduce los datos de la nueva salva por teclado. **5.** El administrador selecciona <Aceptar> **Interfaz V**.	**6.** El sistema ejecuta la salva creada.
Pos-condiciones	La salva a la Base de Datos queda realizada.
Prioridad	Crítico

Tabla 11. Descripción Textual del caso de uso "Gestionar Regla".

Caso de Uso	**Gestionar Regla**
Actores	Administrador de la Base de Datos.

Propósito	El propósito de este caso de uso es que el administrador pueda gestionar reglas en la base de datos.
Resumen	El caso de uso se inicia cuando el administrador desea gestionar las reglas de la Base de Datos. El sistema muestra la interfaz para crear, modificar, eliminar y buscar inconsistencias de las reglas de la base de datos. Finalmente se guardan dichas reglas. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a gestionar reglas en la base de datos. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.4, R4.1, R4.2, R4.3

Interfaz VI



(1) Submenú para seleccionar la opción de gestionar reglas.

Interfaz VII

(1) Botón para Insertar una regla. **Nombre**: Insertar, **tipo**: button.

(2) Botón para Modificar una regla. **Nombre**: Modificar, **tipo**: button.

(3) Botón para Eliminar una regla. **Nombre**: Eliminar, **tipo**: button.

(4) Botón para buscar las inconsistencias. **Nombre:** Inconsistencia, **tipo:** button.

(5) Combobox para seleccionar la tabla de la base de datos. **Nombre:** combox1, **tipo:** string.

(6) Botón para buscar los campos de la tabla de la base de datos seleccionada. **Nombre**: Buscar, **tipo**: button.

(7) Grid para mostrar los campos de la tabla seleccionada, sus tipos y si tienen una regla asociada.

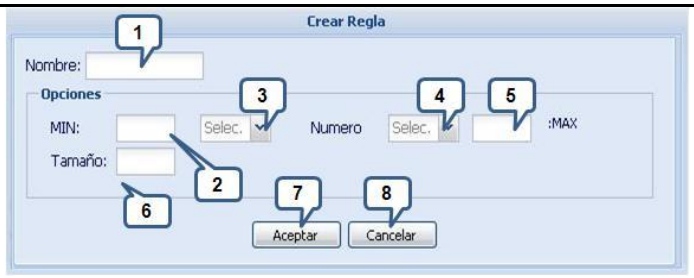
Flujo normal de eventos

Acción del actor	**Respuesta del sistema**
1. El administrador de la base de datos desea gestionar reglas de la misma y selecciona la opción <Gestionar reglas> del menú principal **Interfaz VI.**	**2.** El sistema muestra la **Interfaz VII.**
3. El administrador selecciona la tabla con la cual va a trabajar y oprime el botón buscar.	**4.** El sistema muestra los campos de la tabla seleccionada.

5. El administrador selecciona el campo al cual le va a asignar una regla, y posteriormente selecciona uno de los siguientes escenarios: • Crear reglas (ver sección Insertar). • Modificar reglas (ver sección Modificar). • Eliminar reglas (ver sección Eliminar). • Buscar Inconsistencia (ver sección Buscar).	

Sección Insertar

Interfaz VIII



(1) Edit para el nombre de la regla**. Nombre:** Nombre**, tipo:** string.

(2) Edit para el valor mínimo de la regla. **Nombre:** inter1, **tipo:** int.

(3) Combobox para la condición del valor mínimo. **Nombre:** MIN, **tipo:** string.

(4) Combobox para la condición del valor máximo. **Nombre:** MAX, **tipo:** string.

(5) Edit para el valor máximo de la regla. **Nombre:** inter12, **tipo:** int.

(6) Edit para el tamaño de la regla. **Nombre:** Tamaño, **tipo:** int.

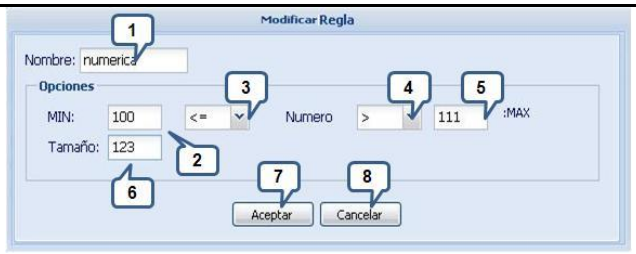
(7) Botón que permite ejecutar la regla. **Nombre:** Aceptar, **tipo:** button.

(8) Botón que cierra la ventana. Nombre: Cancelar, tipo: button.

Acción del actor	**Respuesta del sistema**
1. El administrador desea adicionar una nueva regla y selecciona la opción <Insertar>.	**2.** El sistema le muestra la **Interfaz VIII** para dar la posibilidad de agregar una nueva regla y muestra las opciones Aceptar y Cancelar activas.
3. El administrador introduce los datos de la nueva regla.	
4. El administrador selecciona <Aceptar> **Interfaz VIII**.	**5.** El sistema verifica que los datos introducidos sean correctos. Guarda los cambios en sistema. Devuelve la regla actualizada visualizándolo en el Grid **Interfaz VII** y envía mensaje al actor. 

Sección Modificar

Interfaz IX



(1) Edit para modificar el nombre de la regla**. Nombre:** Nombre**, tipo:** string.

(2) Edit para modificar el valor mínimo de la regla. **Nombre:** inter1, **tipo:** int.

(3) Combobox para modificar la condición del valor mínimo. **Nombre:** MIN, **tipo:** string.

(4) Combobox para modificar la condición del valor máximo. **Nombre:** MAX, **tipo:** string.

(5) Edit para modificar el valor máximo de la regla. **Nombre:** inter12, **tipo:** int.

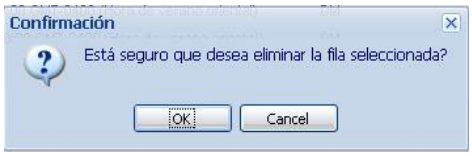
(6) Edit para modificar el tamaño de la regla. **Nombre:** Tamaño, **tipo:** int.

(7) Botón que permite modificar la regla. **Nombre:** Aceptar, **tipo:** button.

(8) Botón que cierra la ventana. Nombre: Cancelar, tipo: button.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la regla que desea modificar en la base de datos y selecciona la opción <Modificar>.	**2.** El sistema muestra la **Interfaz IX**. Muestra las opciones Aceptar y Cancelar activas.
3. El administrador realiza los cambios necesarios, y selecciona <Aceptar>.	**4.** El sistema verifica que los datos modificados sean correctos. Guarda los cambios en el sistema. Devuelve la regla actualizada y envía mensaje al actor. 

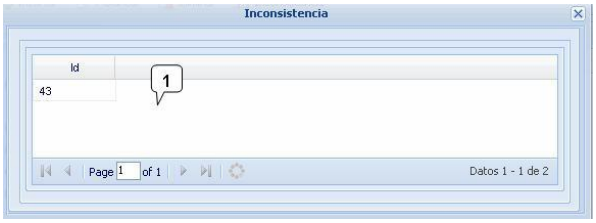
Sección Eliminar

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la regla que desea eliminar de la base de datos y selecciona la opción <Eliminar>. **3.** El administrador responde “OK”.	**2.** El sistema pide confirmación al actor.  **4.** El sistema elimina la regla señalada por el usuario. Guarda los datos en el sistema. Devuelve la regla actualizada y envía un mensaje al usuario.

Sección Buscar

Interfaz X



(1) Grid para mostrar las inconsistencias de los campos de la tabla seleccionada.

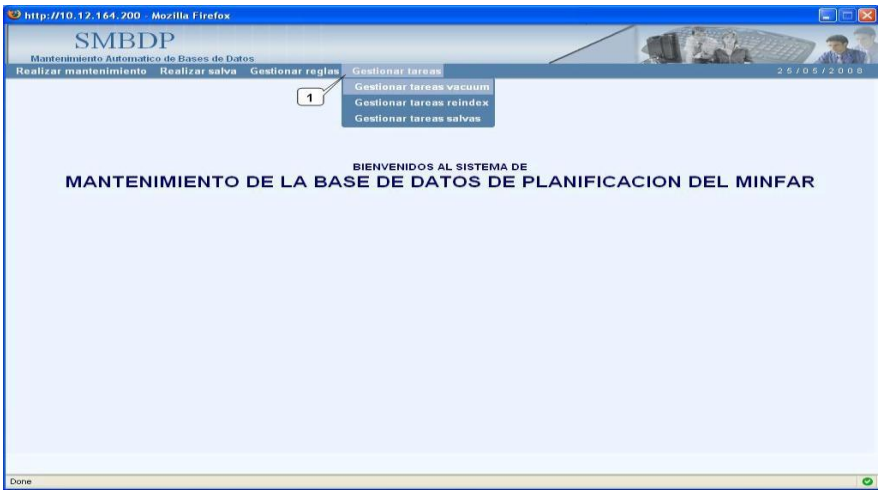
Acción del actor	**Respuesta del sistema**
1. El administrador selecciona el campo que le desea buscar la inconsistencia y oprime el botón Inconsistencias. **3.** El usuario observa y a la vez conoce las inconsistencias.	**2.** El sistema muestra **Interfaz X.**
Pos-condiciones	La regla de la Base de Datos queda gestionada.
Prioridad	Crítico

Tabla 12. Descripción Textual del caso de uso "Gestionar Tareas Vacuum".

Caso de Uso	**Gestionar Tareas Vacuum**
Actores	Administrador de la Base de Datos.
Propósito	El propósito de este caso de uso es que el administrador pueda gestionar las tareas vacuum que se programan sobre la base de datos.

Resumen	El caso de uso se inicia cuando el administrador desea gestionar las tareas vacuum que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan las tareas. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a gestionar las tareas vacuum. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.5, R5.1, R5.2, R5.3

Interfaz XI



(1) Submenú para seleccionar la opción de gestionar tareas vacuum.

Interfaz XII

(1) Botón para Insertar una tarea vacuum. **Nombre**: Insertar, **tipo**: button.

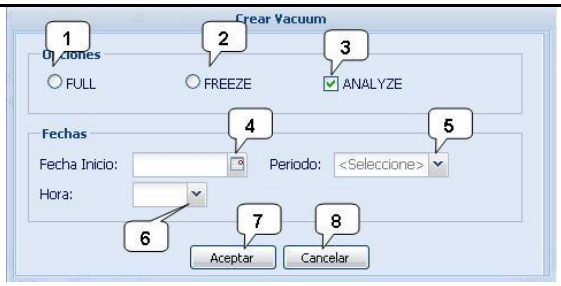
(2) Botón para Modificar una tarea vacuum. **Nombre**: Modificar, **tipo**: button.

(3) Botón para Eliminar una tarea vacuum. **Nombre**: Eliminar, **tipo**: button.

(4) Grid para mostrar las tareas vacuum.

Flujo normal de eventos

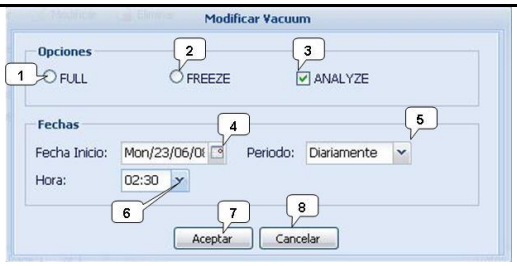
Acción del actor	**Respuesta del sistema**
1. El administrador de la base de datos desea gestionar las tareas vacuum y selecciona la opción <Gestionar tareas vacuum> del menú principal **Interfaz X.**	**2.** El sistema muestra la **Interfaz XI.**
3. El especialista decide: • Crear tareas vacuum (ver sección Insertar). • Modificar tareas vacuum (ver sección Modificar). • Eliminar tareas vacuum (ver sección Eliminar).	

Sección Insertar	
Interfaz XIII	
	
(1) RadioButton que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** FULL, **tipo:** string. (2) RadioButton que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** FREEZE, **tipo:** string. (3) Checkbox que permite seleccionar la opción del tipo de vacuum a realizar. **Nombre:** ANALYZE, **tipo:** string. (4) DateField que permite poner la fecha a la tarea vacuum. **Nombre:** Fecha Inicio, **tipo:** string. (5) ListBox para seleccionar el periodo de la tarea. **Nombre:** Periodo, **tipo:** string. (6) ListBox para seleccionar la hora en que se realiza la tarea. **Nombre:** Hora, **tipo:** time. (7) Botón que permite ejecutar la tarea vacuum. **Nombre:** Aceptar, **tipo:** button. (8) Botón que cierra la ventana. Nombre: Cancelar, tipo: button.	
Acción del actor	**Respuesta del sistema**
1. El administrador desea adicionar una nueva tarea vacuum a la base de datos y selecciona la opción <Insertar>.	**2.** El sistema le muestra la **Interfaz XII** para dar la posibilidad de agregar una nueva tarea vacuum y muestra las opciones Aceptar y Cancelar activas.
3. El administrador introduce los	

datos de la nueva tarea vacuum.	
4. El administrador selecciona <Aceptar>**.**	**5.** El sistema verifica que los datos introducidos sean correctos. Guarda los cambios en sistema. Devuelve el vacuum actualizado visualizándolo en el Grid **Interfaz XI** y envía mensaje al actor. 

Sección Modificar

Interfaz XIV



(1) RadioButton que permite seleccionar la opción del tipo de vacuum a modificar. **Nombre:** FULL, **tipo:** string.

(2) RadioButton que permite seleccionar la opción del tipo de vacuum a modificar. **Nombre:** FREEZE, **tipo:** string.

(3) Checkbox que permite seleccionar la opción del tipo de vacuum a modificar. **Nombre:** ANALYZE, **tipo:** string.

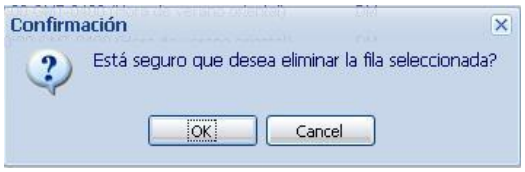
(4) DateField que permite modificar la fecha a la tarea vacuum.

(5) ListBox para modificar el periodo de la tarea. **Nombre:** Periodo, **tipo:** string.

(6) ListBox para modificar la hora en que se realiza la tarea. **Nombre:** Hora, **tipo:** time.

(7) Botón que permite modificar la tarea vacuum. **Nombre:** Aceptar, **tipo:** button.

<table>
<tr><td colspan="2">(8) Botón que cierra la ventana. Nombre: Cancelar, tipo: button.</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El administrador selecciona la tarea que desea modificar en la base de datos y el administrador selecciona la opción <Modificar>.</td><td>2. El sistema muestra la Interfaz XIII. Muestra las opciones Aceptar y Cancelar activas.</td></tr>
<tr><td>4. El administrador realiza los cambios necesarios, y selecciona <Aceptar>.</td><td>5. El sistema verifica que los datos modificados sean correctos. Guarda los cambios en el sistema. Devuelve el vacuum actualizado y envía mensaje al actor.
 
</td></tr>
<tr><td colspan="2">Sección Eliminar</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El administrador selecciona la tarea que desea eliminar de la base de datos y selecciona la opción <Eliminar>.

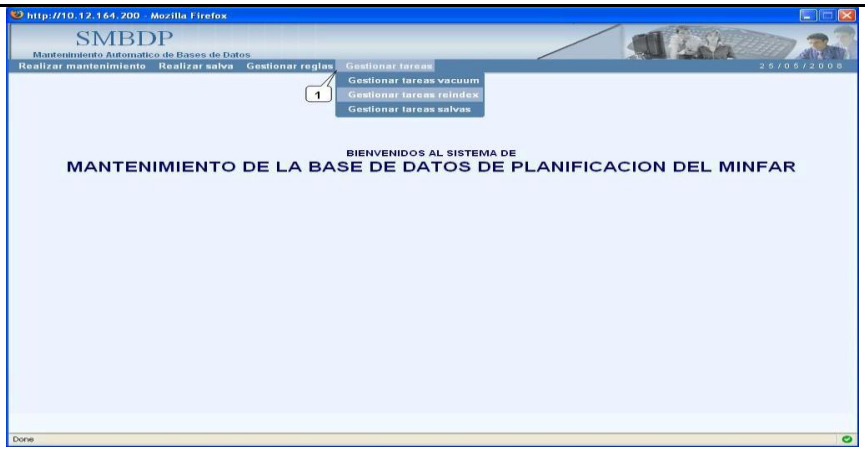
3. El administrador responde “OK”.</td><td>2. El sistema pide confirmación al actor.
 

4. El sistema elimina el vacuum señalado por el usuario. Guarda los datos en el sistema. Devuelve el vacuum actualizado y envía un mensaje al actor.
 
</td></tr>
</table>

Pos-condiciones	La tarea de la Base de Datos queda gestionada.
Prioridad	Crítico

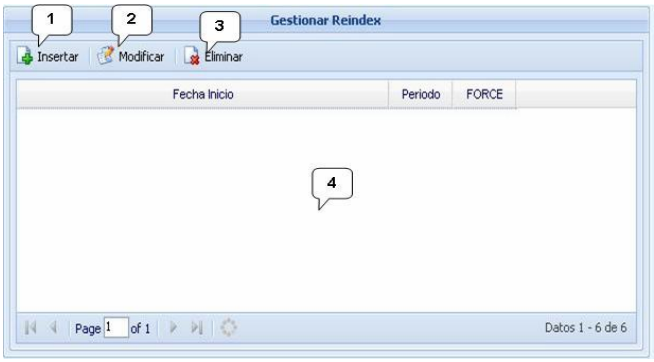
Tabla 13. Descripción Textual del caso de uso "Gestionar Tareas Reindex".

Caso de Uso	**Gestionar Tareas Reindex**
Actores	Administrador de la Base de Datos.
Propósito	El propósito de este caso de uso es que el administrador pueda gestionar las tareas reindex que se programan sobre la base de datos.
Resumen	El caso de uso se inicia cuando el administrador desea gestionar las tareas reindex que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan las tareas. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a gestionar las tareas reindex. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.6, R6.1, R6.2, R6.3
Interfaz XV	



(1) Submenú para seleccionar la opción de gestionar tareas reindex.

Interfaz XVI



(1) Botón para Insertar una tarea reindex. **Nombre**: Insertar, **tipo**: button.

(2) Botón para Modificar una tarea reindex. **Nombre**: Modificar, **tipo**: button.

(3) Botón para Eliminar una tarea reindex. **Nombre**: Eliminar, **tipo**: button.

(4) Grid para mostrar las tareas reindex.

Flujo normal de eventos

Acción del actor	**Respuesta del sistema**
1. El administrador de la base de datos desea gestionar las tareas reindex y selecciona la opción	**2.** El sistema muestra la **Interfaz XV.**

<Gestionar tareas reindex> del menú principal **Interfaz XIV.**	
3. El especialista decide: • Crear tareas reindex (ver sección Insertar). • Modificar tareas reindex (ver sección Modificar). • Eliminar tareas reindex (ver sección Eliminar).	
Sección Insertar	
Interfaz XVII	
	
(1) Checkbox que le permite seleccionar la opcion del reindex a realizar. **Nombre:** FORCE, **tipo:** string. (2) DateField que permite poner la fecha a la tarea reindex. **Nombre:** Fecha Inicio, **tipo:** string. (3) ListBox para seleccionar el periodo del reindex a realizar. **Nombre:** Periodo, **tipo:** string. (4) ListBox para seleccionar la hora en que se realiza el reindex. **Nombre:** Hora, **tipo:** time. (5) Botón que permite ejecutar la tarea reindex. **Nombre:** Aceptar, **tipo:** button. (6) Botón que cierra la ventana. **Nombre:** Cancelar, **tipo:** button.	
Acción del actor	**Respuesta del sistema**

1. El administrador desea adicionar una nueva tarea reindex a la base de datos y selecciona la opción <Insertar>.	**2.** El sistema le muestra la **Interfaz XVI** para dar la posibilidad de agregar una nueva tarea reindex y muestra las opciones Aceptar y Cancelar activas.
3. El administrador introduce los datos de la nueva tarea reindex.	
4. El administrador selecciona <Aceptar>.	**5.** El sistema verifica que los datos introducidos sean correctos. Guarda los cambios en sistema. Devuelve el reindex actualizado visualizándolo en el Grid **Interfaz XV** y envía mensaje al actor. 

Sección Modificar

Interfaz XVIII

(1) Checkbox que le permite modificar la opcion del reindex a realizar. **Nombre:** FULL, **tipo:** string.

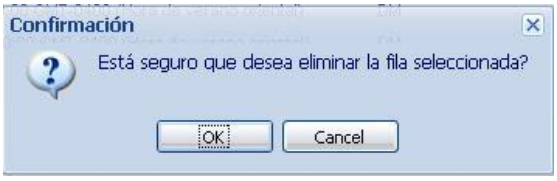
(2) DateField que permite modificar la fecha a la tarea reindex.

(3) ListBox para seleccionar el periodo del reindex a modificar. **Nombre:** Periodo, **tipo:** string.

(4) ListBox para modificar la hora en que se realiza el reindex. **Nombre:** Hora, **tipo:** time.

(5) Botón que permite modificar la tarea reindex. **Nombre:** Aceptar, **tipo:** button.

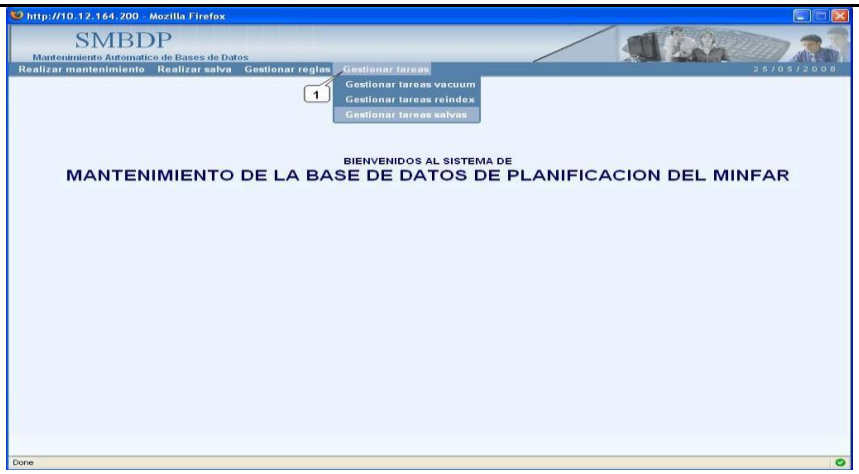
(6) Botón que cierra la ventana. **Nombre:** Cancelar, **tipo:** button.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la tarea que desea modificar en la base de datos y el administrador selecciona la opción <Modificar>.	**2.** El sistema muestra la **Interfaz XVII**. Muestra las opciones Aceptar y Cancelar activas.
3. El administrador realiza los cambios necesarios, y selecciona <Aceptar>.	**4.** El sistema verifica que los datos modificados sean correctos. Guarda los cambios en el sistema. Devuelve el reindex actualizado y envía mensaje al actor. 
Sección Eliminar	
Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la tarea que desea eliminar de la base de datos y el administrador selecciona la opción <Eliminar>. **3.** El administrador responde “OK”.	**2.** El sistema pide confirmación al actor.  **4.** El sistema elimina el reindex señalado por el usuario. Guarda los datos en el sistema. Devuelve el reindex actualizado y envía un mensaje al actor. 
Pos-condiciones	La tarea reindex de la Base de Datos queda gestionada.

Prioridad	Crítico

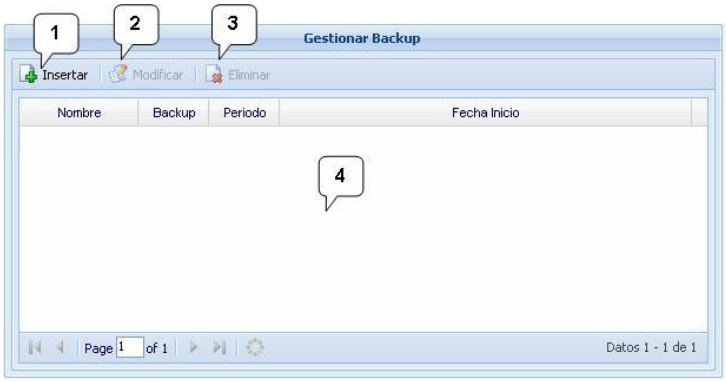
Tabla 14. Descripción Textual del caso de uso "Gestionar Tareas Salvas".

Caso de Uso	**Gestionar Tareas Salvas**
Actores	Administrador de la Base de Datos.
Propósito	El propósito de este caso de uso es que el administrador pueda gestionar las tareas salvas que se programan sobre la base de datos.
Resumen	El caso de uso se inicia cuando el administrador desea gestionar las tareas salvas que se programan sobre la Base de Datos. El sistema muestra la interfaz para crear, modificar y eliminar las tareas. Finalmente se guardan las tareas. Terminando así el caso de uso.
Tipo	Real y expandido.
Precondiciones.	El actor se debe haber autenticado, como tal encontrarse trabajando con el sistema y tener acceso a gestionar las tareas reindex. Al registrarse el actor, el sistema lo identifica como Administrador de la Base de Datos (rol).
Referencias	R.7, R7.1, R7.2, R7.3
Interfaz XIX	



(1) Submenú para seleccionar la opción de gestionar tareas de salvas.

Interfaz XX



(1) Botón para Insertar una tarea salva. **Nombre**: Insertar, **tipo**: button.

(2) Botón para Modificar una tarea salva. **Nombre**: Modificar, **tipo**: button.

(3) Botón para Eliminar una tarea salva. **Nombre**: Eliminar, **tipo**: button.

(4) Grid para mostrar las tareas salva.

Flujo normal de eventos

Acción del actor	**Respuesta del sistema**

1. El administrador de la base de datos desea gestionar las tareas de salvas y selecciona la opción <Gestionar tareas salvas> del menú principal **Interfaz XVIII.**	**2.** El sistema muestra la **Interfaz XIX.**
3. El especialista decide: • Crear tareas de salvas (ver sección Insertar). • Modificar tareas de salvas (ver sección Modificar). • Eliminar tareas de salvas (ver sección Eliminar).	

Sección Insertar

Interfaz XXI

(1) Edit Nombre de la tarea salva a realizar. **Nombre:** Nombre, **tipo:** string.

(2) DateField que permite poner la fecha a la tarea salva.

(3) ListBox para seleccionar el periodo de la tarea salva a realizar. **Nombre:** Periodo, **tipo:** string.

(4) ListBox para seleccionar la hora en que se realiza la tarea salva. **Nombre:** Hora, **tipo:** time.

(5) ListBox para seleccionar a que se le realizará la salva. **Nombre:** Backup a la**, tipo:** string.

(6) ListBox que muestra las tablas de la Base de Datos. **Nombre:** Opciones, **tipo:** string.

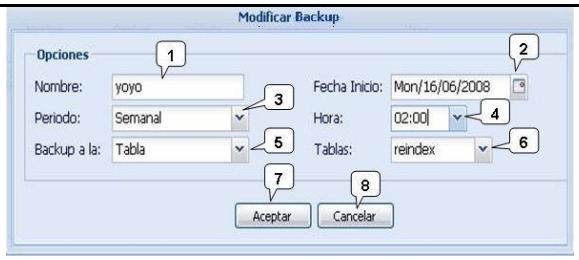
(7) Botón que permite ejecutar la tarea salva. **Nombre**: Aceptar, **tipo**: button.

(8) Botón para cerrar la ventana. **Nombre**: Cancelar, **tipo**: button.

Acción del actor	**Respuesta del sistema**
1. El administrador desea adicionar una nueva tarea de salva a la base de datos y selecciona la opción < Insertar >.	**2.** El sistema le muestra la **Interfaz XX** para dar la posibilidad de agregar una nueva tarea de salva y muestra la opciones Aceptar y Cancelar activa.
3. El administrador introduce los datos de la nueva tarea de salva.	
4. El administrador selecciona <Aceptar>.	**5.** El sistema verifica que los datos introducidos sean correctos. Guarda los cambios en sistema. Devuelve la salva actualizada visualizándola en el Grid **Interfaz XV** y envía mensaje al actor. 

Sección Modificar

Interfaz XXII



(1) Edit Nombre de la tarea salva a modificar. **Nombre:** Nombre, **tipo:** string.

(2) DateField que permite modificar la fecha a la tarea salva.

(3) ListBox para modificar el periodo de la tarea salva. **Nombre:** Periodo, **tipo:** string.

(4) ListBox para modificar la hora en que se realiza la tarea salva. **Nombre:** Hora, **tipo:** time.

(5) ListBox para modificar a que se le realizará la salva. **Nombre:** Backup a la**, tipo:** string.

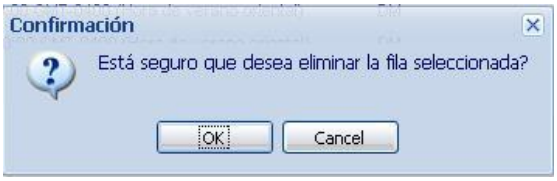
(6) ListBox que muestra las tablas de la Base de Datos. **Nombre:** Opciones, **tipo:** string.

(7) Botón que permite modificar la tarea salva. **Nombre**: Aceptar, **tipo**: button.

(8) Botón para cerrar la ventana. **Nombre**: Cancelar, **tipo**: button.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la tarea que desea modificar en la base de datos y el administrador selecciona la opción <Modificar>.	**2.** El sistema muestra la **Interfaz XXI**. Muestra las opciones Aceptar y Cancelar activas.
3. El administrador realiza los cambios necesarios, y selecciona <Aceptar>.	**4.** El sistema verifica que los datos modificados sean correctos. Guarda los cambios en el sistema. Devuelve la salva actualizada y envía mensaje al actor. 

Sección Eliminar

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona la tarea que desea eliminar de la base de datos y el administrador selecciona la opción <Eliminar>.	**2.** El sistema pide confirmación al actor. 
3. El administrador responde “OK”.	**4.** El sistema elimina la salva señalada por el usuario. Guarda los datos en el sistema. Devuelve el reindex actualizado y envía

	un mensaje al autor. 
Pos-condiciones	La tarea de salva de la Base de Datos queda gestionada.
Prioridad	Crítico

2.8. Conclusiones.

Hasta el momento se ha visto como a partir del análisis de los procesos del negocio y de los artefactos obtenidos en el modelamiento del mismo se identificaron los requerimientos, los cuales ayudarán a desarrollar las funcionalidades que el software a construir debe cumplir. Estas se representaron mediante el Diagrama de Casos de Uso y finalmente se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Todo esto constituye la base para empezar la construcción del sistema.

CAPITULO 3. ANALISIS Y DISENO DEL SISTEMA.

3.1. Introducción.

En este capítulo se procede a la confección de la propuesta de solución, basándose en el análisis hecho anteriormente. Para el desarrollo de este capítulo se tuvieron en cuenta los requisitos funcionales, como no funcionales obtenidos durante el flujo de la captura de requisitos. El mismo constará con varios artefactos que tienen como objetivos mostrar el diseño y la implementación del sistema propuestos como solución.

3.2. Modelo de Análisis.

El modelo de análisis es la primera representación técnica del sistema, por lo que puede considerarse como una primera aproximación al modelo de diseño, y es por tanto una entrada fundamental para las actividades de diseño e implementación subsiguientes. A partir de su realización se refinan y estructuran los requisitos obtenidos con anterioridad proporcionando una visión general del sistema.

El modelo de análisis debe lograr tres objetivos primarios:

1. Describir lo que requiere el cliente.
2. Establecer una base para la creación de un diseño de software.
3. Definir un conjunto de requisitos que se pueda validar una vez que se construye el software.

A continuación se muestran los diagramas de clases del análisis correspondiente a cada uno de los casos de usos que se describieron en el capítulo anterior.

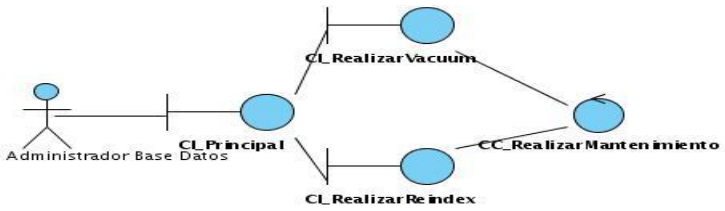


Figura 5: Diagrama de clases del análisis **"Realizar Mantenimiento".**

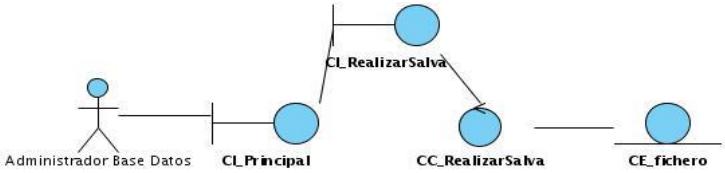


Figura 6: Diagrama de clases del análisis **"Realizar Salva".**

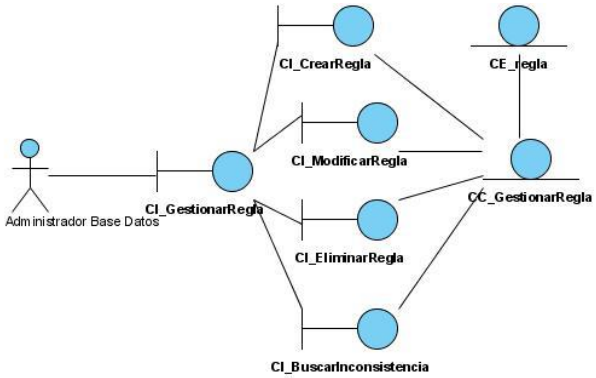


Figura 7: Diagrama de clases del análisis **"Gestionar Regla".**

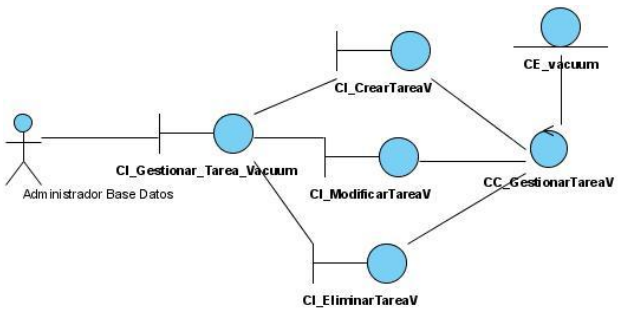


Figura 8: Diagrama de clases del análisis **"Gestionar Tareas Vacuum".**

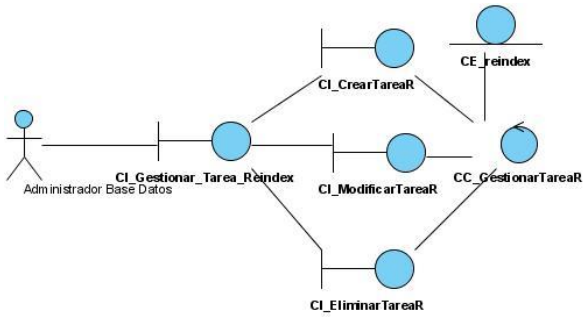


Figura 9: Diagrama de clases del análisis **"Gestionar Tareas Reindex".**

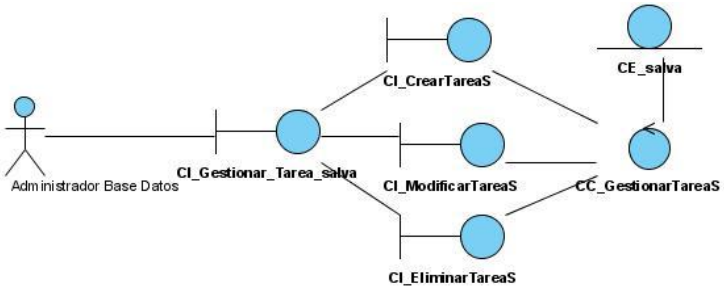


Figura 10: Diagrama de clases del análisis **"Gestionar Tareas Salva".**

3.3. Modelo de Diseño.

El diseño constituye el centro de atención de la propuesta de solución, el mismo contribuye a una arquitectura estable y sólida. En el diseño se modela el sistema y de forma incluida la arquitectura para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se de forma al sistema.

Antes de realizar los diagramas que muestran la solución del sistema, es necesario definir algunos conceptos que ayudan a obtener un producto de mejor calidad.

3.4. Arquitectura.

La arquitectura establece los fundamentos para que analistas, diseñadores, y programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema informático.

Dentro de la arquitectura hay elementos de vital importancia que no se puede dejar de mencionar: *los patrones de arquitectura.*

¿Qué es un patrón?

Un patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizares continuamente.

¿A qué responden los patrones de arquitectura?

1. Son esquemas de organización de un sistema.
2. Especifican una serie de subsistemas y sus responsabilidades.
3. Incluyen reglas para organizar las relaciones entre ellos.

Para el desarrollo de la aplicación se propone utilizar el patón de Capas.

¿Por qué patrón Capas?

Este patrón define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse. A continuación una descripción detallada de la arquitectura que se propone.

Arquitectura Multi- Capas.

Esta arquitectura brinda la ventaja de aislar definitivamente la lógica de negocios de todo lo que tenga que ver con el origen de datos, ya que desde el manejo de la conexión, hasta la

ejecución de una consulta, la manejará la capa de Acceso a Datos. De este modo, ante cualquier eventual cambio, solo se deberá cambiar una capa en específico.

1. **Capa de presentación o capa de interfaz de usuario:** es la forma de ofrecer al usuario un modo de interactuar con la aplicación. Las interfaces de usuario se implementan utilizando formularios, controles u otro tipo de tecnología que permita procesar y dar formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos.
2. **Capa de lógica de negocio** [19]**:** es la parte más importante de una aplicación debido a que encapsula las responsabilidades y los procesos que en ella se comprenden, su funcionamiento basado en el uso de clases o componentes y a su vez compuestos estos por métodos que de una forma u otra establecen la relación entre los componentes de presentación y los componentes de acceso a datos. Para el acceso al subsistema de la capa de acceso a datos se presenta una interfaz denominada "Factoría típica", clase que implementa el patrón para clases con un objetivo y funcionalidades similares, o sea todo flujo de información entre estas dos capas es a través de esta clase.
3. **Capa de Acceso a Datos:** permite conocer el formato de los datos que se intercambian a través de los componentes de acceso a datos y la capa lógica de negocio, permiten el acceso a los datos a través de los mismos. Además usan la tecnología de acceso a datos PDO e implementa un interfaz de programación para la gestión de los datos.
4. **Capa Datos:** Contiene el almacenamiento de Datos.

¿Por qué una Arquitectura Multi-Capas?

1. **Abstracción total acerca del origen de datos:** las distintas capas se especializan absolutamente en la funcionalidad que deben brindar sin importar cual es el origen de los datos procesados.
2. **Bajo costo de desarrollo y mantenimiento de las aplicaciones:** esta arquitectura brinda un control más cercano de cada componente, así como también la posibilidad de una verdadera reutilización del código.

3. **Mejor calidad en las aplicaciones:** como las aplicaciones son construidas en unidades separadas, estas pueden ser probadas independientemente y con mucho mas detalle, esto conduce a obtener un producto mucho más sólido.

4. **Reutilización de código:** la concepción natural de un sistema desarrollado con esta arquitectura, promueve la reutilización de sus componentes en varias partes del propio desarrollo y de futuros sistemas.

En la figura 10 que se muestra a continuación se puede ver el patrón propuesto para la solución.

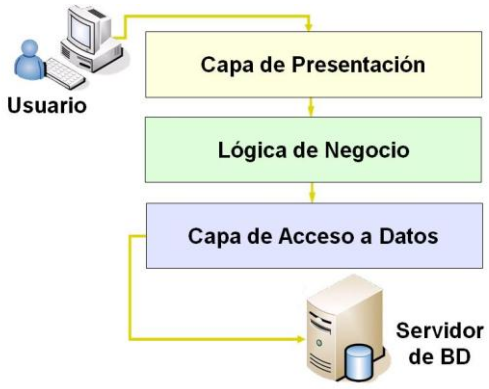


Figura 11. Arquitectura Multi_Capas.

3.5. Mecanismo 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:

- Mantener la homogeneidad en el diseño.
- Reutilizar soluciones anteriormente probadas.

- Reutilizar documentación.

Un ejemplo típico de mecanismos son los patrones de diseño.

¿Qué es un Patrón de Diseño?

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable, igualmente es una descripción de clases y objetos comunicándose entre sí que resuelve un problema de diseño general en un contexto particular.

Los patrones de diseño han contribuido a dar flexibilidad y extensibilidad a los diseños. Además han demostrado ser una forma muy útil y exitosa de reutilizar diseño ya definidos, ya que ellos no sólo nombran, abstraen e identifican aspectos claves de estructuras comunes de diseño, sino que generalmente son descritos en una forma específica documental, haciendo su comprensión y aplicación fácil para el conjunto de desarrolladores.

Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

En la actualidad son muchos los patrones de diseños utilizados en la construcción de una aplicación y en base a las responsabilidades asignadas se clasifican en:

1. Experto.
2. Creador.
3. Alta Cohesión.
4. Bajo Acoplamiento.
5. Controlador.

Dentro de patrón creadores definidos por la GoF (Gang of Four) se encuentra el patrón de Factoría.

Este patrón es un modelo que utiliza abstracción de clases para crear y relacionar objetos sin conocer de qué clase son. Se utiliza cuando la aplicación no sabe de antemano el tipo de objeto que se va a crear, es en tiempo de ejecución cuando toma la decisión.

Mecanismo de 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.

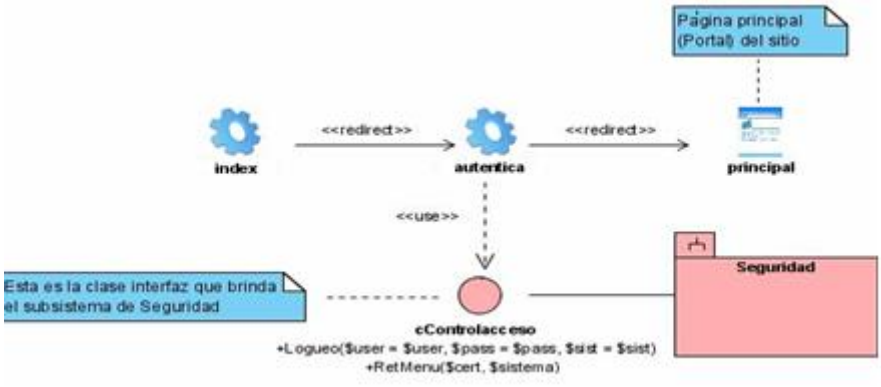


Figura 12. Mecanismo de Seguridad.

Mecanismo de Acceso a Datos.

¿Por qué definir un mecanismo para el 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 20) 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.

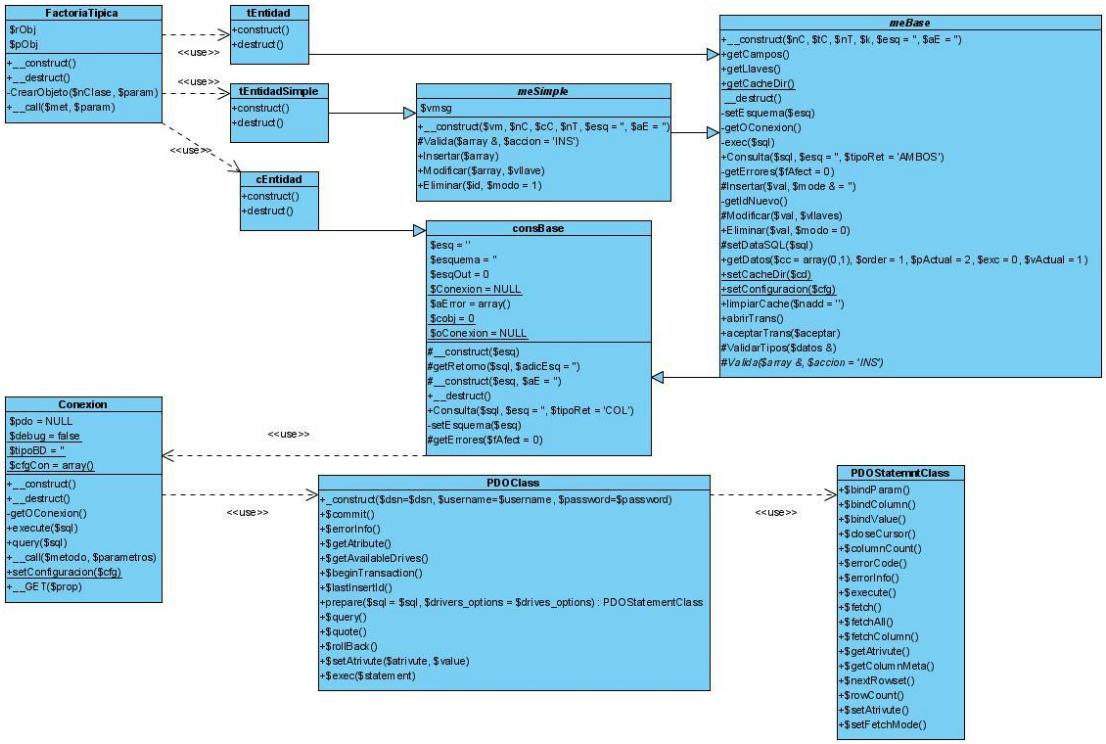


Figura 13. Diagrama general del "Mecanismo de Acceso a Datos".

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

FactoríaTípica: 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. Implementa un método de instancia de clases típicas.

tEntidadSimple: Es una clase que representa a las clases típicas (nomencladores simples) en general de la aplicación. Estas típicas son de una implementación muy sencilla, pues la mayoría de las líneas que normalmente habían que codificar quedaron encapsuladas en la clase base de las mismas. 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, insertar, modificar y eliminar. Hereda de la clase abstracta meSimple.

meSimple: Clase abstracta heredera de meBase, y la vez 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. Redefine las

operaciones básicas que pudieran realizarse a una entidad (insertar, eliminar, modificar) para los nomencladores simples.

meBase: Es una clase abstracta que constituye la base para el resto de las que implementen funcionalidades para el trabajo con las entidades del sistema desarrollado. Implementa además, las operaciones básicas que pudieran realizársele a una entidad como pudieran ser: INSERT, DELETE, UPDATE así como un conjunto de consultas. Esta clase encapsula todo lo relacionado con la conexión al gestor de la bases de datos.

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.

Conexión: Esta clase es la encargada de establecer la conexión con el servidor de la base de datos a través de un objeto PDO de la librería de PHP.

PDO: Es un modelo de acceso a datos para PHP que brinda una capa de abstracción para el acceso a BD desde PHP.

3.6. Diagrama de Clases del Diseño.

Para obtener un nivel correcto de abstracción y detalle que permita obtener un resultado final, es mejor modelar los artefactos del sistema, es decir, modelar las páginas, los enlaces entre ellas, todo el código que irá creando las páginas, así como el contenido dinámico de estas una vez que estén en el navegador del cliente, para ello se realizarán los diagramas de clases.

Su objetivo fundamental es crear una entrada apropiada y un punto de partida para la implementación, capturando los requisitos o subsistemas individuales, interfaces y clases. El diseño convierte el modelo de análisis en diseños de datos, arquitectónicos, de interfaz y a nivel de componentes del software.

Este sistema tiene concebido por cada caso de uso lo siguiente:

- Una clase de lógica de negocio (ln_**NombreCU**) por cada caso de uso.
- Un fichero **GLOBAL** para la configuración de las direcciones de las clases a través de la función __autoload y algún elemento de configuración.

- La página servidora tendrá las instancias del los objetos ln_**NombreCU.**
- Una página servidora **principal** que redireccionará a servidora de cada Caso de Uso.
- Una interfaz **cControlacceso** que permite la autenticación de los usuarios.
- Una clase **lnBase**: Es la clase base de lógica de negocio, de la cual heredarán todas las clases "ln_NombreCU". En ella se define el objeto de la clase Factoría Típica.
- Una clase **FactoríaTípica** que a través de la cual se crean y se manipulan los objetos de acceso a datos como las tablas y consultas. Es una interfaz entre la Lógica de Negocio y la capa de Acceso a Datos. Basada en el patrón Factoría.
- Los ficheros **.js** que son los ficheros ext controls (menú, toolbar, formularios).

Con el objetivo de optimizar la modelación del diseño de la aplicación en desarrollo, se construyó un diagrama genérico de clases del diseño (Figura 13), 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 realizando, 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.

A continuación se muestran los diagramas de clases del diseño.

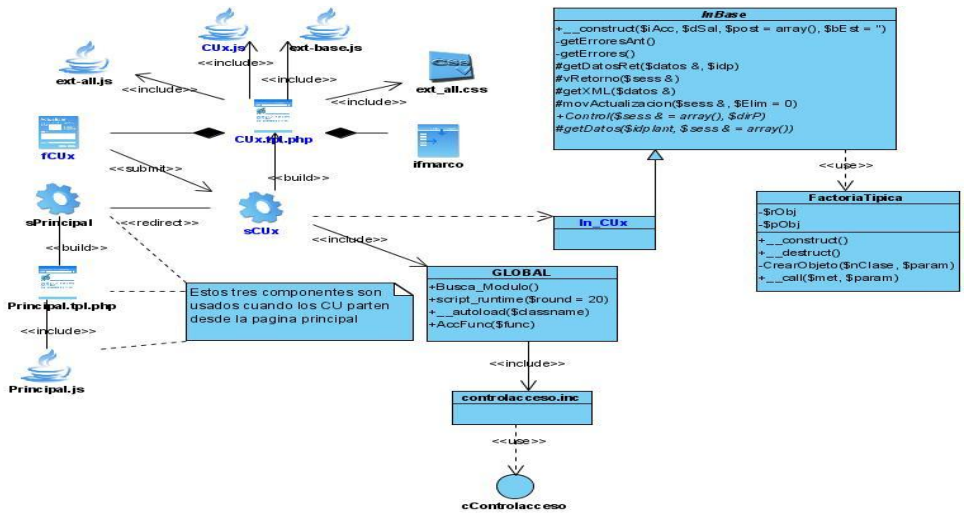


Figura 14: Diagrama Genérico de Clases del Diseño**.**

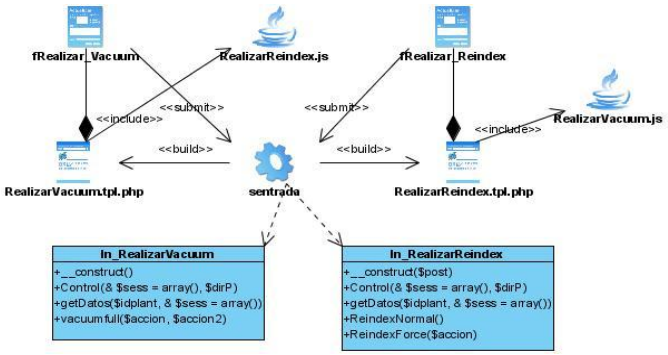


Figura 15: Diagrama de Clase del Diseño del caso de uso **"Realizar Mantenimiento".**

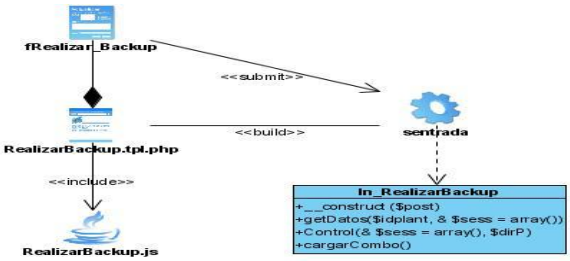


Figura 16: Diagrama de Clase del Diseño del caso de uso **"Realizar Salva".**

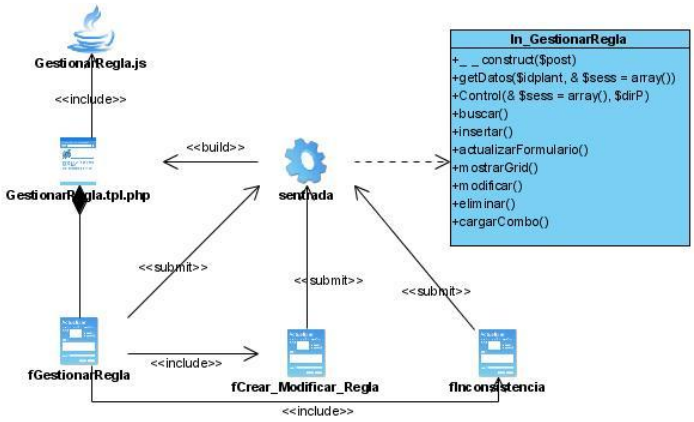


Figura 17: Diagrama de Clase del Diseño del caso de uso **"Gestionar Regla".**

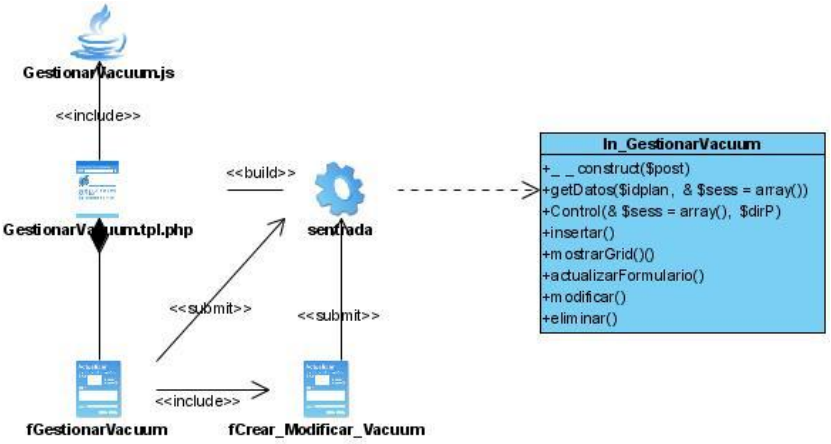


Figura 18: Diagrama de Clase del Diseño del caso de uso **"Gestionar Tarea Vacuum".**

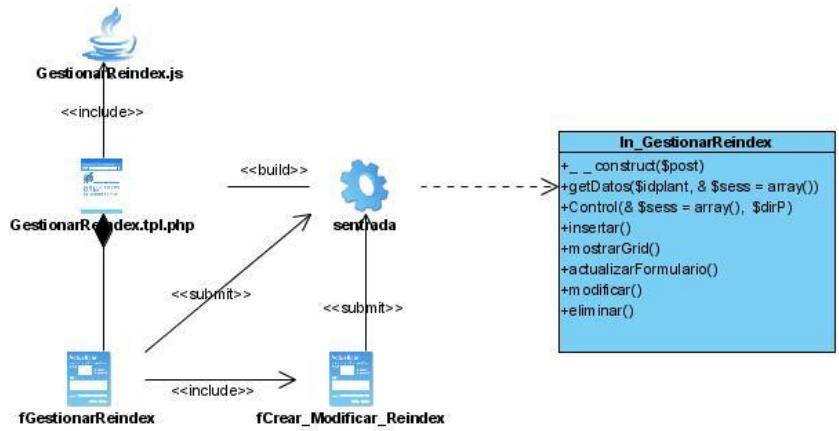


Figura 19: Diagrama de Clase del Diseño del caso de uso **"Gestionar Tarea Reindex".**

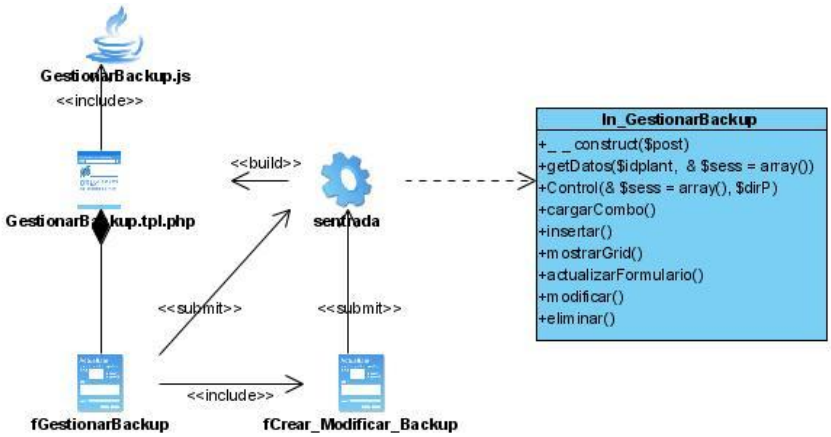


Figura 20: Diagrama de Clase del Diseño del caso de uso **"Gestionar Tarea Salva".**

3.7. Diagramas de Interacción.

Los diagramas de interacción muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas.

Los diagramas de interacción se clasifican el Diagramas de Colaboración y Diagramas de Secuencias.

Un *diagrama de colaboración* muestra las interacciones entre objetos creando enlaces entre ellos y añadiendo mensajes a esos enlaces.

Un *diagrama de secuencia* muestra las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto.

En el diseño, se decidió representar estas interacciones mediante diagramas de secuencias ya que el centro de atención principal es encontrar secuencia de interacciones detalladas y ordenadas en el tiempo.

A continuación se muestran los diagramas de interacción (secuencia) de los distintos casos de uso.

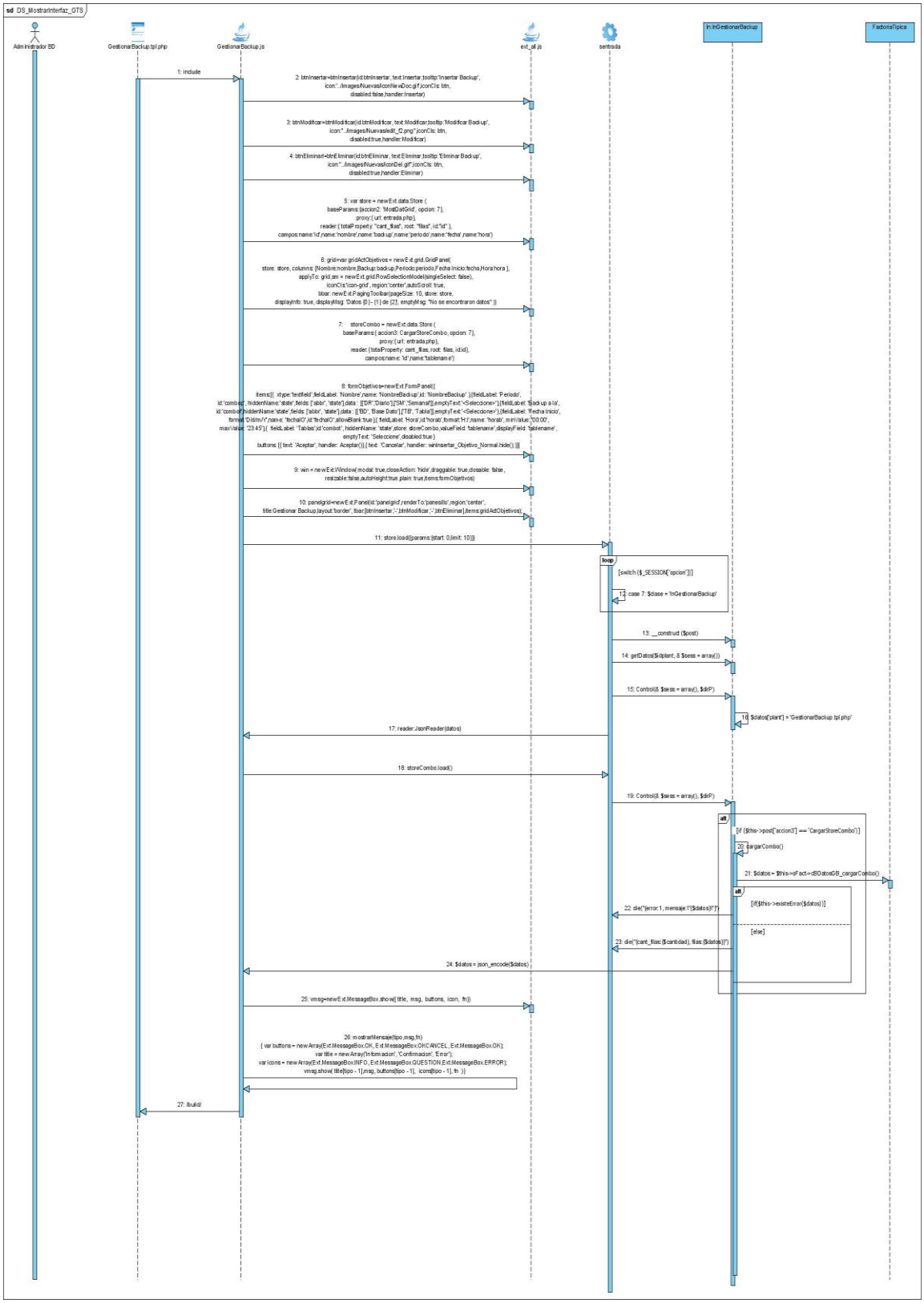


Figura 21. Realización Mostrar Interfaz (Caso de Uso Gestionar Tarea Salva).

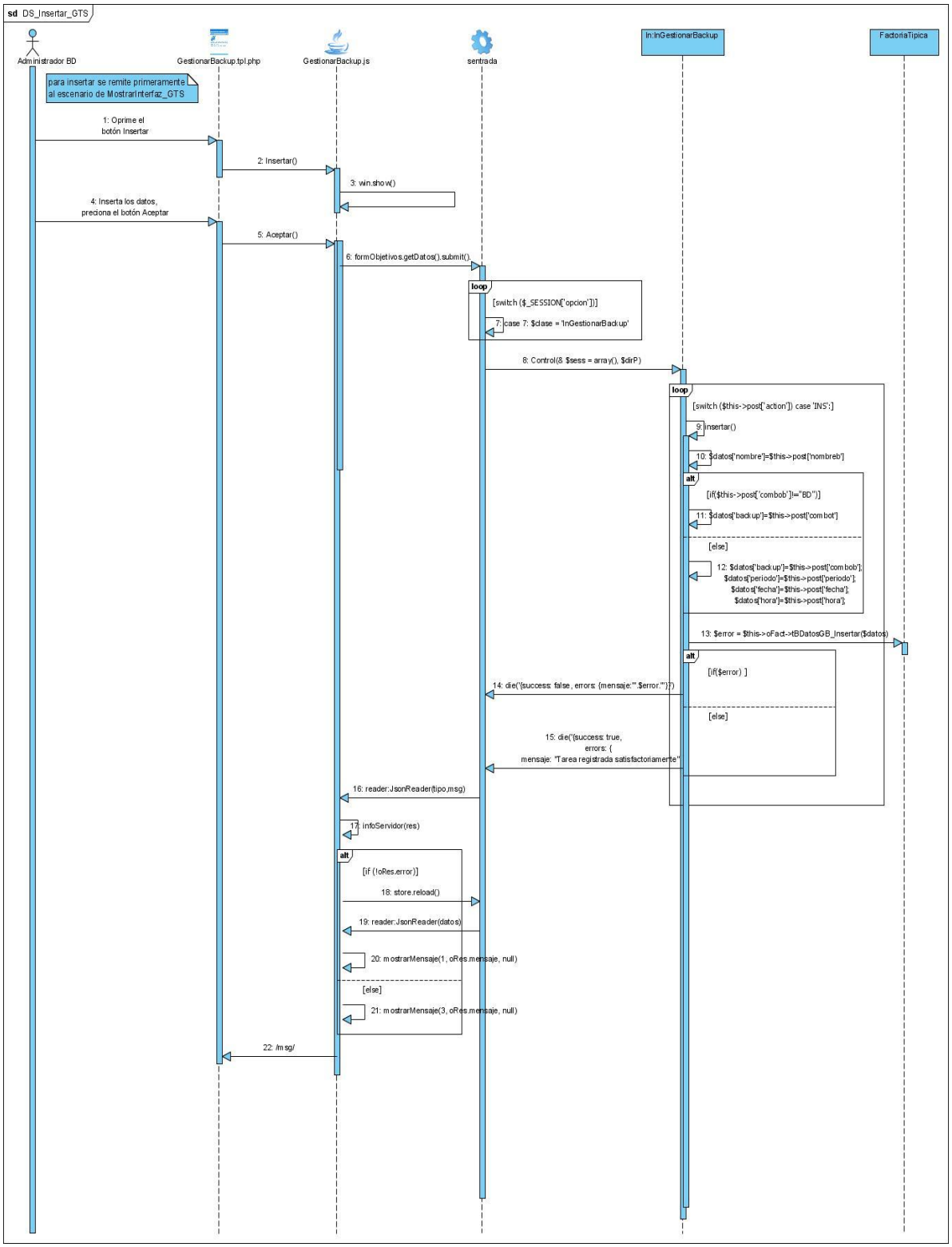


Figura 22. Realización Insertar (Caso de Uso Gestionar Tarea Salva).

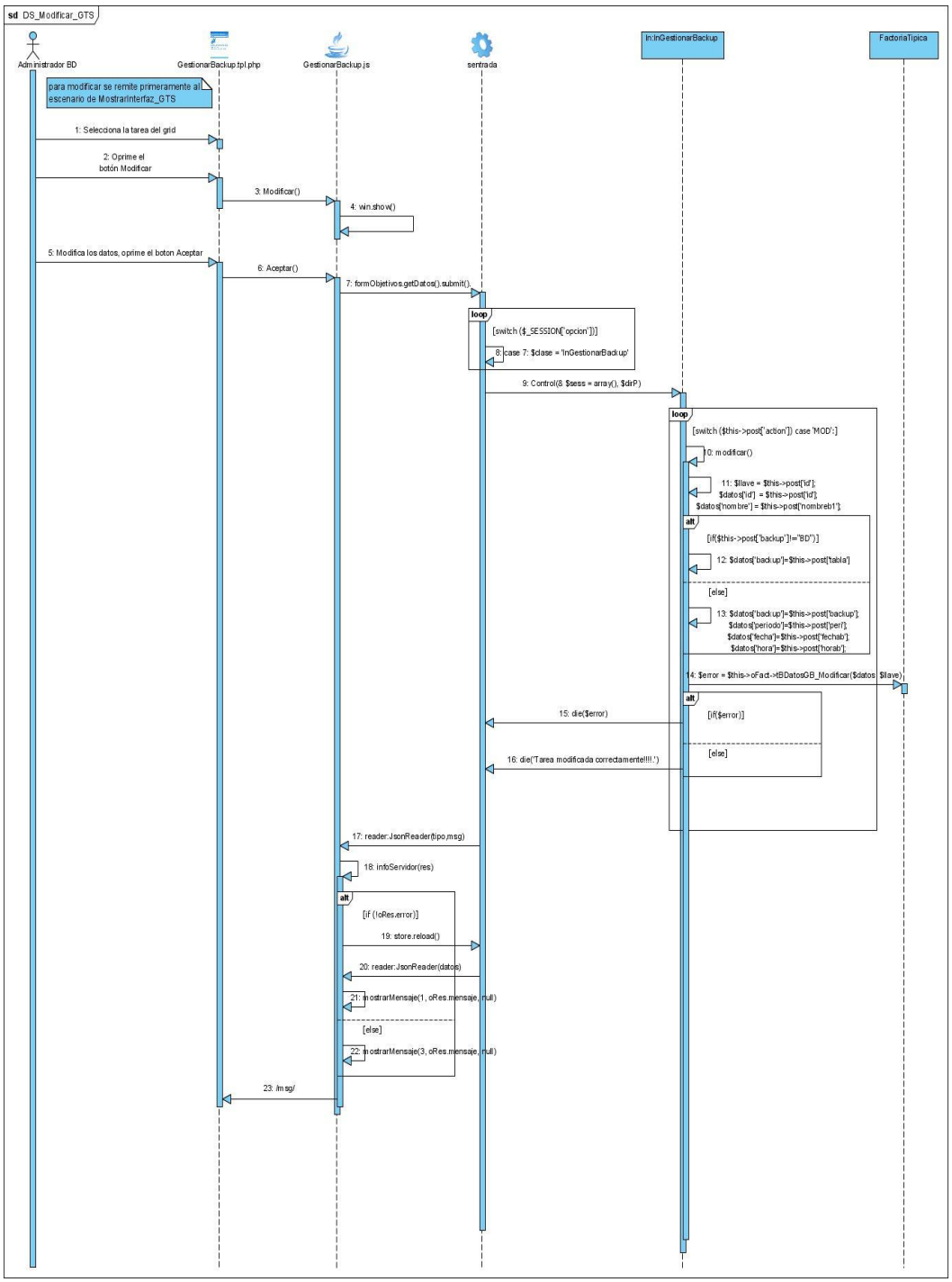


Figura 23. Realización Modificar (Caso de Uso Gestionar Tarea Salva).

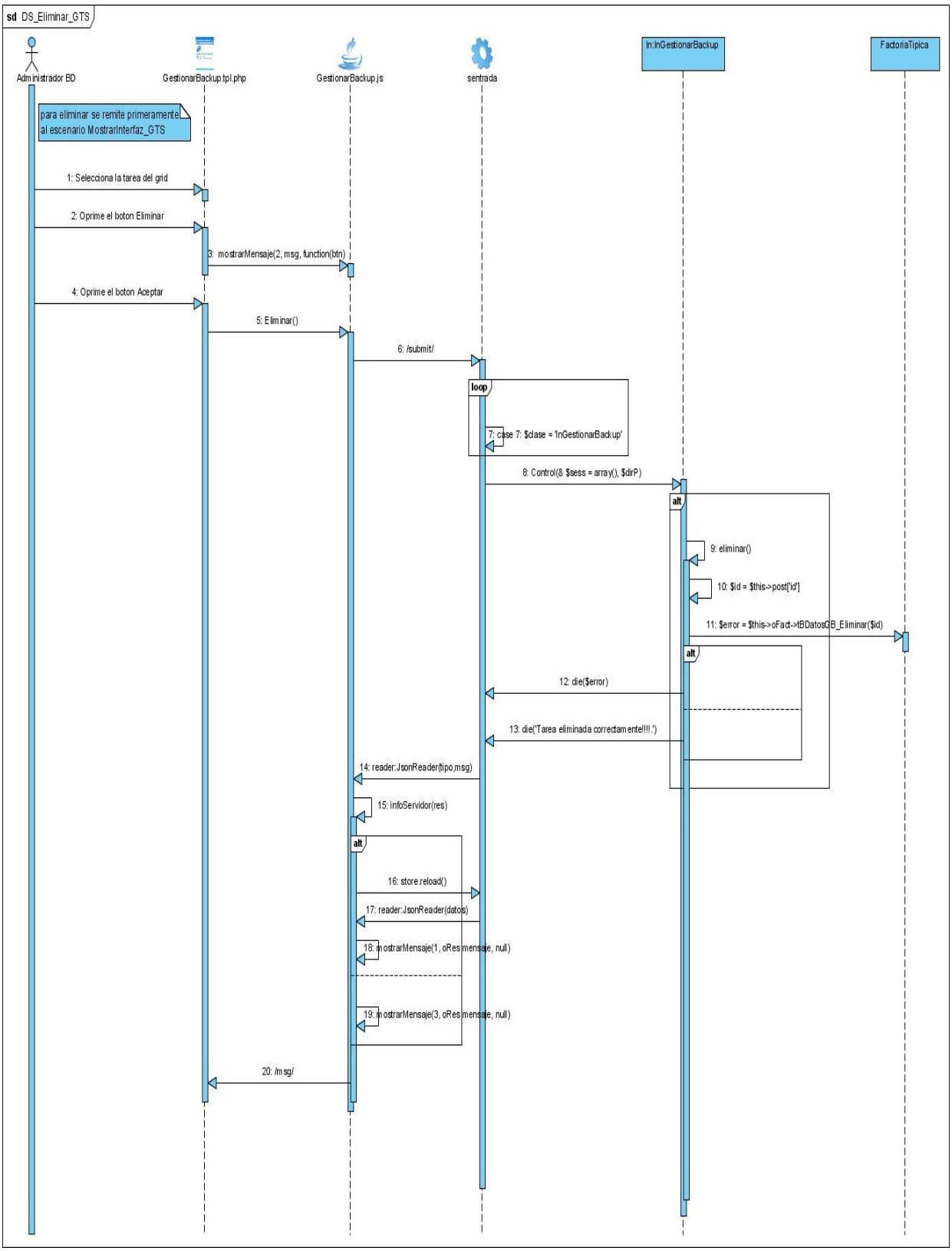


Figura 24. Realización Eliminar (Caso de Uso Gestionar Tarea Salva).

3.8 Diseño de la Base de Datos.

Clases persistentes.

Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Lo contrario son las clases temporales (transient) que son manejadas y almacenadas por el sistema en tiempo de ejecución por lo que dejan de existir cuando termina el programa.

Diagrama Entidad Relación de la BD.

Los diagramas o modelos entidad-relación 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.

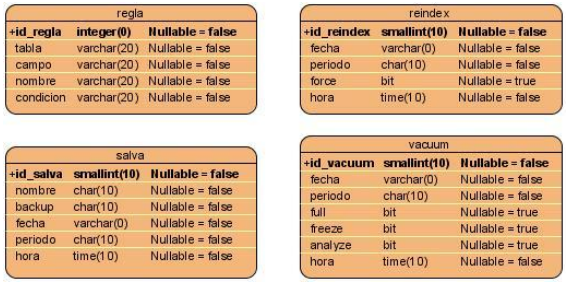


Figura 31. Diagrama que representa el Modelo de Datos.

La descripción de las tablas que contiene la Base de Datos anteriormente mostrada se podrán observar en el Anexo 2.

3.9. Funcionamiento del sistema.

El Demonio Cron es uno de los procesos mas importantes que intervienen en el funcionamiento del sistema. En el sistema operativo Unix, **cron** es un administrador regular de procesos en segundo plano (*demonio*) que ejecuta programas a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el archivo crontab.

Cron se podría definir como el "equivalente" a Tareas Programadas de Windows.

Formato del archivo crontab

Cada uno de los ficheros crontab de configuración están formados por asignaciones de valores a variables de entorno y una línea por actividad que queramos programar su ejecución. Las líneas en blanco, los espacios iniciales y los tabuladores se ignoran.

Los comentarios en este fichero son líneas completas cuyo primer carácter que no sea un espacio es un carácter #.

Las línea de programación de tareas siguen un formato estándar formada por cinco campos que indican un instante de ejecución y la ruta del fichero que hay que ejecutar.

Los campos que describen el instante de ejecución son por orden:

minuto 0-59

hora 0-23

día del mes 0-31

mes 0-12 (o su nombre con las tres primeras letras en inglés)

día semana 0-7 (0 ó 7 indica domingo, o su nombre con las tres primeras letras en inglés)

Un campo puede contener:

Un asterisco (*) para indicar todos los posibles valores.

Un valor fijo para indicar un minuto, hora, día o mes.

Un rango de valores, dos números separados por guiones. Un rango puede terminar en /numero para indicar el incremento.

Una lista de valores separados por comas.

Un valor */numero para indicar todos los valores con incremento de "número". (MARTINEZ, 2000)

Ejemplos

Ejemplo General

* * * * * línea a ejecutar

Ejecutarlo a las 12 de la noche cada día

0 0 * * * /usr/bin/fetchmail

Para agregar, quitar o modificar tareas, hay que editar el crontab. Esto se hace con la orden *crontab –e.*

Para realizar las tareas de dicho sistema el crontab ejecuta cada un minuto el fichero.php, con el objetivo de comparar la fecha y hora en que se debe realizar la(s) tarea(s) programada(s), las cuales son cargadas del fichero.txt, que a la vez este las obtiene de la base de datos cuando el sistema crea, modifica y elimina una tarea.

La base de datos del sistema cuenta con 4 tablas. En las tablas reindex, vacuum y salva se guardaran la información de las tareas a realizar según la operación que se desea ejecutar y en la tabla regla se registrará la(s) condición(es) para la búsqueda de inconsistencias en las tablas de la base de datos de Planificación Material y Financiera del MINFAR.

3.10. Estándares del 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.

Los iconos de las acciones principales que puede realizar el usuario (modificar, eliminar, insertar, aprobar, buscar) se mostrarán en la parte superior de la página.

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, es decir, se trabajará con la familia de fuentes Arial, Geneva, Helvetica, Sans-Serif, el tamaño

de la misma no debe diferir mucho de 11px. Los colores con los que se trabajarán serán tonalidades claras basadas en el azul combinado con el color blanco o gris.

Gracias a los aspectos anteriormente se garantiza que el sistema sea agradable al usuario y muy fácil de usar, pues le permite adaptarse más fácilmente al área de trabajo que ante él se despliega.

3.11. Estándares de Programación.

Resulta ventajoso utilizar un estándar para escribir código, entre dichas ventajas, se tiene las siguientes:

1. Reducir errores
2. Escribir un código comprensible y fácil de leer
3. Garantizar una buena comunicación entre los programadores del equipo
4. Facilitar el mantenimiento del software

A continuación se muestra los principales estándares aplicados:

Tabla 19. Estándares de Código.

Legibilidad **Objetivo:** Nombrar las clases e instancias de las mismas de forma estándar para todas las aplicaciones.		
Apariencia de clases y objetos	Primera letra en mayúscula y serán en singular.	Los nombres de las clases y las instancias de las mismas deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: MiClase (). Los prefijos a utilizar serán los siguientes: t--------- Clases Típicas ln------- Lógica de Negocio.

		c--------Consulta me------Manejadora de Entidad
ggNombre de clases y objetos	Relacionados al propósito	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la clase o instancia de la misma.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación **CamelCasing** [20].
Nombre de atributos	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo dentro de la clase.
Apariencia de las funciones	Primera letra en mayúscula	Los nombres de las funciones deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **PascalCasing** [21]. Si son funciones que obtienen un dato se emplea el prefijo **get** y si fijan algún valor se emplea el prefijo **set.**
Nombre de las funciones	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma dentro de la clase.
Declaración de parámetro en funciones	Agrupados por tipos primero los string, los numéricos y valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen.

Variables y constantes		
Apariencia de constantes	Todas sus letras en mayúscula	Se deben declarar las constantes con todas sus letras en mayúscula.
Apariencia de variables:	Primera letra en minúscula	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.
Nombres de las variables y constantes	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Declaración de constantes y asignación a variables	Una por cada línea	Se recomienda declarar una constante por cada línea y con las asignaciones a las variables sucede lo mismo. Ejemplo:
Identación **Objetivo:** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.		
0 espacios en blanco desde la izquierda en	Require Include Class	No se empleará ningún espacio en blanco desde la izquierda para las instrucciones antes mencionadas. Se tomará como inicio de la página el tag PHP <?
2 espacios en blanco desde la izquierda en	Function Define	Se dejarán dos espacios en blanco desde la izquierda en las instrucciones antes mencionadas.
2 espacios en	Inicio y fin de	Se recomienda dejar dos espacios en blanco

blanco desde la referencia en	bloque	desde la instrucción anterior para el inicio y fin de bloque. Lo mismo sucede para el caso de las instrucciones If, else, For, While, Do While, Switch, Foreach.
Niveles de anidación	Hasta 5 niveles	Se recomienda emplear hasta 5 niveles de anidación en instrucciones If, For, While.
Comentarios, separadores, líneas y espacios en blanco **Objetivo:** Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas. Y se comenta también cuando se cierran los ciclos, clases, instrucciones if y otras.
Separador de instrucciones	Se emplea el punto y coma.	Se recomienda usar el separador al final de cada instrucción y no en la línea de abajo.
Líneas en blanco	Se emplean antes de cada función.	Se recomienda dejar una línea en blanco antes de la definición de cada función para dar claridad al código.
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

3.12. 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.

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.

Algunos errores serán generados por funciones JavaScript para evitar la ejecución de la página en vano. Este es el caso de los formularios de inserción, modificación, se utilizan los errores en forma de mensajes de texto, como alertas de JavaScript, en la misma página donde se ejecutó la acción, de forma que el usuario pueda corregir más fácilmente y continuar. En operaciones muy largas o complicadas, se permite volver atrás, para revisar o modificar la información. Y se utilizan mensajes de confirmación, para acciones que son irreversibles como es el caso de las eliminaciones.

3.13. Conclusiones.

En este capítulo se mostró como a través del análisis y diseño, se transformó el modelo de análisis en un modelo de diseño, es decir, en una estructura de clasificadores y realizaciones de casos de uso. Además se construyeron varios artefactos para llevar a cabo el proceso de implementación del sistema. También se identificaron otras funcionalidades que se deben tener en cuenta para futuras versiones del sistema. Se utilizaron diagramas de clases Web para explicar la lógica del negocio del sistema, se crearon mecanismos de diseño que simplifican el modelado y se modeló la base de datos de la aplicación. En este momento, ya se tiene confeccionada completamente la propuesta que trae este trabajo.

CAPITULO 4. IMPLEMENTACION Y PRUEBA

4.1 Introducción.

En este capítulo se procede a la implementación del sistema y a la realización de pruebas del software para a través de las mismas obtener una revisión final de las especificaciones, del diseño y de la codificación.

Para el desarrollo de este capítulo se tuvieron en cuenta algunos artefactos que fueron generados en el flujo de análisis y diseño. A partir de los cuales se desarrolló la implementación del sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

4.2 Modelo de Implementación.

El modelo de implementación constituye la vista de Implementación de la arquitectura, y como tal, guía las labores de construcción del sistema. Este contiene fundamentalmente los subsistemas de implementación, incluyendo las dependencias y otras informaciones necesarias para su utilización.
Para lograr una mejor comprensión de los componentes que forman el sistema, se presentarán los diagramas de componentes que describen los elementos físicos y lógicos del sistema y sus relaciones.

Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente, las mismas pueden ser de **import** entre subsistemas y de **compilación** entre paquetes de componentes o componentes.

El diagrama que se presenta a continuación tiene como objetivo figurar la estructura general de la aplicación en desarrollo, en términos de componentes, el mismo está estructurado por capas, representando así el estilo arquitectónico utilizado en el sistema. Los subsistemas que se

incluyen en el mismo son agrupados por el criterio de funcionalidad y están compuestos por sus respectivos paquetes de componentes.

A continuación se muestra en la Figura 36 el diagrama de componentes general y el de los paquetes de componentes que conforman los subsistemas se podrán observar en el Anexo 3.

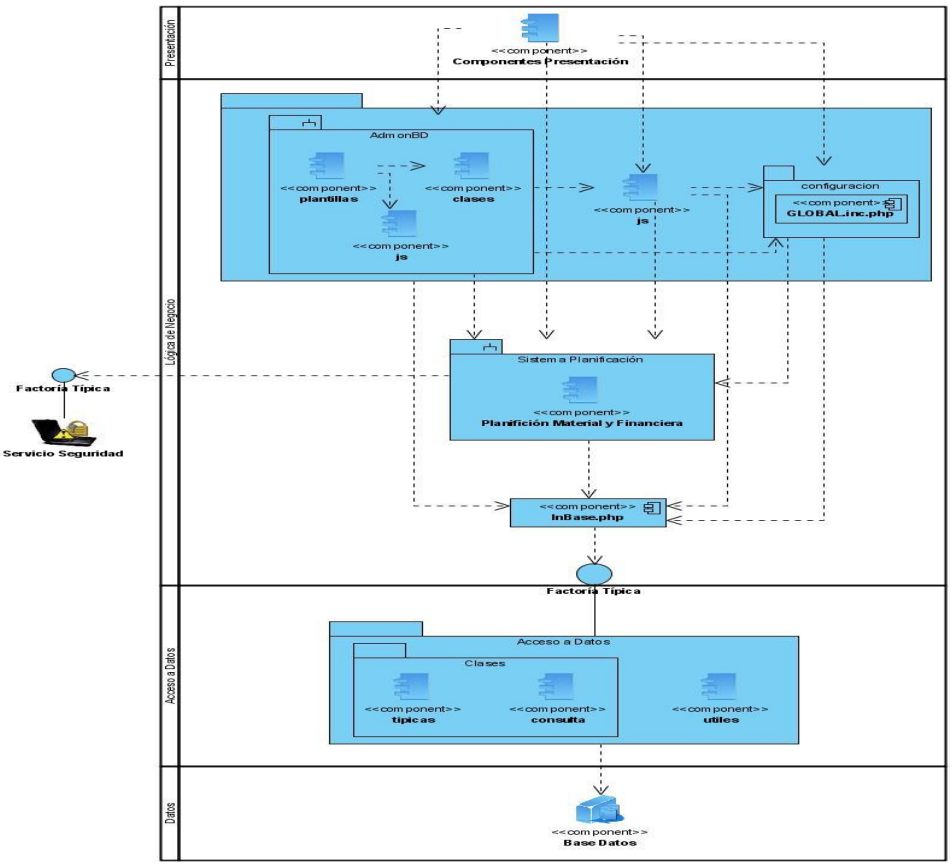


Figura 32: Diagrama general de Componentes.

4.3 Modelo de Despliegue.

El diagrama de despliegue representa la arquitectura de tiempo de ejecución de los procesadores, dispositivos y los componentes de software que se ejecutan en esa arquitectura. Es la última descripción física de la topología del sistema y describe la estructura de las unidades de hardware. Además, representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Un nodo es un recurso de ejecución tal como un procesador, un dispositivo o memoria. En los procesadores es donde se encuentran alojados los componentes.

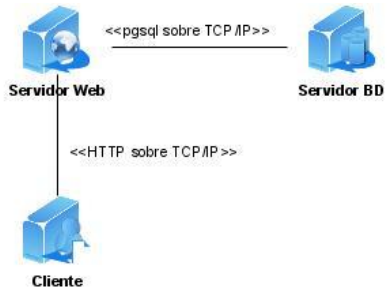


Figura 39: Diagrama de Despliegue.

El diagrama de despliegue mostrado anteriormente representa un modelo ideal que está en correspondencia con la arquitectura cliente-servidor en capas, compuesto por un Servidor Web, un Servidor de Base de Datos, una computadora para visualizar las operaciones sobre ambos servidores y las conexiones entre ellos.

4.4 Modelo de Prueba.

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dentro de las pruebas se encuentran dos fundamentales*: prueba de caja negra y de caja blanca*.

La prueba de la **caja blanca** del software comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

La prueba de **caja negra** se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, a través de los casos de prueba se demuestra que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

La prueba de caja negra es la que se le aplicó al sistema. A continuación se muestran los casos de prueba.

Caso de Uso: Realizar Mantenimiento.

1. Seleccionar el tipo de vacuum y ejecutarlo.
2. Seleccionar el reindex y ejecutarlo.

Figura 40 y 41: Caso de Prueba **"Realizar Mantenimiento"**.

Tabla 20. Caso de prueba "Realizar Mantenimiento".

Caso de Uso	**Realizar Mantenimiento.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Vacuum seleccionado = "FULL".	Como el vacuum seleccionado no presenta ninguna dificultad, el mismo puede ser ejecutado.	El sistema ejecuta el vacuum seleccionado y muestra al usuario un mensaje que el vacuum se ejecutó satisfactoriamente.
2	Reindex seleccionado = "FORCE".	Como el reindex seleccionado no presenta ninguna dificultad, el mismo puede ser ejecutado.	El sistema ejecuta el reindex seleccionado y muestra al usuario un mensaje que el reindex se ejecutó satisfactoriamente.

Caso de Uso: Realizar Salva.

1. Realizar la salva satisfactoriamente.

Figura 42: Caso de Prueba **"Realizar Salva"**.

Tabla 21. Caso de prueba "Realizar Salva".

Caso de Uso	**Realizar Salva.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Nombre = "backup" y Backup a la = "Base a Datos".	Como el nombre de la salva fue insertado, no presenta ninguna dificultad para ser ejecutada.	El sistema ejecuta la salva seleccionada guardando en un fichero a la misma.

Caso de Uso: Gestionar Regla.

1. Se crea una regla satisfactoriamente.
2. No se crea una regla satisfactoriamente.
3. Se modifica una regla.
4. Se elimina una regla satisfactoriamente.
5. No se elimina una regla satisfactoriamente.

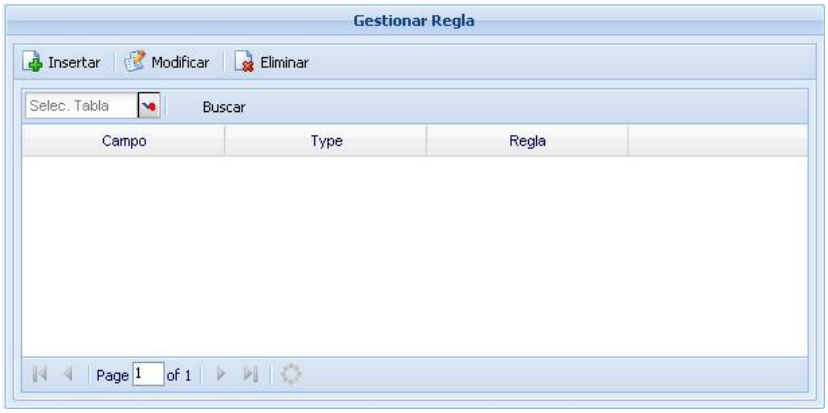


Figura 43: Caso de Prueba **"Gestionar Regla"**.

Tabla 22. Caso de prueba "Gestionar Regla".

Caso de Uso	**Gestionar Regla.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Intervalo insertado = "0" y "100". MIN seleccionado = "3" y MAX = "10". Tamaño insertado = "6".	Como insertó los datos correctamente, la regla no presenta ninguna dificultad para ser creada.	El sistema muestra la regla creada al usuario.
2	Intervalo insertado = "" y "". MIN seleccionado = "5" y MAX = "12". Tamaño insertado = "10".	Como no insertó los datos del intervalo, la regla presenta dificultad para ser crearla.	El sistema le muestra al usuario un mensaje que falta los datos del intervalo.

3	Tamaño seleccionado = “6”. Tamaño modificado = “5”.	Como el tamaño fue cambiado, la regla no presenta dificultad en ser modificada.	El sistema le muestra al usuario la regla modificada.
4	Regla del intervalo = “0” y “100”.	Como la regla se encuentra insertada, no presenta dificultad alguna para eliminarla.	El sistema le muestra al usuario que la regla fue eliminada satisfactoriamente.
5	Regla seleccionada = “”.	Como no seleccionó ninguna regla, presenta dificultad para eliminarla.	El sistema le muestra un mensaje al usuario que no seleccionó regla.

Caso de Uso: Gestionar Tarea Vacuum.

1. Se crea una tarea vacuum satisfactoriamente.
2. No se crea una tarea vacuum satisfactoriamente.
3. Se modifica una tarea vacuum.
4. Se elimina una tarea vacuum satisfactoriamente.
5. No se elimina una tarea vacuum satisfactoriamente.

Figura 44: Caso de Prueba **“Gestionar Tarea Vacuum”**.

Tabla 23. Caso de prueba "Gestionar Tarea Vacuum".

Caso de Uso	**Gestionar Tarea Vacuum.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Tipo de vacuum seleccionado = "FULL" y "ANALYZE". Fecha Inicio = "Thu/05/06/08". Periodo = "Diariamente". Hora = "01:00 ".	Como insertó los datos correctamente, la tarea vacuum no presenta ninguna dificultad para ser creada.	El sistema muestra la tarea vacuum creada al usuario.
2	Tipo de vacuum seleccionado = "". Fecha Inicio = "Thu/05/06/08". Periodo = "Semanalmente". Hora = "02:15 ".	Como no seleccionó el tipo de vacuum, la tarea vacuum presenta dificultad para ser crearla.	El sistema le muestra al usuario un mensaje que seleccione el tipo de vacuum.
3	Tarea vacuum seleccionada con vacuum seleccionado = "FULL" y "ANALYZE". Tipo de vacuum modificado = "FREEZE"	Como el tipo de vacuum fue cambiado, la tarea vacuum no presenta dificultad en ser modificada.	El sistema le muestra al usuario la tarea vacuum modificada.

	y “ANALYZE”		
4	Tarea seleccionada. Vacuum seleccionado = “FREEZE”y “ANALYZE”. Fecha Inicio = “Thu/05/06/08”. Periodo = “Diariamente”. Hora = “01:00 ”.	Como la tarea vacuum se encuentra insertada, no presenta dificultad alguna para eliminarla.	El sistema le muestra al usuario que la tarea vacuum fue eliminada satisfactoriamente.
5	Tarea vacuum seleccionada = “”.	Como no seleccionó ninguna tarea vacuum, presenta dificultad para eliminarla.	El sistema le muestra un mensaje al usuario que no seleccionó tarea vacuum.

Caso de Uso: Gestionar Tarea Reindex.

1. Se crea una tarea reindex satisfactoriamente.
2. No se crea una tarea reindex satisfactoriamente.
3. Se modifica una tarea reindex.
4. Se elimina una tarea reindex satisfactoriamente.
5. No se elimina una tarea reindex satisfactoriamente.

Figura 45: Caso de Prueba **"Gestionar Tarea Reindex"**.

Tabla 24. Caso de prueba "Gestionar Tarea Reindex".

Caso de Uso	**Gestionar Tarea Reindex.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Tipo de reindex seleccionado="FORCE". Fecha Inicio = "Tue/03/06/08". Periodo = "Semanal". Hora = "03:00".	Como insertó los datos correctamente, la tarea reindex no presenta ninguna dificultad para ser creada.	El sistema muestra la tarea reindex creada al usuario.
2	Tipo de reindex seleccionado = "FORCE". Fecha Inicio = "Tue/10/06/08". Periodo = "Semanal".	Como no seleccionó la hora, la tarea reindex presenta dificultad para ser crearla.	El sistema le muestra al usuario un mensaje que seleccione la hora.

	Hora = “”.		
3	Tarea reindex con Periodo seleccionado = “Semanal” Periodo modificado =” Diariamente”.	Como el periodo fue cambiado, la tarea reindex no presenta dificultad en ser modificada.	El sistema le muestra al usuario la tarea reindex modificada.
4	Tarea seleccionada. Reindex seleccionado=“FORCE”. Fecha Inicio = “Tue/03/06/08”. Periodo= “Diariamente”. Hora = “03:00”.	Como la tarea reindex se encuentra insertada, no presenta dificultad alguna para eliminarla.	El sistema le muestra al usuario que la tarea vacuum fue eliminada satisfactoriamente.
5	Tarea reindex seleccionada = “”.	Como no seleccionó ninguna tarea reindex, presenta dificultad para eliminarla.	El sistema le muestra un mensaje al usuario que no seleccionó tarea reindex.

Caso de Uso: Gestionar Tarea Salva.

1. Se crea una tarea salva satisfactoriamente.
2. No se crea una tarea salva satisfactoriamente.
3. Se modifica una tarea salva.
4. Se elimina una tarea salva satisfactoriamente.
5. No se elimina una tarea salva satisfactoriamente.

Figura 46: Caso de Prueba **"Gestionar Tarea Salva"**.

Tabla 25. Caso de prueba "Gestionar Tarea Salva".

Caso de Uso	**Gestionar Tarea Salva.**		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Nombre insertado= "prueba". Backup="Base de Datos". Fecha Inicio=" Wed/11/06/2008". Periodo= "Diario". Hora = "02:30".	Como insertó los datos correctamente, la tarea salva no presenta ninguna dificultad para ser creada.	El sistema muestra la tarea salva creada al usuario.
2	Nombre insertado= "". Backup="Base de Datos". Fecha Inicio="	Como no insertó el nombre, la tarea salva presenta dificultad para ser crearla.	El sistema le muestra al usuario un mensaje que inserte el nombre.

	Tue/10/06/2008”. Periodo= “Semanal”. Hora = “03:00”.		
3	Tarea con nombre seleccionado = “prueba” Nombre modificado = “prueba1”.	Como el nombre fue cambiado, la tarea salva no presenta dificultad en ser modificada.	El sistema le muestra al usuario la tarea salva modificada.
4	Tarea seleccionada. Nombre = “prueba1”. Backup=”Base de Datos”. Fecha Inicio=” Wed/11/06/2008”. Periodo= “Diario”. Hora = “02:30”.	Como la tarea salva se encuentra insertada, no presenta dificultad alguna para eliminarla.	El sistema le muestra al usuario que la tarea salva fue eliminada satisfactoriamente.
5	Tarea salva seleccionada = “”.	Como no seleccionó ninguna tarea salva, presenta dificultad para eliminarla.	El sistema le muestra un mensaje al usuario que no seleccionó tarea salva.

4.5 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

1. Para el desarrollo de este sistema se realizó una investigación de forma exhaustiva de los procesos relacionados con base de datos para realizar el mantenimiento de la misma.
2. Se tuvo en cuenta los requerimientos del sistema.
3. Se logró una especificación de los requisitos fundamentales para el diseño del Sistema.
4. Se definieron los casos de uso que satisfacen los requisitos especificados.
5. Se obtuvo un diseño de la base de datos.
6. Se documentó, de acuerdo con RUP, lo desarrollado.
7. El sistema que se realizó cumple con los objetivos trazados.

RECOMENDACIONES

Los objetivos de este trabajo han sido cumplidos, pero durante el desarrollo del mismo surgieron ideas que se recomiendan desarrollarse en el futuro, de forma que se logre una mayor eficiencia en la aplicación. A continuación se citan algunas de ellas:

1. Brindar la posibilidad de que el sistema sea utilizado en otros módulos del proyecto del MINFAR y también extensible a proyectos externos.
2. Realizar una ayuda al sistema para darle un mejor entendimiento al mismo en otros sectores del MINFAR y fuera de este.
3. Estudiar otras funcionalidades e integrarlas al sistema, para que el mismo sea más consistente y confiable, como la de tratamiento de las inconsistencias localizadas.
4. Realizar una funcionalidad que permita obtener el fichero ”backup”.
5. Reutilizar la metodología, las tecnologías y herramientas empleadas en la construcción de la aplicación, para el desarrollo de futuras iteraciones de la misma.

REFERENCIAS BIBLIOGRÁFICAS

1. (FSF), L. F. P. E. S. L. *Base de datos* Disponible en: http://es.wikipedia.org/wiki/Base_de_datos/.

2. WORSLEY, J. y DRAKE, J. ¿Qué es PostgreSQL? 2001, nº Disponible en: http://www.sobl.org/traducciones/practical-postgres/node13.html.

3. POSTGRESQL, E. D. D. D. Manual del usuario de PostgreSQL. 2007, nº Disponible en: http://www.librosdeluz.net/tag/Libros-de-Bases-de-Datos.

4. HUERTA, A. V. *Seguridad en Unix y redes – El demonio syslogd* Disponible en: http://www.wikilearning.com/tutorial/seguridad_en_unix_y_redes,el_demonio_syslogd/9777-27.

5. DURAN, S. G. *MANUAL BÁSICO DE CRON* Disponible en: http://www.linuxtotal.com.mx/index.php?cont=info_admon_006/.

6. MARTINEZ, P. P. F. *Configuración de Cron* Disponible en: http://dns.bdat.net/documentos/cron/x50.html.

7. JACOBSON, I., BOOCH, G., & RUMBAUGH, J. (2000). *El Proceso Unificado de Desarrollo de Software.* ADDISON WESLEY.

BIBLIOGRAFIA

(FSF), L. F. P. E. S. L. *Base de datos* Disponible en: http://es.wikipedia.org/wiki/Base_de_datos/.

DURAN, S. G. *MANUAL BÁSICO DE CRON* Disponible en: http://www.linuxtotal.com.mx/index.php?cont=info_admon_006/.

HUERTA, A. V. *Seguridad en Unix y redes – El demonio syslogd* Disponible en: http://www.wikilearning.com/tutorial/seguridad_en_unix_y_redes,el_demonio_syslogd/9777-27.

MARTINEZ, P. P. F. *Configuración de Cron* Disponible en: http://dns.bdat.net/documentos/cron/x50.html.

POSTGRESQL, E. D. D. D. Manual del usuario de PostgreSQL. 2007, nº Disponible en: http://www.librosdeluz.net/tag/Libros-de-Bases-de-Datos.

S.PRESSMAN, R. *Ingeniería del Software:Un Enfoque Práctico.* Editado por: Luis Joyanes Aguilar, D. D. D. D. L., Sistemas Informáticos E Ingeniería De Software. Madrid Carchelejo: 2001. vol. Quinta edición,

TORANZO, Y. E. y DERONCERE, A. F. *Sistema Automatizado para la Planificación Material y Financiera del MINFAR.* Tutor: Díaz, R. M. T. Tesis de Grado, Instituto Superior Politécnico José Antonio Echeverría, 2006.

VALIDO, I. R. y MEDEL, L. M. G. *Sistema Informático de Gestión para la Planificación de los Órganos Consumidores del MINFAR.* Tutor: Díaz, I. R. M. T. Tesis de Grado, Universidad de las Ciencias Informáticas, 2007.

WORSLEY, J. y DRAKE, J. ¿Qué es PostgreSQL? 2001, nº Disponible en: http://www.sobl.org/traducciones/practical-postgres/node13.html.

GLOSARIO DE TÈRMINOS

1. **PostgreSQL:** Es un servidor de base de datos relacional libre, liberado bajo la licencia BSD.
2. **Benchemarking:** Es una técnica utilizada para medir el rendimiento de un sistema o componente de un sistema.
3. **Commit:** Aacción de cometer, se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos.
4. **Subselect:** Permite realizar comparaciones con valores obtenidos en otra sentencia select anidada.
5. **Tracking:** Es la realización de un seguimiento de eventos.
6. **Bitácoras:** Es un registro escrito de las acciones que se llevaron a cabo en cierto trabajo o tarea. Incluye todos los sucesos que tuvieron lugar durante la realización de dicha tarea, las fallas que se produjeron, los cambios que se introdujeron y los costos que ocasionaron.
7. **DDL:** Un **lenguaje de definición de datos** (**D**ata **D**efinition **L**anguage, DDL por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.
8. **UNIX:** Es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.
9. **Gurús:** Significa maestro.
10. **HTML:** Acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Este lenguaje se basa en tags (instrucciones que le dicen al texto como deben mostrarse) y atributos (parámetros que dan valor al tag). Es el estándar usado en el World Wide Web.
11. **XML:** Acrónimo de "EXtensible Markup Language". Es decir, lenguaje de marcas extensible, es de reciente creación (febrero de 1998). XML es un metalenguaje, o sea, sirve para crear lenguajes. Es más amplio, más rico y más dinámico que HTML. Fue diseñado para permitir la descripción de información contenida en el WWW a través de estándares y

formatos comunes, de manera que tanto los usuarios de Internet como programas específicos (agentes), puedan buscar, comparar y compartir información en la red.

12. **Programación Orientada A Objetos:** La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.
13. **UML:** Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por la OMG (Grupo dedicado a la promoción de la tecnología orientada a objetos y su estandarización).
14. **RUP:** El Proceso Racional Unificado o RUP (Rational Unified Process), es un proceso de desarrollo de software.
15. **IDE:** Siglas de: Integrated Drive Electronics ó Integrated development environment, es decir, un entorno integrado de desarrollo.
16. **CASE:** Acrónimo inglés de Computer Aided Software Engineering, que viene a significar Ingeniería de Software Asistida por Ordenador.
17. **PDF:** Es la Extensión que corresponde a un tipo de fichero (un libro electrónico) creado con Adobe Acrobat.
18. **HTTP:** Es el protocolo de la Web (WWW), usado en cada transacción. Las letras significan Hyper Text Transfer Protocol, es decir, protocolo de transferencia de hipertexto. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceder a una página web, y la respuesta de esa web, remitiendo la información que se verá en pantalla.
19. **Lógica de negocio:** Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Recibiendo las peticiones del usuario y enviando las respuestas tras el proceso.

20. **Notación CamelCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.
21. **Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula. Ejemplo: NotacionPascalCasing.

ANEXOS

Anexo 1.

Diagramas de Secuencia.

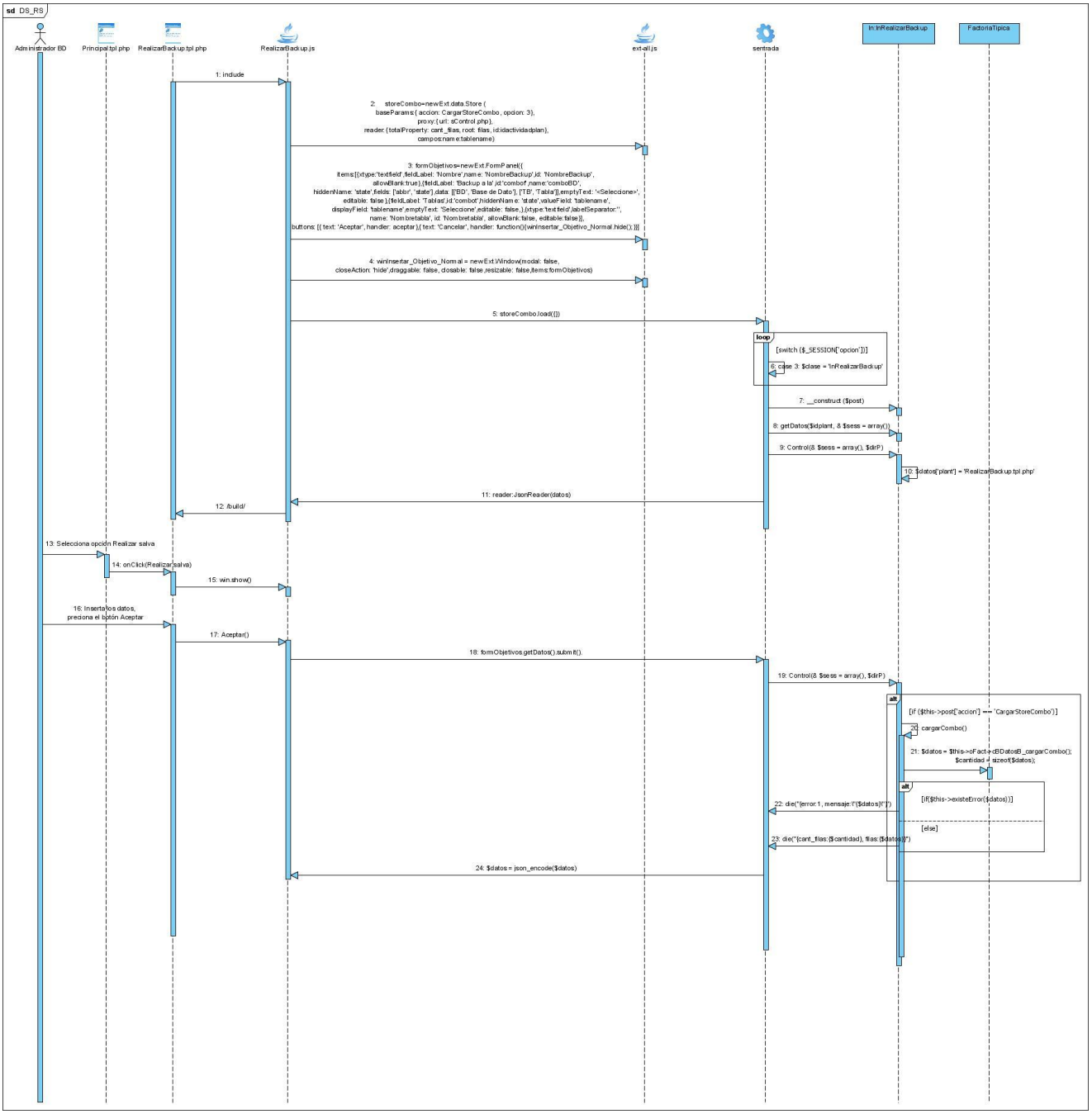


Figura 25. Diagrama de secuencia del Caso de Uso Realizar Salva.

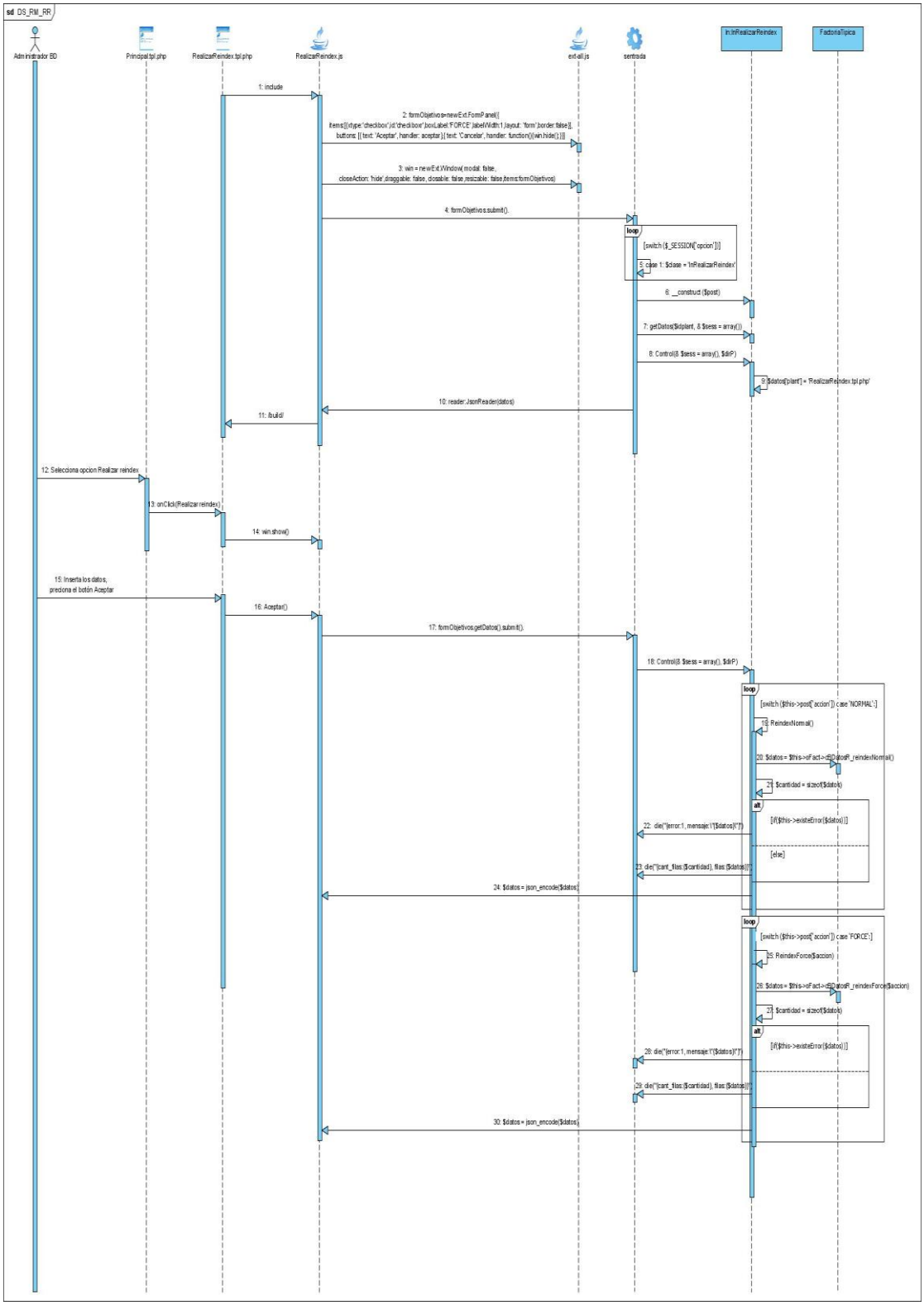


Figura 26. Realización Realizar Reindex (Caso de Uso Realizar Mantenimiento).

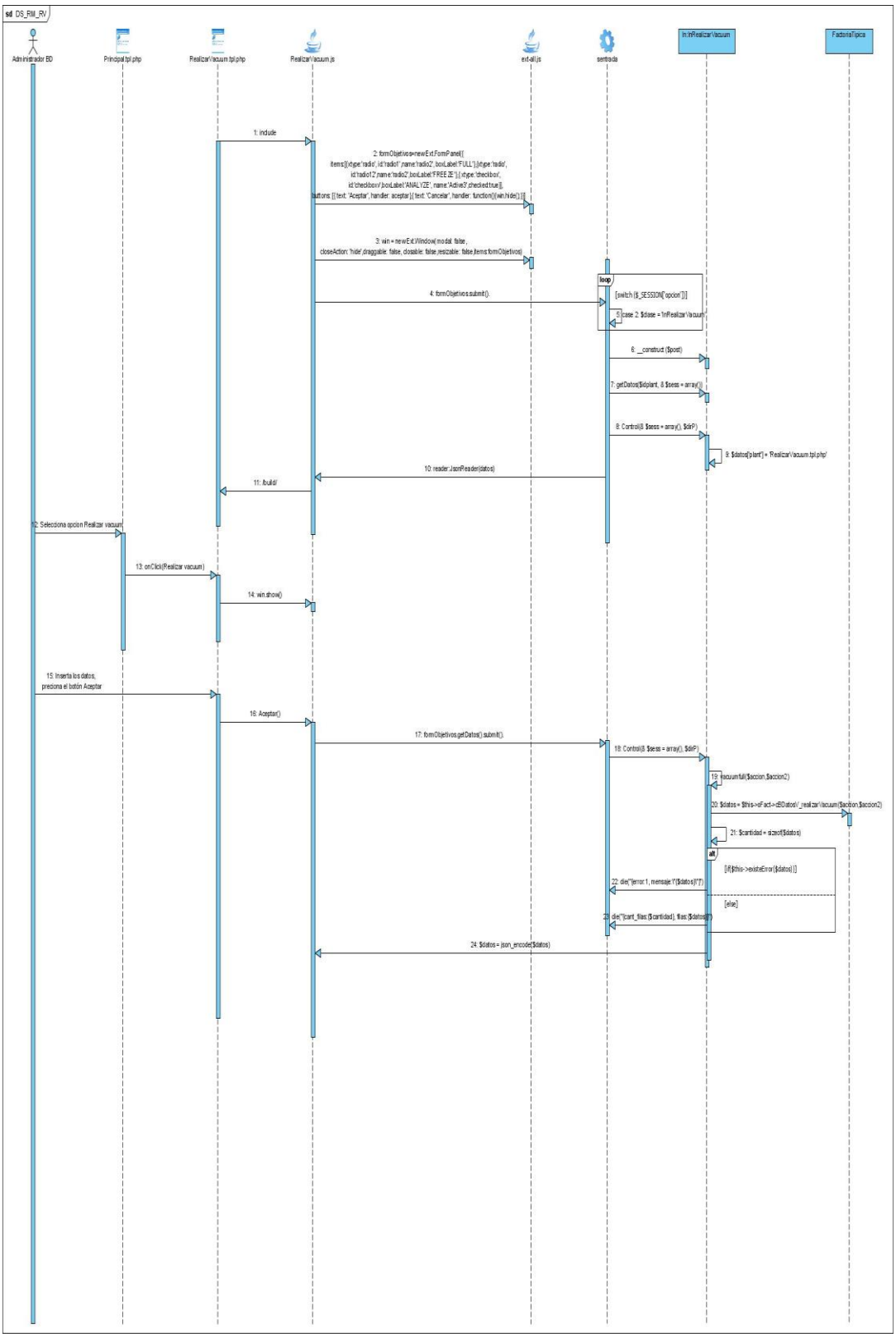


Figura 27. Realización Realizar Vacuum (Caso de Uso Realizar Mantenimiento).

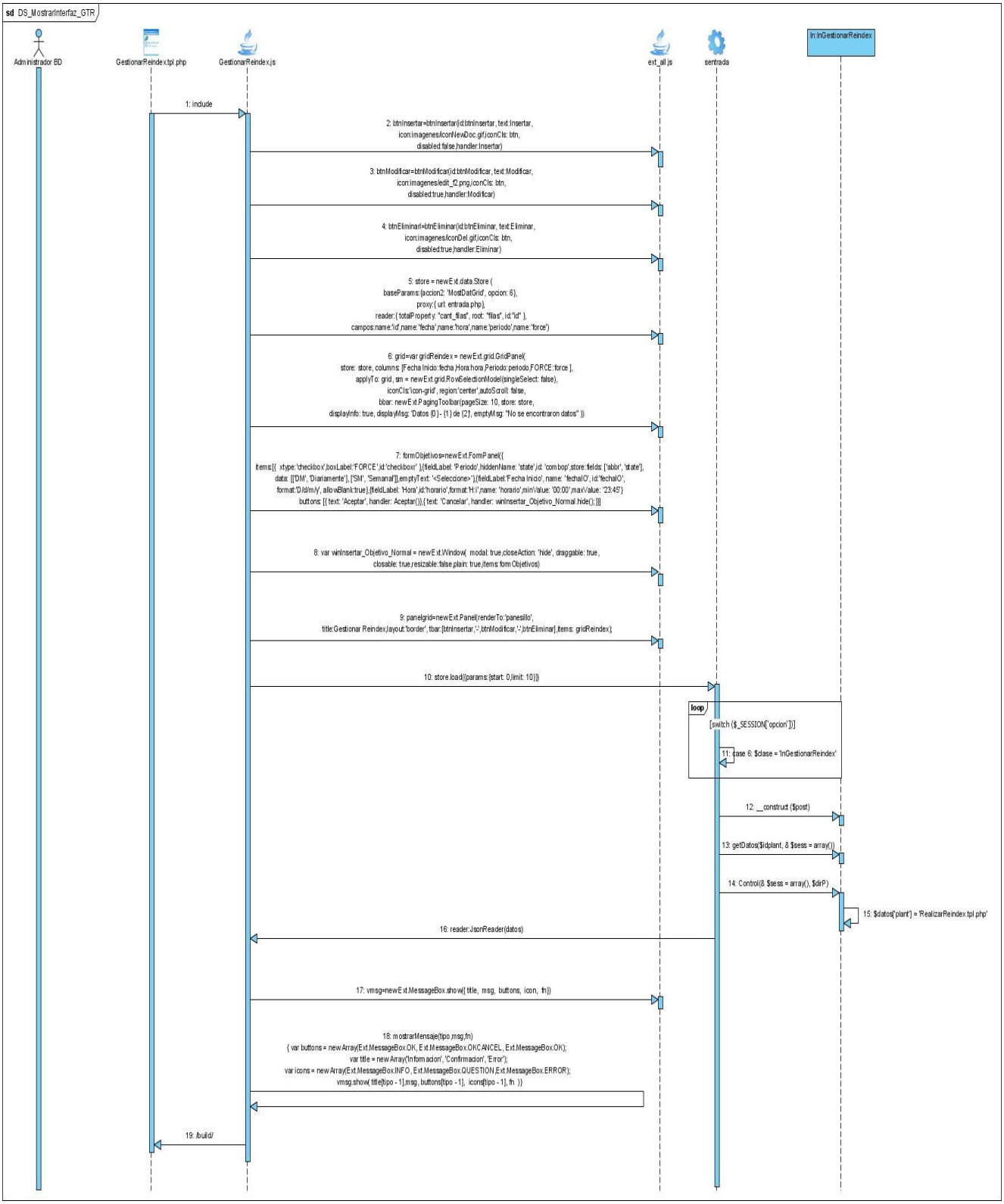


Figura 28. Realización Mostrar Interfaz (Caso de Uso Gestionar Tareas Reindex).

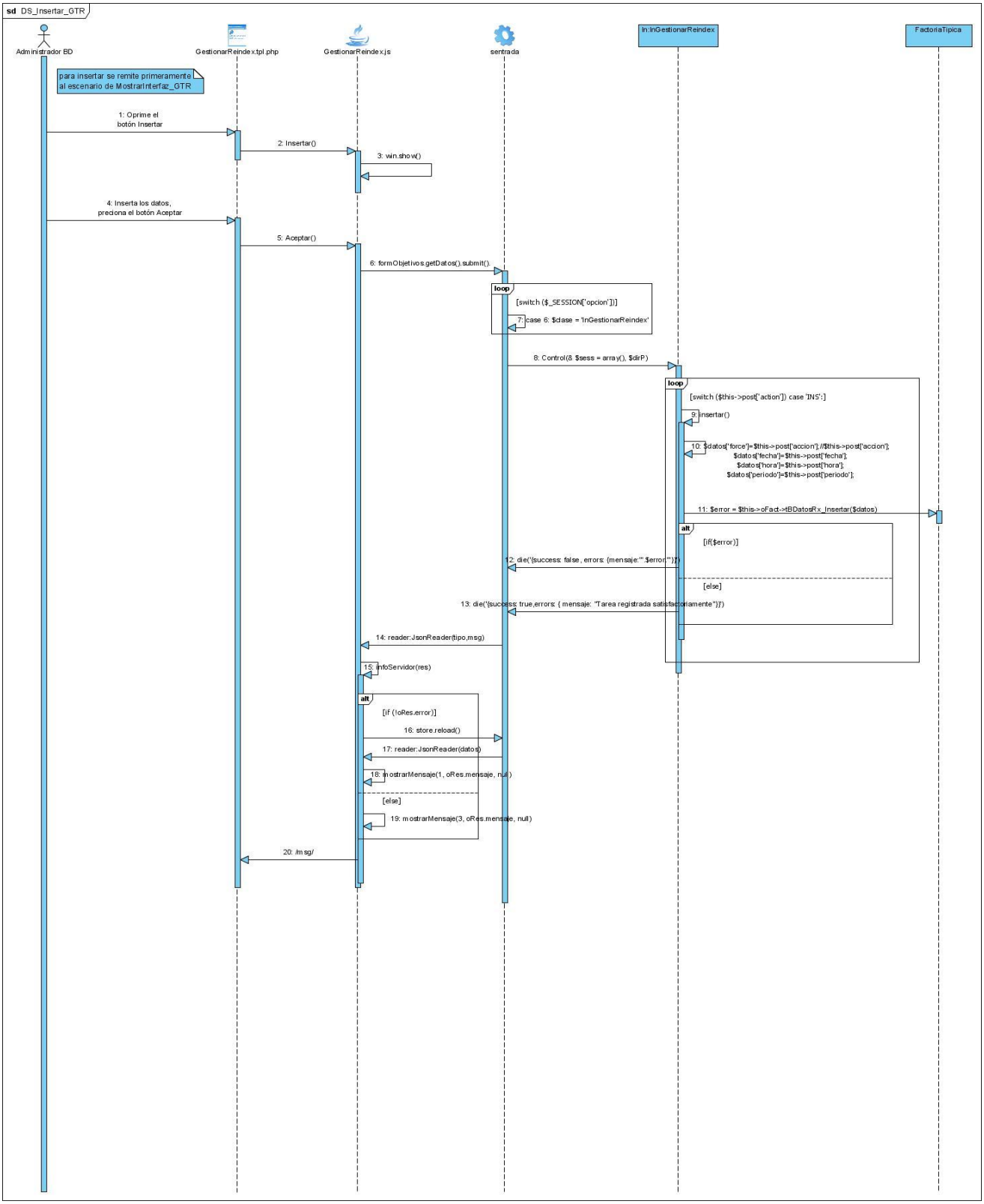


Figura 29. Realización Insertar (Caso de Uso Gestionar Tareas Reindex).

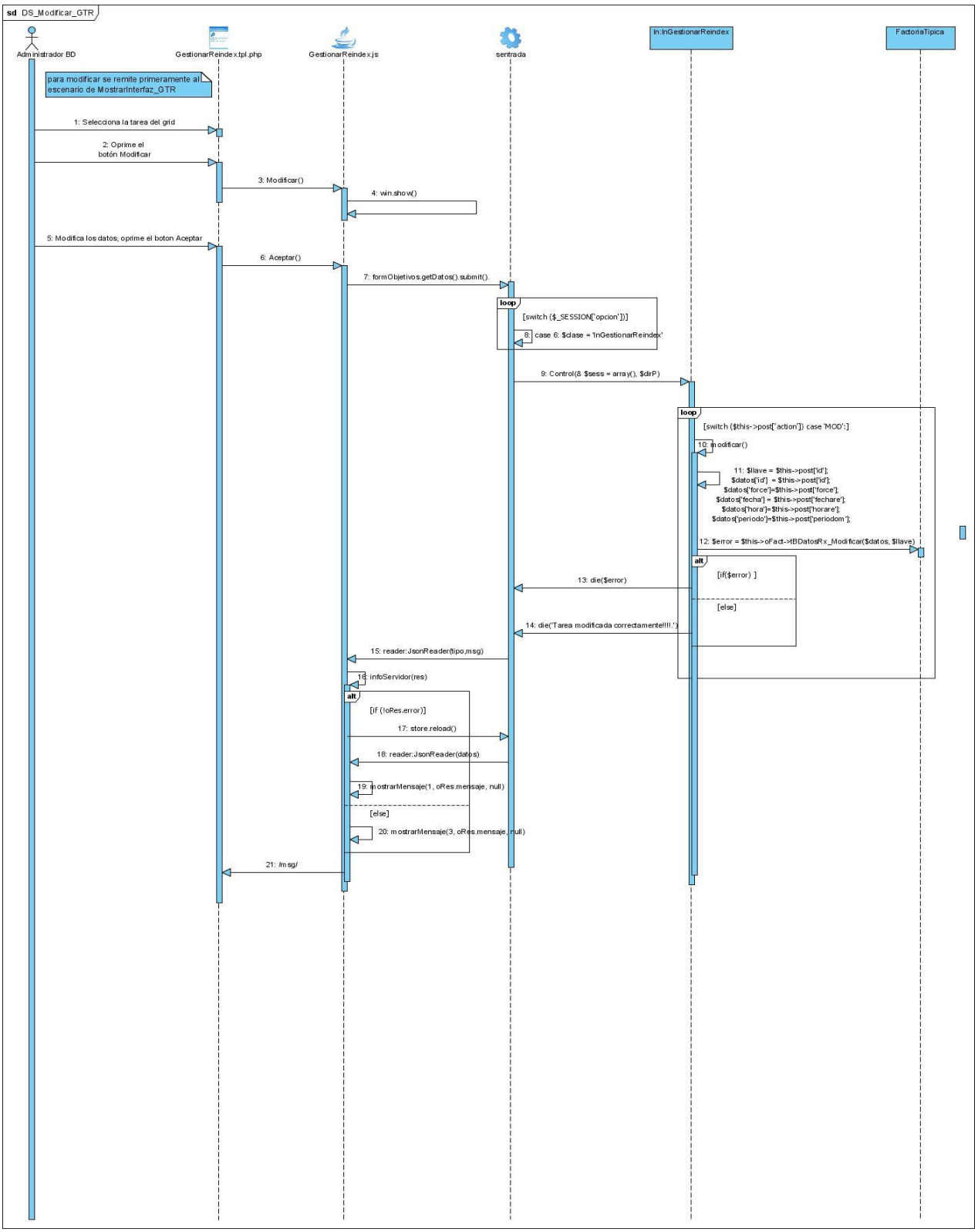


Figura 30. Realización Modificar (Caso de Uso Gestionar Tareas Reindex).

Anexo 2.

Descripción de las tablas de la Base de Datos.

Tabla15. Descripción de la tabla "**regla**" de la base de datos.

Nombre: regla		
Descripción: Contiene todos los datos de una regla determinada.		
Atributo	**Tipo**	**Descripción**
id_regla	integer (0)	Identificador de la regla que se aplica.
tabla	varchar (20)	Tabla a la cual se le aplica la regla.
campo	varchar (20)	Campo de la tabla que se le aplica determinada regla.
nombre	varchar (20)	Nombre que se le pone a la regla que se aplica.
condición	varchar (20)	Condición de una regla que se le aplica a un campo determinado.

Tabla16. Descripción de la tabla "**reindex**" de la base de datos.

Nombre: reindex		
Descripción: Contiene todos los datos de un reindex determinado.		
Atributo	**Tipo**	**Descripción**
id_reindex	smallint (10)	Identificador del reindex a realizar.
fecha	varchar(0)	Fecha en la que se realiza el reindex.
periodo	char (10)	Periodo en el cual se va a realizar el reindex.

hora	time(10)	Hora que se va a realizar la tarea reindex.
force	bit	Estado del atributo (activo o inactivo).

Tabla17. Descripción de la tabla "**salva**" de la base de datos.

Nombre: salva		
Descripción: Contiene todos los datos de una salva determinada.		
Atributo	**Tipo**	**Descripción**
id_salva	smallint (10)	Identificador de la salva a realizar.
nombre	char(10)	Nombre de la salva que se realiza.
backup	char(10)	Atributo que indica a que se le realiza la salva.
fecha	varchar(0)	Fecha en la que se realiza la salva.
hora	time(10)	Hora que se va a realizar la tarea salva.
periodo	char(10)	Periodo en el cual se va a realizar la salva.

Tabla18. Descripción de la tabla **"vacuum"** de la base de datos.

Nombre: vacuum		
Descripción: Contiene todos los datos de un vacuum determinado.		
Atributo	**Tipo**	**Descripción**
id_vacuum	smallint(10)	Identificador del vacuum a realizar.
fecha	varchar(0)	Fecha en la que se realiza el vacuum.

periodo	char(10)	Periodo en el cual se va a realizar el vacuum.
full	bit	Estado del atributo (activo o inactivo).
freeze	bit	Estado del atributo (activo o inactivo).
hora	time	Hora que se va a realizar la tarea vacuum.
analyze	bit	Estado del atributo (activo o inactivo).

Anexo 3.

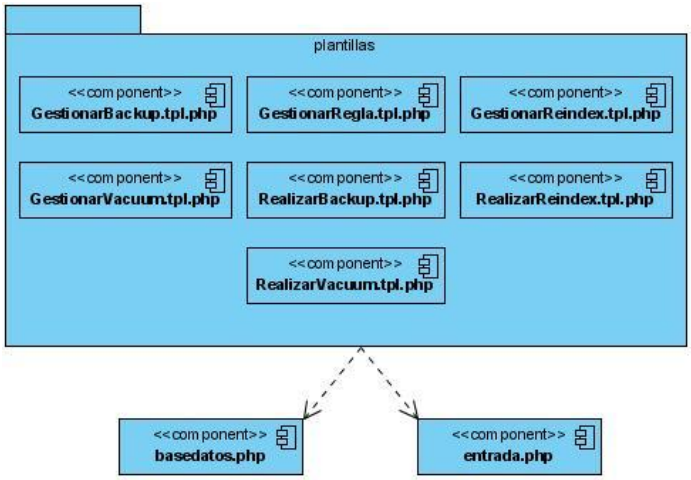


Figura 33. Diagrama de Componentes “plantillas”.

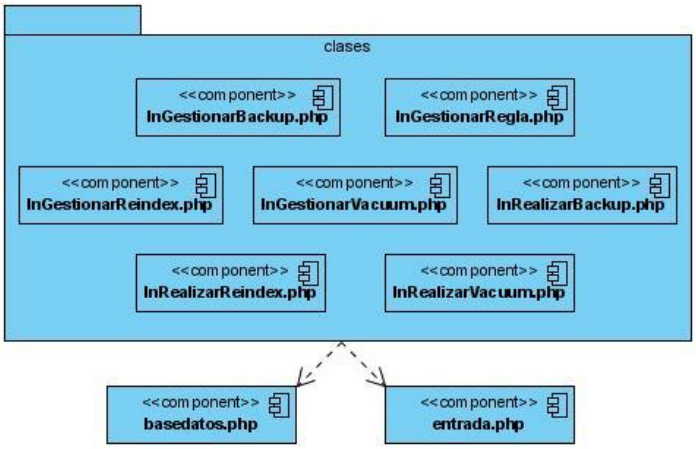


Figura 34. Diagrama de Componentes “clases”.

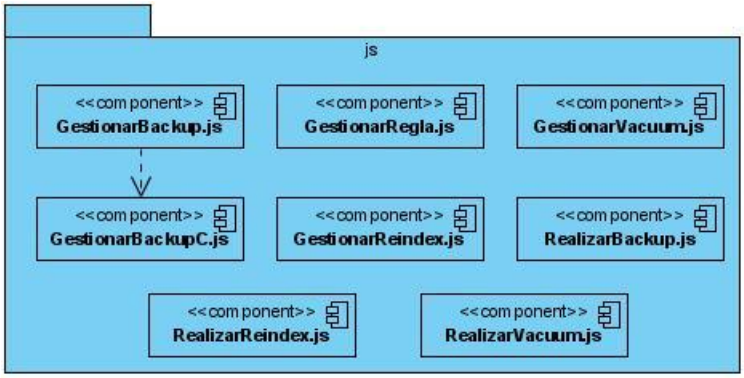


Figura 35. Diagrama de Componentes “js_Admon”.

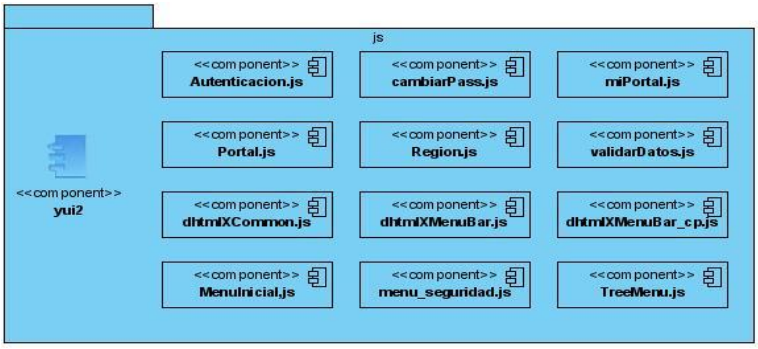


Figura 36. Diagrama de Componentes “js”.

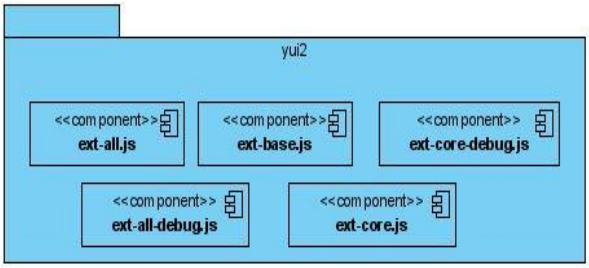


Figura 37. Diagrama de Componentes “yui2”.

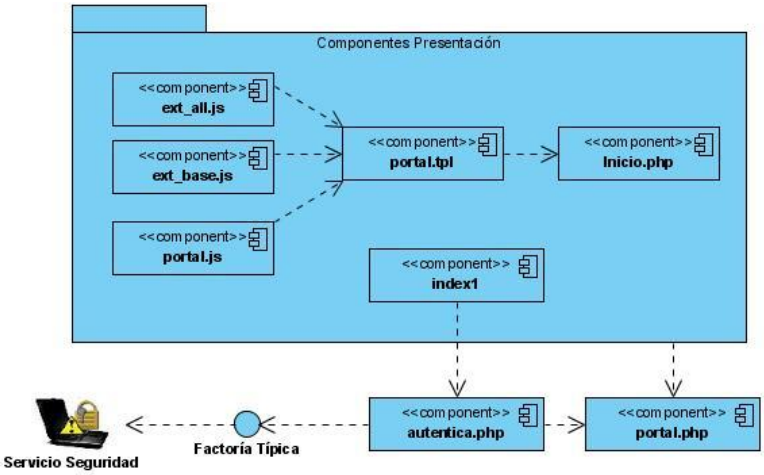


Figura 38. Diagrama de Componentes “Sistema Planificación”.