Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Sistema informático para la administración visual del Servidor Dinámico de Reportes.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Pablo Antonio Hernández Martínez.

Tutores:

Msc. Yanet Villanueva Armenderos.

Ing. Keimer Montes Oliver.

3 de julio del 2014.

Año del 56 Aniversario de la Revolución

"Las personas no son recordadas por el número de veces que fracasan, sino por el número de veces que tienen éxito".

Thomas A. Edison.

DECLARACIÓN DE AUTORÍA.

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Pablo Antonio Hernández Martínez

_____

Firma del Autor

Msc. Yanet Villanueva Armenderos

_____

Firma del Tutor

Keimer Montes Oliver

_____

Firma del Tutor

DATOS DE CONTACTO.

AUTOR:

Pablo Antonio Hernández Martínez

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: pamartinez@estudiantes.uci.cu

TUTORES:

Msc. Yanet Villanueva Armenderos

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: villanueva@uci.cu

Ing. Keimer Montes Oliver

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: kmontes@uci.cu

AGRADECIMIENTOS.

Le agradezco en primer lugar a todas las personas que desde que entre a la universidad han tenido la paciencia de tratar con migo y hacerme entender cuando me equivocaba o cometía algún error, a todos ellos muchas gracias.

Agradecer a todos los integrantes de la tribu, en especial a Orlando, Teylor, Thondique, Carlos y mi hermano Alain, por todos esos días de Lan, por estar siempre hay, por apoyarme y ayudarme, de lo único que me arrepiento sinceramente es de no haberlos conocido antes. A ustedes muchas gracias.

A la gente del dota por tantas noches de osio, chuco, sin dormir hasta el agotamiento de los dedos, en especial a LordReich, Whisper, Deliora y D3xtr.

Agradecer a cuatro personas que se han ganado mi cariño y amistad hasta el punto que los considero como mis hermanos, ellos son peleones, les guastan las aventuras, las fiestas, comer juntos y bastante. Abecés parecen unos viejitos, pero cuando te pelean o te halan de las orejas no es para mal siempre es para guiarte por un buen camino. Adrian, Felix y Sheyla muchas gracias. Dayana a ti no sabría cómo agradecerte, gracias por no rendirte cuando yo lo hice, en vez de eso me diste fuerzas para levantarme, creo que sabes el resto de lo que te voy a decir, simplemente inclúyelo entre líneas. Gracias por estar siempre presente.

También se merecen mi agradecimiento Maira, Maikel, Jorgito, Juana, Yamisela y Jorge, por haberme acogido en su familia con mucho cariño durante el tiempo que pertenecí a ella tratándome como un hijo más.

Agradecer especialmente a mi tutor Keimer, que ahora es mi tutor pero antes fuimos compañeros de estudios, gracias por creer en mí cuando otros pensaban que no lo lograría por darme tu mano y ayuda para salir adelante. Gracias

Por último y no por eso menos importante a toda mi familia gracias, en especial a mis abuelos Toni, Gino y Nancy por ser una guía y ejemplo a seguir, ser además de mis abuelos madre y padre, por ayudarme y estar siempre presente en todos los momentos difíciles e importantes de mi vida, por sacrificarse por mi día a día y por siempre creer en mí, a mi hermana Nancita que desde que llego a este mundo a pesar de su corta edad y de yo no poder haber pasado mucho tiempo con ella (ahora si voy a tener bastante tiempo para verla crecer) me quiere como si estuviera conmigo cada momento y siempre me pregunta cuando voy para mi casa, a mi otra hermana María que desde chiquita siempre me llevo al contraria me molestaba y me hacía incomodar, pero en el fondo verla crecer y hacerse una mujer día a día llena de principios convicciones que parece una mujer vieja, me llena de alegría y orgullo, cada la quiero más y sé que ella siente lo mismo por mí, a mis primos José, Rafe y Anna que son mis otros hermanos, con los cuales compartí mi infancia, siempre metiéndonos en problemas y buscando la manera de arreglarlos antes que se enteraran nuestros padres, a mis primos David, Annalie, Ernesto y Debra, por quererme y apoyarme en todo momento, a mis tíos Jose, Lolín, Zoraya y Zucel, por ayudarme, quererme y darme aliento para seguir adelante y por ultimo a mis padres, esa dos personas por los cuales hoy estoy aquí, gracias por darme la vida, por quererme, por regañarme y castigarme cuando me portaba mal, por preocuparse cundo me enfermo o me va mal los estudios, por ser mis mayores guías y consuelo, en fin gracias por todos los sacrificios que han tenido que hacer por mí, gracias por tomar esa decisión tan difícil que es traer un niño al mundo, aunque nunca se los digo y es mala de mi parte los quiero más de los que son capaces de imaginar. Gracias por ser mis padres.

DEDICATORIA.

Hay dos personas muy especiales, que por mucho que quiera espesar con palabras mi cariño hacia ellos me sería imposible, hoy aunque ya no se encuentren entre nosotros, sé que nunca me han abandonado, ellos siempre han sido y seguirán siendo mi luz, mi ejemplo y camino a seguir, ellos que desde niño me criaron y me dieron siempre su apoyo incondicional. A mis abuelos Pablito y Pipina como cariñosamente le llamábamos, les doy las gracias y les dedico esta tesis.

RESUMEN.

En la actualidad los reportes representan una forma de realizar el análisis de la información almacenada en las bases de datos. Son generados por herramientas informáticas denominadas servidores de reportes, que permiten obtener información de distintas fuentes de datos, dándole un formato para que esta pueda ser interpretada de forma más sencilla. La presente investigación se enmarcó en la implementación de un sistema para la administración visual para el Servidor Dinámico de Reportes (SDR) desarrollado en el departamento de Integración de Soluciones del Centro de Tecnología y Gestión de Datos (DATEC). Con este propósito se realizó un estudio acerca de las herramientas de administración de diferentes servidores de reportes, permitiendo determinar funciones semejantes como: el envío de correos de forma automática, la gestión de usuarios y permisos, así como la gestión de reportes y conexiones. Finalmente se obtuvo la herramienta “SDRAdmin” la cual permite administrar visualmente las funcionalidades del SDR. Esta facilita la gestión de usuarios, reportes y conexiones con las fuentes de datos, la realización de consultas al registro de reportes exportados, así como la programación de tareas automáticas para el envío de correo y la subida de archivos a ftp.

Palabras claves

Herramienta de administración, servidor de reportes, reportes.

ABSTRACT.

Currently reports are a way to make the analysis of data stored in databases. They are generated by software tools known servers reports, which can obtain information from various data sources, giving a format so that it can be interpreted more easily. The present investigation was framed in the implementation of a system for the visual administration of Dynamic Server Reports (SDR) developed at the Integrated Solutions Department of the Data Management and Technology Center (DATEC). For this purpose a study of the administration tools of different servers reports was performed, allowing to determine similar functions as: the sending of emails automatically, user management and permissions, as well as management reports and connections. Finally the "SDRAdmin" tool was obtained, which allows you to visually manage the functionality of the SDR. This facilitates user management, reports, links to data sources, querying the exported log reports and schedule automatic email sending and uploading files to ftp tasks.

Keywords

Administration Tool, Report Server, reports.

ÍNDICE.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

INTRODUCCIÓN.

Los reportes representan una de las formas utilizadas para obtener informes detallados sobre un tema específico, provenido a quienes lo utilizan de una potente herramienta que agiliza la toma de decisiones durante el proceso de análisis de la información. En el ámbito de la informática, los reportes organizan y exhiben la información de diferentes fuentes de datos, entre las que se destacan las bases de datos y almacenes de datos; aplicando a esta un formato para mostrarla por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios. De esta forma le confieren mayor utilidad a los datos, por ser más sencillo representando los datos mediante un gráfico de barras, que con una hoja de cálculos con cientos de campos.

Los reportes son generados a través de aplicaciones informáticas denominadas servidores de reportes. Estas herramientas cuentan con un conjunto de bibliotecas que permiten organizar y gestionar la información para generar los reportes. Los servidores poseen en su mayoría una interfaz gráfica de usuario de interfaz gráfica de usuario, con la que interactúa el usuario para realizar operaciones de mantenimiento, gestionar los reportes y publicarlos.

Dentro de los centros que se especializan en el manejo de información contenida en las bases de datos y la elaboración de reportes en la Universidad de las Ciencias Informáticas (UCI), se encuentra el Centro de Tecnología y Gestión de Datos (DATEC). El mismo se divide en varios departamentos dentro de los cuales se encuentra el departamento de Integración de Soluciones, en donde se desarrollan productos informáticos para el trabajo con reportes como el Servidor Dinámico de Reportes (SDR).

Este SDR presenta una Arquitectura Orientada a Servicios (SOA por las siglas en inglés Service Oriented Architecture) implementando servicios de tipo Transferencia de Estado Representacional (REST por las siglas en inglés de Representational State Transferens). Tiene como clientes a otras aplicaciones, como el Generador Dinámico de Reportes (GDR). Su funcionamiento se basa en servicios web consumidos por las aplicaciones clientes. Brinda servicios para publicar reportes diseñados y exportarlos en distintos formatos. Permite la gestión de reportes de manera rápida y ligera, además posibilita el envío de informes por correo y subidas de archivos a ftp.

A pesar de las facilidades que brindan, las funcionalidades administrativas para la gestión y el mantenimiento del servidor como: la gestión de usuarios, reportes y conexiones con las fuentes de datos, la realización de consultas al registro de reportes exportados, así como la programación de tareas automáticas para el envío de correo y la subida de archivos a ftp, se realiza mediante la consola, lo cual puede ser un proceso engorroso.

Por lo expuesto anteriormente, el **problema de a resolver** queda definido en la siguiente interrogante: ¿Cómo mejorar la administración de los recursos gestionados del SDR?

La presente investigación tiene como **objeto de estudio**: Los procesos de administración de reportes y sus dependencias, enmarcado en el **campo de acción**: Los procesos de administración del SDR.

El **objetivo general** de esta investigación es: Desarrollar un sistema informático para la administración de los recursos de SDR mediante una interfaz gráfica de usuario.

Para dar solución al **objetivo general** se definen los siguientes **Objetivos específicos**:

1. Construir el marco teórico sobre los mecanismos visuales que mejoran la utilización de servidores, así como las tecnologías y herramientas empleadas en la investigación.

2. Realizar el análisis y diseño de la herramienta informática para la administración visual de los recursos del Servidor Dinámico de Reportes.

3. Realizar la implementación y prueba de la herramienta informática para la administración visual de los recursos del Servidor Dinámico de Reportes.

En función de dar cumplimiento a los objetivos planteados se definen las siguientes tareas de investigación:

- Revisión bibliográfica de los mecanismos de administración de servidores, así como de las herramientas y tecnologías necesarias en el desarrollo de la investigación.
- Caracterización de los mecanismos de integración con sistemas basados en arquitectura orientada a servicios, específicamente con los que utilizan servicios de Transferencia de Estado Representacional (REST).
- Realización del análisis de la solución.

- Realización del diseño de la herramienta informática a partir de los artefactos obtenidos en la fase de análisis.
- Implementación de las clases diseñadas.
- Confección de los casos de pruebas para comprobar el funcionamiento de la herramienta informática implementada.
- Realización de las pruebas de calidad a partir de los casos de prueba diseñados para la herramienta informática desarrollada.
- Integración de la herramienta informática implementada con el Servidor Dinámico de Reportes.

Métodos científicos

- **Torbellino de ideas:** Mediante este métodos se desarrollan interrogantes acerca de lo que se quiere o se podría llegar a investigar. Este método se utilizará para obtener las necesidades del cliente y así poder realizar el levantamiento de los requisitos funcionales con los que debe cumplir la Herramienta de Administración.
- **Histórico-Lógico:** Mediante este método se realiza un análisis histórico de la evolución y trayectoria de los elementos a estudiar con el objetivo de describir sus principales características. Este método se utilizará para constatar teóricamente los elementos históricos fundamentales de las herramientas de administración de los motores de reportes.
- **Modelación:** Mediante este método se realiza una abstracción con vistas a explicar la realidad. Este método se utilizará para la modelación de los diagramas correspondientes a la fase de análisis y diseño.

El presente trabajo se ha estructurado de la siguiente manera: Introducción, Capítulo I, Capítulo II, Capítulo III, Conclusiones, Recomendaciones, Trabajos citados, Bibliografía y Anexos. Con el objetivo de lograr una mejor comprensión se presenta una pequeña descripción de los capítulos.

En el **Primer Capítulo: "Fundamento teórico del Sistema informático para la administración visual del Servidor Dinámico de Reportes"**, se presentan los principales concepto a investigar asociados al dominio del problema, se selecciona la metodología para el desarrollo de la solución, así como las herramientas y tecnologías a utilizar.

En el **Segundo Capítulo: "Análisis y diseño del Sistema informático para la administración visual del Servidor Dinámico de Reportes"**, se definen los requisitos que la herramienta debe cumplir, se realizan los diagramas de clases e interacción para los casos de uso arquitectónicamente significativos. Se elabora el Diagrama de Despliegue para dar una visión de cómo debe quedar la solución, además de exponer la arquitectura y los principales patrones de diseño utilizados.

En el **Tercer Capítulo: "Implementación y prueba del Sistema informático para la administración visual del Servidor Dinámico de Reportes"**, se realizan las actividades de implementación y prueba de la solución propuesta. Se realiza el Diagrama de Componentes para los casos de uso arquitectónicamente significativos. Se aplican niveles, métodos y técnicas de prueba relacionados con la implementación, con el fin de comprobar el funcionamiento del sistema y el cumplimiento con los requisitos de software definidos.

CAPÍTULO 1: FUDAMENTACIÓN TEÓRICA DEL SISTEMA INFORMÁTICO PARA LA ADMINISTRACIÓN VISUAL DEL SERVIDOR DINÁMICO DE REPORTES.

INTRODUCCIÓN.

En el presente capítulo se realiza un estudio del arte de las herramientas existentes para el manejo de datos. Además de las diferentes tecnologías, lenguajes y metodologías que servirán de apoyo para dar cumplimientos a los objetivos, las tareas propuestas y de esta forma dar solución al problema planteado.

1.1 Sistema de administración de servidores

Un sistema de administrador de un servidor es un conjunto de elementos interrelacionados de hardware y software que permite el acceso a los recursos de un servidor. De manera general hacen uso de una interfaz gráfica de usuario (GUI por sus siglas en inglés Graphical User Interface), mediante la que se puede organizar, gestionar las características y servicios de un servidor. Los sistemas de administración tienen gran importancia por encargarse de la comunicación visual de un usuario con el servidor, lo que posibilita un mejor control de sus funciones.

1.1.1. JasperReports

JasperReports es un motor de creación de informes en código abierto. Está escrito completamente en Java y es capaz de utilizar los datos procedentes de cualquier tipo de fuente de dato. Permite exportar en una variedad de formatos de documentos incluyendo HTML, PDF, Excel, OpenOffice y Word. Este presenta las siguientes características y funciones: (1)

- La conectividad con los sistemas de gestión de identidades existentes para centralizar y asegurar informes y vistas de análisis.
- Acceso de seguridad granular hasta el nivel celular y en la columna.
- Servidor de informes y análisis de gestión centralizada.
- Integración con los sistemas de gestión de identidades existentes, tales como Protocolo Ligero de Acceso a Directorios para inicio de sesión único a los informes.
- Control de acceso basado en roles a todos los objetos del repositorio.

- Simplificación y mejora del rendimiento en la generación de informes y la entrega a los usuarios.
- Entrega automática de los informes envarios formatos de salida.
- Arquitectura de datos flexible que apoya los informes basados en fuentes de datos relacionales y no relacionales.
- Sub-informes reutilizables y almacenados en el repositorio. (1)

1.1.2. Crystal Reports

Crystal Reports es un producto para la creación e integración de reportes con datos provenientes de múltiples fuentes de datos. Utiliza una tecnología de elaboración de reportes permitiendo el diseño y generación de informes a partir de datos almacenados en una base de datos u otra fuente de información. Gira alrededor del soporte para un tipo de ficheros de formato propietario, que se distingue por la extensión .RPT (report) en el que se almacena la definición de los informes. Entre las principales funciones de Crystal Reports se encuentran: (2)

- Crear reportes.
- Administrar y diseñar los reportes.
- Dar una vista previa de los reportes.
- Exportar los reportes.
- Controlar la fuente de datos.
- Crear una nueva conexión.

Crystal Reports se puede ver como la combinación de tres componentes principales, que son:

- **Motor de Impresión:** Este componente no solo se encarga de lo relacionado con la impresión en papel de los informes, sino de todo lo que tiene que ver con la ejecución de los mismos.
- **Librerías de Código Manejado:** Encapsulan la funcionalidad del Motor de Impresión a través de un conjunto de clases fácilmente accesibles desde aplicaciones escritas en Visual Basic, C# o cualquier otro lenguaje.

- **Diseñador de Informes:** Es el software que presenta la interfaz a través de la cual un usuario, programador o no, puede crear (“diseñar”) un informe y guardarlo en un fichero .RPT para su posterior reutilización. (2)

Luego de realizar un estudio de las herramientas existentes se definió que a realizar debería cumplir con algunas de las características y funciones:

- Permitir una vista previa de los reportes.
- Exportar los reportes en diferentes formatos.
- Crear una nueva conexión con los sistemas gestores de basas de datos.
- Controlar el acceso basándose en roles.
- Entregar automáticamente los informes mediante correo electrónico.
- Crear reportes.

A estas funcionalidades se le agregarán requisitos como:

- Eliminar y probar las conexiones con los sistemas gestores de bases de datos.
- Gestionar los usuarios del sistema y sus roles.
- Visualizar los registros de las acciones sobre los reportes.
- Eliminar, modificar y listar los reportes.

Al realizar el estudio de las herramientas existentes para la administración de servidores se llega a la conclusión, de que el sistema de administración a implementar debe tener como característica principal, la autenticación de los usuarios para tener acceso a las funcionalidades que este brinda. El sistema debe de contar con roles que definan la acciones que se le permite a cada usuario en el sistema y como función principal debe permitir la gestión de usuarios y reportes.

1.2 Servidor Dinámico de Reportes

El Servidor Dinámico de Reportes (SDR) como su nombre indica, es un servidor para gestionar, publicar y exportar reportes de forma dinámica y en múltiples formatos. Está basado en una Arquitectura Orientada a Servicios (SOA por sus siglas en inglés Service Oriented Architecture) implementando servicios de tipo Transferencia de Estado Representacional (REST por sus siglas en inglés de Representational State

Transferens). Este servidor fue desarrollado en el lenguaje de programación Java y utiliza las librerías libre de JaspertReport en su versión 5.0. Se crea principalmente con el objetivo de sustituir antiguo motor de reportes del Generador Dinámico de Reportes (GDR), concebido de tal manera que pueda ser utilizado por cualquier aplicación que necesite un motor de reportes. Además, las aplicaciones existentes para realizar este tipo de gestión son privativas, como es el caso de CristalReport y el JaspertServer.

1.3 Servicios Web

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma. Se pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. Para el uso de esta tecnología generalmente se usan arquitecturas orientadas al consumo de servicios como SOA. (3)

1.3.1. Arquitectura Orientada a Servicios (SOA)

SOA es un modelo de arquitectura que caracteriza el procedimiento para crear y usar los diversos procesos, reunidos en forma de servicios. SOA descompone la funcionalidad deseada en diferentes nodos conectados a través de una red, combinados entre sí para alcanzar el resultado deseado. Estos servicios pueden proporcionar datos o llevar a cabo actividades de coordinación entre uno o varios servicios (3).
Dentro de las ventajas de SOA se encuentran:

- Brinda la posibilidad de cambiar componentes individuales sin afectar a otros componentes.
- Todos los sistemas se conectan al bus de la misma forma, con lo que se gana en homogeneidad. Arquitectura sencilla, robusta y escalable.

1.3.2. REST

REST es un protocolo de comunicación, que define un conjunto de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al

estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. Se rige por cuatro principios fundamentales:

- Utiliza los métodos HTTP de manera explícita.
- No mantiene estado.
- Expone URIs con forma de directorios.
- Transfiere XML, JavaScript Object Notation (JSON), o ambos.

Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP. Por ejemplo, HTTP GET se define como un método productor de datos, cuyo uso está pensado para que las aplicaciones cliente obtengan recursos, busquen datos de un servidor web, o ejecuten una consulta esperando que el servidor web la realice y devuelva un conjunto de recursos.

Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP.

Las notaciones utilizadas para definir el tipo de método que responderá en una llamada de un servicio REST son:

Tabla 1. Métodos definidos para REST.

Anotación	Descripción
@PATH(ruta)	Establece la ruta base + la ruta definida. La ruta base está basada en la ruta de acceso a la aplicación, el servlet y el patrón URL definido en el descriptor de despliegue web.xml.
@POST	Indica que el método atenderá a peticiones HTTP POST
@GET	Indica que el método atenderá a peticiones HTTP GET
@PUT	Indica que el método atenderá a peticiones HTTP PUT
@DELETE	Indica que el método atenderá a peticiones HTTP DELETE
@Produces(tipo [, más-tipos])	@Produces define qué tipo MIME se generará en la llamada al método con la anotación @GET.
@Consumes(tipo [, más-tipos])	@Consumes define qué tipo MIME consumirá el método.
@PathParam	Se usa para inyectar parámetros desde la URL a los parámetros del método.

En la primera línea de esta imagen se define la URL por la cual se accederá a través del navegador, luego se crea la clase, especificando en su constructor el parámetro que recibirá el servicio por la URL, seguido a al constructor se implementan los tipos de peticiones que el servicio brindará, en este caso se especifican las peticiones GET, PUT y DELETE.

1.4. Metodología de Desarrollo

Las metodologías de desarrollo de software constituyen un conjunto de procedimientos y técnicas para ayudar a los desarrolladores con la documentación durante el proceso de desarrollo. Estas describen paso a paso todas las actividades a realizar para lograr el producto informático deseado y definen las personas que participan en el desarrollo de las actividades así como su papel en las mismas. Además detallan la información que se debe producir como resultado de una actividad y la necesaria para comenzarla (5). Las metodologías de desarrollo se encuentran divididas en dos grupos: ágiles y robustas. Dentro de las ágiles se encuentran: Programación Extrema (XP por sus siglas en inglés Xtreme Programation)**,** Open UP, Cristal, entre otras. Entre las pesadas podemos mencionar: Proceso Unificado de Racional (RUP por sus siglas en inglés Rational Unified Process) y Modelos en Espiral (4).

1.4.1. Open UP

Open UP es un proceso unificado que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado. Contiene un conjunto mínimo de prácticas que ayuda al equipo de trabajo a ser efectivo desarrollando software. Tiene una filosofía pragmática de desarrollo, que se enfoca en la naturaleza colaborativa del desarrollo de software. Se trata de un proceso con herramientas neutrales, que puede extenderse para alcanzar una amplia variedad de proyectos. Mantiene las características esenciales de RUP, incluyéndose el empleo de casos de uso, escenarios, manejo de riesgos y diseño basado en la arquitectura (4).

Se decide utilizar la metodología OpenUP debido a que brinda un enfoque ágil y ligero, apropiada para el desarrollo de proyectos pequeños y de poca envergadura. Permite detectar errores tempranos a través de su ciclo de desarrollo iterativo. Nos brinda un conjunto de artefactos los cuales guían el desarrollo del proyecto, al mismo tiempo que permiten documentarlo.

1.5. Herramienta y lenguaje de Modelado.

Las herramientas de Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés Computer Aided Software Engineering) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software y facilitar al usuario el modelado de procesos. Dentro de estas herramientas se encuentran Rational Rose, Visual Paradigm for UML, Easy CASE y CASE Studio. Entre las más se usadas se encuentran: Rational Rose, muy conocida para los clientes de Windows y Visual Paradigm for UML (5).

1.5.1. Lenguaje Unificado de Modelado.

Lenguaje Unificado de Modelado (UML por sus siglas en inglés Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, así como aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (7). Este es utilizado en herramientas CASE para modelar los diferentes diagramas o artefactos necesarios para la realización de un proyecto (6).

1.5.2. Visual Paradigm for UML.

Visual Paradigm for UML 8 es una herramienta de diseño que soporta los diagramas UML, proporciona el uso de amplias funciones de modelado, incluyendo el modelado completo de Diagramas de Casos de Uso y la generación de Diagramas de Actividad. Visual Paradigm for UML8 genera documentación del sistema en formato PDF, HTML y MS Word. La cual apoya gran parte del ciclo de desarrollo del software. (6)
Entre sus principales características se encuentran: modelado de procesos del negocio, administración de requerimientos, generación de la capa objeto-relacional, generación de código e ingeniería inversa. También posee una interfaz de usuario amigable y es multiplataforma, disponible para los Sistemas Operativos Linux, Windows, y Mac OS (6).

Se decide utilizar Visual Paradigm for UML con el objetivo de realizar el modelado de la herramienta. Luego se llevará a cabo la implementación de la aplicación modelada utilizando un conjunto de herramientas y tecnologías que se explican a continuación para facilitar el desarrollo de la misma.

1.6. Herramientas y lenguajes de programación.

Para el desarrollo de la aplicación se hace necesario la utilización de herramientas y lenguajes de programación que faciliten el diseño e implementación. Las herramientas de programación constituyen una unidad donde se combinan compiladores de código, permitiendo realizar programas, rutinas y sistemas que al ser ejecutados en el ordenador realizan las acciones indicadas mediante el código produciendo algún resultado. Entre ellas se encuentran la plataforma de programación NetBeans, el lenguaje de programación JavaScript, el servidor Apache Tomcat y el gestor de bases de datos PGAdmin que se definen a continuación (7).

1.6.1. Entorno de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés de Integrated Development Environment) es un entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, compilador, depurador y constructor de interfaz gráfica de usuario. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes (7).

1.6.2. NetBeans IDE.

El NetBeans IDE es una herramienta para programadores escrita en Java pero puede utilizarse para otros lenguajes de programación. Tiene soporte para crear interfaces gráficas de forma visual, crear aplicaciones para móviles y desarrollar aplicaciones web. Dentro de las ventajas que este ofrece se encuentra que: puede ser usada para desarrollo de aplicaciones, posibilita la reutilización de módulos, brinda una instalación y actualización simple, posee soporte para el lenguaje interpretado PHP. Además es multi-idioma, de licencia gratuita y multiplataforma (7).
Cuenta con un editor de código sensible al contenido con soporte para auto completamiento de código, auto tabulación y uso de abreviaturas para varios lenguajes de programación.

Se selecciona el NetBeans IDE en su versión 7.3 para desarrollar la solución propuesta por las caracteristicas mencionadas anteriormente. Ademas incluye extensiones que pueden ser útiles durante el proceso de desarrollo tales como: editores visuales de interfaces, completamiento de código para diferentes lenguajes y compiladores para algunos de estos. Además brinda un conjunto de librerías útiles que facilitan el desarrollo de aplicaciones informáticas.

1.6.3. Servidores web.

Un servidor web es un sistema cuyos componentes, hardware y software, están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes para el logro de un objetivo. La comunicación con este se establece a través de un protocolo prefijado por un esquema de cliente-servidor. Existen varios servidores web como: Apache Tomcat, Microsoft II y Sun Java System Web.

1.6.4. Apache Tomcat.

Apache Tomcat también conocido como simplemente Tomcat, es un servidor web multiplataforma que funciona como contenedor de servlets (Pequeño programa que corre en un servidor. Por lo general son aplicaciones Java que corren en un entorno de servidor web). Se desarrolla en el proyecto denominado Jackarta bajo la licencia Apache 2.0 que implementa las especificaciones de los servlets de Java Server Pages o JSP de Sun Microsystem. Cuenta con un conjunto de características que han ido evolucionando de acuerdo a las versiones del servidor como son: está implementado con Servlet 3.0 JSP 2.2., detecta y previene "fugas de memoria" en las aplicaciones web, brinda limpieza interna de código y soporte para la inclusión de contenidos externos directamente en una aplicación web. Tiene entre sus ventajas la modularidad; además es de código abierto, multiplataforma y extensible (8).
Se escoge como servidor web para la aplicación a desarrollar el Apache Tomcat por las características anteriormente expuestas, teniendo en cuenta que posee una licencia gratuita, además de ser el servidor donde está montado el SDR.

1.6.5. Sistema Gestor de Bases de datos.

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas que permiten a los usuarios crear y mantener una base de datos. Facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones (9). Entre sus funciones se encuentran consultar, actualizar el diseño y generar bases de datos.

1.6.6. Lenguaje de programación JavaScript.

JavaScript es un lenguaje de programación interpretado, basado en el estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipiado y dinámico. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web que permite mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe la forma de JavaScript del lado del servidor. Este lenguaje se diseñó con una sintaxis similar al lenguaje C, aunque adopta nombres y convenciones del lenguaje de programación Java. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML (10).

1.6.7. Marco de trabajo de desarrollo.

Se define como marco de trabajo un conjunto estandarizados de conceptos, prácticas y criterios que se estanblecen para enfocar un tipo de problemática en particular y que sirve como referencia, para enfrentar y resolver problemas de características similares. En el área de la informática, define conjunto de bibliotecas orientadas a la reutilización a gran escala de componentes software para el desarrollo rápido de aplicaciones. Dentro de las ventajas que nos brindan la utilización de marcos de trabajos son:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un marco de trabajo constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes software.
- El uso y la programación de componentes que siguen una política de diseño uniforme (10).

- **JQuery.**

Es un marco de trabajo de JavaScript el cual cuenta con un conjunto de funciones implementadas para el desarrollo de aplicaciones web. Este nos permite agregar efectos y funcionalidades complejas como galerías de fotos dinámicas y elegantes, validación de formularios y calendarios. También nos brinda la posibilidad de utilizar la tecnología como JavaScript asíncrono y XML (AJAX por sus siglas en inglés *Asynchronous JavaScript And XML)* y Modelo de Objetos del Documento (DOM por sus siglas en inglés *Document Object Model*) si tener en cuenta los detalles complejos de la programación. Por lo anteriormente expuesto es el marco de trabajo seleccionado para desarrollar el sistema para la administración del SDR ya que el mismo debe de ser ligero y amigable para el usuario (11).

1.7. Patrones.

Un patrón es un fragmento nombrado de información instructiva, que captura la estructura esencial y la visión interna de una familia de soluciones con probado éxito sobre un problema recurrente que surge dentro de un cierto contexto. En otras palabras, es reutilizar soluciones que funcionaron bien una vez. En programación orientada a objetos se entiende por patrón una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparecen repetidamente en el desarrollo de sistemas software (12). Existen varios tipos de patrones entre los que se destacan los de arquitectura y diseño.

1.7.1. Patrones Arquitectónicos,

Los patrones arquitectónicos representan un esquema que organiza la estructura de un sistema de software. Estos proveen un conjunto de subsistemas definidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre los sistemas. Los patrones son conocidos también como plantillas concretas para la arquitectura de software que especifican la estructura de una aplicación y que tiene impacto en la arquitectura de los subsistemas (13).

- **Arquitectura por capas.**

Dentro de estos patrones uno de los más conocidos es de arquitectura por capas, este constituye uno de los que aparecen con mayor frecuencia mencionados como categorías mayores de catálogo. La arquitectura por capas se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. En la arquitectura por capas, los conectores se determinan mediante los protocolos que definen en qué forma interactuan las capas (14).

- **Ventajas de la arquitectura en capas.**

  - ✓ Permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
  - ✓ Proporciona una amplia reutilización de los componentes, brindando la posibilidad de definir interfaces de capas estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.
  - ✓ El estilo admite optimizaciones.

- **Desventajas o restricciones del estilo.**

  - ✓ El estilo exige que cada capa opera solo con las capas adyacentes.
  - ✓ A veces se hace difícil encontrar un nivel de abstracción correcto.

1.7.2. Patrones de diseño.

Según “Buschmann” un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software o las relaciones entre ellos. Describen la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (15).

Dentro de los patrones de diseño podemos encontrar los Grupo de los Cuatro (GoF por sus siglas en inglés Gans of Four) y los Patrones de Asignación de Responsabilidades (GRASP por sus siglas en inglés General Responsibility Assignment Software Patterns).

- **Patrones GoF.**

Los patrones Gof fueron sugeridos en 1995 por la banda de los cuatro Erich Gamma, Richard Helm, Ralph Jonson y John Vissidess. Estos promueven un avance en la programación orientada a objetos ya que ayudan a clasificarla según su propósito en: patrones de creación que, como indica su nombre, crean instancias; los estructurales que definen la relación entre las clases dando lugar a la combinación y formación de estructuras mayores y los de comportamiento que describen la interacción y cooperación entre las clases (16).

- ✓ **Patrones creacionales**: Engloban aquellos patrones de software que se centran en la forma de crear las clases y sus instancias, así como el uso que recibirán una vez creadas.
- ✓ **Abstract Factory:** Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- ✓ **Builder:** Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- ✓ **Factory Method:** Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan a qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- ✓ **Prototype:** Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- ✓ **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.
- ✓ **Patrones estructurales:** Engloban aquellos patrones de software que se centran en la composición y estructura de las clases y en la herencia entre ellas.
- ✓ **Adapter:** Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- ✓ **Bridge:** Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

- ✓ **Composite:** Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
- ✓ **Decorator:** Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- ✓ **Facade:** Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
- ✓ **Flyweight:** Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.
- ✓ **Proxy:** Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.
- ✓ **Patrones de comportamiento:** Engloban aquellos patrones de software que se centran en la comunicación entre las distintas clases y objetos.
- ✓ **Chain of Responsibility:** Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- ✓ **Command:** Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.
- ✓ **Interpreter:** Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
- ✓ **Iterator:** Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
- ✓ **Mediator:** Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

- ✓ **Memento:** Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.
- ✓ **Observer:** Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- ✓ **State:** Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Hace parecer que cambia la clase del objeto.
- ✓ **Strategy:** Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
- ✓ **TemplateMethod:** Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
- ✓ **Visitor:** Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera (16).

- ➢ **Patrones GRASP.**

Los patrones GRASP son parejas de problema y solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (16).

Se pueden destacar cinco patrones principales dentro de los patrones GRASP:

- ✓ **Experto:** Asigna la responsabilidad al experto en información (clase que tiene toda la información necesaria para implementar una responsabilidad).
- ✓ **Creador:** Asigna a una clase (B) la responsabilidad de crear una instancia de otra (A) si se cumple alguno de los puntos siguientes:
  1. B contiene a A.

2. B agrega a A.
3. B tiene los datos de inicialización de A.
4. B registra a A.
5. B utiliza estrechamente a A.

- ✓ **Alta cohesión:** Mantiene manejable la complejidad y asigna responsabilidades de manera que la cohesión permanezca alta.

- ✓ **Bajo acoplamiento:** Brinda soporte a las bajas dependencias y al incremento de la reutilización, además asigna responsabilidades de manera que el acoplamiento (innecesario) se mantenga bajo.

- ✓ **Controlador:** Gestiona un evento del sistema, asigna la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas opciones:
  1. Representa el sistema global, dispositivo o un subsistema (controlador de fachada).
  2. Representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o sesión).

- ✓ **Polimorfismo:** Es el responsable de que el comportamiento varíe en función del tipo de clase. Cuando las alternativas o comportamientos relacionados varían, se asigna la responsabilidad del comportamiento utilizando operaciones polimórficas a los tipos para los que varía el comportamiento.

- ✓ **Fabricación Pura:** Asigna un conjunto altamente cohesivo de responsabilidades a una clase de comportamiento artificial o de conveniencia que no representa un concepto del dominio del problema, para dar soporte a la alta cohesión, bajo acoplamiento y la reutilización.

- ✓ **Indirección:** Asigna responsabilidades para evitar el acoplamiento directo.
  Asigna la responsabilidad a un objeto intermedio para mediar entre otros componentes o servicios, de manera que no se acoplan directamente.

- ✓ **Variaciones Protegidas:** Asigna responsabilidades a los objetos, subsistemas, y sistemas de manera que las variaciones o inestabilidad en estos elementos no influya de manera no deseable en otros elementos.
  1. Identifica los puntos de variaciones predecibles o inestabilidad.
  2. Asigna las responsabilidades para crear una interfaz estable alrededor de ellos (16).

CONCLUSIONES PARCIALES.

La investigación realizada sobre los conceptos principales para el análisis de la situación antes expuesta, fue el punto de partida para lograr una mejor comprensión de la aplicación llegando a las siguientes conclusiones:

Los servidores de reportes que cuentan con un sistema de administración visual permiten al usuario una mejor familiarización con la herramienta.

Se seleccionaron como herramientas, metodologías y lenguajes para el desarrollo de la herramienta:

1. El entorno de desarrollo NetBeans IDE en su versión 7.4.
2. Como lenguajes de programación JavaScript del lado del cliente y java del lado del servidor.
3. Se seleccionó como marco de trabajo JQuery en su versión 1.9.1.
4. Como herramienta de modelado Visual Paradigm for UML en su versión 8.0.
5. Como lenguaje de modelado UML en su versión 2.0.
6. Se seleccionó como metodología de desarrollo Open UP, teniendo en cuenta las características del sistema a desarrollar.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA INFORMÁTICO PARA LA ADMINISTRACIÓN VISUAL DEL SERVIDOR DINÁMICO DE REPORTES.

INTRODUCCIÓN

En el presente capítulo se realizará un análisis de las características que debe tener el sistema de administración, identificando los requisitos funcionales y no funcionales. Serán definidos y detallados los casos de uso, identificando actores y su relación con éstos. Se analizarán los patrones de diseño y estilos arquitectónicos. Además del diseño de los diagramas de clases que describen la implementación de la herramienta.

2.1. Modelo del dominio del SDRAdmin

El modelo de dominio es un diagrama de clases donde no solo se capturan los conceptos propios de un sistema sino la realidad física del mismo. Cuando se realiza la programación su funcionamiento interno se aproxima de alguna manera a la realidad. De cierto modo el modelo de dominio constituye una primera versión del sistema (17).

El siguiente modelo de dominio refleja como el SDRAdmin, que consume los servicios brindados por el SDR, le permite al usuario utilizar el sistema para la gestión de usuarios, reportes y la programación tareas automáticas; así como la visualización de las trazas de la gestión de los reportes.

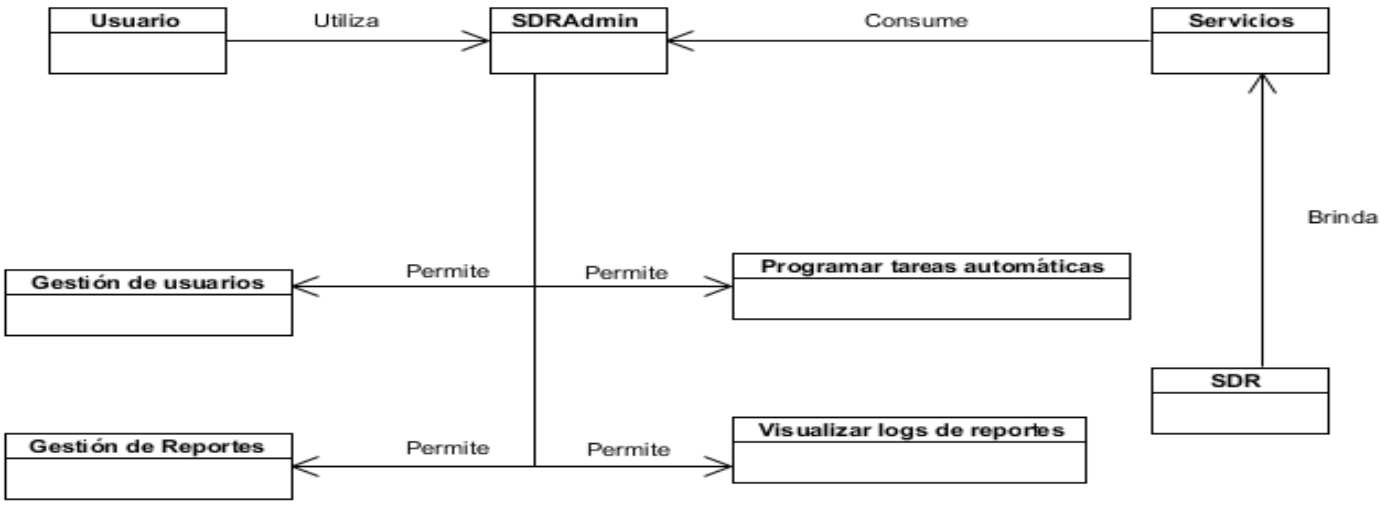

Fig. 1 Modelo del dominio del sistema.

2.1.1. Definición de las clases del Modelo del Dominio del Sistema.

Tabla 2 Descripción de las clases del modelo del dominio.

Clases	Descripción
Usuario	Persona que interactúa con el sistema de administración que puede ejecutar las funciones que brinda el sistema,
SDRAdmin	Herramienta del SDR, que permite la administración grafica del servidor.
Servicios	Prestaciones que brinda el SDR.
SDR	Servidor de reportes el cual brinda los servicios para la gestión de reportes y la programación de tares automáticas de envío de correo.
Gestión de Usuario	Se especializa y contiene todos los métodos con el cual el usuario controlar toda la gestión de información que se realizan sobre la entidad usuarios.
Gestión de Reportes	Se especializa y contiene todos los métodos con el cual el usuario controlar toda la gestión de información que se realizan sobre la entidad reportes.
Visualizar logs de reportes	Clase que muestra el registro de las acciones sobre los reportes en el sistema.
Programar tareas automáticas.	Se especializa y contiene todos los métodos con el cual el usuario controlar toda la gestión de información necesaria para el envío de tareas automáticas.

2.2. Especificación de los requisitos

Los requisitos son un grupo de condiciones que necesita un usuario para solucionar un problema y lograr su objetivo. Estas condiciones tienen que ser alcanzadas por un sistema o componente del mismo para satisfacer un contrato, estándar u otro documento impuesto formalmente.

2.2.1. Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir y que establecen los comportamientos del sistema (1).

Requisitos funcionales identificados.

RF1 Adicionar usuario.

- **Descripción:** El sistema a través de una interfaz gráfica de usuario debe de permitir entrar los datos necesarios para crear un nuevo usuario y luego procesarlos.
- **Entrada:** Datos del nuevo usuario (nombre, usuario, correo y contraseña).
- **Salida:** El nuevo usuario queda adicionado al sistema y a partir de ese momento tiene acceso al sistema según el rol que este tenga en el mismo.

RF2 Modificar usuario.

- **Descripción:** El sistema a través de una interfaz debe de permitir luego de listar los usuarios del sistema, buscar los datos específicos de uno de estos y modificarlo.
- **Entrada:** Datos a modificar del usuario seleccionado (nombre, usuario, correo y contraseña).
- **Salida:** No procede.

RF3 Listar usuario.

- **Descripción:** El sistema a través de una interfaz debe mostrar un listado de los usuarios existentes en el mismo.
- **Entrada:** No procede.
- **Salida:** Muestra el listado de los usuarios del sistema.

RF4 Eliminar usuario.

- **Descripción:** El sistema a través de una interfaz debe permitir luego de listar los usuarios del sistema, seleccionar uno y eliminarlo.
- **Entrada:** Identificador del usuario seleccionado.
- **Salida:** Muestra el listado de los usuarios del sistema actualizado.

RF5 Visualizar trazas de reportes.

- **Descripción:** El sistema a través de una interfaz debe permitir listar todas las acciones realizadas por los usuarios.
- **Entrada:** No procede.
- **Salida:** Muestra el listado de los registros de seguridad.

RF6 Programación de tareas automáticas.

- **Descripción:** El sistema a través de una interfaz debe permitir al usuario entrar los datos de tarea de envío de correo para luego realizarla en la fecha y hora prevista.
- **Entrada:** Fecha, frecuencia, destino, usuario, contraseña y adjunto.
- **Salida:** Envió de correos con la frecuencia y datos especificados al destinatario.

RF7 Adicionar reporte.

- **Descripción:** El sistema a través de una interfaz debe permitir entrar los datos necesarios para crear un nuevo reporte, luego procesarlos.
- **Entrada:** Datos del nuevo reporte (descripción y archivo del reporte en formato .jrxml).
- **Salida:** El nuevo reporte queda registrado al sistema.

RF8 Modificar reporte.

- **Descripción:** El sistema a través de una interfaz debe de permitir luego de listar los usuarios del sistema, buscar los datos específicos de uno de los reportes, modificarlo, actualizar la base de datos y la lista de reportes.
- **Entrada:** Datos a modificar del reporte seleccionado (descripción y archivo del reporte en formato jrxml).
- **Salida:** El reporte queda modificado.

RF9 Listar reporte.

- **Descripción:** El sistema a través de una interfaz debe mostrar un listado de los reportes existentes en el mismo.
- **Entrada:** No procede.
- **Salida:** Muestra el listado de reportes.

RF10 Eliminar reporte.

- **Descripción:** El sistema a través de una interfaz debe permitir luego de listar los reportes del sistema, seleccionar uno y eliminarlo.
- **Entrada:** Identificador del reporte seleccionado.
- **Salida:** Muestra el listado de los reportes del sistema actualizado.

RF11 Exportar reporte.

- **Descripción:** El sistema a través de una interfaz debe permitir luego de listar los reportes, seleccionar uno y exportarlo.
- **Entrada:** Id del reporte seleccionado.

- **Salida:** Crea un archivo en el formato especificado.

RF12 Adicionar conexión.

- **Descripción:** El sistema a través de una interfaz debe de permitir entrar los datos necesarios para crear una nueva conexión y luego procesar los datos.
- **Entrada:** Datos de la nueva conexión.
- **Salida:** La nueva conexión queda adicionada al sistema.

RF13 Modificar conexión.

- **Descripción:** El sistema a través de una interfaz debe de permitir luego de listar las conexiones del sistema, buscar los datos específicos de una conexión y modificarla.
- **Entrada:** Datos a modificar de la conexión.
- **Salida:** La conexión queda modificada en la base de datos del sistema.

RF14 Listar conexión.

- **Descripción:** El sistema a través de una interfaz debe de mostrar un listado de las conexiones existentes en el mismo.
- **Entrada:** No procede.
- **Salida:** Muestra el listado de las conexiones.

RF15 Eliminar conexión.

- **Descripción:** El sistema a través de una interfaz debe permitir luego de listar las conexiones del sistema, seleccionar uno y eliminarlo.
- **Entrada:** Id de la conexión seleccionado.
- **Salida:** Muestra el listado de las conexiones del sistema actualizado.

RF16 Autenticar.

- **Descripción:** El sistema a través de una interfaz debe permitir al usuario autenticarse en el sistema.
- **Entrada:** Datos de usuario (usuario y contraseña).
- **Salida:** El usuario entra al sistema.

2.2.2. Requisitos no funcionales

Normalmente los requisitos no funcionales están vinculados a requisitos funcionales, estos representas las necesidades de la computadora donde será utilizado el sistema, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse y qué cualidades debe tener.

Requisitos de Software:

- **RNF1.** Se debe tener instalado el servidor web Apache Tomcat.
- **RNF2.** Tener instalado la máquina virtual de java jdk.

Requisitos de Hardware:

- **RNF3.** Mínima capacidad para la Memoria de Acceso Aleatorio (RAM por sus siglas en inglés Random Access Memory) debe ser de 512 MB.
- **RNF4.** Capacidad de espacio libre en el disco duro debe ser 500 MB, este es el tamaño mínimo que la aplicación necesita.
- **RNF5.** El patrón arquitectónico que se debe emplear para el desarrollo de la aplicación es de 3 capas.

Requisitos de disponibilidad y usabilidad.

- **RNF6.** La interfaz debe permitir ser utilizadas por personas que posean conocimientos básicos de informática.

- **RNF7.** La interfaz debe ser lo más representativa posible y se podrá acceder a ella siempre que se tenga acceso al SDR.

Requisitos de Soporte:

- **RNF8.** El sistema estará acompañado de un documento de ayuda que guiará a los usuarios teniendo en cuenta cada funcionalidad.

Requisitos de Portabilidad:

- **RNF9.** La herramienta, una vez desarrollada podrá ser usada siempre y cuando se tenga acceso al SDR.

2.3. Modelo de casos de uso.

Representa las relaciones que existen entre el actor y los casos de uso. El actor es un individuo, entidad o máquina, con lo que el sistema interactúa y los casos de uso son porciones de funcionalidades que el sistema ofrece para aportarles un resultado de valor a los actores que interactúen con él (18).

2.3.1. Actores del sistema.

Para el sistema los actores son humanos que interactúan con la aplicación y pueden intercambiar información con él. Los actores poseen un rol que es determinado por el conjunto de casos de uso al que tiene acceso, en este caso se identificaron dos actores principales: usuario y administrador. Este último es una especialización del primero, ya que comparte sus casos de uso y tiene otros que son específicos del mismo rol.

Tabla 3 Descripción de los actores del sistema.

Nombre del actor	Descripción
Usuario	Es el que interactúa con el Sistema de Administración y realiza las operaciones en el mismo (tiene restricciones de permisos).
GgAdministrador	Especialización de usuario que no tiene restricción de permisos en el sistema.

2.3.2. Diagrama de casos de uso del sistema.

El diagrama de casos de uso (DCU) representa la forma de como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

El DUC está compuesto por varios casos de uso. Un caso de uso es una unidad de trabajo significativa, en él se pueden agrupar varios requisitos funcionales de la aplicación. Representan una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema.

Para la aplicación se agruparon dieciséis requisitos funcionales en siete casos de usos: Gestionar usuarios, Gestionar reportes, Autenticar, Programar tareas automáticas, Exportar reportes y Listar logs de reportes. Además se identificaron como actores: Usuario y Administrador.

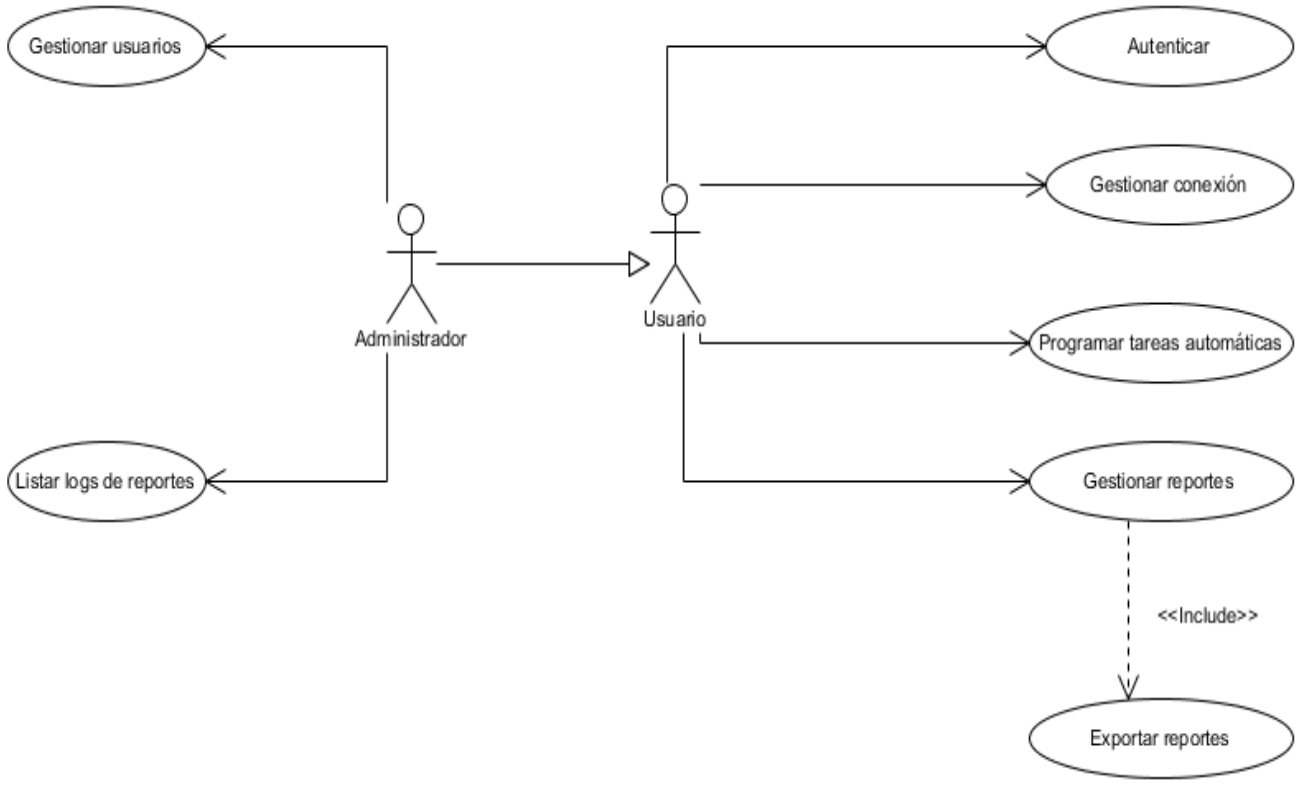

Fig. 2 Diagrama de casos de uso.

2.3.3. Descripción textual del caso de uso arquitectónicamente significativo.

Tabla 4 Descripción del caso de uso Gestionar reportes.

Caso de Uso:	Gestionar reportes.
Actores:	Usuario, Administrador.
Resumen:	El caso de uso inicia una vez que el usuario introduce el usuario y la contraseña y se autentica en el sistema, luego selecciona la pestaña Reportes, dodne se muestra un listodo con todos los reportes en el sistema y un conjunto de botenes que permiten la gestion de reportes.
Precondiciones:	Que el usuario esté autenticado en el sistema.
Referencias	RF6, RF7, RF8, RF9.
Prioridad	Alta.
Flujo Normal de Eventos	
Sección 1 "Adicionar reportes"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario da click en la pestaña "Adicionar reportes".	2. El sistema muestra la interfaz "Adicionar reportes" donde se le solicitan los datos del reporte.
3. El usuario introduce los datos y selecciona la opción: "Aceptar" o "Cancelar".	4. Si el usuario selecciona la opción "Aceptar" el sistema valida los datos, si estos no tienen errores el sistema envía la petición al servidor de insertar usuario, enviando los datos del nuevo usuario. Si los datos presenta errores ver flujo alterno 4.1.
	5. Si no fue seleccionada la opción Adicionar reportes ver flujo alterno.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**

	4.1. Si los datos introducidos por el usuario presenta errores el sistema muestra un mensaje y solicita al usuario insertar los datos nuevamente.
	5.1. Si fue seleccionada la opción Cancelar el sistema reinicia los campos de texto.
Poscondiciones	El reporte queda adicionado a la base de datos.
Sección 2 "Modificar reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario da click en la pestaña "Listado de reportes".	2. El sistema muestra la interfaz "Listado de reportes" donde muestra el listado de los reportes existentes en el sistema.
3. El usuario da click en el botón modificar reporte.	4. El sistema muestra una ventana donde se solicita el id del reporte a modificar.
5. El usuario introduce el id y da click en la opción mostrar.	6. El sistema busca el reporte, si este existe muestra los datos del mismo, sino ver flujo alterno.
7. El usuario modifica los datos que desea y acepta o cierra la ventana.	8. Si el usuario acepta, el sistema valida los datos, si estos no presentan errores modifica el reporte seleccionado, si los datos presentan errores ver flujo alterno.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	6.1. El sistema muestra un mensaje de error.
	8.1. El sistema muestra un mensaje de error y regresa los datos a su estado original.
Poscondiciones	El reporte queda modificado y actualizado en la base de datos.
Sección 3 "Listar reportes"	
Acción del Actor	**Respuesta del Sistema**

1. El usuario da cilck en la pestaña, “Listado de reportes”.	2. El sistema muestra la interfaz “Listar reportes” donde se listan los reportes almacenados en el sistema.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	Se muestra el listado de reportes en el navegador.
Sección 4“Eliminar reportes”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario da click en la pestaña, “Listado de reportes”.	2. El sistema muestra la interfaz “Listado de reportes” donde se muestra el listado de los reportes existentes en el sistema.
3. El usuario da click en el botón eliminar reporte.	4. El sistema muestra una ventana donde se solicita el id del reporte a eliminar.
5. El usuario introduce el id y acepta o cierra la ventana.	6. El sistema busca y elimina el reporte de la base de datos.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	El reporte queda eliminado de la base de datos.

2.4. Modelo del diseño

El modelo del diseño es una simplificación que permite comprender mejor el sistema que se desea implementar. Anticipa y guía el proceso de desarrollo, además de definir una solución del software que satisfaga los requisitos identificados en el análisis (4).

2.4.1. Diagrama de clases del diseño

El diagrama de clases del diseño es una descripción gráfica de las clases e interfaces de una aplicación. Este contiene las definiciones de las entidades del software en vez de conceptos del mundo real. Para la

elaboración del modelo del diseño fueron modelados cinco diagramas en correspondencia a los casos de uso del sistema. Todos los diagramas y sus descripciones se encuentran en el artefacto Modelo del diseño.

El estilo arquitectónico seleccionado y utilizado en el diagrama de diseño del caso de uso arquitectónicamente significativo es el modelo en capas.

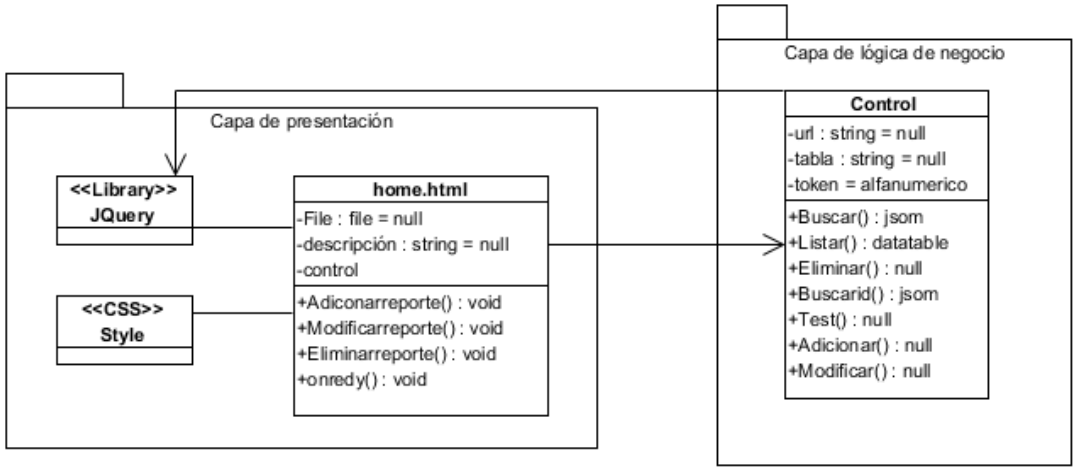

Fig. 3 Diagrama clases del diseño del caso de uso Gestionar reportes.

2.4.2. Descripción de las clases del diagrama de diseño

Tabla 5 Descripción de las clases del diseño.

No	Clase	Tipo de clase	Descripción	Relación	Tipo de Relación
1	home	Interfaz	Clase interfaz encargada de mostrar al usuario los resultados de su acción en el sistema.	2,3,4	Asociación
2	control	Controladora	Clase que almacena todos los métodos de comunicación y captura de los datos del servidor.	1,3,4	Asociación

4	jquery	Marco de trabajo.	Contiene conjunto de métodos utilizados en las clases controladoras e interfaces.	1,2	Asociación
5	css	css	Define y organiza los elementos de las interfaces.	1	Asociación

2.4.3. Diagrama de secuencia

El diagrama de secuencia es un medio gráfico para representar la interacción de los objetos en un instante de tiempo. Muestran a un usuario o actor del sistema, así como los componentes u objetos con el que este interactúa durante la ejecución de un caso de uso. Tiene como punto de partida el cumplimiento de las precondiciones de los contratos de una operación. Estos representan una vía para documentar los escenarios de casos de uso, capturar los objetos necesarios de manera temprana en el análisis y verificar el uso de los objetos en el diseño (18).

En el siguiente diagrama se muestra como el usuario, selecciona en la interfaz la opción mostrar reporte, esta al recibir la petición lo re-direcciona a la interfaz mostrar reportes, que envía una llamada a la clase controladora para buscar los reportes del servidor, la clase controladora envía la petición al servidor el cual retorna los datos, en caso de que estos contengan algún error se muestra un mensaje, si los datos no presentan algún tipo de error se muestra en la interfaz el listado de reportes existentes.

2.5. Patrones utilizados.

2.5.1. Patrones de Casos de Uso

Los patrones de Caso de Uso son comportamientos que deben existir en el sistema, describen el uso y cómo este interactúa con los usuarios.

Durante la fase de análisis y diseño realizada en el desarrollo de la aplicación se utilizó el patrón CRUD completo para los casos de uso Gestionar reportes, Gestionar usuario y Gestionar conexiones, el de inclusión para Gestionar reportes que incluye, Exportar reportes. Por ultimo se tiene el patrón múltiples actores rol común, que se evidencia entre el Usuario y el Administrador.

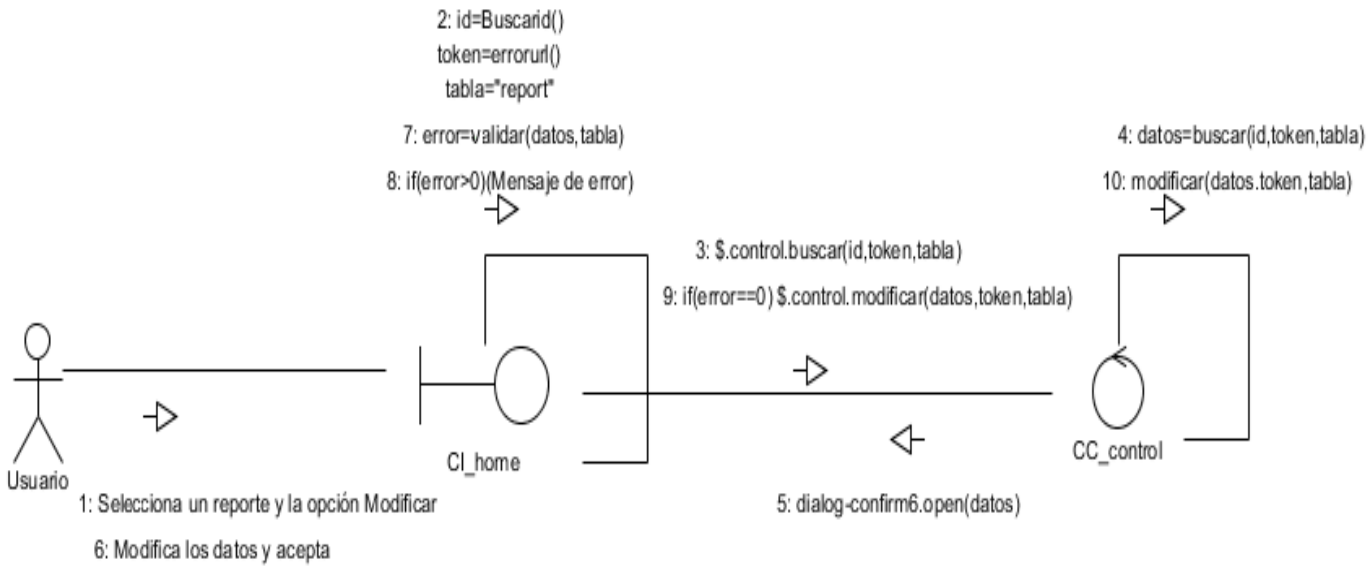

Fig. 4 Diagrama de secuencia del caso de uso Gestionar reportes sección modificar.

2.5.2. Patrón arquitectónico por capas.

Como se describe en el primer capítulo, el patrón arquitectónico definido para la implementación de la aplicación es el patrón por capas, que permite agrupar las clases y entidades pertenecientes al sistema según las funciones y relaciones existentes entre ellas. Permite además la ubicación de las capas en los niveles correspondientes.

En la siguiente imagen se muestra dos niveles fundamentales; nivel de aplicación donde se encuentran la capa de presentación, compuesta por las interfaces de la aplicación que se comunican directamente con las clases que pertenecen a la capa de lógica del negocio constituida por las clases encargadas del manejo de la información. En el nivel de datos se encuentra la capa de datos que en este caso está constituida por el servidor el cual proporciona todos los datos que se manejan el sistema.

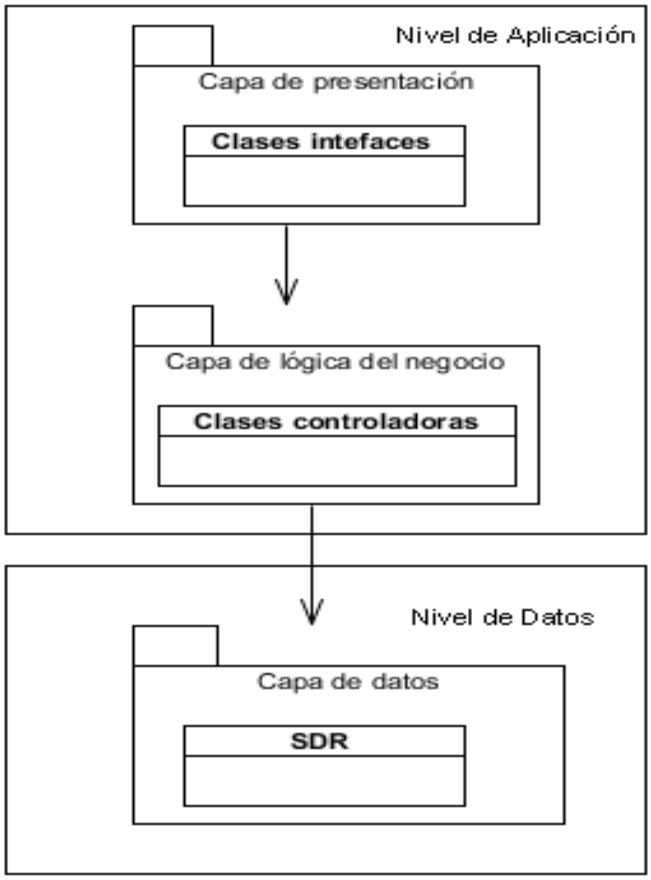

Fig. 5 Arquitectura de la aplicación

2.5.3. Patrones de diseño GoF.

Los patrones GoF se dividen en tres categorías de acuerdo a su propósito dentro las clases: creacionales, estructurales y de comportamiento.

Los patrones de creación: abstraen en el proceso de creación de instancias y como se manipulan estas dentro de las clases.

Los patrones estructurales: estos se ocupan de como las clases están relacionadas entre sí para formar una estructura y proporcionar nuevas funcionalidades.

Los patrones de comportamiento: se relacionan con los algoritmos de asignación de responsabilidades entre los objetos de sistema.

Dentro de estos tres tipos de patrones se aplica al sistema dentro del grupo de creacionales, el patrón de instancia única. Este se evidencia a la hora de crear una instancia única de la clase control en la clase de la interfaz, que es utilizada todos los métodos de la clase que se necesita como se evidencia en la siguiente imagen:

```
var  controlx= $.control;

$("#dialog-confirm1").dialog({
    resizable: false,
    autoOpen: false,
    with : 150,
    modal: true,
    buttons: {
        Aceptar: function() {
            var nombre = document.getElementById("name").value;
            var correo = document.getElementById("correo").value;
            var user = document.getElementById("User").value;
            var pass = document.getElementById("Password").value;
            var rol = Rol();
            var data = JSON.stringify({"email": correo,
                "name": nombre,
                "password": pass,
                "username": user,
                "subNet": "10.0.0.0/8",
                "role": rol});
            $("#name").val("");
            $("#correo").val("");
            $("#User").val("");
            $("#Password").val("");

            $(this).dialog("close");

            controlx.adcionar("user", errorurl(), data);
        },
```

Fig. 6 Evidencia del patrón instancia única

Se encuentra además dentro del grupo de comportamiento, de mediador, que es utilizado en las clases controladoras que se encargan de ser intermediarias entre las clases interfaces y el servidor de las cuales consumen los servicios evidenciándose dicho patrón en la siguiente imagen:

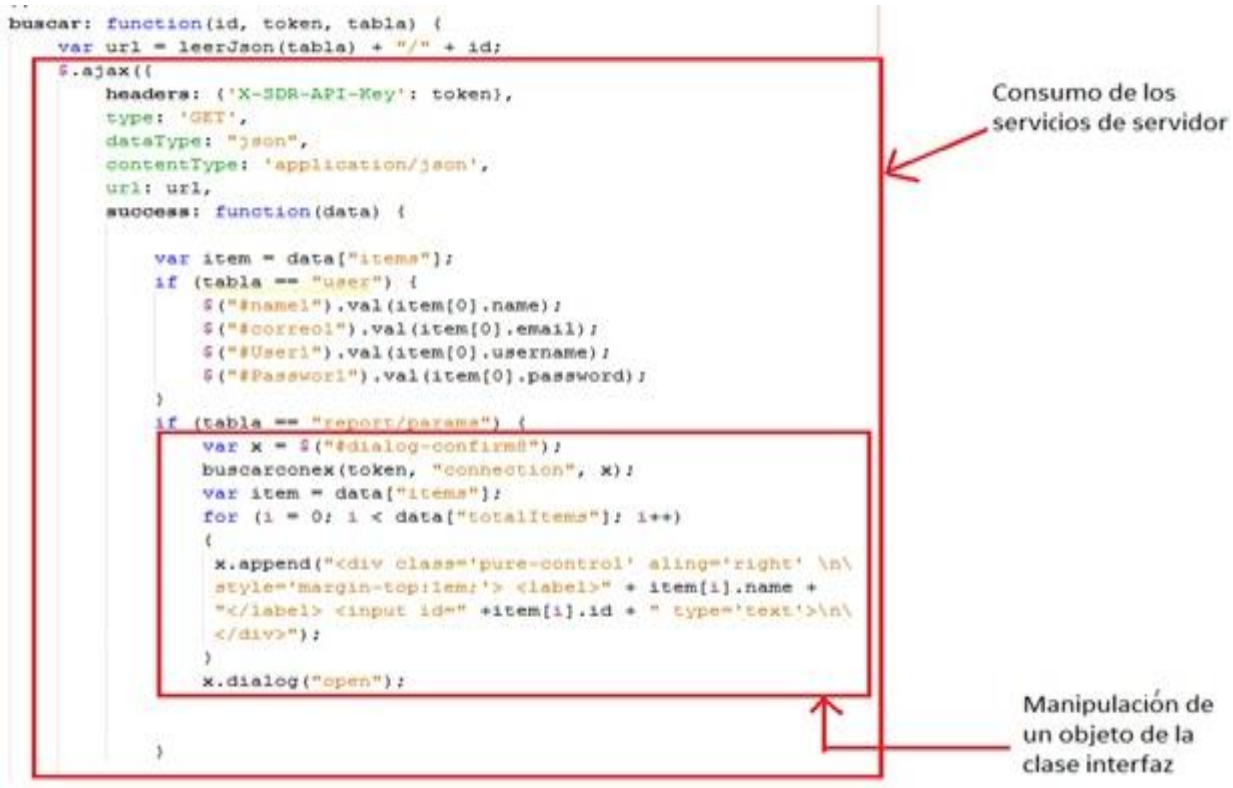

Fig. 7 Evidencia del patrón mediator

2.5.4. Patrones de diseño GRASP

Dentro de los patrones de diseños utilizados en la aplicación se encuentran:

Creador: Se evidencia al asignar las responsabilidades a las clases de crear los objetos. En la solución propuesta la clase interfaz control crea solo los objetos necesarios pertenecientes a la clase home con el objetivo de manipularlos como se evidencia en la siguiente imagen.

```
buscar: function(id, token, tabla) {
    var url = leerJson(tabla) + "/" + id;
    $.ajax({
        headers: {'X-SDR-API-Key': token},
        type: 'GET',
        dataType: "json",
        contentType: 'application/json',
        url: url,
        success: function(data) {

            var item = data["items"];
            if (tabla == "user") {
                $("#name1").val(item[0].name);
                $("#correo1").val(item[0].email);
                $("#User1").val(item[0].username);
                $("#Passwor1").val(item[0].password);
            }
            if (tabla == "report/params") {
                var x = $("#dialog-confirm8");
                buscarconex(token, "connection", x);
                var item = data["items"];
```

Fig. 8 Evidencia del patrón creador

Bajo acoplamiento: En la solución se observa el uso de este patrón en la clase index, perteneciente a las vistas, pues esta es una clase independiente que reduce el impacto de los cambios de otros componentes.

Alta Cohesión: Se evidencia el uso de este patrón en la clase home.html la cual como muestra la figura 7, posee la estructura necesaria para almacenar la información de forma coherente y de acuerdo a la responsabilidad que tenga asignada cada clase.

2.6. Diagrama de despliegue

Un diagrama de despliegue constituye una representación física de cómo están relacionados los elementos en el sistema. A través de este se muestra la configuración de los elementos de procesamiento y los componentes de software en tiempo de ejecución. En el siguiente diagrama se evidencia como se conecta una computadora cliente con el servidor y el tipo de conexión por la cual la accede (19).

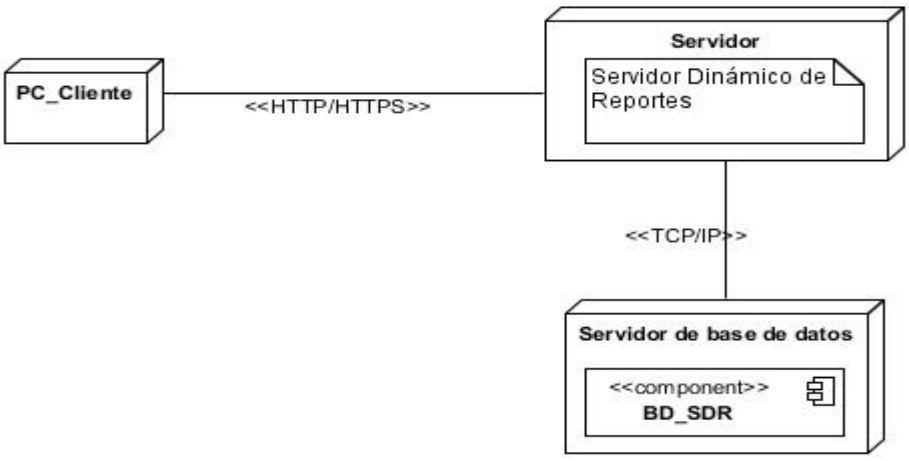

Fig. 9 Diagrama de despliegue de la solución

CONCLUSIONES PARCIALES

En el presente capítulo fueron identificados dieciséis requisitos funcionales que necesita la aplicación para cumplir con las necesidades del cliente, estos se agruparon en siete casos de uso de acuerdo a los patrones definidos. Además se realizó el modelo de dominio el cual facilita la comprensión física dentro del contexto del sistema para la administración visual del SDR. Se detallaron los patrones arquitectónicos y de diseño que serán utilizados en el desarrollo de la aplicación destacándose entre los arquitectónicos el patrón de arquitectura por capas, que permitió separar los conceptos mejorando el desarrollo estructurado del sistema.

CAPITULO 3: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA INFORMÁTICO PARA LA ADMINISTRACIÓN VISUAL DEL SERVIDOR DINÁMICO DE REPORTES.

INTRODUCCIÓN

En el presente capítulo se presentan los diagramas de componentes de los casos de uso arquitectónicamente significativos, que conforman el modelo de implantación correspondiente a la herramienta. Se definirán los métodos de pruebas que se aplican en el proceso de pruebas para validar el funcionamiento de la herramienta de administración del SDR, estos métodos estarán compuestos por niveles, técnicas y casos de pruebas.

3.1 Modelo de implementación.

En el modelo de implementación se describe en términos de código como se implementan los elementos del modelo de diseño. También se describe la ubicación de los componentes de acuerdo a la estructura y modularidad disponible en la implementación y los lenguajes empleados en la implementación. Fundamentalmente, se muestra la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes (6).

3.1.1. Diagrama de componentes.

Un componente es una parte física de un sistema, con un nivel más alto de abstracción que un diagrama de clases, usualmente un componente esta compuesto por una o más clases (u objetos) en tiempo de ejecución. Este se agrupa formando la estructura general del sistema con todos los elementos del mismo.
El diagrama de componentes que se muestra en la Fig. 10, pertenece al caso de uso arquitectónicamente significativo Gestionar reportes.

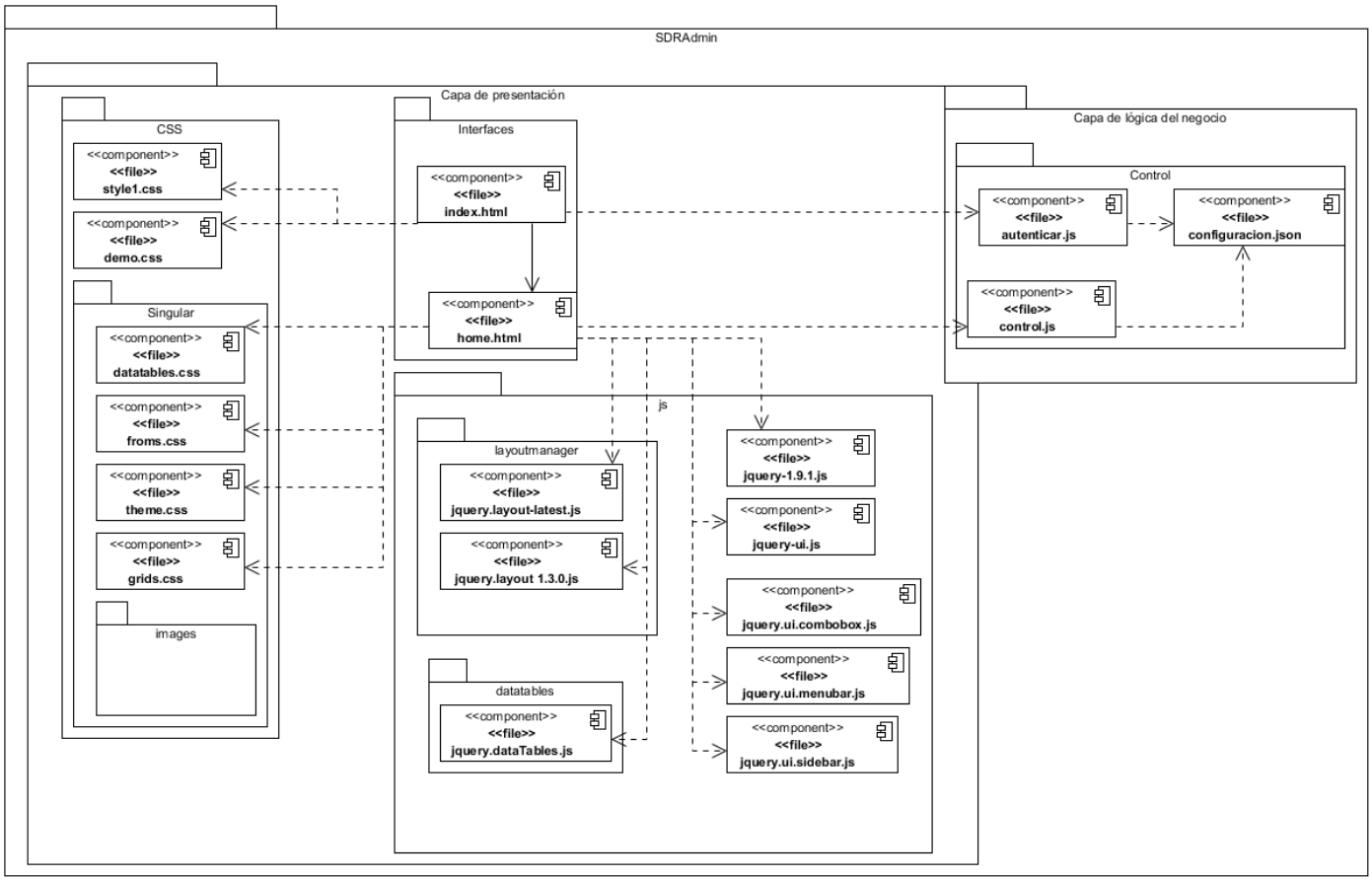

Fig. 10 Diagrama de componentes del Sistema para la administración del SDR.

3.2 Estilos de codificación

Los estilos de codificación constituyen una serie de convenios para escribir el código fuente en los lenguajes de programación. Estos estilos ayudan al mantenimiento de la aplicación, sirven como punto de referencia para los programadores y ayudan a mejorar el proceso de codificación haciéndolo mucho más eficiente.

A continuación se presentan algunos estilos seleccionados para el desarrollo de la herramienta:

Tamaño y organización de las líneas de código:

Las líneas de código no tendrán más de 150 caracteres. En caso de que una misma línea ocupe más espacio esta se podrá dividir después de una coma y la nueva línea debe de estar alineada con la anterior.

Declaraciones.

Las variables locales se inicializaran en el momento de su creación salvo que su valor inicial dependa de un valor calculado posteriormente, estas deben situarse al principio de cada bloque principal en el que se utilicen y no en el momento de su uso.

Los métodos se separarán entre sí por una línea en blanco. Se pondrán las declaraciones al principio de los bloques y no se esperará el primer uso para declararlos.

Convenciones de nombres para las variables.

Los nombres de las variables comenzarán con minúscula, en caso de ser compuestas se escribirán juntos.

Convenciones de nombres para los métodos.

Los métodos comenzarán con letra inicial minúscula, en caso de ser compuesto se escribirán juntos.

Convenciones de nombres para las clases.

Se empleará un sustantivo para el nombre de las clases y comenzara con letra inicial minúscula, en caso de ser compuestos se escribirán juntos y cada palabra comenzará con letra inicial minúscula.

Espacios en blanco.

Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.

- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
  - ✓ Entre métodos.
  - ✓ Antes de un comentario de bloque o de un comentario de una línea.
  - ✓ Entre las distintas secciones lógicas de un método para facilitar la lectura.

Buenas prácticas de programación

- **Asignación sobre variables:** Se deben evitar las asignaciones de un mismo valor sobre múltiples variables en una misma sentencia, porque dichas sentencias suelen ser difíciles de leer. No se deben utilizar asignaciones embebidas o anidadas.
- **Paréntesis:** Se deben utilizar paréntesis en expresiones que incluyan distintos tipos de operadores para evitar problemas de precedencia de operadores.

3.2.1 Interfaces Principales de la Extensión

Para el diseño del sistema se tuvo en cuenta un diseño amigable y sencillo para mejorar la comprensión de los usuarios. A continuación se muestran imágenes algunas interfaces de la herramienta como se muestran en la Fig. 11 y 12.

Fig. 11 Interfaz de autenticación del sistema de administración.

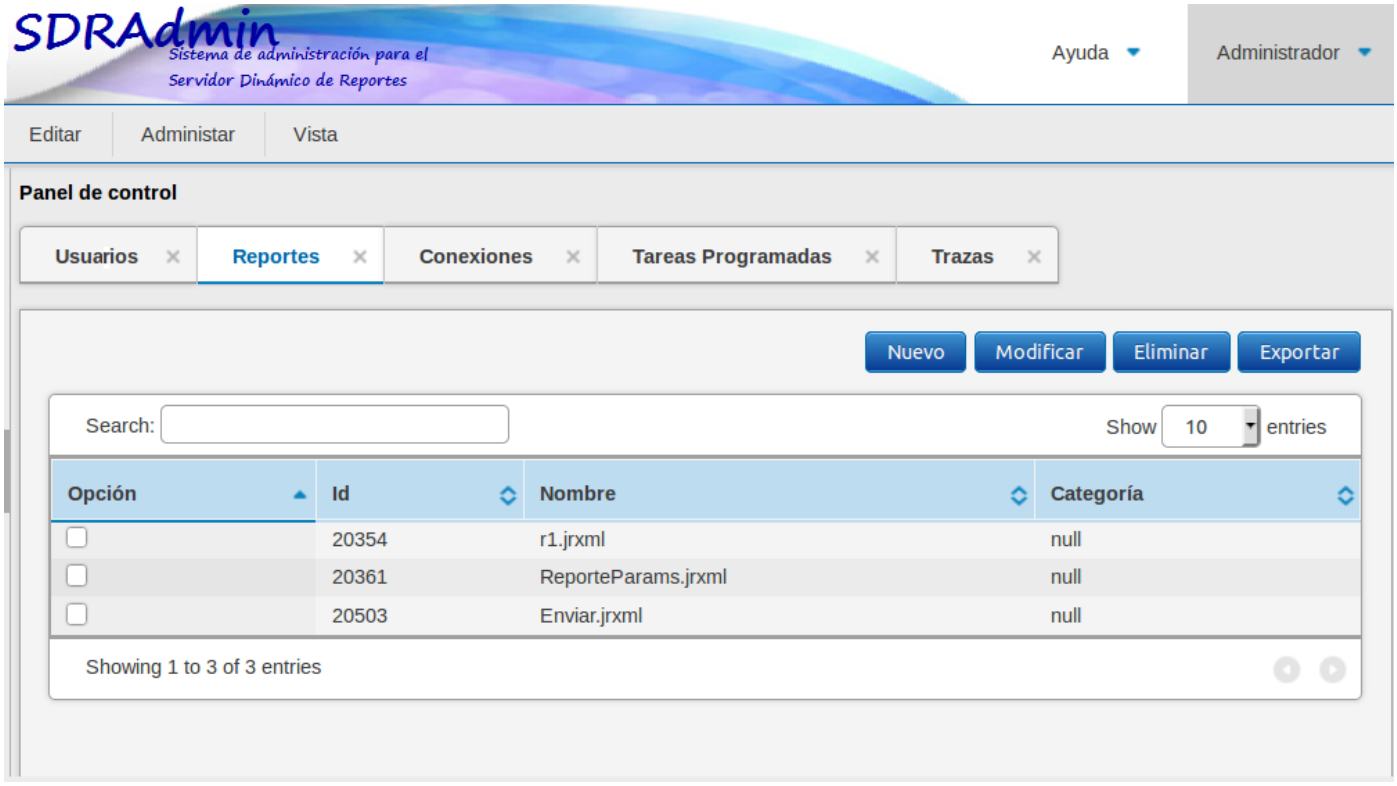

Fig. 12 Interfaz del caso de uso arquitectónicamente significativo Gestionar reportes.

3.3 Pruebas de software

Las pruebas se realizan mediante el proceso de ejecución de un programa con la intención de detectar errores, tiene éxito si identifica un error no detectado hasta entonces. Las pruebas son un elemento para la garantía de que el producto funcione correctamente. Esta etapa implica: Verificar la interacción de componentes, la integración adecuada de los componentes, que todos los requisitos se han implementado correctamente e identificar y asegurar que los defectos encontrados sean corregidos antes de entregar el software al cliente (20).

Las pruebas se realizan en diferentes momentos del ciclo de vida de un software para demostrar hasta qué punto este funciona correctamente. Existen diversos niveles de pruebas, como las pruebas de desarrollador, integración, unidad y aceptación. En el desarrollo de la fase de pruebas del sistema informático para la administración visual del SDR se aplicaron las pruebas de desarrollador. Además de

carga y estrés con el objetivo de probar si el sistema cumplía con los requisitos definidos en la fase de levantamiento de requisitos.

3.3.1 Pruebas de desarrollador

Durante el proceso de desarrollo al ir incorporando nuevas líneas de código, aumenta la posibilidad de cometer errores que comprometan la integridad de la aplicación. En esta etapa el desarrollador realiza un conjunto de pruebas con el objetivo de validar el funcionamiento del sistema antes de avanzar a otros niveles de prueba. Para esto se hace necesario identificar los tipos, métodos y herramientas de pruebas para la revisión del software.

Como tipo de prueba se aplicarán las pruebas funcionales, que tienen como objetivo principal probar que el sistema desarrollado cumpla con las funcionalidades definidas. Estas pruebas generalmente se llevan a cabo por los usuarios finales y los analistas. Se basan en el análisis de los datos de entrada y salida del sistema. Para este tipo de prueba se diseñará el documento de caso de prueba como una herramienta de apoyo al proceso de revisión.

Las pruebas funcionales tienen como objetivo detectar errores presentes en el proceso de entrada y salida de los métodos. Por cada caso de uso se debe realizar un caso de prueba, dividido en las secciones y detallando las funcionalidades descritas en él, además de cada variable que interactúa en el caso de uso en cuestión. Para cada caso de uso se conforman matrices de datos con el uso de valores válidos e inválidos evidenciándose el uso del método de caja negra con la técnica de partición de equivalencia.

Se realizaron un total de cuatro iteraciones de pruebas funcionales aplicando 16 casos de prueba diseñados que arrojaron un total de doce NC (No Conformidades). En la primera iteración se detectaron seis NC. En la segunda iteración se detectaron cuatro NC. En la tercera iteración se detectaron dos NC. En la cuarta iteración no se detectaron NC. Estas pruebas permitieron comprobar el funcionamiento del sistema y la validación de los campos. En el gráfico que se muestra en la Fig. 13 se visualiza la cantidad de pruebas por cada iteración así como el número de no conformidades.

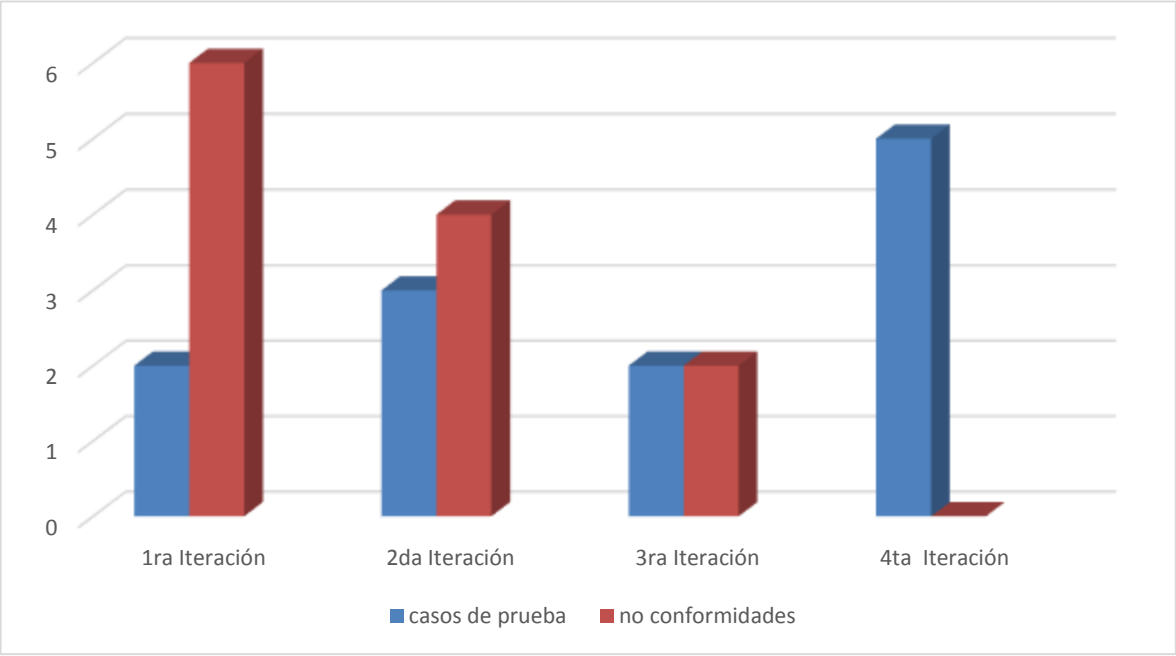

Fig. 13 Gráfica de las pruebas de desarrollador.

3.3.2 Pruebas de carga.

Su principal objetivo es comprobar el comportamiento del sistema frente a un uso concurrente de varios usuarios a la vez sobre el mismo. Durante las pruebas de carga, se pueden obtener tiempos de respuesta del sistema según el número de usuarios, carga de las máquinas y uso de memoria.

3.3.3 Pruebas de estrés.

Se utilizan en conjunto con las pruebas de carga. Estas consisten en aumentar el número de usuarios concurrentes o peticiones (en función del diseño del sistema) que se utilizarán en las pruebas de carga. Con estas pruebas se obtiene cuál es el máximo uso admisible por el sistema. También se podrá determinar qué elemento lo limita para estudiar una futura escalabilidad si es de hardware o de software.

3.3.4 Ejecución de las pruebas.

Se realizaron las pruebas de rendimiento con el uso de la herramienta ApacheJmeter 2.9, simulando peticiones a la aplicación de un conjuto de usuarios conectados concurrentemente. Con esta prueba se midió el tiempo de respuesta de todas las funcionalidades haciendo énfasis en las de mayor nivel de complejidad. Están asociadas al caso de uso Gestionar reportes. En la Tabla 6 se muestran los tiempos medios de respuesta obtenidos a través de la herramienta en segundos para distintas cantidades de usuarios.

Tabla 6 Resultado de las pruebas de rendimiento.

Cantidad de usuarios	Tiempo medio
10	0.2
25	3.7
70	6.8

CONCLUSIONES PARCIALES

El desarrollo de la aplicación permitió poner en práctica algunos estándares y estilos de codificación obteniendo una mejor comprensión del código.

La puesta en práctica de algunos estándares y estilos de codificación permitieron una mejor comprensión del código implementado. Además el análisis del modelo de implementación para la herramienta de administración permitió la obtención el diagrama de componentes del sistema, diseñado a nivel representacional con UML donde se visualiza la interacción entre los componentes. La realización de las pruebas de software empleando varios niveles de prueba y el método de caja negra con la técnica de partición de equivalencia permitió detectar varias NC en las primeras tres iteraciones arribando a una cuarta iteración donde quedaron solucionadas satisfactoriamente.

CONCLUSIONES GENERELALES

Una vez terminada la investigación se puede afirmar que todos los objetivos planteados fueron cumplidos llegando a las siguientes conclusiones:

- El estudio realizado de las principales herramientas de administración existentes y las tecnologías de gestión de reportes permitió obtener buenas prácticas y estilos en el desarrollo de soluciones de este tipo.

- La realización del análisis y diseño de los principales conceptos relacionados con la Herramienta para la administración del servidor dinámico de reportes, obteniendo como resultados los artefactos y diagramas necesarios que se utilizarían para guiar el desarrollo de la aplicación.

- El desarrollo de la herramienta de administración para el SDR permitió resolver los dieciséis requisitos funcionales identificados para facilitar la administración del SDR.

- EL diseño y ejecución de los niveles de prueba carga y extres Desarrollador empleando el método de caja negra y la técnica de partición de equivalencia, posibilitó comprobar el funcionamiento de la aplicación; así como las pruebas de carga y estrés permitió chequear el comportamiento de la herramienta bajo determinadas condiciones de software y hardware.

REFERENCIAS

1. JasperReports. . *JasperReports. .* [En línea] [Citado el: 4 de 18 de 2014.] http://community.jaspersoft.com/project/jasperreports-server..

2. Cristal Reports. *Cristal Reports.* [En línea] [Citado el: 18 de 4 de 2014.] http://www.crystalreports.com.

3. La Arquitectura Orientada a Servicios. *La Arquitectura Orientada a Servicios.* [En línea] [Citado el: 22 de 11 de 2013.] http://mahara.uji.es/artefact/file/download.php?file=54534&view=4648.

4. El Proceso Unificado de Desarrollo de Software. [aut. libro] James Rumbaugh Ivar Jacobson Grady Booch. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n.

5. Visual Paradigm For UML. *Visual Paradigm For UML.* [En línea] [Citado el: 15 de 11 de 2013.] http://www.visual-paradigm.com/product/vpsuite.

6. Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 25 de 11 de 2013.] http://www.visual.

7. IDES. Página Oficial. *IDES. Página Oficial.* [En línea] [Citado el: 9 de 11 de 2013.] http://ides.org.ar/..

8. Tomcat. *Tomcat.* [En línea] [Citado el: 5 de 10 de 2013.] http://tomcat.apache.org/tomcat-3.3-doc/Tomcat-Workers-HowTo.html.

9. PosrgreSQL Cuba. *PosrgreSQL Cuba.* [En línea] [Citado el: 15 de 10 de 2013.] http://postgresql.uci.cu.

10. Repositorio de la Universidad de la Plata. *Repositorio de la Universidad de la Plata.* [En línea] [Citado el: 20 de 10 de 2013.] http://sedici.unlp.edu.ar/bitstream/handle/10915/24942/Documento_completo__.pdf?sequence..

11. jquery. *jquery.* [En línea] [Citado el: 25 de 3 de 2014.] http://jquery.com/.

12. Ptrones. *Ptrones.* [En línea] [Citado el: 25 de 1 de 2014.] http://www.info-ab.uclm.es/asignaturas/42579/pdf/04-pitulo4a.pdf.

13. Arquitectura-de-Software. *Arquitectura-de-Software.* [En línea] [Citado el: 3 de 12 de 2013.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue..

14. Arquitectura de software. *Arquitectura de software.* [En línea] [Citado el: 14 de 2 de 2014.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue..

15. Patrones de Diseño. [En línea] [Citado el: 14 de 3 de 2014.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf..

16. Patrones de Diseño. *Patrones de Diseño.* [En línea] [Citado el: 2 de 10 de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf..

17. Escuela de Informacon y Ciencias de la computacion. *Escuela de Informacon y Ciencias de la computacion.* [En línea] [Citado el: 21 de 2 de 2014.] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm..

18. Jacobsony James, Rumbaugh Grady. *El Proceso Unificado de Desarrollo .* 2000.

19. Ingeniería de Software.Un enfoque práctico 6ta Edición. . [aut. libro] Roger S Pressman.

20. Pruebas de software. *Pruebas de software.* [En línea] [Citado el: 1 de 2 de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm..

BIBLIOGRAFÍA

1. JasperReports. JasperReports. [En línea] [Citado el: 18 de Abril de 2014.] http://community.jaspersoft.com/project/jasperreports-server.

2. Cristal Reports. Cristal Reports. [En línea] [Citado el: 18 de Abril de 2014.] http://www.crystalreports.com/.

3. La Arquitectura Orientada a Servicios (SOA) . La Arquitectura Orientada a Servicios (SOA) . [En línea] [Citado el: 11 de 22 de 2013.] http://mahara.uji.es/artefact/file/download.php?file=54534&view=4648.

4. Eclipse. Sito Oficial. [En línea] [Citado el: 12 de 11 de 2013.] http://www.eclipse.org/epf/general/OpenUP.pdf&prev=/search%3Fq%3Dopenup%26biw%3D1024%26bih%3D639.

5. Manual de Visual Paradimng.

6. Universidad Yacambú. [En línea] [Citado el: 5 de 12 de 2013.] http://www.oocities.org/es/avrrinf/tabd/Foro/Foro_UML.htm.

7. Proyecto de informática y tecnología. [En línea] [Citado el: 11 de 12 de 2013.] http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=844.

8. IDES. Página Oficial. [En línea] [Citado el: 29 de 11 de 2013.] http://ides.org.ar/.

9. Tomcat. [En línea] [Citado el: 25 de 11 de 2013.] http://tomcat.apache.org/tomcat-3.3-doc/Tomcat-Workers-HowTo.html.

10. PGAdmin. [En línea] [Citado el: 13 de 11 de 2013.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

11. Postgres SQL. Sito Oficial. [En línea] [Citado el: 13 de 11 de 2013.] http://www.postgresql.org.es/sobre_postgresql.

12. Escuela de Información y Ciencias de la computación. [En línea] [Citado el: 4 de 12 de 2013.] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

13. Patrones. [En línea] [Citado el: 29 de 11 de 2013.] http://www.info-ab.uclm.es/asignaturas/42579/pdf/04-pitulo4a.pdf. .

14. Patrones de Diseño. Patrones de Diseño. [En línea] [Citado el: 11 de 11 de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

15. Pressman. Ingeniería de Software.Un enfoque práctico. 6ta Edición. . Ingeniería de Software.

16. Valencia, Universidad Politécnica de. Patrones.

17. Larman, Craig. UML y Patrones 2da Edición.
18. Booch, Ivar, Jacobsony James, Rumbaugh Grady. El Proceso Unificado de Desarrollo. 2000.
19. Wordproces. Wordproces. [En línea] [Citado el: 12 de Noviembre de 2013.] http://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/..
20. Teknoda. [En línea] [Citado el: 28 de 11 de 2013.] http://www.teknodatips.com.ar/sap-netweaver/32-esa-y-web-services-en-sap-netweaver-introduccion.html.
21. Repositorio de la Universidad de la Plata. Repositorio de la Universidad de la Plata. [En línea] [Citado el: 11 de Diciembre de 2013.] http://sedici.unlp.edu.ar/bitstream/handle/10915/24942/Documento_completo__.pdf?sequence.
22. Pergamini. Pergamini. [En línea] [Citado el: 11 de Diciembre de 2013.] http://www.pergaminovirtual.com.ar/definicion/JavaScript.html..
23. Patrones de Diseño. [En línea] [Citado el: 29 de 11 de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.
24. Guia De RUP. [En línea] [Citado el: 11 de 11 de 2013.] http://xdeamx.files.wordpress.com/2011/07/guia-2.pdf.
25. Enciclopedia Cubana en la Red. Modelo de Dominio. [En línea] 2011. http://www.ecured.cu/index.php/Modelo_de_dominio.
26. Definición de. Definición de. [En línea] [Citado el: 10 de noviembre de 2013.] http://definicion.de/lenguaje-de-programacion.
27. Arquitectura de la IP. Arquitectura de la IP. [En línea] [Citado el: 29 de Noviembre de 2013.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/arquitectura_soft.mspx ..
28. Apuntes de Java. Apuntes de Java. [En línea] [Citado el: 29 de Noviembre de 2013.] http://www.apuntesdejava.com/2010/11/restful-la-forma-mas-ligera-de-hacer.html..
29. [En línea] [Citado el: 03 de 05 de 2014.] http://centrodeartigos.com/articulos-educativos/article_11443.html.
30. [En línea] [Citado el: 01 de 05 de 2014.] http://www.slideshare.net/rinconsete/pruebas-de-caja-blanca-y-negra.
31. [En línea] [Citado el: 01 de 05 de 2014.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.
32. [En línea] [Citado el: 25 de 04 de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

33. [En línea] [Citado el: 20 de 04 de 2014.] https://docs.google.com/document/d/1rbxDFM0zsbFDNRZeM2FoXfRDbYSiSt6tCdbYPA0qdzs/edit?hl=en_US&pli=1.
34. [En línea] [Citado el: 15 de 04 de 2014.] http://www.casanas.com.ar/attachments/Que_es_-_A_-_Conc_tecnicos.pdf .
35. [En línea] [Citado el: 10 de 04 de 2014.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.
36. [En línea] [Citado el: 15 de 03 de 2014.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.
37. [En línea] [Citado el: 03 de 03 de 2014.] http://www.slideshare.net/DaniSantia/diagrama-de-secuencia-17178366?qid=7fd9e8d5-9451-42f6-8740-1e11ff86f805&v=default&b=&from_search=1.
38. [En línea] [Citado el: 28 de 02 de 2014.] http://www.slideshare.net/andresmacea31/diagramas-de-interaccion-cun-monteria?qid=e384e0eb-480c-4c3f-8665-96fe3015821d&v=qf1&b=&from_search=1.
39. [En línea] [Citado el: 16 de 02 de 2014.] http://www.slideshare.net/andrescofran/diagrama-paquetes-colaboracion-y-componetes-6738524?qid=cfdf8442-8dea-4249-8855-42b34b359a01&v=qf1&b=&from_search=1.
40. [En línea] [Citado el: 14 de 02 de 2014.] http://www.slideshare.net/omarbeltrancelismendoza/05-modelo-dediseno?qid=b9da3fd1-ec6c-4952-82a8-e442d50fe25c&v=default&b=&from_search=5.
41. [En línea] [Citado el: 29 de 11 de 2013.] http://www.info-ab.uclm.es/asignaturas/42579/pdf/04-pitulo4a.pdf.

GLOSARIO DE TÉRMINOS

Artefactos: Son los elementos de entrada y salida de las actividades. Son productos tangibles del proyecto.

Iteraciones: Es un ciclo de desarrollo completo que genera como resultado una entrega de producto ejecutable.

SDRAdmin: Herramienta para mejorar la administración del servidor dinámico de reportes.

REST:

- Protocolo de comunicación, que define un conjunto de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. Se rige por cuatro principios fundamentales.
- Protocolo por el cual el SDR brinda los servicios.