Universidad de las Ciencias Informáticas

FACULTAD 6

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

Autor: Pablo Antonio Hernández Martínez

Tutores:

Ing. Yanet Rosales Morales

Ing. Keimer Montes Oliver

8 de Julio del 2015

"Año del 56 Aniversario de la Revolución"

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 __2015__.

Pablo Antonio Hernández Martínez

_____

Firma del Autor

Ing. Yanet Rosales Morales

_____

Firma del Tutor

Ing. Keimer Montes Oliver

_____

Firma del Tutor

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

Thomas Alva. Edison.

Datos de Contacto.

AUTOR:

Pablo Antonio Hernández Martínez

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

Correo: pamartinez@estudiantes.uci.cu

TUTORES:

Ing. Yanet Rosales Morales

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

Correo: ymorales@uci.cu

Ing. Keimer Montes Oliver

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

Correo: kmontes@uci.cu

Resumen

El SDR brinda un conjunto de funcionalidades las cuales permiten, gestionar reportes, usuarios, y conexiones con fuentes de datos además de programar tareas automáticas, gracias a estos servicios, las aplicaciones que lo utilizan pueden gestionar todos los procesos en la publicación y elaboración de reportes. A pesar de estas facilidades un usuario que trabaje directamente con el servidor lo realizaría mediante consola, necesitando de un alto grado de abstracción y conocimiento en informática. Dada esta situación se propone realizar un sistema para la administración visual del SDR, dicha aplicación tendría que contar con una interfaz gráfica de usuario a través de la cual se pudiera interactuar fácilmente con el servidor.

Luego de la implementación del sistema, se logró obtener una interfaz la cual facilita y guía al usuario durante todas las acciones que este realice en la misma. Mejorando la administración del servidor, además de permitir a un usuario con conocimientos básicos interactuar con SDR sin necesidad de consultar o requerir la presencia de un especialista.

Palabras claves:

Abstracción, Administración, Funcionalidades, Gestionar, Procesos

Abstract

The SDR provides a set of features which allow the user connection management reports with data sources and programming of automatic tasks, because these services use applications that can manage all the processes in publishing and reporting. Although these facilities a user interacts directly with the server take place via console, requiring a high degree of abstraction and computer knowledge. Given this situation is proposed that a system for the visual management of SDR, to improve management by the user, the application would have to have a graphical user interface through which they could interact more easily with server.

After the implementation of the system will be able to obtain an interface which facilitates and guides the user smoothly for all actions that take place in it. Enabling improved server management, and allows a user with basic knowledge in computer interact with SDR without consulting or require the presence of a specialist.

Keywords:

Abstraction, Features, Processes, Management.

Índice

Índice de figuras

Índice de tablas

Introducción

Los reportes representan una de las formas utilizadas para obtener informes detallados sobre un tema específico, dotando a quienes los 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 son generados por herramientas denominadas servidores de reporte, los cuales obtienen la información de diferentes fuentes de datos, aplicándole filtros y formatos para mostrarla a través de un diseño fácil de interpretar por el usuario, confiriéndole mayor utilidad, por ser más sencillo de interpretar los datos mediante un gráfico de barras que una hoja de cálculos con cientos de campos.

Un servidor de reportes es una aplicación en ejecución que contiene un conjunto de bibliotecas a través de las cuales gestionan y generan la información contenidas en los reportes, capaz de procesar las peticiones de un cliente y brindar una respuesta en concordancia. Los clientes usualmente se conectan al servidor a través de la red pero también pueden acceder a él a través de la computadora donde está funcionando.

La Universidad de las Ciencias Informáticas no se encuentra exenta en el desarrollo de aplicaciones para el diseño y la gestión de reportes, se encuentra dividida en varios centros como el Centro de Tecnología y Gestión de Datos (DATEC). Específicamente en el departamento de Desarrollo de Aplicaciones, se desarrollan y da soporte a activos de software que gestionan este tipo de tecnología entre los cuales se encuentra el SDR.

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 las siglas en inglés Service Oriented Architecture), implementando servicios de tipo Transferencia de Estado Representativo (REST por las siglas en inglés *Representational State Transfer*). Fue desarrollado en el lenguaje de programación Java y utiliza las bibliotecas libres de *JaspertReport* en su versión 5.0. Se crea principalmente con el objetivo de sustituir el 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.

A pesar de las facilidades que ofrece el SDR para el trabajo con reportes, a través de servicios web que son utilizados por otras aplicaciones, un usuario que trabaje directamente con el servidor sin utilizar dichas aplicaciones, lo realizaría mediante la consola, requiriendo la presencia de un especialista con

conocimiento del negocio, o un nivel medio de abstracción a la hora de utilizar los servicios que provee el SDR, lo cual puede ser un proceso engorroso, trayendo consigo el aumento del tiempo de administración e incremento de la complejidad al examinar los resultados de la ejecución de los servicios por el usuario en el servidor.

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

La presente investigación tiene como **objeto de estudio**: Los procesos de administración de servidores, enmarcado en el **campo de acción**: Los procesos de administración del Servidor Dinámico de Reportes.

El **objetivo general** de la presente investigación es: Desarrollar un sistema informático para la administración de los recursos de Servidor Dinámico de Reportes mediante una interfaz de usuario gráfica.

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

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

1. Construir el marco teórico sobre los mecanismos visuales para la administración de servidores, así como las tecnologías y herramientas empleadas en la investigación.
2. Análisis de los sistemas de administración de servidores, así como de las herramientas y tecnologías necesarias en el desarrollo de la investigación.
3. Caracterización de los mecanismos de integración con sistemas basados en arquitectura orientada a servicios, específicamente con los que utilizan servicios de REST.
4. Realización del análisis del sistema informático para la administración visual del SDR.

5. Realización del diseño del sistema informático para la administración visual del SDR a partir de los artefactos obtenidos en la fase de análisis.
6. Implementación de las clases diseñadas.
7. Confección de los casos de pruebas para comprobar el funcionamiento del sistema informático para la administración visual del SDR.
8. Realización de las pruebas funcionales a partir de los casos de prueba diseñados para la herramienta informática desarrollada.
9. Despliegue de la herramienta informática implementada.

Métodos teóricos:

Los métodos teóricos permiten revelar las relaciones esenciales del objeto de investigación, no observables directamente. Participan en la etapa de asimilación de hechos, fenómenos y procesos, y en la construcción del modelo e hipótesis de investigación. A continuación se detallan los métodos empleados:

Analítico – sintético: la utilización de este método permitió la comprensión y funcionamiento de los sistemas de administración para servidores a través del análisis de documentos, libros, artículos y otras fuentes bibliográficas de diferentes autores, lo que permitió definir las características principales de estos sistemas.

Inductivo – deductivo: este método permitió caracterizar los componentes fundamentales de los sistemas de administración para servidores.

Modelación: con este método se realizó el proceso de diseño mediante la abstracción de sus elementos fundamentales utilizando un lenguaje de modelado unificado.

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

En el **Primer Capítulo: "Fundamentación teórica de la solución"**, se presentan los principales conceptos 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 de la solución”**, se definen los requisitos que la herramienta debe cumplir, se realizan los diagramas de clases para los casos de uso arquitectónicamente significativos. Se elaboran los Diagramas de Despliegue, Arquitectura y Casos de uso entre otros, 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 de la solución”**, 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: Fundamentación teórica de la solución

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

1.1. Interfaz de Usuario Gráfica

Una interfaz de usuario gráfica (GUI por sus siglas en inglés *Graphical User Interface*), es parte de un programa que permite al usuario interactuar con el mismo, utiliza un conjunto de objetos gráficos e imágenes a través de los cuales presentan la información y las acciones posibles sobre esta. Tiene como principal objetivo proporcionar un entorno gráfico sencillo que permita la utilización de un servidor, sistema operativo o software.

1.1.1. Sistema de administración visual

Los sistemas de administración visual o interfaz de administración, constituye la vista principal de un producto a través de la cual se puede administrar los recursos de la aplicación. Dichas interfaces ofrecen a los usuarios una abstracción total de la configuración, posibilitando su uso para usuarios menos avanzados en el conocimiento de la informática.

Sistema de administración Webmin

Webmin es una herramienta de configuración de sistemas y servidores accesible vía web. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración y apagado del equipo, así como modificar y controlar muchas aplicaciones libres, como los servidores Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros.

Características principales:

- Construido a partir de módulos.
- Interfaz a los archivos de configuración.
- Liberado bajo la licencia Distribución de Software Berkeley (BSD por sus siglas en inglés *Software Distribution*).

Ventajas

- Las tareas básicas de administración como la gestión de usuarios se realizan mediante una GUI.
- Su uso no se limita a personas con conocimientos avanzados de informática, pueden utilizarlo usuarios con conocimiento básico en la configuración y gestión de servidores.
- Facilita las tareas de administración de servidores y configuración de servicios.

Ejemplo de la interfaz de *Webmin*.

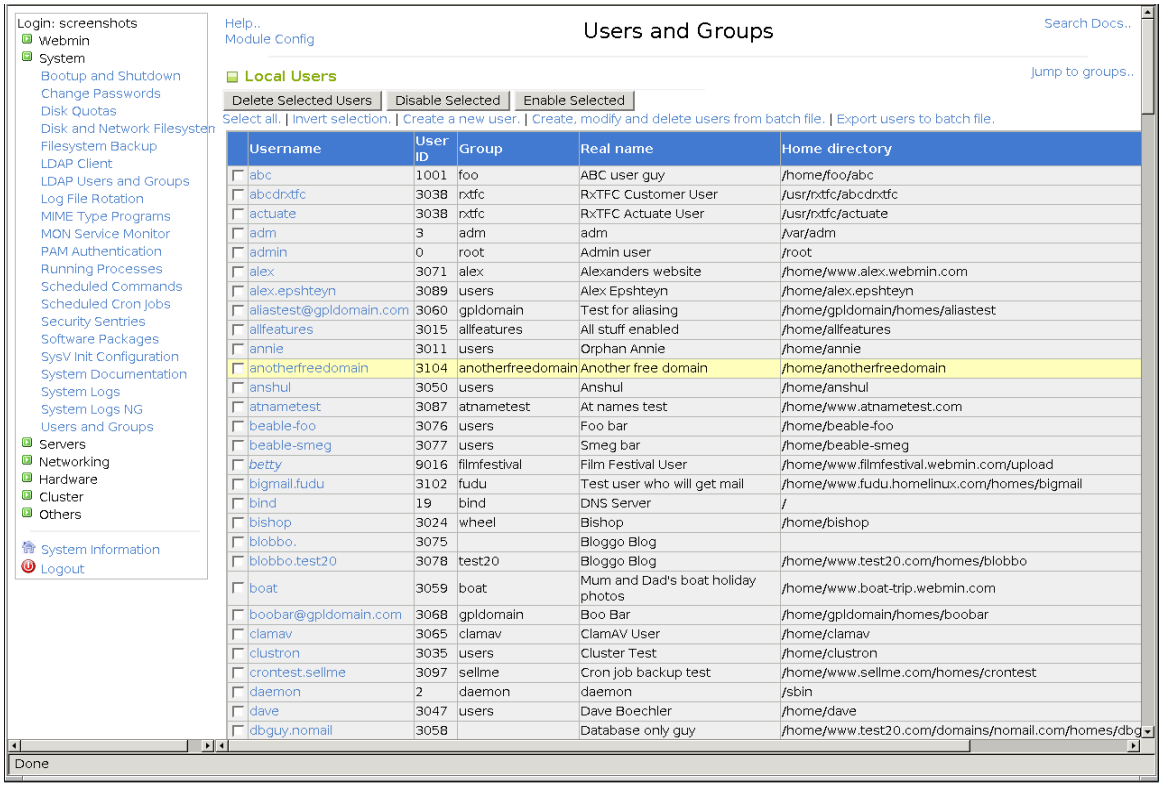

***Fig 1. Gestión de usuarios y grupos*.**

. ***Sistema de administración PhpMyAdmin***

PhpMyAdmin sistema para la administración visual del servidor de bases de datos MySQL, de distribución libre. Permite realizar todo tipo de operaciones sobre las bases de datos, como crear, borrar y modificar tablas, realizar consultas, definir usuarios y asignar permisos, además de poder administrar las bases de datos de manera local y remota.

Características principales:

- Mantenimiento de usuarios y sus privilegios.
- Exportación a varios formatos.
- Creación del despliegue de la base de datos en un gráfico exportado a PDF.

Ventajas:

- Interfaz de administración web.
- Interfaz web intuitiva.
- Administración remota de bases de datos.

Ejemplo de las interfaces de *PhpMyAdmin*

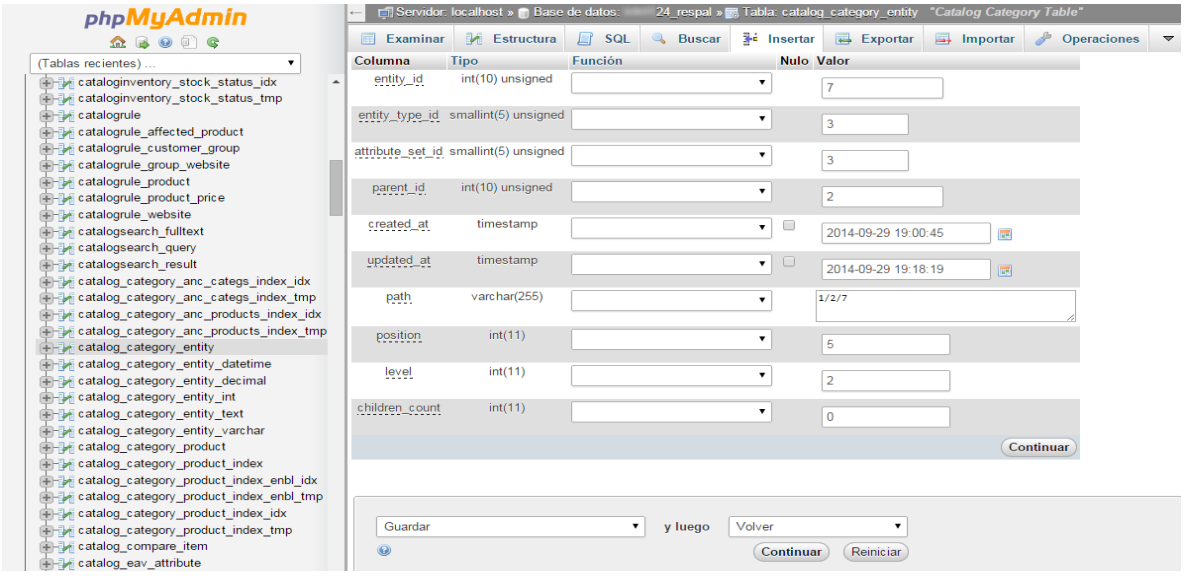

Fig 2. Gestión de tablas y columnas para las bases de datos.

Al realizar un estudio de las herramientas de administración, se determinó que el sistema debe contar con una interfaz intuitiva, con un bajo grado de complejidad que le permita al usuario final realizar las operaciones de forma fluida permitiendo el aprendizaje para garantizar así una mejora en cuanto a la administración de SDR.

1.2. 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 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. (1)

1.2.1. 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, 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. (2)

1.2.2. 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. (3)

Se rige por cuatro principios fundamentales:

- Utiliza los métodos HTTP de manera explícita.
- No mantiene estado.
- Expone *URIs* con forma de directorios.
- Transfiere XML, (JSON por sus siglas en inglés *JavaScript Object Notation*).

Una de las características clave 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 clientes 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, borrar y los métodos HTTP. (3)

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

- **Tabla 1. Métodos definidos que se utilizarán para REST**

Notación	Descripción
@PATH(ruta)	Establece la ruta base más 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 con extensión xml.
@POST	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.

En la tabla anterior se muestran los métodos principales que se utilizarán en la aplicación para realizar las llamadas a los servicios del SDR.

Se decide utilizar el protocolo REST ya que los métodos mencionados en la tabla 1 facilitan la comunicación con el servidor, son flexibles, permitiendo la escalabilidad de la aplicación, además es el protocolo a través del cual el SDR brinda los servicios.

1.3. Metodología de Desarrollo

Las metodologías de desarrollo de software constituyen un conjunto de procedimientos y técnicas para ayudar a los equipos de desarrollo de software con la documentación durante el proceso de concepción de un software. 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. Las metodologías de desarrollo se encuentran divididas en dos grupos: ágiles y robustas, diferenciándose por la flexibilidad, tiempo de desarrollo y documentación generada. (4)

Dentro de las ventajas que se pueden observar en las metodologías ágiles se encuentran: flexibilidad ante los cambios del proyecto de forma rápida y moderada; incluye a los clientes como parte del equipo de

desarrollo y tienen menos dependencia de la documentación. Dentro de las metodologías ágiles se pueden encontrar: Programación Extrema (XP por sus siglas en inglés *Xtreme Programming*)**,** *OpenUP* (Version ágil de RUP) y Cristal.

1.3.1. OpenUP

OpenUp es un proceso ágil y unificado, contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de software. *OpenUp* abraza una filosofía pragmática y ágil que se centra en la naturaleza colectiva de desarrollo de software. Es un proceso iterativo que es mínimo, completo y extensible que puede utilizarse tal cual o ampliarse para obtener una variedad de tipos de proyecto. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y estar guiado por casos de uso. (4)

Luego del estudio realizado se decide utilizar la metodología *OpenUP* ya que está diseñada para equipos pequeños y permite la detención temprana de errores gracias al ciclo iterativo e incremental que posee. Tiene un enfoque centrado en la arquitectura además de evitar la elaboración de grandes cantidades de documentación e iteraciones innecesarias requeridas en la metodología RUP.

1.4. 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 que facilitan al usuario el modelado de procesos. Dentro de estas herramientas se encuentran *Visual Paradigm for UML, Rational Rose, Easy CASE y CASE Studio* siendo la primera una de las más utilizadas. (5)

1.4.1. 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. (5)

1.4.2. Visual Paradigm for UML 8

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 actividades. Esta herramienta genera documentación del sistema en formato PDF, HTML y MS Word, la cual apoya gran parte del ciclo de desarrollo del software. (5)

Entre sus principales características se encuentran: modelado de procesos del negocio, administración de requisitos, generación de la capa objeto-relacional, generación de código e ingeniería inversa. (5) También posee una interfaz de usuario amigable y disponible para los Sistemas Operativos Linux, Windows, y Mac OS y soporta el lenguaje UML como su nombre indica, por lo que se decide la utilización de esta con el fin de realizar el modelado y documentación del sistema.

1.5. Herramientas y lenguajes de programación utilizados

Para el desarrollo de la aplicación se hace necesario la utilización de herramientas y lenguajes de programación que faciliten el diseño e implementación de la misma. 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 obteniendo algún resultado. Entre las herramientas de programación se pueden encontrar: la plataforma de programación *NetBean*s IDE, Eclipse, entre otros

Un lenguaje de programación es el idioma artificial utilizado para expresar procesos que pueden ser llevados a cabo por máquinas. Está compuesto por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Entre los lenguajes de programación se pueden encontrar: *JavaScript, Java, C++, entre otros*.

1.5.1. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés de *Integrated Development Environment*), es un entorno de programación 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. (6)

NetBeans IDE

El *NetBeans IDE* es una herramienta para programadores escrita en Java que 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 encuentran: 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.

También 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. (6)

Se selecciona el *NetBeans* IDE en su versión 7.3 para desarrollar la solución propuesta por las características mencionadas anteriormente, además incluye extensiones que pueden ser útiles durante el proceso de desarrollo tales como: editores visuales de interfaces y compiladores de código, además de un conjunto de bibliotecas útiles que facilitan el desarrollo de aplicaciones informáticas.

1.5.2. Lenguaje de programación JavaScript

JavaScript es un lenguaje de programación interpretado, basado en el estándar *ECMAScript*[1]. Se define como orientado a objetos, basado en prototipos 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. (7)

La decisión de utilizar este lenguaje está dada por lo descrito anteriormente y por la facilidad de interpretación de su sintaxis por parte de los desarrolladores.

1.5.3. Marco de trabajo

Se define como marco de trabajo un conjunto estandarizados de conceptos, prácticas y criterios que se establecen 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, se define conjunto de bibliotecas orientadas a la reutilización a gran escala de componentes software para el desarrollo rápido de aplicaciones. (8)

Dentro de las ventajas que ofrece la utilización de marcos de trabajos están:

- 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 de software.
- La programación de componentes que siguen una política de diseño uniforme.

[1] **ECMAScript**: Especificación de lenguaje de programación publicada por ECMA International.

JQuery

Es un marco de trabajo de JavaScript el cual cuenta con un conjunto de funciones implementadas para el desarrollo de aplicaciones web; este permite agregar efectos y funcionalidades complejas como galerías de fotos dinámicas y elegantes, validación de formularios y calendarios. También 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*) sin tener en cuenta los detalles complejos de la programación. (8)

Por lo anteriormente expuesto es el marco de trabajo seleccionado para desarrollar el sistema para la administración del SDR ya que el mismo debe ser ligero y amigable para el usuario.

1.6. 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*. (9)

1.6.1. Apache Tomcat

Apache Tomcat, es un servidor web multiplataforma que funciona como contenedor de aplicaciones. Se desarrolla en el proyecto denominado *Jackarta* bajo la licencia *Apache* 2.0 que implementa las especificaciones de los *servlets*[2] 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. (9)

---

2 **Servlets**: Pequeño programa que corre en un servidor. Por lo general son aplicaciones Java que corren en un entorno de servidor web.

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.

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 la conclusión de que la herramienta para la administración a implementar debe contar con una interfaz intuitiva, con un bajo grado de complejidad que le permita al usuario final realizar operaciones de forma fluida permitiendo el aprendizaje. También fueron seleccionados las tecnologías, herramientas y metodología que enuncian a continuación, el entorno de desarrollo NetBeans IDE en su versión 7.3, JavaScript como lenguaje de programación del lado del cliente, *JQuery* en su versión 1.9.1 como marco de trabajo, Visual Paradigm for UML en su versión 8.0 como herramienta de modelado, como lenguaje de modelado UML en su versión 2.0 el servidor Apache Tomcat 7 y como metodología de desarrollo OpenUP.

CAPÍTULO 2: Análisis y diseño de la solución

En el presente un capítulo se realiza el análisis de las características que debe tener el sistema de administración. Se identifican los requisitos funcionales y no funcionales. Serán definidos y detallados los casos de uso, actores y sus relaciones. Se selecciona un estilo arquitectónico para la solución e identifican los patrones de diseño necesarios para la organización de las clases.

2.1. Modelo del dominio del SDRadmin

El modelo de dominio es un diagrama 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. (10)

En el modelo de dominio que se presenta a continuación se muestra como un usuario el cual tiene un rol, utiliza una aplicación, a través de la cual puede acceder a los diferentes servicios brindados por el SDR. Dichos servicios pueden ser de tareas programadas, reportes y conexiones.

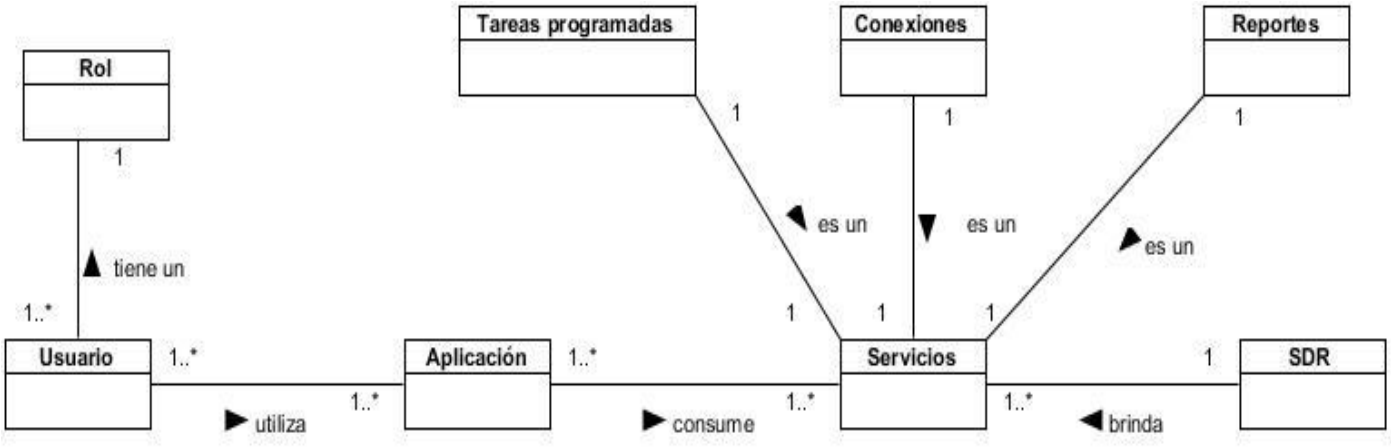

Fig 3. **Modelo del dominio del sistema.**

2.1.1. Definición de los conceptos del Modelo del Dominio del Sistema

Tabla 2.Descripción de los conceptos del modelo de dominio.

Entidades	Descripción
Usuario	Persona que interactúa con la aplicación a través de la que puede acceder a los distintos servicios que brinda el SDR.
Aplicación	Sistema informático que consume los servicios brindados por el SDR.
Servicios	Conjuntos de funcionalidades brindadas por el servidor, utilizados por otras aplicaciones garantizando la comunicación con dicho servidor.
SDR	Aplicación informática que brinda servicios para el trabajo con reportes, control de usuarios, conexiones con las fuentes de datos y programación de tareas.
Reporte	Archivo que contiene información detallada sobre un tema específico de diferentes fuentes de datos.
Tarea programada	Funcionalidad que permite a través del servidor gestionar las tareas que se ejecutan de forma automáticas en el sistema.
Conexión	Vínculo que se establece entre dos o más sistemas informáticos que permite la comunicación entre los mismos.
Rol	Papel desempeñado por los usuarios lo cual permite trabajar con el sistema en dependencia de los permisos del rol.

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. (11)

2.2.1. Requisitos funcionales

Un **requisito funcional** define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requerimientos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que un sistema debe cumplir. (11)

Requisitos funcionales identificados

- **RF1 Adicionar usuario**

Descripción: El sistema a través de una interfaz gráfica de usuario debe permitir entrar los datos necesarios para crear un nuevo usuario y luego procesarlos.

Entrada: Datos del nuevo usuario (nombre, usuario, correo, contraseña, sub-red y rol).

Salida: El nuevo usuario queda adicionado al sistema y a partir de ese momento tiene acceso al sistema según el rol que se le asigne.

- **RF2 Modificar usuario**

Descripción: El sistema a través de una interfaz debe 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, contraseña, sub-red y rol).

Salida: El usuario queda modificado en la base de datos, refrescándose la lista de los usuarios.

- **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 asignado por el SDR a la hora de adicionar el usuario.

Salida: Muestra el listado de los usuarios del sistema actualizado.

- **RF5 Buscar usuario**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los usuarios, seleccionar uno y buscar los datos correspondientes.

Entrada: Identificador del usuario seleccionado asignado por el SDR a la hora de adicionar el usuario.

Salida: Datos del usuario seleccionado (nombre, usuario, correo, contraseña, sub-red y rol).

- **RF6 Listar reportes exportados**

Descripción: El sistema a través de una interfaz debe permitir listar los reportes exportados por el usuario autenticado.

Entrada: No procede.

Salida: Muestra el listado de los reportes exportados.

- **RF7 Visualizar reporte exportado**

Descripción: El sistema a través de una interfaz, luego de listar los reportes exportados, debe dar la opción al usuario de seleccionar un reporte exportado y visualizarlo en el navegador.

Entrada: Datos del reporte seleccionado (identificador del reporte seleccionado).

Salida: Muestra el reporte seleccionado en el navegador.

- **RF8 Eliminar reporte exportado**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes exportados, seleccionar uno y eliminarlo.

Entrada: Datos del reporte seleccionado (identificador del reporte seleccionado).

Salida: Se elimina el reporte del listado de reportes exportados por el usuario.

- **RF9 Buscar reporte exportado**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes exportados seleccionar uno y buscar sus datos.

Entrada: identificador del reporte exportado seleccionado.

Salida: Datos del reporte seleccionado.

- **RF10 Exportar reporte**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes seleccionar uno y exportarlo.

Entrada: Datos del reporte seleccionado (Id del reporte seleccionado, parámetros del reporte (opcional), formato y conexión).

Salida: Crea un archivo en el formato especificado mostrándolo en el navegador y brindándole al usuario la posibilidad de guardarlo.

- **RF11 Adicionar tarea automática**

Descripción: El sistema a través de una interfaz debe permitir al usuario entrar los datos de una tarea en dependencia de su tipo (correo o ftp).

Entrada: Datos de la nueva tarea ftp (ubicación, usuario, contraseña, identificador del reporte seleccionado, identificador de la conexión seleccionada y tiempo en que se ejecuta la tarea), si la tarea es de correo los datos serían (correo de destinatario, identificador del reporte seleccionado, identificador de la conexión seleccionada y tiempo en que se ejecuta la tarea).

Salida: Se adiciona una nueva tarea según su tipo en el sistema.

- **RF12 Listar tarea automática**

Descripción: El sistema a través de una interfaz debe permitir al usuario ver el listado de las tareas automáticas.

Entrada: No procede

Salida: Se muestra un listado de las tareas automáticas del usuario autenticado.

- **RF13 Eliminar tarea**

Descripción: El sistema a través de una interfaz debe permitir luego de listar las tareas seleccionar una y eliminarla.

Entrada: identificador de la tarea seleccionada.

Salida: Se elimina la tarea y se actualiza el listado de tareas automáticas.

- **RF14 Buscar tarea**

Descripción: El sistema a través de una interfaz debe permitir luego de listar tareas, seleccionar una y buscar sus datos.

Entrada: identificador de la tarea seleccionada.

Salida: Busca los datos de la tarea y los muestra al usuario.

- **RF15 Adicionar reporte**

Descripción: El sistema a través de una interfaz debe permitir entrar los datos necesarios para crear un nuevo reporte, luego procesarlos y adicionar el reporte al sistema.

Entrada: Datos del nuevo reporte (descripción, nombre archivo del reporte en formato .jrxml y categoría).

Salida: El nuevo reporte queda registrado en el sistema.

- **RF16 Modificar reporte**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes del usuario autenticado, buscar los datos específicos de un reporte seleccionado, modificarlo, actualizar la base de datos y la lista de reportes.

Entrada: Datos a modificar del reporte seleccionado (descripción, nombre, categoría y archivo del reporte en formato .jrxml).

Salida: El reporte queda modificado y se actualiza la lista de reportes.

- **RF17 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.

- **RF18 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.

- **RF19 Descargar reporte**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes, seleccionar uno y descargarlo en la computadora.

Entrada: Datos del reporte seleccionado (Id del reporte seleccionado).

Salida: Guarda un archivo en el formato jrxml perteneciente al reporte seleccionado.

- **RF20 Vista previa de reporte**

Descripción: El sistema a través de una interfaz debe permitir seleccionar un archivo con extensión jrxml externo e ingresar los datos de una conexión y mostrar en el navegador una vista previa del reporte.

Entrada: Datos del reporte (fichero xml y datos de la conexión).

Salida: Se muestra una vista previa del reporte en el navegador.

- **RF21 Buscar reporte**

Descripción: El sistema a través de una interfaz debe permitir luego de listar los reportes, seleccionar uno y buscar sus datos.

Entrada: Datos del reporte seleccionado (Id del reporte seleccionado).

Salida: Muestra los datos del reporte seleccionado.

- **RF22 Adicionar conexión**

Descripción: El sistema a través de una interfaz debe permitir entrar los datos necesarios para crear una nueva conexión y luego procesar los datos.

Entrada: Datos de la nueva conexión (gestor de base de datos, nombre de la conexión, nombre de la base de datos, puerto, usuario, contraseña e Ip del servidor).

Salida: La nueva conexión queda adicionada al sistema.

- **RF23 Modificar conexión**

Descripción: El sistema a través de una interfaz debe permitir luego de listar las conexiones del sistema, buscar los datos específicos de una conexión y modificarla.

Entrada: Datos a modificar de la conexión seleccionada (gestor de base de datos, nombre de la conexión, nombre de la base de datos, puerto, usuario, contraseña e Ip del servidor).

Salida: La conexión queda modificada en la base de datos del sistema.

- **RF24 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.

- **RF25 Buscar conexión**

Descripción: El sistema a través de una interfaz debe de mostrar un listado de las conexiones existentes en el mismo seleccionar una y buscar sus datos.

Entrada: identificador de la conexión.

Salida: Muestra los datos de la conexión seleccionada (gestor de base de datos, nombre de la conexión, nombre de la base de datos, puerto, usuario, contraseña e ip del servidor).

- **RF26 Eliminar conexión**

Descripción: El sistema a través de una interfaz debe permitir luego de listar las conexiones del sistema, seleccionar una y eliminarla.

Entrada: identificador de la conexión seleccionada.

Salida: Muestra el listado de las conexiones del sistema actualizado.

- **RF27 Autenticar usuario**

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.

- **RF28 Cerrar sección**

Descripción: El sistema a través de una interfaz debe permitir al usuario desconectarse del servidor y se re-direcciona a la página index.

Entrada: Datos (token enviado por el servidor en el momento de la autenticación).

Salida*:* El usuario cierra su sesión y es re-direccionado a la interfaz de autenticación.

2.2.2. Requisitos no funcionales

Normalmente los requisitos no funcionales están vinculados requisitos funcionales, estos representan las necesidades del ordenador donde será utilizado el sistema así como las propiedades que el producto deberá poseer y las restricciones del mismo. Siendo fundamental el cumplimiento de estos requisitos ya que en caso contrario puede afectar la calidad del sistema. (11)

Requisitos de usabilidad

RNF1 Software requerido para el funcionamiento del sistema

Pc servidor:

El sistema operativo: GNU/Linux preferentemente Ubuntu 12.4 o superior.

Paquete: Apache Tomcat 7.2.

Pc cliente:

Navegador web: Firefox versión 29.0 o superior.

RNF2 Hardware requerido para el funcionamiento de la aplicación

Pc servidor:

Ordenador Pentium V o superior

Memoria RAM mínimo: 2Gb.

Disco Duro: 3Gb libres de capacidad.

Restricciones del diseño

RNF3 Marco de trabajo para el desarrollo del sistema del lado cliente

Para el desarrollo de las interfaces de usuario se propone el uso del marco de trabajo *JQuery* en su versión 1.9.1.

RF4 Interfaz

La interfaz debe ser intuitiva y con bajo grado de complejidad, permitiendo ser utilizada por usuarios que posean conocimiento básico de informática.

Requisitos para la documentación de usuarios y ayuda del sistema

Requisitos de licencia

RNF5 Requisitos de licencia

No posee ningún requisito de licencia ya que la aplicación fue implementada con software libre como NetBeansIde.

Requisitos de seguridad

RNF6 Seguridad

Confidencialidad: El sistema debe de garantizar el acceso a la información perteneciente al usuario autenticado protegiendo así la confidencialidad de su información.

Disponibilidad

El sistema debe garantizar el acceso a la información en todo momento siempre y cuando los servidores se encuentren en funcionamiento.

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. (10)

2.3.1. Diagrama de casos de uso del sistema

El diagrama de casos de uso (DUC) representa la forma de como un Cliente (Actor) se relaciona con el sistema en desarrollo, además de la manera, tipo y orden en como los elementos interactúan (operaciones o 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. (10)

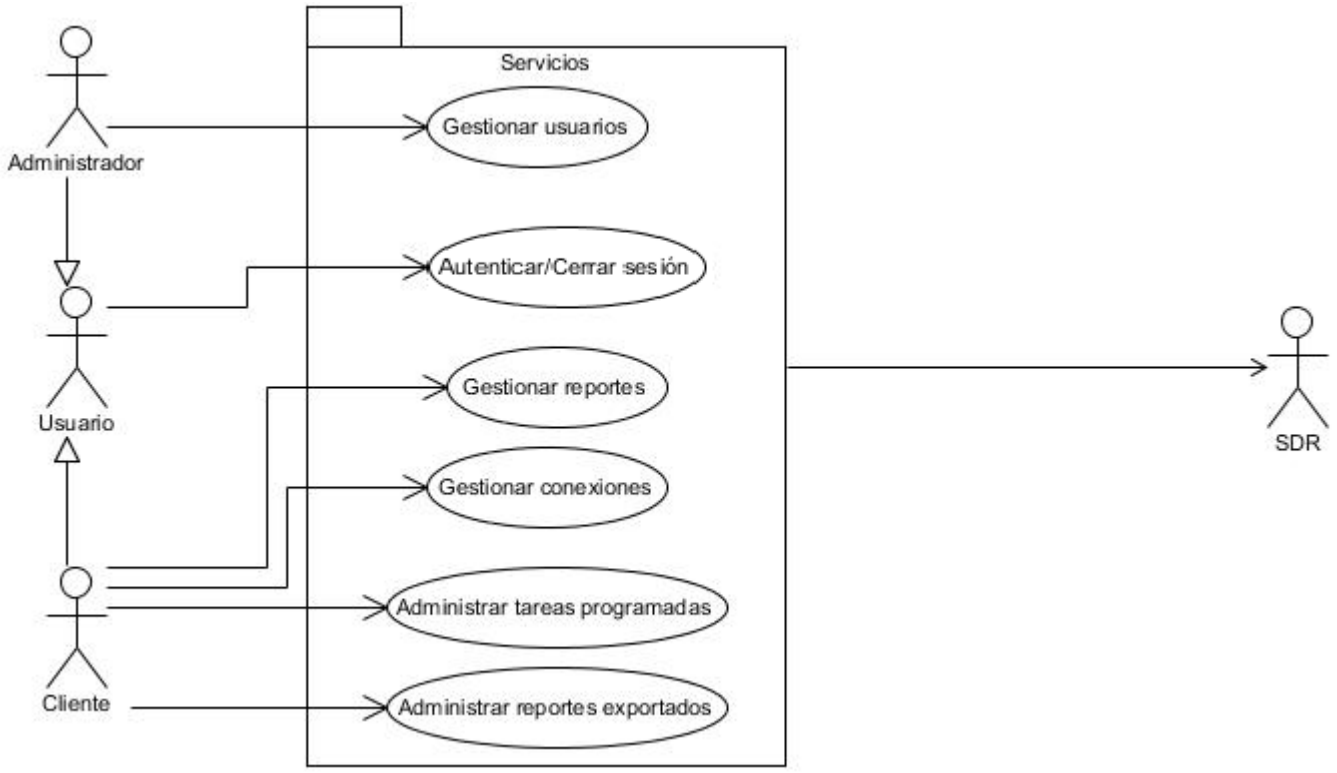

Fig 4.Diagrama de casos de uso.

En la Fig 4 se muestra el diagrama de casos de uso del Sistema para la administración Visual del Servidor Dinámico de Reportes, identificándose seis casos de uso en los cuales se encuentran agrupados los veintiocho requisitos funcionales identificados.

Patrones de casos de uso utilizados

Múltiples actores rol común

Este patrón indica que dos o más actores del sistema juegan un mismo rol sobre un caso de uso. En el sistema en cuestión se evidencia en la relación existente entre los actores Cliente y Administrador con el Usuario.

Fig 5. Evidencia del patrón múltiples actores rol común.

CRUD

Es utilizado para agrupar los requisitos de acuerdo a las operaciones que se realizan sobre un objeto específico, este patrón se divide en dos categorías en dependencia de si se realizan todas las funcionalidades sobre un objeto o parte de estas dando lugar al CRUD total o CRUD parcial respectivamente. Evidenciado el uso del CRUD total en los casos de uso Gestionar usuarios, Gestionar conexiones, Gestionar reportes y el CRUD parcial en los casos de uso Administrar reportes exportados y Administrar tarea programada.

2.3.2. Actores del sistema

Para el sistema en cuestión, se cuenta con 3 actores humanos y el SDR que es un sistema que interactúa con la aplicación con el cual intercambia información. Los actores poseen un rol que es determinado por el conjunto de casos de uso al que tiene acceso, en este caso se identificaron cuatro actores principales: usuario, administrador, cliente y SDR.

Tabla 3 Descripción de los actores del sistema.

Nombre del actor	Descripción
Usuario	Actor general del sistema el cual está asociado al caso de uso Autenticar usuarios/Cerrar sesión, del cual heredan los usuarios Administrador y Cliente.
Administrador	Actor del sistema que hereda del actor Usuario y tiene la función de gestionar los usuarios en el sistema.
Cliente	Actor del sistema el que hereda del actor Usuario y tiene acceso a las demás funciones del sistema.
SDR	Sistema informático el cual recibe peticiones de datos y envía los mismos al cliente mediante servicios.

2.3.3. Descripción textual del caso de uso arquitectónicamente significativo

Tabla 4 Descripción Casos de uso gestionar reportes

Objetivo	El caso de uso permite a los usuarios registrados el sistema la gestión de sus reportes.
Actores	Cliente.
Resumen	El caso de uso se inicia cuando el cliente va a la pestaña Reportes, donde el sistema muestra un listado con los reportes del cliente y presiona cualquiera de los siguientes botones. a) **"Adicionar".** El usuario gestiona los datos necesarios para crear un nuevo reporte. b) **"Modificar datos".** El usuario modifica los datos de un reporte seleccionado. c) **"Eliminar"**. El usuario elimina un reporte seleccionado. d) **"Modificar jrxml"**. El usuario modifica el fichero jrxml de un reporte seleccionado. e) **"Vista previa"**. El usuario obtiene la vista previa de un reporte. f) **"Exportar"**. El usuario exporta los reportes en el formato deseado. g) **"Descargar".** El usuario descarga el fichero .jrxml del reporte seleccionado.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Que el usuario este autenticado en el sistema.

<table>
<tr><td>Referencias</td><td colspan="2">RF15, RF16, RF17, RF18, RF19, RF20, RF21, RF10.</td></tr>
<tr><td>Postcondiciones</td><td colspan="2">En dependencia de la acción, se crea un nuevo reporte, se modifican los datos y el archivo de un reporte existente, se elimina un reporte del sistema, se exporta un reporte, se muestra la vista previa de un reporte antes de añadirlo al sistema y se descarga el fichero .jrxml del reporte.</td></tr>
<tr><td colspan="3">Flujo de eventos</td></tr>
<tr><td colspan="3">Flujo básico Gestionar usuario</td></tr>
<tr><td></td><td>Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td>El usuario presiona cualquiera de los siguientes botones.
“Adicionar”
“Modificar datos”
“Eliminar”
“Modificar jrxml”
“Vista previa”
“Exportar”
“Descargar”</td><td></td></tr>
<tr><td>2.</td><td></td><td>El sistema en dependencia del botón seleccionado por el usuario muestra los siguientes formularios.
Adicionar. Ver sección Nuevo reporte.
Modificar datos. Ver sección Modificar datos.
Eliminar. Ver sección Eliminar reporte.</td></tr>
</table>

		➢ **Modificar jrxml**. Ver sección Modificar jrxml. ➢ **Vista previa**. Ver sección Vista previa. ➢ **Exportar reporte**. Ver sección Exportar reporte. ➢ **Descargar** Ver sección Descargar.
Flujo básico Sección nuevo reporte (acción satisfactoria)		
1.		El sistema muestra el formulario para el nuevo reporte.
2.	El cliente introduce los datos requeridos para el nuevo reporte. (Nombre, Descripción, categoría y fichero).	
3.	El cliente presiona el botón **"Aceptar"**.	
4.		El sistema verifica que ninguno de los campos haya quedado en blanco.
5.		El sistema verifica que los datos cumplan con el formato requerido.
6.		El sistema crea el nuevo reporte refresca el listado de los reporte del sistema finalizando así el caso de uso.
Flujos alternos		
Nº 1 Sección nuevo reporte (Existen campos en blanco)		
	Actor	**Sistema**
4.1		El sistema muestra un mensaje diciendo al usuario que existen campos vacíos volver al fuljo norma de

		eventos 2
Nº 2 Sección nuevo reporte (Los datos no cumplen con el formato específico)		
	Actor	**Sistema**
5.1		El sistema muestra un mensaje diciendo al usuario los datos que no presentan el formato correcto, volver al fuljo norma de eventos 2.
Flujo básico Sección modificar datos (acción satisfactoria)		
1.		El sistema muestra un listado con los reportes existentes.
2.	El usuario selecciona un reporte.	
3.	El usuario presiona el botón **"Modificar datos"**.	
4.		El sistema busca los datos del reporte seleccionado y muestra un formulario con los datos Descripción, Nombre, Categoría y Fichero jrxml.
5.	El usuario modifica los datos deseados.	
6.	El usuario presiona el botón **"Aceptar"**.	
7.		El sistema verifica que no existan campos en blanco.
8.		El sistema comprueba que los datos cumplan con el formato definido por el servidor.
9.		El sistema modifica el reporte con los nuevos datos y refresca el listado de reportes, finalizando así la ejecución del caso de uso
Flujos alternos		

Nº 1 Sección modificar datos (Existen campos en blanco)		
	Actor	**Sistema**
7.1		El sistema muestra un mensaje diciendo al usuario que existen campos vacíos, volver al fuljo norma de eventos 5.
Nº 2 Sección modificar datos (Los datos no cumplen con el formato específico)		
	Actor	**Sistema**
8.1		El sistema muestra un mensaje diciendo al usuario los datos que no presentan el formato correcto, volver al fuljo norma de eventos 5
Flujo básico Sección Eliminar reporte (acción satisfactoria)		
1.		El sistema muestra un listado con los reportes existentes.
2.	El usuario selecciona un reporte.	
3.	El usuario presiona el botón **“Eliminar”**.	
4.		El sistema muestra un mensaje pidiendo al usuario que confirme la acción que va a realizar.
5.	El usuario presiona el botón **“Aceptar”**.	
6.		El sistema elimina el reporte y refresca el listado de finalizando así la ejecución del caso de uso.
Flujos alternos		

Nº 1 Sección Eliminar reporte (El usuario presiona el botón "Cancelar")		
	Actor	**Sistema**
5.1		El sistema cierra el formulario y se cancela la sección eliminar finalizando así el caso de uso.
Flujo básico Sección Modificar jrxml (acción satisfactoria)		
	Actor	**Sistema**
1		El sistema muestra un listado con los reportes pertenecientes al usuario autenticado.
2	El usuario selecciona un reporte y presiona el botón **"Modificar jrxml".**	
3		El sistema muestra un formulario en el cual pide al usuario el nuevo archivo jrxml y una descripción.
4	El usuario introduce los datos y presiona el botón **"Aceptar".**	
5		El sistema valida que no existan campos en blanco.
6		El sistema valida que los datos cumplan con el formato requerido.
7		El sistema modifica el reporte y refresca la lista de reportes finalizando así la ejecución del caso de uso.
Flujos alternos		
Nº 1 Sección Modificar jrxml (Los datos del reporte están incompletos)		
	Actor	**Sistema**
5.1		El sistema muestra un mensaje diciendo al usuario

		que los datos están incompletos.
Nº 2 Sección Modificar jrxml (Los datos no cumplen con el formato requerido)		
	Actor	**Sistema**
6.1		El sistema muestra un mensaje diciendo al usuario que los datos no cumplen con el formato requerido, volver al fuljo norma de eventos 4.

Flujo básico Sección Descargar (acción satisfactoria)		
	Actor	**Sistema**
1		El sistema muestra un listado con los reportes pertenecientes al usuario autenticado.
2	El usuario selecciona un reporte y presiona el botón **"Descargar".**	
3		El sistema muestra al usuario la ventana de descarga del navegador.
4	El usuario selecciona la opción descargar y presiona el botón **"Aceptar".**	
5		El sistema muestra al usuario la ventana definida por el navegador para buscar la ruta donde quiere descargar el fichero.
6	El usuario selecciona la ruta y presiona el botón **"Aceptar".**	
7		El sistema guarda el archivo jrxml en la dirección seleccionada por el usuario finalizando así el caso

		de uso.
Flujo básico Sección Exportar reportes (acción satisfactoria)		
	Actor	**Sistema**
1.		El sistema muestra un listado con los reportes pertenecientes al usuario autenticado
2.	El usuario selecciona el reporte.	
3.	El usuario presiona el botón **"Exportar"**.	
4.		El sistema busca los parámetros asociados al reporte y muestra la interfaz exportar.
5.	El usuario completa los parámetros requeridos.	
6.	El usuario presiona el botón **"Aceptar"**.	
7.		El sistema muestra el reporte en el formato especificado por el usuario en una nueva pestaña brindando la posibilidad de guardarlo finalizando así el caso de uso.
Flujos alternos		
Nº 1 Sección Exportar reportes (El usuario presiona el botón cancelar)		
	Actor	**Sistema**
6.1		El sistema cierra el formulario y se cancela la sección exportar finalizando así el caso de uso.
Flujo básico Sección Vista previa (acción satisfactoria)		
	Actor	**Sistema**

1	El usuario presiona el botón **“Vista previa”.**	
2		El sistema muestra la interfaz vista previa en la cual pide al usuario los datos (puerto, usuario, contraseña, nombre de la base de datos, formato, controlador, Ip y fichero jrxml.
3	El usuario completa los datos y presiona el botón **“Aceptar”**	
4		El sistema verifica que no existan campos en blanco.
5		El sistema valida que los datos cumplan con el formato requerido.
6		El sistema muestra en una pestaña del navegador la vista previa del reporte deseado finalizando así el caso de uso.
Flujos alternos		
Nº 1 Sección Vista previa (Existen campos en blanco)		
	Actor	**Sistema**
4.1		El sistema muestra un mensaje diciendo al usuario que existen campos en blanco.
Nº 2 Sección Vista previa (Existen datos con formato incorrecto)		
	Actor	**Sistema**
5.1		El sistema muestra un mensaje diciendo al usuario que existen datos con formato incorrecto.
Relaciones	**CU Incluidos**	No procede

	CU Extendidos	No procede
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	
Prototipo de la interfaz		

Fig 6. Prototipo de interfaz de usuario para el caso de uso Gestionar Reportes.

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. (11)

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. Éste contiene las definiciones de las entidades del software en vez de conceptos del mundo real. Para la elaboración del modelo del diseño fue elaborado un diagrama ya que el mismo se corresponde para todos los casos de uso del sistema. En las imágenes que se muestran a continuación se muestra el diagrama de clases del diseño.

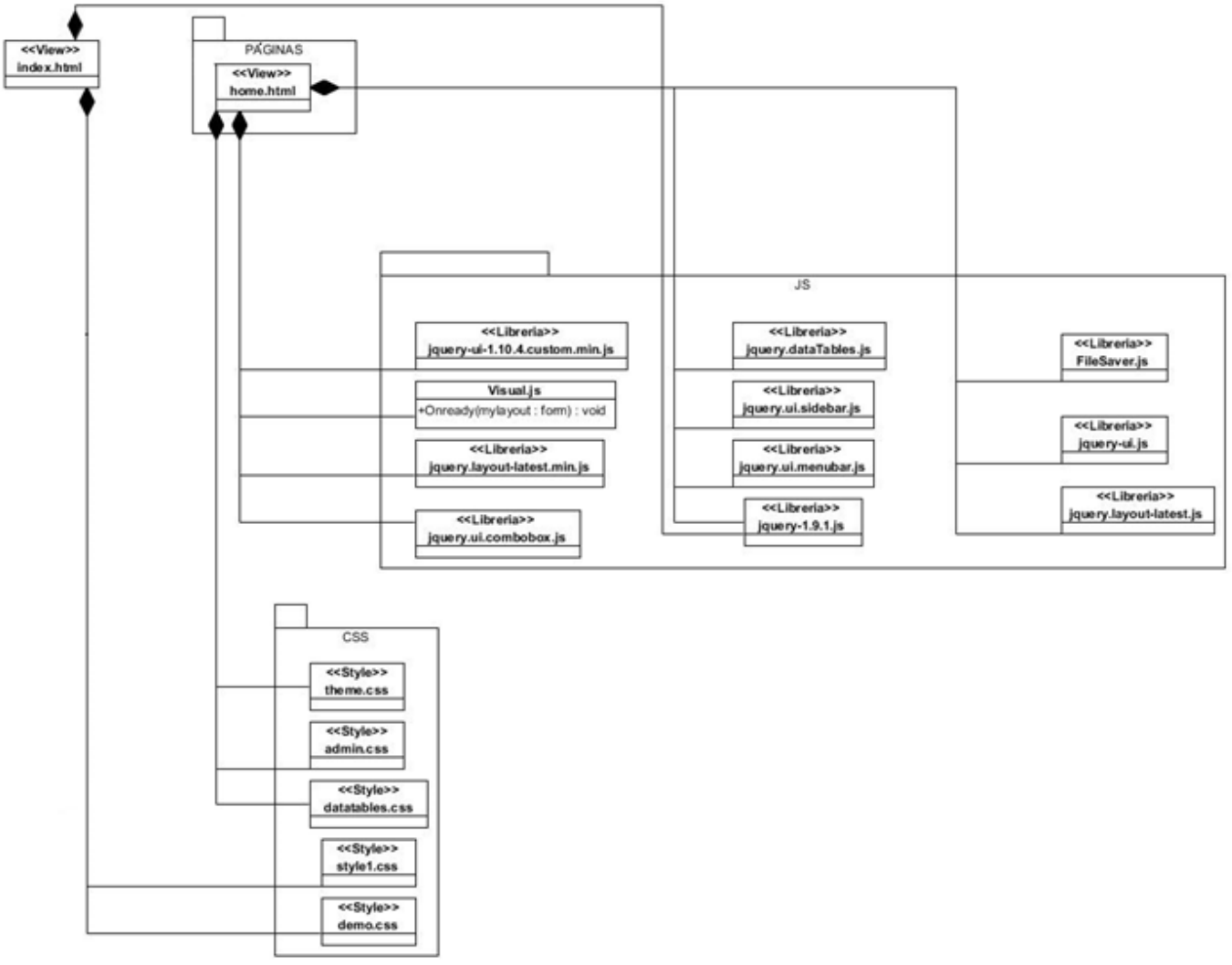

Fig 7. Relación entre las bibliotecas, estilos e interfaces.

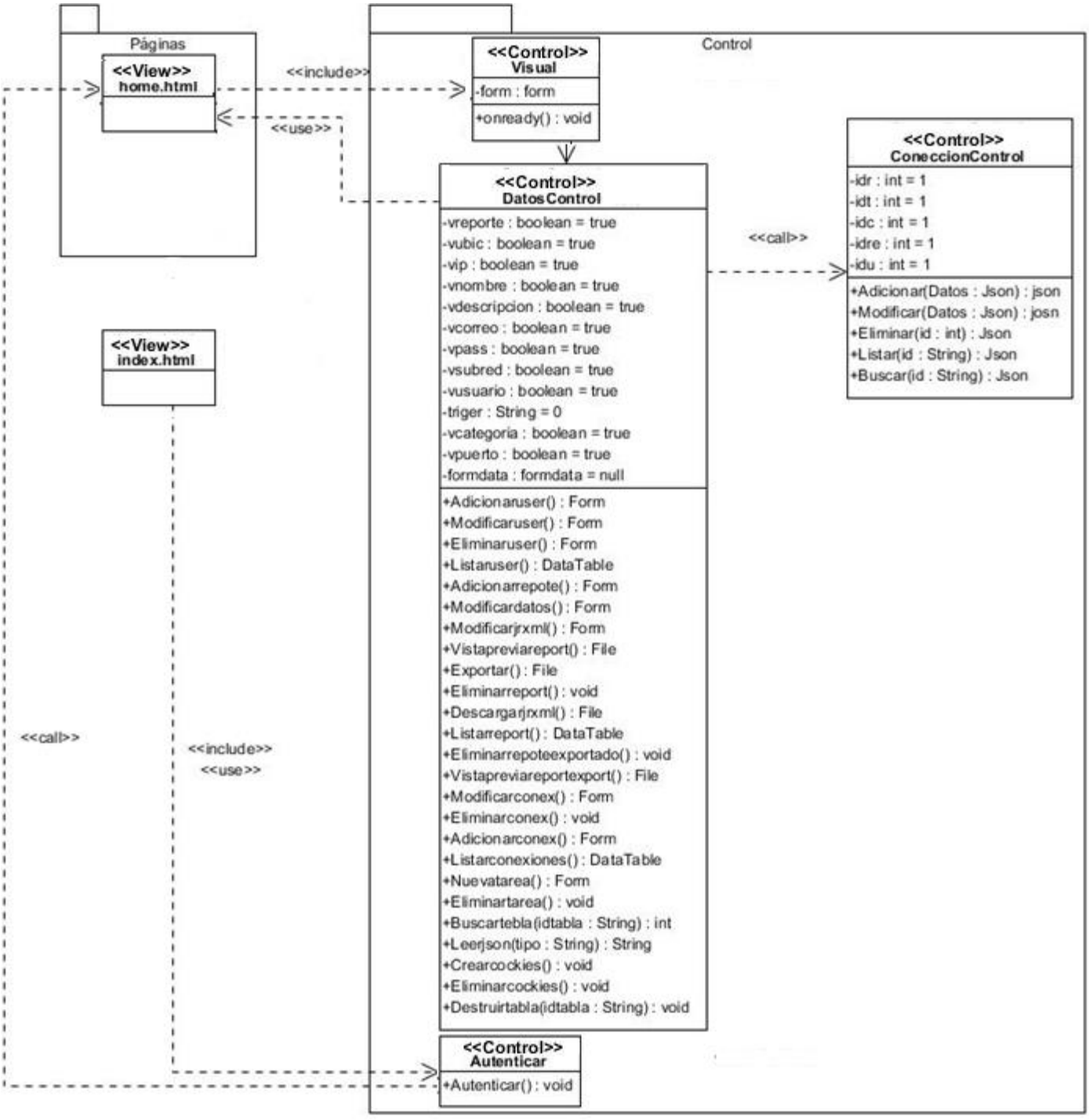

Fig 8. Relación entre las clases de la interfaz y las controladoras

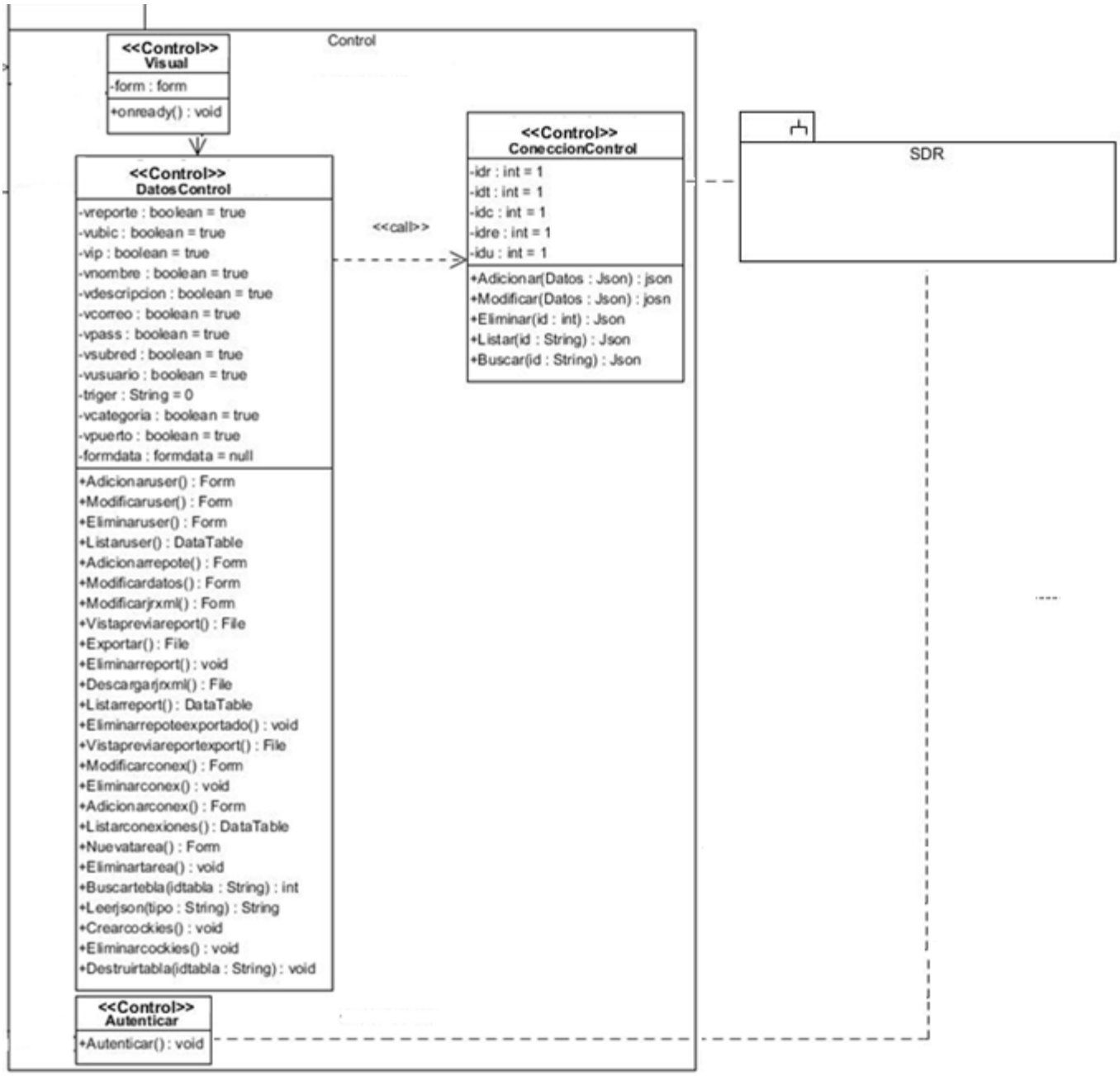

Fig 9. Relación de las clases controladoras con el servidor.

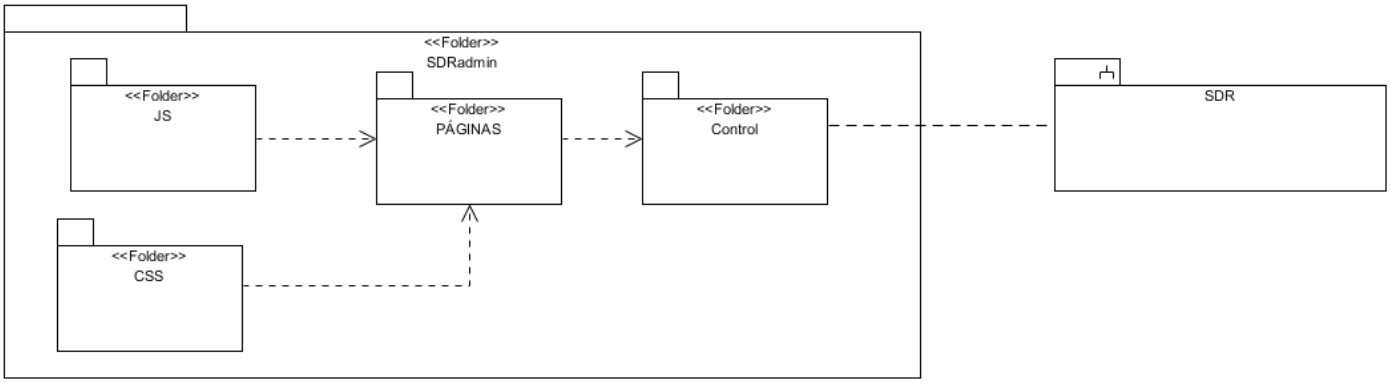

Fig 10. Diagrama de paquetes.

En la Fig 9 se muestra el diagrama de paquetes por los que se encuentra estructurado el sistema.

2.4.2. Descripción de las clases del diagrama de diseño

Tabla 5 Descripción de los elementos del diseño.

No	Clase	Tipo de clase	Descripción	Relación	Tipo de relación
1	**Autenticar.js**	Controladora	Clase que contiene los métodos necesarios para la autenticación del usuario en el servidor.	2,3,6,22	Asociación, Agregación
2	**Index.html**	Vista	Clase interfaz a través de la cual el usuario puede autenticarse en el sistema.	1,14, 19,20	Agregación, Composición
3	**Home.html**	Vista	Clase interfaz a través de la cual se muestran los resultados de la ejecución de los métodos.	1,4,5,7,8,9,10,11,12,13,14,15,16,17,18,21,22	Asociación, Agregación, Composición.
4	**DatosControl.js**	Controladora	Clase donde se encuentran todos los métodos encargados de ejecutar las peticiones del usuario.	3,5,6,22	Asociación, Agregación
5	**ConexionCon**	Controladora	Clase encargada de procesar las peticiones	4,3,22	Agregación

	trol.js		de la clase “DatosControl” y establecer la comunicación con el servidor.		
6	**jquery-ui-1.10.4.custom.min.js**	Biblioteca	Biblioteca para los componentes visuales.	3	Composición
7	**jquery.dataTables.js**	Biblioteca	Biblioteca para los componentes visuales de las tablas.	3	Composición
8	**Visual.js**	Biblioteca	Biblioteca para cargar el visual de la aplicación.	3	Composición
9	**jquery.ui.sidebar.js**	Biblioteca	Biblioteca para las barras deslizables.	3	Composición
10	**jquery.layout-latest.min.js**	Biblioteca	Biblioteca para los formularios.	3	Composición
11	**jquery.ui.menubar.js**	Biblioteca	Biblioteca para el menú.		
12	**jquery.ui.combobox.js**	Biblioteca	Biblioteca para los combobox.	3	Composición
13	**jquery-1.9.1.js**	Biblioteca	Biblioteca del framework de desarrollo.	3,2	Composición
14	**FileSaver.js**	Biblioteca	Biblioteca salvar ficheros.	3	Composición
15	**jquery.layout-latest.js**	Biblioteca	Bibliotecas para los formularios.	3	Composición
16	**jquery-ui.js**	Biblioteca	Bibliotecas para el visual completo de la aplicación.	3	Composición
17	**theme.css**	Biblioteca	Define los estilos visuales de la página home.	3	Composición
18	**demo.css**	Biblioteca	Define los estilos visuales de la página “índex”.	2	Composición

19	**style1.css**	Biblioteca	Define los estilos visuales de la página “índex”.	2	Composición
20	**datatables.css**	Biblioteca	Define los estilos visuales de la página “home”.	3	Composición
21	**admin.css**	Biblioteca	Define los estilos visuales de la página “home”.	3	
22	**SDR**	Subsistema	Servidor dinámico de reportes	5,1	

2.5. Patrones

Un patrón es un fragmento de información que captura la estructura esencial y la visión interna de una solución aplicada con éxito sobre un problema que surge dentro de un contexto. Permite reutilizar soluciones que funcionaron bien en la mayoría de los casos. En la programación orientada a objetos se entiende por patrón a una solución probada que se puede aplicar con éxito a un determinado tipo de problema que aparece repetidamente en el desarrollo de sistemas software. (11) Existen varios tipos de patrones entre los que se destacan los de arquitectura y diseño.

2.5.1. Patrón arquitectónico por capas

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 éstas. Permite además la ubicación de las capas en los niveles correspondientes. (12)

El diseño de la aplicación se basó en la arquitectura en N capas definiéndose tres niveles fundamentales teniendo en cuenta el proyecto SDR, sin embargo la solución se encuentra enmarcada solamente en el nivel de aplicación que a su vez está formado por la capa de presentación y la capa de lógica del negocio.

En la capa de presentación que se encarga de la comunicación con el usuario se encuentran las interfaces de la aplicación “índex” y “home” ambas con la extensión HTML.

En la capa de lógica del negocio se puede encontrar las clases “ConexionControl” encargada de la comunicación con el servidor y la clase “DatosControl” la que atiende todas las peticiones recibidas desde la vista.

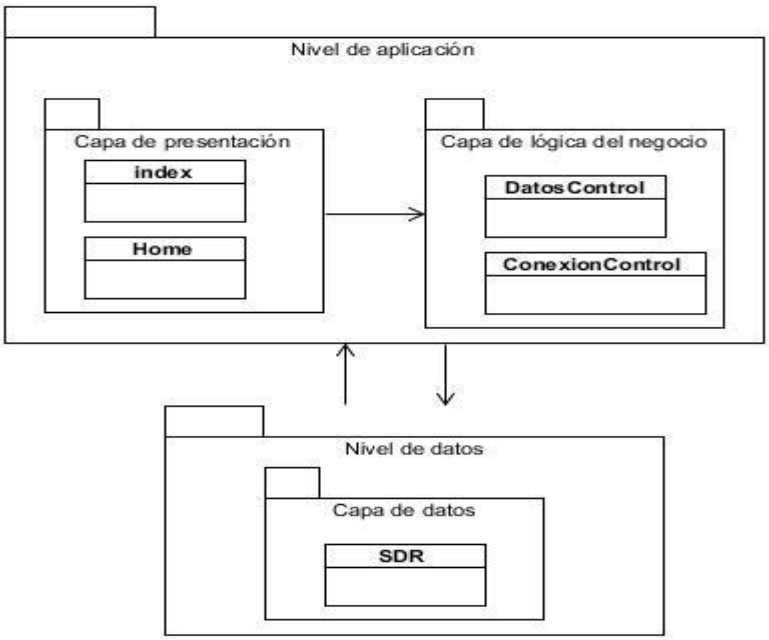

Fig 11. Arquitectura del sistema

2.5.2. 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.

Dentro de los patrones de diseño se puede encontrar el 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*). (13)

Patrones GoF aplicados a la solución

- **Instancia única**

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto. (13)

Este se evidencia a la hora de crear una instancia única de la clase control “ConexionControl” en la clase “DatosControl”, que es utilizada por todos los métodos de la clase que se necesita como se evidencia en la Fig 12.

```
$("#dialog-confirm7").dialog({
    resizable: false,
    autoOpen: false,
    width: 510,
    modal: true,
    buttons: {
        Aceptar: function() {

            var id = buscarch("'#example1");
            $.control.eliminar(id, leercookies("token"), "report/");
            $(this).dialog("close");
        },
        Cancel: function() {
            $(this).dialog("close");
        }
    }
});
```

Fig 12. Evidencia del patrón instancia única.

- **Mediator**

Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.

Se evidencia en la clase controladora DatosControl que se encarga de procesar los datos obtenidos de las clases interfaces y enviárselos a la clase ConexionControl encargada de la comunicación con el servidor.

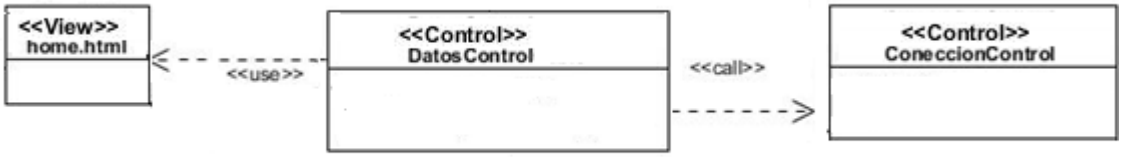

Fig 13. Evidencia del patrón Mediator.

- **Observador**

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

Se evidencia en la clase visual que se mantiene observando el cambio de estados de algunos objetos en las interfaces para notificarlo a la clase controladora “DatosControl”.

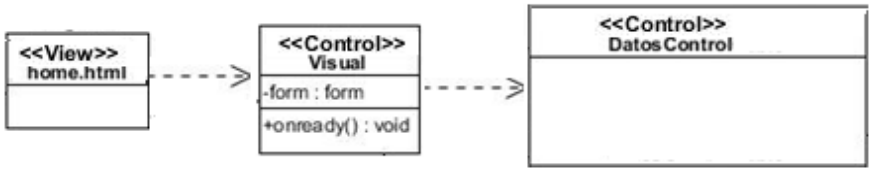

Fig 14. Evidencia del patrón observador.

Patrones GRASP utilizados

- **Experto:**

Se evidencia al asignar las responsabilidades necesarias a una sola clase. En la solución propuesta la clase controladora “ConexiónControl” y “DatosControl” siendo estas las encargadas de la comunicación con el servidor y el manejo de los datos respectivamente.

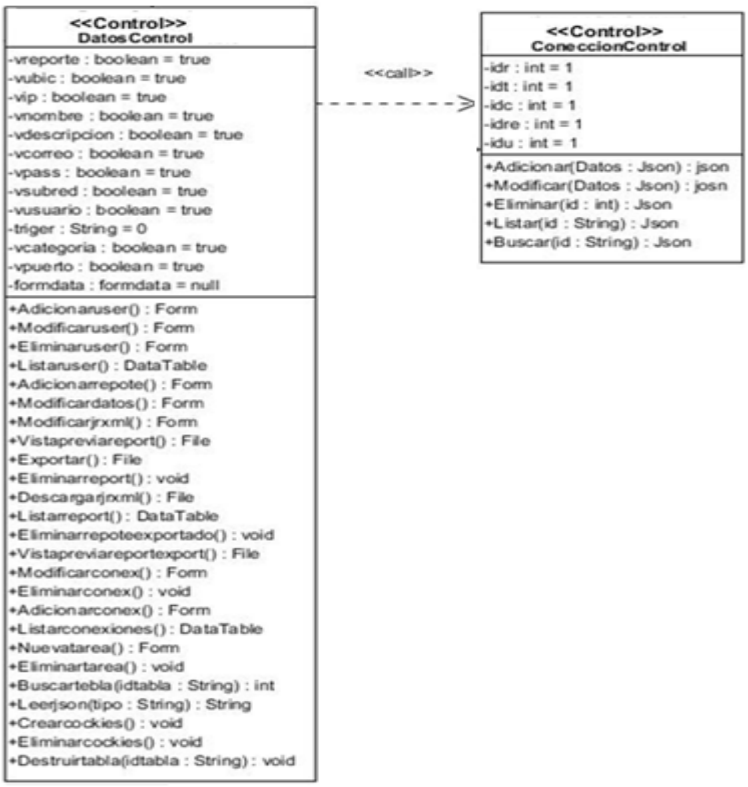

Fig 15. Evidencia del patrón Experto.

Controlador:

La estructura que se propone en la arquitectura evidencia el uso de este patrón definiendo clases controladoras para el manejo de los datos y la conexión con el servidor. Cada una de estas clases posee responsabilidades específicas de controlar el flujo de eventos del sistema.

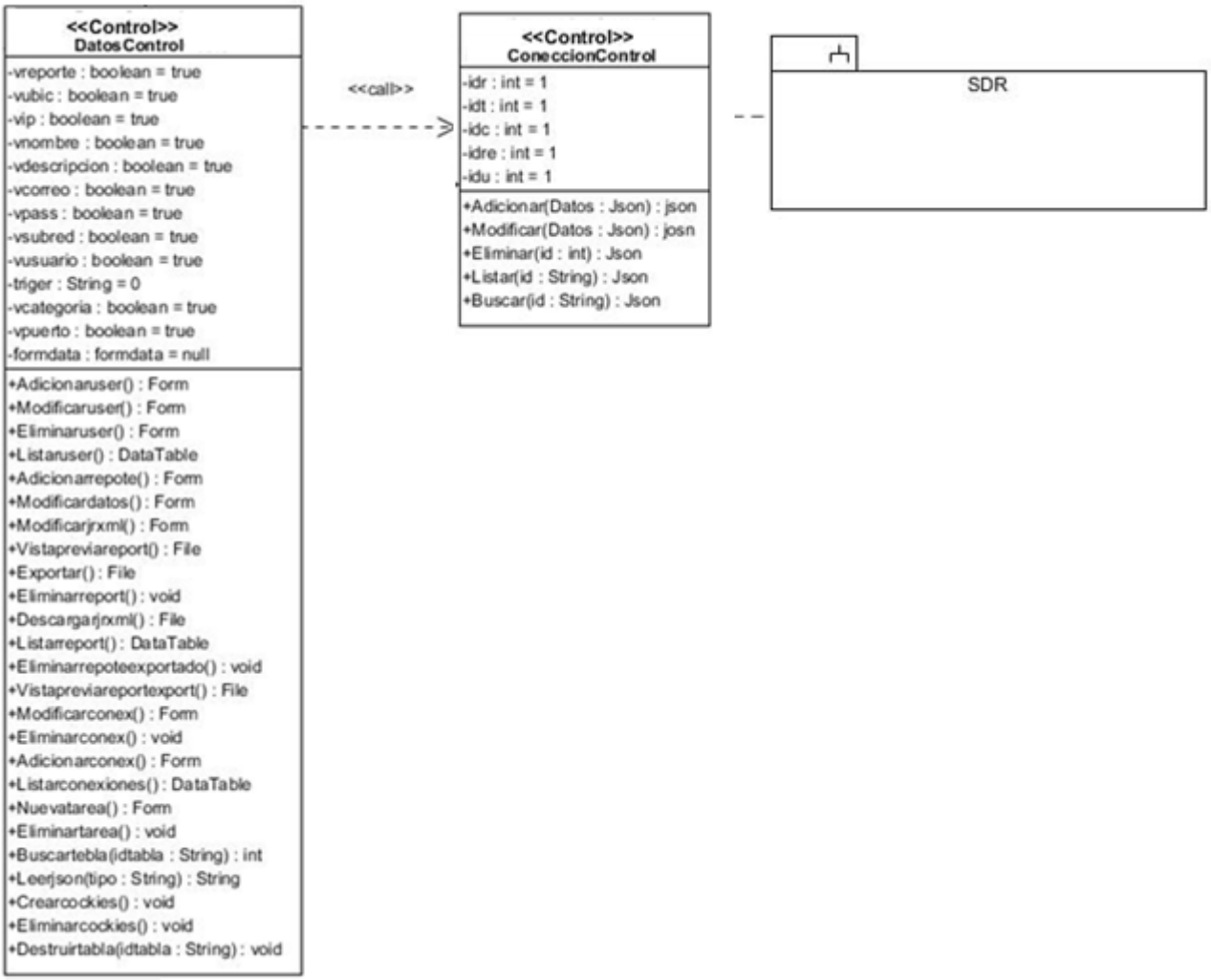

Fig 16. Evidencia del patrón Controlador.

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 muestra como un usuario se conecta a través de una estación de trabajo por el protocolo HTTPS con el puerto 80 al entorno de ejecución donde se encuentra el servidor de aplicaciones, el cual incluye dentro de sus servidores *Tomcat* 7.2 donde se encuentran desplegados el

SDR y el SDRAmin; por otro lado el servidor web se comunica con el servidor de base de datos por el protocolo TCP/IP con el puerto 5432 encontrándose en este la base de datos del SDR.

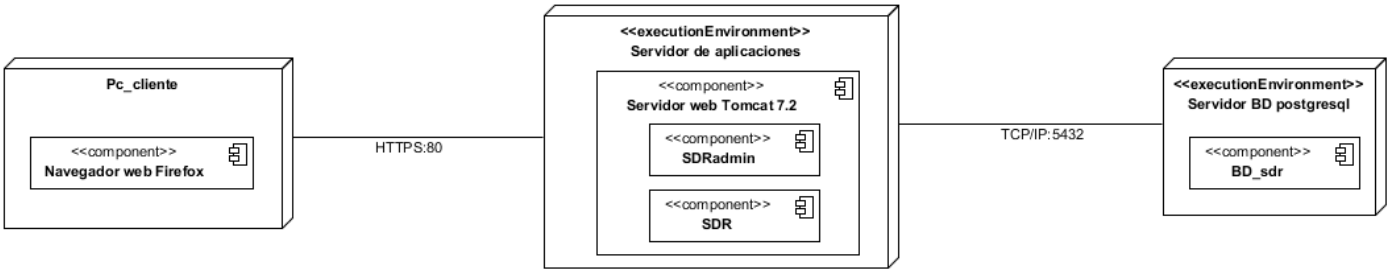

Fig 17. Diagrama de despliegue de la solución.

Conclusiones parciales

Durante las etapas de análisis y diseño de la aplicación fueron identificados veintiocho requisitos funcionales quedando agrupados en seis casos de uso de acuerdo a los patrones definidos y seis requisitos no funcionales necesarios para que la aplicación para cumplir con las necesidades del cliente. Se realizó el modelo de dominio el cual facilitó la comprensión del sistema para la administración visual del SDR. Se detalló el patrón arquitectónico 3 capas que permitió separar conceptos mejorando el desarrollo estructurado del sistema. Se elaboró el diagrama de despliegue el cual brinda una visión de cómo quedarán distribuidos los diferentes nodos.

CAPÍTULO 3: Implementación y prueba de la solución

En el presente capítulo se presentan los diagramas de componentes de los casos de uso arquitectónicamente significativos que conforman el modelo de implementación. Se definen los métodos y estrategias de pruebas que se aplican en el proceso de pruebas para validar el funcionamiento de la herramienta de administración del SDR.

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. (10)

Fundamentalmente, se muestra la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes.

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 está 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.

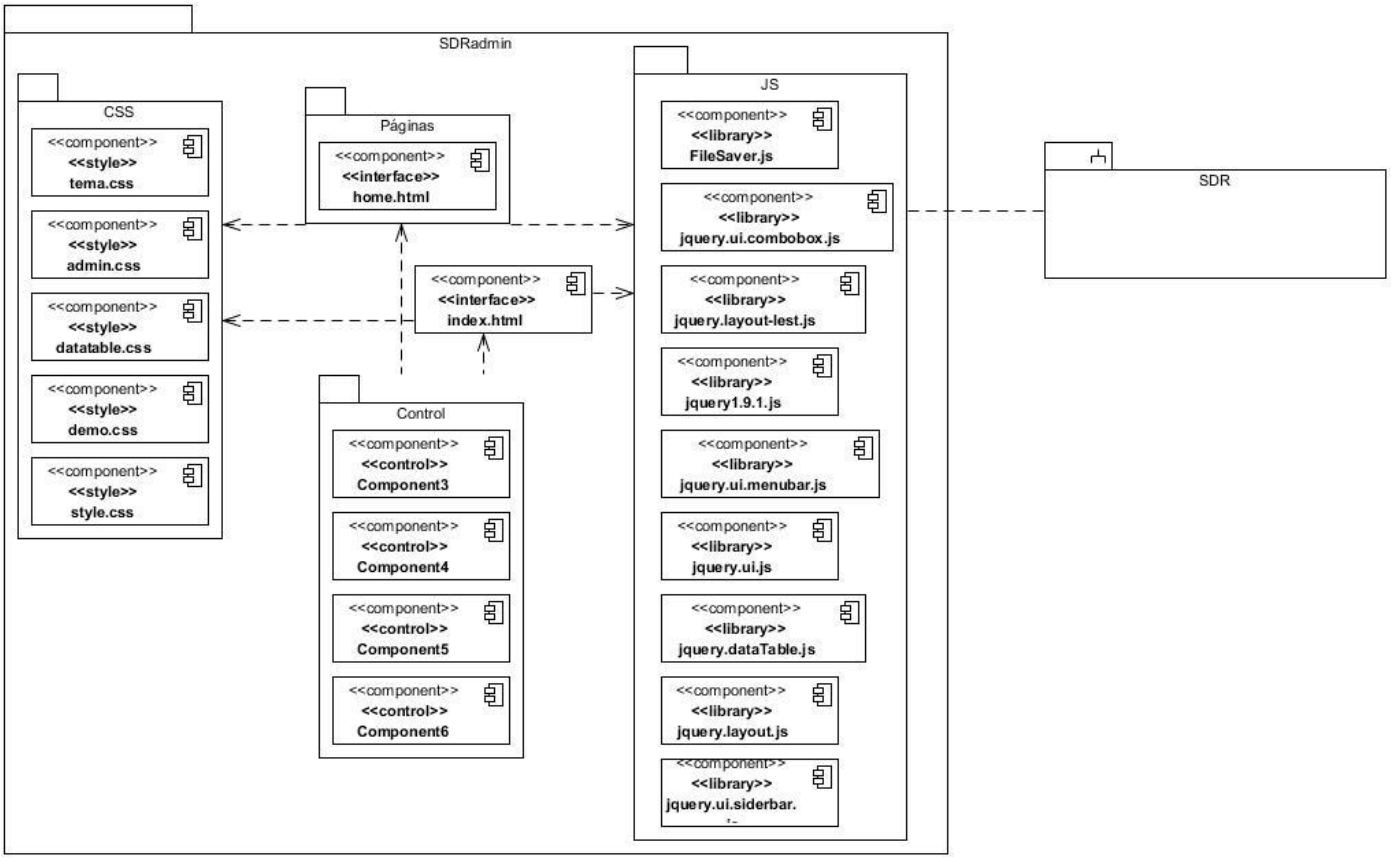

Fig 18. Diagrama de componentes

El diagrama de componentes en la Fig 8, pertenece a todos los casos de uso de la aplicación ya que los mismos hacen utilización de todos los componentes representados.

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.

3.2.1. 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 como se muestra en la Fig 19:

```
$("#mensage").append("<div id='msg' class='msg-error msg' ><div class='msg-content .msg-content-error' >,
Existen campos en blanco o con datos no validos</div></div>")
$("#msg").slideDown("slow");
var  timer = setTimeout(function() {$("#msg").slideUp()}, 3000);
document.getElementById("Passworpv").style.borderColor = "red";
```

Fig 19. Tamaño y organización de las líneas de código.

3.2.2. Declaraciones

Las variables locales se inicializarán en el momento de su creación salvo que su valor inicial dependa de un valor calculado posteriormente, éstas deben situarse al principio de cada bloque principal en el que se utilicen y no en el momento de su uso como se muestra en la Fig 20:

```
var vcorreo=false;
var vpass=false;
var vsubred=false;
var vip=false;
var vusuario=false;
var vnombre=false;
```

Fig 20. Evidencia de las declaraciones.

3.2.3. Métodos

Los métodos se separarán por una línea en blanco. Se pondrán las declaraciones al principio de los bloques evidenciándose en la Fig 21:

```
function crearcookie(key, value) {
    expires = new Date();
    expires.setTime(expires.getTime() + 31536000000);
    cookie = key + "=" + value + ";expires=" + expires.toUTCString();
    return document.cookie = cookie;
}

function leercookies(key) {
    keyValue = document.cookie.match("(^|;) ?" + key + "=([^;]*)(;|$)");
    if (keyValue) {
        return keyValue[2];
    } else {
        return null;
    }
}
```

Fig 21. Evidencia de los métodos.

3.2.4. Convenciones de nombres para las variables

Los nombres de las variables comenzarán con minúscula, en caso de ser compuestas se escribirán juntos como se muestra en la Fig 22:

```
var urlcon = leerJson('report/preview ', 'url');
var nombrebd = document.getElementById("namepv").value;
var usuario = document.getElementById("userpv").value;
var password = document.getElementById("Passworpv").value;
var controlador = document.getElementById("skillspv").value;
var puerto = document.getElementById("puertopv").value;
```

Fig 22. Evidencia de los nombres de las variables.

3.2.5. Convenciones de nombres para los métodos

Los métodos comenzarán con letra inicial minúscula, en caso de ser compuestos se escribirán juntos evidenciando el uso de este patrón en la Fig 23:

```
function destruirtablas(id, id2) {

    $('#' + id).dataTable({
        "bFilter": false,
        bPaginate: false,
        "bDestroy": true,
    });
    $('#' + id).remove();
    $.control.buscarTodos(leercookies("token"), id2);
    $("#msg").remove();
    $("#mensage").append("<div id='msg' class='msg-success msg' >,
    <div class='msg-content .msg-content-success' >Base de datos actualizada.</div></div>")
    $("#msg").slideDown();
    var  timer = setTimeout(function() {$("#msg").slideUp()},1000);
}
```

Fig 23. Evidencia de los nombres de los métodos.

3.2.6. Convenciones de nombres para las clases

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

3.2.7. 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.

3.2.8. 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 ni anidadas.

Paréntesis: Se deben utilizar paréntesis en expresiones que incluyan distintos tipos de operadores para evitar problemas de precedencia de operadores.

3.3. Interfaces Principales de la Aplicación.

Para el diseño del sistema se tuvo en cuenta un diseño sencillo y fácil de entender para mejorar la comprensión de los usuarios como se muestran en las Fig.24, 25, 26:

Fig 24 Interfaz de autenticación.

En la Fig 24 se presenta la interfaz de autenticación, a través de la cual un usuario registrado en el SDR tiene acceso al sistema de administración.

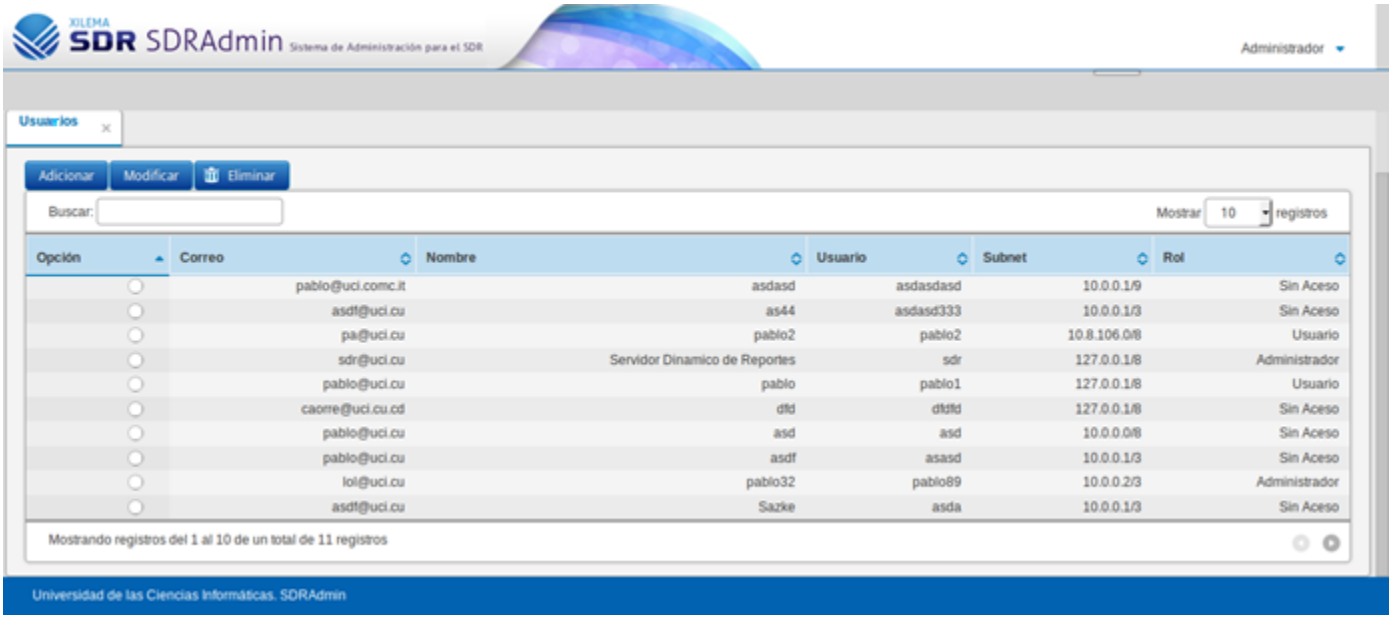

Fig 25. Interfaz de administración para el rol administrador.

La Fig 25 Muestra la interfaz a través de la cual el administrador puede gestionar los usuarios que tendrán acceso al sistema de administración.

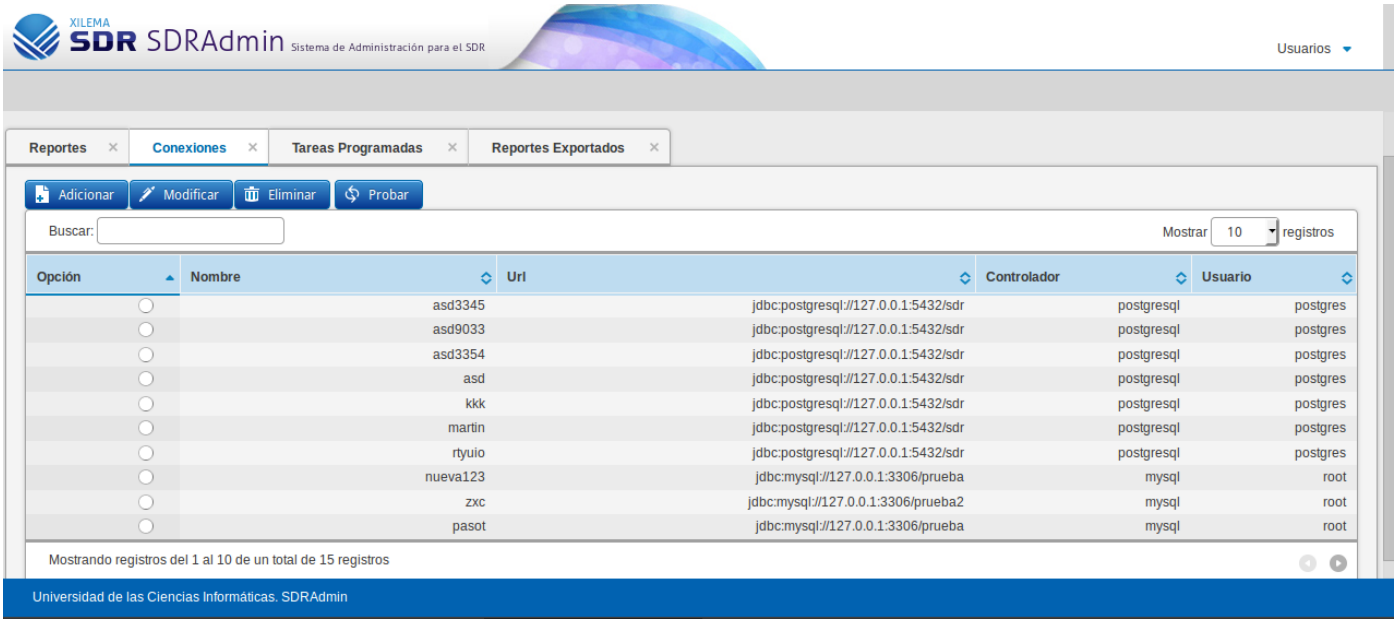

Fig 26. Interfaz de administración para el rol cliente.

En la Fig 26 se muestra la interfaz para el cliente el a través de la cual puede gestionar toda la información perteneciente a los reportes, tareas programadas, conexiones y reportes exportados.

3.4. Pruebas de software

Las pruebas se realizan mediante el proceso de ejecución de un programa con la intención de detectar errores, tienen éxito si detectan un error no identificado. Esta etapa implica: verificar la integración adecuada de los componentes así como su interacción, que todos los requisitos se hayan implementado correctamente, e identificar y asegurar que los defectos encontrados sean corregidos antes de entregar el software al cliente.

Cada uno de los niveles de prueba engloba una técnica específica la cual define los puntos de calidad que se van a verificar por cada tipo de prueba a realizar. Para la realización de las pruebas a la aplicación desarrollada se utilizaron las pruebas a nivel de desarrollador, la cual se diseña e implementa por el equipo de desarrollo durante el proceso de implementación de un software.

La técnica seleccionada es la de prueba funcional que tiene como objetivo verificar que todos los requisitos funcionales definidos funcionen adecuadamente, esto incluye la entrada de datos y su validación, así como su procesamiento y obtención del resultado esperado. La aplicación de esta técnica se realizará a través del método de prueba de caja negra.

3.4.1. Pruebas de caja negra

El método de caja negra o pruebas de comportamiento, se ejecuta por cada caso de uso haciendo uso de datos válidos y no válidos, donde se verifica que los resultados obtenidos se correspondan con los mensajes de error o aceptación respectivamente. Dichas pruebas se aplicarán sobre las interfaces de la aplicación, siendo indiferente el comportamiento interno y la estructura de la programación. Los casos de pruebas constituyen un conjunto de requisitos de ejecución, elaborados para una función en particular, con el objetivo de evaluar el estado de funcionalidad en el que se encuentra el software.

Para la elaboración de los casos de prueba se utilizó la técnica de Partición de Equivalencia esta técnica plantea la división de los campos de entrada de la aplicación en variables con distintos juegos de datos que tienden a evaluar las funcionalidades del software. En esencia, mediante la técnica se dividen las entradas en un número finito de variables de equivalencia, de modo que se pueda asumir que una prueba realizada con un valor de cada variable es equivalente a cualquier otro valor de la misma variable, representando un conjunto de estados válidos y no válidos, para las entradas de un programa.

3.5. Diseño de los casos de prueba

Los casos de prueba se elaboran según las funcionalidades que se describen en los casos de uso. Tienen como propósito principal lograr una comprensión de las especificaciones de los datos insertados y la respuesta esperada, cada plantilla responde a un caso de uso específico, la cual está dividida por escenarios, definiendo las funcionalidades y describiendo cada una de las variables que se recogen en el caso de uso.

Tabla 6. Diseño del caso de prueba de la sección adicionar del caso de uso Gestionar reportes.

Escenario	Descripción	categoría	descripción	nombre	archivo	parámetros	Respuesta del sistema
Adicionar Reporte acción satisfactoria	En este escenario el usuario inserta todos los datos y variables de forma correcta.	**N/A**	**V** “Listado de la UJC del departamento”	**V** “Nombre de los usuarios”.	**V** Listado usuario UJC.jrxml	**N/A**	Mensaje de confirmación "Datos guardados"
EC 1.2 Adicionar duplicación de nombres y fichero.	En este escenario el usuario inserta todos los datos con un nombre del reporte o archivo jrxml que ya existen en el sistema.	**N/A**	**V** “Listado de la UJC del departamento”	**V** “Nombre de los usuarios”	**V** Listado .jrxm	**N/A**	Mensaje de error “El reporte ya existe"
EC 1.4	En este	N/A	V	V	I	N/A	Mensaje de error

Adicionar Reporte Jrxml Incorrecto	escenario el usuario inserta todos los datos que no presentan una estructura correcta.		“Listado de la UJC del departamento”.	“Nombres de los usuarios”	Listado incorrecto.jrxml		“El fichero no presenta el formato requerido"
EC 1.5 Adicionar Reporte Nombre Vacío	Escenario donde se omite el nombre del reporte.	N/A	V “Listado de la UJC del departamento”.	I	V Listado .jrxml	N/A	Mensaje de error “Inserte el nombre"

3.5.1. Aplicación de las pruebas

Se realizaron un total de cuatro iteraciones de pruebas funcionales aplicando siete casos de prueba que arrojaron un total de treinta No Conformidades (NC). En la primera iteración se encontraron un total de 12 NC, quedando solucionadas en la segunda iteración donde fueron detectadas de 10 NC, de las cuales se solucionaron 6, en la tercera iteración se detectaron 8 NC, quedando todas resueltas en una cuarta iteración donde no se encontraron no conformidades.

Fig 27. Gráfica de no conformidades.

3.6. Evaluación del cumplimento del problema de la investigación

Al verificarse el funcionamiento de la aplicación acorde a los requisitos funcionales mediante las pruebas realizadas, se procede a comprobar el cumplimiento del problema de la investigación, mediante una comparación entre la administración del SDR a través de consola y la solución propuesta.

3.6.1. Comparación de las formas de administración

Consola administración

La administración de SDR se realizaba por una interfaz de consola utilizando ficheros con extensión .sh, donde el usuario que lo ejecuta debía tener conocimientos de curl[3]. Además de conocer el token de seguridad enviado por el servidor a la hora de autenticarse e ingresarlo en cada momento en que se le solicite. Por otra parte los datos insertados por el usuario tendrían que cumplir con un formato definido por el servidor que el usuario podría desconocer, provocando una demora en la ejecución correcta de un servicio web al servidor, es decir el usuario al recibir la respuesta del servidor con los errores cometidos debe identificarlos para luego corregirlo y volver a realizar la petición, aumentando la carga de trabajo del servidor.

[3] **curl:** es una herramienta software para transferencia de archivos con sintaxis URL mediante intérprete de comandos.

En la figura que se muestran a continuación se puede observar el código de un fichero con extensión .sh a través del cual se inserta un nuevo reporte en el servidor:

```
read value < /tmp/token.sdr
echo "Entre el token de autenticidad o presione ENTER para usar este ($value): "
read token
if [ $token ]; then
   echo $token > /tmp/token.sdr
else
   token=$value
fi

echo "Entre el formato de exportación o presione ENTER para usar este (pdf): "
read format
if ! [ $format ]; then
   format=pdf
fi

curl -i -X POST http://localhost:8084/sdr/services/rest/report/preview \
    -H 'X-SDR-API-Key: '$token \
    -H 'Accept:application/json' \
    -F file=@/media/07135ade-8fd6-4543-97c4-
d5756dc9b763/CasosDePrueba_Liberación_SDRv1.0/CU_GestionarReporte/SC6
_VistaPreviaReporte/Listado_Fuentes_Datos.jrxml \
    -F connection='{"password":"postgres","url":"jdbc:postgresql://127.0.0.1:5432/sdr","username":"postgres"}'\
    -F params="[{\"name\":\"IDS\",\"value\":\"2013\",\"type\":\"String\"}]"\
    -F format="$format"

echo ""
```

Fig 28. Código de un fichero, utilizado para obtener la vista previa de un reporte.

Sistema implementado SDRadmin

La segunda forma de administración se realiza mediante la interfaz del sistema implementado, en el cual un usuario no necesita conocer la forma en la que se comunica con el servidor, el token enviado en el momento de la autenticación le es totalmente indiferente, los datos insertados por el usuario son validados automáticamente. La interfaz es sugerente en cuanto a las distintas operaciones que se pueden realizar sobre los datos permitiendo a un usuario con conocimientos básicos en informática interactuar de forma fluida, los resultados de las peticiones al servidor son interpretados por la interfaz mostrándolos de una forma fácil de entender. Por último, cada error cometido por el usuario que no se valida en la aplicación, se muestra en forma de mensajes de error después de recibir la respuesta del servidor, brindando en detalle los campos en los cuales puede haber errores, los usuarios accederían directamente a la información y funcionalidades a los cuales tiene acceso en el servidor.

En la imagen que se presentan a continuación se muestran las interfaces para la gestión de reportes por el lado de los usuarios.

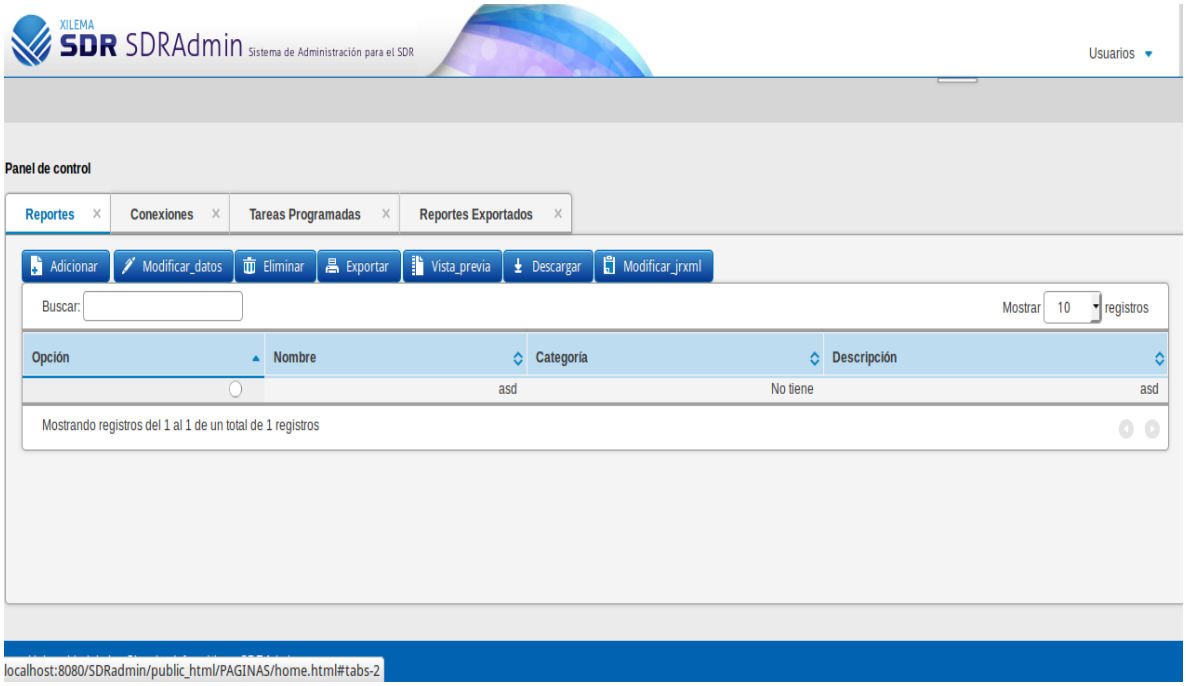

Fig 29. Interfaz para el usuario para la gestión reportes, conexiones, tareas programadas, conexiones con las fuentes de datos y reportes exportados.

En la Fig 29 se muestra la interfaz del cliente a través de la cual se pueden gestionar los reportes, conexiones, tareas programadas y los reportes exportados.

Variables y fórmulas para la evaluación

Luego de realizar la comparación de las herramientas existentes para la administración del servidor y basándose en las características mencionadas de cada una de ellas se seleccionan las siguientes variables y rangos para la puntuación, reflejadas en la siguiente tabla:

Tabla 7. Variables y rangos de calificación

Variables a comprobar	Rango de calificación
Tiempo de administración.	0-5
Facilidad de uso.	0-5
Fluidez de trabajo.	0-5
Esfuerzo realizado.	0-5

Tabla 8. Forma de puntuación de las variables

Variables	Interpretación
Tiempo de administración. (Ta)	La puntuación 5 para un menor tiempo y 0 el mínimo para un tiempo mayor en la administración del SDR.
Facilidad de uso.(Fu)	La puntuación 5 para una mayor facilidad y 0 el mínimo para la dificultad en la administración del SDR.
Fluidez de trabajo. (Ft)	La puntuación 5 para la forma en que el usuario utiliza el SDR siguiendo un conjunto de pasos lógicos y 0 el mínimo para la falta total de pasos lógicos que guíen al usuario.
Esfuerzo requerido.(Er)	La puntuación 5 para el esfuerzo mínimo y 0 la necesidad de un esfuerzo elevado para la administración del SDR.

La evaluación de la administración se realizará a través de las siguientes fórmulas $k = \sum_{x=1}^{x=n} S/E \;\; x = \sum_{x=1}^{x=n} K/m \;\; p = x/\mathrm{E}$. Siendo *p* la facilidad de administración, *k* el resultado de la sumatoria de valor dado por los expertos a cada una de las variables entre la cantidad de expertos, s el valor de cada variable dado por los expertos, *m* la cantidad de variables a evaluar, x la sumatoria final de k entre m y *E* la cantidad de expertos involucrados en la evaluación.

El rango de evaluación para la facilidad de administración (*p*) será de 0 a 1 siendo cero la puntuación más baja y 1 la máxima puntuación en cuanto a la facilidad de administración del servidor.

Selección de los expertos

Se seleccionaron un grupo de cinco expertos del departamento de Desarrollo de Aplicaciones perteneciente al Centro de Tecnología y Gestión de Datos. La selección se realizó a personas interesadas a participar en la prueba teniendo en cuenta los aspectos que se enuncian a continuación:

- Graduado de nivel superior.
- Un año de experiencia como mínimo.
- Vinculación al desarrollo de productos informáticos.
- Vinculados al proyecto SDR

Análisis de los resultados

Las pruebas fueron aplicadas a los expertos haciendo uso de todas las funcionalidades del sistema y la evaluación a cada variable definida, aplicándose a cada una de las formas de administración (ver Fig 20 y Fig 21). A partir de los gráficos se obtuvo el promedio para cada variable y se calculó la facilidad administración del SDR a través de la fórmula definida. (ver Tabla 9 y10).

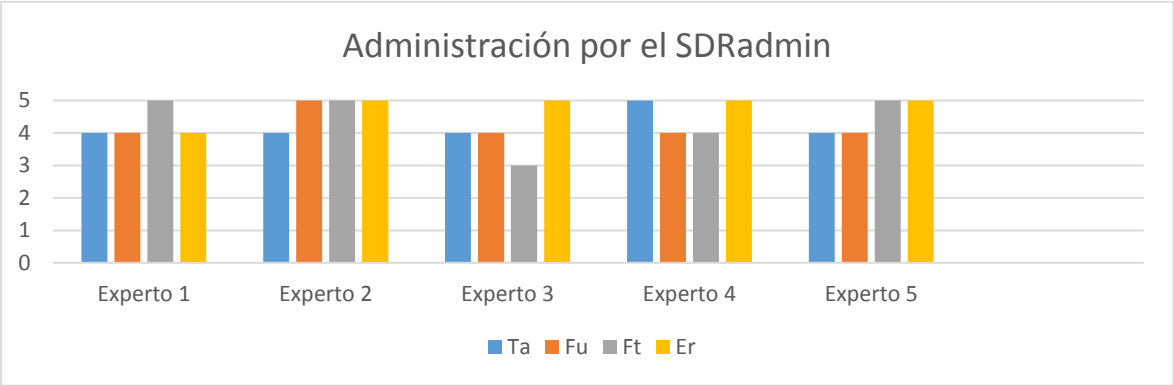

Fig 30. Tiempo de las variables por cada experto para SDRadmin.

Aplicando la formula $k = \sum_{x=1}^{x=n} S/E$ por cada variable.

Tabla 9 Resultados promedio del tiempo de las variables.

Variables.	Resultado.
Ta	4
Fu	4
Ft	4
Er	5

El resultado reflejado (k_i) es el tiempo promedio de cada una de las variables de acuerdo a las puntuaciones dadas por los expertos para el SDRadmin.

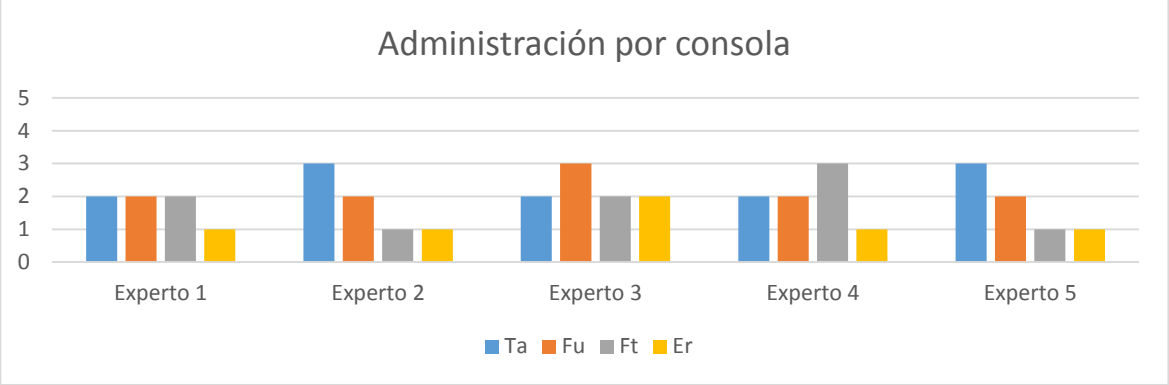

Fig 31. Tiempo de las variables por cada experto para la consola.

Aplicando la formula $k = \sum_{x=1}^{x=5} S/E$ por cada variable.

Tabla 10. Resultados promedio del tiempo de las variables.

Variables	Resultado
Ta	2
Fu	2
Ft	2
Er	1

El resultado reflejado (k_j) en la tabla es el tiempo promedio de cada una de las variables de acuerdo a las puntuaciones dadas por los expertos para la consola.

Aplicando la fórmula

Facilidad de administración para la consola

$x = \sum_{x=1}^{x=4} K/m$

$m = 4 \quad \sum_{x=1}^{x=4} k = 2 + 2 + 2 + 1 = 7$

$x = 7/4 = 1{,}75$

$p = (x/E) \quad = 1{,}75/5 \ = 0{,}35$

Según el resultado obtenido la evaluación de la consola es de un 0.35

Facilidad de administración para el SDRadmin

$x = \sum_{x=1}^{x=4} K/m$

$m = 4 \quad \sum_{x=1}^{x=4} K = 4 + 4 + 4 + 5 = 17$

$x = 17/4 = 4{,}25$

$p = (x/E) = 4{,}25/5 \ = 0{,}85$

Según el resultado obtenido la evaluación del SDRadmin es de un 0.85.

Análisis de los resultados

Luego de realizar el cálculo de los promedios, que evalúan la facilidad de administración del SDR, se obtuvo un valor de 0.85 para el SDRadmin y 0.35 por consola con una diferencia de 0.5, considerándose significativa en la administración del SDR ver (Fig 22). Con estos resultados queda demostrado que el Sistema para la Administración Visual del SDR mejora la administración del SDR, respondiendo así al problema planteado.

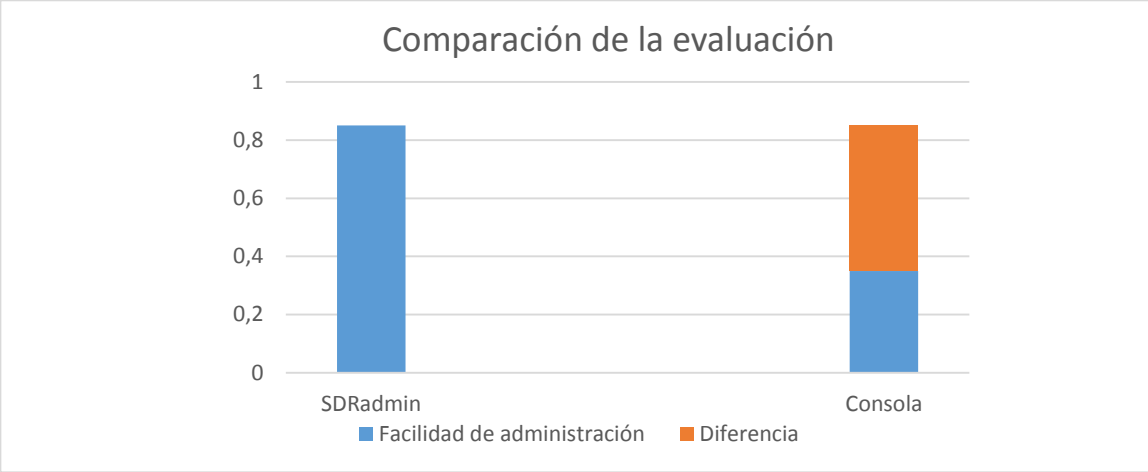

Fig 32. Comparación de las evaluaciones.

Conclusiones parciales

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 del diagrama de componentes del sistema, diseñado a nivel representativo con UML, donde se visualiza la interacción entre los componentes. La realización de las pruebas de software empleando el método de Caja Negra con la técnica de Partición de Equivalencia, permitió detectar varias treinta NC en las primeras tres iteraciones, arribando a la cuarta iteración donde quedaron solucionadas satisfactoriamente.

Luego de realizadas las pruebas, se pasó a verificar el cumplimiento del problema de la investigación, obteniendo como resultado una diferencia de 0.5 de puntuación para la facilidad de administración del SDR, a través del SDRadmin, quedando demostrado que el sistema implementado mejora la administración del SDR.

Conclusiones generales

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, permitió corroborar que la herramienta para la administración implementada, al contar con una interfaz intuitiva y con un bajo grado de complejidad, le permite al usuario final realizar operaciones de forma fluida facilitando su aprendizaje.
- El análisis y diseño de los principales conceptos relacionados con el sistema para la administración del SDR, permitió obtener los artefactos y los diagramas de casos de uso, despliegue, clases del diseño, arquitectura y componentes que se utilizaron para guiar el desarrollo de la aplicación.
- La implementación y prueba de la herramienta de administración dio cumplimiento a los veintiocho requisitos funcionales identificados para facilitar la administración del SDR.
- La comparación y evaluación de las herramientas para la administración del SDR, permitió demostrar que el sistema implementado mejora el tiempo de administración, facilidad de uso, fluidez de trabajo y esfuerzo requerido por los especialistas.

Recomendaciones

Implementar un módulo que permita la consulta a las trazas del sistema así como proporcionar estadísticas del uso de las cuentas de usuario y conexiones con las bases de datos.

Bibliografía

1. **Cauldwell, Patric.** *SERVICIOS WEB XML: PROFESIONAL.* 2002.

2. **Erl, Thomas.** *Service - Oriented Arquitecture: Concepts, Technology and Design.* 2005.

3. **Dallas, Alexandros.** *Restful web services with dropwizard.* 2014.

4. **Gimson, Loraine.** *Metodologías ágiles y desarrollo basado en conocimiento.* 2012.

5. **Paradig.ogr, Virsuar.** *Manual de Visual Paradigm.* 2010.

6. **jimenez, Enrique Gomez.** *Desarrollo de software con netbeans.* 2012.

7. **Duckett, Jon.** *JavaScript and Jquery: Interactive Front-End Web Development.* 2014.

8. **Alvarez, Miguel Angel.** *Manual de jquery.* 2010.

9. **Darwin, Ian F.** *Tomcat La guia definitiva.* 2008.

10. **Larman, Craig.** *Uml y patrones.* 2002.

11. **Sommerville, Ian.** *Software Engineering, 8th ed.* 2006.

12. **César de la Torre Llorente, Unai Zorrilla Castro, Javier Calvarro Nelson, Miguel Ángel Ramos Barros.** *Guía de Arquitectura N capas.* 2010.

13. **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2005.

14. **Raphaël Hertzog, Roland Mas.** *The debian administrator handbook.* 2013.

15. **Pressman, Roger.** *Ingeniería de Software un enfoque práctico 6ta edición.* 2005.

16. **Gamma, Erich Et al.** *PATRONES DE DISEÑO: ELEMENTOS DE SOFTWARE ORIENTADO A OBJETOS REU TILIZABLES.* 2002.

17. **Delsile, Marc.** *Dominar phpMyAdmin para una administración efectiva de MySQL.* 2007.