Universidad de las Ciencias Informáticas

Facultad 6

Sistema de Administración del Servidor Dinámico de Reportes (SDR-GUI)

Autores: *Carlos Osvaldo Chao Cortes*

Arian Cruz Suárez

Tutores: *Msc. Yadira Robles Aranda*

Ing. Yudeily Ledesma Tamayo

***Co-tutor**:* *Ing. Keimer Montes Oliver*

La Habana, junio de 2015

Año 57 de la Revolución

“La innovación es lo que distingue a un líder de sus seguidores”

Steve Job

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Carlos Osvaldo Chao Cortes

_____
Firma del Autor

Yadira Robles Aranda

_____
Firma del Tutor

Arian Cruz Suárez

_____
Firma del Autor

Yudeily Ledesma Tamayo

_____
Firma del Tutor

Keimer Montes Oliver

_____
Firma del Tutor

DATOS DE CONTACTO

Msc. Yadira Robles Aranda:
Especialidad de graduación: Ingeniero en Ciencias Informáticas
Categoría docente: Asistente
Categoría científica: Master en Ciencias
Años de graduado: 7
Email: yrobles@uci.cu

Ing. Yudeily Ledesma Tamayo:
Especialidad de graduación: Ingeniero en Ciencias Informáticas
Categoría docente: Ninguna
Categoría científica: Ninguna
Años de graduado: 2
Email: yledesma@uci.cu.

Ing. Keimer Montes Oliver:
Especialidad de graduación: Ingeniero en Ciencias Informáticas
Categoría docente: Ninguna
Categoría científica: Ninguna
Años de graduado: 2
Email: kmontes@uci.cu

AGRADECIMIENTOS

Agradecimientos Carlos:

Hoy es un día muy importante para mí y no quiero dejar de agradecer a todas las personas que de una forma u otra han contribuido en mi formación como profesional.

A la persona que más quiero en el mundo mi "Mama" por darme la vida y ser mi guía en los momentos buenos y malos.

A mi hermano que aunque siempre nos fajemos los quiero con mi vida.

A Laritza por soportarme todos estos años en la universidad y por darme todo su amor y cariño.

A Braulio por ayudarme en esta etapa tan importante para mí.

A todos mis compañeros de aula por estar conmigo estos 5 años de carrera.

Agradecimientos Arian

Deseo agradecerles a toda mi familia en especial a mi mamá, papá, hermanos, abuelos y amigos. También a mis compañeros de aula los que empezaron en conjunto conmigo, los vi graduarse y a los que cuando volví me aceptaron como uno más de ellos. Junto a los cuales antes y ahora he pasado grandes momentos que siempre recordare y contare historias.

Agradecimientos Arian

DEDICATORIA

Dedicatoria Carlos:

A mi mamá por ser lo que más quiero en el mundo.

Dedicatoria Arian:

A mi familia que siempre me apoyo en cada momento de mi vida y han sabido guiarme.

Resumen

Entre los proyectos que se desarrollan en la Universidad de las Ciencias Informáticas se encuentra el Servidor Dinámico de Reporte (SDR), el cual es una herramienta informática que permite de manera eficiente la gestión, compilación, publicación y exportación de reportes creados en herramientas de diseño de reportes. En el SDR actualmente el manejo de los recursos es mediante comandos de consola o mediante una aplicación que consuma los servicios que brinda la API, lo que dificulta la gestión de la información para los usuarios con pocos conocimientos en informática. El trabajo de diploma está orientado al desarrollo de un Sistema de Administración para el SDR, el cual posibilitará de manera visual realizar la configuración, administración y la gestión de los recursos que maneja el servidor. Como metodología de desarrollo se utilizó *Agile Unified Process* (AUP). Para el modelado de la solución se utilizó Visual Paradigm en su versión 8.0 y como Entorno Integrado de Desarrollo (IDE) NetBeans 7.4 junto con el *plugins* del IReport 5.2 de dicho IDE para el diseño de los reportes, como sistema gestor de base de datos se utilizó PostgreSQL v9.3 y como herramienta de administración de base de datos PgAdmin III v1.14. Para la implementación de la solución se utilizó el lenguaje JavaScript y como marco de trabajo Qooxdoo v4.1.

Palabras Claves: Reportes, Servidor Dinámico de Reportes, Sistema de Administración

Abstract

Among the projects developed at the University of Informatics Science is the Dynamic Reporting Server (SDR), which is an informatics tool to efficiently manage, compiling, publishing and exporting reports created in report design tools. Currently in the SDR the management of resources is done through console commands or by an application that consume the services provided by the API, what makes difficult the information management for users with limited knowledge on informatics. This diploma work focuses on the development of a Management System for the SDR, which will allow to perform the configuration, administration and management of the resources managed by the server visually. Agile Unified Process(AUP) was used as development methodology. Visual Paradigm v8.0 tool was used to model the solution and NetBeans v 7.4 as Integrated Development Environment(IDE) along with IReport v5.2 plug-ins for report design, as management system database it was used PostgreSQL v9.3 and it was used as a management tool database PgAdmin III v1.14. For the implementation of the solution was used JavaScript language and as framework Qooxdoo v4.1.

Keywords: Reports, Dynamic Report Server, Management System

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TICs) caracterizan a la sociedad actual, donde la información juega un papel determinante en el desarrollo económico y es un factor clave para la búsqueda de nuevas alternativas que ayuden al proceso de toma de decisiones, la administración y el control estadístico de cualquier entidad moderna (Chen 2008). En el campo empresarial el análisis de la información es vital para el desarrollo y sostén de una institución, debido a esto la mayoría de las empresas se unen a la idea de la implantación y fomento del uso de las tecnologías para realizar cada uno de los diversos procesos de la empresa y analizar los datos generados en esos procesos. La acumulación de la información es un comportamiento común en la mayoría de las empresas, la creación de reportes de forma dinámica a partir de la información almacenada contribuye notablemente a la toma de decisiones al disminuir el tiempo y esfuerzo en el análisis de los datos. (Leticia Calzada 2009)

La Universidad de las Ciencias Informáticas (UCI), forma parte de las nuevas alternativas que se han llevado a cabo en Cuba para lograr un avance en el uso de las TICs. En su estructura productiva cuenta con varios centros de desarrollo de software entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC), con la misión de crear bienes y servicios informáticos relacionados con la gestión de datos. En el Departamento de Desarrollo de Aplicaciones de DATEC se desarrollan diferentes productos que ayudan a la gestión empresarial y a la toma de decisiones entre los cuales se encuentra el Servidor Dinámico de Reportes (SDR).

El SDR es un software que permite de manera eficiente la gestión, compilación, publicación y exportación de reportes creados en herramientas de diseño de reportes como el Generador Dinámico de Reportes (GDR) u otra aplicación que solicite el servicio. El mismo está basado en una arquitectura orientada a servicios utilizando servicios de Transferencia de Estado Representacional (REST[1] por las siglas en inglés de *Representational State Transfer*), la cual posibilita que aplicaciones con interfaces basadas en AJAX[2] puedan conectarse, publicar y consumir recursos, lo que permite que el servicio sea utilizado por distintos clientes escritos en diferentes lenguajes, corriendo en diversas plataformas y dispositivos. Entre los

[1] La Transferencia de Estado Representacional (*Representational State Transfer*) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web*.

[2] **AJAX**, acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

recursos que brinda el servidor se encuentran la gestión de usuarios, reportes publicados, reportes exportados, categorías, conexiones y tareas programadas.(Aguila 2014)

En el SDR actualmente el manejo de los recursos se realiza mediante comandos de consola o utilizando las API[3] de acceso a los servicios. Dichas API brindan un conjunto de funcionalidades que posibilitan la comunicación con el servidor, pero es necesario que el usuario final tenga conocimientos de programación para lograr integrar la API a su sistema, consumiendo tiempo y recursos en el proceso. Para gestionar los recursos mediante la consola, los usuarios finales deben tener conocimientos de cómo manejar los comandos que esta brinda para poder comunicarse con el servidor. Debido a esto para los administradores es engorroso realizar el mantenimiento y configuración ya que no cuentan con ninguna herramienta que le facilite realizar estas tareas.

Por lo antes planteado se presenta el siguiente **problema a resolver**: ¿Cómo gestionar de forma visual los recursos almacenados en el SDR?

Este trabajo tiene como **objeto de estudio** herramientas de administración de los servidores de reportes, enmarcado en el **campo de acción** herramienta de administración del Servidor Dinámico de Reportes.

De acuerdo con el problema planteado anteriormente se propone como **objetivo general**: Desarrollar una aplicación para la administración de forma visual de los recursos almacenados en el Servidor Dinámico de Reportes.

Para lograr el cumplimiento de los objetivos generales se proponen las siguientes **tareas de la investigación**:

1. Análisis de los principales conceptos asociados a las herramientas de administración de los motores de reportes para obtener la base teórica necesaria en la elaboración del marco teórico de la investigación.
2. Caracterización de las soluciones informáticas de herramientas de administración en servidores de reportes para recopilar las principales funcionalidades que estos presenten e incluirlas en el Sistema de Administración del Servidor Dinámico de Reportes.
3. Definición de la metodología y las herramientas informáticas a usar para el desarrollo del Sistema de Administración del Servidor Dinámico de Reportes.
4. Diseño de los reportes sobre el uso de los recursos del Servidor Dinámico de Reportes para brindarle al administrador una herramienta de ayuda en la toma de decisiones.

[3] *Application Programming Interface por sus siglas en ingles*

5. Definición de los principios de funcionamiento y diseño del Sistema de Administración del Servidor Dinámico de Reportes para guiar el proceso de implementación.
6. Implementación de los principios de funcionamiento y diseño del Sistema de administración del Servidor Dinámico de Reportes para satisfacer las necesidades administrativas de los recursos almacenados en el servidor.
7. Validación del sistema a partir de pruebas de software para comprobar su correcto funcionamiento.

Preguntas de investigación:

- ¿Cuáles son los fundamentos teóricos que respaldan el proceso de administración del SDR?
- ¿Cuáles son las características y capacidades que debe tener la aplicación para que con esta se permita la administración de los recursos almacenados en el SDR?
- ¿Cómo se debe estructurar el proceso de desarrollo del Sistema de Administración del SDR para que permita la gestión y administración de los recursos del servidor?
- ¿Cómo validar el correcto funcionamiento del Sistema de Administración del SDR?

Métodos teóricos:

- **Analítico-Sintético**: Este método permite realizar el estudio teórico sobre los servidores de reportes más conocidos en la actualidad, para seleccionar la información más importante sobre sus herramientas de administración y emplear la misma en el diseño del Sistema de Administración del SDR.
- **Modelación:** Facilita la realización de los diagramas correspondientes en las fases de: Inicio, Elaboración, Construcción y Transición de la solución propuesta.

Métodos empíricos:

- **Análisis Estático**: Permite observar al Servidor Dinámico de Reportes, con el objetivo de clasificar las dificultades existentes y las funcionalidades que debe tener el sistema a implementar a partir del análisis de los procesos de administración de otros servidores de reportes.
- **Revisión documental**: Se pone de manifiesto al revisar la documentación oficial de los motores de reportes y sus procesos administrativos para obtener el conocimiento necesario para el desarrollo del Sistema de Administración del SDR.
- **Tormenta de Ideas**: Este método fue utilizado en la reuniones con el cliente para hacer la identificación de los requisitos con lo que debía contar el sistema de administración del SDR.

INTRODUCCIÓN

El presente trabajo de diploma estará estructurado en III capítulos:

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA.

Durante el presente capítulo se realiza un análisis valorativo de algunas de las herramientas para la generación de reportes existentes en el mercado y de estos sus herramientas administrativas, dando lugar a una visión general del estado del arte de las mismas. Se efectúa un estudio de las herramientas y tecnologías que se utilizan para el ciclo de desarrollo de software basado en la web. Además, se seleccionan cuáles de los elementos estudiados serán aplicados durante la solución del problema, fundamentando las potencialidades que presentan cada una de estas herramientas.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL SISTEMA DE ADMINISTRACIÓN DEL SDR.

En este capítulo se realiza el análisis y diseño del Sistema de Administración del SDR. Para desarrollar estas etapas del proceso de desarrollo de software, se plantea una representación visual de las clases conceptuales del negocio que se deben analizar, a través del modelo de dominio. Posteriormente se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para dar cumplimiento al objetivo general de la investigación. Además se muestran los diagramas de clases del diseño, por medio de los cuales se representa la estructura estática del sistema y los patrones de arquitectura y diseño utilizados en el sistema.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE ADMINISTRACIÓN DEL SDR.

En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado. Como parte del mismo se presenta el diagrama de componentes del CU Gestionar Reportes y los estándares de codificación utilizados para el desarrollo de la aplicación. Además se describen la estrategia de pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento del Sistema de Administración del SDR en distintos momentos del ciclo de vida del software.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Durante el presente capítulo se realiza un análisis valorativo de algunas de las herramientas para la generación de reportes existentes en el mercado y de estos sus herramientas administrativas, dando lugar a una visión general del estado del arte de las mismas. Se efectúa un estudio de las herramientas y tecnologías que se utilizan para el ciclo de desarrollo de software basado en la web. Además, se seleccionan cuáles de los elementos estudiados serán aplicados durante la solución del problema, fundamentando las potencialidades que presentan cada una de estas herramientas.

1.1. Conceptos asociados al dominio de la investigación

1.1.1. Reporte

Un reporte es un informe o una noticia capaz de brindar información. Este tipo de documento (que puede ser impreso, digital o audiovisual) pretende transmitir una información, aunque puede tener diversos objetivos. Generalmente agrupan información de acuerdo a un interés específico, muestran el resultado de una búsqueda determinada. (Cortés 2014)

En la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos, para mostrarlos por medio de un diseño atractivo y fácil de interpretar por los usuarios. (Cortés 2014)

1.1.2. Herramientas para la generación de reportes

Los sistemas de generación de reportes están generalmente compuestos por dos elementos básicos: un diseñador de reportes y un motor de reportes. Este último es el encargado de construir el informe con los datos a partir de una plantilla previamente elaborada desde el diseñador de reportes. Entre sus capacidades se encuentra la obtención de reportes en diferentes formatos de salida. En la actualidad, existen herramientas que viabilizan el proceso de generación de reportes, dentro de las más importantes se encuentran:(Cortés 2014)

Crystal Reports

Crystal Reports facilita la creación de reportes simples y dispone también de poderosas herramientas necesarias para generar reportes complejos o especializados. Está diseñado para generar el reporte prácticamente desde cualquier origen de datos. Los asistentes incorporados guían al usuario paso a paso a través de la creación de reportes y la ejecución de tareas comunes relacionadas con el uso de reportes. Presenta licencia privativa y soporta los siguientes sistemas operativos: Microsoft Windows 7, Windows

Vista SP2, Windows XP SP3, Windows Server 2008 R2, Windows Server 2008 SP2, Windows Server 2008, Windows Server 2003 R2 SP2, Windows Server 2003 SP2.(SAP 2013)

Características de la herramienta:

- Administración del Servidor:

Crystal Reports Server incluye una serie de servicios de gestión de plataformas pre-configuradas y dedicadas a tareas como la administración de contraseñas, la programación de informes y el control de acceso de usuarios para soportar las funciones de gestión descentralizada. Las tareas de gestión pueden personalizarse y automatizarse utilizando los SDK[4] de *Crystal Reports Server*.(Crick 2008)

- Consola de administración centralizada:

La gestión de una serie de requisitos de interacción para el usuario final, así como los derechos de acceso y seguridad, requieren un entorno de administración potente, pero a la vez fácil de manejar, para un control total del sistema desde una sola interfaz web. *Crystal Reports Server* incluye la Consola de Administración Centralizada (CMC): un entorno web .NET o Java al 100% para la administración, utilización y configuración del sistema central. La CMC proporciona un control flexible, potente y granular del entorno para tareas como: configuración de funciones del usuario, acceso de seguridad, administración del servidor, gestión de la contraseña, entre otras. Además, permite que los administradores accedan y configuren con facilidad el sistema mientras controlan los derechos de acceso, las aplicaciones y la visualización por parte del usuario final.(Crick 2008)

Las herramientas de gestión y administración de *Crystal Reports Server* para el usuario final están creadas con los *componentes* para desarrolladores de software. La CMC se incluye para el control completo del entorno. La administración y gestión se benefician del almacenamiento central de todos los informes *Crystal Reports*, carpetas y perfiles de usuarios, acceso a las bases de datos de derechos de seguridad para el usuario, funciones y seguridad de documentos.(Crick 2008)

- Administración centralizada de configuración:

El administrador de configuración central (CCM) está exclusivamente diseñado para la administración del servidor y la configuración de los servicios de *Crystal Reports Server*. Esta herramienta permite iniciar, detener, habilitar e inhabilitar servidores.(Crick 2008)

[4] Un kit de desarrollo de software o SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto

La Figura 1 muestra la consola de administración central de *Crystal Report* en la misma se puede apreciar algunas de las funcionalidades con las que esta cuenta como son la gestión de categorías, tareas programadas y permisos.

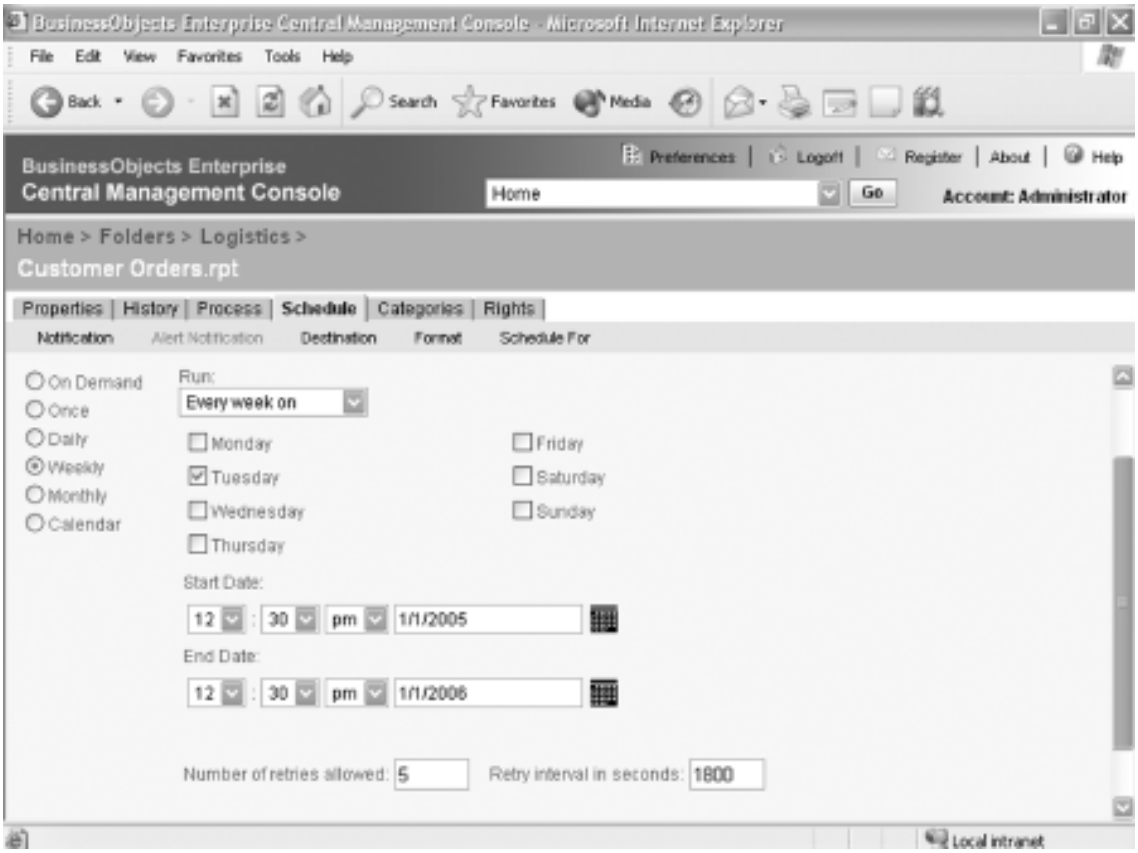

Fig 1- Consola central de administración del *Crystal Report.*

Active Reports Server

Active Report es una herramienta de plataforma propietaria que permite diseñar y crear reportes de forma fácil y rápida. Se caracteriza por dar soporte tanto a aplicaciones web como de escritorio, siendo capaz de emplear una amplia variedad de orígenes de datos.

Características de la herramienta:

- Licencia: privativa perteneciente a *GrapeCity*, inc.
- Sistema Operativo: Windows 7, Windows Server 2008, Windows Vista, Windows XP, Windows Server 2003. (ComponenteSource 2011)

Administración del Servidor:

Active Report permite múltiples opciones de administración de sus recursos y contiene las siguientes opciones:

- Gestión de Modelos: Mediante esta funcionalidad se puede ver y modificar todos los modelos de datos que tiene disponible para la creación de informes sobre el tablero de instrumentos del administrador.
- Gestión de Conexiones: Mediante esta funcionalidad los usuarios pueden gestionar los diferentes orígenes de datos que utilizarán para exportar los reportes.
- Gestión de Reportes: Los informes guardados en el servidor se muestran en el tablero de administración en la lista de informes. Los administradores pueden cambiar el nombre, establecer permisos, descargar, ver, borrar o subir los reportes.
- Gestión de Tareas Automáticas: Le permite a los usuarios la gestión de tareas automáticas, las cuales permiten la exportación de reportes de manera automática, en una fecha y hora específica.
- Gestión de Seguridad: ayuda a los administradores a gestionar usuarios, roles y permisos.
  i) Gestión de usuarios: Permite añadir y eliminar usuarios, modificar las credenciales de usuario, contraseña y roles.
  ii) Administración de funciones: Permite agregar y eliminar funciones, y para modificar los permisos de los roles.
  iii) Gestión de Permisos: Permite controlar los permisos que se pueden establecer en los informes y modelos.
- Gestión de la Configuración: Permite a los administradores administrar la configuración y diagnosticar problemas del servidor.
  i) Configuración de correo electrónico: Configuración de los ajustes de SMTP[5] para notificar las funciones de informes de error y el recordatorio de la contraseña.
  ii) Configuración del sitio URL de notificación: Configuración de una URL base para enlaces enviados en las notificaciones para las contraseñas perdidas y los informes programados.
  iii) Gestión de la configuración de auditoría: Configurar la auditoría en el servidor, los informes y datos se configura automáticamente.

[5]El Simple Mail Transfer Protocol (SMTP) (Protocolo para la transferencia simple de correo electrónico), es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (P DA, teléfonos móviles, etc.)

iv) Diagnósticos del servidor: Permite el diagnóstico de problemas del servidor. (GrapeCity 2011)

En la Figura 2 se puede observar la pizarra de administración del *Active Report Server* mediante la cual se puede apreciar algunas de las funcionalidades que esta permite con respecto a la gestión de sus recursos, como son los modelos, reportes y tareas programadas.

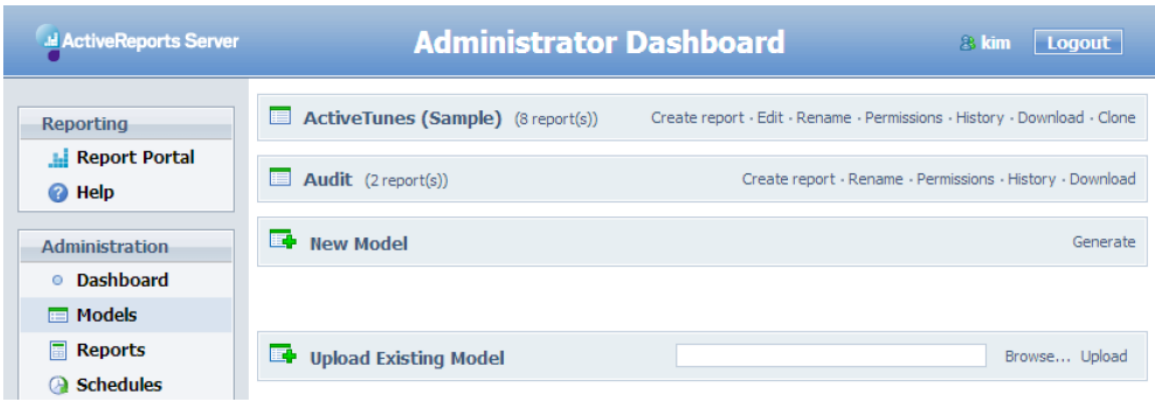

Fig 2-*Dashboard (Pizarra)* de administración del *Active Reports.*

Servidor Dinámico de Reportes

El SDR es un software que permite de manera eficiente la gestión, compilación, publicación y exportación de reportes creados en herramientas de diseño de reportes, como el GDR u otra aplicación que solicite el servicio. Permite además que un usuario independiente, que de forma directa interactúe con el servidor a través de alguna herramienta de comunicación como CURL pueda consumir los servicios, siempre y cuando tenga los debidos permisos recibirá exitosamente la respuesta a la petición realizada.

El SDR garantiza la seguridad en el acceso a los servicios y recursos a través de un sistema de generación de *token* de seguridad, que consiste en una cadena de caracteres de 32 bits que se le envía al usuario luego de autenticarse con la contraseña correcta, y debe ser incluido en la cabecera de cada petición del usuario mientras el tiempo de vida del *token* sea válido. Además se verifica el permiso de acceso de la dirección IP (*Internet Protocol*) desde donde se realiza la petición, así como si el usuario tiene permisos para acceder a los recursos que solicita.(Aguila 2014)

Características y funcionalidades principales:

- Permite la generación reportes en múltiples formatos como PDF, XML, HTML, CSV, XLS, RTF, TXT, JPEG, DOC, ODT, PPT, ODS a partir de diferentes fuentes de datos.

- Las conexiones a las fuentes de datos de los reportes se limita solo por el soporte de los API JDBC[6] (*Java Data Base Connection*) de Java.
- Permite la generación de Subr-eportes.
- Visualiza las vistas previas de los reportes.
- Posibilita la selección de estilos de estructura para un grupo de reportes.
- No limita el tamaño del diseño del reporte para la generación de reportes.
- Atiende peticiones de aplicaciones web o de escritorio a través de servicios REST.
- Es multiplataforma.
- Implementa un conjunto de políticas de seguridad para el acceso a los recursos a varios niveles.
- Provee un mecanismo de integración con cualquier sistema informático encargado del diseño de los reportes desarrollado en PHP, de forma que resulta transparente la comunicación entre ellos.(Aguila 2014)

1.1.3. Conclusiones de la investigación

Después de realizar un análisis sobre los diferentes servidores de reportes más usados y específicamente de sus herramientas administrativas, se puede decir que en todos se realizan la gestión de usuarios, reportes, tareas programadas, conexiones y otros recursos que maneje el servidor. Para el cumplimiento de estos objetivos, dichos servidores de reportes cuentan con una interfaz visual la cual posibilita una interacción dinámica con los recursos del servidor. Como parte del mantenimiento y configuración, el *Crystal Report* y *Active Report* cuentan con un conjunto de funcionalidades como el diagnóstico, mantenimiento, informe de errores y notificaciones por correo. Para lograr que la interacción de los usuarios con el SDR se realice de una manera más intuitiva, es necesario que este cuente con una herramienta administrativa la cual permita de manera visual la gestión de los recursos que se almacenan en el mismo. Además debe posibilitarle a los administradores conocer el estado del servidor, gestionar la configuración y realizar su mantenimiento.

1.2. Metodología de desarrollo de software

Una metodología de desarrollo de software es el conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo sistema informático. Su objetivo es guiar y estructurar actividades que conlleven a las metas trazadas por el equipo de desarrollo. Las

[6] **JDBC**, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el lenguaje SQL del modelo de base de datos que se utilice.

mismas se clasifican en tradicionales y ágiles. Las primeras centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con el plan de proyecto. Las segundas generan menos documentación y son más orientadas al código, dentro de estas metodologías se encuentra AUP, la cual es la utilizada en la Universidad de las Ciencias Informáticas para el desarrollo en los proyectos productivos. Se seleccionó como metodología de desarrollo *AUP*, ya que el SDR está guiado por la misma, además de ser una metodología ágil que posee como una de sus principales ventajas que se adapta perfectamente a proyectos de corta duración al estar diseñada para soportar equipos de trabajos de pocos integrantes. Con esta metodología se evita también la confección de una extensa documentación, diagramas e iteraciones menos importantes.

AUP:

El Proceso Unificado Ágil de Scott Ambler (*Agile Unified Process* (AUP) en inglés) es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP aplica técnicas ágiles incluyendo:

- Desarrollo dirigido por pruebas (*Test Driven Development* - TDD en inglés).
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.(Sánchez 2014)

Fases AUP

1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.(Sánchez 2014)

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) existe una modificación para el ciclo de vida de los proyectos de la UCI la cual es mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la cual es llamada Ejecución y se agrega la fase de Cierre, como se muestra en la Figura 3.(Sánchez 2014)

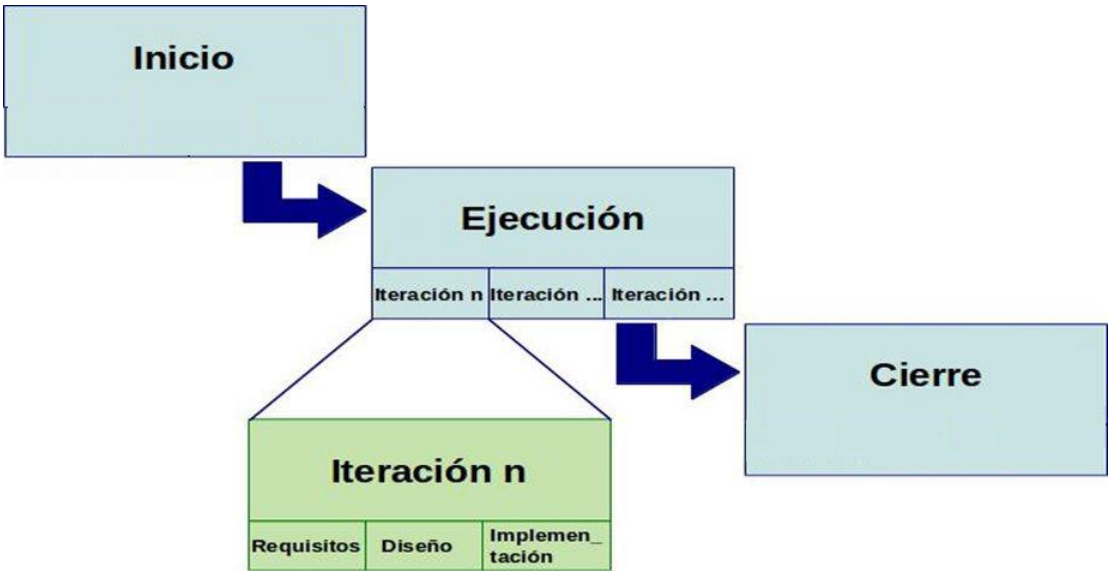

Fig 3- Adaptación del ciclo de vida de la metodología AUP con el ciclo de vida definido por el Modelo de Desarrollo-Producción UCI.

AUP propone 7 disciplinas (Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decidió para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Las disciplinas: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional.(Sánchez 2014)

1.3. Servidor web

Un servidor web es un programa que implementa el protocolo HTTP (*Hypertext Transfer Protocol).* Este protocolo está diseñado para transferir hipertextos, páginas web o páginas HTML *(Hypertext Markup Language*): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. (Cristalab 2008)

Apache 2.2

Apache es un servidor de código abierto para plataformas Unix, Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se

basó inicialmente en código del popular NCSA HTTPD 1.3[7], pero más tarde fue reescrito por completo. (Sanchez 2011)

Ventajas de Apache 2.2

- Modular
- Código abierto
- Multi-plataforma
- Extensible

1.4. Gestor de base de datos.

Los Sistemas Gestores de Bases de Datos (SGBD) están constituidos por un paquete de *software* cuya función es la gestión del acceso a la base de datos. Las operaciones fundamentales son: crear, modificar, eliminar y obtener la estructura asociada al esquema lógico de una base de datos. El objetivo principal es proporcionar un entorno práctico y eficiente a la hora de almacenar y recuperar la información de la base de datos, lo que facilita la gestión de grandes volúmenes de datos.

PostgreSQL 9.3

PostgreSQL es un SGBD objeto relacional. Presenta propiedades como atomicidad, consistencia, aislamiento y durabilidad. La atomicidad asegura la realización de una operación, por lo tanto ante un fallo del sistema no puede quedar a medias. La consistencia posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. El aislamiento asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error. Durabilidad es la propiedad que asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema.(PostgreSQL 2012)

Principales Características:

- Incorpora una estructura de datos de arreglos.
- Soporta el uso de índices, reglas y vistas.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes entre otras.

[7] NCSA HTTPD era un Servidor web desarrollado originalmente en el National Center for Supercomputing Applications por Robert McCool y una lista de colaboradores

- Incluye herencia entre tablas aunque no entre objetos, ya que no existen, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de estos.
- Soporta distintos tipos de datos: Además de los datos base, soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.(PostgreSQL 2012)

PgAdmin III v 1.14

PgAdmin III es una herramienta gráfica y multiplataforma para administrar y desarrollar bases de datos en PostgreSQL. Está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Ofrece una interfaz gráfica que soporta todas las características de PostgreSQL 9.3, permitiendo realizar cambios en la base de datos desde el sistema de manera sencilla e incluye un editor con resaltado de sintaxis SQL. (PgAdmin 2010)

1.5. Lenguaje de Modelado

Es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos. Generalmente se usa extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

Lenguaje Unificado de Modelado (UML 2.0)

UML 2.0 prescribe un conjunto de notaciones y diagramas estándares para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real, ayudando al usuario a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el coste y el tiempo empleado en la construcción de las piezas que constituirán el modelo. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos pues permite la modelación de sistemas con tecnología orientada a objetos y describe lo que

supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.(Object Management Group 2012)

Los principales beneficios de UML son:

- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.6. Herramienta de Modelado Case

Las Herramientas CASE (*Computer Aided Software Engineering*, del español Ingeniería de Software Asistida por Computadoras) son aplicaciones de apoyo al desarrollo, mantenimiento y documentación automatizados de software. Permiten aplicar la metodología de análisis y diseño orientados a objetos. Cuanto más grande es un proyecto, más importante es utilizar una herramienta CASE, ya que contribuyen a aumentar la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y dinero.(Méndez 2013)

Visual Paradigm for UML 8.0

Visual Paradigm for UML es una herramienta CASE de diseño UML implementada para ayudar al desarrollo del ciclo de vida completo de un software. Ofrece un conjunto de herramientas necesarias para la realización de las diferentes tareas del proceso, tales como: captura de requisitos, planificación y modelado de clases. Genera documentación en formato PDF, Word o HTML. Cuenta con múltiples versiones para cada necesidad y disponibilidad de integrarse, en los principales IDE (Entorno de Desarrollo Integrado). (Visual Paradigm 2012)

1.7. Lenguaje de Programación

Un lenguaje de programación es una construcción incremental del ser humano para escribir programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás.

JavaScript

JavaScript se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se

activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.(Pérez 2009)

1.8. Entorno de Desarrollo Integrado (IDE)

NetBeans 7.4

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un conjunto de herramientas para el programador que incluyen un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores e integración con sistemas controladores de versiones o repositorios.

NetBeans IDE 7.4 es un entorno de desarrollo integrado de código abierto, distribución gratuita, sin restricciones de uso y apoyado además por una gran comunidad de desarrolladores. Aunque está escrito en Java puede ser usado con otros lenguajes de programación como Java, C/C++, JavaScript, PHP y Phyton. Permite el desarrollo de aplicaciones de escritorio, web y para móviles. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas.(Oracle 2012)

Características:

- Incluye el control de versiones y compilación avanzada.
- Es multiplataforma, está disponible para GNU/Linux, Windows, Mac OS X y Solaris.
- Separa el diseño del software de la implementación con modelado UML.
- Creación visual de componentes gráficos.
- Dispone de soporte para la creación de interfaces gráficas de forma visual, agregando y alineando el espacio de trabajo.(Oracle 2012)

1.9. Diseñador de reportes

Proporciona las herramientas necesarias para darle formato en forma de documento a la información almacenada en las bases de datos. Entre las funcionalidades que presenta se encuentran:

- Creación de conexiones desde diferentes BD.
- Crear consultas SQL.
- Añadir elementos gráficos para la visualización de la información como tablas y gráficas.

iReport v5.2

iReport es un diseñador de reportes de código abierto para JasperReports y JasperReports Server.Crea diseños muy sofisticados que contienen gráficos, imágenes, subreportes, tablas de contingencia y mucho

más. Puede acceder a sus datos a través de JDBC, JavaBeans, fuentes XML, Hibernate, CSV, y personalizados. (Jaspersoft Community 2014)

La lista siguiente describe algunas de las características importantes de IReport:

- 100% escrito en Java y además de ser código abierto y gratuito.
- Maneja el 98% de las etiquetas de JasperReport.
- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de texto, cartas y subreportes.
- Soporta internacionalización nativamente.
- Recopilador y exportador integrados.
- Soporta JDBC.
- Soporta JavaBeans como orígenes de datos.
- Incluye asistentes para crear automáticamente reportes y para generar los subreportes.

1.10. Marco de trabajo

Desde el punto de vista del desarrollo de software un *framework* o marco de trabajo es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Los *frameworks* suelen incluir:

- Soporte de programas.
- Lenguaje de scripting.
- Herramienta para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas.(Alegsa 2014)

Qooxdoo v 4.1

Es un marco de trabajo desarrollado en JavaScript de carácter universal que le permite crear aplicaciones para una amplia gama de plataformas. Con su modelo de programación orientada a objetos se puede construir aplicaciones ricas e interactivas, aplicaciones nativas para dispositivos móviles, aplicaciones web tradicionales o incluso aplicaciones de escritorio. (Qooxdoo developers 2014)

Características principales:

- Permite al desarrollador abstraerse del HTML, CSS y DOM de la aplicación.
- Programación orientada a objetos.
- Multi navegador.
- Soporte Ajax.

- Componentes de aspecto similar a los usados en los entornos de escritorio.

1.11. Conclusiones del capítulo.

En el capítulo que finaliza se determinó que para el correcto funcionamiento de todo servidor de reportes es necesario tener una aplicación que se encargue de la administración del mismo, la cual visualice el estado de los recursos del servidor y permita realizarle el mantenimiento y configuración. Además se definieron un conjunto de funcionalidades con las cuales debe contar el Sistema de Administración del SDR para lograr que los usuarios finales tengan una herramienta que les facilite la interacción con el servidor. Se seleccionó como metodología de desarrollo de software AUP, la cual guiará todo el proceso de desarrollo de la herramienta propuesta. Se estudiaron las herramientas y tecnologías que se emplearán en la solución seleccionando UML 2.0 como leguaje de modelado, Visual Paradigm 8.0 como herramienta CASE, JavaScript como lenguaje de programación. Se utilizará como marco de trabajo Qooxdoo v4.1, como IDE de desarrollo a utilizar será el NetBeans 7.4. Como SGBD se empleará PostgreSQL 9.3 y para su administración gráfica el PgAdmin III 1.14; Apache 2.2 será utilizado como servidor web. Estas herramientas apoyarán el desarrollo general del módulo, brindando facilidades a los desarrolladores en cuanto a rapidez, facilidad de uso y variedad de funciones, sin dejar de mencionar sus licencias no son propietarias.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL SISTEMA DE ADMINISTRACIÓN DEL SDR

En este capítulo se realiza el análisis y diseño del Sistema de Administración del SDR. Para desarrollar estas etapas del proceso de desarrollo de software, se plantea una representación visual de las clases conceptuales del problema de la investigación que se deben analizar, a través del modelo de dominio. Posteriormente se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para dar cumplimiento al objetivo general de la investigación. Además se muestran los diagramas de clases del diseño, por medio de los cuales se representa la estructura estática del sistema y los patrones de arquitectura y diseño utilizados en el sistema.

2.1. Modelo de Dominio

"Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se le denomina modelo conceptual, pues ayuda a comprender los conceptos clave de un negocio o un dominio del problema." (Larman 2004)

A continuación se presenta el modelo de dominio del Sistema de Administración del SDR y se describen sus clases.

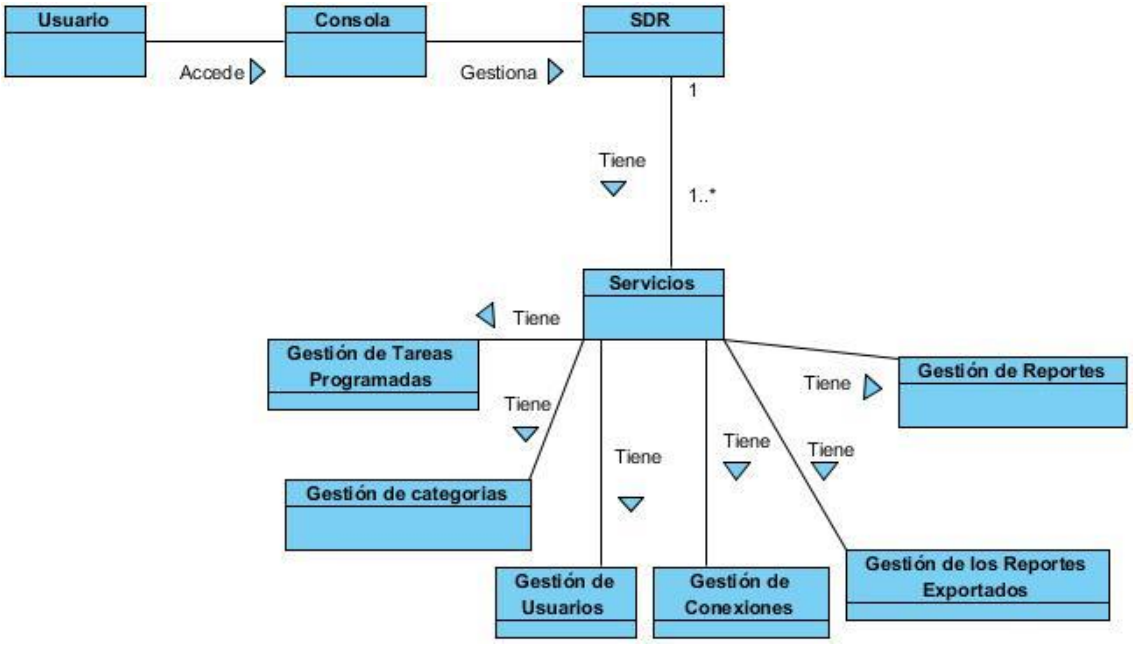

Fig 4-Modelo de Dominio del Sistema.

1. Consola: Objeto que representa a la consola del sistema mediante la cual se va a acceder al SDR utilizando los comandos que brinda para gestionar sus recursos.

2. Usuario: Objeto que representa al cliente que utiliza la consola del SDR para gestionar los recursos, los usuarios están divididos en 2 roles el Administrador y el Sistema Informático, este último es la entidad o aplicación que consume los servicios que brinda el SDR.

3. SDR: Objeto que representa al Servidor Dinámico de Reportes el cual debe brindar una capa de servicios a través de su integración con cualquier otro sistema web o de escritorio. A través de este los sistemas podrán generar reportes en una gran gama de formatos sobre múltiples fuentes u orígenes de datos.

4. Servicios: Objeto que representa los servicios que soporta la versión 1.0 del Servidor Dinámico de Reportes. Los servicios están asociados o en correspondencia con la gestión de la seguridad, gestión de reportes, gestión conexiones a fuentes de datos y la gestión de las tareas programadas respectivamente.

4.1. Gestión de reportes: Objeto que representa a los servicios asociados al diseño de los reportes, sobresalen las opciones de listar, publicar, actualizar, validar, compilar y eliminar los diseños de reportes en el servidor.

4.2. Gestión de conexiones: Objeto que representa a los servicios asociados a la gestión de conexiones a fuentes de datos, permitiendo las opciones de listar, crear, actualizar y eliminar las variables de conexión a las fuentes de datos. Serán soportadas las fuentes de datos sobre PostgreSQL, Oracle, MySQL, SQLite, otras que contengan un conector JDBC y sean accesibles desde el servidor.

4.3. Gestión de tareas programadas: Objeto que representa a los servicios asociados a la gestión de las tareas programadas, las mismas son utilizadas para notificar a los usuarios o para ejecutar acciones en un instante definido.

4.4. Gestión de usuarios: Objeto que representa los servicios asociados a la gestión de usuarios permitiendo crear, actualizar, listar y eliminar los usuarios del sistema.

4.5. Gestión de reportes exportados: Objeto que representa los servicios asociados a la gestión de los reportes permitiendo crear, actualizar, listar y eliminar los reportes exportados del sistema.

4.6. Gestión de categorías: Objeto que representa los servicios asociados a la gestión de las categorías de los reportes la cual permite crear, actualizar, listar y eliminar.

2.2. Especificación de los Requisitos del sistema.

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. “El esfuerzo principal en la disciplina Requisitos es desarrollar el modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto”.(Sánchez 2014)

2.2.1. Requisitos Funcionales

El Sistema de Administración del SDR debe cumplir los siguientes requisitos funcionales:

RF-1. Adicionar usuario.

Entrada: name, username, password, email, subNet y rol.

Descripción: El sistema debe permitir adicionar un usuario mediante una interfaz gráfica en la cual el usuario insertará los datos. El sistema valida los datos entrados, y si todo es correcto crea el usuario, sino devuelve las excepciones en la respuesta.

Salida: Se añade un usuario al sistema.

RF-2. Listar usuarios.

Entrada: ---

Descripción: El sistema debe permitir visualizar la respuesta de la petición realizada, mostrando una lista de los usuarios existentes en el sistema.

Salida: Se listan todos los usuarios del sistema.

RF-3. Modificar usuario.

Entrada: id, name, password, username, email,subNet y rol.

Descripción: El sistema comprueba la existencia del usuario a modificar por su identificador (id), si no existe muestra una excepción de otra forma debe permitir editar un usuario mediante una interfaz en la cual el usuario insertará los datos.

Salida: Se modifica el usuario seleccionado.

RF-4. Eliminar usuario.

Entrada: id.

Descripción: El sistema verifica en la base de datos la existencia del id del usuario a eliminar enviado por parámetros, si la verificación es satisfactoria elimina el usuario de la base de datos del servidor y el fichero del disco duro correspondiente al usuario, donde se encuentran todos los reportes y exportaciones asociados a él.

Salida: Se elimina el usuario seleccionado.

RF-5. Buscar usuario por parámetro.

Entrada: name, subNet y rol

Descripción: El sistema busca en la base de datos del servidor el usuario especificado por el parámetro enviado y lo devuelve, sino devuelve la excepción en la respuesta.

Salida: Se listan los usuarios que correspondan con el filtro seleccionado.

RF-6. Adicionar conexión.

Entrada: name, url, nameDB, username, password y puerto

Descripción: El sistema debe permitir la opción de adicionar una conexión, mostrándole al usuario un formulario donde introducirá los datos de la nueva conexión.

Salida: Se adiciona una conexión a la base de datos.

RF-7. Listar conexiones.

Entrada: ---

Descripción: El sistema permite realizar un listado de todas las conexiones existentes en la base de datos.

Salida: Se muestran todas las conexiones del sistema.

RF-8. Modificar conexión.

Entrada: id, name, url, nameDB, username, password y puerto

Descripción: El sistema comprueba que existe la conexión seleccionada, si no lo encuentra muestra un error de forma contraria debe permitir la opción de modificar la conexión.

Salida: Se modifica la conexión seleccionada.

RF-9. Eliminar conexión.

Entrada: id.

Descripción: El sistema verifica en la base de datos la existencia del id de la conexión a eliminar enviado por parámetros, si la verificación es satisfactoria elimina la conexión de la base de datos del servidor.

Salida: Se elimina la conexión seleccionada de la base de datos.

RF-10. Buscar conexión por parámetros.

Entrada: name, driver y username

Descripción: El sistema permite realizar una búsqueda por un parámetro introducido por el usuario.

Salida: Se listan las conexiones que correspondan con el filtro seleccionado.

RF-11. Probar conexión.

Entrada: name, url, nameDB, username, password y puerto

Descripción: El sistema debe permitir probar una conexión con la fuente de datos que el usuario paso por parámetros.

Salida: Se envía la respuesta de si estableció o no la conexión con la base de datos.

RF-12. Autenticar usuario.

Entrada: username y password

Descripción: El sistema debe permitir iniciar y cerrar sesión a los usuarios del sistema, verificando que en la base de datos del servidor exista el usuario y contraseña. El sistema crea una clave (token) de autenticación para el usuario.

Salida: Entra el usuario en el sistema.

RF-13. Publicar reporte.

Entrada: name, category, description y Jrxml

Descripción: El sistema debe permitir la opción de publicar un reporte a la base de datos del servidor a partir del fichero .jrxml y el json del reporte recibido por los parámetros entrados por el usuario.

Salida: Se crea un reporte en la base de datos.

RF-14. Listar reportes publicados.

Entrada: ---

Descripción: El sistema muestra un listado de todos los reportes existentes en la base de datos.

Salida: Se muestran todos los reportes publicados en el servidor.

RF-15. Modificar reportes publicados.

Entrada: id, name, category y description.

Descripción: El sistema debe permitir la opción de modificar los datos de un reporte a partir del id pasado por parámetros, actualizando los campos de la base de datos del servidor del reporte modificado.

Salida: Se modifican los datos del reporte seleccionado.

RF-16. Eliminar reportes publicados.

Entrada: id.

Descripción: El sistema verifica en la base de datos la existencia del id del reporte a eliminar enviado por parámetros, si la verificación es satisfactoria elimina el reporte de la base de datos del servidor.

Salida: Se elimina el reporte seleccionado de la base de datos.

RF-17. Buscar reportes publicados por parámetros.

Entrada: name, category, createAt, updateAt, lastAccess

Descripción: El sistema debe permitir la opción de buscar en la base de datos del servidor el reporte especificado por los parámetros entrados en la petición.

Salida: Se listan los reportes que correspondan con el filtro seleccionado.

RF-18. Exportar reporte publicado.

Entrada: conexión, formato y parámetros.

Descripción: A partir del id del reporte seleccionado, el sistema permite exportar el mismo en los diferentes formatos que soporta el servidor.

Salida: Se devuelve la dirección del reporte exportado.

RF-19. Visualizar reporte publicado.

Entrada: Dirección del reporte exportado.

Descripción: El sistema muestra la vista previa de un reporte exportado después de una petición de exportar un reporte.

Salida: Se visualiza el reporte exportado.

RF-20: Descargar JRXML de reporte publicado.

Entrada: id.

Descripción: El sistema debe permitir la opción de descargar el fichero .jrxml de un reporte especificado por su identificador que debe ser entrado por el usuario.

Salida: Se muestra el JRXML del reporte seleccionado.

RF-21: Modificar JRXML de reporte publicado.

Entrada: Jrxml

Descripción: El sistema debe permitirle al usuario modificar el JRXML de un reporte previamente creado.

Salida: Se modifica el Jrxml del reporte seleccionado.

RF-22. Adicionar tarea programada.

Entrada: name, frecuency, report, connection, email y ftp

Descripción: El sistema debe permitir la opción de adicionar una tarea programada. Para adicionar una tarea programada se deben especificar los valores de todos los campos según el tipo de tarea programada que se haya seleccionado.

Salida: Se crea una tarea programada en la base de datos.

RF-23. Listar tareas programadas.

Entrada: ---

Descripción: El sistema debe permitir visualizar el listado de tareas programadas.

Salida: Se muestran todas las tareas programadas del sistema.

RF-24. Modificar tarea programada.

Entrada: id, name, frecuency, report, connection, email y ftp.

Descripción: El sistema debe permitir la opción de modificar una tarea programada. Para modificar una tarea programada se pueden modificar los datos asociados a las variables en dependencia del tipo de envío seleccionado.

Salida: Se modifica la tarea programada seleccionada.

RF-25. Eliminar tarea programada.

Entrada: id.

Descripción: El sistema verifica la existencia de la tarea seleccionada por su id, si no existe devuelve una excepción, de forma contraria le permite al usuario la opción de eliminar la tarea programada seleccionada.

Salida: Se elimina la tarea programada seleccionada.

RF-26. Buscar tarea programada por parámetros.

Entrada: format, nameTask y triggerTask.

Descripción: El sistema permite realizar una búsqueda de tarea o tareas programas según los parámetros entrados por el usuario.

Salida: Se devuelve las tareas programadas que cumplan con el filtro seleccionado.

RF-27. Listar reportes exportados

Entrada: ---

Descripción: El sistema muestra un listado de todos reportes exportados.

Salida: Se listan todos los reportes exportados del sistema.

RF-28. Visualizar reporte exportado.

Entrada: id.

Descripción: El sistema muestra el reporte en el formato que el usuario seleccionó para ser exportado.

Salida: Se visualiza el reporte exportado seleccionado.

RF-29. Eliminar reporte exportado.

Entrada: id.

Descripción: El sistema debe permitir la opción de eliminar un reporte según el identificador que introduzca el usuario. Donde luego de eliminar el reporte de la base de datos del servidor, se elimina la

carpeta del sistema de archivos del servidor donde estaban contenidas todas las exportaciones del reporte en cuestión.

Salida: Se elimina el reporte exportado seleccionado.

RF-30. Buscar reporte exportado por parámetros.

Entrada: format, createAt, remoteIp y status.

Descripción: El sistema debe permitir la opción de buscar en la base de datos del servidor el reporte especificado por los parámetros entrados.

Salida: Se devuelve los reportes exportados que cumplan con el filtro seleccionado.

RF-31. Listar reportes del uso del sistema.

Entrada: ---

Descripción: El sistema muestra un listado de todos los reportes del uso del sistema.

Salida: Se muestra un listado de los reportes del uso del sistema almacenados en la base de datos.

RF-32. Visualizar reporte del uso del sistema.

Entrada: id.

Descripción: El sistema muestra el reporte del uso del sistema seleccionado por el usuario.

Salida: Se visualiza el reporte del uso del sistema seleccionado.

RF-33. Realizar mantenimiento del sistema.

Entrada: id.

Descripción: El sistema realiza las acciones de mantenimiento según el reporte que se esté visualizando.

Salida: Se envía una notificación por correo a los usuarios que presenten problemas según el reporte que se seleccionó.

RF-34. Adicionar consulta de mantenimiento.

Entrada: reporte, consulta y mensaje.

Descripción: El sistema le brinda una interfaz al usuario donde se introducen los datos para crear una consulta de mantenimiento.

Salida: Se crea una consulta de mantenimiento en la base de datos.

RF-35. Listar consulta de mantenimiento.

Entrada: ---.

Descripción: El sistema debe permitir visualizar la respuesta de la petición realizada, mostrándose un listado con todas las consultas de mantenimiento del sistema.

Salida: Se muestra una lista de todas las consultas de mantenimiento del sistema.

RF-36. Modificar consulta de mantenimiento.

Entrada: reporte, consulta y mensaje.

Descripción: El sistema verifica la existencia de la consulta seleccionada, si no existe lanza una excepción, de forma contraria muestra una interfaz para la modificación de la consulta.

Salida: Se modifica la consulta de mantenimiento seleccionada.

RF-37. Eliminar consulta de mantenimiento.

Entrada: id.

Descripción: El sistema verifica la existencia de la consulta seleccionada por el id, si no existe lanza una excepción, de forma contraria elimina la consulta de forma satisfactoria.

Salida: Se elimina la consulta de mantenimiento seleccionada.

RF-38. Visualizar logs[8] del sistema.

Entrada: ---.

Descripción: Muestra los logs del uso del sistema, permitiéndole conocer al usuario qué acciones se realizan en determinado momento.

Salida: Se visualizan todos los logs del sistema almacenados en los archivos de logs.

RF-39. Buscar logs del sistema por parámetros.

Entrada: tipo, fecha y usuario.

Descripción: El sistema permite buscar los logs por el elemento pasado por parámetros.

Salida: Se muestran los logs del sistema que correspondan con el filtro introducido.

RF-40. Editar parámetros de configuración del servidor.

Entrada: port, server, user , password, trustServer, protocol y timeout

Descripción: El sistema permite la edición de los parámetros de configuración del servidor.

Salida: Se editan los parámetros de configuración del servidor.

RF-41. Adicionar categoría.

Entrada: name, description y fatherCategory.

Descripción: El sistema le brinda una interfaz al usuario donde se introducen los datos de la categoría a crear y después de verificar los datos enviados se crea la categoría.

Salida: Se adiciona una categoría a la base de datos.

RF-42. Listar categorías.

[8] Un **log** es un registro oficial de eventos durante un rango de tiempo en particular.

Entrada: ---.

Descripción: El sistema debe permitir visualizar la respuesta de la petición realizada, mostrándose un listado con todas las categorías que él ha creado.

Salida: Se muestra un listado con todas las categorías del sistema

RF-43. Modificar categoría.

Entrada: name, description y fatherCategory.

Descripción: El sistema verifica la existencia de la categoría seleccionada, si no existe lanza una excepción, de forma contraria muestra una interfaz para la modificación de la categoría.

Salida: Se modifica la categoría seleccionada.

RF-44. Eliminar categoría.

Entrada: id.

Descripción: El sistema verifica la existencia de la categoría seleccionada por id, si no existe lanza una excepción, de forma contraria elimina la categoría de forma satisfactoria.

Salida: Se elimina la categoría seleccionada.

RF-45. Buscar categoría por parámetro.

Entrada: name y fatherCategory.

Descripción: El sistema busca en la base de datos del servidor la categoría especificada por el parámetro enviado y la devuelve, sino devuelve la excepción en la respuesta.

Salida: Se devuelve las categorías que cumplan con el filtro seleccionado.

2.2.2 Requisitos No Funcionales

Este tipo de requisito describe las “restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Estos requisitos a menudo se aplican al sistema en su totalidad.”(Sommerville 2005)

Usabilidad

RnF1: Tipo de Aplicación Informática.

La herramienta debe ser web, pero con características muy similares a las aplicaciones de escritorio en cuanto al diseño de las interfaces visuales y los tiempos de respuesta de la interacción del usuario con el sistema.

RnF2. Finalidad: El objetivo que persigue la aplicación es que los usuarios puedan tener una herramienta que les permita interactuar de manera visual con los recursos que gestiona el SDR.

RnF3. Software requerido para desplegar y utilizar la aplicación:

Host para instalar la aplicación: La máquina donde se instalará la aplicación debe cumplir con los siguientes requisitos de software:

- Sistema Operativo: GNU/Linux Ubuntu 12.04 o superior, Debian 7 GNU/Linux o superior.
- Usuario con privilegios de administración del Sistema Operativo.
- Tener instalador PostgreSQL en su versión 9.3 o superior.

RnF4. Hardware requerido para desplegar y utilizar la aplicación.

Las PC clientes deben cumplir con los siguientes requisitos de hardware:

- Ordenador Pentium IV o superior, con 3.0 GHz de velocidad de microprocesador.
- Memoria RAM, mínimo 1GB.

Confiabilidad.

RnF5. Fiabilidad.

El sistema debe estar disponible las 24 horas del día. En caso de fallo, pudiera estar fuera de servicio por un período de 72 horas máximo. La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información que brinda el SDR.

Restricciones diseño e implementación

RnF6. Marco de trabajo para el desarrollo del sistema del lado del cliente.

El marco de trabajo que se utilizará en el desarrollo del Sistema es Qooxdoo en su versión 4.1.

Interfaz

El usuario deberá acceder a la aplicación a través del protocolo HTTP o HTTPS usando el navegador Firefox en sus versiones 24.0 a 36.0.

RnF7. Interfaces de usuario: Las interfaces de usuario serán diseñadas a modo de aplicaciones RIA (*Rich Internet Application*) lo que permite a los usuarios contar con una aplicación web con una experiencia de usuario similar a la de las aplicaciones de escritorio.

RnF8. Interfaces de Comunicación.

El sistema puede ser desplegado sobre red LAN, MAN o WAM; siempre y cuando la velocidad de conexión sea mayor que 1 Mbit/s.

2.3. Modelo de Casos de Uso del Sistema

2.3.1. Diagrama de Casos de Uso del Sistema

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar.

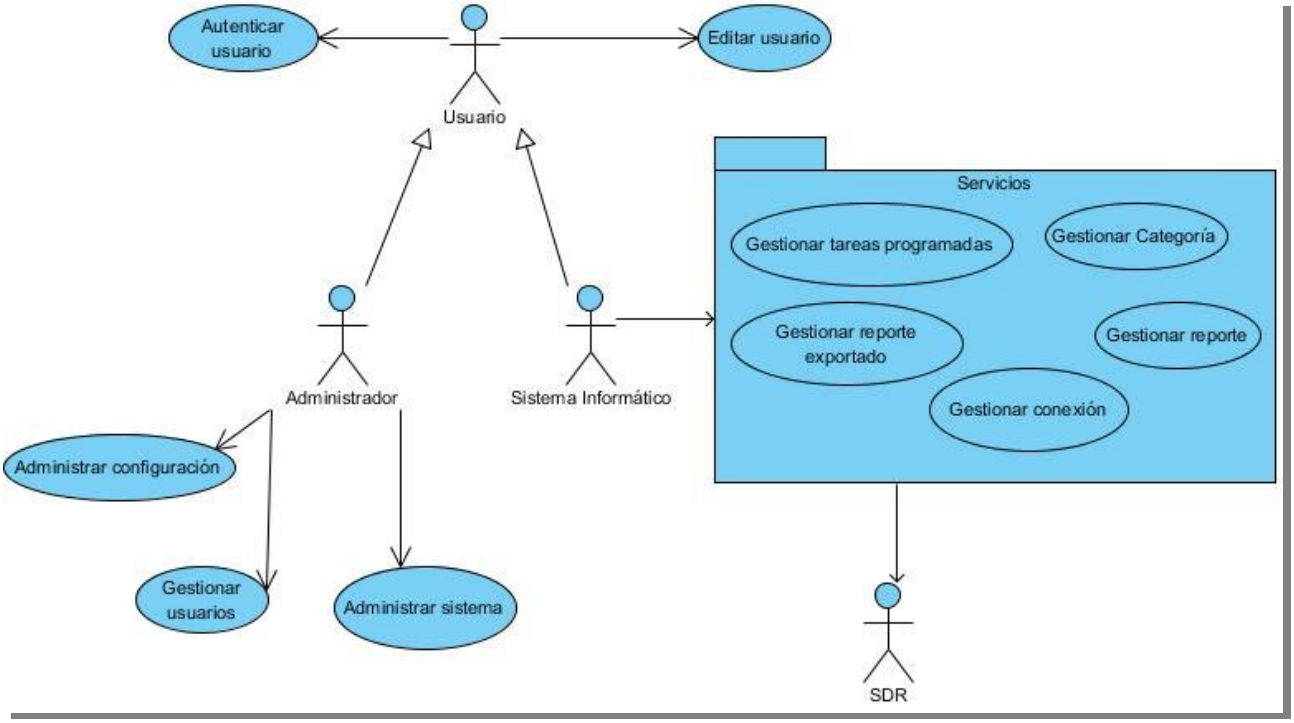

Fig 5-Diagrama de Casos de Uso del Sistema.

Autenticar: Permite a los usuarios del sistema autenticarse en el mismo y el sistema le entrega un token de autentificación.

Editar usuario: Permite a un cliente del sistema editar los siguientes datos del usuario: nombre, contraseña y correo.

Servicios: Conjunto de servicios que brinda el servidor mediante los cuales se realiza la gestión de los recursos del servidor.

Gestión de conexiones: Permite las opciones de listar, crear, actualizar y eliminar las variables de conexión a las fuentes de datos. Serán soportadas las fuentes de datos sobre PostgreSQL, Oracle, MySQL, SQLite, otras que contengan un conector JDBC y sean accesibles desde el servidor.

Gestión de reportes: Permite listar, publicar, actualizar, compilar y eliminar los diseños de reportes en el servidor.

Gestión de reportes exportados: Permite mostrar, listar, buscar y eliminar los reportes exportados del sistema.

Gestión de categorías: Permite crear, actualizar, listar y eliminar las categorías del sistema.

Gestión de tareas programadas: Permite listar, actualizar, crear y eliminar las tareas programadas que se ejecutan en el servidor.

Gestión de usuarios: Permite crear, actualizar, listar y eliminar los usuarios del sistema.

Administrar sistema: Permite para visualizar los reportes del uso el sistema y realizar el mantenimiento del servidor, además de crear, editar, eliminar y listar las consultas de mantenimiento.

Administrar configuración: Permite la visualización de los logs del sistema y la configuración sus parámetros.

2.3.2. Patrones de casos de uso

CRUD completo: Este patrón se evidencia en el CU Gestionar usuarios pues en él se agrupan los siguientes requisitos funcionales: adicionar, modificar, eliminar y visualizar función.

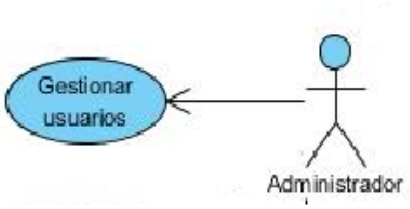

Fig 6-Evidencia del patrón CRUD completo.

Múltiples Actores. Roles Comunes: Los usuarios Administrador y Sistema Informático tienen acceso al CU Autenticar usuario y aplicándole el patrón Roles Comunes queda como se evidencia en la figura 8.

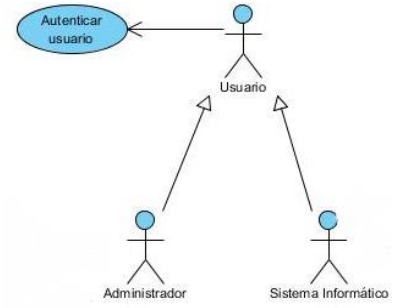

Fig 7-Evidencia del patrón Múltiples Actores. Roles Comunes.

2.3.3. Descripción de los Casos de Uso.

Para consultar los restantes casos de usos ver artefacto “Especificación de Casos de Usos”

Tabla 1 -Descripción parcial del CU2- Gestionar Reportes

Objetivo	Cuando el sistema informático interactúa con el caso de uso persigue el objetivo de operar las funcionalidades relacionadas con los reportes.

Actor	Sistema Informático
Resumen	Agrupa todos los Requisitos Funcionales que satisfacen las necesidades relacionadas con los reportes: adicionar, modificar, buscar, listar, exportar, vista previa, obtener y eliminar.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El usuario ha sido autenticado exitosamente en el sistema y le ha sido asignado un token de autenticación.
Postcondiciones	- Se adiciona el reporte a la base de datos del servidor y queda almacenado en el sistema de ficheros del disco duro. - Se modifican los datos del reporte especificado. - Se devuelve el reporte buscado. - Se listan los reportes que coinciden con los parámetros entrados por el usuario. - Se elimina el reporte de la base de datos y se borran los ficheros relacionados con él.

Flujo de eventos		
Flujo básico Gestionar Reporte		
	Actor	**Sistema**
1.		1. El sistema permite realizar varias acciones con un reporte: - Publicar un reporte. Ver Sección 1: Publicar reporte. - Exportar un reporte. Ver Sección 5: Exportar reporte. Para consultar las restantes secciones ver el artefacto descripción de casos de usos del expediente de proyecto
Sección 1: “Publicar reporte”		
Flujo básico Publicar reporte		
	Actor	**Sistema**
1.	1. Selecciona la opción de adicionar reporte en el módulo de reportes.	2. Muestra una interfaz que cuenta con los campos para publicar un reporte.
2.	3. Rellena los campos para crear un reporte y	4. Valida los datos enviados por parámetros y si son válidos se publica el

	selecciona la opción aceptar.	reporte en el servidor. En caso de que los datos sean incorrectos ver sección ¨**Excepciones en la respuesta¨.**
3.		5. Termina el caso de uso.
Flujos alternos		
2a ¨Excepciones en la respuesta¨		
	Actor	**Sistema**
1		1. El sistema retorna una respuesta de fracaso de la petición, que contiene la excepción ocurrida y el mensaje informativo.
Sección 5: "Exportar reporte"		
Flujo básico Exportar reporte		
	Actor	**Sistema**
1.	1. Selecciona el reporte en la lista de reportes y escoge la opción exportar.	2. Muestra una interfaz que contiene los parámetros del reporte en caso de tener, la conexión y el formato al cual se quiere exportar.
2.	3. Rellena los campos y selecciona la opción aceptar	4. Validad los datos enviados y en caso de estar correctos se exporta el reporte y se muestra el mismo en otra pestaña del navegador. En caso contrario *ver Flujo Alterno 2a. ¨**Excepciones en la respuesta***¨.
3.		5. Termina el caso de uso
Flujos alternos		
2a ¨*Excepciones en la respuesta*.¨		
	Actor	**Sistema**
1.		1. El sistema retorna una respuesta de fracaso de la petición, que contiene la excepción ocurrida y el mensaje informativo.

Relaciones	**CU incluidos**	No aplica
	CU extendidos	No aplica
Requisitos no funcionales	-	
Asuntos pendientes	-	

2.4. Modelo de Diseño

El modelo de diseño describe la realización de casos de uso, y sirve como una abstracción del modelo de implementación y su código fuente. Se utiliza como parte esencial para las actividades en ejecución y prueba, que se basa en el análisis y los requisitos de la arquitectura del sistema. Representa los componentes físicos y determina su colocación adecuada y el uso dentro de la arquitectura en general del sistema.

2.4.1. Patrones arquitectónicos.

Patrón n Capas, 2 Capas.

Con la arquitectura cliente/servidor en dos capas se añade una capa entre el cliente y el servidor donde se implementa la lógica de la aplicación. De esta forma el cliente es básicamente una interfaz, que no tiene por qué cambiar si cambian las especificaciones de la base de datos o de la aplicación; queda aislado completamente del acceso a los datos.

Capa de presentación: Esta capa es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable para el usuario generalmente se presentan como formularios.

Capa de Negocio: Aquí es donde, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.(Macias 2013)

2.4.2. Diagramas de Clases del Diseño

*"*El diagrama de clase del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información: clases, asociaciones, atributos, interfaces con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad y dependencias." *(Larman 2004)*

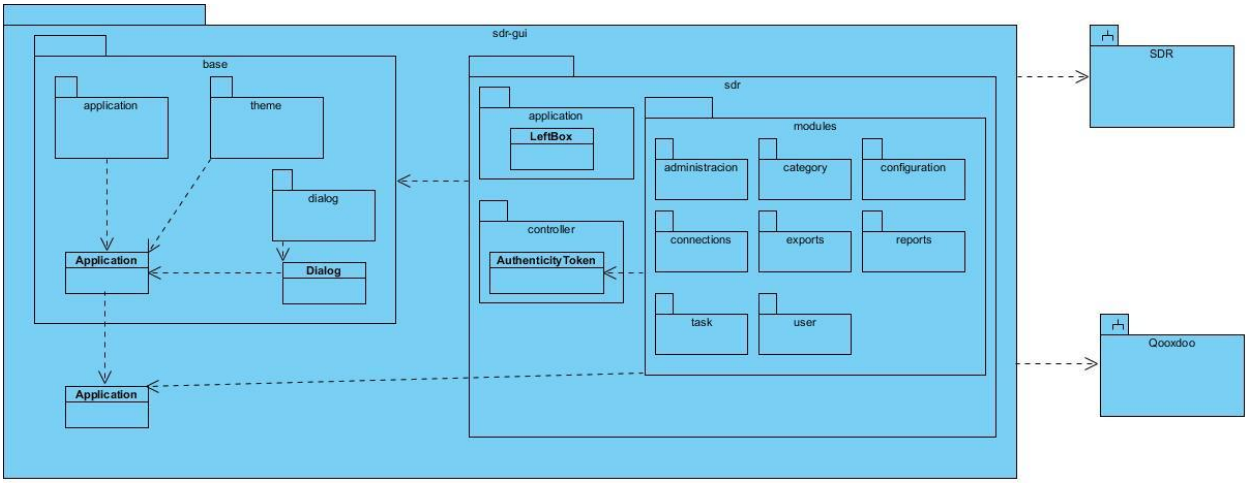

Fig 8- Diagrama de Paquetes del Sistema de Administración del SDR.

La Figura 9 muestra cómo se encuentra estructurado el Sistema de Administración del SDR. Este está dividido en 2 capas, la primera es la capa de presentación la cual está representada por el paquete sdr-gui, la segunda es el subsistema SDR la cual corresponde a la capa de negocio. El paquete sdr-gui cuenta con 2 paquetes principales, el paquete base y el paquete sdr. En el paquete base se encuentran todas las clases de las cuales van a extender todos los módulos de la aplicación que están ubicados en el paquete sdr, además contiene otras clases la cual sirven de soporte para la aplicación. En el paquete sdr se encuentran todos los módulos del sistema los cuales están agrupados en el paquete module. Toda la aplicación está desarrollada utilizando el *framework* Qooxdoo por lo que utiliza sus su estructura y clases.

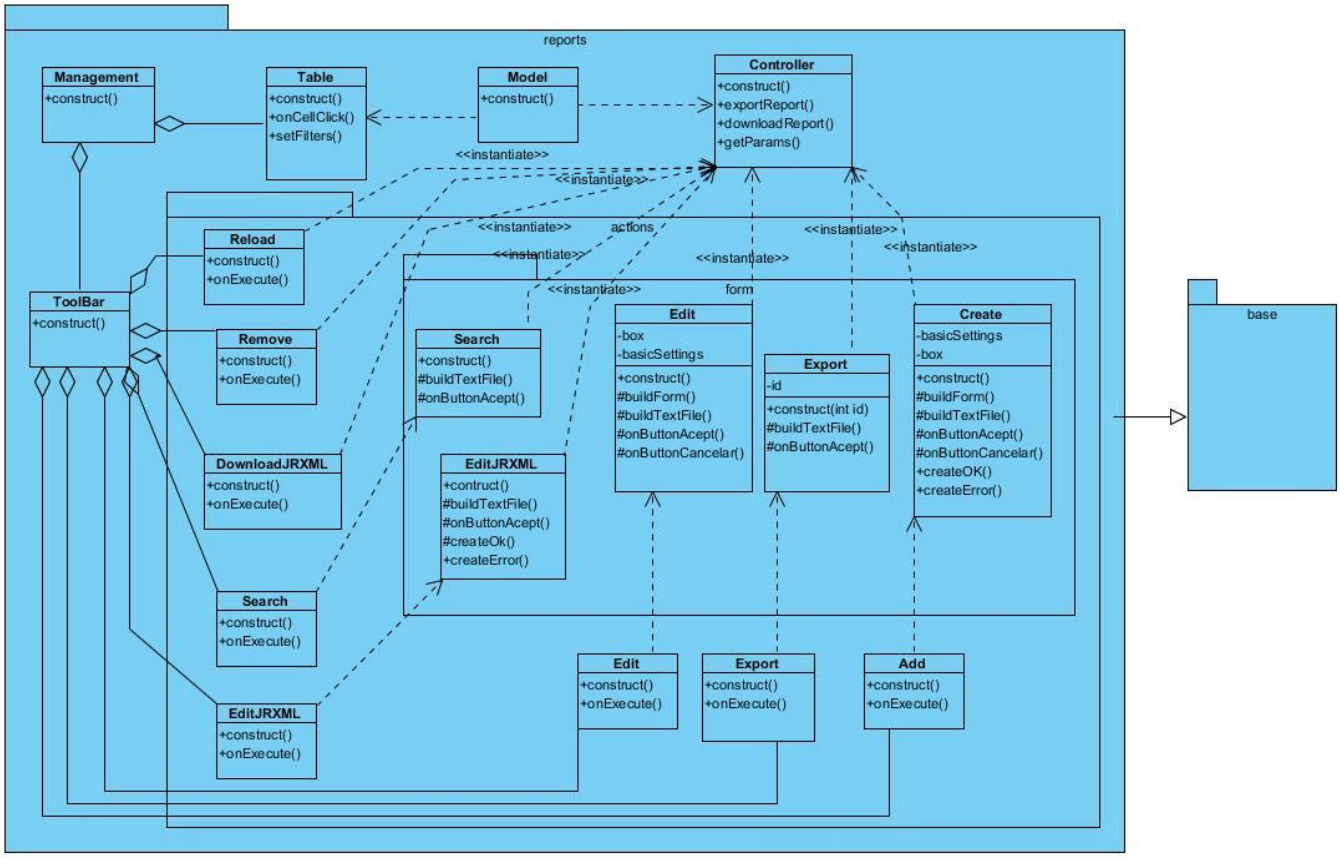

Fig 9- Diagrama de clases del diseño de CU5-Gestionar reporte.

En la Figura 10 se evidencian las clases que componen el caso de uso Gestionar Reporte. En el paquete report es en donde se encuentran todas las clases necesarias para gestionar los reportes. La clase **Management** está compuesta por las clases **ToolBar** y **Table**. En clase **ToolBar** es donde se construye el menú para realizar las acciones, la misma está compuesta por las clases **Search**, **Remove**, **Reload**, **Export**, **Edit** y **Add** del paquete action. La clase **Table** es la encargada de mostrar el contenido de los datos del usuario y en esta se inicializa la clase **Model** que es la que da el formato de cómo queda estructurado el contenido a mostrar. La clase **Model** obtiene sus datos de la clase Controller que es la encargada de interactuar con los servicio del servidor (SDR). En el paquete action se encuentran las clases para realizar las operaciones pertinentes sobre la tabla del contenido, la clase **Search** es la encargada de buscar los elementos según la opción pasada por parámetros, la clase **Reload** recarga el contenido de la tabla, la clase **Remove** envía la petición de eliminar un elemento seleccionado a la clase Controller, la clase **Export** exporta el reporte seleccionado utilizando a la clases **Export** del paquete form

para mostrar los parámetros del reporte a exportar, la clase **Edit**, edita los valores del elemento seleccionado utilizando la clase **Edit** del paquete form y posteriormente mandándole los datos a la clase Controller y la clase **Add**, adiciona nuevos elementos utilizando la clase **Add** del paquete form y posteriormente mandándole estos datos a la clase **Controller**. Todas las clases del paquete report extienden de sus respectivas clases en el paquete base.

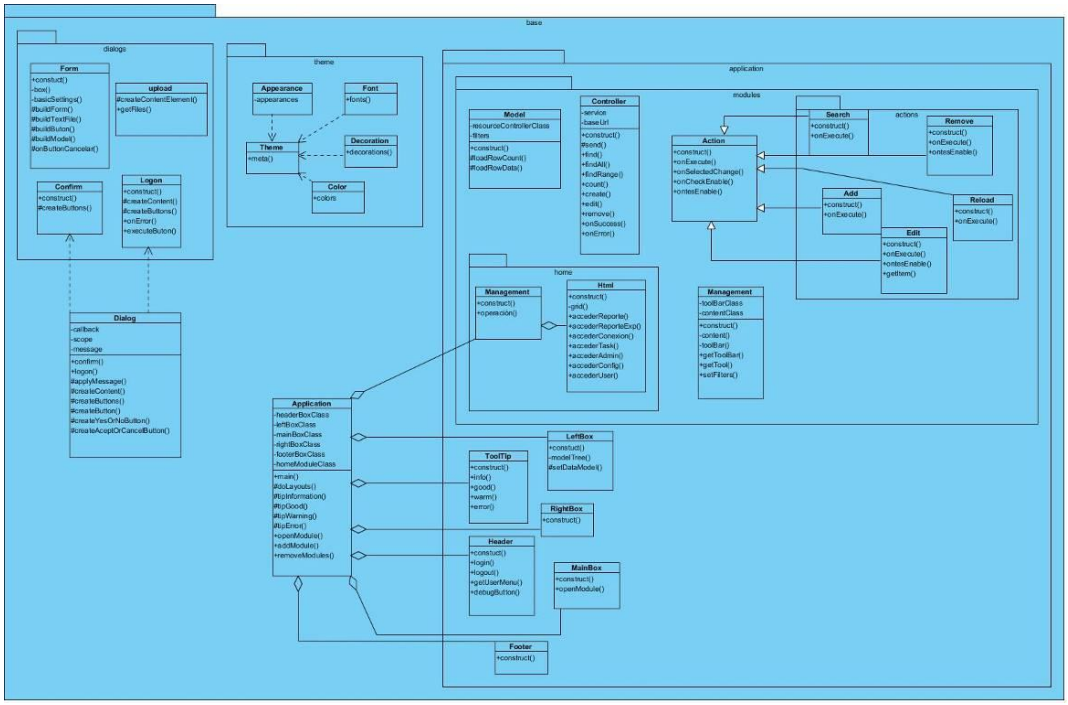

Fig 10- Paquete base SDR-GUI.

2.5. Patrones utilizados en la solución.

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas.(Larman 2004)

2.5.2. Patrones de diseño

Patrones GRASP

En el caso de los patrones GRASP (*General Responsibility Assignment Software Patterns*), como su nombre indica, describen la asignación de responsabilidades a objetos. A continuación se muestran los patrones de esta clasificación que fueron utilizados:

- **Patrón Experto**

  "*Este patrón se aplica para asignar responsabilidades a aquellas clases que contienen la información necesaria para cumplir con dicha responsabilidad*". (Larman 2004)

  El uso de este patrón se aprecia en la clase **AutenticityToken** la cual es la clase que tiene toda la información necesaria para realizar la autenticación del usuario en el sistema.

Fig 11-Evidencia del patrón Experto.

- **Patrón Creador**

  "*El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento".* (Larman 2004)

  El uso de este patrón se aprecia en la clase **Application** del módulo base la cual es la encargada de crear las instancias de las clases **Management** para cargar los módulos del sistema.

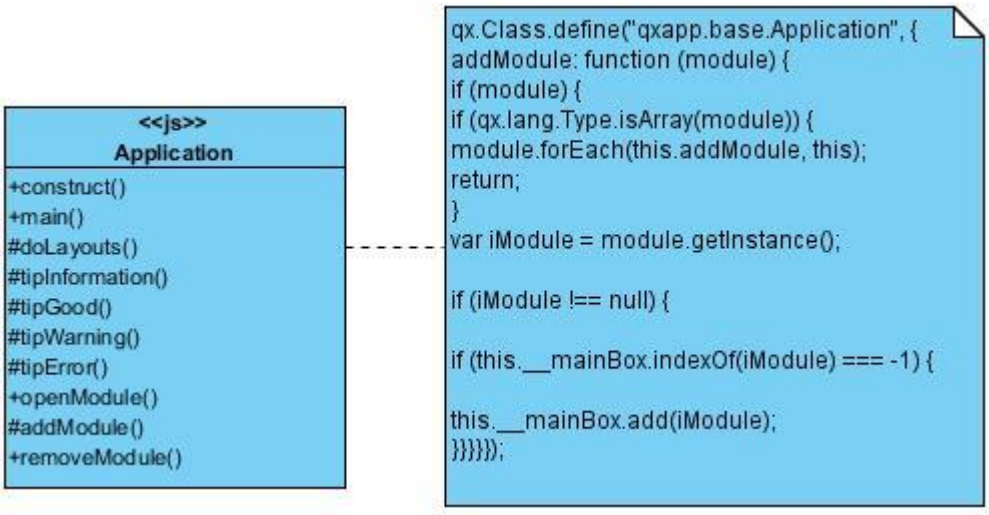

Fig 12-Evidencia del patrón Creador.

- **Patrón Bajo Acoplamiento**

  "*Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases".* (Larman 2004) El uso de este patrón se aprecia en el diseño del módulo reports de donde cada clase tiene una responsabilidad específica, de tal forma que si se modifica algunas de estas no se afecta de ninguna manera la funcionalidad de la otra. Ejemplo: La clase **Management** contiene las instancias de las clases **ToolBar** y **Table**, las modificaciones que se realicen en estas dos últimas clases no repercuten en la implementación de la clase **Management**.

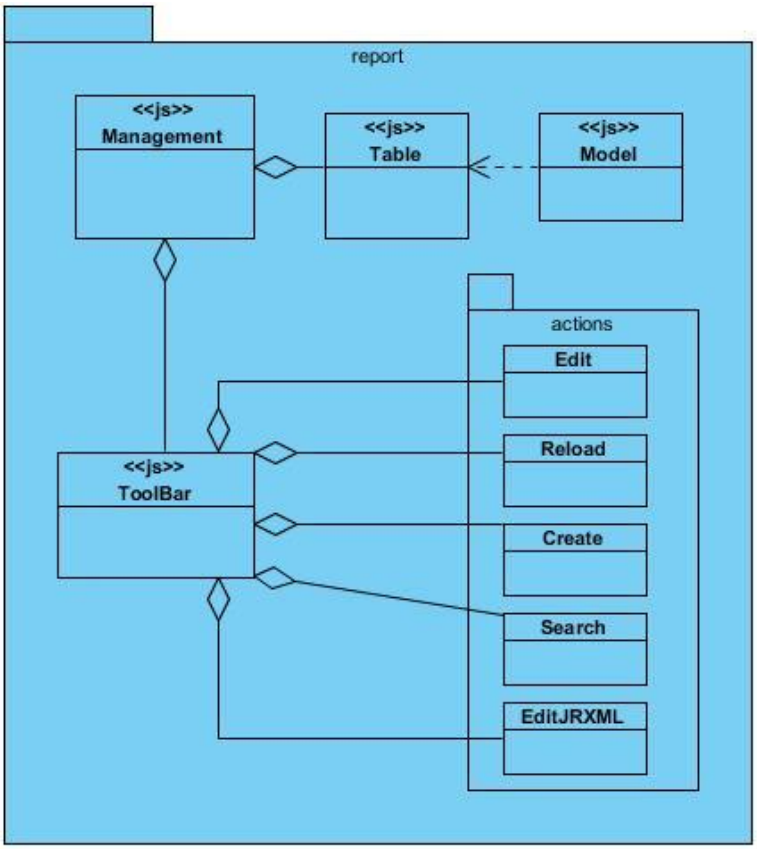

Fig 13-Evidencia del patrón Bajo Acoplamiento.

- **Patrón Alta Cohesión**

  "Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande".(Larman 2004)

  Este patrón se puede evidenciar en el módulo report ya que la clase **Management** es la vía de acceso a todas las funcionalidades del módulo y solo tiene la tarea de instanciar a las clases **ToolBar** y **Table.**

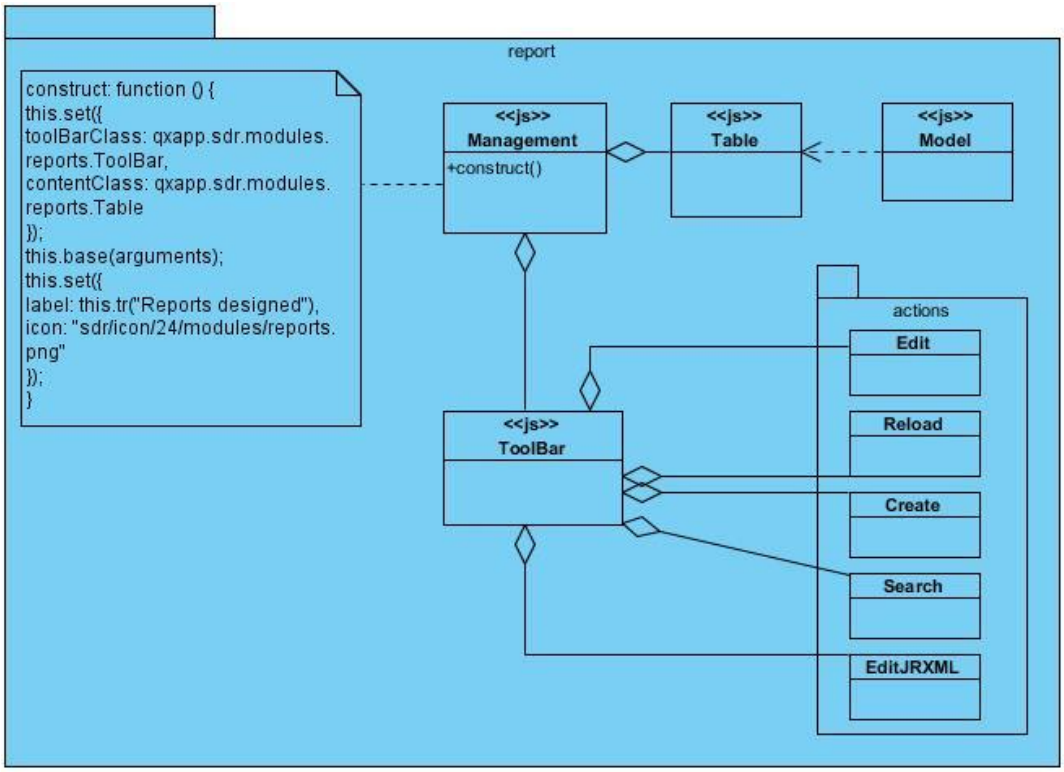

Fig 14- Evidencia del patrón Alta Cohesión.

- **Patrón Controlador**

  "*El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado".* (Larman 2004)

  Esto se puede observar en la clase **Controller** la cual es la encargada de consumir los servicios del SDR y enviarle a las clases las respuestas de las peticiones que realizaron.

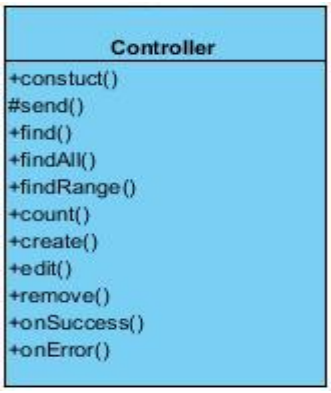

Fig 15-Evidencia del patrón Controlador.

- **Patrón Polimorfismo**

  "*Siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo, cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asigne la responsabilidad para el comportamiento- utilizando operaciones polimórficas- a los tipos para los que varía el comportamiento. Asigna el mismo nombre a servicios en diferentes objetos*". (Larman 2004)

  El uso de este patrón se aprecia en las clases **Create** de los módulos report y connections las cuales extienden de la clase base **Form** y modifican sus métodos y agregan métodos propios.

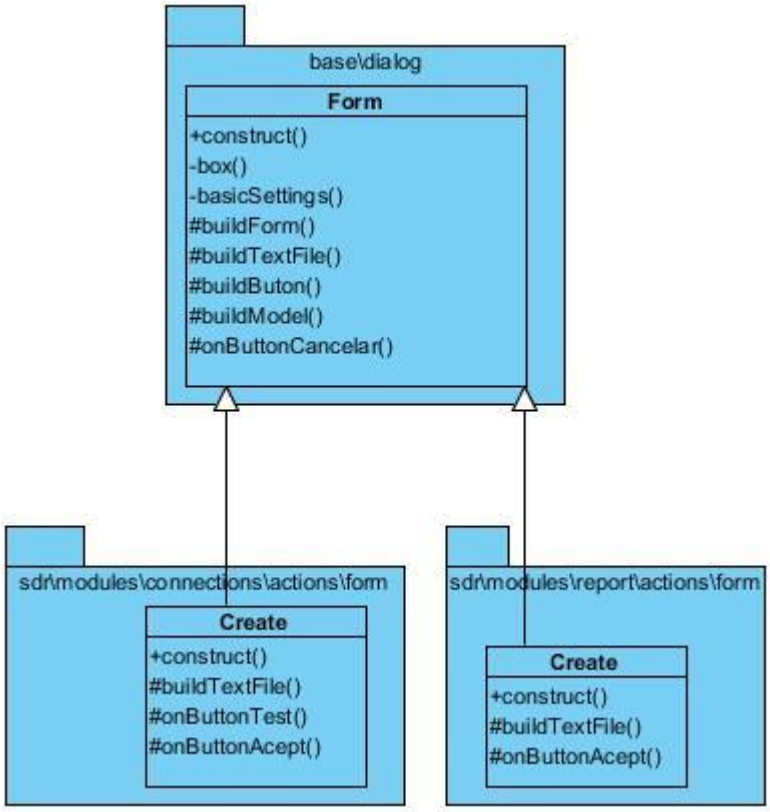

Fig 16- Evidencia del patrón Polimorfismo.

Patrones GOF

Dentro de los patrones de diseño se encuentran los patrones Gang of Four o Grupo de los Cuatro. Estos patrones definen el comportamiento entre las clases y los objetos. Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento. En el desarrollo de la aplicación se utilizaron los siguientes patrones GOF:

- **Patrón Builder**

  "Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones". (Larman 2004)

  El uso de este patrón se aprecia en la clase **LeftBox** del paquete sdr la cual extiende de la clase **LeftBox** del paquete base que es donde se encuentran declaradas las principales funcionalidades.

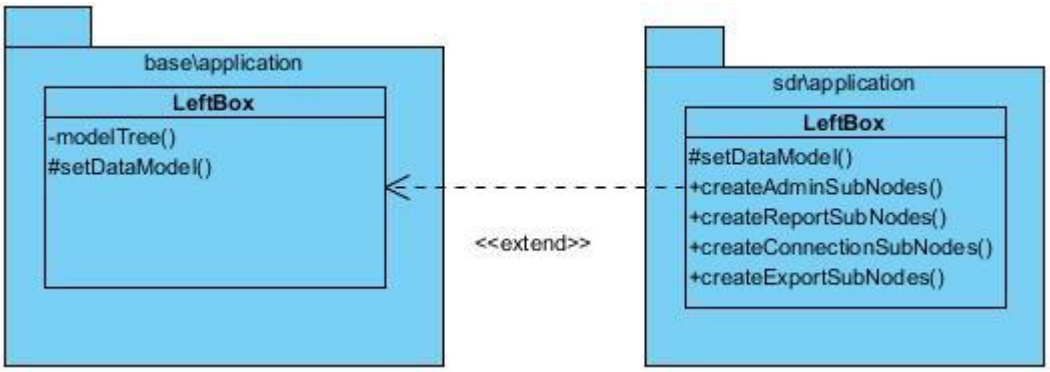

Fig 17- Evidencia del patrón Builder.

- **Patrón Composite**

"Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos". (Larman 2004)

El uso de este patrón se aprecia en la clase **Application** ya que en esta se construyen todos los módulos que constituyen la aplicación.

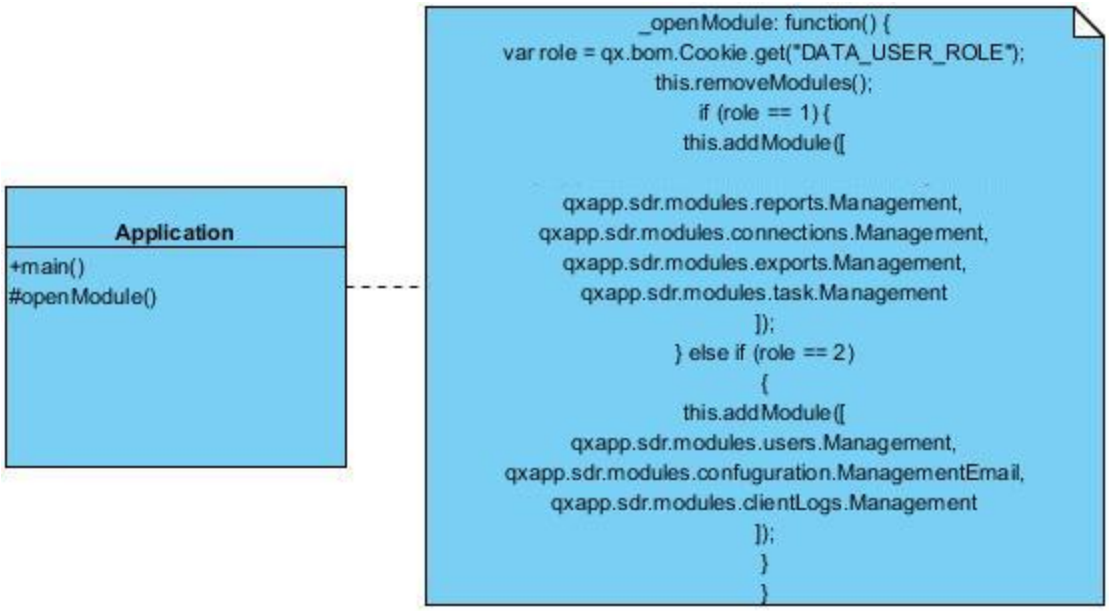

Fig 18-Evidencia del patrón Composite.

- **Patrón Singleton**

  Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. (Larman 2004)

  El uso de este patrón se apreciar en la clase Controller del módulo report mediante la cual se accede a los servicios que brinda el SDR, las clases pueden utilizarla para acceder a estas funcionalidades instanciándola ya que el *framework* Qooxdoo la reconoce como **singleton** y existe siempre solo una instancia de la clase.

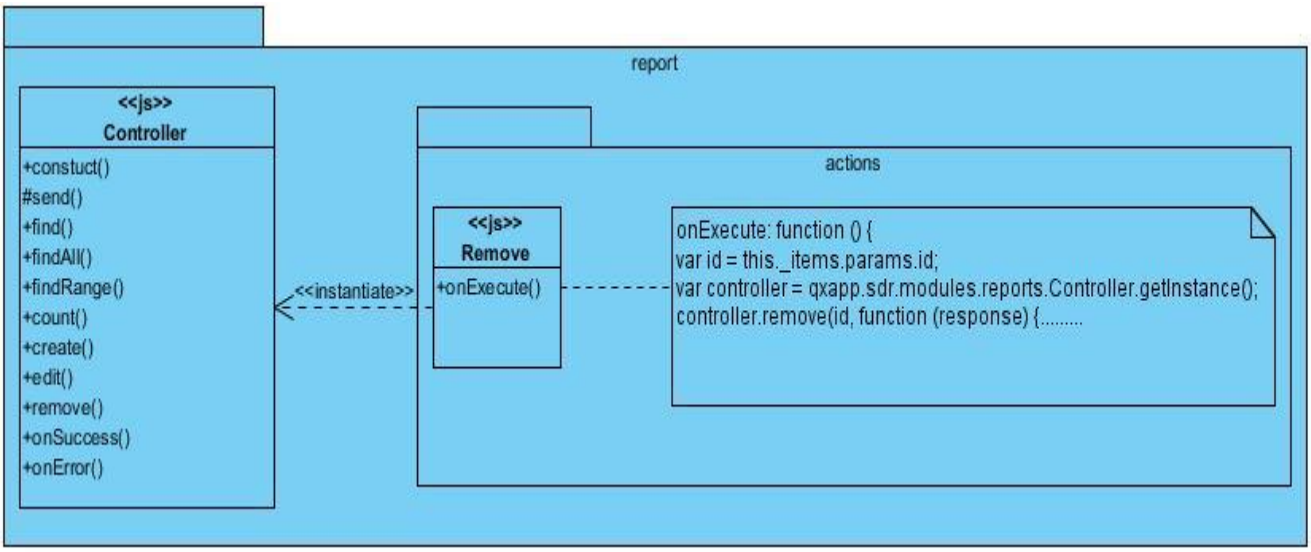

Fig 19-Evidencia del patrón Singleton.

2.6. Modelo de Despliegue

*"*Los elementos de diseño al nivel del despliegue indican cómo se ubicarán la funcionalidad y los subsistemas dentro del entorno computacional físico que soportará al software*." (Pressman 2011)*

Este diagrama muestra el entorno computacional, pero no indica de manera explícita detalles de configuración.

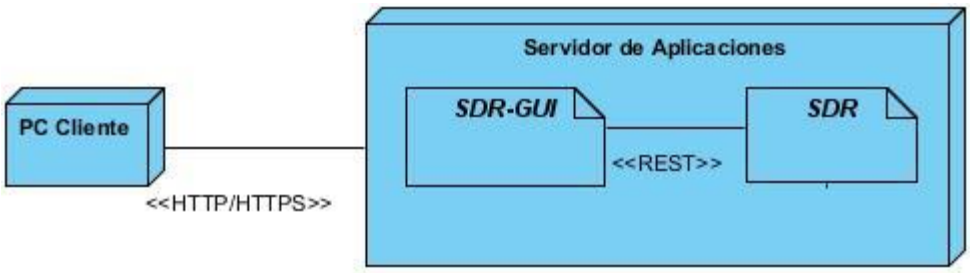

Fig 20- Modelo de despliegue del Sistema de Administración del SDR.

En el diagrama que se muestra en la figura 21 se representan los 2 nodos principales para el despliegue del sistema. El nodo PC_Cliente representa la computadora del usuario desde donde se accederá mediante al SDR-GUI. Esta estación de trabajo deberá contar con un navegador capaz de soportar JavaScript. El nodo Servidor de Aplicaciones representa al servidor donde se encontrará publicado el SDR-GUI y el SDR. La comunicación entre el nodo PC_Cliente y Servidor de Aplicaciones será a través del protocolo de transferencia de hipertexto HTTP (por su siglas en inglés *Hypertext Transfer Protocol*) o a través del protocolo de transferencia de hipertexto en modo seguro HTTPS (por su siglas en inglés *Hypertext Transfer Protocol Secure*). La comunicación entre el SDR-GUI y el SDR se realizará utilizando el servicio representacional de estado (REST).

2.7. Conclusiones del Capítulo.

En el presente capítulo se realizó el modelo de dominio del Sistema de Administración del SDR, con el cual se logró exponer los conceptos claves del sistema y las relaciones existentes entre las entidades comprendidas en el ámbito del dominio del problema. Además se definieron los requisitos funcionales y no funcionales de la aplicación para especificar las características y capacidades que debe cumplir el sistema. Asociado a esto y para lograr un mayor nivel de detalle, se realizó la descripción de CU. En correspondencia con esta descripción, se realizó el modelo de diseño empleando patrones de diseño y patrones arquitectónicos, para mostrar la estructura interna de la aplicación así como los métodos que se deben implementar para darle cumplimiento a los requisitos definidos.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE ADMINISTRACIÓN DEL SDR

Introducción

En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado. Como parte del mismo se presenta el diagrama de componentes del CU Gestionar Reportes y los estándares de codificación utilizados para el desarrollo de la aplicación. Además se describe la estrategia de pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento del Sistema de Administración del SDR en distintos momentos del ciclo de vida del software.

3.1. Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se puede encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, agrupándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos. Para representar los diagramas del modelo de implementación se puede emplear el diagrama de componentes.(Sommerville 2005)

3.1.1 Diagrama de componentes

Un diagrama de componente es utilizado para representar la separación de un sistema de software en componentes físicos y mostrar las dependencias entre estos. Estos componentes incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes.

En la figura 22 que se muestra a continuación se presenta el diagrama de componente del CU Gestionar Reporte. El mismo está divido en 2 paquetes de implementación básicos:

- Módulo reports: agrupa todos los componentes relacionados con la gestión de los reportes.
- Base: paquete que contiene todos los elementos básicos que componen a un módulo.

Además se muestran los subsistemas con los que interactúa el sistema:

- SDR: Representa a los servicios que brinda el Servidor Dinámico de Reporte los cuales son consumidos por el sistema.
- Qooxdoo: Framework JavaScript en el cual está basado el sistema.

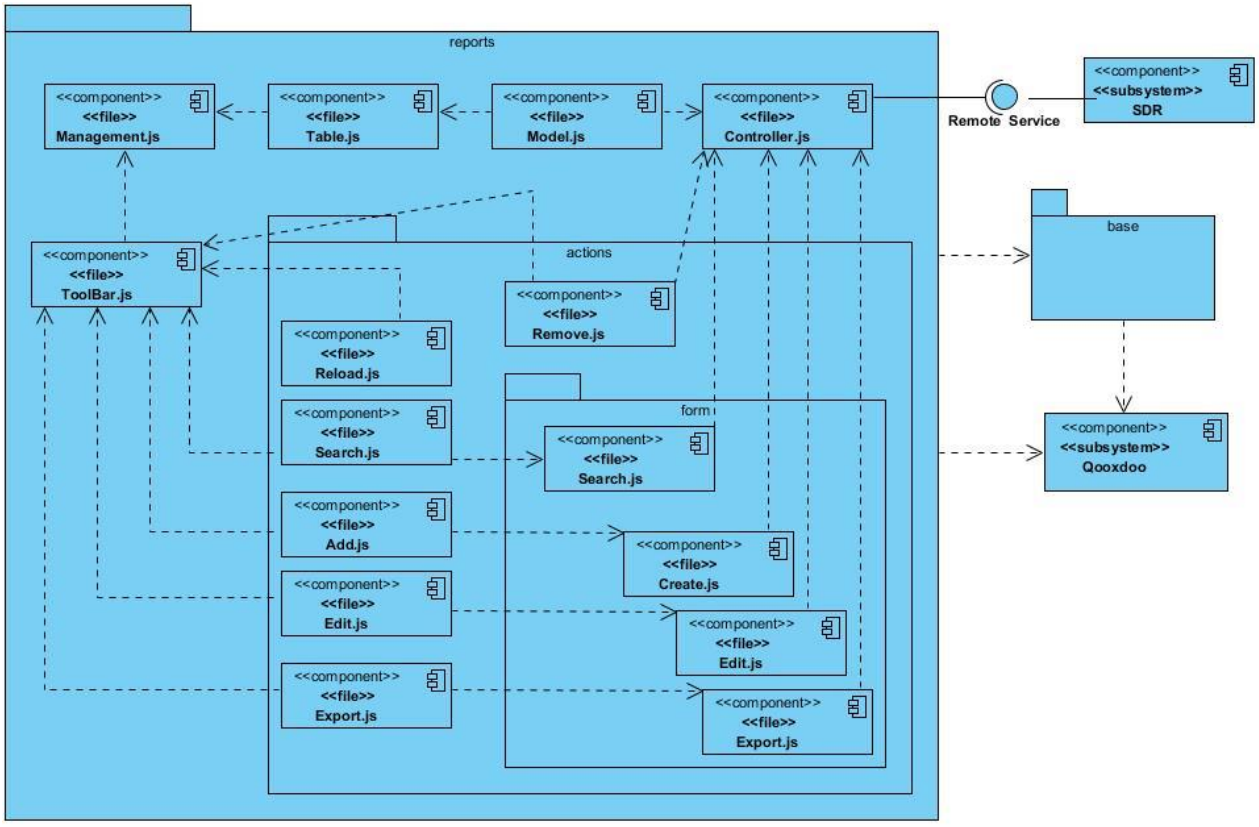

Fig 21- Diagrama de componentes del CU Gestionar Reportes.

3.2. Código Fuente

El código fuente de un programa informático (o software) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Este describe las funcionalidades que debe realizar el software.

En el siguiente fragmento de código se muestra a la clase **Controller** del módulo de reportes esta es una de las más importantes del sistema ya que contiene los métodos para exportar reportes, devolver parámetros de los reportes y crear reportes, dichas funcionalidades son las más críticas del sistema.

```
members: {
    exportReport: function (pData, pCallBack, pScope) {
        // Call remote service
        var action = qx.bom.Template.render("{{service}}/exports", {
            service: this.getService()
        });
        this.setMethod("POST");
        this._send(action, pData, pCallBack, pScope);
    },
    downloadReport: function (pId,pToken, pCallBack, pScope) {
        // Call remote service
        var action = qx.bom.Template.render("{{service}}/download/{{id}}/{{token}}", {
            service: this.getService(),
            id: pId,
            token: pToken
        });
        this.setMethod("GET");
        this._send(action,null, pCallBack, pScope);
    },
```

Fig 22- Fragmento de código de la clase Controller del Módulo Reports.

3.2.1 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armónico, como si un único programador hubiera escrito todo el código de una sola vez. Posibilita que un equipo de programadores mantenga un código de calidad sobre el que se efectuarán luego revisiones.

CamelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas, dentro de este estilo existen dos tipos:

- **UpperCamelCase:** Las palabras que forman el nombre se escriben juntas y la primera letra de cada una de ellas en mayúscula.

- **lowerCamelCase:** Las palabras que forman el nombre se escriben juntas, la primera letra de la primera palabra en minúscula y del resto de las palabras, la primera letra en mayúscula.
- **Estilo de codificación utilizado**
- Los nombres de las clases deben ser escritos utilizando el estándar de codificación UpperCamelCase.

- Los nombres de las funciones deben ser lo suficientemente elocuentes como para describir su propósito y comportamiento. Los nombres de las variables deben ser siempre claros, evidentes y prácticos para describir los datos que el desarrollador pretende almacenar en ellas. Sus nombres deberán estar escritos utilizando el estándar de codificación lowerCamelCase.
- Los comentarios de implementación están delimitados por /*...*/ o //.
- Todas las funciones y clases estarán comentadas para explicar el propósito de las mismas.

A continuación se ejemplifican algunos de los estilos antes descritos

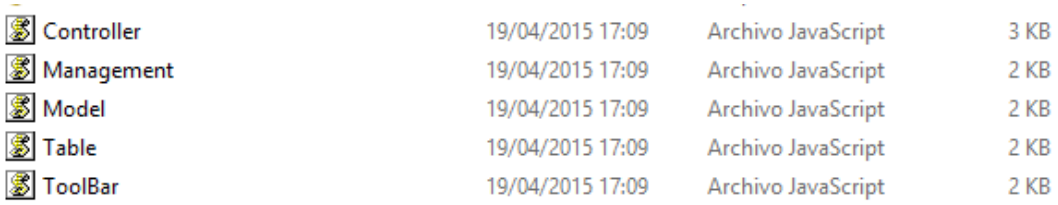

Fig 23- Nombre de las clases con estándar de codificación UpperCamelCase.

```
exportReport: function (pData, pCallBack, pScope) {
    // Call remote service
    var action = qx.bom.Template.render("{{service}}/exports", {
        service: this.getService()
    });
    this.setMethod("POST");
    this._send(action, pData, pCallBack, pScope);
},
getParams: function (pId, pCallBack, pScope) {
    // Call remote service
    var action = qx.bom.Template.render("{{service}}/params/{{id}}", {
        service: this.getService(),
        id: pId
    });
    this.setMethod("GET");
    this._send(action, null, pCallBack, pScope);
},
```

Fig 24-Nombre de funciones con estándar de codificación lowerCamelCase.

3.3. Pruebas de Software

Una vez que el código fuente ha sido generado, el software debe ser probado para descubrir tantos errores como sea posible antes de la entrega a su cliente. El objetivo de las pruebas es diseñar una serie de casos de prueba que tienen una alta probabilidad de encontrar errores. Estas técnicas proporcionan

una guía sistemática para el diseño de pruebas de la lógica interna y las interfaces de todos los componentes de software y el ejercicio de los dominios de entrada y salida del programa para descubrir errores en función del programa, el comportamiento y el rendimiento.(Pressman 2011)

3.3.1. Niveles de Prueba

A la hora de evaluar dinámicamente un sistema software se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos niveles de trabajo, dentro de estos se distinguen:

- Pruebas de desarrollador
- Pruebas independientes
- Pruebas de unidad
- Pruebas de integración
- Pruebas al sistema
- Pruebas de aceptación

Los niveles que se han seleccionado para llevar a cabo en la aplicación son:

- **Pruebas de unidad**

Son diseñadas y ejecutadas por el desarrollador una vez concluido el desarrollo de cada componente. Es la prueba que valida si los componentes individuales de un programa están funcionando adecuadamente.(Pressman 2011)

- **Pruebas de sistema**

Son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio. Las pruebas del sistema examinan qué tan bien el sistema cumple con los requerimientos de la organización y su utilidad, seguridad y desempeño. También se realizan estas pruebas a la documentación del sistema. Concretamente se debe comprobar que:

- Se cumplen los requisitos funcionales establecidos.
- Sea correcto el funcionamiento y rendimiento de las interfaces hardware, software y de usuario.
- Sea apropiada la documentación de usuario.
- Se verifique el rendimiento y respuesta en condiciones límite y de sobrecarga.(Pressman 2011)

- **Pruebas de Aceptación**

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Las pruebas de aceptación son realizadas en dos etapas, una llamada *alpha*, en la cual los usuarios prueban el sistema usando datos de prueba, y la segunda llamada *beta* en la cual los usuarios empiezan a utilizar el sistema con los datos reales, pero siendo aún monitoreados cuidadosamente previendo que se presenten errores.

3.3.2. Técnicas de pruebas

Cada nivel de prueba engloba una técnica de prueba específica según los atributos de calidad que se deseen verificar con las pruebas al software.

Pruebas de Funcionalidad

Entre las técnicas de pruebas que se realizan en un sistema está la que evalúa la funcionalidad de éste. Se pueden tener distintos tipos de prueba, según los parámetros de evaluación que abarca la técnica entre estas están:

- **Prueba funcional.**

Objetivo: Asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Metas**:** Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio y la apropiada aceptación de datos.

Enfoque**:** Los requisitos funcionales y las reglas del negocio.

Método**:** Caja Negra: Se ejecuta cada caso de uso, flujo de caso de uso, o función, usando datos válidos e inválidos, para verificar lo siguiente:

- Que se aplique apropiadamente cada regla de negocio.
- Que los resultados esperados ocurran cuando se usen datos válidos.
- Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

- **Prueba de seguridad.**

Objetivo:

- Nivel de seguridad de la aplicación: Verifica que un actor sólo pueda acceder a las funciones y datos que su usuario tiene permitido.

- Nivel de Seguridad del Sistema: Verifica que sólo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

Áreas:

- Seguridad del sistema, incluyendo acceso a datos o funciones de negocios.
- Seguridad del sistema, incluyendo ingresos y accesos remotos al sistema.

Garantiza:

- Que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que está autorizado a acceder.
- Que sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema.

Técnicas:

- Identificar cada tipo de usuario y las funciones y datos a los que se debe autorizar.
- Crear pruebas para cada tipo de usuario y verificar cada permiso, creando transacciones específicas para cada tipo de usuario.

Criterio de completitud.

Para cada tipo de usuario conocido, las funciones y datos apropiados y todas las transacciones funcionan como se esperaba.

3.3.3. Métodos de Prueba

- **Método de pruebas de caja blanca**

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba.(Pressman 2011)

La pruebas de caja blanca intentan garantizar que: se ejercite por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites; y ejerciten las estructuras internas de datos para asegurar su validez.(Pressman 2011)

- ✓ **Técnica camino básico**

Una de las técnicas de prueba de caja blanca más usada es el camino básico, la cual determina la complejidad ciclomática de una porción de código. La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa.(Pressman 2011)

Cuando se usa el camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar.(Pressman 2011)

- ❖ **Método de pruebas de caja negra**

La prueba de la caja negra es un método de diseño de casos de prueba que se centra en los requisitos funcionales del software, por lo que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se acepten de forma adecuada y que la integridad de la información externa se mantenga.(Pressman 2011)

La prueba de caja negra intenta identificar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.(Pressman 2011)

- ✓ **Técnica particiones equivalentes**

Una de las técnicas de prueba de caja negra más usada es la partición equivalente, la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.(Pressman 2011)

3.3.4. Diseño de casos de prueba

Se trata de diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. (Pressman 2011)

En la prueba de la caja negra, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.(Pressman 2011)

3.3.5. Proceso de prueba del Sistema

Con la intención de poder descubrir y corregir el máximo de errores posibles en la herramienta desarrollada, se decidieron realizar las pruebas siguientes. A nivel de unidad se aplicara el método de prueba de caja blanca utilizando la técnica de camino básico, además de las pruebas funcionales que se realizaron a medida que se implementaban las funcionalidades. A nivel de sistema se aplicaran las

técnicas de pruebas de funcionalidad donde se utilizará el método de caja negra y dentro de este la técnica de particiones equivalentes para verificar el correcto funcionamiento de la aplicación, además se realizarán las pruebas de seguridad para verificar que en la aplicación todos los usuarios accedan a las funcionalidades que les corresponde según su rol. A nivel de aceptación se realizaran pruebas tipo *alpha* las cuales serán ejecutadas con el cliente y su grupo de trabajo.

3.3.6. Aplicación de las pruebas de seguridad.

Para ejecutar las pruebas de seguridad se realizarán un conjunto de casos de pruebas para comprobar que cada usuario tenga acceso solo a las funcionalidades que les fueron definidas. En cada escenario del caso de prueba se probará el acceso a cada módulo utilizando los 2 roles existentes en el sistema (Administrador y Sistema Informático).

Para ver los casos de pruebas de seguridad realizados, comprobar el artefacto “Diseño de Casos de Prueba de Seguridad” el cual se puede consultar en el expediente de proyecto.

3.3.7. Aplicación de las pruebas de caja blanca.

Técnica de camino básico.

Como se mencionó en acápites anteriores, una de las técnicas de prueba de caja blanca más usada es la de camino básico, que se aplica a fragmentos de código. En el proceso de prueba para validar la herramienta desarrollada se decidió aplicar esta técnica a unos de los métodos del caso de uso significativo “Exportar reporte”.

Al aplicar la técnica de camino básico al código del método **_buidTextFile** se identificaron 9 bloques de ejecución, los cuales se enumeraron (Véase, figura 26) para que puedan ser reconocidos. Además se determinó el camino básico que se muestra en la figura 27, donde existe un nodo predicado del cual se derivan más de un camino a seguir, tal es el caso del nodo 2 y las aristas indican los posibles caminos a seguir a partir del nodo correspondiente.

```
_buildTextFile: function () {
        var form = new qx.ui.form.Form();
        form.addGroupHeader(this.name);
        var listTextFild = null;
1 →     this.type=[];
        var action = qxapp.sdr.modules.reports.Controller.getInstance();
        action.getParams(this.id, function (response) {
            listTextFild = response;
            this.validator = new qx.ui.form.validation.Manager();
2 →         for (var i = 0; i < response.items.length; i++) {
3 →             var item = response.items[i];
4 →             if (item.type == "java.util.Date" || item.type == "java.util.Datetime") {
                    this.type.push(item.type.split('.')[2]);
                    var time = new qx.ui.form.DateField();
5 →                 time.setRequired(true);
                    time.setMinWidth(200);
                    form.add(time, item.name, null, item.name);
                    this.validator.add(time);
                }
                else {
                    this.type.push(item.type.split('.')[2]);
                    var name = new qx.ui.form.TextField();
6 →                 name.setRequired(true);
                    name.setMinWidth(200);
                    form.add(name, item.name, null, item.name);
                    this.validator.add(name);
                }
            }
            var items = Array();
            var controller = qxapp.sdr.modules.connections.Controller.getInstance();
7 →         controller.findAll({}, function (response) {
                items = response.items;
            });
            var connection = new qx.ui.form.SelectBox();
8 →         for (var i = 0; i < items.length; i++) {
9 →             connection.add(new qx.ui.form.ListItem(items[i].name, null, items[i].id));
            }
            form.add(connection, this.tr("Connection"), null, "idConnection");
10 →    }, this);

        var format = new qx.ui.form.SelectBox();
        var formats = ["pdf", "html", "pptx", "docx", "rtf", "odt", "ods", "xslx", "xls", "cvs", "
11 →    for (var i = 0; i < formats.length; i++) {
12 →        format.add(new qx.ui.form.ListItem(formats[i], null, formats[i]));
        }
        form.add(format, this.tr("Formats"), null, "format");

        this._buildButon(form);
        this.form = form;
13 →    // binding ////////////////////////
        var controller = new qx.data.controller.Form(null, form);
        var model = this.model = controller.createModel();

    }
```

Fig 25- Código fuente del método _buildTextFile.

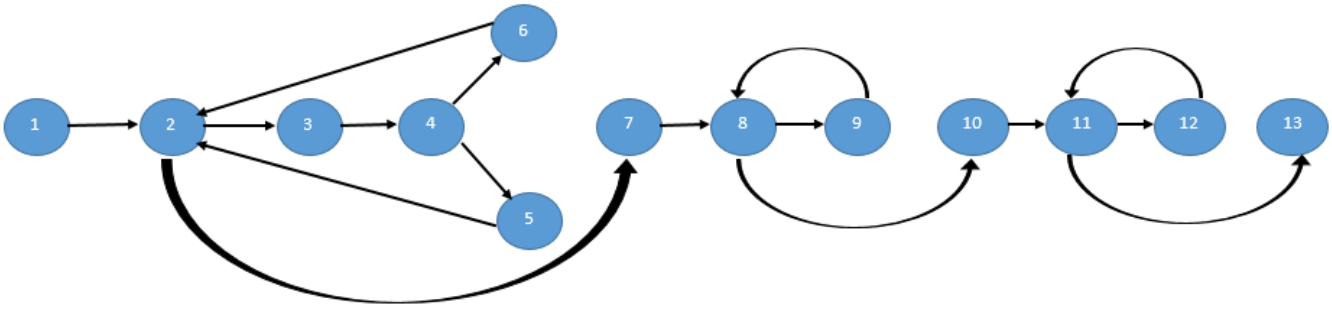

Fig 26- Grafo de flujo del método _buildTextFile.

Se procede entonces al cálculo de la complejidad ciclomática del grafo de flujo (VG). Para esta operación existen tres vías de solución, las cuales se enuncian a continuación:

- VG = (A – N) + 2
- VG = P + 1
- VG = R

Siendo A la cantidad total de aristas, N la cantidad total de nodos, P la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas) y R la cantidad total de regiones. Se utilizó la fórmula V (G) = A-N + 2, para la cual se identificaron 16 artistas y 13 nodos, por lo tanto: V (G) = 16-13 + 2, quedando V (G) = 5. De la misma forma se pueden comprobar que las otras variantes explicadas de calcular la complejidad ciclomática arriban al mismo resultado.

Después de analizar los resultados se puede comprobar que el sistema no cuenta con algoritmos de alta complejidad, ya que el método analizado anteriormente tiene una complejidad ciclomática baja y el mismo es uno de los más complejos del sistema.

3.3.8. Casos de Prueba de caja negra.

Para aplicar las pruebas al sistema se determinaron los casos de pruebas, para verificar el cumplimiento de un objetivo en particular.

Un caso de prueba se diseña según las funcionalidades descritas en los casos de uso. El propósito que se persigue con este artefacto es lograr una comprensión común de las condiciones específicas que la solución debe cumplir. Se parte de la descripción de los casos de uso del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. Se efectuaron los casos de pruebas a los 10 casos de uso del sistema, plasmándose en la documentación del proyecto. (Ver planilla Diseño de Casos de Prueba). A continuación se presenta un resumen caso de prueba para el caso de uso Gestionar Reporte.

Tabla 2- Caso de prueba CU-Gestionar Reporte.

	Descripción	id	category	description	name	token	jrxmlName	params	categoría	Respuesta del sistema	Flujo central
EC 1.1 Adicionar Reporte_ True	Este es el escenario ideal donde todos los datos de las variables son correctas y consecuentemente es exitosa la ejecución de la prueba.	N/A	V	V	V	V	V	N/A	V	El sistema muestra el mensaje el reporte ha sido creado correctamente	Acceder después de autenticarse a el módulo reportes: 1- Escoger la opción adicionar conexión. 2-Rellenar los campos correctamente. 3-Esperar la respuesta del servidor.
			SDR	Breve listado del comportamiento de la información	Reporte de Fuentes de Datos	97A377D0FEA14260B6B6FF5CE075C966	/media/07135ade-8fd6-4543-97c4-d5756dc9b763/CasosDePrueba_Liberación_SDRv1.0/CU_GestionarReporte/SC1_AdicionarReporte/Listado_Fuentes_Datos.jrxml		123		
EC 1.2 Adicionar Reporte_ False_RestricciónBD	Escenario que comprueba la correcta realización de la restricción de la base de datos que establece que la unión del identificador del usuario y el nombre del jrxml del reporte debe ser única, o sea, no pueden adicionarse dos reportes con el mismo sys_user_id y jrxml_name.	N/A	V	V	V	V	V	N/A	V	Error 109: JxrmlName duplicado viola la restricción de llave única.	Acceder después de autenticarse a el módulo reportes: 1- Escoger la opción adicionar conexión. 2-Rellenar los campos seleccionado un mismo Jrxml que ya está en la BD subida por el usuario. 3-Esperar la respuesta del servidor.
			SDR	Breve listado del comportamiento de la información de las fuentes de datos de los reportes almacenada en la tabla Conexiones de la base de datos del servidor.	Reporte de Fuentes de Datos	97A377D0FEA14260B6B6FF5CE075C966	/media/07135ade-8fd6-4543-97c4-d5756dc9b763/CasosDePrueba_Liberación_SDRv1.0/CU_GestionarReporte/SC1_AdicionarReporte/Listado_Fuentes_Datos.jrxml		123		

3.3.9. Resultados de las pruebas.

Se realizaron pruebas de aceptación de tipo *alpha* donde el cliente verificó que el sistema cumpliera con todos los requisitos definidos y elaboró una carta de aceptación que valida el correcto funcionamiento de la aplicación, dicha carta se puede consultar en los anexos de este documento. Como resultado de la aplicación de las pruebas de seguridad se verificó que cada usuario accede solo a las funcionalidades que le fueron definidas. Se realizaron las pruebas funcionales detectándose 16 no conformidades durante 3 iteraciones, las cuales fueron resueltas en la medida en que se detectaron. A continuación se muestra un gráfico que ilustra este resultado.

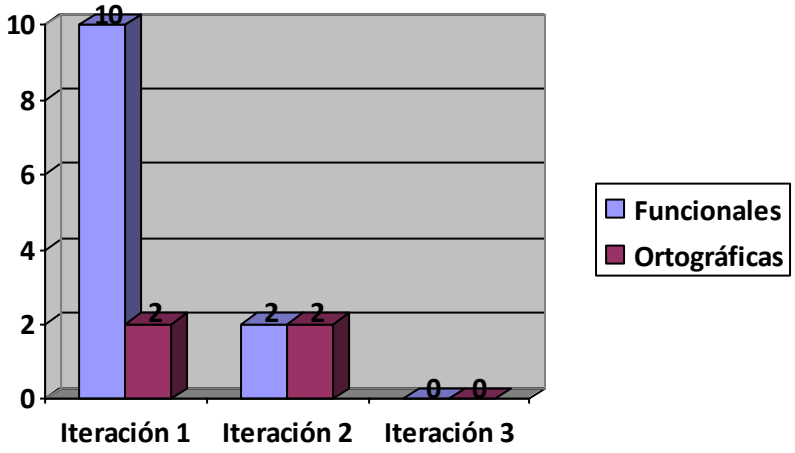

Fig 27- Resultados de las pruebas de caja negra.

3.4. Conclusiones del Capítulo.

En el presente capítulo se realizó el modelo de implementación del Sistema de Administración del SDR. Como parte de este se diseñaron los diagramas de componentes de los CU para mostrar una representación de los componentes físicos del sistema. Se definieron los estándares de codificación a utilizar para lograr un código fuente más organizado y entendible, además de facilitar su posterior mantenimiento. Se le realizaron pruebas al sistema a nivel de unidad y a nivel de sistema, para garantizar un producto sin errores y con un buen funcionamiento. Para esto se utilizó el método de caja blanca con la técnica de camino básico y de caja negra acompañado de la técnica de partición de equivalencia y las pruebas de aceptación y seguridad.

CONCLUSIONES

Luego de haber terminado la presente investigación se arriban a las siguientes conclusiones:

- El estudio de los principales servidores de reportes, permitió conocer cómo se realizan la administración de los recursos que ellos gestionan, permitiendo sentar las bases para el desarrollo del Sistema de Administración del SDR.
- A partir de las funcionalidades descritas se realizó el análisis y el modelo de diseño del Sistema de Administración del SDR proporcionando el punto de partida para las actividades de implementación.
- Como resultado se obtuvo el Sistema de Administración del SDR, un software capaz de visualizar todos los recursos almacenados en el Servidor Dinámico de Reportes, permitiéndole a los usuarios tener una herramienta mediante la cual puedan administrar sus recursos y a los administradores les provee de un mecanismo para conocer el estado del servidor y darle mantenimiento al mismo.
- El diseño y ejecución de las pruebas a nivel de unidad y de sistema permitió comprobar el correcto funcionamiento del Sistema de Administración del SDR.

RECOMENDACIONES

Luego de haber analizado los resultados de la investigación se recomienda:

- Posibilitarle a los usuarios de la aplicación ver los reportes del uso de sus recursos.

REFERENCIAS

AGUILA, K. M. O. Y. B. H. C. Y. Y. P. Servidor Dinámico de Reportes. *Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2014)* [Type of Work]. 2014, [cited 20 de abril del 2015.

ALEGSA. Definición de términos informáticos [online]. 2015. Available from World Wide Web:<www.alegsa.com.ar/Diccionario/diccionario.php>.

ALEGSA, L. Definición de Framework 2014, [cited 14 de Mayo del 2015. Available from Internet:<http://www.alegsa.com.ar/Dic/framework.php>.

COMMUNITY, J. iReport Designer. In., 2014.

COMPONENTESOURCE. ActiveReports-6. 2011, [cited 2 de Octubre de 2011. Available from Internet:<http://www.componentsource.com/products/activereports-6/summary-es.html/>.

CORTÉS, R. F. B. G. Y. Y. R. B. Y. C. R. H. Y. C. M. D. R. Y. M. C. GeReport: Sistema de Gestión de Reportes Dinámicos. 2014, vol. 8, [cited 25 de mayo del 2015. Available from Internet:<http://rcci.uci.cu/index.php?journal=rcci&page=article&op=view&path[]=726&path[]=296>. ISSN 2227-1899.

CRICK, J. *Crystal Reports Server XI Visión general de su funcionalidad*. edited by B. OBJECTS. Edtion ed., 2008.

CRISTALAB. Apache Server. 2008, [cited 24 de Octubre de 2011. Available from Internet:<http://www.cristalab.com/tutoriales/crear-tu-propio-servidor-web-4.-apache-php-y-mysql-c51133l/>.

CHEN, Y. L. E. H. X. *Architecture of Information System Combining SOA and BPM*. edited by I.M.A.I.E. INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT. Edtion ed., 2008.

DEVELOPERS, Q. *qooxdoo Documentation*. Edtion ed., 2014.

GRAPECITY *ActiveReports 8 Server Administrator Guide*. Edtion ed., 2011.

GROUP, O. M. Object Management Group - UML. 2012, [cited 23 de noviembre de 2012. Available from Internet:<http://www.uml.org/>.

LARMAN, C. UML y Patrones. In *Introduccion al analisis y diseno orientado a objetos.*, 2004.

LETICIA CALZADA, J. L. A. *El impacto de las herramientas de inteligencia de negocios en la toma de decisiones de los ejecutivos*. Edtion ed., 2009. ISBN 1870-557X.

MACIAS, G. MODELO 3 CAPAS (N CAPAS). 2013, [cited 20/02/2015. Available from Internet:<https://prezi.com/gl7pxorrhibn/modelo-3-capas-n-capas/>.

MÉNDEZ, I. G. Definicion de herramienta CASE. 2013, [cited 10 de febrero del 2015. Available from Internet:<http://ithuejutlaisabelgarciamendez.blogspot.com/2013/02/1_9013.html>.

ORACLE. NetBeans. 2012, [cited 10 de 6 de 2012. Available from Internet:<http://netbeans.org/index_es.html>.

PARADIGM, V. Sitio Oficial del Visual Paradigm. 2012, [cited 01 de Junio de 2012. Available from Internet:<http://www.visual-paradigm.com/product/vpuml/>.

PÉREZ, J. E. Introducción a JavaScript. In., 2009.

PGADMIN. Sitio oficial del PgAdmin. 2010, [cited 15 de Abril del 2015. Available from Internet:<http://www.pgadmin.org/>.

POSTGRESQL, S. PostgreSQL. 2012, [cited 20 de Noviembre de 2012. Available from Internet:<http://www.postgresql.org.es/>.

PRESSMAN, R. S. *Ingeniería del software. Un enfoque práctico. 6ta edición*. Edtion ed., 2011.

REFERENCIAS

SANCHEZ, M. N. SAD. 2011, [cited 01 de Junio de 2012. Available from Internet:<http://mistock.lcompras.biz/tallersoftware/1259-apache-una-alternativa-viable-para-el-servidor-web/>.

SÁNCHEZ, T. R. Metodología de desarrollo para la Actividad productiva de la UCI. [online]. 2014 [cited 19 de Marzo del 2015. Available from World Wide Web:<http://excriba.prod.uci.cu/page/context/shared/document-details?nodeRef=workspace://SpacesStore/a6002b5f-d3cb-4217-b90d-ddd55621c271>.

SAP. The best-Run businesses Run SAP. 2013, [cited 04 de 10 de 2013. Available from Internet:<http://www.sap.com/solution/sme/software/analytics/crystal-reports/index.html/>.

SOMMERVILLE, I. *Ingeniería del software. 7ma edición.* Edtion ed. Madrid, 2005.

BIBLIOGRAFÍA

ACHILLO, E. P. G. Viceministerio Ciencia y Tecnología [online]. 2012. Available from World Wide Web:<http://www.cienciaytecnologia.gob.bo/vcyt2012/uploads/cap-2-funciones-y-procedimientos.pdf/>.
AENOR.UNE-EN-ISO Gestión de la calidad y aseguramiento de calidad 2000.
AGUILA, K. M. O. Y. B. H. C. Y. Y. P. Servidor Dinámico de Reportes. *Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2014)* [Type of Work]. 2014, [cited 20 de abril del 2015.
ALEGSA Herramientas de Modelado 1998-2015.
ALEGSA, L. Definición de Framework 2014, [cited 14 de Mayo del 2015. Available from Internet:<http://www.alegsa.com.ar/Dic/framework.php>.
ÁLVAREZ, M. A. web2py v1 documentation [online]. 2010. Available from World Wide Web:<http://www.latinuxpress.com/books/drafts/web2py/caps/cap2.html>.
AXIOSSYSTEMS.
BARCHINI, G. E. Métodos "I + D" de la Informática. In *Revista de Informática Educativa y Medios* BEST, J. *Damned Lies and Statistics*. Edtion ed., 2012. 224 p. ISBN 9780520274709.
BOEHM, B. Verifying and validating software Requirements and desing Spaecifications 1984.
CMMI Capability Maturity Model Integration. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing 2002.
COMMUNITY, J. iReport Designer. In., 2014.
COMPONENTESOURCE. ActiveReports-6. 2011, [cited 2 de Octubre de 2011. Available from Internet:<http://www.componentsource.com/products/activereports-6/summary-es.html/>.
CORTÉS, R. F. B. G. Y. Y. R. B. Y. C. R. H. Y. C. M. D. R. Y. M. C. GeReport: Sistema de Gestión de Reportes Dinámicos. 2014, vol. 8, [cited 25 de mayo del 2015. Available from Internet:<http://rcci.uci.cu/index.php?journal=rcci&page=article&op=view&path[]=726&path[]=296>. ISSN 2227-1899.
CRICK, J. *Crystal Reports Server XI Visión general de su funcionalidad*. edited by B. OBJECTS. Edtion ed., 2008.
CRISTALAB. Apache Server. 2008, [cited 24 de Octubre de 2011. Available from Internet:<http://www.cristalab.com/tutoriales/crear-tu-propio-servidor-web-4.-apache-php-y-mysql-c51133l/>.
CZARNECKI Feacture models are views on ontologies s.I: 10th International, Software Product Line Conference 2006.
CHEN, Y. L. E. H. X. *Architecture of Information System Combining SOA and BPM*. edited by I.M.A.I.E. INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT. Edtion ed., 2008.
DEFINICIÓN.DE. Definición de reporte - Qué es, Significado y Concepto. 2011, [cited 7 de Noviembre de 2011. Available from Internet:<http://definicion.de/reporte/>.
DEVELOPERS, Q. *qooxdoo Documentation*. Edtion ed., 2014.
ECLIPSE. OpenUp. 2014, [cited 10 de 2 de 2014. Available from Internet:<http://epf.eclipse.org/wikis/openup/>.
ECURED. Herramienta CASE. 2013. Available from Internet:<http://www.ecured.cu/index.php/Herramienta_CASE/>.
GRAPECITY *ActiveReports 8 Server Administrator Guide*. Edtion ed., 2011.
GROUP, O. M. Object Management Group - UML. 2012, [cited 23 de noviembre de 2012. Available from Internet:<http://www.uml.org/>.
HERNÁNDEZ, P. V. *Uso de patrones de arquitectura*. Edtion ed., 2011.

BIBLIOGRAFÍA

IEEE (IEEE-610-12-1990) de ANSI 1990.

KIT, E. Software Testing In the Real World: Improving Thhe Peocess. Addison Wesley ISBN 0201877562 1995.

LARMAN, C. UML y Patrones. In *Introduccion al analisis y diseno orientado a objetos.*, 2004.

LETICIA CALZADA, J. L. A. *El impacto de las herramientas de inteligencia de negocios en la toma de decisiones de los ejecutivos*. Edtion ed., 2009. ISBN 1870-557X.

MACIAS, G. MODELO 3 CAPAS (N CAPAS). 2013, [cited 20/02/2015. Available from Internet:<https://prezi.com/gl7pxorrhibn/modelo-3-capas-n-capas/>.

MÉNDEZ, I. G. Definicion de herramienta CASE. 2013, [cited 10 de febrero del 2015. Available from Internet:<http://ithuejutlaisabelgarciamendez.blogspot.com/2013/02/1_9013.html>.

MICROSOFT. Microsoft SQL Server 2005 Reporting Services. 2011, [cited 2 de Octubre de 2011. Available from Internet:<http://justindeveloper.wordpress.com/2008/10/20/microsoft-sql-server-2005-reporting-services-part-i/>.

MICROSOFT. Microsoft Application Architecture Guide, 2nd Edition. 2012. Available from Internet:<https://www.microsoft.com/en-us/download/details.aspx?id=16236>.

MUÑOZ, D. L. *Manual de Estadística*. Edtion ed., 2010. ISBN ISBN-13: 84-688-6153-7.

NAN-TIC. NAN -TIC Creamos y adaptamos software libre. 2013, [cited 10 de 10 de 2013. Available from Internet:<http://www.nan-tic.com/es/rd/jasper-reports/>.

ORACLE. NetBeans. 2012, [cited 10 de 6 de 2012. Available from Internet:<http://netbeans.org/index_es.html>.

PARADIGM, V. Sitio Oficial del Visual Paradigm. 2012, [cited 01 de Junio de 2012. Available from Internet:<http://www.visual-paradigm.com/product/vpuml/>.

PÉREZ, J. A. H. Módulo de Administración en un Sistema Informático. 2008.

PÉREZ, J. E. Introducción a JavaScript. In., 2009.

PGADMIN. Sitio oficial del PgAdmin. 2010, [cited 15 de Abril del 2015. Available from Internet:<http://www.pgadmin.org/>.

POSTGRESQL, S. PostgreSQL. 2012, [cited 20 de Noviembre de 2012. Available from Internet:<http://www.postgresql.org.es/>.

PRESSMAN, R. S. *Ingeniería del software. Un enfoque práctico. 6ta edición*. Edtion ed., 2011.

ROJAS, M. A. Introduccion y Principios Básicos del Desarrollo de Software Basado en Componentes. 2004.

SANCHEZ, M. N. SAD. 2011, [cited 01 de Junio de 2012. Available from Internet:<http://mistock.lcompras.biz/tallersoftware/1259-apache-una-alternativa-viable-para-el-servidor-web/>.

SÁNCHEZ, T. R. Metodología de desarrollo para la Actividad productiva de la UCI. [online]. 2014 [cited 19 de Marzo del 2015. Available from World Wide Web:<http://excriba.prod.uci.cu/page/context/shared/document-details?nodeRef=workspace://SpacesStore/a6002b5f-d3cb-4217-b90d-ddd55621c271>.

SAP. The best-Run businesses Run SAP. 2013, [cited 04 de 10 de 2013. Available from Internet:<http://www.sap.com/solution/sme/software/analytics/crystal-reports/index.html/>.

SCALONE, F. Estudio comparativo de los modelos y estándares de calidad de software 2006.

SENCHA. Licencias de Sencha. 2015, [cited 10 de marzo del 2015. Available from Internet:<http://www.sencha.com/legal/>.

SERVICES, P. D. R. Presentación de Reporting Services. 2011. Available from Internet:<http://msdn.microsoft.com/es-es/library/ms155786%28v=sql.90%29.aspx/>.

SOMMERVILLE, I. *Ingeniería del software. 7ma edición.* Edtion ed. Madrid, 2005.

GLOSARIO DE TÉRMINOS

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. (Alegsa 2015)

CSS Las hojas de estilo en cascada (*Cascading Style Sheets*, o sus siglas CSS) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML, pero también puede ser aplicado a cualquier tipo de documentos XML, incluyendo SVG y XUL. (Alegsa 2015)

CURL es una herramienta para usar en un intérprete de comandos para transferir archivos con sintaxis URL, soporta FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP. CURL soporta certificados HTTPS, HTTP POST, HTTP PUT, subidas FTP, subidas mediante formulario HTTP, proxies, cookies, autenticación mediante usuario+contraseña, continuación de transferencia de archivos*, tunneling* de proxy http y muchas otras prestaciones. CURL es open source/software libre distribuido bajo la Licencia MIT. (Alegsa 2015)

CVS *Concurrent Versions System* o simplemente CVS, también conocido como *Concurrent Versioning System*, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL. (Alegsa 2015)

DOC, es el formato que visualiza o procesa todos los textos Microsoft Windows, el cual es Microsft Office Word. (Alegsa 2015)

DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto

estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente. (Alegsa 2015)

GrapeCity inc: Es una empresa privada, corporación multinacional de software con sede en Sendai, Japón, que desarrolla sus propios productos de software y ofrece servicios externalizados de desarrollo de productos, servicios de consultoría, software y servicios de gestión de relaciones con los clientes. GrapeCity también estableció *WINEstudios*, un diseño de medios de comunicación y las instalaciones de producción digital en Japón. (GrapeCity 2011)

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de *NHibernate*) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. (Alegsa 2015)

HTML, siglas de *HyperText Markup Language* («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que, en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. (Alegsa 2015)

JavaBeans Los JavaBeans son un modelo de componentes creado por *Sun Microsystems* para la construcción de aplicaciones en Java. Se usan para encapsular varios objetos en un único objeto (la vaina o *Bean* en inglés), para hacer uso de un solo objeto en lugar de varios más simples. La especificación de JavaBeans de *Sun Microsystems* los define como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción. (Alegsa 2015)

JDBC *Java Database Connectivity*, más conocida por sus siglas JDBC, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice. (Alegsa 2015)

GLOSARIO DE TÉRMINOS

JPEG. Formato de archivos de fotografías más conocido y utilizado, principalmente por la versatilidad y posibilidades al permitir hacer los archivos hasta 10 veces más pequeños que formatos como RAW o TIFF. (Alegsa 2015)

ODT El Formato de Documento Abierto para Aplicaciones Ofimáticas de OASIS (en inglés, OASIS Open *Document Format for Office Applications*), también referido como formato *OpenDocument* (ODF), es un formato de archivo abierto y estándar para el almacenamiento de documentos ofimáticos tales como hojas de cálculo, textos, gráficas y presentaciones. (Alegsa 2015)

PDF (sigla del inglés *portable document format*, formato de documento portátil) es un formato de almacenamiento de documentos digitales independiente de plataformas de software o hardware. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto). Fue inicialmente desarrollado por la empresa *Adobe Systems*, oficialmente lanzado como un estándar abierto el 1 de julio de 2008 y publicado por la Organización Internacional de Estandarización como ISO 32000-1. (Alegsa 2015)

Licencia Dual: Se puede utilizar con fines académico, pero no se puede utilizar para la comercialización. (Alegsa 2015)

REST, (acrónimo *REpresentational State Transfer*) es una arquitectura de servicios Web, para sistemas distribuidos como la *World Wide Web*. El término fue acuñado por Roy Fielding en el 2000. (Alegsa 2015)

RTF(formato de texto enriquecido a menudo abreviado como RTF) es un formato de archivo informático desarrollado por Microsoft en 1987 para el intercambio de documentos multiplataforma. La mayoría de procesadores de texto son capaces de leer y escribir documentos RTF. (Alegsa 2015)

SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones. (Alegsa 2015)

TIFF (*Tagged-Image File Format*) Es un formato flexible de imágenes de mapa de bit. (Alegsa 2015)

TXT Los ficheros de texto son la forma que se utiliza en programación (informática) parar salvar los datos (variables y valores) procesados por los programas y que éstos, los datos, no tengan que ser introducidos constantemente para su posterior re-procesamiento por el programa. (Alegsa 2015)

Widgets Los *widgets* de escritorio también se conocen como *gadgets* de escritorio, y son una nueva categoría de mini aplicaciones; diseñadas para proveer de información o mejorar una aplicación o

servicios de un ordenador o computadora, o bien cualquier tipo de interacción a través del *World Wide Web*, por ejemplo una extensión de alguna aplicación de negocios, que nos provea información en tiempo real del estatus del negocio u organización. (Alegsa 2015)

XML, siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. (Alegsa 2015)

XLS Microsoft Excel es una aplicación distribuida por Microsoft Office para hojas de cálculo. Este programa es desarrollado y distribuido por Microsoft, y es utilizado normalmente en tareas financieras y contables. (Alegsa 2015)