Universidad de las Ciencias Informáticas

FACULTAD 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes

Autores:

José Alejandro Bidot Hernández

Fabiel León Oliva

Tutor(es):

MSc. Yudelkis Abad Fuentes

Ing. Félix González Martínez

La Habana, junio de 2015

"Año 57 de la Revolución"

"Aprende como si fueras a vivir toda la vida
y vive como si fueras a morir mañana."

Charlie Chaplin

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la tesis **"Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes"** y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

José Alejandro Bidot Hernández **Fabiel León Oliva**

_____ _____

Firma del Autor Firma del Autor

MSc. Yudelkis Abad Fuentes **Ing. Félix González Martínez**

_____ _____

Firma del Tutor Firma del Tutor

DATOS DE CONTACTO

Autores:

José Alejandro Bidot Hernández.

E-mail: jabidot@estudiantes.uci.cu

Fabiel León Oliva.

E-mail: fleon@estudiantes.uci.cu

Tutor: MSc. Yudelkis Abad Fuentes.

E-mail: yabad@uci.cu

- ✓ Ing. En Ciencias Informáticas, Universidad de Ciencias Informáticas, 2007.
- ✓ Máster en Bioinformática, Universidad de Ciencias Informáticas, 2010.
- ✓ Profesora Asistente de la Universidad de las Ciencias Informáticas.
- ✓ 7 años de experiencia en la educación superior impartiendo las asignaturas Sistemas de Bases de Datos I y Sistemas de Bases de Datos II.
- ✓ 5 publicaciones en memorias de eventos como primer autor, 4 en revistas arbitradas y má s de 12 participaciones en eventos científicos.
- ✓ Ha impartido cursos de postgrado como “Tecnologías y estrategias didácticas en la enseñ an-za de Bases de Datos” y “Actualidad en PostgreSQL”.

Tutor: Ing. Félix González Martínez.

E-mail: felix@uci.cu

- ✓ Ing. En Ciencias Informáticas, Universidad de Ciencias Informáticas, 2013.
- ✓ Vinculado al Centro de Tecnologías de Gestión de Datos (DATEC).

AGRADECIMIENTOS

De Bidot:

En el transcurso de estos 5 años he vivido experiencias increíbles e inolvidables que me han hecho superarme día tras día. Cada año fue un nuevo reto para mí, que puede vencer en gran parte por el apoyo que me brindaron personas a las que hoy quiero agradecer.

En primer lugar y compartido a mi papá por ser el ejemplo a seguir en todo momento, tu apoyo y tu preocupación hicieron que me esforzara al máximo en estos 5 años; y a mi mamá por sus buenos consejos, que aunque a veces parecían demasiados, supieron hacer de su príncipe enano un profesional.

A mis abuelos que aunque la vida me los quitó muy temprano, me guían desde el cielo en cada paso que doy.

A mis tíos Gladys, Iraida, Orlando, mi primo Orlandito, mi viejita linda Clara y a mi familia de Varadero que siempre me han querido en todo momento.

A grandes amigos que conocí en la universidad y no puedo dejar de mencionar hoy (esto es sin importar el orden): Andy al que le debo gran parte de este título, a mi gatika Yiselita, a mis hermanos integrales el Yera y Eriberto el loco, a mi tata Tahimi, a mi Bbcita, a mi sister Anita, a Sarita mi negitra millonaria, a Claudiña, a las Nurys, a Frank el Metra, mi compay El pocho, Eduardito el Croque, Jorgito Carachimba, mi compañero de vicio el Fuma, a mis amigos del antiguo y actual grupo 4, a todos los que están hoy aquí, y por su puesto a mi compañero de tesis Fabio el phanton.

A mis hermanitos de la vida en todo momento que no por últimos son menos importantes: Odecita, El filo, Rey, Julito, Daniel y Chicho.

A profesores que me han apoyado, influyendo en mi formación como profesional y que hoy considero amigos: A mis profes Silvia y Alcires por mi preparación pre-universitaria, al profe Tellez del cual recibí un apoyo incondicional, a mi teacher "el Jingilin jeem" por estar disponible cada vez que lo necesito, a mi padrino "Castaño "con el cual conté en todo momento, a la profe Beatriz, al profe Omar, a la profe Yuya, a Oscar, Garnache, Yuned, Julie. A los miembros del tribunal: Adisleys, Pacheco, Luna y Esley, gracias a sus exigencias y críticas constructivas obtuvimos un documento con más calidad, y en especial a mis tutores Félix y Yudelkis gracias por su paciencia y por guiarnos en el desarrollo de la investigación.

AGRADECIMIENTOS

De Fabiel

Quiero agradecer a mi familia en especial a mi padre que no se encuentra en este mundo pero que siempre va estar conmigo, gracias por ser mi ejemplo. A mi madre que me ha guiado en la vida para lograr ser una persona de bien, son pocas las palabras para agradecerte todo lo que has hecho por mi. A mi tía Mayra que sin ella tampoco hubiera sido posible llegar hasta donde he llegado, un millón de besos. A mi abuela gracias por sus enseñanzas que siempre las escucho aunque parezca que no. A mi hermano Frank y mi primo Ronald que son parte importante de mi familia. A mi familia en general por mantener la alegría siempre a pesar de los malos momentos. A todos mis tíos, tías y primos(as).

A todas las amistades que he conocido en la UCI desde los primeros años hasta los días de hoy que sigo haciendo amigos. A toda la gente que vivieron conmigo en el primer apto que residí Thondigue, Koko que son los que quedan en la UCI de aquellos tiempos. A las niñas del aula Juliet , Marianela y Arianna que siguen también aquí. A la gente de segundo año no puedo dejar de agradecer a mi hermanito Carlinhos y al Juanca.

A la gente de mi aula Aylin, Tahimi, Laura, Jisel, el Bernard , Emilio, Alle…

Quiero agradecer a todos los profesores no solo aquellos que me han dado clases sino a todos los que hacen un gran trabajo ejerciendo su labor. A los que han ayudado para lograr este trabajo de diploma a mis tutores Felix y Yudelkis. A la profe Beatriz, y a Keimer gracias por ayudarnos.

A mi compañero de tesis el próximo productor de los Rastamenba, Bidot.

Gracias.

DEDICATORIA

De Bidot:

A mi padre José Alejandro Bidot Peláez por no poder estar con él en el "Día de los Padres" porque tenía que puntualizar cosas de la tesis…

A mi madre Dora Iracema Hernández García por justificarme con él…

De Fabiel:

Dedico este trabajo a mi familia, en especial a mis padres Mirtha Oliva Márquez y Juan Francisco León Guerra.

RESUMEN

El Centro de Tecnologías de Gestión de Datos DATEC (del inglés *Data Technologies Center*) de la Universidad de las Ciencias Informáticas (UCI), cuenta actualmente con el proyecto Servidor Dinámico de Reportes (SDR), el cual se encarga de la exportación y publicación de reportes. En su versión actual el SDR requiere de diseñadores de reportes que exploten sus funcionalidades dependiendo de conocimientos técnicos, para poder administrar los recursos, dado que no posee una interfaz gráfica de usuario que facilite este proceso. El objetivo del presente trabajo de diploma consiste en desarrollar una extensión al iReport que garantice la usabilidad en la gestión de reportes del SDR. Para guiar el proceso de desarrollo de la extensión se utilizó la metodología OpenUP, NetBeans IDE como entorno de desarrollo, el lenguaje Java y Visual Paradigm for UML para el modelado de los artefactos. Se realizaron pruebas unitarias, de integración, validación y de usabilidad para comprobar el correcto funcionamiento y calidad de la extensión. Como resultado se obtuvo una extensión de iReport que permite la integración entre este último y el SDR, lo cual posibilitó la comunicación entre estas dos soluciones permitiendo diseñar reportes en el iReport y publicarlos en el SDR mediante una interfaz gráfica de usuario.

PALABRAS CLAVE

Extensión al iReport, reporte, SDR, usabilidad.

ABSTRACT

The Data Technologies Center (DATEC) from the University of Informatics Sciences (UCI), it has the project Reports Dynamic Server (SDR), which is responsible for exporting and publishing reports. In their current version the SDR requires of designers of reports that they exploit its functionalities depending on technical knowledge, to be able to administer the resources, since it doesn't possess user's graphic interface that facilitates this process. The objective of the present diploma work consists on developing an extension to the iReport that guarantees the use in the administration of reports of the SDR. To guide the extension development process it was used the OpenUP methodology, NetBeans IDE as a development environment, the language Java and Visual Paradigm for UML for modeling artifacts. They were carried out unitary tests, of integration, validation and use to check the correct operation and quality of the extension. As a result an iReport extension was obtained that allows the integration between this last one and the SDR, that which facilitated the communication among these two solutions allowing to design reports in the iReport and to publish them in the SDR by means of user's graphic interface.

KEY WORDS

Extension to the iReport, report, SDR, usability.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad las entidades u organizaciones manejan grandes volúmenes de datos como resultado de los procesos de negocios llevados a cabo en el entorno en que se desenvuelven. Estos requieren de aplicaciones y tecnologías que permitan organizar y extraer la información de bases de datos (BD), siendo estas de utilidad para la planeación estratégica, el control y la toma de decisiones de especialistas y directivos. Este proceso de obtención de la información correcta en la forma adecuada para tomar la acción precisa, es lo que se define en el ámbito empresarial como gestión de la información.

Bill Gates en 1999 definió en su libro “Los negocios en la era digital”: la gestión de la información con la siguiente expresión: *“(…) Lograr que la información correcta y pertinente llegue a la persona u organización que la necesita en el momento preciso (…)”(GATES 1999)*. A nivel internacional se han desarrollado numerosos sistemas que facilitan la gestión de la información entre los que destacan los generadores de reportes, los cuales muestran de manera organizada y resumida datos relevantes para un fin determinado.

Un reporte consta de dos fases fundamentales: creación y publicación. En la primera fase el reporte puede ser concebido de dos maneras, haciendo uso de una herramienta llamada diseñador de reportes o mediante sentencias de código de forma manual. La segunda fase, o sea la publicación, ocurre en el servidor encargado de publicar estos reportes.

Los diseñadores de reportes son herramientas complementarias de los sistemas de información que utilizan un lenguaje transparente para el usuario, capaz de realizar consultas a la BD para obtener información, transformarla y organizarla en forma de reportes. Estas herramientas facilitan el modelado de los datos a través de una interfaz gráfica de usuario que permite la creación de hojas de datos, gráficas, reportes y tablas dinámicas. Cuentan además con opciones para la gestión de los usuarios del sistema, los grupos, conexiones de BD, para creación de proyectos, así como para la distribución de los reportes mediante archivos físicos a los cuales el usuario final puede acceder a través de un visor de reportes o una ruta web, utilizando un navegador u otra herramienta.

En el mundo existe una amplia variedad de diseñadores de reportes que por lo general publican sobre un servidor específico. Entre estos se destacan Pentaho Reporting y Crystal Reports quienes ofrecen un

acabado del trabajo para el cliente permitiendo visualizar y obtener en diferentes formatos y variables dinámicas al reporte en el tiempo deseado. Estas características hacen que los diseñadores de reportes se hayan generalizado a la mayoría de las empresas que manejan información digital.

Las empresas en Cuba no se quedan exentas de este tipo de herramientas, por ejemplo el Generador de Reportes del Sistema para la integración de la información de Recursos Humanos (GREHUCORP) para la gestión de recursos humanos, GeReport para el diseño, generación y configuración de los reportes relacionados con los datos históricos almacenados en una fuente de datos y el Generador Dinámico de Reportes (GDR) desarrollado en la Universidad de las Ciencias Informáticas (UCI).

La UCI es una universidad dedicada a la producción de soluciones informáticas, que orienta su desarrollo de software en productos y servicios para evitar la adquisición de herramientas o recursos informáticos privativos con altos costos en el mercado internacional. Es soportada por una red de centros productivos entre los que resaltan el Centro de Informática Médica (CESIM), el Centro de Tecnologías para la formación (FORTES) y el Centro de Tecnologías de Gestión de Datos (DATEC).

DATEC posee varios productos de software entre los que destacan las soluciones para almacenes de datos, captura y procesamiento de la información, siendo esta mostrada a través de reportes. Los reportes son diseñados por una herramienta propia del centro llamada GDR que le da soporte al ciclo de vida. Actualmente la versión 2.0 de dicho software ha mejorado sus prestaciones en cuanto al diseño visual y la publicación de reportes, y emplea el Servidor Dinámico de Reportes (SDR) como parte de la solución.

A pesar que el SDR surge de manera complementaria para el GDR en su versión 2.0, puede coexistir separado de un diseñador, lo que convierte a esta herramienta en una solución más independiente debido a que se puede integrar con cualquier diseñador que sea compatible con el servidor. El SDR cuenta con una interfaz de servicios que permite la interoperabilidad para la publicación de reportes y la gestión de los recursos como tareas programadas, reportes exportados, entre otros. Se basa en las librerías de código abierto de JasperReport. Estas librerías son utilizadas por el sistema de generación de reportes JasperSoft de carácter privativo y emplea como diseñador nativo la herramienta informática iReport siendo en la actualidad un potente generador de reportes.

iReport constituye una alternativa a GDR debido a que este último está enfocado a un ambiente web posibilitando mayor accesibilidad. Es una solución desarrollada para un entorno de escritorio, que permite realizar diseños de plantillas y reportes sin necesidad de conexión. Esto le facilita a la universidad y al país contar con ambas soluciones que evitan la adquisición de productos con funcionalidades similares, algunos con altos costos de licencias en el mercado internacional.

Para el SDR como servidor de reportes sería de mucha utilidad contar con diseñadores de reportes que exploten las funcionalidades implementadas. Además requiere de conocimientos técnicos para poder administrar los recursos del servidor dado que no posee una interfaz gráfica de usuario que facilite este proceso. Esto conlleva a que no se logren gestionar los reportes de manera fácil para los usuarios que no están especializados en el uso de herramientas y tecnologías relacionadas con los servicios web.

Después de analizar la problemática anteriormente descrita, surge el siguiente **problema de investigación:** ¿Cómo garantizar la usabilidad en la gestión de reportes para los usuarios del Servidor Dinámico de Reportes?

Se define como **objeto de estudio** de la investigación: La gestión de reportes, enmarcado en el **campo de acción:** Gestión de reportes en el Servidor Dinámico de Reportes.

Para darle solución a la problemática identificada se define como **objetivo general:** Desarrollar una extensión al iReport que permita garantizar la usabilidad en la gestión de reportes del Servidor Dinámico de Reportes mediante la integración de ambas soluciones.

Las siguientes **preguntas científicas** guiarán el desarrollo de la investigación:

1. ¿Cuáles son los referentes teóricos relacionados con la implementación de la extensión al iReport para la gestión de reportes en el SDR?
2. ¿Cuáles son las características y cualidades que debe tener la extensión al iReport para la gestión de reportes en el SDR?
3. ¿Cómo desarrollar una extensión que permita integrar el iReport con el SDR para la gestión de reportes en el SDR?
4. ¿Cómo integrar la extensión al iReport para la gestión de reportes en el SDR?
5. ¿Cómo verificar el correcto funcionamiento de la extensión al iReport para la gestión de reportes en el SDR?

6. ¿Cómo guiar al usuario durante la instalación y uso de la extensión?

Para dar cumplimiento a las preguntas planteadas se proponen las siguientes **tareas de la investigación**:

1. Caracterización de los diseñadores de reportes para identificar los elementos principales de la publicación de reportes.
2. Identificación de los principios arquitectónicos, las Interfaces de Programación de Aplicaciones (API, del inglés *Application Programming Interface*) y componentes para desarrollar extensiones al iReport.
3. Definición de los requisitos funcionales de la extensión para la identificación de las funcionalidades que debe realizar la extensión.
4. Elaboración del modelo de casos de uso del sistema para especificar la comunicación y el comportamiento de la extensión.
5. Elaboración del diagrama de clases del diseño para especificar la relación que existirá entre las clases identificadas en la extensión.
6. Elaboración del modelo de implementación para representar los componentes de la extensión y sus dependencias.
7. Realización de pruebas a la solución implementada para comprobar el funcionamiento de la extensión y la usabilidad de la misma.
8. Confección del Manual del Usuario para que este conozca los procesos que puede realizar con la herramienta implementada.
9. Confección del Manual de Instalación para que el usuario conozca cómo integrar la extensión al Servidor Dinámico de Reportes.

Métodos de investigación científica

Para llevar a cabo la extensión del iReport se emplearán los métodos de la investigación científica permitiendo una mejor recopilación de la información. Según el Dr. Cs. Carlos Alvarez de Zayas, *"(…) los métodos de la investigación científica son agrupados en métodos teóricos y empíricos, los cuales a lo largo de toda investigación científica están dialécticamente relacionados por lo que uno ni se desarrolla ni existe sin el otro"* (ALVAREZ 1995). De los métodos teóricos fueron seleccionados la modelación y el

analítico-sintético y de los métodos empíricos se emplea la técnica de recopilación de información tormenta de ideas.

La **modelación** se utiliza a través de los modelos de análisis, diseño e implementación para representar abstracciones con vistas a explicar la realidad y para modelar la solución de la extensión. El **analítico-sintético** se emplea en el análisis de los conceptos teóricos que existen y en la extracción de características y elementos a tener en cuenta para el desarrollo de la aplicación.

Se emplea como técnica de recopilación de información la **tormenta de ideas** con el objetivo de identificar las características y capacidades con las que debe cumplir la extensión al iReport para la gestión de reportes en el SDR, teniendo en cuenta los criterios del jefe de proyecto, los tutores de la investigación, desarrolladores del SDR y los autores.

El contenido de esta investigación estará estructurado en tres capítulos. A continuación se realiza una breve descripción de los elementos principales contenidos en cada uno de los capítulos.

- ◊ En el **capítulo 1** titulado “Fundamentos teóricos de la extensión al iReport para la publicación de reportes en el Servidor Dinámico de Reportes” se definirán los fundamentos teóricos y conceptos para entender el problema, así como, los principales pasos a seguir durante el desarrollo de la investigación. También se describe la metodología, el lenguaje, y las herramientas que existen, posibilitando una adecuada selección de los elementos a utilizar para el desarrollo del trabajo de diploma.
- ◊ En el **capítulo 2** titulado “Análisis y Diseño de la extensión al iReport para la publicación de reportes en el Servidor Dinámico de Reportes” se realizará el análisis y diseño de la extensión al iReport para la gestión de reportes. Para esto se especificarán los requisitos funcionales (RF) y los requisitos no funcionales (RNF) de la extensión, además de los patrones de diseño empleados en las clases de la extensión.
- ◊ En el **capítulo 3** titulado “Implementación y Prueba de la extensión al iReport para la publicación de reportes en el Servidor Dinámico de Reportes” se implementarán los RF definidos en la fase de requerimientos. Se realizará el Diagrama de componentes, teniendo en cuenta la arquitectura definida. Posteriormente serán realizadas las pruebas para comprobar el funcionamiento de la extensión demostrando así el cumplimiento de las metas propuesta.

Capítulo 1: Fundamentos teóricos de la extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes

En el presente capítulo se muestran los conceptos relacionados con el problema de la investigación. Se realiza un estudio bibliográfico de algunas aplicaciones que existen para generar reportes, además se describen las herramientas, metodología y tecnologías que servirán de apoyo a la investigación para dar solución al problema planteado.

1.1 Reporte

Un reporte es un informe o una noticia que puede ser impreso, digital o audiovisual, pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y específicamente en el ámbito de 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. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados. Según la herramienta informática y la base de datos en cuestión, los reportes permiten la creación de etiquetas y la elaboración de facturas.

Los reportes son de gran importancia pues permiten al usuario realizar de forma transparente consultas a las bases de datos y obtener información de estas en un determinado formato, con más rapidez y un mayor nivel de detalle y flexibilidad.

1.2 Gestión de reportes

Los sistemas de generación de reportes son herramientas de apoyo a los sistemas de información, teniéndose en cuenta que los indicadores que generan proyectan resultados que ayudan a la toma de decisiones en el ámbito empresarial. Estos sistemas generalmente están compuestos por dos elementos básicos: un diseñador de reportes y un motor de reportes, pero en ocasiones también poseen un servidor de publicación como herramienta para facilitar la gestión de los reportes.

1.2.1 Diseñadores de reportes

Los diseñadores o generadores de reportes son herramientas complementarias de los sistemas de información, incluidos en la mayoría de los productos de software empresariales, cuya función principal es el diseño de reportes. Estos permiten crear desde simples listados, hasta complejos reportes de datos ofreciéndole al cliente la información deseada. Proveen una forma transparente al usuario de realizar consultas a las bases de datos y obtener información de esta en forma de reporte. Tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones (SILVA 2003). Existen numerosas herramientas a nivel global para la generación de reportes como iReport, Pentaho Reporting, Crystal Reports Designer y GDR.

1.2.2 Motores de reportes

La calidad y rapidez en la generación de los reportes depende en gran medida del motor utilizado para este fin. Los motores de reportes son principalmente los encargados de obtener el diseño del reporte para realizar el proceso de compilado, llenado de los datos a partir de la información almacenada en la fuente de datos y luego exportarlo en un formato determinado. En el mundo existe una amplia gama de motores de reporte reconocidos tanto en el ámbito nacional como internacional, algunos de código abierto y otros privativos (HERNÁNDEZ 2014).

1.2.3 Servidores de reportes

Un servidor de reportes tiene la misión de proveer a los usuarios una manera transparente y rápida de exportar los reportes previamente diseñados y parametrizados, además de gestionar la administración de los recursos y garantizar la seguridad de los procesos. Los principales servidores de reportes utilizados en la actualidad son Crystal Reports y JasperReports Server, ambos licenciados bajo software privativo, pues aunque JasperReports Server se puede utilizar con fines académicos se prohíbe su uso con fines comerciales sin haber adquirido la licencia que permite este fin. Motivo principal que dio surgimiento al Servidor Dinámico de Reportes v1.0 (SDRv1.0) (HERNÁNDEZ 2014).

1.3 Herramientas que existen para la generación de reportes

1.3.1 Pentaho Reporting

"*Pentaho Reporting es un conjunto de herramientas de reportes de código abierto que permite la creación de informes relacionales y análisis de una amplia gama de fuentes de datos*" (TALES 2014). El propio Pentaho se define como una plataforma de BI (del inglés *Business Intelligence*) "orientada a la solución" y "centrada en procesos", fácil de utilizar y flexible para que los diseñadores creen de forma ágil los reportes de acuerdo a los estándares y necesidades del cliente. Permite la distribución de los resultados de los análisis en varios formatos como pdf, html, xls y en texto plano. Existen tres productos con diferentes enfoques: Pentaho Report Designer, Pentaho Report Design Wizard y Web ad-hoc reporting (Dataprix 2014).

1.3.2 Crystal Reports

"*Crystal Reports es una aplicación de inteligencia empresarial utilizada para diseñar y generar reportes desde una base de datos*" (TUERP 2014). Es un software propietario que integra estrechamente capacidades de diseño, modificación y visualización en aplicaciones .Net y Java. Se distribuye bajo los términos de la EULA, licencia por la cual el uso de un producto sólo está permitido para un único usuario, el comprador. Permite exportar los informes a diferentes formatos (Crystal Reports (rpt), xls, html, doc, pdf). Sus informes pueden ser almacenados en plataformas Windows y permite publicar informes Crystal en un servidor web. Facilita la creación de reportes simples y dispone también de poderosas herramientas necesarias para generar reportes complejos o especializados.

1.3.3 Generador Dinámico de Reportes (GDR)

El Generador Dinámico de Reportes es un sistema especializado en la generación de reportes, permitiendole al usuario el diseño de reportes personalizados. Es una herramienta basada en tecnología Web que permite conectarse a diversas fuentes de datos. Mediante esta herramienta los reportes pueden ser exportados a formatos de salida tales como: html, pdf, xls y csv. Tiene una interfaz gráfica amigable y está orientado al usuario final. Se encuentra en constante perfeccionamiento con el objetivo de lograr satisfacer las expectativas del mercado. (ORIHUELA 2013).

1.3.4 iReport

iReport es el diseñador de reportes nativo de JasperServer, con licencia GPL (Licencia Pública General) que se distribuye sin costo y por tiempo ilimitado. Permite a los usuarios editar visualmente cualquier tipo de informe complejo con gráficas, imágenes y subinformes. Con esta herramienta visual de diseño es posible crear nuevos reportes fácilmente usando asistentes de configuración y plantillas, con la opción de poder tener una pre-visualización de la exportación de los reportes en PDF, HTML, XLS, CSV, TXT y XML. Es un diseñador visual de informes poderoso, intuitivo y fácil de usar. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes y subinformes entre otros.

Algunas de las características a favor de iReport son: estar 100% escrito en Java, manejar el 98% de las etiquetas de JasperReports, facilidad de instalación, tener asistentes para las planillas y para generar los subreportes además de permitir diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de las cajas de texto, cartas y subreportes.

1.3.5 Jasper Server

Jaspers Server es un software propietario bajo licencia Dual, lo que implica que es libre con fines educativos y privativo si se trata de comercializar con él. Está compuesto por un conjunto de librerías desarrolladas en lenguaje Java para facilitar la generación de reportes en aplicaciones web y de escritorio. Esta escrito completamente en Java y es capaz de utilizar los datos procedentes de cualquier fuente de datos soportada. Presenta los documentos con precisión de pixel, dichos documentos se pueden ver, imprimir o exportar en una gran variedad de formatos. Permite la generación de subreportes e imprimir o exportar en una variedad de formatos de documentos incluyendo html, pdf, xls y doc (MONTES 2014).

1.3.6 Servidor Dinámico de Reportes

El SDR es un software que permite la gestión, compilación, publicación y exportación de reportes creados en herramientas de diseño de reportes. Está desarrollado completamente en Java utilizando las librerías de código abierto jasper-reports 5.2. Basado en una arquitectura orientada a servicios por lo que utiliza servicios de Transferencia de Estado Representacional (REST, del inglés *Representational State Transfer*). Posibilita que aplicaciones con interfaces basadas en “json” o “xml” puedan conectarse,

publicar y consumir recursos, lo que permite que el servicio sea utilizado por distintos clientes escritos en diferentes lenguajes, ejecutándose en diversas plataformas y dispositivos. Además permite que un usuario independiente, interactúe de forma directa con el servidor a través de algún protocolo de comunicación como el CURL (del inglés *Command Uniform Resource Locator*), y pueda consumir los servicios, siempre y cuando tenga los debidos permisos (MONTES 2014).

A continuación se muestra la Tabla 1 donde se expone una comparación entre algunas herramientas de reportes usadas en la actualidad. Para la confección de esta tabla se tuvo en cuenta una serie de propiedades que se consideró, influyen notablemente en la adquisición de una de estas herramientas.

Tabla 1: Comparación entre algunas herramientas para el trabajo con reportes.

Propiedades	Crystal Reports	Jasper Server	SDR	iReport	GDR	Pentaho
Sistema Operativo Multiplataforma	No (Windows)	Si	Si	Si	Si	Si
Fuente de datos	Múltiples	Múltiples	Múltiples	Múltiples	Múltiples	Múltiples
Lenguaje de Programación	.NET	Java	Java	Java	PHP	Java
Servicios Web	Si	Si	Si	No	Si	Si
Programación de tareas	No	Si	Si	No	No	Si
Licencia	Privativa (EULA)	Dual (GPL)	Propio	Libre (GPL)	Propio	Libre (GPL)
Ligero	Menos ligero	Menos ligero	Más ligero	Más ligero	Más ligero	Menos ligero
Diseño de reportes	Si	Si	No	Si	Si	Si
Formatos de salida	pdf, xml, html, csv, xls, rtf, txt, jpeg, doc, odt, pptx, docx, ods, xslx	pdf, xml, html, csv, xls, rtf, txt, jpeg, doc, odt, pptx, docx, ods, xslx	pdf, xml, html, csv, xls, rtf, txt, jpeg, doc, odt, pptx, docx, ods, xslx	pdf, xml, html, xhtml csv, xls, rtf, txt, docx, odt, pptx, docx, ods, xslx	html, pdf, xls, csv	html, pdf, xls, csv

Luego de analizar los datos contenidos en la tabla comparativa, fue posible arribar a las siguientes conclusiones: Crystal Report es una herramienta privativa por lo que su uso representa un costo y

JasperServer utiliza licencia dual la cual implica un pago en caso de utilizar la herramienta con fines comerciales. Pentaho Reporting es una herramienta muy poderosa para el diseño de reportes, pero está orientado mayormente a su uso dentro de la plataforma de Pentaho para apoyar la toma de decisiones y ser integrado con el resto de las aplicaciones de la suite y GDR es un diseñador enfocado a entorno web que requiere un sistema de reportes que tenga la menor cantidad de dependencias y permita la convergencia hacia la solución deseada.

La investigación permitió concluir que el diseñador más idóneo es el iReport, debido a que es una herramienta con altas prestaciones en la generación de reportes, enfocada al entorno de escritorio, bajo licencia GPL lo cual facilita su adquisición y contribuye a la soberanía tecnológica aspirada por la universidad y por Cuba. Además está implementado sobre la plataforma NetBeans, en Java al igual que el SDR, este último utilizando las librerías de JasperReport diseñadas para JasperSoft, software del cual iReport es el diseñador nativo, siendo estas dos herramientas de reporte totalmente compatibles, lo que conlleva a que se tuvieran en cuenta estas soluciones libres para una posterior integración.

1.3.7 Extensión

Según Jürgen Petri, autor de “Netbeans Platform 6.9 Developer's Guide”, una extensión es “(…) *un componente de software independiente con el que otras extensiones pueden comunicarse a través de interfaces bien definidas, detrás de las cuales quedan ocultas sus implementaciones. Cada característica importante de una aplicación se crea dentro de una extensión, dándole a cada una de estas una responsabilidad claramente definida dentro de la aplicación*” (PETRI 2010).

Elementos arquitectónicos para extender el iReport

El diseñador de reportes iReport está desarrollado sobre la Plataforma NetBeans. La misma ofrece las ventajas genéricas de una plataforma de cliente enriquecida, basándose en normas internacionales y componentes reutilizables que pueden ser empleados en aplicaciones propias del usuario, brindando elementos de interfaz como ventanas, menús, barras de herramientas y otros componentes como sistemas de ayuda, internacionalización, representación de datos y las API. Esto es logrado mediante la implementación de módulos que se acoplan gracias a la arquitectura modular.

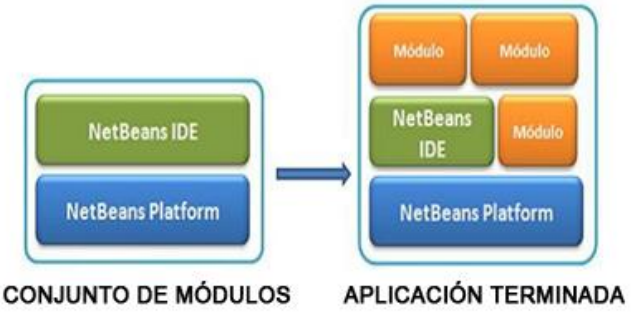


Fig. 1: Arquitectura modular para el desarrollo de aplicaciones con NetBeans IDE y la plataforma NetBeans.

¿Cómo se puede extender mediante módulos el iReport?

El iReport soporta un acoplamiento flexible de los módulos, descrito por los datos de su fichero de manifiesto, junto con los datos especificados en los archivos relacionados con el Lenguaje Extensible de Marcas (XML, del inglés *Extensible Markup Language*). Un módulo o extensión consta de las siguientes partes:

- Manifiesto de archivo (manifest.mf)
- Capa de archivo (layer.xml)
- Los archivos de clases
- Los recursos como iconos, librerías y licencia.

A continuación se muestra parte de la configuración del archivo layer.xml para el menú de una ventana.

```
<folder name="Menu">
 <folder name="Edit">
  <file name="MyAction.shadow">
   <attr name="originalFile"
    stringvalue="Actions/Edit/com-galileo-netbeans-module-
   MyAction.instance"/>
  </file>
 </folder>
</folder>
```

Fig. 2: Configuración del archivo layer.xml para un menú

Otros aspectos de interés en la estructura de los módulos son: el diseño de clases que desarrolla el programador para garantizar el funcionamiento de la solución propuesta y el archivo Bundle.properties el cual permite localizar la información registrada en el archivo de manifiesto.

Una vez comprendido el funcionamiento de los módulos se puede crear un proyecto para extender las funcionalidades del iReport. El entorno provee facilidades que permiten estructurarlo como se muestra en la Fig. 3 partiendo de una plantilla.

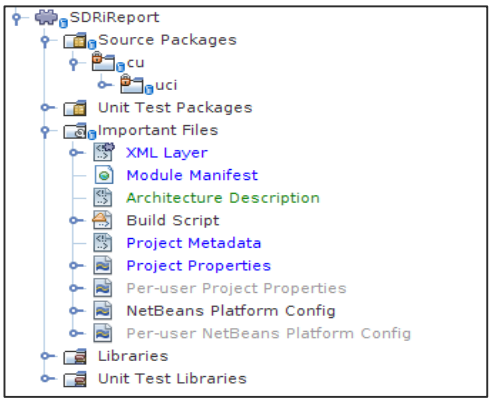


Fig. 3: Estructura de un proyecto para extender el iReport.

1.3.8 API

Para llevar a cabo la extensión será necesario el empleo de las API como bibliotecas de ayuda para garantizar la comunicación entre iReport y el SDR. Francis y Dominique, en su libro "La alquimia de las multitudes", define las API como: "*Puertas que los creadores de los programas abren de forma voluntaria para permitir que otros creadores puedan apropiarse de los elementos que interesen de dichos programas y añadir sus propios servicios*" (PISANI 2009). Las API pueden desarrollarse para un sistema operativo o para todos al mismo tiempo, siendo útiles para acceder a servicios web.

Funcionamiento de las API

Las API de programación representan una interfaz de comunicación entre componentes de software. Uno de los principales propósitos de las API consiste en proporcionar un conjunto de funciones de uso general. De esta forma, los programadores se benefician de las ventajas haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.(ESCALLÓN 2006)

Ventajas de las API

Las API hacen más fácil el desarrollo de un programa, porque proporcionan todos los conceptos para construirlo. Están diseñadas especialmente para los programadores, porque garantizan que todos los programas que utilizan API, tendrán interfaces similares. Esto le facilita al usuario aprender la lógica de nuevos programas. Cuando se realiza una solicitud, el servidor llamará a las API, ofreciendo la ventaja de disponer de una mayor cantidad de servicios (UCA 2012)

1.3.9 Servicios REST

Teniendo en cuenta que el SDR *"atiende peticiones de aplicaciones web o de escritorio a través de servicios REST"* (MENDOZA GARNACHE 2014) se utilizará esta tecnología para facilitar la comunicación entre iReport y el SDR. La Transferencia de Estado Representacional permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP.

1.4 Metodología y herramientas empleadas en la solución

Uno de los aspectos fundamentales en la elaboración de un software es seleccionar las tecnologías y tendencias que mayores beneficios aporten. Para el desarrollo del presente trabajo se realizó un estudio de las posibles metodologías y herramientas a utilizar, coincidiendo el criterio de selección con las definidas por la arquitectura de DATEC por el impacto que manifiestan y las ventajas que poseen. A continuación se hará una descripción detallada de cada una de estas.

1.4.1 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y soporte documental que permite la creación de software. Sirve de guía en el proceso de desarrollo de software y se agrupan en dos grandes grupos, las tradicionales y las ágiles. Las tradicionales se enfocan en el plan del proyecto, estableciendo rigurosamente las actividades implicadas, los artefactos que se deben producir, además de las herramientas y notaciones que se utilizarán. Por otra parte las metodologías ágiles se centran en satisfacer las necesidades del cliente, dándole mayor valor al individuo (TINOCO 2014).

Para caracterizar la extensión a desarrollar y estimar cuán ágil o robusto debía ser el enfoque a utilizar, se emplea el modelo de Boehm & Turner. Este método plantea cinco criterios mediante los que se

estará valorando el proyecto: tamaño del equipo de desarrollo, criticidad del producto, dinamismo de los cambios, cultura del equipo y experiencia del personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta para seleccionar uno u otro enfoque (BOERAS 2012). Después de realizar un estudio de la solución a desarrollar se determinó que esta debería tener un enfoque ágil teniendo en cuenta que el tamaño del equipo de desarrollo es pequeño; la cultura de los desarrolladores es media, siendo capaces de comprender y adaptar los métodos y las técnicas empleadas y la criticidad es mínima porque un fallo en la extensión no trae pérdidas de vidas humanas, daño al medio ambiente ni grandes pérdidas económicas.

OpenUP

El Proceso Unificado Abierto (OpenUp, del inglés *Open Unified Process*) es una metodología ágil que aplica un enfoque iterativo e incremental dirigida a la gestión y desarrollo de proyectos de software. Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo del software. Mantiene las características esenciales del Proceso Unificado de Rational (RUP, del inglés *Rational Unified Process*), permitiendo el desarrollo de aplicaciones mediantes la generación de artefactos ligeros apropiados, e incrementando así las probabilidades de éxito en función del costo, tiempo y alcance (SCRIBD 2010). Esta metodología se conforma por cuatro fases: concepción, elaboración, construcción y transición.

Se decide utilizar la metodología OpenUp como guía en el proceso de desarrollo de software porque esta beneficia el desarrollo de proyectos pequeños y de corto plazo permitiendo disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito. Es modificable porque le permite al cliente incorporar ideas nuevas y evita la elaboración exhaustiva de documentación y diagramas. Además es la metodología de desarrollo propuesta por la línea base de la arquitectura del proyecto SDR.

1.4.2 El lenguaje unificado de modelado

El Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language) permite "(…) *especificar, visualizar y construir los artefactos de los sistemas de software*" (LARMAN 1999). Define los componentes reutilizables, la construcción de los diagramas de clases, los componentes del sistema, despliegue e interacción. En la presente investigación se hace uso de este lenguaje de modelado para realizar una correcta verificación y validación del modelo realizado. Es un lenguaje visual, fácil de

entender por cualquier persona del equipo de desarrollo, además varias herramientas CASE (del inglés *Computer Aided Software Engineering*) utilizan el Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language) como lenguaje de modelado para generar artefactos de un sistema de software, entre las más conocidas se encuentra Visual Paradigm for UML.

1.4.3 Herramienta CASE de modelado

Las herramientas CASE se pueden definir como un conjunto de programas y ayudas que dan asistencia, durante el ciclo de vida de un software, proporcionándole un aumento en su productividad y logrando un mayor ahorro de tiempo (SCRIBD 2014).

Visual Paradigm for UML

Visual Paradigm for UML 8.0 es una herramienta que soporta el ciclo de vida completo de desarrollo de un software: análisis y diseño OO, construcción, pruebas y despliegue. Permite representar gráficamente los diagramas y facilita la generación de código. Además incorpora el soporte para trabajo en equipo, que permite a varios desarrolladores trabajar a la vez en el mismo diagrama viendo en tiempo real los cambios hechos por sus compañeros (PARADIGM 2010).

En el ciclo de desarrollo de la extensión se decide utilizar Visual Paradigm for UML en su versión 8.0 para guiar el modelado de la solución partiendo de que esta herramienta es multiplataforma y presenta una interfaz de usuario intuitiva y sencilla que permite realizar los diagramas y artefactos que se generan durante el desarrollo del software. Es fácil de instalar y actualizar e incluye gran variedad de estereotipos para la creación de diagramas de fácil entendimiento, los cuales se organiza automáticamente. Además es la utilizada por el equipo de trabajo del SDR y la definida por DATEC.

1.4.4 Lenguaje de programación

Un lenguaje de programación es un lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por una computadora. Es una herramienta que permite crear programas y software. Un lenguaje de programación está formado por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, además el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento lógico de una máquina y para expresar algoritmos con precisión. Este lenguaje permite ser leído y escrito por personas y a su vez resulta

independiente del modelo de computadora a utilizar. Existen varios lenguajes de programación tales como: Python, Java y C++ (PROGRAMATION 2010).

Java

Es orientado a objetos y desarrollado por la compañía Sun Microsystems. Inspirado al C++, Java fue proyectado con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos, lo que significa que los programas Java pueden ser ejecutados sobre cualquier computadora en la cual sea instalada la máquina virtual (SOFTWARE 2014). Se decidió utilizar como lenguaje de programación Java en su versión 6.35 porque iReport está implementado sobre este lenguaje.

1.4.5 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*) es una herramienta libre que permite el desarrollo de aplicaciones con características modulares. Entre sus funcionalidades permite escribir, depurar, compilar y ejecutar programas, además soporta otros lenguajes de programación. Es compatible con el lenguaje de programación Java el cual es utilizado para el desarrollo tanto del SDR como de iReport y además es el IDE seleccionado por el grupo de arquitectura del Departamento de Integración de Soluciones. Para la implementación de la extensión se utilizó como entorno de desarrollo integrado el NetBeans en su versión 8.0 por las características antes expuestas.

Plataforma NetBeans

La plataforma NetBeans 6.5 ofrece las ventajas genéricas de una plataforma de cliente enriquecida, además de numerosos marcos como el de interfaz de usuario, el de configuración de usuario y el marco asistente. Es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes porque permite hacer uso de varios elementos que facilitan la construcción de extensiones y aplicaciones con arquitectura modular. Entre estos se encuentra la internacionalización, el editor de datos para agregar funcionalidades y un generador de ayuda para el proyecto que se desarrolla en conjunto con las API y componentes reutilizables (NETBEANS 2010).

La plataforma NetBeans proporciona una arquitectura modular fiable, flexible y fácil para crear aplicaciones que son robustas y extensibles. El diseñador de reportes iReport fue desarrollado sobre

esta plataforma lo que conllevó a que se tomara como plataforma de desarrollo evitando incompatibilidades y asegurando una rápida integración.

1.5 Conclusiones parciales del capítulo

El estudio de los principales conceptos relacionados con los sistemas de generación de reportes y extensiones mediante módulos permitió sentar las bases para el desarrollo de la propuesta de solución, además del porqué, así como la necesidad de la realización del presente trabajo. La investigación realizada sobre servidores similares al SDR en el mundo evidenció que si se quiere pocicionar al nivel de los sistemas de las grandes empresas que trabajan con reportes, debe contar con diseñadores de reportes que exploten sus funcionalidades, surgiendo de esta forma la necesidad de integrarlo con iReport. Al tener en cuenta las políticas y normativas de DATEC y las particularidades de la solución que se propone se selecciona como metodología de desarrollo OpenUp, como entorno de desarrollo integrado NetBeans 8.0 y el lenguaje de programación Java, además de Visual Paradigm for UML 8.0 como herramienta CASE haciendo uso del lenguaje de modelado UML 2.0. Durante el presente capítulo se tuvo en cuenta como principal premisa lograr la libertad tecnológica del producto respetando la política de migración a software libre de Cuba.

Capítulo 2: Análisis y diseño de la extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes

En el presente capítulo se describe la solución propuesta para obtener una noción en general de la extensión al iReport para la gestión de reportes en el SDR. Se realiza el modelo conceptual o de dominio, además de un estudio del negocio permitiendo identificar los RF y los RNF. Se identifican los casos de uso que se implementarán en la extensión describiendo cada uno para guiar el proceso de desarrollo del software.

2.1 Modelo de dominio

Un modelo del dominio es una representación de las clases conceptuales que tiene como objetivo comprender y describir las clases más importantes dentro del contexto del sistema, por lo cual este puede ser tomado como el punto de partida para el diseño del sistema.

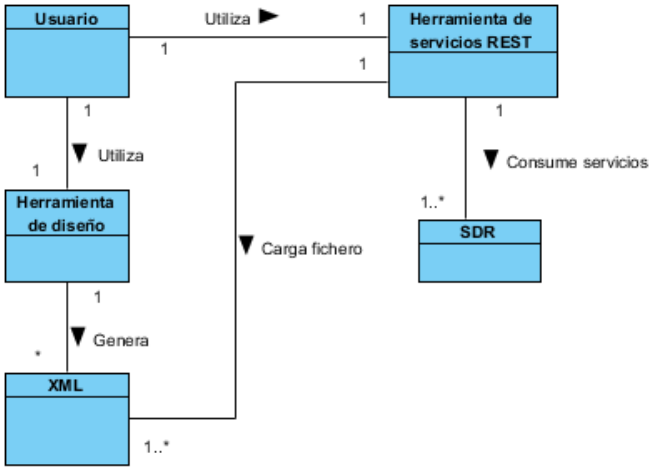


Fig. 4: Modelo del dominio

En el modelo anterior se representa el proceso de gestión de reportes por parte de los usuarios del sistema haciendo uso de una herramienta para el diseño de reportes. Esta herramienta genera una plantilla XML la cual es cargada por el usuario utilizando una herramienta para consumir servicios REST. Una vez cargada esta plantilla consume una serie de servicios que permiten la gestión de los reportes mediante el SDR.

2.1.1 Descripción de las clases del dominio

A continuación se describen explícitamente las clases que componen el modelo de dominio:

1. **Usuario**: Constituyen los clientes del iReport que realizan los reportes en la herramienta de diseño.
2. **Herramienta de diseño**: Constituye una herramienta para diseñar reportes.
3. **XML:** Representa la plantilla del reporte creado en la herramienta de diseño
4. **Herramienta de servicios REST:** Constituye una herramienta a través de la cual se cargarán las plantillas diseñadas y se consumirán los servicios que ofrece el SDR.
5. **SDR:** Representa el servidor donde se publicarán los reportes diseñados en la herramienta de diseño.

A continuación se mencionan los servicios con los que cuenta el SDR.

- ◊ **Gestión de Conexiones**: Se encarga de gestionar conexiones a fuentes de datos permitiendo las opciones de listar, probar, buscar, crear, actualizar y eliminar las variables de conexión a las fuentes de datos.
- ◊ **Gestión de Reportes**: Se encarga de diseñar, listar, publicar, actualizar, validar, compilar y eliminar los reportes del servidor.
- ◊ **Gestión de Reportes Exportados**: Permite exportar, buscar, eliminar, listar y visualizar reportes a un formato específico.
- ◊ **Gestión de Tareas Programadas**: Constituyen tareas que se programan para la exportación automática de reportes en un determinado momento y su envío por correo electrónico o por FTP. Permite adicionar, eliminar, buscar y mostrar tareas programadas.

2.2 Requisitos de la extensión

Según Ian Sommerville en el libro Ingeniería del Software Parte II de la 7ma Edición," (…) *los requisitos del sistema se clasifican en requisitos funcionales (RF) los cuales expresan capacidad y requisitos no funcionales (RNF) los cuales expresan cualidad*." (SOMMERVILLE 2005). El principal objetivo de los mismos es determinar las funcionalidades que tendrá la extensión para guiar el desarrollo de la aplicación hacia la obtención de un diseño que cumpla con las expectativas requeridas por el cliente. La especificación de requisitos es la actividad que enumera y describe los requisitos necesarios para una correcta implementación de la extensión.

2.2.1 Requisitos funcionales de la extensión

Los requisitos funcionales (RF) de la extensión describen con detalles las funcionalidades de la misma, lo que debe hacer, sus entradas y salidas, excepciones, etcétera. Los requisitos funcionales de una aplicación son las capacidades o condiciones que la extensión debe cumplir para satisfacer las necesidades primarias del cliente (SOMMERVILLE 2005). Para la extracción de los requisitos funcionales se utilizó la técnica de recopilación de información tormenta de ideas.

A continuación se definen los requisitos funcionales identificados para el desarrollo de la extensión.

RF1: Consumir servicio “listar conexiones”.
Descripción: El usuario presiona clic sobre la pestaña “Conexiones” y automáticamente se listan todas las conexiones que existen.
Salida: El listado de todas las conexiones que existen en el servidor.

RF2: Consumir servicio “adicionar conexión”.
Descripción: El usuario presiona clic sobre el botón “Añadir” y completa los campos necesarios creando así una nueva conexión.
Entrada: Los detalles de la nueva conexión (nombre, controlador, usuario, contraseña, anfitrión, nombre BD, puerto, URL).
Salida: Actualización del listado de conexiones.

RF3: Consumir servicio “modificar conexión”.
Descripción: El usuario selecciona la conexión que desea modificar y presiona clic sobre el botón “Editar” introduciendo los datos que desea modificar.
Entrada: Los datos de los campos que desea modificar de la conexión (nombre, controlador, usuario, contraseña, anfitrión, nombre BD, puerto, URL)**.**
Salida: La conexión editada.

RF4: Consumir servicio “eliminar conexión”.
Descripción: El usuario selecciona la conexión que desea eliminar y presiona clic sobre el botón “Eliminar”. La extensión muestra un mensaje de confirmación “¿Desea eliminar la selección?”, el usuario presiona clic sobre el botón “Aceptar” eliminándose la conexión seleccionada.
Entrada: Se selecciona de la lista de conexiones la conexión que se desea eliminar.

Salida: Se actualiza la lista de conexiones.

RF5: Consumir servicio “buscar conexión por parámetros”.
Descripción: El usuario presiona clic sobre el botón “Buscar” y selecciona los campos por los que desea filtrar la búsqueda.
Entrada: Criterio de búsqueda.
Salida: La lista de conexiones que corresponda con el criterio de búsqueda.

RF6: Consumir servicio “probar conexión”.
Descripción: El usuario selecciona la conexión que desea probar y presiona clic sobre el botón “Probar conexión” y el sistema muestra el mensaje “Prueba de conexión exitosa”.
Entrada: Se selecciona de la lista de conexiones la conexión que se desea probar.
Salida: Muestra si existe o no conexión.

RF7: Consumir servicio “iniciar sesión”.
Descripción: El usuario presiona clic sobre el botón “Autenticar” se autentica satisfactoriamente y de esta forma inicia sesión.
Entrada: Contraseña y usuario de la persona que desea entrar al sistema.
Salida: Se actualiza el token de autenticación del usuario.

RF8: Consumir servicio “cerrar sesión”.
Descripción: Al usuario cerrar la ventana “Conexión SDR” de la aplicación iReport se cierra la sesión.

RF9: Consumir servicio “listar reportes”.
Descripción: El usuario selecciona la pestaña “Reporte” y se listan automáticamente todos los reportes que existen.
Salida: El listado de todos las reportes que existen en el servidor.

RF10: Consumir servicio “eliminar reporte”.
Descripción: El usuario selecciona el reporte que desea eliminar y presiona clic sobre el botón “Eliminar”. El sistema muestra un mensaje de confirmación “Desea eliminar la selección”, el usuario presiona clic sobre el botón “Aceptar” eliminándose el reporte seleccionado.
Entrada: Se selecciona de la lista de reportes el que se desea eliminar.

Salida: Se actualiza la lista de reportes.

RF11: Consumir servicio “adicionar reporte”.
Descripción: El usuario presiona clic sobre el botón “Añadir” introduce los datos necesarios y crea un nuevo “reporte”.
Entrada: Detalles del nuevo reporte que se desea adicionar (listar, eliminar, adicionar, modificar, buscar, exportar, vista previa).
Salida: Actualización del listado de reportes.

RF12: Consumir servicio “modificar reporte”.
Descripción: El usuario selecciona el reporte que desea modificar y presiona clic sobre el botón “Editar” introduciendo los datos nuevos.
Entrada: Los datos de los campos que desean modificar del reporte.
Salida: Reporte modificado.

RF13: Consumir servicio “buscar reporte por parámetros”.
Descripción: El usuario presiona clic sobre el botón “Buscar” y completa los campos por los que desea filtrar la búsqueda.
Entrada: Criterio de búsqueda.
Salida: La lista de reportes que corresponda con el criterio de búsqueda.

RF14: Consumir servicio “exportar reporte”.
Descripción: El usuario selecciona el reporte que desea exportar y presiona clic sobre el botón “Exportar”.
Entrada: Formato al que se desea exportar el reporte.
Salida: El reporte exportado.

RF15: Consumir servicio “vista previa del reporte”.
Descripción: El usuario selecciona el reporte que desea visualizar y presiona clic sobre el botón “Descargar”. Luego el sistema le muestra un mensaje al usuario preguntándole si desea visualizar el reporte.
Salida: El reporte visualizado.

RF16: Consumir servicio “descargar JRXML de reporte”.

Descripción: El usuario selecciona el reporte que desea descargar y presiona clic sobre el botón “Descargar”.

Salida: El JRXML del reporte descargado.

RF17: Consumir servicio “modificar JRXML”.

Descripción: El usuario selecciona el reporte JRXML que desea modificar, presiona clic sobre el botón “Editar”, luego selecciona la opción “Editar Archivo” y carga el archivo modificado. El usuario presiona clic sobre el botón “Editar” y se actualiza el archivo JRXML en el servidor.

Entrada: Los datos de los campos que desean modificar del JRXML (nombre, descripción, categoría, id).

Salida: El JRXML modificado.

RF18: Consumir servicio “listar tareas programadas”.

Descripción: El usuario selecciona la pestaña “Tareas Programadas” y las tareas se listan automáticamente.

Salida: El listado de todas las tareas programadas que existen en el servidor.

RF19: Consumir servicio “adicionar tarea programada”.

Descripción: El usuario presiona clic sobre el botón “Adicionar” e introduce los datos necesarios creando una nueva conexión.

Entrada: Se registran los detalles de la nueva tarea programada (nombre, controlador, usuario, contraseña, anfitrión, nombre BD, puerto, URL).

Salida: Actualización del listado de tareas programadas.

RF20: Consumir servicio “modificar tarea programada”.

Descripción: El usuario selecciona la tarea que desea modificar y presiona clic sobre el botón “Modificar”.

Entrada: Los datos de los campos que desean modificar de la tarea programada.

Salida: La tarea programada modificada.

RF21: Consumir servicio “eliminar tarea programada”.

Descripción: El usuario selecciona la tarea que desea eliminar y presiona clic sobre el botón “Eliminar”.

Entrada: Se selecciona de la lista tareas programadas la que se desea eliminar.
Salida: Se actualiza la lista de tareas programadas.

RF22: Consumir servicio “buscar tarea programada por parámetros”.
Descripción: El usuario presiona clic sobre el botón “Buscar” y completa los campos por los que desea filtrar la búsqueda.
Entrada: Criterio de búsqueda.
Salida: La lista de tareas programadas que corresponda con el criterio de búsqueda.

RF23: Consumir servicio “listar reportes exportados”.
Descripción: El usuario selecciona la pestaña “Exportaciones” y los reportes exportados se listan automáticamente.
Salida: El listado de todos los reportes exportados que existen en el servidor.

RF24: Consumir servicio “visualizar reporte exportado”.
Descripción: El usuario selecciona el reporte que desea visualizar y presiona clic sobre el botón “Descargar”. El sistema muestra un mensaje preguntándole al usuario si desea visualizar el reporte.
Salida: Se muestra el reporte exportado.

RF25: Consumir servicio “eliminar reporte exportado”.
Descripción: El usuario selecciona el reporte exportado que desee eliminar y luego presiona clic sobre el botón “Eliminar”.
Salida: Se actualiza la lista de reportes exportados.

RF26: Consumir servicio “buscar reporte exportado por parámetros”.
Descripción: El usuario presiona clic sobre el botón “Buscar” y completa los campos por los que desea filtrar la búsqueda.
Entrada: Criterio de búsqueda.
Salida: La lista de reportes exportados que corresponda con el criterio de búsqueda.

2.2.2 Requisitos no funcionales de la extensión

Los requisitos no funcionales, como su nombre sugiere, son aquellos que no se refieren específicamente a las funciones que proporciona el software, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (SOMMERVILLE 2005). Son propiedades o cualidades fundamentales en el éxito del producto que lo hacen más atractivo, usable, rápido o confiable. A continuación se definen los requisitos no funcionales identificados para el desarrollo de la extensión.

- ◊ **Requisitos de Usabilidad**

RNF1: La extensión está diseñada de forma que las interfaces son descriptivas, permitiendo que las operaciones que realizan los usuarios estén detalladas.

RNF2: Se realizará un Manual del Usuario y un Manual de instalación para guiar al usuario durante la instalación y uso de la extensión.

- ◊ **Requisitos de Eficiencia**

RNF3: Requisitos mínimos de hardware para el funcionamiento de la extensión.

1. Procesador: 800MHz Intel Pentium III o equivalente.
2. Memoria RAM: 512 MB.
3. Espacio libre en disco duro: 750 Mb.

- ◊ **Requisitos de restricción de diseño**

RNF4: Requisitos de software.

1. Máquina virtual de Java V 6.0.

- ◊ **Requisitos de soporte**

RNF5: Una vez integrado el iReport con el Servidor Dinámico de Reportes se podrá ejecutar en diferentes sistemas operativos por ser el iReport multiplataforma.

◊ **Requisitos legales y de derecho de autor**

RNF6: Los requisitos legales y el derecho de autor serán registrados por la Universidad de la Ciencias Informáticas.

2.3 Diagrama de Casos de uso

Según el ingeniero de software estadounidense Roger S. Pressman un Caso de uso es*" (…) simplemente, un texto escrito que describe el papel de un actor que interactúa (…)*". Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto, estos determinan los requisitos funcionales de la extensión y representan las funciones que puede realizar.

Después de identificados los requisitos funcionales fueron agrupados en 5 casos de usos que se listan a continuación:

◊ Autenticación de usuarios.
◊ Gestión de conexión.
◊ Gestión de reportes.
◊ Gestión de reportes exportados.
◊ Gestión de tareas programadas.

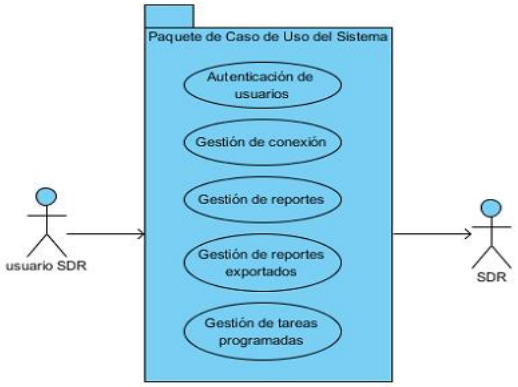


Fig. 5: Diagrama de casos de uso de la extensión.

Descripción de los Casos de uso de la extensión:

◊ **Autenticación de usuario:** Permite realizar el proceso de iniciar y cerrar sesión de los usuarios de la extensión.

◊ **Gestión de conexión:** Permite adicionar, editar, eliminar, buscar, listar o probar una conexión.

◊ **Gestión de reportes:** Permite adicionar, editar, eliminar, buscar, listar, exportar, visualizar o descargar un reporte.

◊ **Gestión de reportes exportados:** Permite eliminar, buscar, visualizar o listar un reporte exportado.

◊ **Gestión de tareas programadas:** Permite adicionar, editar, eliminar, buscar y listar una tarea programada.

2.3.1 Patrones de Casos de uso

"Los patrones de Caso de uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso de la extensión y cómo este interactúa con los usuarios" (RIVERO 2013). Para la confección del Diagrama de Caso de uso del sistema se utilizó el patrón CRUD (del inglés *Create Read Update Delete*) Completo y el patrón Múltiples actores: Roles Distintos. A continuación se describen ambos patrones.

CRUD: Completo

En el diseño de la extensión se utilizó el patrón de Caso de uso CRUD Completo para modelar todas las operaciones que pueden ser realizadas sobre una parte de la información, tales como la creación, lectura, actualización y eliminación. Ejemplo de esto se evidencia en los Casos de uso Gestión de reportes, Gestión de tareas programadas, Gestión de conexión y Gestión de reportes exportados representados en el diagrama de Casos de uso del sistema (Ver Fig. 5).

Múltiples actores: Roles Distintos

En el diseño de la extensión se utilizó el patrón de Caso de uso Múltiples actores para modelar la condición de dos actores (usuario SDR y SDR) que interactúan de forma diferente con un mismo Caso de uso (Ver Fig. 5).

A continuación se muestra la descripción detallada del Caso de uso arquitectónicamente significativo **Gestión de reportes** donde se especifican: resumen, complejidad, actores que intervienen, prioridad, precondiciones, pos-condiciones, flujos básicos y flujos alternos del Caso de uso**.**

Tabla 2: Especificación del Caso de uso del sistema "Gestión de reportes"

Caso de uso	Gestión de reportes.
Actores	Usuario.
Resumen	El Caso de uso inicia cuando el usuario elige la opción adicionar, modificar, buscar, listar, exportar, vista previa, obtener o eliminar reporte. El Caso de uso termina cuando el sistema muestra el resultado de la petición realizada.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	El usuario ha sido autenticado exitosamente en el sistema y le ha sido asignado un token de autenticación.
Pos-condiciones	- Se adiciona el reporte a la base de datos del servidor y queda almacenado en el sistema de ficheros del disco duro. - Se elimina el reporte deseado de la base de datos del servidor. - Se modifica el reporte deseado y queda almacenado en el sistema de ficheros del disco duro. - Se listan los reportes que cumplan con un criterio de búsqueda. - Se busca el reporte deseado en la base de datos del sistema. - Se modifica el fichero JRXML deseado. - Se exporta el reporte deseado. - Se descarga el fichero JRXML deseado. - Se visualiza el reporte deseado.
Flujo de eventos	
Flujo básico Gestión de reportes	
Actor	**Sistema**

	1. El usuario presiona clic sobre la pestaña “Reportes”	2. El sistema muestra una ventana donde permite realizar varias acciones con un reporte: - Adicionar un reporte. *Ver Sección 1: Consumir servicio adicionar reporte.* - Buscar un reporte. *Ver Sección 2: Consumir servicio b*uscar reporte por parámetros*.* - Listar reportes. *Ver Sección 3: Consumir servicio listar reporte.* - Exportar un reporte. *Ver Sección 4: Consumir servicio exportar reporte.* - Ver la vista previa de un reporte. *Ver Sección 5: Consumir servicio vista previa del reporte.* - Modificar un reporte. *Ver Sección 6: Consumir servicio modificar reporte.* - Descargar el fichero JRXML de un reporte*. Ver Sección 7: Consumir servicio descargar JRXML de reporte.* - Eliminar un reporte. *Ver Sección 8: Consumir servicio eliminar reporte.* - Modificar el fichero JRXML de un reporte. *Ver Sección 9: Consumir servicio modificar JRXML.*
Sección 1: “Consumir servicio adicionar reporte”		
Flujo básico Consumir servicio adicionar reporte		
	Actor	**Sistema**
	1. El usuario selecciona la pestaña “Reportes” y luego presiona clic sobre el botón “Añadir”.	2. El sistema muestra una ventana con los campos que el usuario debe llenar para adicionar un reporte (Nombre, Descripción, Categoría, Archivo).
	3. El usuario completa los	4. El sistema adiciona correctamente el reporte

	campos y presiona clic sobre el botón “Añadir”.	listándolo automáticamente”. En caso contrario ver **Flujo Alterno 1a ¨ Campo nombre vacío ¨ *o* 2a ¨ JRXML incorrecto.¨**
		5. Termina así el Caso de uso.

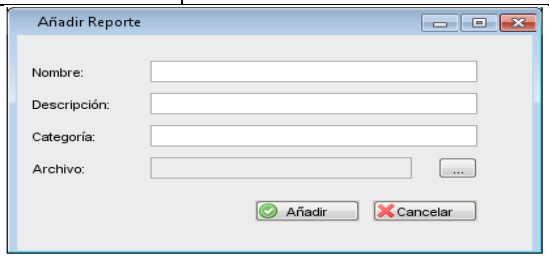


Fig. 6: Prototipo de interfaz de usuario # 1: Añadir reporte.

Flujos alternos

1a ¨ Campo nombre vacío ¨

	Actor	**Sistema**
		1. El sistema le muestra un mensaje al usuario “El campo nombre no puede estar vacío”. Luego se va al paso tres del flujo básico de eventos.

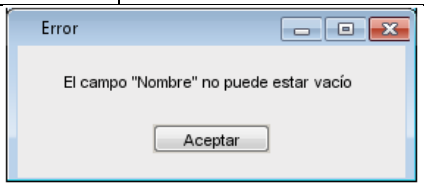


Fig. 7: Prototipo de interfaz de usuario # 2: Error en el nombre.

2a ¨ JRXML incorrecto ¨

	Actor	**Sistema**
		1. 1. El sistema le muestra un mensaje al usuario “Seleccione el archivo JRXML”. Luego se va al paso tres del flujo básico de eventos.

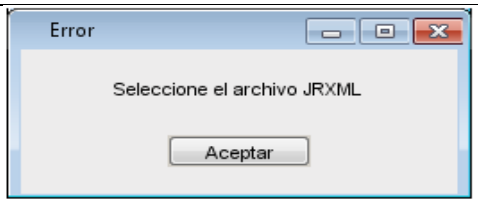


Fig. 8: Prototipo de interfaz de usuario # 3: Error en el archivo JRXML.

Sección 2: "Consumir servicio buscar reporte por parámetros"

Flujo básico Consumir servicio buscar reporte por parámetros

	Actor	**Sistema**
	1. El usuario selecciona la opción "Avanzada".	2. El sistema le muestra al usuario la ventana del buscador.
	3. El usuario filtra la búsqueda por los parámetros que desee.	4. El sistema le muestra al usuario el o los reporte(s) cuyos parámetros coincidan con los introducidos por el usuario, terminando así el Caso de uso.

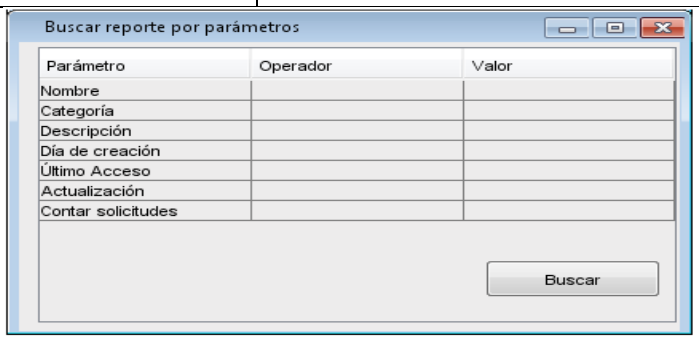


Fig. 9: Prototipo de interfaz de usuario #4 "Buscar por parámetros".

Sección 3: "Consumir servicio listar reporte"

Flujo básico Consumir servicio listar reporte

	Actor	**Sistema**
	1. El usuario selecciona la pestaña "Reporte".	2. El sistema lista todos los reportes que existen.
		3. Termina así el Caso de uso.

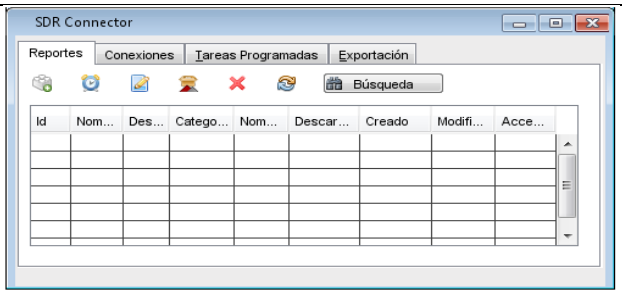


Fig. 10: Prototipo de interfaz de usuario #5 “Listar reportes”.

Sección 4: “Consumir servicio exportar reporte”

Flujo básico Consumir servicio exportar reporte

	Actor	Sistema
	1. El usuario selecciona la pestaña “Reporte” y luego selecciona el reporte que desea exportar, presionando posteriormente clic sobre el botón “Exportar”.	2. El sistema muestra una ventana con una conexión y un combobox para seleccionar el formato de reporte deseado, sino se selecciona un formato, el sistema lo exporta en pdf.
	3. El usuario presiona clic sobre el botón exportar.	4. El sistema muestra un mensaje “El reporte ha sido descargado, desea visualizarlo”.
	5. Si el usuario habilita dicha opción el reporte se carga dependiendo de un visor externo, en caso contrario el reporte se guarda en la dirección que el usuario especifique.	6. Termina así el Caso de uso.

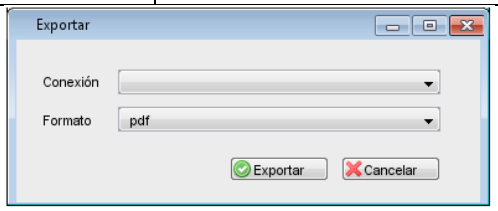


Fig. 11: Prototipo de interfaz de usuario #6 “Exportar reportes”.

Sección 5: “Consumir servicio modificar reporte”

Flujo básico Consumir servicio modificar reporte		
	Actor	**Sistema**
	7. El usuario selecciona el reporte que desea modificar y presiona clic sobre el botón "Editar".	8. El sistema muestra una ventana con los atributos modificables de un reporte (Nombre, Descripción, Categoría), listos para ser editados.
	3. El usuario hace todos los cambios que desee sobre el reporte seleccionado y presiona clic sobre el botón "Editar".	4. El sistema modifica correctamente los datos. En caso contrario *ver Flujo Alterno 1a. ¨* **Campo nombre vacío***¨.*
		5. Termina así el Caso de uso.

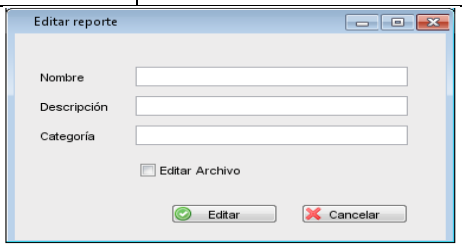


Fig. 12: Prototipo de interfaz de usuario #7 "Editar reportes".

Flujos alternos		
1a ¨ Campo nombre vacío ¨		
	Actor	**Sistema**
		1. Si el campo "Nombre" queda vacío el sistema le muestra un mensaje al usuario "El campo nombre no puede estar vacío". Luego se va al paso tres del flujo básico de eventos.

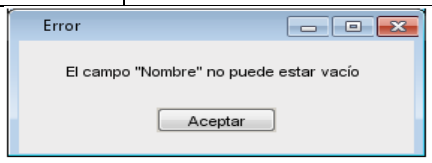


Fig. 13: Prototipo de interfaz de usuario #8. Error_nombre.

Sección 6: "Consumir servicio modificar JRXML"
Flujo básico Consumir servicio modificar JRXML

	Actor	**Sistema**
	1. El usuario selecciona el reporte que desea modificar y presiona clic sobre el botón “Editar”.	2. El sistema muestra una ventana con nombre, descripción, categoría), y un recuadro de selección “Editar Archivo”.
	3. El usuario completa los campos deseados y habilita el campo de selección.	4. El sistema genera automáticamente un “Archivo” con un botón “Seleccionar” donde permite cambiar el archivo.jrxml por otro cualquiera.
	5. El usuario presiona clic sobre el botón “Seleccionar” y elige el nuevo archivo.jrxml y presión clic sobre el botón “Editar”.	6. El sistema guarda los cambios. En caso contrario *ver Flujo Alterno 1a.* ¨ **Campo nombre vacío¨ o 2a ¨ Archivo JRXML inválido¨.**
		7. Termina así el Caso de uso.

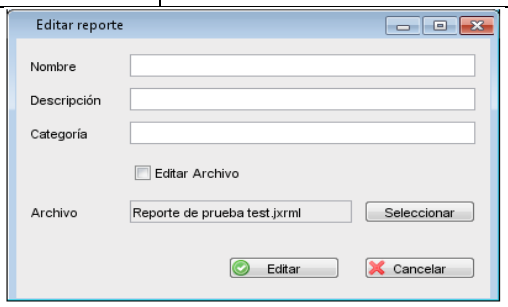


Fig. 14: Prototipo de interfaz de usuario #9 “Modificar JRXML”.

Flujos alternos

1a ¨ Campo nombre vacío ¨

	Actor	**Sistema**
		1. Si el campo “Nombre” queda vacío el sistema le muestra un mensaje al usuario “El campo nombre no puede estar vacío”. Luego se va al paso tres del flujo básico de eventos.

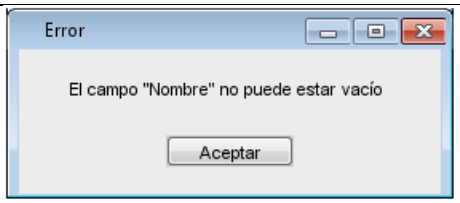


Fig. 15: Prototipo de interfaz de usuario #10. Error_nombre.

Flujos alternos

2a ¨ Archivo JRXML inválido¨

	Actor	Sistema
		1. Si se habilita el campo “Editar Archivo” y por algún motivo ese archivo no existe, el sistema le muestra un mensaje al usuario “Seleccione un archivo jrxml o deshabilite la opción "Editar Archivo". Luego se va al paso cinco del flujo básico de eventos.

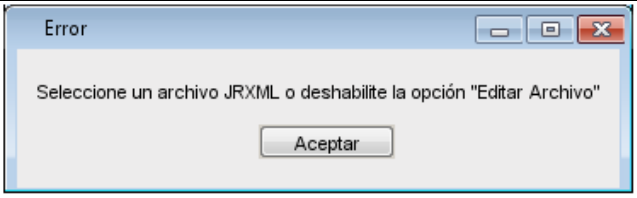


Fig. 16: Prototipo de interfaz de usuario #11. Error_jrxml.

Sección 7: “Consumir servicio eliminar reporte”

Flujo básico Consumir servicio eliminar reporte

	Actor	Sistema
	1. El usuario selecciona el reporte que desea eliminar y presiona clic sobre el botón “Eliminar”.	2. El sistema muestra un mensaje de confirmación preguntándole al usuario si está seguro que desea eliminar dicho reporte.
	9. El usuario presiona clic sobre el botón “Aceptar”.	10. El sistema elimina el reporte deseado. Termina así el Caso de uso.

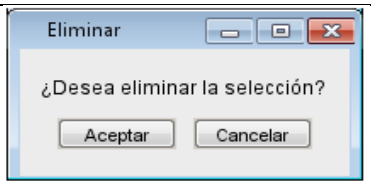


Fig. 17: Prototipo de interfaz de usuario #12 “Eliminar reportes”.

Sección 8: “Consumir servicio modificar JRXML”		
Flujo básico Consumir servicio modificar JRXML		
	Actor	**Sistema**
	1. El usuario selecciona el reporte que desea modificar y presiona clic sobre el botón “Editar”.	2. El sistema muestra una ventana con 3 de los 4 atributos de un reporte (Nombre, Descripción, Categoría), y un recuadro de selección “Editar Archivo”.
	3. El usuario marca el recuadro de selección.	4. El sistema genera automáticamente un label “Archivo” con un botón “Seleccionar” donde permite cambiar el archivo.jrxml por otro cualquiera.
	5. El usuario presiona clic sobre el botón “Seleccionar” y elige el nuevo archivo.jrxml.	6. El sistema guarda los cambios, terminando así el Caso de uso.
Sección 9: “Consumir servicio adicionar tarea programada”		
Flujo básico Consumir servicio adicionar tarea programada opción email		
	Actor	**Sistema**
	1. El usuario presiona clic sobre el botón “Programar”.	2. El sistema muestra una ventana con los campos nombre, conexión, opción, correo, formato y frecuencia.
	3. El usuario completa los campos y presiona clic sobre el botón “Añadir” creando así una nueva tarea programada.	

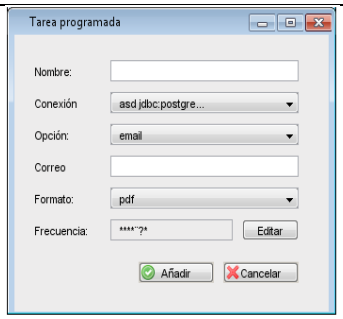


Fig. 18: Prototipo de interfaz de usuario #13 “Tarea programada opción email”.

Flujo básico Consumir servicio adicionar tarea programada opción ftp

	Actor	Sistema
	1. El usuario presiona clic sobre el botón “Programar”.	2. El sistema muestra una ventana con los campos nombre, conexión, opción, usuario, contraseña, dirección ip, ubicación, formato y frecuencia.
	3. El usuario completa los campos y presiona clic sobre el botón “Añadir”.	4. El sistema añade una nueva tarea programada. En caso contrario ver **Flujo Alterno 1a ¨ Campo nombre vacío ¨ *o* 2a ¨ Campo correo inválido ¨.** y así finaliza el Caso de uso.
		5. Termina así el Caso de uso.

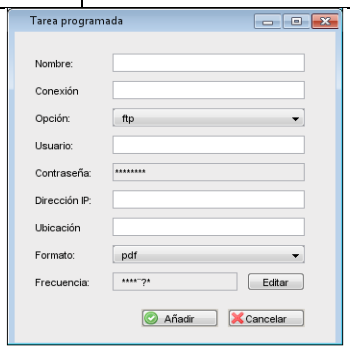


Fig. 19: Prototipo de interfaz de usuario #14 “Tarea programada opción ftp”.

Flujos alternos

1a ¨Campo nombre vacío¨.

	Actor	Sistema

		1. El sistema le muestra un mensaje al usuario “El campo nombre no puede estar vacío”. Luego se va al paso tres del flujo básico de eventos.

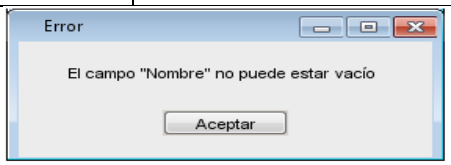


Fig. 20: Prototipo de interfaz de usuario #15. Error_nombre.

2a ¨Campo correo inválido ¨

	Actor	**Sistema**
		1. El sistema le muestra un mensaje al usuario “El campo correo no es un correo válido”. Luego se va al paso tres del flujo básico de eventos.

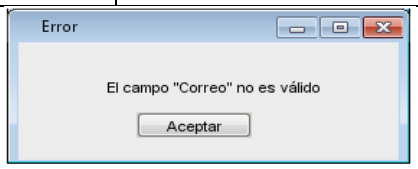


Fig. 21: Prototipo de interfaz de usuario #16. Error_correo_inválido.

3a ¨Campo usuario inválido ¨

	Actor	**Sistema**
		1. El sistema le muestra un mensaje al usuario “El campo usuario debe contener solo caracteres alfanuméricos en minúscula comenzando con letra”. Luego se va al paso tres del flujo básico de eventos.

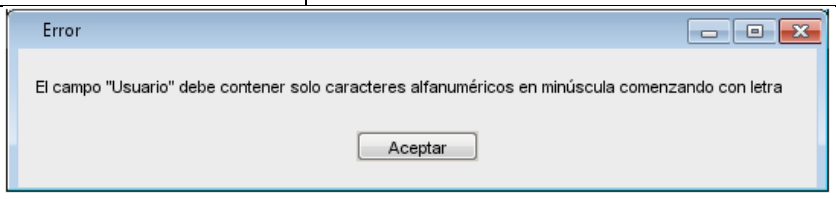


Fig. 22: Prototipo de interfaz de usuario #17. Error_usuario_inválido.

4a ¨Campo contraseña inválida ¨

	Actor	**Sistema**

		1. El sistema le muestra un mensaje al usuario ““El campo “Contraseña” debe tener una cantidad de caracteres mayor que 8 y menor que 100”. Luego se va al paso tres del flujo básico de eventos.

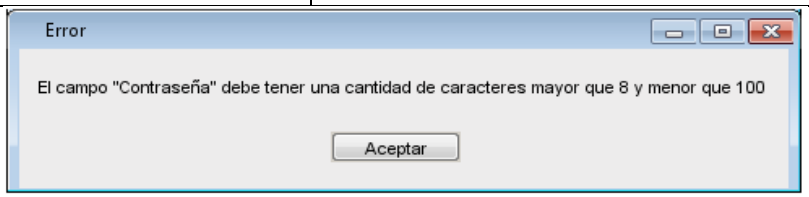


Fig. 23: Prototipo de interfaz de usuario #18. Error_contraseña_inválida.

5a ¨ Campo dirección ip inválida ¨

	Actor	**Sistema**
		1. El sistema le muestra un mensaje al usuario “El campo “El campo Dirección Ip no es un ip válido”. Luego se va al paso tres del flujo básico de eventos.

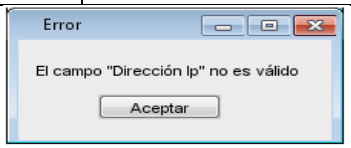


Fig. 24: Prototipo de interfaz de usuario #19. Error_dirección_ip.

6a ¨ Campo ubicación vacío¨.

	Actor	**Sistema**
		1. El sistema le muestra un mensaje al usuario “El campo "Ubicación" debe contener la dirección donde la tarea programada debe almacenar el reporte”. Luego se va al paso tres del flujo básico de eventos.

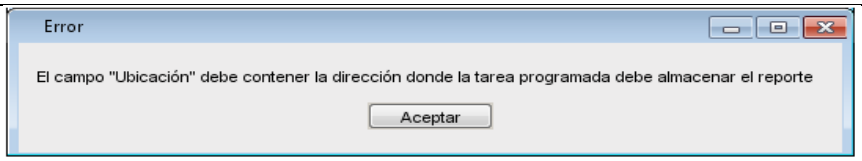


Fig. 25: Prototipo de interfaz de usuario #20. Error_ubicación.

2.4 Modelo de diseño

El modelo de diseño describe la realización de Casos de uso, sirviendo como una abstracción del modelo de aplicación y su código fuente. Se utiliza como parte esencial para las actividades en ejecución y prueba, basándose en el análisis además de los requisitos de la arquitectura de la extensión. Representa los componentes de aplicación y determina su colocación adecuada y el uso dentro de la arquitectura en general de la extensión (RIVERO 2013).

2.4.1 Diagrama de paquetes

La organización física de los elementos que conforman la extensión se realizará teniendo en cuenta los principios establecidos por NetBeans IDE que estructuran el proyecto en paquetes. En la Fig. 26 se muestra detalladamente la estructura del paquete Extensión.

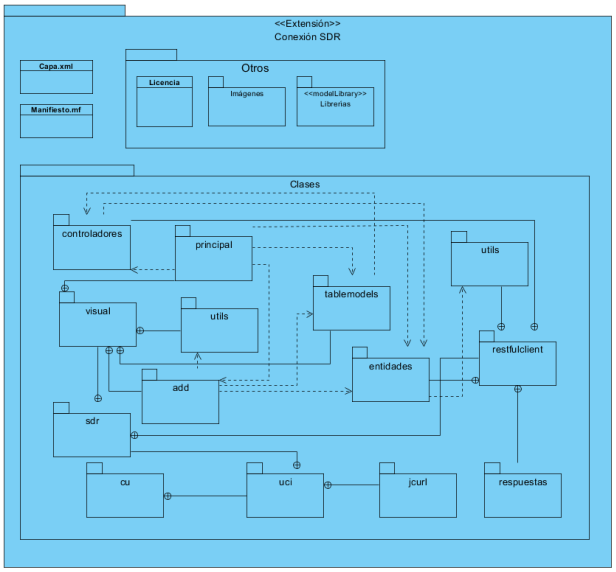


Fig. 26: Diagrama de paquetes de la extensión

La extensión está compuesta por una carpeta denominada “controladores” donde se almacenan todos los controladores de la extensión. Estos harán uso de las entidades correspondientes en dependencia de las acciones que realicen. La carpeta “entidades” contiene las entidades que se utilizan en la extensión (Report, Connection, Task y Export). La carpeta “principal” contiene los principales componentes del visual.

2.4.2 Arquitectura utilizada

La selección de una arquitectura basada en módulos se centra en la utilización del IDE NetBeans para el desarrollo de la extensión. La Plataforma NetBeans ofrece las características de una aplicación de cliente enriquecida en base a su arquitectura modular, formada por un conjunto de API, en donde cada módulo se identifica como una nueva funcionalidad. Ofrece facilidades en cuanto a la integración, reutilización de módulos, implementación manejable, rapidez y detección de errores. La arquitectura modular está estructurada por 4 paquetes fundamentales como se muestra en la Fig. 27.

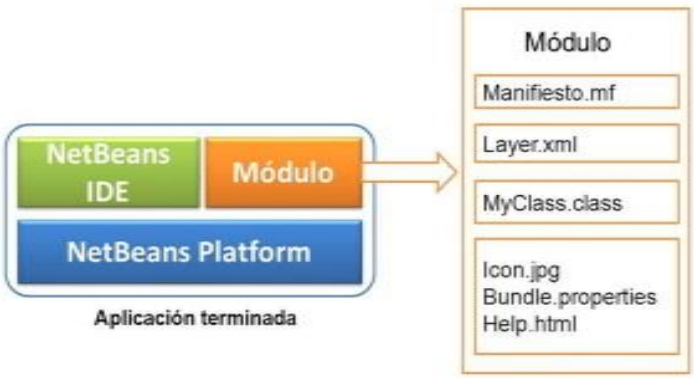


Fig. 27: Arquitectura modular del NetBeans IDE.

Manifest.mf: Es un fichero que contiene la descripción textual del módulo y su entorno. Cuando se carga un módulo, este es el primer archivo leído por el sistema de módulos.

Layer.xml: Proporciona información específica del usuario y define la integración de un módulo en la plataforma. El mismo detalla todo lo que un módulo quiere añadir a la plataforma, que va desde las acciones de los elementos de menú a los servicios. Además este formato es un sistema jerárquico que contiene las carpetas, archivos y atributos.

MyClass.class: Proporciona un paquete donde se almacenan todas las clases empleadas en el desarrollo de la extensión.

Otros: Proporciona un paquete donde se almacenan los recursos útiles como iconos, librerías y licencia entre otros.

2.4.3 Diagrama de clases del diseño

El diagrama de clase del diseño describe gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación, conteniendo información como: clases, asociaciones y atributos, interfaces, con sus operaciones y constructores, métodos, tipos de atributos y dependencias. En la Fig. 28 se muestra la representación del diagrama de clase del diseño del Caso de uso arquitectónicamente significativo “Gestión de reportes”.

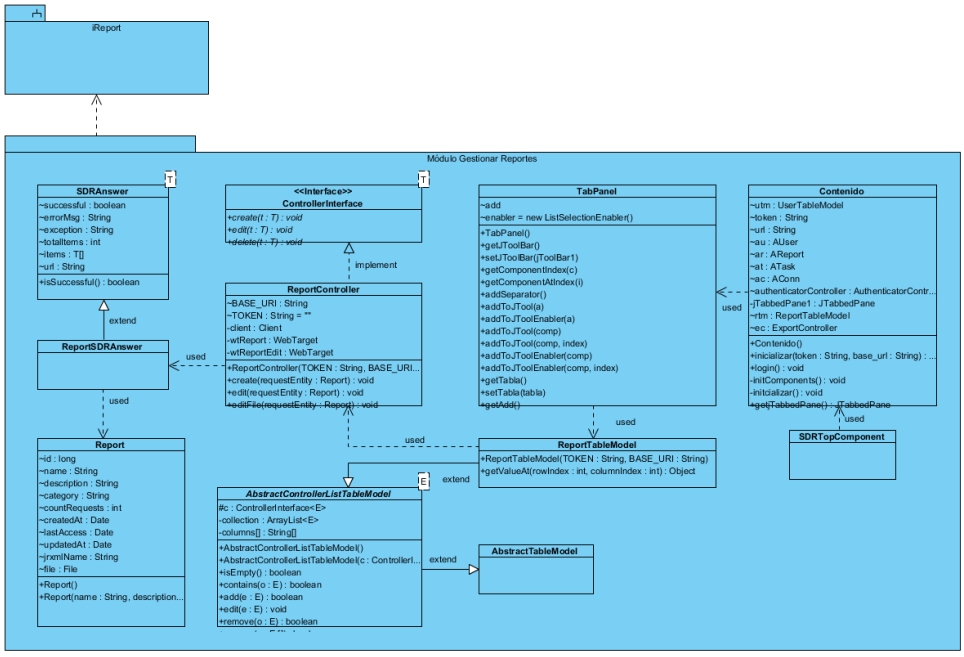


Fig. 28: Diagrama de clase del diseño del Caso de uso “Gestión de reportes”.

La clase “SDRTopComponent” contiene un atributo de clase “Contenido” el cual le permite acceder a todos los atributos de la clase “Contenido”. Este a su vez contiene cuatro atributos de tipo “TabPanel” que permiten mostrar y gestionar las entidades. Según el modelo de tabla (“ReportTableModel”), “TabPanel” se comunica con su correspondiente controlador “ReportController” el cual permite gestionar

los reportes. La clase "ReportSDRAnswer" representa la respuesta enviada por el servidor la cual contiene entre otros atributos la entidad "Report".

La clase "ReportTableModel" extiende de la clase abstracta "AbstractControllerListTableModel", la cual a su vez extiende de la clase "AbstractTableModel". La clase "ReportController" implementa una interface "ControllerInterface" y la clase ReportSDRAnswer extiende de la clase SDRAnswer.

A continuación se realiza una breve descripción de los elementos que componen el Diagrama de clases del diseño para el Caso de uso "Gestión de reportes" que se representa en la imagen anterior.

Tabla 3: Descripción de las clases principales del diagrama de clases

Nombre: SDRTopComponent
Descripción: Esta clase permite al usuario la gestión de reportes. Crea el área de gestión y muestra las propiedades de los reportes. Posee el acceso a las funcionalidades adicionar, editar, exportar y eliminar reportes.
Nombre: Contenido
Descripción: Esta clase permite al usuario gestionar los reportes que serán utilizados en el área de trabajo. Provee varias funcionalidades que están unidas a los reportes que se manejan en el área de trabajo de la clase SDRTopComponent, permitiendo adicionar, editar, exportar y eliminar reportes, además de mostrar las propiedades para los reportes listados.
Nombre: TabPanel
Descripción: Esta clase es la encargada de gestionar entidades. Representa una interfaz genérica de usuario que permite gestionar visualmente entidades. Dentro de las funciones genéricas que posee se encuentra adicionar, editar, eliminar y actualizar página.
Nombre: ReportTableModel
Descripción: Esta clase representa el modelo de la tabla de reportes contenida en el área de trabajo. Posee los atributos de los reportes que se muestran en la tabla, además la lista de reportes.
Nombre: ReportController
Descripción: Esta clase es la encargada de gestionar los reportes. Permite añadir, editar, exportar y eliminar reportes consumiendo los servicios brindados por el SDR.
Nombre: ReportSDRAnswer
Descripción: Esta clase contiene la respuesta enviada por el servidor cuando se gestionan los

reportes.
Nombre: Report
Descripción: Esta clase contiene las propiedades de los reportes, o sea los diseños de los reportes enviados por los usuarios al servidor.

2.4.4 Patrones de diseño

Los patrones de diseño, constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño Orientado a Objetos (OO, del inglés *Object Oriented*). Relacionados con el diseño de los objetos y marcos de trabajo de pequeña y mediana escala. Aplicables al diseño de una solución para conectar los elementos de gran escala que se definen mediante los patrones de arquitectura, y durante el trabajo de diseño detallado para cualquier aspecto del diseño local. También se conocen como patrones de micro-arquitectura. Los patrones de diseño software son los que permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos (LARMAN 2002). Para la solución se aplicaron los siguientes patrones de diseño:

GRASP (del inglés *General Responsibility Assignment Software Patterns*)**:** Son patrones de asignación de responsabilidades que constituyen un apoyo para la enseñanza permitiendo entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (LARMAN 2002).

- ◊ **Experto**: Este patrón se utiliza con frecuencia en la asignación de responsabilidades. Está diseñado para que la responsabilidad de realizar una labor sea de la clase que tiene o puede tener los atributos involucrados, su teoría radica en otorgar responsabilidades a los individuos con la información necesaria para llevar a cabo una tarea (LARMAN 2002). Este patrón se evidencia en las clases *Task, Report, Connection y Export.*
- ◊ **Alta cohesión**: Grady Booch establece que existe alta cohesión funcional cuando los elementos de un componente "(…) *trabajan todos juntos para proporcionar algún comportamiento bien delimitado*", o sea una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas (LARMAN 2002). Este patrón se utilizó en la mayoría de las clases de la extensión porque se asigna la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las

responsabilidades que debe manejar cada clase, evadiendo un trabajo excesivo. Esto permite una mayor claridad en el entendimiento de la extensión y una mayor capacidad de reutilización.

- ◊ **Bajo Acoplamiento**: "Una clase con "bajo acoplamiento" confía poco en otras clases (LARMAN 2002). Este patrón guía la asignación de responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, de manera que de producirse una modificación en algunas de ellas se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las mismas. Este patrón está presente, al igual que el patrón Alta cohesión, en la mayoría de las clases de la extensión debido a que muchas de estas son muy genéricas por naturaleza, y con una probabilidad de reutilización alta.
- ◊ **Controlador**: "Normalmente, un controlador delega en otros objetos el trabajo que se necesita hacer y coordina o controla la actividad. No realiza mucho trabajo por sí mismo" (LARMAN 2002). La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica operada por un usuario. En este caso, como se está diseñando orientado a objetos hay que elegir los controladores que manejen los eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Se evidencia en las clases *Contenido* y *TabPanel* que son las que reciben los eventos de añadir modificar eliminar, etcétera y según sea el caso se maneja o controla el evento.

GOF (del inglés *Gang of Four*)**:** Conocidos también como Grupo de los Cuatro, se consideran generalmente la base para todos los demás modelos. Se clasifican en tres grupos: creacional, estructural y de comportamiento.

- ◊ **Singleton**: Pertenece al grupo creacional. Asegura que una clase posea una sola instancia y proporciona un punto de acceso global a ella (GAMMA 1995). Este patrón se ve evidenciado en la clase *SDRTopComponent* debido a que siempre se hace una sola instancia de esta clase.
- ◊ **Observador (Publicar-Suscribir):** Pertenece al grupo "comportamiento". Permite definir la dependencia de uno o más objetos de forma que cuando un objeto cambie su estado, los restantes sean notificados y se actualicen (GAMMA 1995). Este patrón se evidencia en los componentes del área de diseño que se encuentran en la clase *ListSelectionListener*, debido a que permite realizar la acción de activar algunos componentes si se ha seleccionado alguna fila en la tabla o desactivarlos si no existe selección en la tabla.

2.5 Modelo de despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes de hardware y software en la extensión final. Se componen por nodos, dispositivos y conectores donde los nodos son elementos de procesamiento; los dispositivos son nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela y los conectores expresan el tipo de protocolo utilizado entre el resto de los elementos del modelo (RIVERO 2013). A continuación la Fig. 29 muestra el diagrama de despliegue de la extensión.

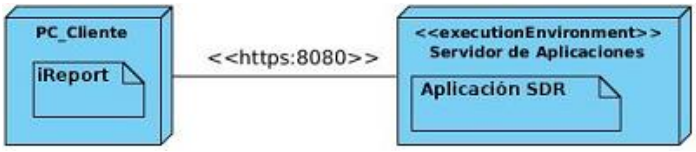


Fig. 29: Diagrama de Despliegue del Caso de uso de la extensión.

El diagrama realizado representa 2 nodos principales. El nodo “PC_Cliente” en el cual deberá estar instalado el diseñador de reportes iReport y el nodo “Servidor de Aplicaciones” que contendrá el SDR. El nodo PC Cliente estará conectado al nodo Servidor de aplicación por el Protocolo Seguro de Transferencia de Hipertexto (HTTPS, del inglés *Hypertext Transfer Protocol Secure*) como protocolo de comunicación. La conexión entre la PC_Cliente y el Servidor de Aplicaciones se realizará mediante el puerto de comunicación 8080.

2.6 Conclusiones parciales del capítulo

El estudio de las principales características de los componentes de Ingeniería de Software, empleando la metodología de OpenUP, permitió obtener el modelo de dominio para comprender y describir la estructura del negocio. La confección de los diagramas de clases del diseño permitió visualizar la relación que existe entre las clases u objetos significativos del problema. A partir del levantamiento de requisitos se obtuvieron 6 requisitos no funcionales y 26 funcionales, estos últimos fueron agrupados en 5 casos de uso, garantizando que el desarrollo de la extensión cumpla con las necesidades de los clientes. El diseño del diagrama de Caso de uso del sistema brindó la posibilidad de mostrar gráficamente los procesos y su interacción con los actores. La utilización de los patrones seleccionados proporcionó un correcto diseño de la extensión y la realización del diagrama de despliegue permitió comprender la distribución física de la extensión..

Capítulo 3: Implementación y prueba de la extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes

En el presente capítulo se abordarán los aspectos esenciales relacionados con la fase de implementación y prueba del software. Se modela la extensión en términos de componentes, especificándose los casos de pruebas realizados a la extensión para validar su correcto funcionamiento.

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 de la extensión. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes, clases del modelo de diseño, subsistemas y componentes físicos.

3.1.1 Diagrama de componentes

Un Diagrama de componentes representa la separación de un sistema de software en componentes físicos y muestra las dependencias entre estos. Cada subsistema corresponde a un paquete físico y cada componente a un módulo, fichero o biblioteca que existe en la memoria de almacenado (BOOCH 2001). Los diagramas de componentes ilustran las piezas del software que conformarán un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, usualmente un componente se implementa por una o más clases u objetos en tiempo de ejecución. A continuación la Fig. 30 muestra el diagrama de componentes correspondiente al caso de uso Gestión de reportes.

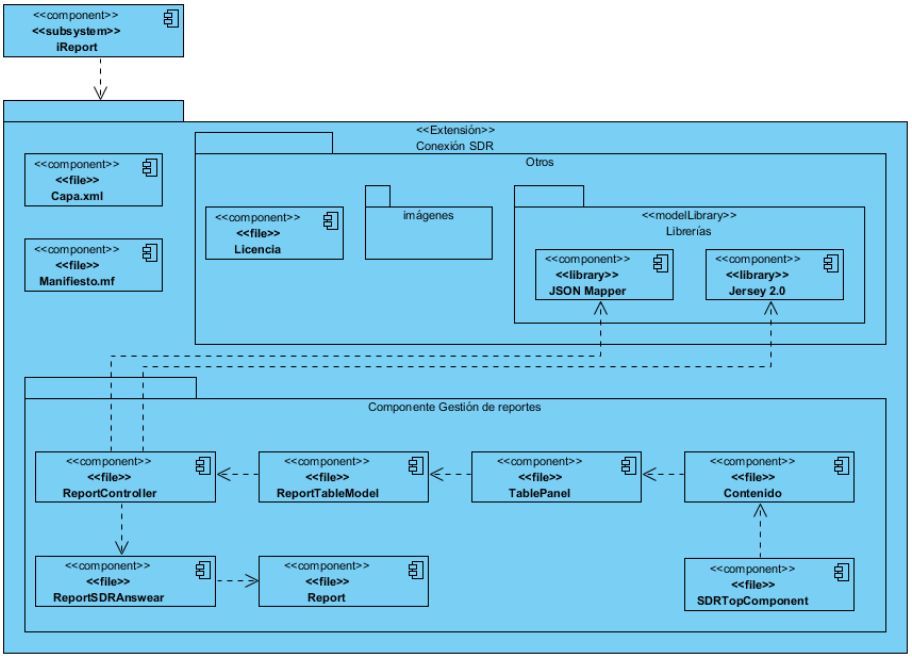


Fig. 30: Diagrama de componentes del Caso de uso “Gestión de reportes”.

3.2 Consumo de servicios web REST en Java

Una petición a un servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP, todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta.

Para consumir el servicio se analizó en un momento inicial qué método se deseaba consultar, REST indica que a la hora de obtener un recurso el método que se debe utilizar es GET, para añadirlo POST, para actualizarlo PUT y para eliminarlo DELETE. Si el servicio web que se va a consumir es por el método GET simplemente se utiliza la clase de Java URL (del inglés *Uniform Resource Locator*) para abrir la conexión mediante el método openConnection por defecto este método crea una conexión utilizando el método GET y devuelve un objeto de la clase de Java HttpURLConnection, esta clase se utiliza para descargar los ficheros de tipo jrxml y las exportaciones. La biblioteca Jersey se utiliza para consumir los servicios de añadir, modificar y eliminar. Para listar y buscar se emplea la clase JCURL la cual permite enviar texto realizando una petición OPTIONS al servidor. Esta petición se realiza mediante la librería Jettison 1.3.3 para conformar un objeto en json utilizando la clase JsonObject. Luego de haber enviado la petición y capturado el texto de respuesta correspondiente, se emplea la librería Jackson-mapper-asl 1.9.11 la cual interpreta el texto y permite crear objetos en Java.

3.3 Pruebas de software

Un software es desarrollado por personas, existiendo la posibilidad que contenga errores. Una buena práctica constituye planificar tareas que estén relacionadas con la garantía de la calidad en todas las fases del flujo de trabajo. En este epígrafe se validará la solución propuesta mediante la realización de pruebas apoyando de esta manera la búsqueda de errores con el objetivo de medir la calidad y rendimiento de la aplicación desarrollada, y que la misma cumpla con los requerimientos de la extensión.

Luego del estudio de varios autores se decidió escoger a Roger Pressman para guiar el desarrollo de este epígrafe por ser una de las figuras de mayor impacto en el tema. Según Pressman *"(…) las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos niveles de trabajo, definiendo 4 niveles: unidad, integración, validación y sistema*" (PRESSMAN 2002). Luego de realizar una estrategia de prueba se decidió aplicar pruebas de unidad, integración, validación y además se incluyeron las pruebas de usabilidad para medir el cumplimiento del objetivo principal de la investigación.

3.3.1 Pruebas de unidad

Las pruebas de unidad se enfocan en un programa o un componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Los programadores siempre prueban el código durante el desarrollo, por lo que las pruebas unitarias son realizadas solamente después de que el programador considera que el componente está libre de errores (MONNÉ ROQUE 2009). Para el desarrollo de este nivel se decidió utilizar los métodos: Caja blanca y Caja negra.

Método de Caja blanca

Se refiere a las pruebas que validan el código del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones. Requiere del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. En los sistemas orientados a objetos las pruebas de caja blanca pueden aplicarse a los métodos de la clase.

Desarrollo de la prueba

Para realizar las pruebas mediante el método de Caja blanca se utilizó la biblioteca JUnit, a continuación se muestran, en la Fig. 31, los resultados obtenidos tras aplicarse 3 iteraciones al método “search” de la clase “ReportController”.

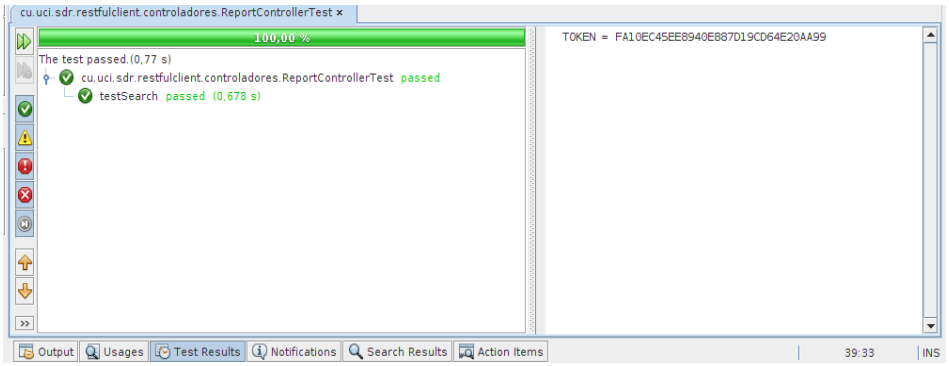


Fig. 31: Tercera iteración del método Caja blanca aplicado mediante la biblioteca JUnit.

Método de Caja negra

Esta prueba se realiza sobre la interfaz del software y para llevarla a cabo es necesario tener conocimiento sobre los escenarios de prueba y secciones que serán probadas. Esta examina algunos aspectos del modelo fundamentalmente de la extensión sin tener mucho en cuenta la estructura interna del software. Permite identificar las posibles fallas en el funcionamiento de la extensión.

Para la aplicación del Método de Caja negra se empleó la técnica Partición equivalente por ser una de las más efectivas, pues permite examinar los valores válidos e inválidos de las entradas que existen en la extensión. Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Desarrollo de la prueba

El diseño de los casos de pruebas se realiza con el objetivo de determinar que una funcionalidad ha sido implementada satisfaciendo las necesidades del cliente. Para cada caso de uso debe estar asociado un caso de prueba que recoja la especificación de ese 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.

En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso Gestionar usuario.

Tabla 4: Tabla de variables para el caso de prueba “Gestión de reportes”

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	**Id**	N/A	No.	Cadena de caracteres que representa el valor único que identifica a cada reporte.
2	**nombre**	N/A	No.	Cadena de caracteres que representa el nombre del reporte.
3	**descripción**	N/A	No.	Breve descripción textual del reporte.
4	**categoría**	N/A	*Si.*	Cadena de caracteres que define una categoría por la cual puedan ser agrupados los reportes.
5	**nombre_jrxml**	N/A	No.	Fichero JRXML que contiene el diseño del reporte.
6	**descarga**	N/A	Si.	Cantidad de veces que ha sido descargado el reporte.
7	**creado**	N/A	*Si.*	Fecha en la que fue creado el reporte.
8	**modificado**	N/A	*No.*	Fecha en la que fue modificado el reporte.
9	**accedido**	N/A	*No.*	Fecha en la que se accedió al reporte.

Matriz de Datos

Escenario Adicionar Reporte

Tabla 5: Escenario “Adicionar reportes”

Escenario	Descripción	1	2	3	4	5	6	7	8	9	Respuesta de la extensión	Flujo Central
EC 1.1 Adicionar Reporte correctamente	Este es el escenario donde todos los datos de las variables son correctos y es exitosa la ejecución de la	V	V	V	V	V	V	V	V	V	El sistema adiciona correctamente el reporte y luego permite la inserción de un nuevo reporte	A través del iReport, se accede a la extensión “Conexión SDR”: 1. Se selecciona la pestaña "Reportes" y luego se presiona clic sobre el botón "Añadir". 2. Se llenan correctamente todos los campos (Nombre,

	prueba.											Descripción, Categoría, Archivo). 3. Se presiona clic sobre el botón "Aceptar" y se adiciona correctamente un reporte.
EC 1.2 Adicionar Reporte_False_JRXML_Incorrecto	Escenario donde se ingresa incorrectamente la dirección física del JRXML.	V	V	V	I	V	V	V	V	V	No se modifican los datos del reporte y el sistema devuelve el siguiente mensaje: "Seleccione el archivo jrxml o deshabilite la opción Editar Archivo".	A través del iReport, se accede a la extensión “Conexión SDR”: 1. Se selecciona la pestaña "Reportes" y luego se presiona clic sobre el botón "Añadir". 2. Se llenan correctamente todos los campos (Nombre, Descripción, Categoría, Archivo). 3. Se presiona clic sobre el botón "Aceptar" y se adiciona correctamente un reporte.
EC 1.3 Adicionar Reporte_False_NombreVacío	Escenario donde el campo Nombre está vacío	V	I	V	V	V	V	V	V	V	No se adiciona el reporte y el sistema devuelve el siguiente mensaje: "El campo Nombre no puede estar vacío"	A través del iReport, se accede a la extensión “Conexión SDR”: 1. Se selecciona la pestaña "Reportes" y luego se presiona clic sobre el botón "Añadir". 2. Se llenan correctamente todos los campos (Nombre, Descripción, Categoría, Archivo). 3. Se presiona clic sobre el botón "Aceptar" y se adiciona correctamente un reporte.

Luego de haber realizado 26 casos de pruebas para comprobar el funcionamiento de la extensión se obtuvieron un total de 12 no conformidades en una primera iteración y 3 en una segunda iteración, las

cuales fueron solucionadas satisfactoriamente en una última iteración. En la siguiente gráfica se muestra un resumen de los resultados obtenidos.

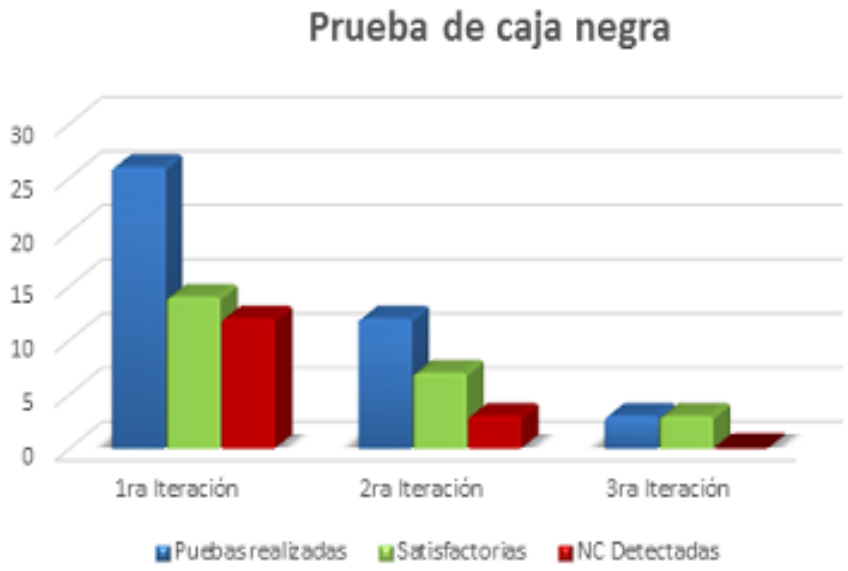


Fig. 32: Gráfico de barras del método Caja negra.

3.3.2 Pruebas de integración

Pressman define: "(…) *la prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (…)*" (PRESSMAN 2002).

Existen dos tipos de pruebas de integración, "incremental" y "no incremental". La integración no incremental combina todos los módulos por anticipado y se prueba todo el programa en conjunto. La integración incremental construye y prueba en pequeños segmentos siendo más fáciles de aislar y de corregir los errores. Existen dos tipos de integración incremental denominadas "ascendente" y "descendente". La ascendente plantea que se integran de abajo hacia arriba comenzando con los módulos de los niveles más bajos de la estructura del programa. La incremental descendente define que se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (PRESSMAN 2002).

Para el desarrollo de este nivel se realizó una prueba de integración incremental ascendente.

Desarrollo de la prueba

Luego de realizar las pruebas de caja negra se integró la extensión al sistema del SDR y se pudo comprobar el correcto funcionamiento con los requisitos definidos. Para realizar el proceso de integración se siguieron los siguientes pasos:

Pasos para realizar la integración con iReport:

- ◊ Dentro de la aplicación iReport, en el menú se selecciona la opción “Herramientas” y se presiona clic sobre la opción “Complementos”.
- ◊ Luego se selecciona la pestaña “Descargado”.
- ◊ Se presiona clic sobre el botón “Agregar complementos”.
- ◊ Se selecciona la extensión y se presiona clic sobre el botón “Abrir”.
- ◊ Luego se presiona clic sobre el botón inferior izquierdo “Instalar”.
- ◊ La aplicación muestra una ventana de Bienvenida, se presiona clic sobre el botón siguiente.
- ◊ Se aceptan todos los términos de la licencia y se presiona clic sobre el botón “Instalar”.
- ◊ La instalación concluye cuando el usuario presiona clic sobre el botón “Finalizar”.

La integración con el SDR ocurre consumiendo los servicios de tipo REST que brinda el servidor. REST intenta emular al protocolo HTTP, o protocolos similares, mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar. Por tanto, este estilo se centra más en interactuar sobre recursos y estado que con mensajes y operaciones. Para llevar a cabo el proceso de integración se crearon clases que consumen los servicios. Como ejemplo si tiene la clase ReportController que consume los servicios crear, editar, eliminar, buscar, descargar, compilar y listar reporte.

3.3.3 Prueba de validación

Tras culminar la prueba de integración se procede a realizar la prueba de validación la cual se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente. Las pruebas de aceptación alfa y beta demuestran la conformidad del cliente con los requisitos (PRESSMAN 2002).

Las pruebas de aceptación se realizan cuando se implementa un software a medida para un cliente, permitiendo que este valide todos los requisitos. La mayoría de los desarrolladores de productos de software llevan a cabo un proceso denominado prueba alfa y beta para detectar errores que sólo el

usuario final puede descubrir. La alfa se lleva a cabo por un cliente, con el desarrollador como observador del usuario, registrando los errores y los problemas de uso en un entorno controlado. La beta se lleva a cabo por los usuarios finales en un entorno de trabajo real. El usuario registra todos los problemas (reales o imaginarios) e informa al desarrollador cada cierto tiempo (PRESSMAN 2002). Para el desarrollo de este nivel se decidió realizar una prueba de aceptación de tipo alfa en el laboratorio del centro DATEC (Ver Anexo 3).

3.3.4 Pruebas de usabilidad

La usabilidad es un criterio de calidad con la que se podría medir el cumplimiento de inteligibilidad siendo este un factor que establece que la información debe presentarse de manera clara y aprovechable por los usuarios. La ISO/IEC 9126 define usabilidad como: *"(…) la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso" (MATOS 2013).*

Existen varios métodos para la valoración de la usabilidad basándose en la examinación de la interfaz de usuario de una aplicación por parte de los expertos (REYES VERA 2011). En aras de garantizar la usabilidad de la extensión fue necesario llevar a cabo la prueba empleando como herramienta una "Lista de chequeo" que realiza la Dirección de calidad del software de la universidad.

La Lista de chequeo es un artefacto que consiste en analizar la interfaz basándose en una serie de principios reconocidos de usabilidad. Un evaluador realiza un diagnóstico de la interfaz, con base en una "Lista de chequeo" creada para tal fin, luego se realiza una sesión de análisis para discutir los elementos encontrados en la interfaz. Finalmente se genera un reporte que contiene todos los aspectos encontrados cualitativamente.

En el desarrollo de la prueba, de los 110 indicadores que conforman la Lista de chequeo se analizaron un total de 70 indicadores que fueron los que se ajustaron a la extensión, obteniéndose 4 no conformidades y 66 indicadores satisfactorios. Las no conformidades encontradas fueron solucionadas posteriormente. En la siguiente gráfica se muestra un resumen de los resultados obtenidos.

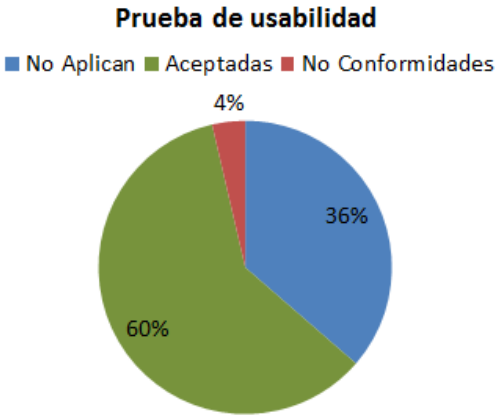


Fig. 33: Gráfico de pastel de la prueba de usabilidad.

3.4 Conclusiones parciales del capítulo

Como resultado del proceso de ingeniería se obtuvo el diagrama de componentes que permitió representar una vista estática de la extensión, mostrando la organización y la dependencia que existe entre los componentes físicos que se necesitan para ejecutar la misma. Las pruebas pertenecientes a los niveles de unidad, integración y validación permitieron comprobar el cumplimiento de los requisitos funcionales de la extensión. Las pruebas de usabilidad garantizaron la familiaridad y comprensión del usuario con la interfaz. Además se demostró que la extensión se adapta al ambiente para el cual fue concebida permitiendo la gestión de reportes en el SDR.

CONCLUSIONES

Con la realización del presente trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar una extensión para gestionar reportes del SDR desde el iReport arribando a las siguientes conclusiones:

- ◊ La investigación realizada sobre los conceptos principales relacionados con el problema de investigación fue el punto de partida para identificar las funcionalidades de la extensión.
- ◊ A partir del diagrama de clases del diseño se estructuraron las clases de acuerdo con los patrones identificados lo que permitió su posterior implementación.
- ◊ Se implementaron los componentes definidos, y se obtuvo la primera versión de la Extensión al iReport para la publicación de reportes en el Servidor Dinámico de Reportes.
- ◊ Se probaron los componentes definidos, y se obtuvo la primera versión de la Extensión al iReport para la publicación de reportes en el Servidor Dinámico de Reportes.
- ◊ La investigación realizada en conjunto con el proceso de desarrollo de software empleado permitió obtener una extensión al iReport que garantiza la usabilidad en el proceso de gestión de reportes del SDR, agilizando este proceso.
- ◊ La extensión al iReport garantiza que SDR como servidor en su versión 1.0 obtenga un valor agregado al incorporarle un diseñador de reportes que se comunique con él permitiendo utilizar al máximo sus funcionalidades, convirtiéndose así en un producto más completo.

RECOMENDACIONES

Se le recomienda al trabajo de diploma titulado “Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes” los siguientes aspectos:

◊ Realizar la prueba de aceptación de tipo beta a la extensión, teniendo en cuenta que aún no ha sido desplegada y este tipo de prueba se realiza en un ambiente real de trabajo.

REFERENCIAS

1. ALVAREZ, C. *Metodología de la Investigación Científica*. Edtion ed. Santiago de Cuba: Centro de estudios de educación superior Manuel F. Gran, 1995.
2. BOERAS, M. Aplicando el método de Boehm y Turner. In*.* La Habana 2012, vol. vol. 5.
3. BOOCH, G. *Análisis de diseño orientado a objetos con aplicaciones*. edited by S.A. ALHAMBRA MEXICANA. Edtion ed., 2001. 672 p. ISBN 9789684443525.
4. DATAPRIX. Dataprix. In., 2014, vol. 2014.
5. ESCALLÓN, A. P., RICARDO. Implementación de una API para la interacción del guante P5 con entornos de realidad virtual implementados en Java y Java 3D. Facultad de Ingeniería Pontificia Universidad Javeriana, 2006.
6. GAMMA, E., HELM, RICHARD, JOHNSON, RALPH, VLISSIDES, JOHN *Design Patterns - Elements of Reusable Object-Oriented Software*. Edtion ed.: Addison-Wesley Professional, 1995. ISBN 978-0201634983.
7. GATES, B. *Los negocios en la era digital*. Edtion ed. Mexico, 1999. ISBN 9788401376337.
8. HERNÁNDEZ, B., MENDOZA, ALBERTO, MONTES, KEIMER, PACHECO YOANDRY, RIVERO, YUNED, NICOT, ASDRUBAL, GUERRERO, JULIET. SOFTWARE LIBRE: MÚLTIPLES APLICACIONES POR UNA META COMÚN. In *V TALLER SOFTWARE LIBRE. PRESENTE Y FUTURO* Villa Clara, 2014.
9. LARMAN, C. Introducción al análisis y diseño Orientado aa Objetos. In P. HALL ed. Mexico, 1999.
10. LARMAN, C. *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Edtion ed., 2002. ISBN 9788420534381.
11. MATOS, R. La Usabilidad Como Factor De Calidad De Páginas Web. Universidad Nacional de La Plata, 2013.
12. MENDOZA GARNACHE, A., NICOT GARCÍA, ABSDRUBAL. Análisis de las Pruebas aplicadas al Servidor Dinámico de Reportes. Curso de Postgrado Universidad de las Ciencias Informáticas, 2014.
13. MONNÉ ROQUE, D., LORES SÁNCHEZ, LINET. Aplicación de las pruebas de liberación al sistema informático de Genética Médica. Universidad de las CIencias Informáticas, 2009.
14. MONTES, K. Servidor Dinámico de Reportes. In *Excellence in Engineering To Enhance a Country's Productivity.* Guayaquil, 2014.
15. NETBEANS. Net Beans. In., 2010.
16. ORIHUELA, S., MONTESLIER, LIANDRY. Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes. 2013.
17. PARADIGM, V. Visual Paradigm. In., 2010.
18. PETRI, J. NetBeans Platform 6.9 Developer's Guide. In *Packt Publishing* Birmingham, 2010.
19. PISANI, F., PIOTET, DOMINIQUE *La alquimia de las multitudes*. Edtion ed. Mexico, 2009. ISBN 978-84-493-2196-2.
20. PRESSMAN, R. *Ingenieria del Software, Un Enfoque Practico*. Edtion ed. Madrid, 2002. ISBN 8448132149.
21. PROGRAMATION, I. Lenguajes de Programacion. In., 2010.
22. REYES VERA, J., LIBREROS GIRALDO, FREDDY ALEJANDRO. MÉTODO PARA LA EVALUACIÓN INTEGRAL DE LA USABILIDAD EN SISTEMAS E-LEARNING. In *Revista Educación en Ingeniería.* Cali, Colombia, 2011.
23. RIVERO, Y., CRUZ, RENÉ LEANDRO. Módulo de Seguridad para la Solución Integral de Gestión de Datos (SIGDAT). Universidad de las Cinecias Informáticas UCI, 2013.

24. SCRIBD. OpenUp. In., 2010.
25. SCRIBD. Herramientas CASE. In., 2014.
26. SILVA, I. A. Generador Automático de Reportes Dinámicos. Centro de Investigación y de estudios avanzados del I.P.N., 2003.
27. SOFTWARE, B. babylon. In., 2014.
28. SOMMERVILLE, I. *Ingeniería del Software*. edited by R. EDUCACIÓN. Edtion ed. Madrid, 2005. 122 p. ISBN 84-7829-074-5.
29. TALES, R. On-Reporting. In., 2014, vol. 2014.
30. TINOCO, O. Criterios de selección de metodologías de desarrollo de software. In., 2014.
31. TUERP, S. Tuerp. In., 2014.
32. UCA. Tecnologías para la Integración de Bases de Datos en el Web. In., 2012.

BIBLIOGRAFÍA

1. ABREU, A. J. Generador dinámico de reportes. In. La Habana, 2012, vol. vol. 5.
2. ALVAREZ, C. *Metodología de la Investigación Científica*. Edtion ed. Santiago de Cuba: Centro de estudios de educación superior Manuel F. Gran, 1995.
3. ARMOA, R. Sistema Avanzado de Distribucion (SAD). In., 2015.
4. BOERAS, M. Aplicando el método de Boehm y Turner. In. La Habana 2012, vol. vol. 5.
5. BOOCH, G. *Análisis de diseño orientado a objetos con aplicaciones*. edited by S.A. ALHAMBRA MEXICANA. Edtion ed., 2001. 672 p. ISBN 9789684443525.
6. DATAPRIX. Dataprix. In., 2014, vol. 2014.
7. GAMMA, E., HELM, RICHARD, JOHNSON, RALPH, VLISSIDES, JOHN *Design Patterns - Elements of Reusable Object-Oriented Software*. Edtion ed.: Addison-Wesley Professional, 1995. ISBN 978-0201634983.
8. GATES, B. *Los negocios en la era digital*. Edtion ed. Mexico, 1999. ISBN 9788401376337.
9. HERNÁNDEZ, B., MENDOZA, ALBERTO, MONTES, KEIMER, PACHECO YOANDRY, RIVERO, YUNED, NICOT, ASDRUBAL, GUERRERO, JULIET. SOFTWARE LIBRE: MÚLTIPLES APLICACIONES POR UNA META COMÚN. In *V TALLER SOFTWARE LIBRE. PRESENTE Y FUTURO* Villa Clara, 2014.
10. ISO, T. ISO/TMBG. In., 2000.
11. LARMAN, C. Introducción al análisis y diseño Orientado aa Objetos. In P. HALL ed. Mexico, 1999.
12. LARMAN, C. *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Edtion ed., 2002. ISBN 9788420534381.
13. LÓPEZ, M. Generador Dinámico de Reporte. In. La Habana, 2011.
14. MATOS, R. La Usabilidad Como Factor De Calidad De Páginas Web. Universidad Nacional de La Plata, 2013.
15. MENDOZA GARNACHE, A. Definición de la arquitectura de software del Grupo de Desarrollo para la Gestión de Equipos Médicos. Universidad de la Ciencias Informáticas UCI, 2009.
16. MENDOZA GARNACHE, A., NICOT GARCÍA, ABSDRUBAL. Análisis de las Pruebas aplicadas al Servidor Dinámico de Reportes. Curso de Postgrado Universidad de las Ciencias Informáticas, 2014.
17. MEZA, C., LÓPEZ, RAFAEL, CEBALLOS, JOSÉ, RODRÍGUEZ, ABELARDO. Una Arquitectura Modular para el desarrollo de un IDE que apoye a la Enseñanza de los Fundamentos de la Programación Orientada a Objetos. In *Revista Internacional de Educación en Ingeniería.* 2010, vol. 3.
18. MICROSOFT, I. Información general de Crystal Report Designer. In., 2008.
19. MONNÉ ROQUE, D., LORES SÁNCHEZ, LINET. Aplicación de las pruebas de liberación al sistema informático de Genética Médica. Universidad de las CIencias Informáticas, 2009.
20. MONTES, K. Servidor Dinámico de Reportes. In *Excellence in Engineering To Enhance a Country's Productivity.* Guayaquil, 2014.
21. NAVARRO, A. N. Asistente de Reportes para el módulo Diseñador de Reportes del Generador Dinámico de Reportes versión 2.0. In. La Habana, 2011.
22. NETBEANS. Net Beans. In., 2010.
23. ORIHUELA, S., MONTESLIER, LIANDRY. Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes. 2013.
24. ÖVERGAARD, G., PALMKVIST, KARIN *Use Cases Patterns and Blueprints*. Edtion ed.: Addison Wesley Professional, 2004. ISBN 0-13-145134-0.

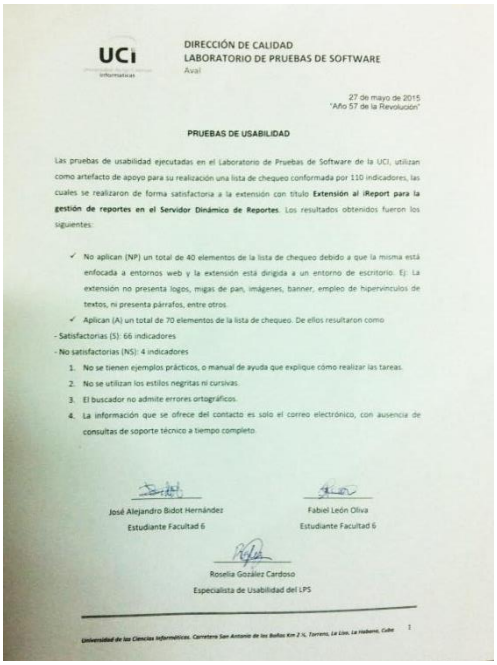
25. PARADIGM, V. Visual Paradigm. In., 2010.
26. PETRI, J. NetBeans Platform 6.9 Developer's Guide. In *Packt Publishing* Birmingham, 2010.
27. PISANI, F., PIOTET, DOMINIQUE *La alquimia de las multitudes*. Edtion ed. Mexico, 2009. ISBN 978-84-493-2196-2.
28. PRESSMAN, R. *Ingenieria del Software, Un Enfoque Practico*. Edtion ed. Madrid, 2002. ISBN 8448132149.
29. PROGRAMATION, I. Lenguajes de Programacion. In., 2010.
30. REYES VERA, J., LIBREROS GIRALDO, FREDDY ALEJANDRO. MÉTODO PARA LA EVALUACIÓN INTEGRAL DE LA USABILIDAD EN SISTEMAS E-LEARNING. In *Revista Educación en Ingeniería.* Cali, Colombia, 2011.
31. RIVERO, Y., CRUZ, RENÉ LEANDRO. Módulo de Seguridad para la Solución Integral de Gestión de Datos (SIGDAT). Universidad de las Cinecias Informáticas UCI, 2013.
32. SAP, S. SAP Crystal Reports. In., 2014.
33. SCRIBD. OpenUp. In., 2010.
34. SCRIBD. Herramientas CASE. In., 2014.
35. SILVA, I. A. Generador Automático de Reportes Dinámicos. Centro de Investigación y de estudios avanzados del I.P.N., 2003.
36. SOFTWARE, B. babylon. In., 2014.
37. SOMMERVILLE, I. *Ingeniería del Software*. edited by R. EDUCACIÓN. Edtion ed. Madrid, 2005. 122 p. ISBN 84-7829-074-5.
38. TALES, R. On-Reporting. In., 2014, vol. 2014.
39. TIBCO, S. iReport Designer. In., 2015.
40. TINOCO, O. Criterios de selección de metodologías de desarrollo de software. In., 2014.
41. TUERP, S. Tuerp. In., 2014.

ANEXOS

Anexo 1. Imagen de bienvenida a la extensión.

Fig. 34: Imagen de bienvenida a la extensión.

Anexo 2. Prueba de usabilidad realizada en los Laboratorios de pruebas de software UCI.



UCI

DIRECCIÓN DE CALIDAD
LABORATORIO DE PRUEBAS DE SOFTWARE
Aval

27 de mayo de 2015
"Año 57 de la Revolución"

PRUEBAS DE USABILIDAD

Las pruebas de usabilidad ejecutadas en el Laboratorio de Pruebas de Software de la UCI, utilizan como artefacto de apoyo para su realización una lista de chequeo conformada por 110 indicadores, las cuales se realizaron de forma satisfactoria a la extensión con título **Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes**. Los resultados obtenidos fueron los siguientes:

- ✓ No aplican (NP) un total de 40 elementos de la lista de chequeo debido a que la misma está enfocada a entornos web y la extensión está dirigida a un entorno de escritorio. Ej: La extensión no presenta logos, migas de pan, imágenes, banner, empleo de hipervínculos de textos, ni presenta párrafos, entre otros.
- ✓ Aplican (A) un total de 70 elementos de la lista de chequeo. De ellos resultaron como

- Satisfactorias (S): 66 indicadores
- No satisfactorias (NS): 4 indicadores

1. No se tienen ejemplos prácticos, o manual de ayuda que explique cómo realizar las tareas.
2. No se utilizan los estilos negritas ni cursivas.
3. El buscador no admite errores ortográficos.
4. La información que se ofrece del contacto es solo el correo electrónico, con ausencia de consultas de soporte técnico a tiempo completo.

José Alejandro Bidot Hernández
Estudiante Facultad 6

Fabiel León Oliva
Estudiante Facultad 6

Roselia González Cardoso
Especialista de Usabilidad del LPS

Fig. 35: Aval de la prueba de usabilidad

Anexo 3. Acta de Aceptación realizada en el laboratorio del centro DATEC.

Centro de Tecnologías de Gestión de Datos
DATEC

La Habana 1 de junio del 2015
"Año 57 de la Revolución"

ACTA DE ACEPTACIÓN

De una parte, el Centro de Tecnologías de Gestión de Datos, en lo sucesivo DATEC, de la Universidad de las Ciencias Informáticas UCI, representado en este acto por
Ing. Alberto Mendoza Garmendia, Ing. Beatriz Hernández Cervantes y de **otra parte** los estudiantes
José Alejandro Bolet Hernández y Fabiel León Oliva
Primero: Que en cumplimiento de las requeridas funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los Hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.
CONSIDERANDO: Que los Hitos aquí se han sido dado cumplir con los requerimientos establecidos.
CONSIDERANDO: Que el producto constituye un aporte relevante para el uso del Servidor Dinámico de Reportes (SDR) a partir de la complementación de la Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes.

POR TANTO: **Las Partes** acuerdan formalizar mediante la presente Acta, la aceptación del producto Extensión al iReport para la gestión de reportes en el Servidor Dinámico de Reportes.

Y para que así conste, se suscribe la presente **Acta** en dos (2) ejemplares, todos por **Las Partes**.

Entregan

José Alejandro Bolet Hernández

Fabiel León Oliva

Reciben

Ing. Alberto Mendoza Garmendia

Ing. Beatriz Hernández Cervantes

Decanato Facultad

Fig. 36: Aval de la prueba de aceptación