Universidad de las Ciencias Informáticas

“Facultad de Ciencias y Tecnologías Computacionales”

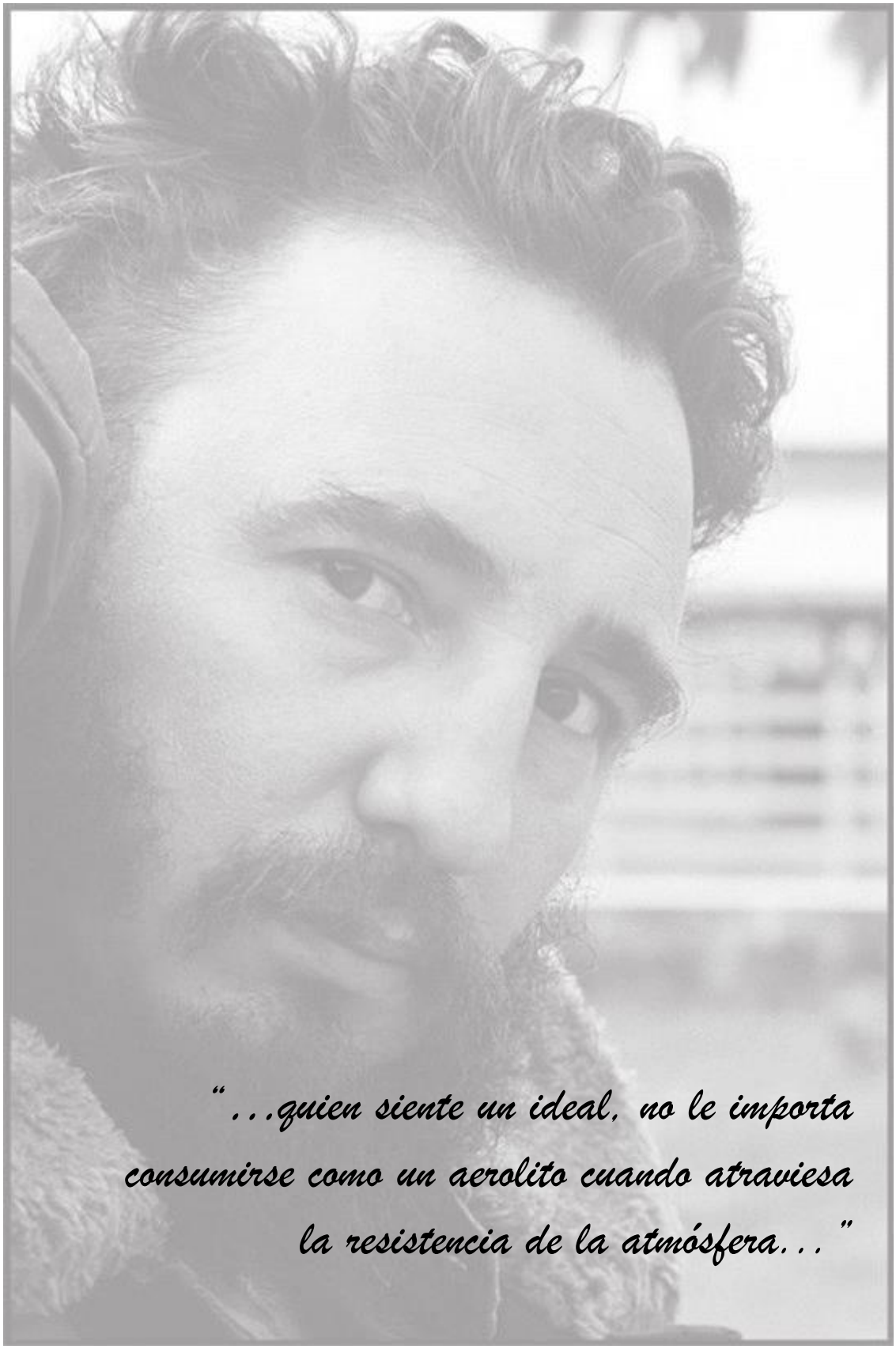
“Módulo Visor de Reportes para el Sistema Integrado de Gestión Estadística v4.0”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Ariam Lázaro Alvarez Rodríguez
René Manuel Puig Pérez

Tutores: Ing. Glennis Tamayo Morales
Ing. Juan Miguel Pérez Almaguer

La Habana, junio de 2017
“Año 59 de la Revolución”



"...quien siente un ideal, no le importa consumirse como un aerolito cuando atraviesa la resistencia de la atmósfera..."

Declaración del autor

Declaramos ser autores de la tesis “Módulo Visor de Reportes para el Sistema Integrado de Gestión Estadística (SIGE v4.0)” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2017.

Ariam Lázaro Alvarez Rodríguez

Firma de Autor

René Manuel Puig Pérez

Firma de Autor

Ing. Glennis Tamayo Morales

Firma de Tutora

Ing. Juan Miguel Pérez Almaguer

Firma de Tutor

Datos del Autor

Tutora: Ing. Glennis Tamayo Morales
Universidad de las Ciencias Informáticas, La Habana, Cuba
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Años de graduada: 7
Años de experiencia en el tema: 7
Correo Electrónico: gtamayo@uci.cu

Tutor: Ing. Juan Miguel Pérez Almaguer
Universidad de las Ciencias Informáticas, La Habana, Cuba
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Años de graduado: 3
Años de experiencia en el tema: 3
Correo Electrónico: jmalmaguer@uci.cu

Autor: Ariam Lázaro Alvarez Rodríguez
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo Electrónico: alalvarez@estudiantes.uci.cu

Autor: René Manuel Puig Pérez
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo Electrónico: rmpuig@estudiantes.uci.cu

Dedicatoria de Ariam

A mi Mamá

Por ser lo más bello que Dios ha puesto en mi camino. Por darme la vida, por criarme y seguir haciéndolo, por quererme mucho, creer en mí, porque siempre me apoyaste, por tus consejos, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor. Mamá gracias por darme una carrera para mi futuro, todo esto te lo debo a ti.

A mi papá

Por darme la vida, quererme mucho, creer en mí, porque siempre me apoyaste, por ser quien con tus consejos has sabido guiarme para culminar mi carrera profesional, por ser además de un padre, mi amigo y mi confidente. Y porque sé que estás orgulloso de la persona en la cual me he convertido.

Dedicatoria de René

A má y pá, por su constante dedicación y paciencia, por confiar en mí y convertirme en lo que soy.

A ustedes va dedicado este trabajo.

Agradecimientos de Ariam

Quiero expresar un profundo agradecimiento a quienes con su ayuda, apoyo y comprensión me alentaron a lograr esta hermosa realidad.

A mis padres por siempre creer en mí y alentarme en todo momento. Muchas gracias por su amor, su preocupación y confianza y por ser la luz que me ha guiado durante todos estos años. Gracias por haberme educado con esos principios sin los cuales hoy no podría ser lo que soy. Los quiero mucho.

A mi esposa Yadira por acompañarme durante todo este arduo camino y compartir conmigo alegrías y fracasos.

A mis hermanos Eliecer e Irelsis por querer que sea el ingeniero de la familia.

A mis tías Maira, Hilda y Nelsys por quererme y ser como mis segundas madres.

A mis amigos Rolando Enrique y Yanet Parra por ayudarme con la tesis y ser mis rivales en el dominó.

A mi amigo Lázaro, mi compañero de estudio de estos cincos años, por ser como mi hermano pequeño.

A mis amigos Julio y Tony los del doble, del gimnasio, de la piscina por siempre estar presente en los momentos necesarios.

A mis amigos Ramiro y Luis Miguel compañeros todos los años de la Copas de programación, debates y prácticas de futbol, del chucho.

A mi amigo William, mi hermano de crianza, la persona del barrio con la cual compartir, jugar fútbol, ir a las fiestas y estudiar juntos.

A mi compañero de tesis Rene por trabajar día y noche en la tesis para esta saliera.

A mis tutores por apoyarnos.

Al tribunal y al oponente por las sugerencias y recomendaciones realizadas en la predefensa de la tesis.

A todas las personas que ayudaron directa e indirectamente en la realización de la tesis y a las que se preocuparon y preguntaron: ¿Y la tesis? o ¿Cuándo te gradúas?

Agradecimientos de René

Quiero dar mi agradecimiento a todos los que de una forma u otra brindaron su ayuda para que este deseo se cumpliera. A mis tutores y a mi oponente por guiarme en este largo proceso, por las críticas constructivas, las recomendaciones y las constantes revisiones. Luego de todos estos años finalmente puedo decir que he logrado uno de mis propósitos.

Que mejor momento para agradecer a esas personitas que conocí hace cuatro años atrás, y que han hecho de mi día a día acá en la escuela una experiencia que nunca olvidaré. Sé que estos serán los últimos días juntos para muchos de nosotros, pues por ley de la vida cada cual ha de tomar su rumbo, es por esto que dedico este momento tan especial para agradecerle a cada uno de ustedes.

A Suinny, Yane, Mid, Clau, Mili, Lizsandra, por toda la ayuda brindada, por ser excelentes compañeras de aula y por los buenos momentos que compartimos juntos acá en la escuela, nunca las voy a olvidar.

A Maykel y Asdrubal, amigos de andanzas acá en la escuela y fuera de ella con los cuales compartí buen tiempo, salidas, fiestas, momentos inolvidables, espero que nuestra amistad se mantenga a pesar de la distancia.

A Randy, tal y como me dijiste el año pasado, que pronto sería ingeniero, ya ves, acá estoy, gracias por los ánimos y por el tiempo que compartimos juntos.

A Ilsen, Lachy, Ernesto y Claudio, por estar ahí siempre que los necesité, han demostrado ser unos amigos excepcionales.

A mis amigos de fuera de la escuela Marlon y Dany, a Yosmel, por estar siempre pendientes y compartir conmigo sus días.

A Julito, Asiel, el Yonky, Lian y David. Gracias a las chicas del laboratorio 103, Yuned y Juliet, a pesar de conocernos desde hace poco dedicaron parte de su tiempo y me brindaron su ayuda.

A mi dúo de tesis, Ariam, por haber puesto su máximo empeño en la realización de esta tesis.

Por último, pero que, a la verdad, para mí son lo más importante en el mundo, a mis padres, gracias por confiar en mí, y animarme en todo momento a pesar de los contratiempos. Por destinar todos los recursos para que pudiera cursar todos estos años de universidad sin complicaciones. Gracias por la paciencia y la dedicación diaria, por entenderme, después de todo son ustedes precisamente la razón de lo que soy hoy y por eso viviré eternamente agradecido.

Resumen

La información es una herramienta de suma importancia para las empresas en la actualidad, pues al analizar los datos resultantes de la misma, la toma de decisiones ocurre de forma sencilla y fluida. El Sistema Integrado de Gestión Estadística (SIGE); tiene como principal objetivo la informatización de los procesos concernientes al diseño de formularios y la captura y validación de datos. Sus procesos fundamentales son: el diseño de plantillas de tipo formulario, la captura de datos sobre estos diseños de plantillas, la validación de los datos capturados y la recuperación de información a través de reportes ofreciendo apoyo en la toma de decisiones en las empresas. Actualmente en la Universidad de las Ciencias Informáticas se encuentra en desarrollo la versión 4.0 del producto, utilizando para la lógica del negocio y el acceso a los datos el framework Symfony v2.8 y para el desarrollo de la interfaz gráfica de usuario ExtJS v6.0. El uso de estas tecnologías ha traído como consecuencia que en estos momentos no es posible la visualización de reportes en el sistema, pues no se integra el nuevo sistema en desarrollo con el Módulo Visor de Reportes que se utiliza en SIGE v3.0; el cual fue desarrollado con el framework Symfony v1.2.7 y ExtJS v2.2. En el presente trabajo se trazó como objetivo desarrollar el Módulo Visor de Reportes para SIGE v4.0 que permita mostrar la información almacenada.

Palabras clave: estadística, reportes, visor.

Abstract

Information is a very important tool in today's companies, because when analyzing the data resulting from it, decision making happens easily. The Integrated Statistical Management System (SIGE); has as main objective the computerization of the processes concerning to the design of forms and the capture and validation of data, helping to make decisions. Its fundamental processes are: the design of form-type templates, the capture of data on these template designs, the validation of the captured data and the retrieval of information through reports offering support in decision making in companies. SIGE v4.0 is currently being developed, using for the business logic and the data access the framework Symfony v2.8 and for the development of the graphical user interface ExtJS v6.0. The use of these technologies has meant that right now isn't possible to view reports in the system, because it can't be integrated with the Report Viewer Module which is used in SIGE v3.0; which was developed using Symfony v1.2.7 and ExtJS v2.2. The main objective in the present work was to develop the Report Viewer Module for SIGE v4.0, allowing to display the stored information.

Keywords: *reports, statics, viewer.*

Índice

Índice de figuras

Índice de tablas

Índice de gráficos

Introducción

En la actualidad la tecnología es considerada una herramienta potente para el desarrollo de cualquier organización, pues el buen uso de la misma brinda gran ayuda a la toma de decisiones con el fin de lograr los objetivos y las metas propuestas por cada organismo. Con el creciente avance de las Tecnologías de la Información y las Comunicaciones (TIC), ha aumentado el volumen de información a manejar en las empresas, convirtiéndose esta en el activo fundamental de las mismas. Los datos necesitan ser gestionados y almacenados en bases de datos para luego ser analizados mediante una aplicación que facilite el trabajo con los mismos. Sin esta aplicación la manipulación de los datos resultaría engorrosa, pues existiría un flujo de información demasiado grande, ocasionando pérdida de tiempo y gasto innecesario de recursos.

La mayoría de las organizaciones utilizan reportes para analizar los datos almacenados y de esta manera los directivos puedan dar seguimiento al negocio, propiciando un correcto funcionamiento de la empresa. Al ser este un proceso que ocurre de forma manual, se ha necesitado informatizar su funcionamiento, pues el aumento de datos trae consigo la ocurrencia de errores humanos que pueden alterar la información.

A raíz de este problema surgen los generadores de reportes, los cuales han demostrado ser una alternativa viable para el control de los datos, como consecuencia, los mismos son considerados una necesidad principal en la Inteligencia de Negocio, pues organizan la información para que esta sea comprendida por el usuario, mostrándola con un diseño atractivo. Son herramientas complementarias a los sistemas de información; utilizan un lenguaje transparente para el cliente por medio del cual este realiza consultas a la base de datos y se obtiene información de ella en forma de reporte. (Silva Hernández, 2010)

Cuba no está ajena al uso de estas tecnologías, las cuales se actualizan constantemente para estar a la par del resto de mundo. Múltiples son las instituciones que en el país se encargan del desarrollo de soluciones informáticas destinadas a mejorar la calidad del trabajo en las empresas, siendo la Universidad de las Ciencias Informáticas (UCI) uno de los mayores centros de producción de *software*. La UCI está estructurada por diversos centros productivos, entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC), ubicado en la Facultad de Ciencias y Tecnologías Computacionales.

DATEC se encarga de la realización del proyecto Sistema Integrado de Gestión Estadística (SIGE); el cual es una aplicación *web* enriquecida desarrollada sobre tecnologías libres, que tiene como principal objetivo la informatización de los procesos de captura de datos de una institución brindando así ayuda a la toma de decisiones. Sus procesos fundamentales son el diseño de plantillas de tipo formulario y encuestas, la captura de datos sobre estos diseños de plantillas, la validación de los datos capturados y la visualización

de estos a través de reportes. El mismo almacena toda la estadística en una sola base de datos y ofrece un apoyo para la toma de decisiones en las empresas mediante los reportes.

SIGE presenta la versión 3.0, ofreciendo una personalización para cada institución en la que es desplegado. El sistema utiliza el Módulo Visor de Reportes del Generador Dinámico de Reportes (GDR v1.8), para la visualización de los reportes del sistema. Esta versión del sistema está desarrollada con el framework Symfony v1.2.7 y como librería JavaScript ExtJS v2.2. Actualmente se está implementando la versión 4.0 del producto, utilizando para la lógica del negocio y el acceso a los datos el framework Symfony v2.8 y para el diseño de interfaz gráfica de usuario ExtJS v6.0; trayendo como consecuencia que en estos momentos no es posible la visualización de reportes en SIGE v4.0, pues los sistemas no se integran debido a la diferencia tecnológica existente.

Debido a la necesidad de dar solución a la situación planteada se identifica como **problema de la investigación**: El Sistema Integrado de Gestión Estadística (SIGE v4.0) no permite visualizar los reportes, lo que trae consigo la imposibilidad de mostrar la información almacenada en el sistema para dar apoyo a la toma de decisiones.

Para dar solución al problema de investigación planteado se define como **objetivo general:** desarrollar el Módulo Visor de Reportes para SIGE v4.0 que permita visualizar la información almacenada en el sistema. Desglosándose en los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación para desarrollar el Módulo Visor de Reportes.
2. Realizar el análisis de las funcionalidades del Módulo Visor de Reportes.
3. Diseñar los componentes del Módulo Visor de Reportes.
4. Implementar las funcionalidades del Módulo Visor de Reportes.
5. Verificar el correcto funcionamiento del Módulo Visor de Reportes.

El **objeto de estudio** está dirigido al proceso de generación de reportes, enmarcado en el **campo de acción**: el proceso de visualización de reportes en SIGE v4.0.

Para darle cumplimiento al objetivo general se definen las siguientes **tareas de la investigación**.

- Selección de las herramientas, tecnologías y metodología para el desarrollo del Módulo Visor de Reportes para SIGE v4.0.
- Análisis de los conceptos técnicos implicados en el desarrollo del Módulo Visor de Reportes.
- Especificación de los requisitos de *software* del Módulo Visor de Reportes para describir las funcionalidades que debe tener el sistema a desarrollar.
- Definición de la arquitectura del Módulo Visor de Reportes para establecer la estructura del sistema.

- Definición de las clases del Módulo Visor de Reportes para establecer la guía de la implementación.
- Implementación de las funcionalidades del Módulo Visor de Reportes para dar respuesta a los requisitos del mismo, haciendo uso de los lenguajes de programación.
- Elaboración de los casos de prueba que serán aplicados al Módulo Visor de Reportes para comprobar su correcto funcionamiento.
- Aplicación de las pruebas de *software* al Módulo Visor de Reportes para identificar posibles errores y corregirlos.

Se pretende dar solución además a las siguientes **preguntas de la investigación**:

- ¿Cuáles son los fundamentos teórico – técnicos en los que se basa el proceso de desarrollo del Módulo Visor de Reportes?
- ¿Cuáles son las características que debe tener el Módulo Visor de Reportes para que permita la integración con SIGE v4.0?
- ¿Qué tecnologías, metodología y herramientas utilizar para el desarrollo del Módulo Visor de Reportes?
- ¿Cómo estructurar el proceso de implementación del Módulo Visor de Reportes que permita una mejor representación de los componentes a desarrollar?
- ¿Cómo validar el correcto funcionamiento del Módulo Visor de Reportes?

Para el cumplimiento del objetivo general planteado se utilizarán los siguientes **métodos de investigación**:

Métodos teóricos:

1. Analítico - Sintético: es utilizado para entender los diversos procesos que se utilizan en la creación del Módulo Visor de Reportes y profundizar en el estudio de cada una de sus partes para luego sintetizarlos en la propuesta de solución.
2. Inducción y deducción: la inducción es un procedimiento que permite a partir de hechos aislados arribar a proposiciones generales. La deducción es un procedimiento que permite a partir del estudio de conocimientos generales inferir casos particulares por un razonamiento lógico. Se realizó un estudio del funcionamiento del visor de reportes de GDR v1.8, extrayendo así las principales funcionalidades que debía tener el Módulo Visor de Reportes para SIGE v4.0.

Métodos empíricos:

1. Entrevista no estructurada: se realiza a especialistas del proyecto SIGE, con el objetivo de obtener información actualizada sobre cómo funciona la visualización de reportes; así como para definir los requisitos funcionales y no funcionales del Módulo Visor de Reportes para SIGE v4.0.

2. Análisis estático: permite observar la estructura de SIGE, con el objetivo de clasificar las dificultades existentes y las funcionalidades que debe tener el sistema a implementar.

El presente trabajo diploma está estructurado de la siguiente manera:

1. Resumen.
2. Introducción.
3. Capítulo 1: Fundamentación teórica para el desarrollo del Módulo Visor de Reportes para SIGE v4.0.
4. Capítulo 2: Análisis y diseño del Módulo Visor de Reportes para SIGE v4.0.
5. Capítulo 3: Implementación y pruebas del Módulo Visor de Reportes para SIGE v4.0.
6. Conclusiones generales.
7. Recomendaciones.
8. Referencias bibliográficas.
9. Bibliografía.
10. Anexos

La temática de los capítulos se describe a continuación:

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA PARA EL DESARROLLO DEL MÓDULO VISOR DE REPORTES PARA SIGE V4.0

En este capítulo se hace un estudio de los diferentes sistemas que poseen generadores de reportes para conocer el estado actual de esos sistemas a nivel mundial. Además, se fundamentan las tecnologías, herramientas y metodología sobre las cuales se asentará el desarrollo posterior del módulo.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL MÓDULO VISOR DE REPORTES PARA SIGE V4.0.

En este capítulo se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema funcione correctamente. Se construyen los artefactos correspondientes al análisis y diseño, modelando la estructura del módulo.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DEL MÓDULO VISOR DE REPORTES PARA SIGE V4.0.

En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes del módulo visor de reportes. Se describen las pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento del sistema.

Capítulo 1: Fundamentación teórica del desarrollo del Módulo Visor de Reportes para SIGE v4.0

En este capítulo se define el marco teórico de la investigación. En el mismo se abordan los conceptos fundamentales relacionados con el problema de investigación; se exponen además las características de los sistemas informáticos generadores de reportes y se realiza el estudio de la metodología, herramientas, tecnologías y lenguajes a utilizar en el desarrollo de la solución.

1.1. Conceptos relacionados con el dominio del problema.

A continuación, se muestran los conceptos fundamentales manejados en el dominio del problema para lograr un mejor entendimiento del mismo.

Reporte: un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital o audiovisual) pretende transmitir una información, aunque puede tener diversos objetivos. Generalmente agrupan información de acuerdo a un interés específico, muestran el resultado de una búsqueda determinada. En la esfera 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 que sea fácil de interpretar por los usuarios (Real Academia Española, 2010).

Generador de Reportes: los generadores de reportes son programas diseñados para crear y administrar informes en una amplia variedad de formatos. Estos proporcionan más control, pueden manejar datos de cálculos y lógica compleja antes de darles la salida. Se caracterizan por dos elementos básicos: un diseñador o editor de informes y un motor de generación. (Ambiente de configuración para el sistema SCADA "Guardián del ALBA", 2013).

Módulo: en programación, un módulo es un *software* que agrupa un conjunto de subprogramas y estructuras de datos. Los módulos son unidades que pueden ser compiladas por separado y los hace reusables y permite que múltiples programadores trabajen en diferentes módulos en forma simultánea, produciendo ahorro en los tiempos de desarrollo. Los módulos promueven la modularidad y el encapsulamiento, pudiendo generar programas complejos de fácil comprensión. (Alegsa, 1998 - 2016)

1.2. Soluciones existentes para la generación y visualización de reportes

A continuación, se realiza una descripción de los sistemas para la generación y visualización de reportes existentes en la actualidad que sustentan la investigación.

Microsoft SQL Server 2005 Reporting Services: es una solución que se utiliza para generar informes empresariales que extraen contenido de una variedad de orígenes de datos relacionales y multidimensionales, que publica informes que se pueden ver en diversos formatos, y que administra la seguridad y las suscripciones de manera centralizada. Los informes creados se pueden visualizar y administrar mediante una aplicación de *Microsoft Windows* o un portal de *SharePoint*[1]*.* Puede crear informes tabulares, matriciales o de formato libre. También puede crear informes adaptados a una circunstancia determinada, que utilicen modelos y orígenes de datos predefinidos. (Microsoft SQL Server 2005 Reporting Services, 2015)

Reporting Services contiene los componentes principales siguientes:

- Un conjunto completo de herramientas que se pueden utilizar para crear, administrar y ver informes.
- Un componente servidor de informes que aloja y procesa informes en diversos formatos. Los formatos de salida incluyen HTML, PDF, TIFF, Excel, CSV, entre otros.
- Una API[2] (*Application Programming Interface*) que permite a los programadores integrar o extender procesamientos de datos e informes en aplicaciones personalizadas o crear herramientas personalizadas para generar y administrar informes.
- Sistema Operativo: Microsoft Windows, preferiblemente en sus versiones para servidores.
- Licencia: Se distribuye bajo licencia privativa perteneciente a Microsoft Corporation, dicha licencia forma parte de la licencia de Microsoft SQL Server 2000 o 2005 (Microsoft Corporation).

No se propone el uso de esta aplicación pues al ser una herramienta propietaria de Microsoft el usuario final debe pagar la licencia establecida para su uso, provocando así la ocurrencia de gastos para la institución. Tampoco se pretende usar sus características en el desarrollo de la solución, pues la bibliografía consultada del servicio de reportes en cuestión, no brinda una exhaustiva información de cada una de sus funcionalidades.

Active Reports: es una herramienta de plataforma propietaria que permite diseñar y crear reportes de forma fácil y rápida. Se caracteriza por dar soporte tanto a aplicaciones *web* como de escritorio siendo capaz de emplear una amplia variedad de orígenes de datos. Además, soporta gráficos, imágenes, sub-reportes y permite la exportación de los reportes a los formatos PDF, Excel, RTF, TIFF, XML y HTML. Active Reports incluye un control para visualizar informes que admite zoom y vista previa de estos, múltiples fichas para visualización de hipervínculos, vistas divididas y de múltiples páginas, un panel de índice de contenidos, miniaturas, búsquedas de texto, anotaciones y personalización de barra de herramientas. Se distribuye bajo licencia privativa perteneciente a *GrapeCity, Inc*[3]*.*

1. Microsoft Office SharePoint Portal Server es una plataforma *web* de trabajo colaborativo y gestión documental, especialmente orientada a documentos Microsoft Office.
2. Un API es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.
3. *Grape City Inc.* es una corporación de software que desarrolla sus propios productos de software y ofrece servicios de desarrollo de productos, servicios de consultoría, software y servicios de gestión de relaciones con clientes

Se estudian de *Active Reports* las principales características como guía para el desarrollo de la aplicación, aunque no se utiliza el *software* pues posee carácter privativo. El mismo opera bajo licencia de *GrapeCity Inc.* y su uso supone un gasto innecesario para la institución.

PHPReports: es un motor de generación de reportes para PHP, es *software* libre y está bajo licencia GPL[4]. A pesar de ser multiplataforma, este *software* funciona mejor con el uso de GNU / Linux como sistema operativo y Apache como servidor *web*. Su funcionamiento se basa en transformaciones XSL (siglas de *Extensible Stylesheet Language*) conocidas como XSLT a partir de la clase *PHPReportsMaker*. Los reportes en *PHPReports* están divididos en capas que contienen unas a otras, entre ellas se encuentran la capa Documento y la capa Página, cada una tiene su propio encabezado y pie de página. Por último, está la capa Grupo en la que se cargarán los campos obtenidos de la consulta a la base de datos. (Rangel, 2005)

SIGE v4.0 utiliza como motor de reportes SDR v2.0; por esto no se decide utilizar *PHPReports* como motor de reportes (a pesar de que es el utilizado en el Sistema Integrado de Gestión Estadística v3.0), pues no existiría compatibilidad entre el visor de reportes en cuestión y SIGE v4.0

Generador Dinámico de Reportes 2 (GDR v2.0): es una aplicación *web* para la gestión de la información de cualquier empresa o institución. Posibilita diseñar reportes tabulares (con gráficos incluidos), tabla pivote y cruzada desde los gestores de Bases de Datos: SQL Server, SQLite, Oracle, MySQL y PostgreSQL. Dicha herramienta permite a los usuarios abstraerse de los conocimientos relacionados con los gestores de bases de datos y generar reportes en varios formatos con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos. (Generador Dinámico de Reportes, 2012)

La versión 2.0 del generador incluye nuevas características y mejoras, donde se destaca la incorporación de nuevas funcionalidades para enriquecer las opciones de diseño y manipulación de reportes. Fue desarrollado bajo la utilización de *software* libre y permite además la exportación de reportes en diferentes formatos (pdf, html, xls, docx, xlsx, pptx, odt, ods, rtf y xml), así como la exportación de informes que contengan dos o más reportes consecutivos. Otra característica que posee la herramienta es la de redefinir vista previa del reporte, mediante la cual se puede aumentar o disminuir el zoom o limitar la salida de este a la cantidad de páginas deseadas. Esta versión del producto utiliza como motor de reportes SDR v2.0.

No se utiliza el visor de reportes existente en este sistema pues está enmarcado en un campo de desarrollo que difiere del utilizado en la creación de SIGE v4.0, y al existir diferencias tecnológicas no sería posible la integración. Se estudiaron las características del visor de reportes de GDR v2.0 y se decide utilizar las mismas para establecer la guía del desarrollo de la solución.

4. GPL es la licencia pública general de GNU. Es ampliamente usada en el mundo del software libre y garantiza al usuario final la libertad de usar, estudiar, compartir y modificar el software

Servidor Dinámico de Reportes (SDR v2.0): es una aplicación informática desarrollada completamente con tecnologías libres, cuyo principio de funcionamiento consiste en brindar una capa de servicios que facilita su integración con cualquier otro sistema, *web* o de escritorio, con independencia de sus plataformas de desarrollo, a través del cual los clientes pueden exportar reportes en una gran cantidad de formatos sobre múltiples fuentes de datos; también provee funciones administrativas y de gestión que garantizan el mantenimiento de los recursos almacenados en el servidor, así como la programación de tareas para la exportación automática de reportes según una frecuencia de tiempo determinada. (Servidor Dinámmico de Reportes, 2016).

SDR v2.0 es el servidor de reportes utilizado en GDR v2.0. Luego de realizado el estudio se decide utilizar esta herramienta en el desarrollo de la aplicación pues también se utiliza en el desarrollo de SIGE v4.0, lo cual garantiza que el visor de reportes que se desea realizar se integre correctamente al sistema de gestión estadística.

Sistema Integrado de Gestión Estadística (SIGE v3.0): este sistema permite definir qué información se obtendrá a través de las encuestas económicas y los formularios estadísticos. Está diseñado como un sistema distribuido con arquitectura modular fomentando la flexibilidad y escalabilidad. De esta forma se garantiza un sistema adaptable a diferentes ámbitos del negocio estadístico.

SIGE responde a los procesos fundamentales identificados en el marco de gestión de la información estadística; tales procesos comprenden la creación de formularios y encuestas, la captura de información, validación de la información captada, su almacenamiento en la base de datos y el traspaso con posterioridad al almacén de datos para su consulta y procesamiento (Pérez, y otros, 2002).

Este sistema posee internamente el visor de reportes de GDR v1.8. El Generador Dinámico de Reportes incluido en la solución es una herramienta desarrollada por DATEC para la realización de los reportes ejecutivos. En el contexto de SIGE es la herramienta para la consulta de la información estadística. El sistema cubre el ciclo de vida de un reporte desde su creación, hasta su consulta y distribución. Entre las principales funcionalidades están: la gestión de los modelos de datos para el reporte (fuente de datos), el diseño visual de consultas, el diseño de reportes parametrizados o no, la visualización de los reportes, la gestión de los reportes y su distribución por correo electrónico, la gestión intensiva de la seguridad. (Robert Lobo, y otros, 2011)

A pesar de realizar el estudio del visor de reportes de SIGE v3.0 no se decide utilizar como solución, pues está desarrollado en un marco de trabajo que difiere al de SIGE v4.0, ocasionando así que los sistemas no

se integren. A pesar de lo mencionado anteriormente, se utilizan sus características para establecer la guía del desarrollo de la solución.

Luego del estudio de las herramientas se decide no utilizar los visores de reporte de *Microsoft SQL Server Reporting Services* ni de *Active Reports* porque son herramientas de carácter propietario y su uso supone el pago de una licencia ocasionando gasto innecesario a la institución. *PHPReports* no será el motor de reportes utilizado para la solución, en su lugar se utilizará SDR v2.0 pues es el utilizado en el desarrollo de SIGE v4.0. Se descarta el uso de los visores de reportes de GDR v2.0 y de SIGE v3.0 pues utilizan una tecnología que no es compatible con la utilizada en el desarrollo de SIGE v4.0, por lo que no garantizarían la integración. Por lo tanto, se propone desarrollar un visor de reportes con las principales características de los visores de reportes de GDR v2.0, SIGE v3.0, SDR v2.0 y *Active Reports*, las cuales se listan a continuación:

Tabla 1: Características de los visores de reportes

Características	Visor de SIGE v3.0	Visor de GDR v2.0	Visor de SDR v2.0	Visor de Active Reports
Visualizar lista de reportes existentes	Si	Si	Si	Si
Actualizar lista de reportes	Si	Si	Si	Si
Buscar reporte registrado en el sistema	Si	Si	Si	Si
Mostrar vista previa del reporte	Si	Si	No	Si
Exportar reporte en diferentes formatos(HTML, PDF, XLS, CSV, DOCX, XLSX, ODS, ODT, RTF, XML y PPT)	Si (Excepto XLS, DOCX, XLSX, ODS, RTF, XML)	Si	Si	Si (Excepto XLS, CSV, DOCX, XLSX, ODS, ODT y PPT)
Limitar la salida de los datos del reporte	Si	Si	No	No
Listar parámetros asociados al reporte	Si	Si	Si	No

Listar Campos de la Fuente de Datos Asociada al Reporte	No	Si	No	No
Modificar Valor del Parámetro del Reporte	Si	Si	Si	No
Expandir Árbol de Categorías	Si	Si	Si	Si
Colapsar Árbol de Categorías	Si	Si	Si	Si
Exportar Informe	Si	Si	No	Si
Exportar Reporte	Si	Si	Si	Si
Definir Vista Previa del Reporte	No	Si	No	Si
Limpiar Parámetros	Si	Si	No	No
Filtrar Avanzado	No	Si	No	No
Adicionar Criterio de Filtrado	Si	Si	No	No
Eliminar Criterio de Filtrado	Si	Si	No	No

Luego de realizada la comparación entre los cuatro visores de reportes se decide escoger las características más predominantes entre ellos, quedando expuestas las principales funcionalidades que debe tener el visor de reportes para SIGE v4.0 que a continuación se presentan:

- Exportar reportes a XLS: Se construirá el reporte en formato XLS.
- Exportar reportes a DOCX: Se construirá el reporte en formato DOCX.
- Exportar reportes a XLSX: Se construirá el reporte en formato XLSX.
- Exportar reportes a ODS: Se construirá el reporte en formato ODS.
- Exportar reportes a RTF: Se construirá el reporte en formato RTF.
- Exportar reportes a XML: Se construirá el reporte en formato XML.
- Listar los parámetros del reporte: Permitirá mostrar los parámetros por los que está compuesto el reporte.
- Mostrar vista previa: Permitirá mostrar la vista previa de un reporte.
- Visualizar lista de reportes: Permitirá visualizar la lista de reportes existentes en el sistema.

- Expandir árbol de categorías: Permitirá expandir el árbol que muestra los reportes existentes en el sistema por categorías.
- Contraer árbol de categorías: Permitirá colapsar el árbol que muestra los reportes existentes por categoría, una vez que este haya sido expandido.
- Definir vista previa del reporte: Permitirá aumentar o disminuir el tamaño de la vista previa del reporte.

1.3. Metodología, tecnologías y herramientas a emplear en la solución

En el presente epígrafe se muestran las herramientas y tecnologías que se utilizarán para el desarrollo del Módulo Visor de Reportes para SIGE v4.0. Se decide utilizar las tecnologías expuestas pues es necesaria la integración con el sistema de gestión existente, implementándose la solución en el mismo marco de trabajo y usando las herramientas destinadas al desarrollo de SIGE v4.0.

1.3.1. Metodologías de Desarrollo de Software

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Se basa en una combinación de los modelos de proceso genéricos, definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto *software*, desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado. (Pressman, 2009)

Una metodología de desarrollo de *software* es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de cualquier sistema. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. (Laboratorio Nacional de Calidad de Software, 2009), entre las metodologías de desarrollo se encuentra *Agile Unified Process* (AUP).

AUP es la metodología de desarrollo de *software* de la cual se utiliza una variación adaptada a los proyectos productivos de la UCI. Esta metodología describe una manera sencilla de entender la forma de desarrollar aplicaciones de *software* de negocio. La metodología consta de tres fases: inicio, ejecución y cierre. (UCI, 2015).

Los artefactos de esta metodología que serán generados posteriormente en el desarrollo de la solución son:

- Modelo de dominio.
- Especificación de requisitos de *software*.
- Especificación de casos de uso.
- Modelo de diseño.

- Diseño de casos de prueba.
- Reglas del negocio.
- Manual de usuario.

1.3.2. Lenguaje de modelado

El Lenguaje Unificado de Modelado o UML (del inglés *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. UML 2.0 proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto los elementos conceptuales (procesos del negocio y funciones del sistema) como los elementos concretos (clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes *software* reutilizables). (uml-diagrams, 2010).

Las funciones de UML 2.0 son las siguientes:

Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.

Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.

Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.

Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.3.3. Herramientas CASE

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas son de gran ayuda en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (Martínez, 2016)

Visual Paradigm v8.0

Visual Paradigm es una herramienta CASE: Ingeniería de *Software* Asistida por Computación. La misma propicia un conjunto de herramientas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (Paradigm, 2010).

Características:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un *software* de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita y comercial.
- Fácil de instalar y actualizar.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Diagramas de flujo de datos.
- Integración con Visio - Dibujo de diagramas UML con plantillas (*stencils*) de Microsoft Visio.
- Editor de figuras.

1.3.4. Lenguaje de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y *software* existentes. (Kaneiwa, y otros, 2015)

PHP

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo *web* y que puede ser incrustado en HTML. Además, es un lenguaje interpretado de alto nivel embebido (introducido) en páginas HTML y ejecutado en el servidor. El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP. Una de las características más potentes de PHP es su soporte para una gran cantidad de bases de datos. (Kaneiwa, y otros, 2015)

Se seleccionó para el desarrollo de la solución la versión 7.0 del lenguaje de programación PHP, por las siguientes características:

- Fácil manejo de errores.
- Mejora de la jerarquía de las excepciones.
- Soporta sistemas Windows de 64-bits.
- Incorpora clases anónimas.

- Existe una alta reducción del uso de la memoria.
- Generador de números aleatorios más seguro.
- Facilita las importaciones desde el mismo *namespace*[5].

Java Script

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas *web* dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Eguíluz Pérez, 2012)

Características de JavaScript:

- Es sencillo (su hermano mayor: el Java, es bastante más complejo).
- Es útil (el desarrollo de Internet, se prevé muy rápido en los próximos años).
- Es potente: permite la moderna POO (programación orientada a objetos).
- Es barato: sólo necesitamos un editor de textos y un navegador.
- Es visual: permite la moderna "programación visual" (ventanas, botones, colores, formularios).

1.3.5. Marcos de trabajo

En el desarrollo de *software*, un marco de trabajo, es una estructura conceptual y tecnológica de soporte definido con módulos de *software* concretos, sirviendo de base para que otro proyecto de *software* pueda ser fácilmente organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y otras herramientas, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Anglada, 2013).

Symfony

Symfony es un marco de trabajo de PHP basado en la arquitectura MVC (Modelo-Vista-Controlador). Fue diseñado para optimizar el desarrollo de aplicaciones *web*, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto.(Potencier, 2009).

Symfony cuenta con su propia consola de instrucciones o tareas, que permite al desarrollador ejecutar un grupo de comandos desde una terminal en la cual el marco de trabajo genera por si solo el código de la orden dada ganando en velocidad a la hora del desarrollo pues evita que el programador pierda tiempo escribiendo el mismo código varias veces. Otra de las funcionalidades más interesantes, es que contiene un submarco de trabajo para realizar formularios. Lo anteriormente mencionado permite crear una clase orientada a objetos que representa el formulario HTML. (Potencier, 2009).

5. En PHP un *namespace* es una utilidad que sirve de contenedor para el código, de modo que, al crearse constantes, funciones o clases, estas queden en un ámbito restringido, evitando colisiones o conflictos de nombres con otros elementos que sean creados.

Symfony v2.8 será la versión utilizada para el desarrollo de la solución, de la cual se muestran las características a continuación:

- Sistema multiplataforma que posee la licencia MIT de software libre.
- Es compatible con los sistemas de gestión MySQL, PostgreSQL, Oracle y SQL Server de Microsoft.
- Proporciona una potente línea de comandos para generar código.
- Orientado al desarrollo ágil de aplicaciones web.

ExtJS v6.0

ExtJS es la elección de los desarrolladores para la construcción de aplicaciones *web* de escritorio utilizando JavaScript y estándares *web* con facilidad. Mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes se pueden mantener fácilmente. ExtJS ofrece una colección enciclopédica de widgets de interfaz de usuario con un elegante tema de partida. (Chen, 2011).

La versión 6.0 cuenta con las siguientes características:

- Fusiona Ext JS para aplicaciones de escritorio y el sencha para aplicaciones móviles.
- Su interfaz está construida a base de ventanas.
- Administración rápida y sencilla de usuarios
- Generación dinámica de roles

1.3.6. Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado IDE (*Integrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Visual Basic, entre otros. (IBM, 2009).

NetBeans v8.0.2

Esta es una herramienta de código abierto diseñada para el desarrollo de aplicaciones, fácilmente portable entre las distintas plataformas, haciendo uso de la tecnología Java. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones *Web*, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles y funcionalidades ampliables mediante la instalación de paquetes. Dentro de las ventajas que brinda el uso de NetBeans como IDE se pueden destacar que el mismo tiene una instalación y actualización simple, un diseñador visual de formularios y presenta características visuales para el desarrollo *web*. Da soporte a las siguientes

tecnologías: Java, PHP, Groovy, C/C++, HTML5, además puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS, entre otros. (genbetadev, 2014)

1.3.7. Gestor de Base de Datos (SGBD)

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: *Database Management System*) es un sistema de *software* que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (OBE, y otros, 2015)

PostgreSQL v9.4

PostgreSQL es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. (OBE, y otros, 2015).

Las principales características de esta versión de PostgreSQL son: (Hispamerica, 2014)

- Utiliza una licencia de tipo BSD[6] "permisiva".
- Trabajadores dinámicos en background.
- Mejoras en la escalabilidad de escritura.
- Mejoras en el desempeño de agregados.
- Disminución del nivel de bloqueo en algunos comandos de ALTER TABLE.

PgAdmin III

Es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados (*EnterpriseDB, PostgresPlus, Advanced Server y GreenplumDatabase)*. Se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris. (Michaud, 2015).

1.3.8. Servidor de aplicaciones

Un servidor de aplicaciones permite mejorar tres aspectos fundamentales en una aplicación: la alta disponibilidad, la escalabilidad y el mantenimiento. Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas, además de preservar la integridad de datos y códigos. (Servidor de Aplicaciones, 2012).

6. La licencia BSD es la licencia de software principalmente otorgada para los sistemas BSD (Berkeley Software Distribution). Es una licencia permisiva que permite el uso del código fuente en software no libre.

Apache *web* server

El proyecto Apache HTTP Server es un esfuerzo de desarrollo de *software* de colaboración destinadas a crear una aplicación robusta, de código fuente de calidad comercial, con muchas características, y libremente disponible en un servidor HTTP (*Web*). El proyecto es administrado conjuntamente por un grupo de voluntarios ubicados en todo el mundo, utilizando la *Web* para comunicarse, planear, desarrollar y documentar dicho proyecto; el mismo forma parte de la Fundación de *Software* Apache. Además, cientos de usuarios han contribuido con ideas, código y documentación del proyecto. (Moro, y otros, 2013)

La versión que será utilizada en la solución del problema de la investigación es Apache v2.4, teniendo en cuenta las siguientes características:

- Es principalmente una liberación de seguridad y corrección de errores.
- Filtro inteligente.
- Almacenamiento en caché mejorado.
- AJP Proxy.
- Equilibrio de carga.
- Soporte de proxy de cierre normal.
- Soporte de archivos grandes.
- Rediseño y autenticación / autorización.

Motor de Reportes

SDR v2.0 será el motor de reportes utilizado en el desarrollo de la aplicación pues es el utilizado en el Sistema Integrado de Gestión Estadística v4.0.

Conclusiones del Capítulo

Durante el capítulo se hizo un estudio de los principales aspectos de las herramientas existentes en el mercado que poseen visores de reportes. Luego del análisis realizado se decide construir una nueva solución tomando como base las principales funcionalidades de GDR v2.0, SIGE v3.0 y SDR v2.0. Como metodología para guiar el proceso de desarrollo se hará uso de AUP adaptada a los entornos de desarrollo de la UCI. Las herramientas que se seleccionaron para desarrollar el *software* fueron: como herramienta para el modelado Visual Paradigm v8.0 empleando el lenguaje de modelado UML v2.0. Durante la implementación de la solución se hará uso del IDE NetBeans 8.0.2, utilizando como lenguajes de programación PHP v7.0 y Java Script, soportados por los marcos de trabajos Symfony v2.8 y ExtJS v6.0 respectivamente. Además, como gestor de base de datos se usará PostgreSQL v9.4; para la administración de la misma PgAdmin III y como servidor *web* Apache 2.4.

Capítulo 2: Análisis y diseño del Módulo Visor de Reportes para SIGE v4.0

El diseño de un producto es una etapa de gran importancia para el desarrollo de una solución informática; en este se desarrollan, revisan y documentan los requisitos del producto y los detalles de implementación. En el presente capítulo se describe la propuesta de solución al problema de investigación planteado respetando las disciplinas de la metodología AUP desarrollada por la UCI. Se realiza el modelo de dominio, en el cual se describe la estructura actual del sistema existente mediante los principales conceptos de su entorno, para luego realizar la fase de ejecución del Módulo Visor de Reportes para SIGE v4.0. Posteriormente se realiza el levantamiento de requisitos funcionales y no funcionales, cumpliendo así con uno de los objetivos generales de la investigación. Se muestran además los diagramas de clases del diseño para representar la estructura del sistema y los patrones de arquitectura y diseño utilizados en el sistema.

2.1. Modelo de Dominio

El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema ya que puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. El modelo de dominio es utilizado como un medio para comprender el sector de negocios al cual el sistema va a servir (Larman, 2008).

El presente modelo de dominio tiene como objetivo contribuir a la comprensión del Módulo Visor de Reportes de SIGE v3.0.

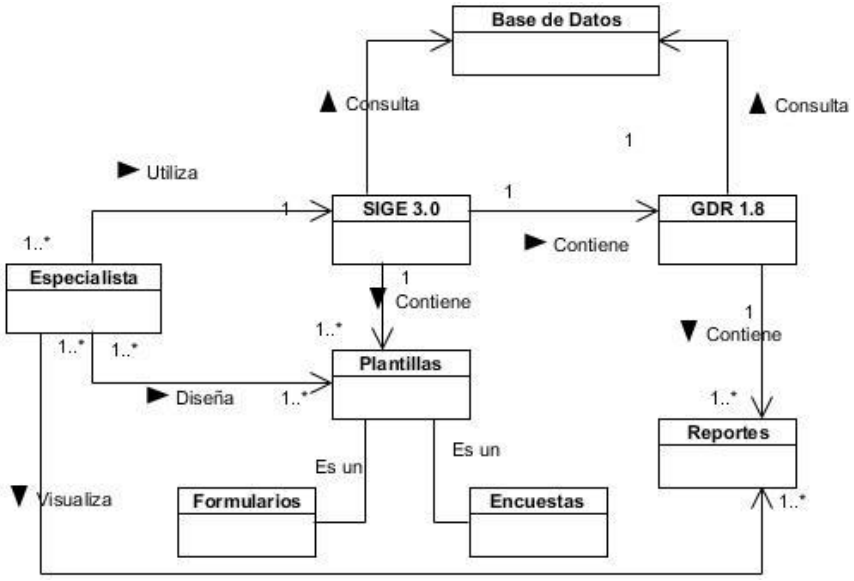


Figura 1: Modelo de Dominio

Un especialista utiliza SIGE v3.0 para diseñar plantillas de tipo formulario y de tipo encuesta. El especialista, además, visualiza los reportes que tiene contenido GDR v1.8, este último sistema se encuentra embebido dentro de SIGE v3.0 y junto a este, consulta la base de datos que contiene los datos que son utilizados por ambos sistemas.

Definición de las clases del modelo de dominio

Especialista: usuario encargado de visualizar los reportes.
GDR 1.8: Generador Dinámico de Reportes, encargado de mostrar la información que se encuentra en la base de datos y organizarla en forma de reportes.
SIGE 3.0: Sistema Integrado de Gestión Estadística, dentro del cual se encuentra el Generador Dinámico de Reportes GDR que posee el Módulo Visor de Reportes.
Base de Datos: abstracción en la que se encuentran los datos disponibles para el uso del visor de SIGE v3.0 y GDR v1.8.
Reportes: plantilla en la cual se muestran los datos obtenidos de la base de datos luego de la previa obtención de los mismos desde la base de datos.
Plantillas: definen el subconjunto de datos a digitar de acuerdo a la naturaleza económica o social de la unidad de observación en cuestión.
Formularios: presentes en el Módulo diseñador de Formularios son los encargados de recopilar la información estadística en las unidades de observación.
Encuestas: presentes en el módulo de entrada de datos de SIGE recopilan información en las unidades de observación. Son otra forma de obtener la información con la que trabaja el sistema.

2.2. Especificación de los requisitos del sistema

Los requisitos para un sistema de *software* determinan lo que hará el sistema y definen las restricciones de su operación e implementación. Se pueden clasificar en: funcionales y no funcionales. (Pressman, 2009). Partiendo de la descripción de clases representadas en el Modelo de Dominio y las necesidades del cliente, fueron determinados los requisitos funcionales y no funcionales del sistema

2.2.1. Requisitos Funcionales

Los requisitos funcionales definen el comportamiento interno de un *software*, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de *software*. (Pressman, 2009).

A continuación, se muestra el listado de requisitos funcionales que debe cumplir el sistema:

RF-01 Exportar reporte en HTML.

RF-02 Exportar reporte en PDF.

RF-03 Exportar reporte en XLS.

RF-04 Exportar reporte en CSV.

RF-05 Exportar reporte en DOCX.

RF-06 Exportar reporte en XLSX.

RF-07 Exportar reporte en ODS.

RF-08 Exportar reporte en ODT.

RF-09 Exportar reporte en RTF.

RF-10 Exportar reporte en XML.

RF-11 Exportar reporte en PPT.

RF-12 Exportar Informe en HTML.

RF-13 Exportar Informe en PDF.

RF-14 Exportar Informe en XLS.

RF-15 Exportar Informe en CSV.

RF-16 Exportar Informe en DOCX.

RF-17 Exportar Informe en XLSX.

RF-18 Exportar Informe en ODS.

RF-19 Exportar Informe en ODT.

RF-20 Exportar Informe en RTF.

RF-21 Exportar Informe en XML.

RF-22 Exportar Informe en PPT.

RF-23 Listar parámetros asociados al reporte.

RF-24 Modificar parámetros del reporte.

RF-25 Limpiar parámetros.

RF-26 Visualizar lista de reportes existentes.

RF-27 Expandir árbol de categorías.

RF-28 Contraer árbol de categorías.

RF-29 Redefinir vista previa del reporte.

RF-30 Mostrar vista previa del reporte.

2.2.2. Requisitos No Funcionales

Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Permiten apreciar cuáles son las características que debe tener el entorno de despliegue para el correcto funcionamiento de la solución informática propuesta. (Pressman, 2009).

Requisitos de *Software*:

RNF-1 Servidor que aloja la aplicación:
Sistema Operativo (SO): GNU/Linux Ubuntu 16.04 o superior, Debían 7 GNU/Linux o superior.
Paquetes: apache2, php7, libapache2-mod-php7, php7-cli, php7-pgsql, php7-xsl, php7-gd, php7-curl.

RNF-2 Servidor que aloja la vista de la aplicación:
Sistema Operativo (SO): GNU/Linux Ubuntu 16.04 o superior, Debían 7 GNU/Linux o superior.

RNF-3 Servidor base de datos:
Sistema Operativo (SO): GNU/Linux Ubuntu 16.04 o superior, Debían 7 GNU/Linux o superior.
PostgreSQL en su versión 9.4.

RNF-4 PC Cliente:
Para el acceso a la aplicación desde la PC cliente se deben tener en cuenta los siguientes requisitos de *software*:
Navegador *Web*: Firefox v45.0 como mínimo.

Requisitos de Hardware:

RNF-5 Servidor que aloja la aplicación:
Ordenador Dual Core o superior, con 2.0 GHz de velocidad de microprocesador.
Memoria RAM mínimo 2GB
Disco Duro con 30GB de capacidad para instalar el sistema.

RNF-6 Servidor que aloja la vista de la aplicación:
Ordenador Dual Core o superior, con 2.0 GHz de velocidad de microprocesador.
Memoria RAM mínimo 2GB

Disco Duro con 10GB de capacidad para instalar el sistema.

RNF-7 Servidor de base de datos:
Ordenador Dual Core o superior, con 2.0 GHz de velocidad de microprocesador.
Memoria RAM mínimo 2GB
Disco Duro con 50GB de capacidad para instalar el sistema.

RNF-8 Las PC cliente deben cumplir con los siguientes requisitos de hardware:
Ordenador Pentium IV o superior, con 2.0 GHz de velocidad de microprocesador.
Memoria RAM, mínimo 1GB.

Restricciones de Diseño e Implementación:

RNF-9 Lenguaje y marco de trabajo para el desarrollo del lado del servidor.
El módulo será implementado en el lenguaje de programación PHP v7.0, utilizando como framework de desarrollo Symfony v2.8, estableciendo la arquitectura Modelo-Vista-Controlador.
Se utilizará la arquitectura SOA para la comunicación del controlador la vista.

RNF-10 Lenguaje y marco de trabajo para el desarrollo del sistema del lado del cliente.
Como librería JavaScript se utilizará ExtJS 6.

Interfaz:

RNF-11 Las interfaces serán diseñadas respetando las marcas sombrillas de la UCI. Para la aplicación se utilizará XEDRO, marca que agrupa a las aplicaciones desarrolladas para el ámbito empresarial e industrial.

2.3. Modelo de Casos de Uso del Sistema

El modelo de casos de uso describe las funcionalidades propuestas del nuevo sistema. Un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema, es una unidad simple de trabajo significativo compuesto por actores, casos de uso y la relación que existe entre ellos (Sparx, 2000-2007).

Caso de uso: un caso de uso es una técnica de modelado usada para describir lo que debería hacer un sistema nuevo o lo que hace un sistema que ya existe. Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario, permiten definir los límites del sistema y las relaciones entre el sistema y el entorno (Jacobson, y otros, 2004).

Actores del sistema:

A continuación, se muestran los actores del sistema:

Tabla 2: Descripción de los actores del sistema

Actor	Descripción
Especialista	Actor que tiene la responsabilidad de iniciar las funcionalidades relacionadas con la visualización de reportes. Tiene los privilegios para administrar los reportes y sus parámetros, además de privilegios para exportar un reporte.
SDR	Actor que representa el Servidor Dinámico de Reportes (SDR v2.0).

2.3.1. Diagrama de Casos de Uso del Sistema

Los diagramas de caso de uso del sistema muestran el funcionamiento de un sistema desde el punto de vista del actor. Estos son la representación de los requisitos funcionales del sistema agrupados por casos de uso.

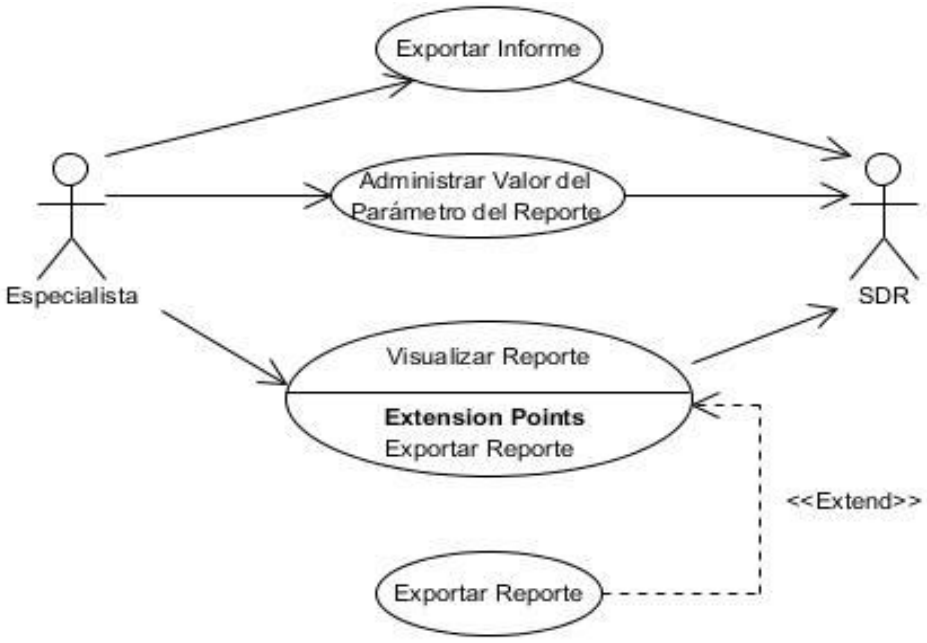


Figura 2: Diagrama de Casos de Uso del Sistema

El actor Especialista inicia los casos de uso “Administrar Valor del Parámetro del Reporte”, “Visualizar Reportes” y “Exportar Informe”. Estos casos de uso se relacionan con el actor SDR, el cual envía al sistema

la información para su posterior visualización. Además, el especialista podrá exportar un reporte previamente visualizado.

Patrones de casos de uso utilizados

Un patrón de casos de uso no describe un uso particular de un sistema. Más bien, captura las técnicas para que el modelo sea reusable y entendible. Están diseñados para ofrecer mejores prácticas para modelar casos de uso. (Pressman, 2009). A continuación, se listan los patrones de casos de uso utilizados en el diseño del diagrama de casos de uso del sistema:

Extensión concreta: Consiste en dos casos de uso y una relación extendida entre ellos. Puede ser instalado en sí mismo, así como extendido en el caso de uso base. El referente puede ser concreto o abstracto. Este patrón se aplica cuando un flujo puede extender de otro caso de uso. En este caso, el caso de uso “Exportar Reporte” extiende del caso de uso “Visualizar Reporte”, cumpliendo con el patrón.

Múltiples actores – Rol Diferente: Se evidencia cuando a 2 o más casos de uso ingresan más de dos actores y existe una relación entre esos casos de uso y su correspondiente actor. En la figura 2 los actores Especialista y SDR tienen relación con los casos de uso “Administrar Valor del Parámetro del Reporte”, “Exportar Informe” y “Visualizar Reporte”.

A continuación, se muestra la tabla que relaciona los requisitos funcionales con casos de uso:

Tabla 3: Relación entre requisito funcional y caso de uso

Requisito Funcional	Caso de Uso
RF-01 Exportar reporte en HTML	Exportar Reporte
RF-02 Exportar reporte en PDF	
RF-03 Exportar reporte en XLS	
RF-04 Exportar reporte en CSV	
RF-05 Exportar reporte en DOCX	
RF-06 Exportar reporte en XLSX	
RF-07 Exportar reporte en ODS	
RF-08 Exportar reporte en ODT	

RF-09 Exportar reporte en RTF	
RF-10 Exportar reporte en XML	
RF-11 Exportar reporte en PPT	
RF-12 Exportar Informe en HTML	Exportar Informe
RF-13 Exportar Informe en PDF	
RF-14 Exportar Informe en XLS	
RF-15 Exportar Informe en CSV	
RF-16 Exportar Informe en DOCX	
RF-17 Exportar Informe en XLSX	
RF-18 Exportar Informe en ODS	
RF-19 Exportar Informe en ODT	
RF-20 Exportar Informe en RTF	
RF-21 Exportar Informe en XML	
RF-22 Exportar Informe en PPT	
RF-23 Listar parámetros asociados al reporte	Administrar Valor de Parámetros del Reporte
RF-24 Modificar parámetros del reporte	
RF-25 Limpiar parámetros	
RF-26 Visualizar lista de reportes existentes	Visualizar Reporte
RF-27 Expandir árbol de categorías	
RF-28 Colapsar árbol de categorías	
RF-29 Redefinir vista previa del reporte	
RF-30 Mostrar vista previa del reporte	

Descripción textual del caso de uso "Administrar Valor del Parámetro del Reporte"

A continuación, se muestra la descripción textual del caso de uso Administrar Valor del Parámetro del Reporte (Ver: Tabla 4).

Tabla 4: Descripción del caso de uso Administrar Valor del Parámetro del Reporte

Objetivo	Administrar los parámetros asociados a un reporte determinado	
Actores	Especialista (Inicia).	
Resumen	El caso de uso inicia cuando el usuario accede al área destinada a los valores de los parámetros asociados a un reporte y modifica o añade los valores asociados al mismo. Finaliza el caso de uso cuando se ha modificado o añadido un valor satisfactoriamente.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	El especialista ha seleccionado un reporte.	
Postcondiciones	Se ha administrado un parámetro.	
Flujo de eventos		
Flujo básico: "Modificar Valor del Parámetro del Reporte"		
	Actor	**Sistema**
1.	El actor especialista selecciona un reporte de la lista.	
2.		Obtiene los datos del actor SDR y los envía a la vista.
3.		Muestra los parámetros del reporte seleccionado.
4.	El especialista selecciona un parámetro de la lista y da doble clic en él para habilitar la edición.	
5.		Habilita el campo para modificarlo.

6.	El especialista introduce el nuevo valor	
7.	El especialista presiona botón “Actualizar”.	
8.		Finaliza el caso de uso
Flujo Alterno: Limpiar Parámetro		
6.1.	Selecciona el botón “Limpiar”	
6.2.		Elimina los valores asociados a los parámetros del reporte seleccionado.

Relaciones	**CU incluidos**	-
	CU extendidos	-
Requisitos No Funcionales	**RNF-11** Las interfaces serán diseñadas respetando las marcas sombrillas de la UCI. Para la aplicación se utilizará XEDRO, marca que agrupa a las aplicaciones desarrolladas para el ámbito empresarial e industrial.	
Asuntos pendientes	-	

Prototipo elemental de interfaz gráfica de usuario

2.4. Diseño del sistema

El diseño del sistema permite determinar cómo funcionará el producto final de forma general sin entrar en detalles incorporando consideraciones de la implementación tecnológica. Consiste en el diseño de los

componentes del sistema que dan respuesta a las funcionalidades y requisitos descritos en la etapa de especificación. Generalmente se realiza con diagramas que permitan describir las interacciones entre las entidades y su secuenciado, impone una estructura del sistema. (Bruegge, 2008).

Se define además como propósito del diseño (Larman, 2008):

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

2.4.1. Arquitecturas utilizadas para la solución

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de *software* en ingeniería de *software*. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de *software*, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor. (Pressman, 2009)

Patrón arquitectónico: se utilizará el patrón Modelo-Vista-Controlador debido a que el módulo se desarrollará utilizando el framework de desarrollo Symfony; el mismo está basado en un patrón clásico de diseño *web* conocido como arquitectura MVC (Potencier, 2009), el cual está formada por tres niveles:

1. **El Modelo** representa la información con la que trabaja la aplicación. Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tantas consultas como especificaciones de la aplicación. Este envía a la “vista” aquella parte de la información que en cada momento el usuario le solicita para su posterior presentación.
2. **La Vista** transforma el modelo en una página *web*, permitiendo al usuario interactuar con la aplicación.
3. **El Controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o la vista. Responde a eventos e invoca peticiones al “modelo” cuando se hace alguna solicitud sobre la información. Puede enviar comandos a la “vista” correspondiente si se solicita un cambio en la forma en que se presenta el modelo. Es un intermediario entre la “vista” y el “modelo”.

La arquitectura MVC separa la lógica de negocio y la presentación logrando conseguir un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. Se basa en las ideas de reutilización de código y la separación de conceptos.

Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (SOA, siglas del inglés Service Oriented Architecture) es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos. Las soluciones SOA han sido creadas para satisfacer los objetivos de negocio las cuales incluyen facilidad y flexibilidad de integración con sistemas legados, alineación directa a los procesos de negocio reduciendo costos de implementación, innovación de servicios a clientes y una adaptación ágil ante cambios incluyendo reacción temprana ante la competitividad. Permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios lo cual facilita la interacción entre diferentes sistemas propios o de terceros. Entre los servicios que utiliza esta arquitectura se encuentran: XML, HTTP, SOAP, REST, WSDL y UDDI. (Service - Oriented Architecture Compass, 2009)

En el desarrollo de la aplicación se utiliza la arquitectura orientada a servicios para realizar la conexión del servidor con la vista, para la misma se utiliza el servicio REST (*Representational State Transfer*) el cual cumple con las siguientes características: (Navarro Marset, 2007)

- Escalabilidad de la interacción con los componentes
- Generalidad de interfaces
- Puesta en funcionamiento independiente
- Compatibilidad con componentes intermedios

2.4.2. Modelo de diseño

El modelo de diseño es un refinamiento y formalización adicional del modelo del análisis, donde se toman en cuenta las consecuencias del ambiente de implementación. Describe la realización de los casos de uso y al mismo tiempo constituye una abstracción del modelo de implementación y del código fuente. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos incluyendo sus operaciones y atributos. (Bruegge, 2008).

2.4.3. Diagrama de clases de diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de *software* y las interfaces de una aplicación. Es una representación concreta de lo que se debe implementar y forma parte de la estética del sistema, representándolo mediante clases y sus relaciones (Booch, y otros, 2009).

A continuación, se muestra el diagrama de clases de diseño del caso de uso Administrar Valor del Parámetro del Reporte (Ver: Figura 3).

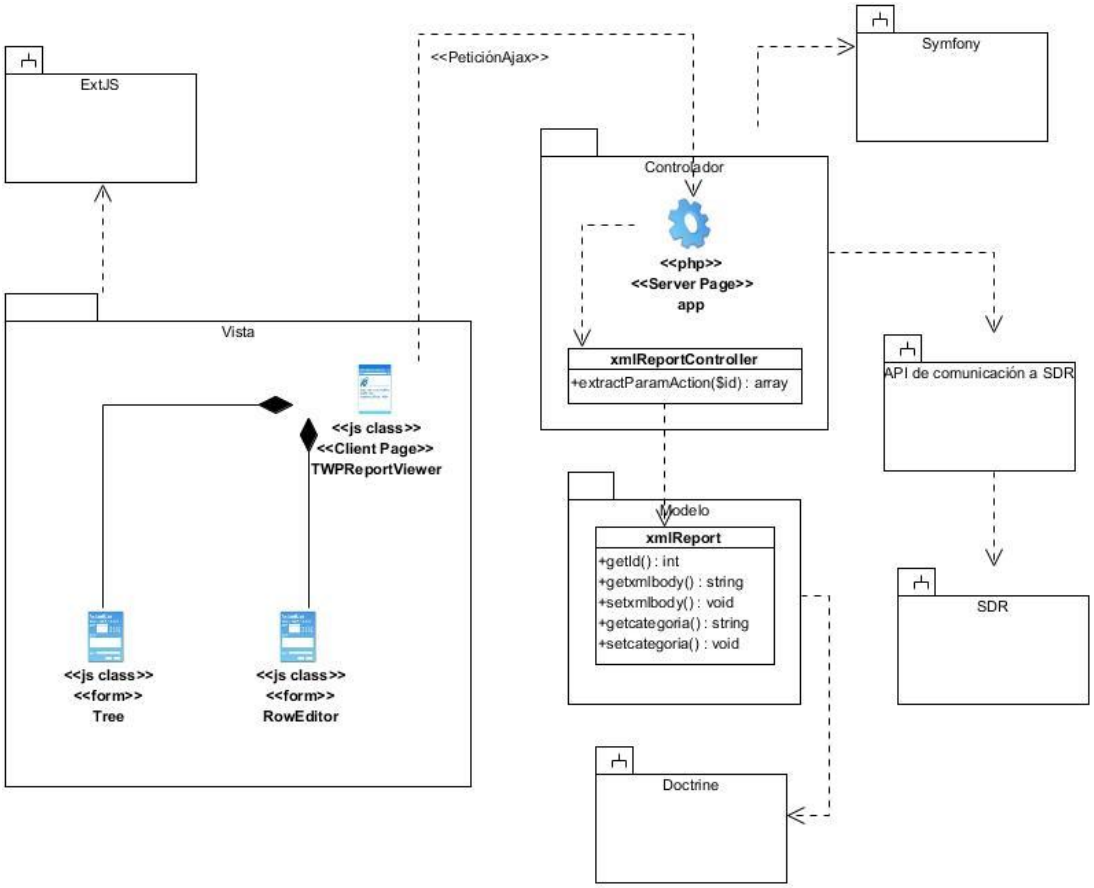


Figura 3: Diagrama de clases de diseño del caso de uso Administrar Valor del Parámetro del Reporte

El diagrama de clases de diseño representa todas las clases con los métodos y atributos correspondientes al caso de uso Administrar Valor del Parámetro del Reporte. En el diseño del mismo se utilizó la arquitectura Modelo-Vista-Controlador. En el controlador se encuentra la clase manejadora de los eventos de las vistas, además de la clase *xmlReportController*, estas están implementadas en lenguaje PHP y dependen del subsistema Symfony. En la vista se encuentran todos los elementos observables por el cliente, los cuales responden a las necesidades del sistema; estos son implementados en el lenguaje JavaScript, utilizando

los componentes del subsistema ExtJS. Los elementos del modelo corresponden a la clase generada por el ORM Doctrine. Se describen las clases del caso de uso Administrar Valor del Parámetro del Reporte.

Paquete Vista:

- TWPReportViewer: clase que representa el panel principal de la aplicación.
- Tree: clase que representa el área destinada para visualizar la lista de reportes existentes agrupados en el árbol de categorías.
- RowEditor: clase que representa el área destinada a la modificación de los parámetros del reporte.

Paquete Modelo:

- xmlReport: clase php que hace referencia a la tabla de los reportes de la base de datos.

Paquete Controlador:

- app: clase encargada de manejar los eventos provenientes de la vista.
- xmlReportController: clase encargada de recibir las peticiones provenientes del controlador frontal y de enviar las mismas a la clase index.html de la vista.

Subsistemas representados:

- ExtJS: atiende las peticiones del lado del cliente.
- Symfony: atiende las peticiones del lado del servidor.
- Doctrine: maneja los datos y realiza la comunicación con los distintos gestores de base de datos.
- SDR: servidor Dinámico de Reportes, nutre al visor de reportes para poder visualizar un reporte determinado.
- API de comunicación con el SDR: subsistema intermedio entre el controlador y SDR, utilizado para la comunicación entre la clase y el subsistema.

Patrones GRASP

GRASP (General Responsability Asignment *Software* Patterns) o patrones generales de *software* para asignar responsabilidades describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones. (Gamma, y otros, 2009).

A continuación, se muestran los patrones GRASP utilizados en el diseño del diagrama de clases del diseño Administrar Valor del Parámetro del Reporte (Figura: 3).

Experto: asignar una responsabilidad al experto en información o a la clase que tiene la información requerida. Se utiliza este patrón pues Symfony utiliza el ORM Doctrine, permitiendo encapsular el acceso a

los datos y son generadas las clases con todas las funcionalidades comunes de las entidades. En la Figura 4 se muestran las clases *xmlReport* y *xmlParam* con las características anteriormente mencionadas.

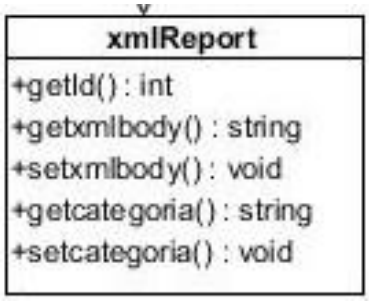


Figura 4: Patrón Experto

Alta Cohesión: define lo relacionadas que están las responsabilidades de una clase, o una clase con responsabilidades altamente relacionadas y que no lleva a cabo gran cantidad de trabajo. La clase *app.php* solamente se encarga de manejar los eventos provenientes de la vista y de devolver las respuestas a estas peticiones. Ver Figura 5.

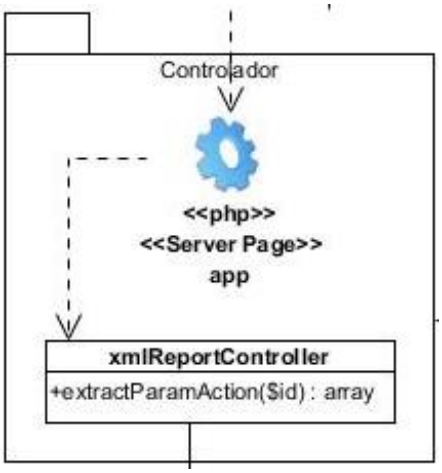


Figura 5: Patrón Alta Cohesión

Bajo Acoplamiento: es la medida de cuanto una clase está conectada con otras clases. El bajo acoplamiento permite que el diseño sea más independiente. Las clases que implementan el acceso a los datos se encuentran en el modelo, proporcionando que la independencia en este caso sea baja. En este caso en la Figura 6 puede observarse que cada una de las clases poseen el mínimo de dependencias, estableciéndose el mínimo de relaciones entre las clases de la vista.

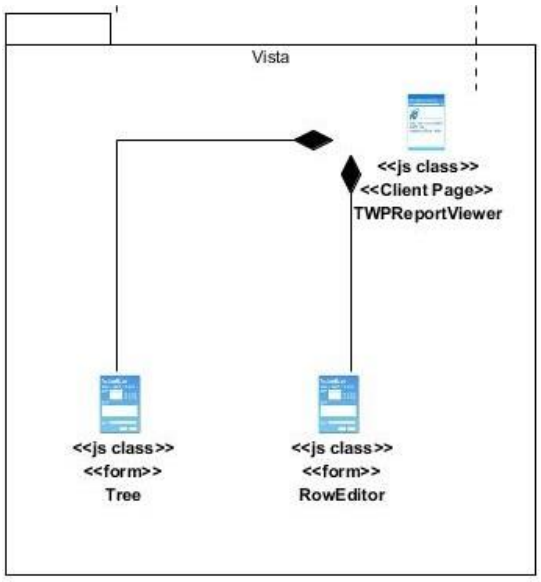


Figura 6: Patrón Bajo Acoplamiento

Controlador: este objeto no pertenece a la interfaz de usuario, es el encargado de controlar el flujo de acciones y peticiones realizadas en la *web*. Este define el método para la operación del sistema, Todas las peticiones son manipuladas por un solo controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado. En la Figura 7 la clase "app.php" escucha cada una de las peticiones realizadas, manejando luego la acción requerida para satisfacer la petición.

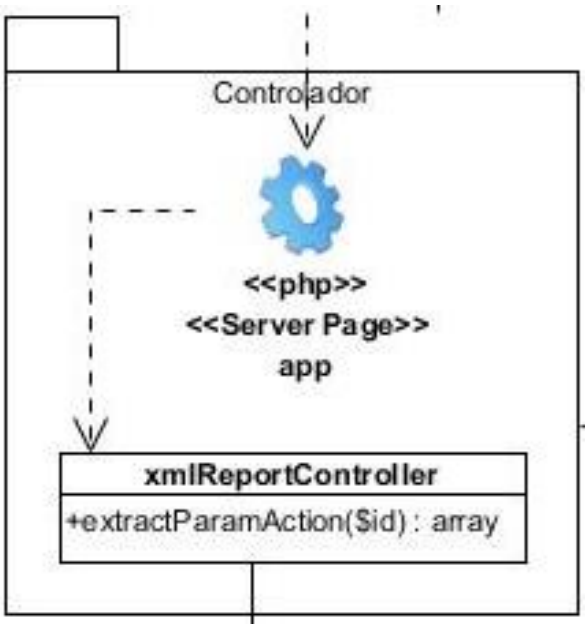


Figura 7: Patrón Controlador

Patrones GoF

Los patrones GoF (*Gang of Feur*) se basan en la experiencia acumulada al resolver problemas reiterativos. Ayudan a construir un *software* basados en la reutilización. (Jacobson, y otros, 2004).

Fachada (Estructural): se requiere una interfaz para el manejo de las funciones e implementaciones existentes. En el caso del diagrama, se define la clase "index.html" presentando una única interfaz responsable de colaborar con los clientes, evidenciándose el patrón Controlador de Fachada. (Figura 8).

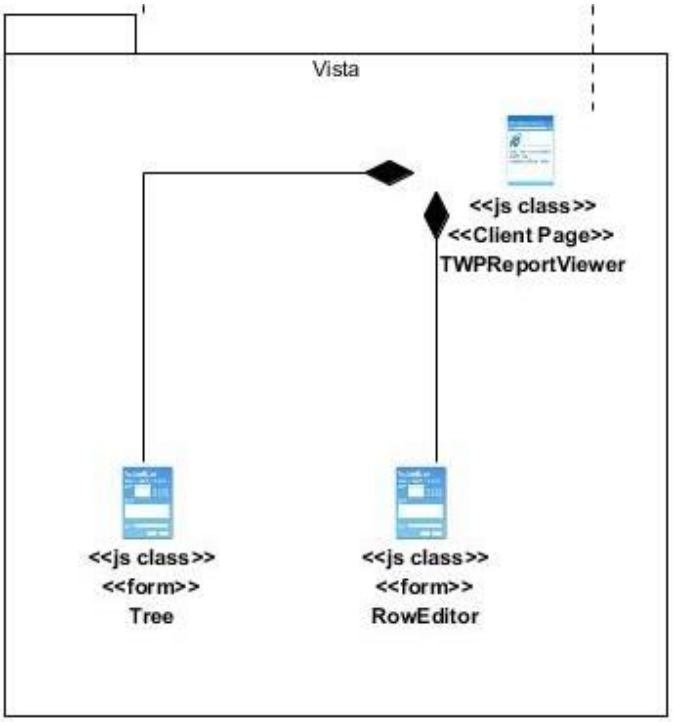


Figura 8: Patrón Controlador de Fachada

2.4.4. Diagrama de secuencia

Un diagrama de secuencia destaca el orden temporal de los mensajes. un diagrama de secuencia se forma colocando en primer lugar los objetos que participan en la interacción en la parte superior del diagrama, a lo largo del eje X. Normalmente, se coloca a la izquierda el objeto que inicia la interacción y los objetos subordinados a la derecha. A continuación, se colocan los mensajes que estos objetos envían y reciben a lo largo del eje Y, en orden de sucesión en el tiempo, desde arriba hasta abajo. Esto ofrece al lector una señal visual clara del flujo de control a lo largo del tiempo. (Booch, y otros, 2009).

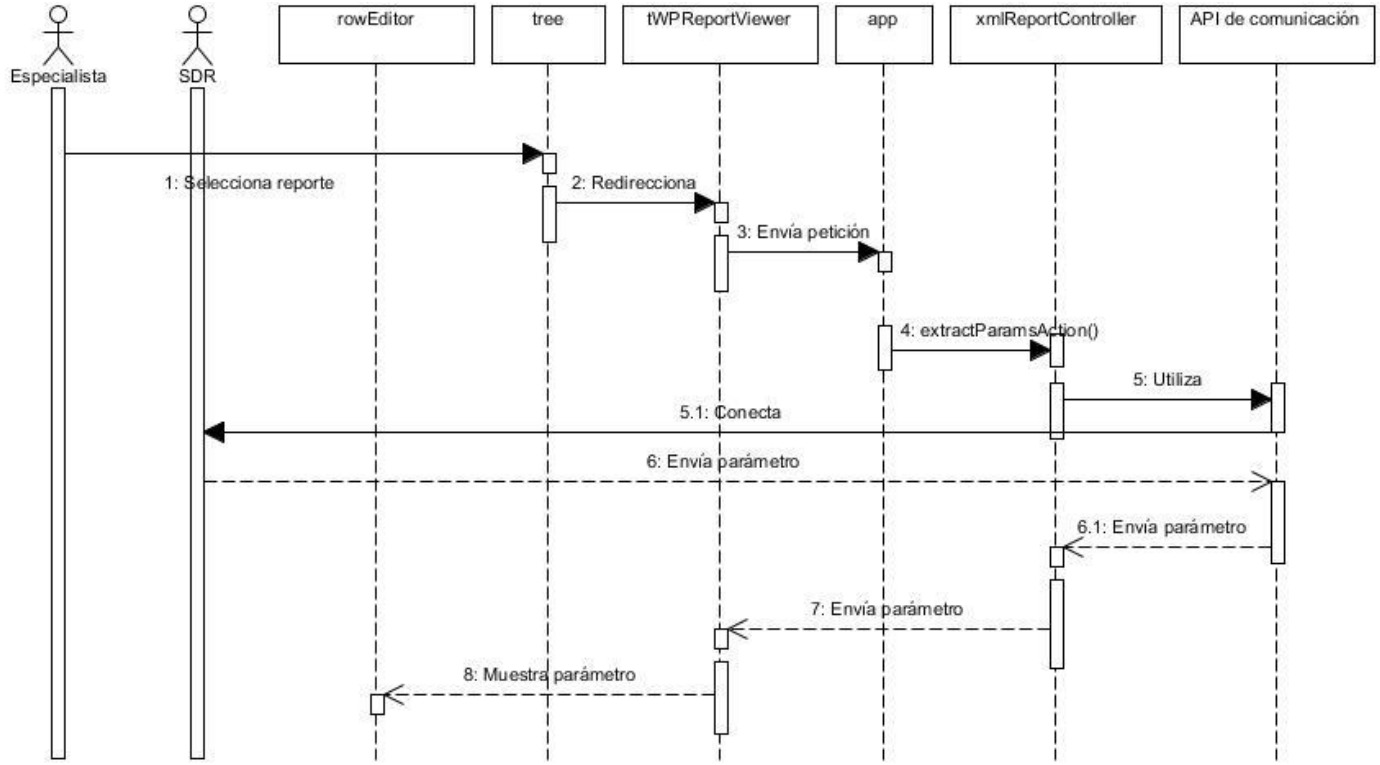


Figura 9: Diagrama de secuencia correspondiente al escenario listar parámetros del reporte del CU Administrar Valor del Parámetro del Reporte

El diagrama de la Figura 9 el actor Especialista selecciona un reporte en el árbol de categorías, luego la clase *Tree* redirecciona a la clase *twpReportController*; esta clase realiza una petición Ajax al controlador frontal *app.php*. Se hace la llamada al método *extractParamsAction()* de la clase *xmlReportController*. Esta clase utiliza el API de comunicación para conectarse con el SDR y así extraer los parámetros del reporte deseado. Luego se envían los parámetros a través de la clase *xmlReportController*, hasta llegar a la clase *twpReportController* de la vista, la cual finalmente muestra el parámetro en la clase *rowEditor*.

2.4.5. Diagrama entidad-relación

El diagrama entidad-relación o DER es una notación gráfica utilizada para el modelado de datos de alto nivel que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. Las interrelaciones entre entidades pueden ser n-arias aunque usualmente se trabaja con relaciones binarias. Por cada entidad pueden existir en un momento dado cero, una o muchas instancias, las mismas toman valores para sus atributos de los dominios de datos definidos para ellos. Las instancias de una relación son pares ordenados de instancias de las entidades que dicha relación vincula. (Pressman, 2009).

A continuación, se muestra el diagrama entidad relación que representa la tabla de la base de datos que contiene la información de los reportes:

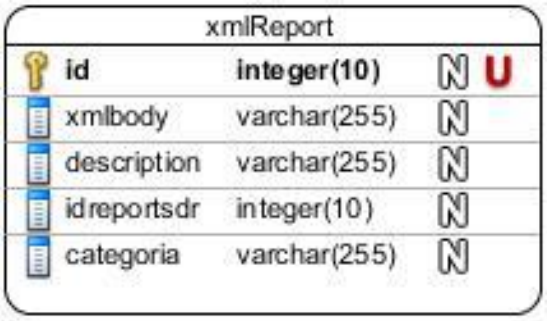


Figura 10: Diagrama Entidad – Relación xmlReport

2.4.6. Modelo de despliegue

Los diagramas de despliegue muestran la configuración física de un sistema, revelando qué piezas de *software* se ejecutan sobre las piezas de hardware. Muestran el hardware de un sistema, el *software* instalado en él y el medio utilizado para conectar unos nodos con otros. Generalmente es utilizado cuando se tienen aplicaciones en diferentes máquinas. (Pressman, 2009).

El siguiente diagrama de despliegue muestra cómo estará desplegado el sistema en su entorno real:

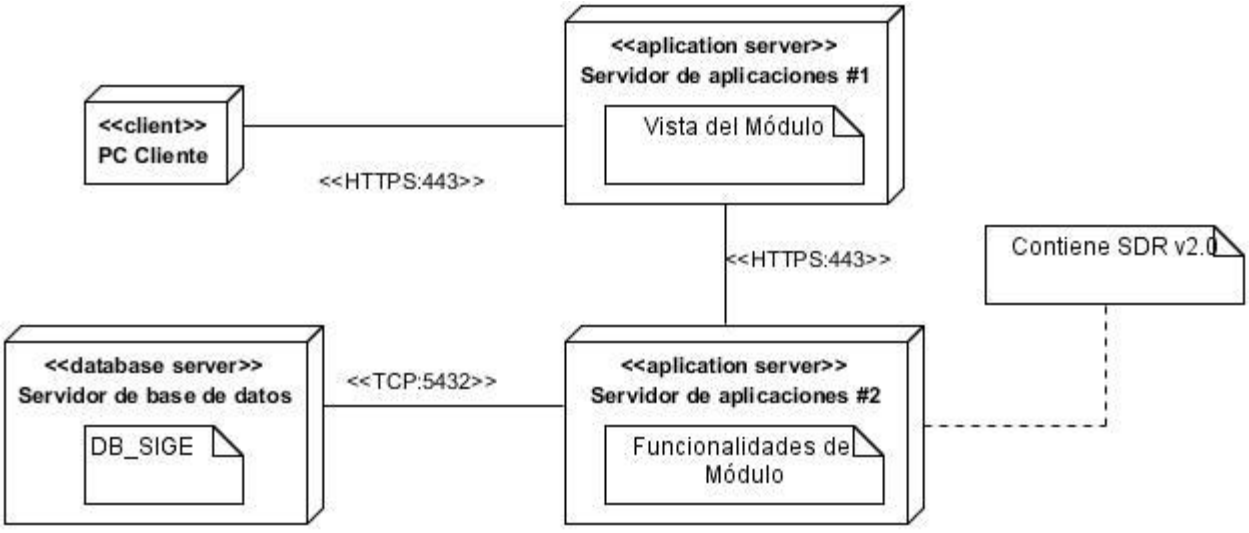


Figura 11: Diagrama de Despliegue

PC Cliente: Estaciones de trabajo que utilizará el usuario para acceder al sistema.

Servidor de Aplicación #1: Servidor en el que se encuentra hospedada la vista de la aplicación, la PC Cliente se conecta a este mediante el protocolo HTTPS (*Hypertext Transfer Protocol Secure*).

Servidor de Aplicación #2: Servidor en el que se encuentran hospedadas las funcionalidades de la aplicación, el servidor de aplicaciones #1 se conecta a este mediante el protocolo HTTPS (*Hypertext Transfer Protocol Secure*). Este servidor #2 se conecta al servidor de base de datos mediante el protocolo TCP/IP. En este nodo estará alojado el Servidor Dinámico de Reportes (SDR v2.0).

Servidor de Base de Datos: Servidor en el que se encuentran todos los datos que utiliza la aplicación para su funcionamiento. En el estarán recopilados todos los datos de los nodos regionales. Se utiliza PostgreSQL como servidor de base de datos, disponible en Linux y Windows.

Protocolo HTTPS: Protocolo de comunicación bidireccional utilizado entre el Servidor de Aplicaciones y la PC Cliente.

Protocolo TCP: Protocolo de comunicación utilizado para establecer la conexión entre el Servidor de Aplicaciones y el Servidor de Base de Datos.

Conclusiones del Capítulo

En el desarrollo del presente capítulo se realizó una representación visual de las clases conceptuales del entorno real de los objetos del proyecto a través del modelo de dominio y se describieron los 30 requisitos funcionales y los 11 no funcionales que el sistema debe cumplir para su correcto funcionamiento. Además, se identificaron los actores del sistema, 4 casos de uso y la relación existente entre ellos reflejada en el diagrama de casos de uso del sistema el cual fue diseñado utilizando los patrones “Extensión Concreta” y “Múltiples Actores – Roles Diferentes”. La realización del modelo de diseño permitió conocer cómo debe ser implementado el sistema a nivel de clases y sus relaciones a través de los diagramas de clases, haciendo uso de los patrones GRASP: Experto, Controlador, Alta Cohesión y Bajo Acoplamiento, además, del patrón GoF: Fachada Estructural. Se propuso el diagrama de despliegue quedando conformado por cuatro nodos y sus relaciones permitiendo mostrar la distribución física del sistema.

Capítulo 3: Implementación y pruebas del Módulo Visor de Reportes para SIGE v4.0

El presente capítulo está dedicado a los procesos que se llevan a cabo durante la actividad de implementación y pruebas. Se realiza el modelo de implementación y se diseñan y aplican las pruebas para comprobar el correcto funcionamiento del módulo.

3.1. Modelo de Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Un modelo de implementación muestra las dependencias entre las partes del código del sistema.(Pressman, 2009).

3.1.1. Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El principal uso de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando así los subsistemas de implementación y sus dependencias a la hora de importar el código y cómo organizar los subsistemas de implementación en capas.

Los diagramas de componentes se utilizan además para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultado de la compilación, dependencias de elementos de implementación y los correspondientes elementos de diseño que son implementados. (Booch, y otros, 2009).

A continuación, se muestra el diagrama de componentes del caso de uso Administrar Valor del Parámetro del Reporte.

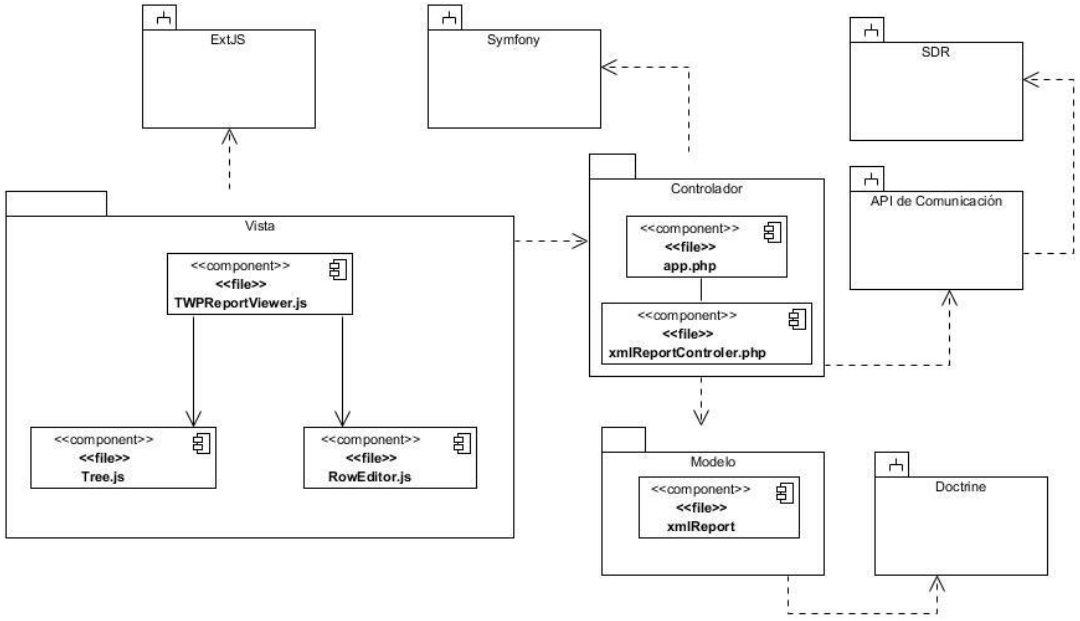


Figura 12: Diagrama de Componentes del CU Administrar Valor del Parámetro del Reporte

El diagrama de componentes (Figura 12) es utilizado para mostrar la relación entre los elementos y clases que componen el caso de uso. Para el desarrollo del módulo se utilizó el patrón utilizado en Symfony, Modelo – Vista – Controlador. Los elementos visibles para el cliente están agrupados en el paquete Vista y están representados por tres componentes, los cuales son los encargados de controlar el flujo de las interfaces, para su funcionamiento utilizan el subsistema ExtJS; Estas clases generan el código pertinente para realizar las llamadas a la clase app.php, perteneciente al paquete Controlador; en este paquete cada una de las acciones para dar respuesta al servidor están implementadas en lenguaje PHP y dependen del subsistema Symfony. El componente del paquete Modelo corresponde a la clase generada por el ORM Doctrine. La clase que implementa la lógica del negocio y el acceso a los datos se encuentra en el modelo.

3.2. Código fuente

El código fuente se define como el conjunto de líneas de texto que son directrices que debe seguir la computadora para realizar dicho programa. El mismo está escrito en un lenguaje de programación determinado, el cual no puede ser ejecutado directamente por la computadora, sino que debe ser traducido a lenguaje de máquina utilizando los compiladores, ensambladores e intérpretes, a este proceso de traducción se le denomina compilación. (Booch, y otros, 2009).

3.2.1. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo

práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de *software*, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de *software* incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de *software* creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente. (Bruegge, 2008).

Estándares de codificación utilizados

- Los bloques de código siempre deben estar encerrados por llaves, si consta de una línea no es necesario utilizar llaves.
- Los nombres de clases pueden contener sólo caracteres alfanuméricos.
- Los números están permitidos en los nombres de clase.
- Los nombres de funciones y variables deben empezar siempre con una letra minúscula utilizando la forma “camelCase”.
- Se recomienda en sentido general la elocuencia, los nombres de las funciones deben describir su propósito y comportamiento.
- Los ficheros JavaScript tienen que concluir con la extensión .js y no deben incluir signos de puntuación excepto – (guión medio) o _ (guión bajo).
- Los ficheros de Symfony tienen que concluir con la extensión .php y no deben incluir signos de puntuación excepto – (guión medio) o _ (guión bajo).
- Los métodos de los objetos deben ser nombrados utilizando la forma ‘camelCase ‘.
- Usar paréntesis en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los operadores. Incluso sí parece claro el orden de precedencia de los operadores, podría no ser así para otros, no se debe asumir que otros programadores conozcan el orden de precedencia.
- Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.

A continuación, se muestra un ejemplo del estándar de codificación utilizado en el desarrollo de la aplicación:

```php
public function extractParamsAction(Request $request) {
    try {
        $id = $request->get('idReport');
        $em = $this->getDoctrine()->getManager();
        $report = $em->getRepository("ReportViewerBundle:xmlReport")->find($id);
        $idSdr = $report->getIdreportsdr();
        $sc = new SdrComunicationApi\AuthenticityServices();
        $response = json_decode($sc->login());
        if ($response)
            $token = $response->items[0]->token;
        $paramsReport = array(
            'id' => $idSdr,
            'token' => $token
        );
        $sdrReport = new SdrComunicationApi\ReportServices();
        $data = $sdrReport->showparam($paramsReport);
        $aux = json_decode($data);
        $parametros = array();
        foreach ($aux->items as $nodo) {
            $reportObject['param'] = $nodo->name;
            $reportObject['type'] = $nodo->type;
            $reportObject['valor'] = $nodo->value;
            $parametros[] = $reportObject;
        }
        $response = new Response("{'success': true, "
                . "'data':" . json_encode($parametros) . "}", HttpCode::HTTP_OK);
        return $response;
    } catch (\Exception $e) {
        return new Response(json_encode(array("success" => false,
                    "message" => $e->getMessage(), HttpCode::HTTP_SERVER_ERROR)));
    }
}
```

Figura 13: Código Fuente de la clase extractParamsAction destinada a la exportación de reportes.

3.3. Interfaces de la aplicación

A continuación, se muestran las interfaces con las que interactuará el usuario para el manejo de la aplicación. En la figura 14 se muestra la interfaz principal, cumpliendo con los requisitos funcionales del sistema muestra las diferentes funcionalidades, entre las que se encuentran la exportación de reportes en diferentes formatos, la administración de los parámetros del reporte, el listado de reportes y la funcionalidad dedicada a redefinir la vista previa del reporte.

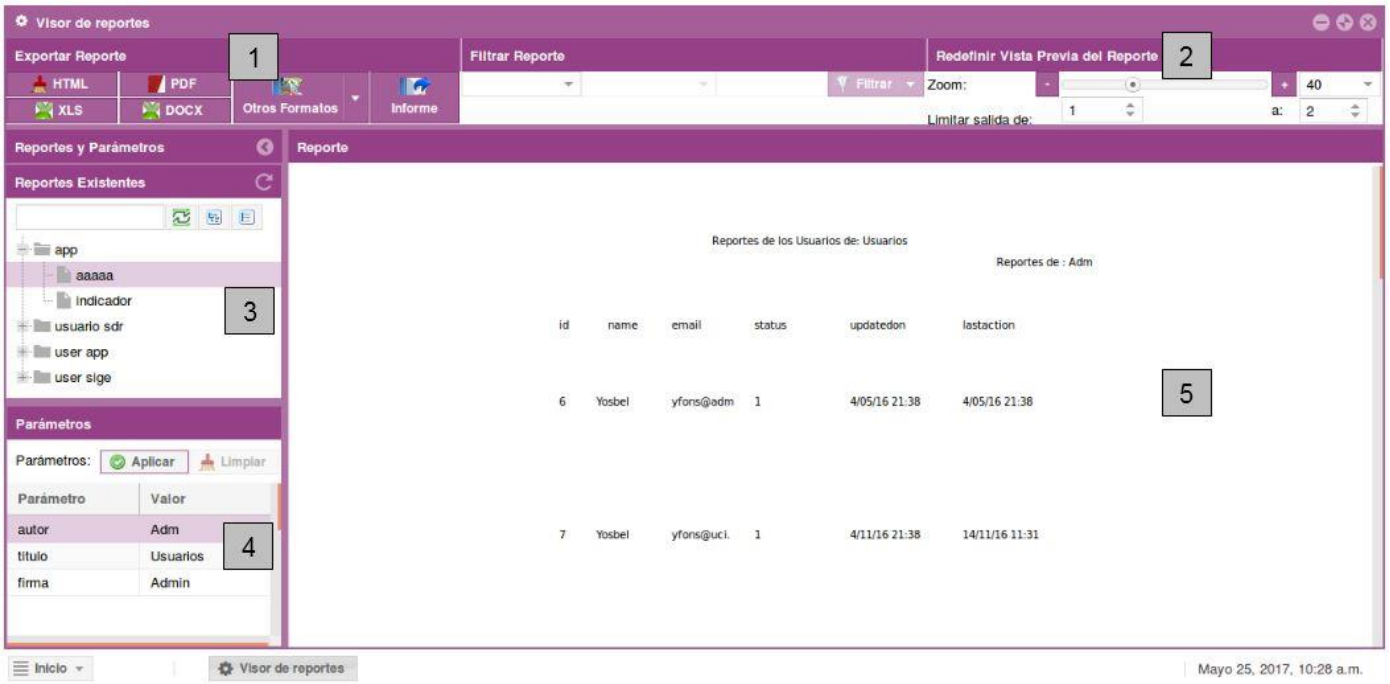


Figura 14: Interfaz Principal

En el área destinada a la exportación de reportes (1) se pueden exportar los mismos a diversos formatos, entre los que se encuentran HTML, PDF, XLS, CSV, DOCX, XLSX, ODS, ODT, RTF, XML y PPT, una vez seleccionada la opción deseada el usuario podrá obtener su reporte en el formato solicitado. Además, se puede generar un informe con todos los reportes seleccionados y exportarlo a uno de los formatos anteriormente mencionados.

Redefinir vista previa del reporte (2) permitirá al usuario modificar la forma en que se verá el reporte en el campo destinado para visualizar el mismo. Se podrá modificar el “zoom” para aumentar o disminuir el tamaño en que se ve el reporte, así como limitar la salida de datos en una página del reporte a cierta cantidad de datos introducida por el usuario.

En el área destinada a listar los reportes existentes (3) se mostrarán cada uno de los reportes almacenados en el sistema. Se puede seleccionar un reporte para visualizarlo, además se podrá expandir y colapsar el árbol donde estarán los reportes almacenados organizados por categoría.

El área de Administrar Valor del Parámetro del Reporte (4) permitirá al usuario modificar, limpiar y aplicar los parámetros al reporte visualizado.

En la vista del reporte (5) se visualiza el reporte requerido por el usuario.

3.4. Pruebas de *software*

El proceso de pruebas se centra en los procesos lógicos internos del *software*, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores. Además, son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. (Pressman, 2009).

3.5. Estrategia de Pruebas

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba (unidad, integración, sistema y aceptación) a ser realizadas y el tipo de prueba a ser ejecutadas (funcional, aceptación e integración) (Pressman, 2009).

La estrategia define:

- Técnicas de pruebas y herramientas a ser utilizadas.
- Qué criterios de éxitos y culminación de la prueba serán usados.
- Consideraciones especiales afectadas por requisitos de recursos o que tengan implicaciones en la planificación.

Niveles de Prueba

La prueba se aplica a diferentes objetos del sistema en diferentes escenarios o niveles de trabajo. Se establecen los siguientes niveles de prueba para evaluar el módulo:

- Prueba de Unidad.
- Prueba de Integración.
- Prueba de Sistema.
- Aceptación.

3.5.1. Prueba de Unidad

Es la prueba enfocada a los elementos más pequeños del *software*. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. El diseño de casos de prueba de una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente. (Pressman, 2009).

Método de Caja Blanca

Las pruebas de Caja Blanca se centran en los detalles procedimentales del *software*, por lo que su diseño está fuertemente ligado al código fuente. Se escogen distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven valores de salida adecuados. (Pressman, 2009).

Mediante la prueba de Caja Blanca se pueden obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Por lo anteriormente mencionado se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importante que se le aplica al *software*, logrando como resultado que disminuya el número de errores existente en los sistemas, aumentando así la calidad y confiabilidad. (Pressman, 2009).

Método de Caja Blanca aplicando la técnica del Camino Básico

La técnica del camino básico permite obtener una medida de la complejidad lógica de un diseño y utilizar esta medida como una guía para la definición de un conjunto básico. En la técnica se derivan los casos de prueba a partir de un conjunto dado de caminos independientes por los cuales se puede circular el flujo de control. Para obtener el conjunto de caminos se realiza el grafo de flujo asociado y se calcula su complejidad ciclomática. (Pressman, 2009).

A continuación, se muestran los pasos para aplicar la técnica:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Paso 1: Notación del Grafo de Flujo

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo. Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo. (Pressman, 2009).

```
public function extractParamsAction(Request $request) {
    try {
        $id = $request->get('idReport');
        $em = $this->getDoctrine()->getManager();
        $report = $em->getRepository("ReportViewerBundle:xmlReport")->find($id);
        $idSdr = $report->getIdreportsdr();
        $sc = new SdrComunicationApi\AuthenticityServices();
        $response = json_decode($sc->login());
        if ($response)
            $token = $response->items[0]->token;
        $paramsReport = array(
            'id' => $idSdr,
            'token' => $token
        );
        $sdrReport = new SdrComunicationApi\ReportServices();
        $data = $sdrReport->showparam($paramsReport);
        $aux = json_decode($data);
        $parametros = array();
        foreach ($aux->items as $nodo) {
            $reportObject['param'] = $nodo->name;
            $reportObject['type'] = $nodo->type;
            $reportObject['valor'] = $nodo->value;
            $parametros[] = $reportObject;
        }
        $response = new Response("{'success': true, "
                . "'data':" . json_encode($parametros) . "}", HttpCode::HTTP_OK);
        return $response;
    } catch (\Exception $e) {
        return new Response(json_encode(array("success" => false,
                    "message" => $e->getMessage(), HttpCode::HTTP_SERVER_ERROR)));
    }
}
```

Figura 15: Fragmento de código a analizar "Administrar Valor del Parámetro del Reporte"

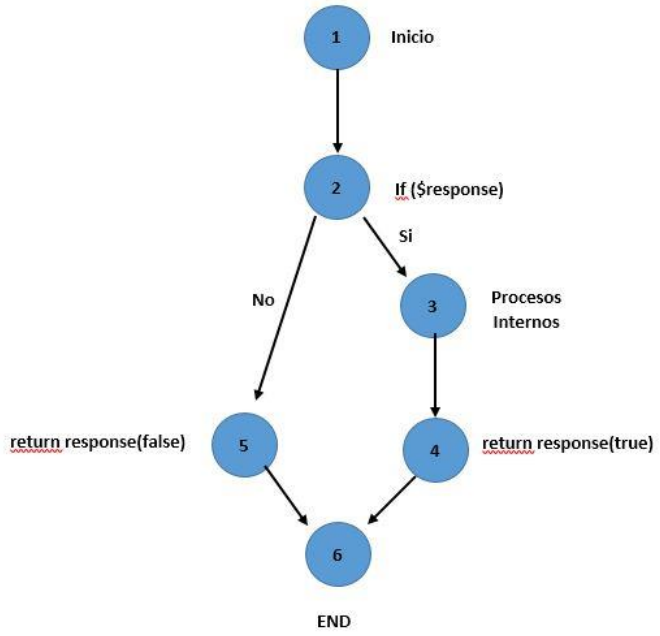


Figura 16: Representación del Grafo de Flujo

Paso 2: Complejidad ciclomática

La complejidad ciclomática es una métrica de *software* que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. (Pressman, 2009).

La complejidad ciclomática V(G), se define como:
V(G) = A – N + 2
Dónde: A es el número de aristas del grafo y N es el número de nodos.
V(G) = 6 aristas - 6 nodos + 2 = 2.

Concluyéndose así que la complejidad ciclomática del grafo de flujo de la figura 15 es igual a 2, lo cual supone que existen solo dos caminos básicos.

Paso 3: Definición de los caminos básicos

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos a una arista que no haya sido recorrida anteriormente.

En la Figura 14 los ejemplos de caminos independientes serían:

Camino 1: 1 – 2 – 3 – 4 – 6

Camino 2: 1 – 2 – 5 - 6

Paso 4: Caso de prueba

En el caso de prueba habrá tantos caminos como grado de complejidad posea el código. En este caso se tiene un camino satisfactorio:

Camino 1: 1 – 2 – 3 – 4 – 6.

1. Inicio
    1.1. Se realiza la petición del ID del reporte
    1.2. Crea el manejador para acceder a la base de datos
    1.3. Busca el reporte en la base de datos
    1.4. Guardar el ID del SDR perteneciente al reporte seleccionado
    1.5. Se crea la conexión para consumir el servicio *AuthenticityServices();*
    1.6. Consume el servicio *AuthenticityServices*
2. Verifica que la respuesta sea correcta *if($response)*
    2.1. Se selecciona el token de autenticación
3. Procesos Internos
4. Retorna la respuesta correcta *return response (true)*

Luego de realizar el traceo del código en el caso de prueba se observa que la respuesta arrojada es satisfactoria, obteniéndose como resultado final la acción de modificar el valor del parámetro de un reporte.

Resultado de las Pruebas de Caja Blanca

Luego de tener elaborados los grafos de flujos y los caminos a recorrer, se prepararon los casos de prueba que forzaron la ejecución de cada uno de esos caminos. Se escogieron los datos de forma que las condiciones de los nodos predicados estuvieran adecuadamente establecidas, con el fin de comprobar cada camino. Luego de confeccionados los casos de prueba se ejecutaron cada uno de estos y se compararon los resultados con los esperados. Una vez terminados todos los casos de prueba se está seguro de que todas las sentencias del programa se ejecutaron por lo menos una vez.

3.5.2. Prueba de Integración

La prueba de integración es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores en las

especificaciones de las interfaces de los paquetes; debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas. (Pressman, 2009).

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. En este se verifica que módulos individuales son combinados y probados como un grupo. En una interfaz es posible perder datos, un módulo podría tener un efecto adverso e inadvertido sobre otro (Pressman, 2009).

Se decide aplicar como tipo de prueba la prueba de integración ascendente, debido a que la aplicación a implementar es un módulo que se integrará posteriormente a un sistema, se tiene en cuenta que comenzará la construcción en el nivel más bajo, llamado también nivel atómico, en el que se encuentran los componentes más bajos de la estructura del programa. Debido a que los componentes se integran de abajo hacia arriba siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel y se elimina la necesidad de resguardarlos.

Aplicación de la Prueba de Integración

Luego de realizar las pruebas de caja blanca se integró el módulo al Sistema Integrado de Gestión Estadística (SIGE v4.0) y se pudo comprobar el correcto funcionamiento con los requisitos definidos. Para realizar el proceso de integración se siguieron los siguientes pasos:

Vista:

1. Crear el paquete del módulo en la vista /package/local/ReportViewer
2. Incluir el nombre paquete en packages/local/common/overrides/sagi2-config.js
   2.1 Adicionar el nombre del paquete report_viewer: “report_viewer”
3. Compilar el sistema Sencha app watch

Servidor:

1. Adicionar el Bundle “ReportViewerBundle” en la ruta src/SIGE/
2. Configurar el fichero routing.yml ubicado en la ruta app/config/
   2.1 Adicionar el código:

   Report_viewer:
   resource: “@ReportViewerBundle/resources/config/routing.yml”
   prefix: /report_viewer/
3. Adicionar el Bundle en el fichero AppKernel.php

   New SIGE\ ReportViewerBundle\ ReportViewerBundle()

Resultado de las Pruebas de Integración

Luego de corregidas las no conformidades detectadas al aplicar el método de caja blanca se comprueba que el módulo resultante se integra al Sistema Integrado de Gestión Estadística (SIGE v4.0). Para ello se ejecutan una serie de pasos que consisten en importar las dependencias y paquetes existentes en el módulo creado, al paquete ReportViewerBundle tanto en la vista de la aplicación como en el servidor. Finalmente se comprueba que el Módulo Visor de Reportes se integra a SIGE v4.0.

3.5.3. Pruebas de Sistema

Son las pruebas que se hacen cuando el *software* está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de *software* y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados. (Pressman, 2009).

Proporciona un aseguramiento final de que el *software* cumple con todos los requisitos funcionales, de comportamiento y desempeño. La prueba se concentra en las acciones visibles para el usuario y en la salida que este puede reconocer. La validación se alcanza cuando el *software* funciona de tal manera que satisface las expectativas razonables (especificación de requisitos de *software*) del cliente. (Pressman, 2009).

Se decide realizar las Pruebas Funcionales como tipo de prueba pues asegura que los requisitos funcionales estén implementados según los requerimientos, esto incluye el correcto funcionamiento de la navegación, entrada de datos, procesamiento, validación y obtención de resultados. Las pruebas funcionales están enfocadas en los casos de uso, aunque pueden estar basadas directamente en los requisitos funcionales y las reglas del negocio. La meta es verificar que existe un correcto funcionamiento en la parte visual de la aplicación, verificando la navegabilidad, la validación de datos, así como la entrada y salida de los mismos. Estas pruebas están basadas en la técnica de Caja Blanca y Caja Negra, esta última verifica el sistema y sus procesos internos mediante la interacción con la aplicación y analiza la salida y los resultados.

Método de Caja Negra

Se centra en los requisitos funcionales del *software*. Permite a los ingenieros de *software* obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (Pressman, 2009).

Método de Caja Negra aplicando la técnica de Partición Equivalente

Es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El diseño de casos de prueba para partición

equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Se realiza con el objetivo de determinar que 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. (Pressman, 2009).

Se obtiene como precondición para la realización de este caso de prueba que se debe seleccionar un reporte obligatoriamente antes de ejecutar cualquier funcionalidad.

A continuación, se detallan las variables que se encuentran asociadas al caso de uso Administrar Valor del Parámetro del Reporte:

Tabla 5: Variables asociadas al CU Administrar Valor del Parámetro del Reporte

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Valor	Campo de Texto	No	Valor asignado a un reporte determinado. Este valor es una cadena de texto que no acepta caracteres numéricos. No puede tomar valor nulo.

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se identificó la técnica de partición equivalencia.

A continuación, se muestra la matriz de datos asociada al escenario Modificar Valor del Reporte.

Tabla 6: Matriz de Datos del escenario Modificar Parámetros del Reporte

Escenario	Variables	Respuesta del sistema	Flujo Central
	1		
EC 1	**V (Ejemplo 1)**	Se modifica el parámetro del reporte	1. Se selecciona el reporte en el listado de reportes existentes

Modificar Parámetro del Reporte			2. Se selecciona un parámetro y se modifica su valor correspondiente. 3. Se selecciona el botón Actualizar.
EC 2 Introducir un valor incorrecto	**I (12567)**	-	1. Se selecciona el reporte en el listado de reportes existentes 2. Se selecciona un parámetro y se modifica su valor correspondiente. 3. Se selecciona el botón Actualizar.

A continuación, se detallan las variables que se encuentran asociadas al caso de uso Exportar Reporte:

Se obtiene como precondición para la realización de este caso de prueba que se debe visualizar un reporte antes de ejecutar cualquier funcionalidad.

Tabla 7: Variables asociadas al caso de uso Exportar Reportes

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Dirección	Campo de Texto	No	Ruta definida para guardar el reporte.
2	Nombre	Campo de Texto	No	Nombre con el que se va a guardar el fichero.

A continuación, se muestra la matriz de datos asociada al escenario Exportar Reporte.

Tabla 8: Matriz de Datos del escenario Exportar Reporte

Escenario	Variables		Respuesta del sistema	Flujo Central
	1	**2**		

EC 1 Exportar Reporte	**V** **(/home/usuario)**	**V** **(Nombre1)**	Se exporta un reporte satisfactoriamente.	1. Se selecciona el reporte en el listado de reportes existentes 2. Se aceptan los parámetros con los que se va a exportar el reporte 3. Se selecciona el formato de exportación 4. Se selecciona el directorio en el que se desea guardar. 5. Se establece el nombre del reporte 6. Se selecciona Aceptar.

Resultado de las Pruebas de Caja Negra

Después de realizar las pruebas de caja negra mediante los casos de prueba asociados a cada caso de uso, se comprobó el correcto funcionamiento del módulo y la correcta validación de los campos, verificando que solo se acepten los caracteres válidos para los mismos. Fueron detectadas 13 no conformidades (Tabla 9) a las que se les dio seguimiento y fueron eliminadas luego de realizar 3 iteraciones como se muestra en el Gráfico 1.

Tabla 9: Resumen de no conformidades detectadas aplicando la prueba de caja negra

Iterac.	**#**	**Descripción de la no conformidad**	**Tipo**	**Estado**	**Respuesta de los desarrolladores**
Primera Iteración	1	Al exportar un informe la ventana que aparece al seleccionar un reporte determinado no permanece delante si se da un clic fuera de la misma.	Interfaz	Corregida	-
	2	El mensaje emergente que aparece al posicionar el cursor sobre el botón de exportar reporte a XLS dice “CLS”.	Ortografía	Corregida	-

	3	No existe coincidencia con los reportes exportados y los reportes visualizados de la categoría “app”.	Sistema	Corregida	-
	4	Al realizar la técnica del camino básico al método de cargar los reportes se encontró que no devuelve la salida esperada pues tiene un ciclo infinito	Sistema	Corregida	
	5	En la funcionalidad Exportar Informe, el nombre de la columna “Campo” tiene la primera letra en minúscula mientras que las demás columnas las tienen en mayúscula.	Ortografía	Corregida	-
	6	El botón de exportar reporte a PPT no muestra el mensaje emergente al posicionar el cursor sobre él como los demás botones del área	Interfaz	Corregida	
	7	La opción de establecer zoom al 80% no se ejecuta	Sistema	Corregida	
Segunda Iteración	8	El campo “valor del parámetro” no debe aceptar caracteres numéricos, si se introduce un carácter de este tipo el sistema sigue mostrando los reportes una vez aplicados los parámetros.	Validación	Corregida	-
	9	El botón “Aceptar” se deshabilita una vez que se visualiza un reporte determinado	Sistema	Corregida	-

	10	No se pueden limpiar los valores de los parámetros luego de accionar el botón “Aceptar”	Sistema	Corregida	-
	11	Al modificar un parámetro el botón “Update” debe aparecer en español	Ortografía	Corregida	-
	12	Cambiar nombre a la funcionalidad “Limitar salida de:” ubicada en el área de redefinir vista previa del reporte.	Interfaz	Corregida	-
	13	Al visualizar un reporte no se muestran los límites de inicio ni fin del mismo.	Interfaz	Corregida	-

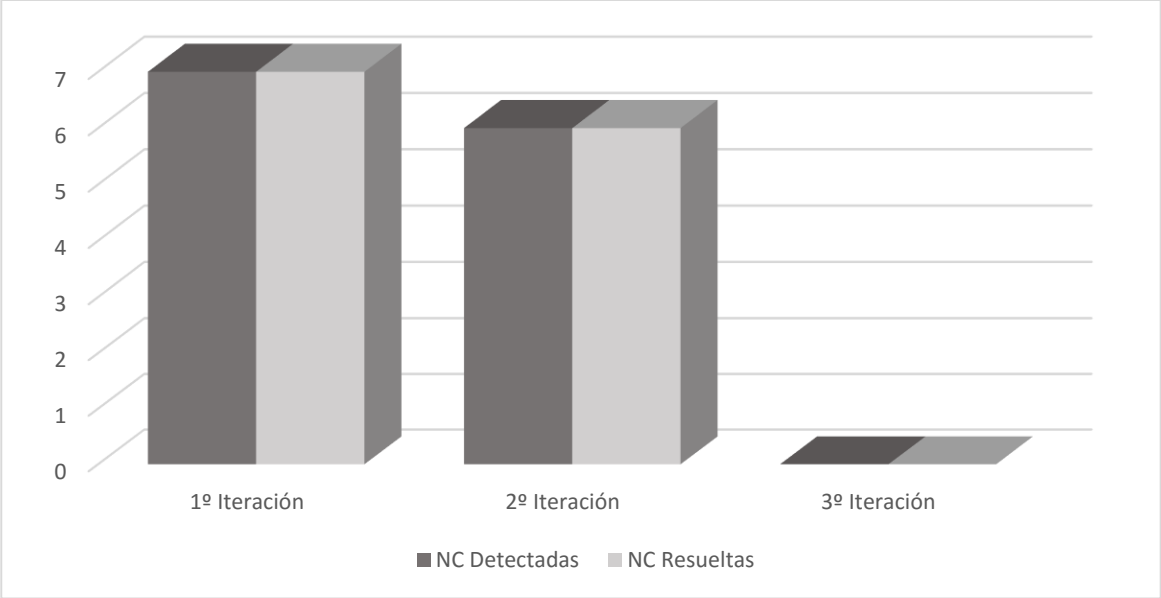


Gráfico 1: Gráfico que representa la relación de no conformidades detectadas y no conformidades corregidas durante las pruebas de caja negra.

3.5.4. Pruebas de Aceptación

Las pruebas de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el *software* está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el *software* fue construido. Son aquellas que son diseñadas por el propio

equipo de desarrollo en base a los requisitos funcionales especificados, y ejecutadas por el propio usuario de manera que este dé validez y conformidad al producto que se les está entregado en base a lo que se acordó inicialmente. De forma general las pruebas de aceptación pueden afrontarse mediante dos tipos de procedimiento para realizarlas: (Pressman, 2009).

- Pruebas Alfa: realizadas por el usuario con el desarrollador como observador en un entorno controlado (simulación de un entorno de producción).
- Pruebas Beta: realizadas por el usuario en su entorno de trabajo y sin observadores.

Se decide aplicar las pruebas alfa ya que en ellas se le entrega a un usuario final todo el producto terminado, junto a su documentación correspondiente para que este, en presencia del desarrollador y en entornos previamente preparados para el proceso de dichas pruebas, vaya informando las inconsistencias y errores que detecte.

A continuación, se muestran las no conformidades detectadas luego de aplicada la prueba de aceptación:

Tabla 10: No conformidades detectadas en la prueba de aceptación

#	Descripción de la no conformidad	Tipo	Estado	Respuesta de los desarrolladores
1	Al cambiar el zoom de un reporte por la barra de desplazamiento queda invalidada la función de cambiarlo por la lista desplegable	Interfaz	Corregido	-
2	Al expandir la categoría de reportes “user app” se muestran los mismos reportes que en la categoría “app”.	Sistema	Corregida	-
3	No se exporta un informe en formato “PPT”	Sistema	Corregida	-

Resultado de las pruebas de *software*

A continuación, se muestran las no conformidades detectadas luego de aplicadas las pruebas de *software*:

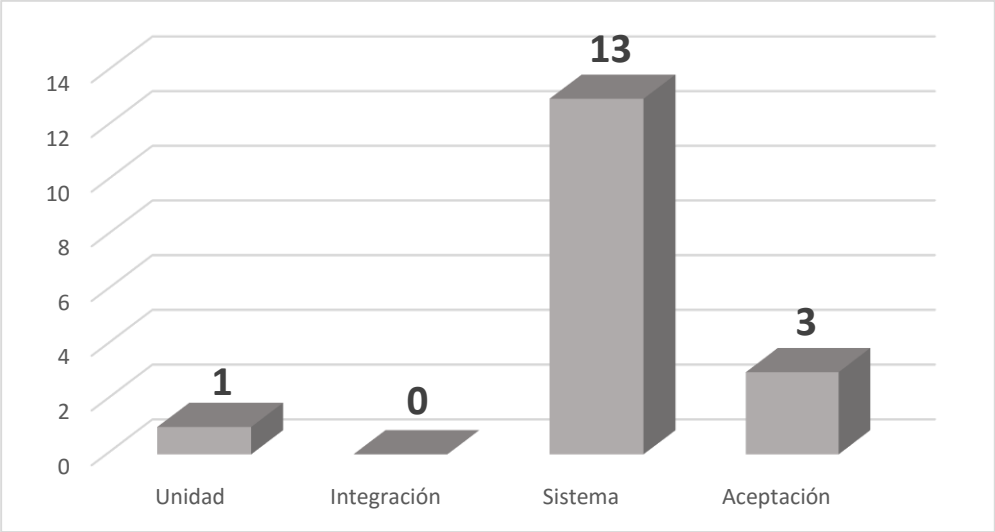


Gráfico 2: Resultado de las pruebas de software

Conclusiones del capítulo

En el desarrollo del presente capítulo se realizaron los diagramas de componentes del Módulo Visor de Reportes con el propósito de mostrar los componentes del sistema y sus relaciones. Se especificó el uso de los estándares de codificación para lograr un estilo claro y organizado del código durante la fase de implementación. Al implementarse el sistema se obtuvo una aplicación funcional, de la cual se presenta la interfaz final explicando cada una de sus funcionalidades. Para validar las funciones de la aplicación y lograr corregir los errores detectados se realizaron pruebas a nivel de unidad, integración, sistema y aceptación, identificándose un total de 17 no conformidades.

Conclusiones Generales

Con la realización de este trabajo diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado al inicio de la investigación. Luego de desarrollar el Módulo Visor de Reportes para SIGE v4.0 se concluye lo siguiente:

1. El análisis crítico de las soluciones existentes para la generación y visualización de reportes realizado evidenció la necesidad de desarrollar una nueva solución tomando como base las principales funcionalidades de los visores de reportes de GDR v2.0, SIGE v3.0, Active Reports y SDR v2.0.
2. La técnica de entrevista no estructurada aplicada permitió obtener los requisitos funcionales y no funcionales del Módulo Visor de Reportes para SIGE v4.0 los cuales fueron validados por el cliente.
3. El diseño de las clases a partir de los requisitos funcionales definidos posibilitó la implementación del Módulo Visor de Reportes para SIGE v4.0.
4. La implementación de las funcionalidades del Módulo Visor de Reportes para SIGE v4.0 permitió que se obtuviera un sistema que respondiera a las necesidades del cliente definidas durante el proceso análisis.
5. El diseño y ejecución de las pruebas de unidad, integración, sistema y aceptación permitieron comprobar el correcto funcionamiento del Módulo Visor de Reportes para SIGE v4.0 detectando las 17 no conformidades que fueron solucionadas posteriormente.

Recomendaciones

Luego del análisis de los resultados del presenta trabajo diploma, se propone la siguiente recomendación:

- Implementar las funcionalidades “Filtro Simple” y “Filtro Avanzado” de reportes para ofrecer una alternativa de búsqueda de datos en un reporte.

Referencias Bibliográficas

1. **ABC, Definición. 2010.** Definición de Reporte. [En línea] 2010. [Citado el: 13 de octubre de 2016.] http://www.definicionabc.com/comunicacion/reporte.php.

2. **Booch, Grady, Rumbrag, James, Jacobson y Ivar. 2009.** *El lenguaje unificado de modelado. Manual de.* 2009.

3. **Bruegge, Bernd, Duotoit, Allen. 2008.** *Ingeniería de software orientado a objetos*.

4. **Chen, Dao-xin. 2011.** *EXTJS Framework in the Development of Application [J]. Computer Knowledge and Technology.* 2011.

5. **disca. 2012.** [En línea] upv, enero de 2012. http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

6. **ECLIPSE. 2008.** [En línea] enero de 2008. [Citado el: 19 de Octubre de 2016.] http://epf.eclipse.org/wikis/openup.

7. **genbetadev. 2014.** Desarrollo de Software. [En línea] 9 de enero de 2014. [Citado el: 15 de diciembre de 2015.] http://www.genbetadev.com/herramientas/netbeans-1.

8. **Medina, Aldis Joan Abreu, y otros. 2012.** *Generador Dinámico de Reportes.* 11, 2012, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. 5, pág. 2.

9. **GEPROIN. 2010.** Consultoría, análisis y programación en la plataforma de desarrollo de aplicaciones empresariales Velneo V7. [En línea] 2010. [Citado el: 18 de Octubre de 2016.] http://www.geproin.es/aplicacion-semana-velneo-adinfrep.

10. **Hispamerica, Databases. 2014.** PostgreSQL 9.4. [En línea] 15 de 4 de 2014. http://www.databases-la.com/?q=node/104.

11. **IBM. 2009.** Knowledge Center. *Knowledge Center.* [En línea] diciembre de 2009. [Citado el: 24 de octubre de 2015.] /www-01.ibm.com/support/knowledgecenter/SSEPGG_8.2.0/com.ibm.db2.udb.doc/ad/c0009415.htm?lang=es.

12. **Kaneiwa, Ken, Mizoguchi, Riichiro y Nguyen, Philip HP. 2015.** *A Logical and Ontological Framework for Compositional Concepts of Objects and Properties.* s.l. : New Generation Computing, 2015.

13. **Laboratorio Nacional de Calidad de Software. 2009.** *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA.* Madrid : Instituto Nacional de Tecnologías de la Comunicación INTECO , 2009.

14. **Larman, C. 2008.** *UML y patrones.* 2008.

15. **Michaud, Frederic. 2015.** *Identifying Key Attack Surface Resources with Dynamic Analysis.* 2015.

16. **Microsoft SQL Server 2005 Reporting Services.** [En línea] 2015. [Citado el: 18 de Octubre de 2016.] https://justindeveloper.wordpress.com/2008/10/20/microsoft-sql-server-2005-reporting-services-part-i/.

17. **Rodríguez, Julio César Brito, y otros. 2013.** *Módulo diseñador de modelos para el generador dinámico de reportes.* 9, Habana : Serie Científica de la Universidad de las Ciencias Informáticas, 2013, Vol. 6. 2306-2495.

18. **Moro, Alfonso Infante, Pérez, Oscar Gallego y Macías, Amparo Sánchez. 2013.** *Los gadgets en las plataformas de telefomación: el caso del proyecto DIPRO 2.0. Pixel-Bit: Revista de medios y educación,* 2013.

19. **Navarro Marset, Rafael. 2007.** *REST vs Web Services.* s.l. : Diseño e Implementación de Servicios Web, 2007. ELP-DSIC-UPV.

20. **OBE, Regina O. y HSU, Leo S. 2015.** *PostGIS in action.* s.l. : Manning Publications Co, 2015.

21. **Paradigm. 2010.** [En línea] www.visual-paradigm.com, 2010. [Citado el: 15 de noviembre de 2015.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

22. **Pérez, N, y otros. 2002.** *Informatización de los Procesos de Gestión en Cuba.* Habana : s.n., 2002.

23. **Potencier, François Zaninotto. 2009.** Symfony, la guia definitiva. [En línea] 2009. [Citado el: 20 de noviembre de 2015.] http://librosweb.es/libro/symfony_1_4/.

24. **Pressman, Roger. 2009.** *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw-Hill/Interamericana, 2009.

25. **Rangel, Eustaquio. 2005.** *PHPReports Manual.* Brasil : s.n., 2005.

26. **Real Academia Española. 2010.** Definición de Reporte. [En línea] 2010. [Citado el: 13 de octubre de 2016.] http://definicion.de/reporte.

27. **Robert Lobo, Armando, y otros. 2011.** *Informatización de los procesos de gestión estadística en Cuba.* Habana : s.n., 2011.

28. **Bieberstein, Norbert, Laird, Robert y Jones, Keith. 2009.** *Service - Oriented Architecture Compass.* Pearson : s.n., 2009. 0-13-187002.

29. *Servidor de Aplicaciones.* **2012.**

30. **Guerra, Ing. Yuned Rivero, y otros. 2016.** *Servidor Dinámmico de Reportes.* La Habana : s.n., 2016.

31. **Silva Hernández, Iliana. 2010.** *Generador Automático de Reportes Dinámicos: Departamento de Computación, Centro de Investigación y de Estudios Avanzados del IPN 2003.* Ciudad México : s.n., 2010.

32. **Sparx. 2000-2007.** [En línea] 2000-2007. [Citado el: 21 de enero de 2016.] http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html. ISBN.

33. **UCI. 2015.** *Metodología de desarrollo para la Actividad productiva de la UCI.* Habana : s.n., 2015.

34. **uml-diagrams. 2010.** [En línea] 2010. [Citado el: 14 de octubre de 2015.] http://www.uml-diagrams.org/.

Bibliografía

1. **Larman, C.** *UML y patrones.* 2008.

2. **GestioPolis**. *Los sistemas integrados de gestión de la información.* [En línea] [Citado el: 7 de noviembre de 2011.] http://www.gestiopolis.com/administracion-estrategia-2/sistemas-integrados-gestion-informacion.htm.

3. **Centro de Tecnologías y Gestión de Datos.** *Caracterización del centro.*

4. Definición ABC. [En línea] [Citado el: 7 de noviembre de 2011.] http://www.definicionabc.com/comunicacion/reporte.php.

5. **Lafaurie, José Rolando.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Universidad de las Ciencias Informáticas, La Habana: s.n., 2008.

6. **Marrero, Ernesto.** *Implementación del módulo de diseño de reportes para el SCADA Guardián del ALBA.* Ciudad de la Habana: s.n., 2009.

7. geproin. Consultoría, análisis y programación en la plataforma de desarrollo de aplicaciones empresariales Velneo V7. *Aplicación de la semana en Velneo.* [En línea] [Citado el: 12 de noviembre de 2011.] http://www.geproin.es/aplicacion-semana-velneo-adinfrep/.

8. Microsoft SQL Server 2005 Reporting Services. [En línea] [Citado el: 2 de octubre de 2011.] http://justindeveloper.wordpress.com/2008/10/20/microsoft-sql-server-2005-reporting-services-part-i/.

9. Presentación de Reporting Services. [En línea] [Citado el: 3 de octubre de 2011.] http://msdn.microsoft.com/es-es/library/ms155786%28v=sql.90%29.aspx.

10. **Duarte, Ailin Orjuela y Rojas, Mauricio.** *Las Metodologías para Desarrollo Ágil como una Oportunidad para la Ingeniería de Software Educativo.* Colombia: s.n., mayo de 2008.

11. **Jamir Antonio Avila Mojica**. *Introducción a patrones de software*.

12. **Pressman, Roger S**.; 2007. “Ingeniería de *software*. Un enfoque práctico”. 6ta Edición. Parte I.

Bibliografía

13. **Kernighan, Brian W, Ritchie, Dennis M. y Gómez, Néstor**. *Lenguajes de programación.* [En línea] 1991.

14. **Eguíluz Pérez, Javier**. Introducción a JavaScript. [En línea] 2008. http://www.libros*web*.es.

15. **Danciu, Teodor y Chirita, Lucian**. *The Definitive Guide to JasperReports™*. 2007.

16. Netbeans. [En línea] [Citado el: 17 de noviembre de 2011.] http://netbeans.org.

17. **Rovelo, Efren A.** symfony. [En línea]. http://www.symfony-project.org/.

18. Modelado de Sistemas com UML. *Popkin Software and Systems.* [En línea] [Citado el: 5 de diciembre de 2011.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

19. Presentación de Reporting Services. [En línea] [Citado el: 3 de octubre de 2011.] http://msdn.microsoft.com/es-es/library/ms155786%28v=sql.90%29.aspx.

20. **Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Universidad de las Ciencias Informáticas, La Habana: s.n., 2008.

21. **ABC, Definición.** Definición de Reporte. [En línea] [Citado el: 13 de octubre de 2016.] http://www.definicionabc.com/comunicacion/reporte.php.

22. **ALEGSA. 1998 - 2016.** ALEGSA.com.ar. *Diccionario de Informática y Tecnología.* [En línea] 1998 - 2016. [Citado el: 18 de Octubre de 2016.]

23. **Cruz, Yaniel, Martinez, Kerton y Borges, Miguel Ángel. 2013.** *Ambiente de configuración para el sistema SCADA "Guardián del ALBA".***.** 7, Habana : Serie Científica de la Universidad de las Ciencias Informáticas, 2013, Vol. 6. 2306-2495.

24. **Anglada, Alain Abel. 2013.** Revista Cubana de Ciencias Informáticas. [En línea] abr-jun. de 2013. [Citado el: 26 de octubre de 2015.] http://scielo.sld.cu/scielo.php?pid=S2227-18992013000200006&script=sci_arttext. ISSN 2227-1899.

25. **CCM.** Comunidad Informática. [En línea] CCMBenchamark group. [Citado el: 18 de Octubre de 2016.] http://es.ccm.net/contents/269-protocolo-ldap.

Bibliografía

26. **Chen, Dao-xin. 2011.** *EXTJS Framework in the Development of Application [J]. Computer Knowledge and Technology.* 2011.

27. **ComponentSource.** [En línea] [Citado el: 18 de Octubre de 2016.] https://www.componentsource.com/es/product/activereports-7.

28. **disca. 2012.** [En línea] upv, enero de 2012. http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

29. **Medina, Aldis Joan, y otros. 2012.** *Generador Dinámico de Reportes.* 11, 2012, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. 5, pág. 2.

30. **GEPROIN.** Consultoría, análisis y programación en la plataforma de desarrollo de aplicaciones empresariales Velneo V7. [En línea] [Citado el: 18 de Octubre de 2016.] http://www.geproin.es/aplicacion-semana-velneo-adinfrep.

31. **Hernández, Yasmany. 2010.** *Manual de Usuarios v1.7 del Generador Dinámico de Reportes.* 2010.

32. **IBM. 2009.** IBM Knowledge Center. [En línea] IBM, diciembre de 2009. [Citado el: 24 de octubre de 2015.]

33. **Jacobson, Ivar y Booch, Grady, Rumbaugh, James. 2004.** *El Proceso Unificado.* 2004.

34. **Kaneiwa, Ken, Mtzogushi, Riichiro y Nguyen, Philip Hp. 2015.** *A Logical and Ontological Framework for Compositional Concepts of Objects and Properties.* s.l. : New Generation Computing, 2015.

35. **Laboratorio Nacional de Calidad de *Software*. 2009.** *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA.* Madrid : Instituto Nacional de Tecnologías de la Comunicación INTECO , 2009.

36. **Martínez, Marilé, et al. 2016.** *Sistema para la Informatización de la Gestión de Anuarios Estadísticos en la ONEI.* s.l. : Serie Científica-Universidad de las Ciencias Informáticas, 2016.

37. **Michaud, Frederic. 2015.** *Identifying Key Attack Surface Resources with Dynamic Analysis.* 2015.

38. **Microsoft SQL Server 2005 Reporting Services.** [En línea] [Citado el: 18 de Octubre de 2016.] https://justindeveloper.wordpress.com/2008/10/20/microsoft-sql-server-2005-reporting-services-part-i/.

Bibliografía

39. **OBE, Regina O. y HSU, Leo S. 2015.** *PostGIS in action.* s.l. : Manning Publications Co, 2015.

40. **Paradigm. 2010.** En línea] www.visual-paradigm.com, 2010. [Citado el: 15 de noviembre de 2015.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

41. **Potencier, François. 2009.** Symfony, la guia definitiva. [En línea] 2009. [Citado el: 20 de noviembre de 2015.] http://libros*web*.es/libro/symfony_1_4/.

42. **Rangel, Eustaquio. 2005.** *PHPReports Manual.* Brasil : s.n., 2005.

43. **Real Academia Española.** Definición de Reporte. [En línea] [Citado el: 13 de octubre de 2016.] http://definicion.de/reporte.

44. **Silva, Iliana. 2003.** *Generador Automático de Reportes Dinámicos: Departamento de Computación, Centro de Investigación y de Estudios Avanzados del IPN 2003.* Ciudad México : s.n., 2003.

45. **Stewart, Claire R., et al. 2016.** *Sensory symptoms and processing of nonverbal auditory and visual stimuli in children with autism spectrum disorder.* s.l. : Journal of autism and developmental disorders, 2016.

46. **uml-diagrams. 2010.** [En línea] 2010. [Citado el: 14 de octubre de 2015.] http://www.uml-diagrams.org/.

Anexos

1. Reconocimiento entregado por presentar el Módulo Visor de Reportes para SIGE v4.0 en la 12º Peña Tecnológica de estudiantes y profesionales.

12ma Peña Tecnológica de estudiantes y jóvenes profesionales

El Comité Organizador de la 12ma Peña Tecnológica de estudiantes y profesionales otorga el presente:

RECONOCIMIENTO

Al (los) autor(es):

Arian Lázaro Álvarez Rodríguez, Rene Manuel Puig Pérez, Rolando Enrique Fuentes Orozco, Glennis Tamayo Morales, Juan Miguel Pérez Almaguer

Por haber presentado la ponencia:

Módulo Visor de Reportes para el Sistema Integrado de Gestión Estadística v4.0

Y obtenido la categoría:

RELEVANTE

En el evento:

X Taller Científico Nacional de jóvenes de la especialidad de tecnologías y sistemas

Entregado en la Universidad de la Ciencias Informáticas, La Habana, a los 19 día del mes de mayo del año 2017.

TC. Dr. Josué Cardoso Castro
Presidente del Comité Organizador del Evento
Presidente del Consejo Científico DTS-MININT

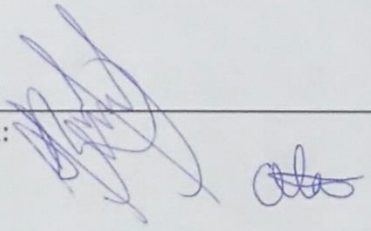
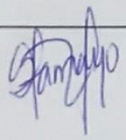
2. Carta de aceptación

UCI Universidad de las Ciencias Informáticas

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución de la tesis: "Módulo Visor de Reportes para el Sistema Integrado de Gestión Estadística v4.0", se hace entrega de los productos que se relacionan a continuación:

- "Módulo Visor de Reportes para el Sistema Integrado de Gestión Estadística v4.0" (código fuente).
- Manual de usuario.

Entrega	**Recibe**
Nombre y apellidos: Ariam Lázaro Álvarez Rodríguez Rene Manuel Puig Pérez	**Nombre y apellidos**: Glennis Tamayo Morales
Cargo: Tesistas	**Cargo:** Jefe de Departamento Desarrollo de Componentes
Firma: 	**Firma:** 

Fecha: 15/06/2015