Universidad de las Ciencias Informáticas

Facultad 6

Módulo de Reportes para el Sistema Integral de Gestión de Datos (SIGDAT).

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Orlando David Mir Fonseca

Alain Milo Solano

Tutores: Ing. Yoander Iñiguez Bermúdez

Ing. Adaily Hernández Carballé

"Siempre le digo a las personas que la libertad es un regalo que se debe compartir"

Matías- Cáceres

Declaración de Autoría

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

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

Orlando D. Mir Fonseca	**Alain Milo Solano**	**Ing.**
Firma del Autor.	Firma del Autor.	Firma del Tutor.

Ing.

Firma del Tutor.

DATOS DE CONTACTO

Tutora: Ing. Adaily Hernández Carballé
Especialidad de graduación:
Categoría docente:
Años de graduado:
Correo Electrónico: acarballe@uci.cu

Tutor: Ing. Yoander Íñiguez Bermúdez
Especialidad de graduación:
Categoría docente:
Años de graduado:
Correo Electrónico: yiniguez@uci.cu

Alain

Dedico el trabajo a mi madre Magalys Solano Crespo por brindarme su amor y dedicación durante toda mi vida y porque no existe persona que merezca un premio más grande que ella.

Orlando

Dedico este trabajo a mis padres María Elena Fonseca Blanco y Orlando P. Mir Rodríguez, pilares imprescindibles en mi formación, a mi futura niña que desde la barriga ya la amo, a mis tutores por su apoyo y de manera general a todo aquel que de una forma u otra aportó un granito de arena a esta investigación.

Alain

Agradezco a mi madre Magalys Solano Crespo, por brindarme confianza, seguridad apoyo y amor durante todo el tiempo en la universidad y sobre todo en esta última etapa de la carrera. A mi abuela Teresa Martínez Pelegrí, por inculcarme todos los valores que me han formado y mantenido por el camino recto. A mi novia Maricelys Navarrete Méndez, que me ha servido de inspiración para avanzar frente a cualquier obstáculo que se me pudiera presentar. A mis compañeros y amigos que he logrado tener en todo el tiempo en la universidad. También agradezco a mis tutores Adaily Hernandez Carballé y Yoander Íñiguez Bermudez por ofrecerme todo el apoyo y ayuda durante todo el proceso de tésis. Al tribunal de tésis y a la oponente Claudia García por ofrecerme críticas constructivas que sirvieron de guía para el desarrollo del trabajo.

Orlando

Agradezco a mis padres Maria Elena Fonseca Blanco y Orlando P. Mir Rodríguez por brindarme su poyo y amor incondicional sin el cual nunca hubiese podido llegar tan lejos. Agradezco también a mi familia en general y a mi novia Roxana Batista Castillo por darme su apoyo. A mis compañeros de guerra con los cuales he compartido cinco largos años de desvelo. También quiero ofrecer mis más sinceros agradecimientos a mis tutores Adaily Hernadez Carballe y Yoander Íñiguez Bermudez por ofrecerme su ayuda incondicional, al tribunal y oponente por las críticas constructivas que nos han guiado en todo este proceso.

RESUMEN

El Departamento de Integración de Soluciones del Centro de Tecnologías de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas (UCI) desarrolló el Sistema Integral de Gestión de Datos (SIGDAT) para informatizar el diseño de los formularios o modelos de recogida de datos, que a su vez soporta los componentes y valida los campos. Los usuarios que interactuan con SIGDAT no tienen forma de obtener reportes detallados sobre los datos que son gestionados a través de las encuestas realizadas. Para la obtención de esta información detallada, el usuario debe realizar consultas a la base de datos donde se encuentran almacenadas las encuestas, para lo que necesita poseer conocimientos de base de datos o apoyarse en un especialista. Con el objetivo de dar solución a esta problemática, se ha desarrollado el Módulo de Reportes para SIGDAT que permite mostrar la información mediante reportes personalizados de los datos recogidos en las encuestas. El Módulo de Reportes para SIGDAT obtenido como resultado del trabajo de diploma, brinda la posibilidad de realizar reportes personalizados sobre datos recogidos por las encuestas mediante una interfaz amigable y de fácil uso; automatizando el proceso de asignación del modelo de datos a los componentes del diseño. Permite exportar los reportes obtenidos a diversos formatos (PDF, Word, Excel), convirtiéndose en una herramienta útil para el apoyo a la toma de decisiones según las necesidades de los usuarios.

PALABRAS CLAVES:

Centro de Tecnologías de Gestión de Datos (DATEC), Módulo de Reportes para SIGDAT, reportes personalizados, Sistema Integral de Gestión de Datos (SIGDAT).

ABSTRACT

The Department of Integration Solutions of Data Management Technology Center (DATEC) at the University of Informatics Sciences (UCI) developed the Comprehensive Data Management System (SIGDAT) to computerize the layout of the forms or templates for data collection, which in turn supports the components and validates fields. Users interact with SIGDAT have no way to get detailed reports on the data that are managed through surveys. To obtain this detailed information, the user must query the database in which are stored the surveys, and need to have knowledge of database or supported by a specialist. In order of solving this problem have been developed the Reports Module for SIGDAT for displaying information through custom data reports collected in the surveys. Reports Module for SIGDAT obtained as a result of the diploma work, offers the possibility of customized reports on data collected through surveys and use a friendly interface; automating the process of assign the data model to design components and allows exporting obtained reports to several formats (PDF, Word, Excel), becoming a useful tool to support decision making according to user needs.

KEYWORDS:

Technology Center of Data Management (DATEC), Comprehensive Data Management System (SIGDAT), custom data reports, Reports Module for SIGDAT.

Índice

Índice de Tablas

Índice de Figuras

Introducción

La gestión de la información en la actualidad persigue la creación de una administración eficiente, que satisfaga las necesidades reales de los usuarios, tales como poder almacenar la información o realizar gráficas basadas en distintos indicadores. Este proceso se hace posible con la introducción de mecanismos que promuevan el desarrollo de servicios de mayor calidad. Además de sistemas de control de la información que otorguen una plena transparencia de los procesos, planes y resultados, para perfeccionar el sistema de información mejorando la agilidad y uso de dicha información.

Hoy en día la mayoría de las empresas y países del mundo se encuentran en proceso de informatización, debido a la gran cantidad de información que se debe procesar. En gran medida estas instituciones se apoyan en softwares que les permitan proporcionar información útil y en tiempo real.

Cuba a pesar de ser un país bloqueado se encuentra inmerso en este proceso, gracias a los avances y recursos dedicados por la Revolución a la informática. La Universidad de las Ciencias Informáticas (UCI) como institución de avanzada lleva la vanguardia en la producción de software con el objetivo de recaudar ingresos para el país, y la informatización del mismo. En la UCI existen varias facultades conformadas por centros de desarrollo de software, departamentos y proyectos productivos, entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC). DATEC cuenta con el departamento de integración de soluciones donde se desarrollan productos como la Solución Integral de Gestión de Datos (SIGDAT). SIGDAT tiene como objetivo realizar el diseño, publicación y captación de encuestas. Está compuesto por cuatro módulos: Seguridad, Fuente de Datos, Encuestas y Diseñador. Brinda a los usuarios la posibilidad de diseñar encuestas, soportando componentes y validaciones para campos del tipo texto, números, fecha y hora. Permite además gestionar las fuentes de datos para publicar los diferentes diseños realizados. Asegura capturas y persistencias de datos asociados a un diseño. Garantiza la gestión de usuarios, roles y permisos de estos sobre cada recurso del sistema.

La herramienta SIGDAT no le permite a los usuarios, teniendo en cuenta sus necesidades, obtener de forma detallada la información recogida en las encuestas publicadas. No brinda apoyo en la toma de decisiones ya que resulta muy complejo realizar un análisis cuantitativo de los datos almacenados. Para la obtención de información detallada los usuarios necesitan tener conocimientos sobre base de datos o contar con un

especialista que le brinde asistencia técnica para realizar las consultas a la base de datos que le permitan obtener los datos.

De la situación anteriormente descrita surge el siguiente **problema de la Investigación:** ¿Cómo obtener reportes detallados en la herramienta SIGDAT a partir de las necesidades particulares de los usuarios que les permita realizar análisis de la información para la toma de decisiones?

Como **objeto de estudio** se define: la elaboración y visualización de reportes personalizados en los Sistemas de Gestión de Datos, enmarcado en el **campo de acción**: la elaboración y visualización de reportes personalizados en SIGDAT. Para darle solución al problema se plantea el siguiente **objetivo general:** Desarrollar un Módulo de Reportes para el Sistema Integral de Gestión de Datos (SIGDAT) que permita a los usuarios realizar reportes personalizados. El cuál se desgloza en los siguientes objetivos específicos.

Objetivos Específicos:

- Analizar los conceptos relacionados con los sistemas generadores de reportes y las nuevas tecnologías.
- Definir las funcionalidades del Módulo de Reportes.
- Diseñar un Módulo de Reportes para SIGDAT.
- Implementar las funcionalidades definidas para el Módulo de Reportes.
- Aplicar pruebas al Módulo de Reportes desarrollado para validar su correcto funcionamiento.

Para el cumplimiento del objetivo general planteamos las siguientes **Tareas de la investigación:**

- Descripción de las tendencias actuales de los sistemas generadores de reportes en Cuba y el mundo.
- Identificación de los procesos y escenarios a tener en cuenta para el diseño de reportes en SIGDAT.
- Definición de los componentes o submódulos necesarios para el módulo de reportes (área de diseño, componentes de diseño, propiedades, modelo de datos).
- Definición de los elementos a usar en el proceso de diseño de reportes (etiquetas de texto, componentes de datos, gráficos de barra y pastel).
- Identificación y definición de los componentes que forman parte del diseño de los reportes.

- Definición de campos calculables y operaciones matemáticas entre campos para mostrar totales, averages, cantidades y promedios.
- Diseño de un módulo de reportes para SIGDAT.
- Implementación del módulo de reportes para SIGDAT.
- Realización de pruebas al módulo de reportes para SIGDAT, para validar el funcionamiento del mismo.

Las tareas de la investigación se derivaron de las siguientes **preguntas de la investigación**:

- ¿Cuáles son las principales herramientas generadoras de reportes que apoyen a que el Módulo de Reportes sea integrable a la herramienta SIGDAT?
- ¿Qué funcionalidades debe tener el Módulo de Reportes para lograr que sea integrable a la herramienta SIGDAT?
- ¿Cómo se han de desarrollar las funcionalidades identificadas?
- ¿Cómo comprobar la integración del Módulo de Reportes a la herramienta SIGDAT?

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

- Análitico-Sintético: Es utilizado para distinguir los elementos de un fenómeno y se procede a revisar ordenadamente cada uno de ellos por separado cediendo la extracción de la información que apoye desde el punto de vista teórico y práctico los elementos que se relacionan con Gestores de Reportes.
- Tormenta de ideas: Es utilizado para conseguir las necesidades del cliente y realizar el levantamiento de los requerimientos con los que debe cumplir el Módulo de Reportes para SIGDAT.
- Modelación: Se demuestra en los diagramas correspondientes a los módelos de análisis, diseño e implementación del Módulo de Reportes para SIGDAT.

El presente trabajo, estructurado en 3 capítulos, resume la siguiente información:

- **Capítulo 1: "Fundamentos Teóricos del Módulo de Reportes para SIGDAT"** se abordan algunos de los elementos y conceptos teóricos necesarios para la concepción del trabajo. Se expondrán las características de las herramientas utilizadas para el desarrollo del módulo y las metodologías de desarrollo de software más conocidas y empleadas.

- **Capítulo 2: “Descripción de la Solución Propuesta”** se identifican, modelan y describen los procesos de negocio, utilizando la metodología de desarrollo seleccionada. Se especifican los requisitos funcionales y no funcionales que tendrán lugar en la implementación. Se identifican el actor y los casos de uso del sistema, así como las relaciones entre ellos mediante el diagrama de casos de uso del sistema. Finalmente se proponen los elementos del diseño a emplear, además de los diagramas de secuencia y despliegue del sistema.
- **Capítulo 3: “Implementación y prueba”** desarrollo y construcción de la propuesta de solución, y la realización de pruebas en busca de no conformidades. En este capítulo se realizará la implementación del módulo, así como las pruebas funcionales de este para garantizar que se corresponda el sistema desarrollado con los requisitos de software definidos. Se realiza el modelo de implementación que muestra el diseño de la solución, así como el diagrama de componentes de la extensión. Se especifican además las pruebas realizadas a la extensión, con el objetivo de comprobar sus funcionalidades en los diferentes escenarios y de esta forma verificar en todos los casos que los resultados de las pruebas sean los esperados.

Capítulo 1: Fundamento Teórico del Módulo de Reportes para SIGDAT

El presente capítulo sienta las bases para el desarrollo de la investigación. En el mismo se abordarán los conceptos y aspectos teóricos que permitirán una mejor comprensión del tema tratado. Se describen las características de las herramientas y la metodología de desarrollo de software utilizadas.

1.1 Gestión de Datos.

Los datos deben que almacenarse de forma segura, pero su acceso con fines de análisis debe ser sencillo. El diseño de un sistema de gestión de la información se ajusta a los principios básicos de la elaboración de datos. En la base de datos se almacenan los datos originales en bruto y el sistema de gestión de datos facilitará integrarse asimismo con el sistema de recopilación de datos. Por lo que se define como gestión de datos: el desarrollo y ejecución de políticas, procedimientos y prácticas que facilitan la gestión de las necesidades del ciclo de vida de los datos (1).

1.1.1. Solución Integral de Gestión de Datos (SIGDAT).

SIGDAT es un software para el diseño y gestión de modelos dinámicos de captura de datos. Está compuesto por cuatro módulos: módulo de seguridad con el componente *Safety*, módulo diseñador con el componente *Designer,* módulo de encuestas con el componente *Survey* y el módulo orígenes de datos con el componente *DataSourceManage*. El sistema brinda a los usuarios la posibilidad de diseñar y personalizar formularios o modelos de recogida de datos; soportando componentes y validaciones para campos del tipo texto, números, fecha, hora, elección, selección simple y otros de diseño. Permite además gestionar las fuentes de datos, locales o remotas, donde publicar los diferentes diseños realizados; efectuar las correspondientes capturas y persistencias de los datos asociados a un diseño. Garantiza la gestión de usuarios, roles y permisos de estos sobre cada recurso procesado en el sistema. Cuenta con dos procesos fundamentales: el diseño de plantillas de encuestas y la captura de datos (2).

1.2 Tendencias Actuales de los Sistemas Gestores de Datos.

Como causa del desarrollo de la humanidad, su necesidad de almacenamiento y consulta de los datos que son recogidos, se hace necesaria la gestión de los mismos mediante Sistemas Gestores de Datos. Estos softwares brindan a las instituciones un manejo más fácil, eficiente y seguro de la información. Algunos de estos programas son:

1.2.1. JasperReport:

Es una librería para la generación de reportes, está escrita en java y es libre. El funcionamiento consiste en escribir un XML donde se recogen las particularidades del informe. Este XML lo tratan las clases de jasper para obtener una salida, la cual puede ser en diversos formatos, como son: pdf, xml, html, csv, xls, rtf, txt. Otra ventaja de utilizar JasperReport es que se integra perfectamente con el JFreeChart que es una librería libre para la generación de todo tipo de gráficas. Utiliza plantillas para generar informes, las que se conforman por diferentes secciones, tales como, título, resumen y detalle. Cada sección tiene un *layout* independiente donde se puede incluir diferentes tipos de elementos: imágenes y campo de texto. El motor de reportes usa la plantilla para organizar los datos dentro de un fichero XML(JRXML) o para crearlo programáticamente utilizando la API que proporciona. Los datos que aparecen en el reporte pueden proceder de diferentes lugares (*data sources*), bases de datos, colecciones, arreglos de objetos java o datos XML. Los desarrolladores podrán crear sus propias fuentes de datos implementando una interface (3).

1.2.2. Crystal Reports

Crystal Reports es una herramienta potente. Brinda a los usuarios la posibilidad de diseñar, explorar y visualizar reportes complejos a través de la web. Posibilitando establecer fácilmente los criterios de formato, agrupamiento, gráficos y filtrado de la información. Es una software privativo, que utiliza como motor para generar los reportes CrystalDecisions. Crystal Reports permite asociar al reporte datos desde diversos orígenes de bases de datos, posibilitando almacenar los reportes en una aplicación Web (4).

1.2.3. Fast Report Studio

FastReport es una herramienta privativa que permite crear reportes orientados a bandas como: título, encabezado, datos, sumario. Utiliza más de 13 motores los cuales son empleados para generar reportes, vistas previas y exportar a diferentes formatos. Puede ser utilizado como una solución independiente para la elaboración de reportes. Brinda la posibilidad de conectarse a diversas bases de datos o crear consultas propias en el lenguaje SQL. Permite el diseño de varios formularios de diálogo para la consulta previa de

parámetros. Crea vistas previas de los reportes diseñados y exportarlos a diversos formatos. Facilita el uso de gráficas en los reportes (5).

1.2.4. Generador Dinámico de Reportes

El Generador Dinámico de Reportes (GDR) es uno de los sistemas desarrollados en el departamento DATEC en la UCI, tiene como propósito garantizar la generación dinámica de reportes partiendo de datos persistentes en algún origen de datos soportado por el sistema (PostgreSQL, MySQL Server, SQLlite, Microsoft SQL Server,Oracle). Está compuesto por un conjunto de módulos que brindan funcionalidades para el trabajo con los reportes brindándoles soporte durante todo el ciclo de vida (6).

1.2.5. Servidor Dinámico de Reportes(SDR)

Este servidor está basado en una arquitectura orientada a servicios utilizando valores de Transferencia de Estado Representacional (REST). Es un software que posibilita la gestión, compilación, publicación y exportación de reportes creados en herramientas de diseño de reportes como el GDR u otra aplicación que solicite el servicio. Además permite incluso a un usuario independiente, que de forma directa interactúe con el servidor a través de algún protocolo de comunicación pueda consumir los servicios; siempre y cuando tenga los debidos permisos recibirá exitosamente la respuesta a la petición realizada. Utiliza la librería JasperReport para generar los reportes (7).

Del estudio realizado anteriormente se llega a la conclusión de que no es factible la integración de las herramientas CrystalReport y FastReport a SIGDAT por ser de carácter privativo. La herramienta CrystalReport aportó a la investigación la idea de integrar el área de diseño del reporte con la vista previa del mismo (brinda al usuario la posibilidad de realizar una vista previa del reporte sin salir del área de diseño). La herramienta FastReport aportó a la investigación la manera de estructurar la hoja del diseño del reporte (dividir la hoja del reporte en bandas). El GDR brinda a la investigación la idea para el diseño del editor de consulta (permite al usuario construir consultas básicas mediante una interfaz visual). GDR a pesar de ser una herramienta potente, fácil de integrar, no es libre pero si accesible por ser un producto desarrollado en la UCI. Se decide no utilizarla por dos razones:

- Cuando se realizó el estudio de GDR este se encontraba en su versión 1.8, usando como motor para la generación de reportes PHPReport, el cual se encuentra actualmente superado por motores generadores de reportes como JasperReport. De igual forma los desarrolladores de GDR en la versión 2.0 pensaban migrar hacia el motor de reportes de JasperReport.

- Se persigue automatizar el proceso de asignación del modelo de datos de las encuestas publicadas a los componentes del diseño del reporte. Dicho proceso en GDR se realiza de forma manual, los usuarios deben introducir el modelo de datos para luego asignárselos a los componentes en el diseñador.

Para la visualización de los reportes se utilizará el SDR. Permitiendo exportar los reportes a diferentes formatos y le brinda al usuario la posibilidad de contar con una vista previa del reporte en diseño.

1.3 Metodología de desarrollo de software.

SIGDAT es una herramienta completamente funcional por lo que se adoptó la política de respetar la metodología usada en la elaboración del sistema en el módulo de reportes, la cual fue *Open Unified Process* (OPENUP).

1.3.1. Open UnifiedProcess

OpenUP es una metodología ágil e iterativa, completa y extensible. Constituye un marco de trabajo para procesos de desarrollo de software. Como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Es un proceso de desarrollo unificado que está basado en *Rational Unified Process* (RUP) y forma parte de la familia del Proceso Unificado que lo conforman además de los mencionados *Agile Unified Process* (AUP), *Enterprise Unified Process* (EUP) y *Basic Unified Process* (BUP). OpenUP preserva la esencia del *UnifiedProcess* mantiene las mismas características de RUP: desarrollo iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura (8).

1.4 Herramientas asociadas al desarrollo del software.

En este epígrafe se describirán y se realizará un estudio sobre las herramientas que se emplearán en la realización del Módulo de Reportes para SIGDAT, lo que permitirá realizar una correcta selección de las mismas.

1.4.1. Herramientas CASE.

Las Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida del desarrollo de un software

(investigación preliminar, análisis, diseño, implementación e instalación). Es la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. Permite que las compañías puedan desarrollar rápidamente sistemas de mejor calidad para soportar procesos críticos del negocio, asistir en el desarrollo, promoción intensiva de la información de productos y servicios (9).

1.4.2. Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML por sus siglas en inglés *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un plano del sistema (modelo). Incluye aspectos conceptuales tales como procesos de negocios, funciones del sistema, además de aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (10).

1.4.3. Visual Paradigm 8.1

Visual Paradigm provee soluciones de software que permiten a las organizaciones desarrollar aplicaciones con calidad. Brinda un desarrollo contínuo de software que ayuda a los clientes a transformar con precisión sus necesidades de sistema en soluciones de software, todo ello con el mínimo riesgo y el máximo retorno de la inversión. Todos los productos de Visual Paradigm están diseñados y desarrollados para eliminar la complejidad, mejorar la productividad y desarrollo de software (11).

1.4.4. Entorno integrado de desarrollo (IDE)

Un IDE es un programa que está basado en materiales de programación, el cual puede dedicarse a un solo lenguaje de programación o bien puede estar dirigido para utilizarse en varios lenguajes. Se define como una aplicación que consiste en un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario (GUI) (12).

NetBeans 7.4

El IDE netbeans es un entorno de programación para varios lenguajes, incluyendo a Java y C++. Este desarrollo es de código fuente abierto, se proporciona el código fuente del entorno para que se pueda modificar de acuerdo a ciertos parámetros de licencia. Es también una plataforma de ejecución de aplicaciones que facilita la escritura de aplicaciones Java, proporcionando una serie de servicios comunes que a su vez están disponibles a través del IDE. Esta nueva versión posee las características siguientes:

- Desarrollo en HTML5 para dispositivos Androide e iOS.

- Desarrollo en HTML5 en Java EE y aplicaciones PHP.
- Soporte de edición para los entornos Knockout y Angulas JS.
- Soporte para Java SE 8.
- Soporte rediseñado para JavaFX, de acuerdo a la arquitectura JDK8 (13).

1.4.5. Sistema Gestor de Base de Datos (SGBD)

Un SGBD consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a los mismos. Normalmente se sitúan en el núcleo de un Sistema de Información (SI). En principio se utilizaron para almacenar los atributos temáticos asociados a un conjunto de entidades especiales almacenadas en formato vectorial, hoy en día se están empezando a utilizar además para el almacenamiento de la información geométrica de las entidades especiales. Aunque se han hecho algunos intentos para almacenar información en formato raster en un SGBD (14).

PostgreSQL 9.1

PostgreSQL 9.1 es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia *Berkeley Software Distribution* (BSD[1]) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. *"PostgreSQL 9.1 provee algunas de las más avanzadas características empresariales de cualquier base de datos de código abierto, y es soportado por una entusiasta e innovativa comunidad con probados casos de éxito. PostgreSQL está muy bien preparada para construir y correr aplicaciones en la nube."* palabras de Charles Fan, Sr. VP R&D, VMware (15).

PgAdmin III 1.14

PgAdmin es una herramienta de código abierto para la administración de bases de datos PostgreSQL, multiplataforma. La administración y la elaboración de consultas se realizan a través de una interfaz gráfica. Presenta un editor de código procedural y cuenta con un agente de planificación SQL. No se requieren

[1] Licencia BSD: Licencia de software libre permisiva. Permite el uso del código fuente en software privativo.

controladores adicionales para comunicarse con el servidor de base de datos. Es un software libre publicado bajo la licencia PostgreSQL (16).

1.5 Lenguaje de Programación

Se entiende como un sistema de comunicación que posee una determinada estructura, contenido y uso. La programación es, en el vocabulario propio de la informática, el procedimiento de escritura del código fuente de un software. De esta manera, puede decirse que la programación le indica al programa informático qué acción tiene que llevar a cabo y cuál es el modo de concretarla. Es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora (17).

1.5.1. PHP 5.3

PHP (*Hypertext Preprocessor*, Preprocesador de Hipertexto) es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto) en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web (18).

1.5.2. Java Script

JavaScript es un lenguaje interpretado orientado a objetos que es pequeño y ligero; no es útil como un lenguaje independiente ya que está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores web. Dentro de un entorno anfitrión, JavaScript puede ser conectado a los objetos de su entorno para proveer un control programable sobre estos (19).

Se empleará JavaScript para la manipulación de los eventos del lado del cliente, mientras que PHP 5.3 permitirá codificar la lógica de negocio del módulo que se encuentra del lado del servidor.

1.6 Marcos de trabajo de desarrollo (Framework)

En este epígrafe se realiza el estudio sobre los framework que se utilizarán para el desarrollo del módulo de reportes, Symfony 2.0, Ext JS 3.4 y Lycan. La herramienta SIGDAT se encuentra desarrollada en dichos frameworks lo que posibilitará una mejor integración del módulo con la herramienta.

El concepto framework se emplea en muchos ámbitos del desarrollo de software, no es solo en el de aplicaciones web. Se pueden encontrar en el desarrollo de aplicaciones médicas, de visión por computadoras o para el desarrollo de juegos. En general, el término framework se refiere a una estructura de software compuesto de componentes personalizables e intercambiable para el desarrollo de una aplicación. Es una aplicación genérica y configurable para construir un software. Los objetivos principales que persigue son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo con el uso de patrones (20).

1.6.1. Symfony 2.0

Symfony 2.0 es un framework diseñado para la optimización del desarrollo de las aplicaciones web. Basado en el patrón Modelo Vista Controlador. Este separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony 2.0 está desarrollado completamente en PHP 5.3 y es compatible con la mayoría de gestores de bases de datos, como Misal, Postgres, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas (Unix, Linux) como en plataformas Windows (21).

1.6.2. Ext JS

Ext JS es una biblioteca o conjunto de librerías de JavaScript para el desarrollo de aplicaciones web interactivas. Usa tecnologías AJAX, DHTML y DOM. Permite realizar completas interfaces de usuario, fáciles de usar, parecidas a las aplicaciones de escritorio. Esto brinda a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones y no en las advertencias técnicas. Puede usarse como extensión para las bibliotecas jQuery y Prototype. Originalmente fue desarrollado por Jack Slocum como una extensión de la librería YUI en 2006. Posee una serie *Componentes de usuario. Ext JS* que incluye un conjunto de controles o widgets para el desarrollo de interfaces en los desarrollos web, algunos de estos son:

- Campos de texto.
- Campos de fecha con un menú calendario desplegable.
- Campos numéricos.
- Listas y combos.

- Radios y CheckBox's.
- Grids para desplegar información, con diferentes modelos de selección, ordenamiento y columnas arrastrables (22).

1.6.3. Lycan

Lycan es un framework desarrollado en la UCI. Se utiliza para el diseño integrado de componentes por lo que asiste este trabajo para Ext JS de manera intuitiva, rápida y cómoda, promoviendo la usabilidad y elevando la productividad de los desarrolladores. Este framework surge para resolver la principal dificultad de Ext JS que radica en la curva de aprendizaje requerida por un desarrollador para su dominio y por la baja productividad debido a la inmadurez de los entornos de desarrollo integrado (23).

Características:

- Mecanismos flexibles y escalables de extensión a nivel de plataforma.
- Soporte íntegro a los componentes de formulario
- Soporte íntegro a árboles.
- Compatibilidad con diferentes navegadores
- Soporte íntegro a Grid.

Se decide utilizar como framework de desarrollo del lado del servidor Symfony ya que es un framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador y se encuentra desarrollado completamente en PHP 5. Del lado del cliente se utiliza el framework ExtJS por ser una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. También se utiliza Lycan por brindar facilidades a los desarrolladores.

Se hace necesario definir el IDE de desarrollo a través del cual se podrá trabajar con las tecnologías y frameworks de desarrollo definidas.

1.7 Servidor de Aplicaciones

Se denomina servidor de aplicaciones a una red de computadoras que ejecuta ciertas aplicaciones. Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios que aporta son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones (24).

1.7.1. Apache 2.2.22

Apache es un servidor HTTP de código libre, ejecutable en varios sistemas operativos. Permite múltiples lenguajes de script (PHP, Perl, Tcl, Python). Presenta soporte para J2EE[2] (*Java Platform, Enterprise Edition*, Java Empresarial). Posibilita la implementación de *Virtual Host*[3] (Anfitrión Virtual). Posee un diseñador modular el cual permite crear nuevos módulos con el API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) de Módulos de Apache (25).

1.8 Patrones

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan (26).

1.8.1. Patrones de Arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico. Presenta un esquema genérico y probado de su solución. Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos (27).

Modelo Vista Controlador

El patrón de arquitectura de modelo-vista-controlador (MVC) divide una aplicación interactiva en tres partes. El modelo contiene los datos y la funcionalidad esencial. Las vistas despliegan la información al usuario. Los controladores manejan todas las peticiones que se realizan al sistema. Las vistas y los controladores juntos componen la interfaz con el usuario. El mecanismo de cambio-propagación asegura la consistencia de la interfaz con el modelo (28).

1.8.2. Patrones de diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. Brinda una solución a un

[2] J2EE: Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

[3] Virtual Host: Hacer funcionar más de un sitio web en una sola máquina.

problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (26).

Patrones GoF

En 1994 fue escrito el libro *DesignPatterns: Elements of Reusable ObjectOrientedSoftware* por *Ganf of Four* (GoF, Pandilla de los Cuatro), formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. El grupo de GoF clasificaron los patrones en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (29).

Patrones creacionales: Engloban aquellos patrones de software que se centran en la forma de crear las clases y sus instancias, así como el uso que recibirán una vez creadas.

Abstract Factory: Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

Builder: Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

Factory Method: Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.

Prototype: Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.

Singleton: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Patrones estructurales: Engloban aquellos patrones de software que se centran en la composición y estructura de las clases y en la herencia entre ellas.

Adapter: Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

Bridge: Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

Composite: Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

Decorator: Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

Facade: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

Flyweight: Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.

Proxy: Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

Patrones de comportamiento: Engloban aquellos patrones de software que se centran en la comunicación entre las distintas clases y objetos.

Chain of Responsibility: Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

Command: Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.

Interpreter: Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.

Iterator: Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

Mediator: Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

Memento: Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

Observer: Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

State: Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.

Strategy: Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables, permite que un algoritmo varíe independientemente de los clientes que lo usan.

TemplateMethod: Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

Visitor: Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

Patrones Grasp

Describen los principios fundamentales a la hora de la asignación de responsabilidades a objetos, los cuales son expresados en forma de patrones. Grasp es un acrónimo que significa General Responsability asignament software pattern. El nombre se eligió para resaltar la importancia de adoptar estos principios, si se quiere realizar un diseño eficaz de un software orientado a objetos. Patrones de Software Generales para la Asignación de Responsabilidades (GRASP) (30).

Tabla 1 Patrones GRASP

Patrón	Descripción
Experto en Información	Asigna la responsabilidad al experto en información (clase que tiene toda la información necesaria para implementar una responsabilidad): -La clase que tiene la información necesaria para realizarla.
Creador	Asigna a una clase (B) la responsabilidad de crear una instancia de otra (A) si se cumple alguno de los puntos siguientes: 1. B contiene a A. 2. B agrega a A. 3. B tiene los datos de inicialización de A. 4. B registra a A. 5. B utiliza estrechamente a A.
Controlador	Gestiona un evento del sistema. Asigna la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas opciones:

	-Representa el sistema global, dispositivo o un subsistema (controlador de fachada). -Representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o sesión).
Bajo Acoplamiento (evaluativo)	Brinda soporte a las bajas dependencias y al incremento de la reutilización Asigna responsabilidades de manera que el acoplamiento (innecesario) se mantenga bajo.
Alta Cohesión (evaluativo)	Mantiene manejable la complejidad. Asigna responsabilidades de manera que la cohesión permanezca alta.
Polimorfismo	Es el responsable cuando el comportamiento varía en función del tipo. Cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asigna la responsabilidad del comportamiento utilizando operaciones polimórficas a los tipos para los que varía el comportamiento.
Fabricación Pura	Asigna un conjunto altamente cohesivo de responsabilidades a una clase de comportamiento artificial o de conveniencia que no representa un concepto del dominio del problema, para dar soporte a la alta cohesión, bajo acoplamiento y la reutilización.
Indirección	Asigna responsabilidades para evitar el acoplamiento directo. Asigna la responsabilidad a un objeto intermedio para mediar entre otros componentes o servicios, de manera que no se acoplan directamente.
Variaciones Protegidas	Asigna responsabilidades a los objetos, subsistemas, y sistemas de manera que las variaciones o inestabilidad en estos elementos no influya de manera no deseable en otros elementos. Identifica los puntos de variaciones predecibles o inestabilidad. Asigna las responsabilidades para crear una interfaz estable alrededor de ellos.

Conclusiones parciales del Capítulo

Se comprobó que la información permite resolver problemas y tomar decisiones a través de su gestión, por lo que en la actualidad se desarrollan software para apoyar a la toma de decisiones en la mayoría de instituciones o empresas. Se analizaron algunas de las herramientas que son utilizadas para la generación

de reportes, las cuales aunque no se utilizan por las razones descritas, aportaron las ideas para la confección de la hoja de diseño del módulo de reportes. Se seleccionó como metodología de desarrollo de software OpenUP, la cual guiará todo el proceso de desarrollo del módulo. Se estudiaron las herramientas y tecnologías que se emplearán en la solución seleccionando UML 2.0 como leguaje de modelado, Visual Paradigm 8.1 como herramienta CASE, PHP 5.3 y JavaScript como lenguajes de programación. Se utilizarán como marcos de trabajo Symfony 2.0 del lado del servidor y ExtJS 3.4 con Lycan del lado del cliente; el IDE de desarrollo a utilizar será el NetBeans 7.4. Como gestor de base de datos se empleará PostgreSQL 9.1 y para su administración gráfica el PgAdmin III 1.14. Apache 2.2.22 será utilizado como servidor web. Estas herramientas apoyarán el desarrollo general del módulo, brindando facilidades a los desarrolladores en cuanto a rapidez, facilidad de uso y variedad de funciones. Se realizó un estudio de los patrones arquitectónicos y de diseño, los cuales son fundamentales para lograr un sistema bien estructurado y reutilizable.

Capítulo 2: Descripción de la Solución Propuesta

Introducción

En el presente capítulo se realiza el diseño del Módulo de Reportes para la herramienta SIGDAT. Se explica el modelo del dominio como representación conceptual del entorno en el que se encuentra enmarcada la solución. Se definen los requisitos funcionales y los no funcionales; posteriormente los requisitos se agrupan en casos de uso y se realiza el diagrama de casos de uso del sistema. También se identifican los patrones a utilizar en la solución del móduloasí como el diseño de los diagramas de clases que describen y guían la implementación del sistema.

2.1 Modelo de Dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis (31). Este modelo es una representación conceptual de la realidad del ambiente de desarrollo del módulo.

En este caso se refleja que el usuario utiliza SIGDAT, esta herramienta se encarga de la gestión de encuestas. Las encuestas manejan cierta cantidad de información relevante, estos datos son gestionados por consultas que realiza el usuario para obtener información. El usuario utilizará esta información para satisfacer hasta cierto punto sus necesidades (ver figura 1).

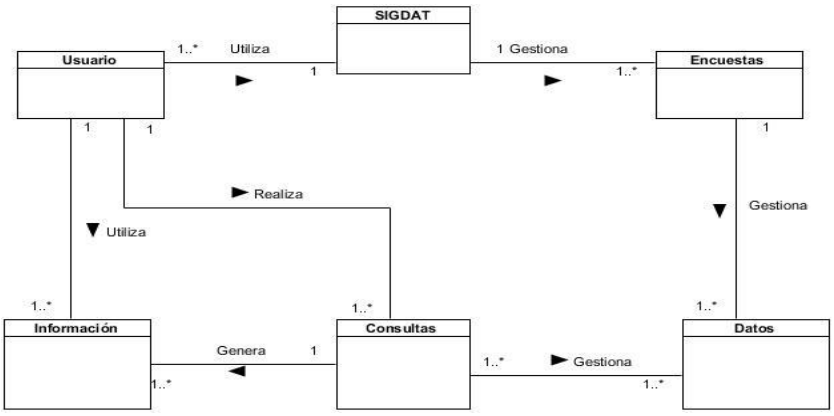


Fig. 1 Modelo del Dominio

2.1.1. Descripción del Modelo de Dominio

- **Usuario:** Representa la persona que accede a SIGDAT.
- **SIGDAT:** Sistema que brinda al usuario la posibilidad de diseñar y personalizar formularios o modelos de recogida de datos. Soporta componentes y validaciones para campos del tipo texto, números, fecha, hora, elección, selección simple y otros de diseño. Permite además gestionar las fuentes de datos, locales o remotas, donde publicar los diferentes diseños realizados. Realiza las correspondientes capturas y persistencias de los datos asociada a un diseño, así como la gestión de usuarios, roles y permisos de estos sobre cada recurso procesado en el sistema.
- **Encuestas:** Recopilación de información de diversas materias destinadas al manejo de estos datos.
- **Consultas:** Método para acceder y buscar datos atendiendo a las solicitudes del usuario.
- **Datos:** Información recogida por las encuestas publicadas, que se encuentran almacenadas en la base de datos y representan la información a mostrar en los reportes.
- **Información:** Respuesta obtenida a través de las consultas realizadas por el usuario sobre los datos.

2.2 Especificación de los requisitos del sistema

2.2.1. Requisitos funcionales

Los requisitos son un grupo de condiciones que necesita un usuario para solucionar un problema y lograr su objetivo, las cuales tienen que ser alcanzadas por un sistema o componente del mismo para satisfacer un contrato, estándar u otro documento impuesto formalmente (32).

Requisitos funcionales que debe cumplir el Módulo de Reportes:

RF1. Adicionar reporte.

- **Descripción:** Adicionar un nuevo reporte al sistema.
- **Entrada:** Datos del nuevo reporte.
- **Salida:** Mensaje de Información y actualización de la base de datos.

RF2. Modificar reporte.

- **Descripción:** Modifica un reporte existente en el sistema.
- **Entradas:** Nombre de reporte, campos a modificar.
- **Salida:** Mensaje de información y actualización de la base de datos.

RF3. Listar reporte.

- **Descripción:** Lista los reportes existentes en el sistema de una plantilla publicada.
- **Entradas:** Identificador de una plantilla.
- **Salida:** Todos los reportes asociados a la plantilla.

RF4. Eliminar reporte.

- **Descripción:** Elimina un reporte existente en el sistema
- **Entradas:** Reporte a eliminar.
- **Salida:** Mensaje de información y actualización de la base de datos.

RF5. Exportar reporte.

- **Descripción:** Exporta el reporte a un formato determinado, Word o PDF.
- **Entradas:** Diseño del reporte.
- **Salida:** Reporte exportado al formato indicado.

RF6. Adicionar componentes de diseño al reporte.

- **Descripción:** Adiciona componentes de diseño como Label, Logo, separador de línea.
- **Entradas:** Componente de diseño.
- **Salida:** Componente adicionado en el área de diseño.

RF7. Modificar componentes de diseño del reporte.

- **Descripción:** Modifica los componentes del diseño, cambia el tamaño, nombre que va a mostrar el componente en el reporte, que campos del modelo de datos de la plantilla se asocia.
- **Entradas:** Componente a modificar.

- **Salida:** Componente modificado en el área de diseño.

RF8. Eliminar componentes de diseño del reporte.

- **Descripción:** Elimina un componente seleccionado del área de diseño del reporte.
- **Entradas:** Componente de diseño.
- **Salida:** Componente eliminado en el área de diseño.

RF9. Alinear componentes de diseño del reporte.

- **Descripción:** Cuando existen componentes en el área de diseño del reporte, permite alinearlos, en cualquier sentido, izquierda, derecha, en el centro.
- **Entradas:** Al menos un componente en el área de diseño.
- **Salida:** Componente o componentes alineados.

RF10. Duplicar componentes de diseño del reporte.

- **Descripción:** Duplica un componente del área de diseño del reporte, crea un componente con las mismas propiedades y características que el original.
- **Entradas:** Al menos un componente en el área de diseño.
- **Salida:** Componente duplicado.

RF11. Guardar cambios sobre el diseño del reporte.

- **Descripción:** Deshace una acción no deseada sobre el área de diseño del reporte.
- **Entradas:** Diseño del reporte.
- **Salida:** Diseño del reporte.

RF12. Cerrar el área de diseño del reporte.

- **Descripción:** Retoma una acción pasada sobre el área de diseño del reporte.
- **Entradas:** Diseño del reporte.
- **Salida:** Diseño del reporte.

RF13. Vista Previa del reporte.

- **Descripción:** Una vista previa de cómo quedaría el diseño del reporte una vez terminado.
- **Entradas:** El código del diseño en formato HTML.
- **Salida:** Una página en formato HTML.

RF14. Adicionar Modelo de Datos

- **Descripción:** Adiciona un modelo de datos a los componentes del diseño del reporte como: tablas, gráficas, datos calculables.
- **Entradas:** Componentes del diseño del reporte, como: tablas, gráficas, datos calculables.
- **Salida:** La respuesta de una consulta a la base de datos, ya sea un nombre, una comparación o cálculo entre datos calculables.

RF15. Publicar reporte.

- **Descripción:** Adicionar un nuevo reporte publicado al sistema.
- **Entrada:** Datos del nuevo reporte publicado.
- **Salida:** Actualización de la base de datos.

RF16. Modificar reporte publicado.

- **Descripción:** Modifica un reporte publicado existente en el sistema.
- **Entradas:** Nombre de reporte, campos a modificar.
- **Salida:** Mensaje de información y actualización de la base de datos.

RF17. Listar reporte publicado.

- **Descripción:** Lista los reportes publicados existentes en el sistema.
- **Salida:** Todos los reportes publicados existentes en el sistema.

RF18. Eliminar reporte publicado.

- **Descripción:** Elimina un reporte publicado existente en el sistema.

- **Entradas:** Reporte a eliminar.
- **Salida:** Mensaje de información y actualización de la base de datos.

RF19. Adicionar estilo.

- **Descripción:** Adicionar un nuevo estilo al diseño del reporte.
- **Entrada:** Datos del nuevo estilo.
- **Salida:** Actualización de Json del modelo del diseño.

RF20. Modificar estilo.

- **Descripción:** Modifica un estilo existente en el modelo del diseño.
- **Entradas:** Nombre, color, tamaño de letra y tipo de letra.
- **Salida:** Actualización del modelo del diseño.

RF21. Listar estilo.

- **Descripción:** Lista los estilos existentes en el modelo del diseño.
- **Salida:** Todos los estilos existentes en el modelo del diseño.

RF22. Eliminar estilo.

- **Descripción:** Elimina un estilo existente en el modelo del diseño.
- **Entradas:** Estilo a eliminar.
- **Salida:** Actualización del modelo del diseño.

RF23. Crear consulta SQL.

- **Descripción:** Crea una consulta y se guarda en el modelo del diseño del reporte.
- **Entradas:** Consulta SQL.
- **Salida:** Actualización del modelo del diseño.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales generalmente se vinculan a los requisitos funcionales. Después de conocido lo que el software debe hacer, se analizan las necesidades tanto de hardware como de software para que el sistema funcione correctamente (32).

Requisitos de Software:

RNF1. Se debe tener instalado el servidor web Apache.

RNF2. Gestor de Base de Datos PostgresSQL 9.1.

RNF3. Se debe tener instalado el navegador Mozilla Firefox en su versión 16.0.0 mínimo.

Requisitos de Hardware:

RNF4. Mínima capacidad para la Memoria de Acceso Aleatorio (RAM por sus siglas en inglés *Random Access Memory*) debe ser de 512 MB.

RNF5. El microprocesador con velocidad de 1.7GHz como mínimo.

RNF6. Disco duro con capacidad de 10Gb como mínimo.

Restricciones de diseño e implementación:

RNF7. Se hace uso de la herramienta CASE “Visual Paradigm for UML” en su versión 8.1 y el IDE NetBeans 7.4.

RNF8. Los frameworks de desarrollo utilizados son Ext JS 3.4, Lycan, Symfony 2

Requisitos de Disponibilidad:

RNF9. La aplicación posibilita la gestión de reportes en SIGDAT.

RNF10. Debe ser lo más representativa posible y se podrá acceder a ella siempre que se tenga acceso al SIGDAT.

Requisitos de Eficiencia

RNF12. Cantidad de Usuarios conectados concurrentemente. El sistema debe permitir la concurrencia de al menos 50 usuarios en línea.

2.3 Modelo de Casos de Uso del Sistema

Representa las relaciones que existen entre el actor y los casos de uso. El actor es un individuo, entidad o máquina, con los que el sistema interactúa y los casos de uso son porciones de funcionalidades que el sistema ofrece para aportarles un resultado de valor a los actores que interactúen con él (33). En el Módulo de Reportes se identificó el actor Usuario y los casos de uso (Gestionar Reporte, Realizar Diseño, Adicionar Modelo de Datos, Crear Estilo al Diseño y Gestionar Reporte Publicado). A continuación se presenta el modelo de casos de uso.

2.3.1. Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso permite representar de la manera en que el actor (Usuario), interactúa con el sistema que se encuentra en desarrollo. Los casos de uso no deben encontrarse aislados sin relación con el actor o con otro caso de uso pues esto reflejaría un error en el diagrama. Los requisitos funcionales se encuentran agrupados en los cinco casos de uso representados (ver figura 2).

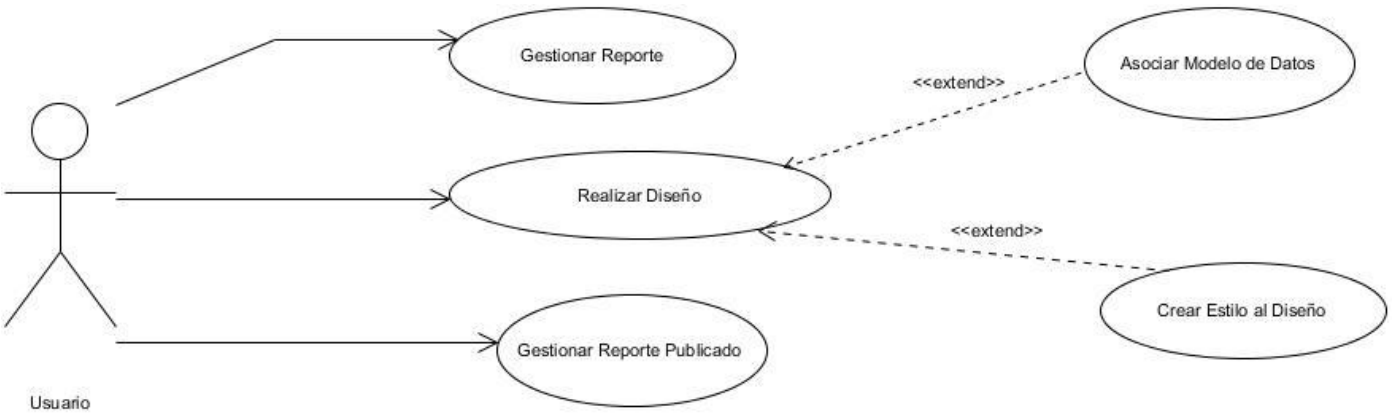


Fig. 2 Diagrama de Casos de Uso

- **Gestionar Reporte:** permite adicionar, modificar, listar y eliminar un reporte del sistema además de permitir la actualización de la lista de reportes en la base de datos.

- **Realizar Diseño:** Permite adicionar, modificar, eliminar, alinear y duplicar componentes del diseño del reporte. También guardar cambios, crear una consulta SQL, mostrar una vista previa del diseño del reporte y cerrar el área de diseño del reporte.
- **Gestionar Reporte Publicado:** Permite publicar, modificar, listar, eliminar y exportar un reporte publicado del sistema. Además de permitir la actualización de la lista de reportes publicados en la base de datos.
- **Asociar Modelo de Datos:** Permite adicionar un modelo de datos a componentes del diseño del reporte que lo requieran.
- **Crear Estilo al Diseño:** Permite adicionar, modificar, listar y eliminar estilos existentes en el modelo del diseño del reporte.

2.3.2. Patrones de Casos de Uso del Sistema

Los patrones de casos de uso describen el uso del sistema, como debería ser estructurado y cómo este interactúa con el usuario. Capturan mejores prácticas para modelar los casos de uso (34).

Extensión Concreta

Este patrón consiste en una relación de extensión entre dos casos de uso y es aplicable cuando un flujo puede extender del flujo de otro caso de uso. Se utiliza el patrón de casos de uso extensión concreta, para modelar la relación de extensión que existe entre el caso de uso Realizar diseño y Adicionar Modelo de Datos (34).

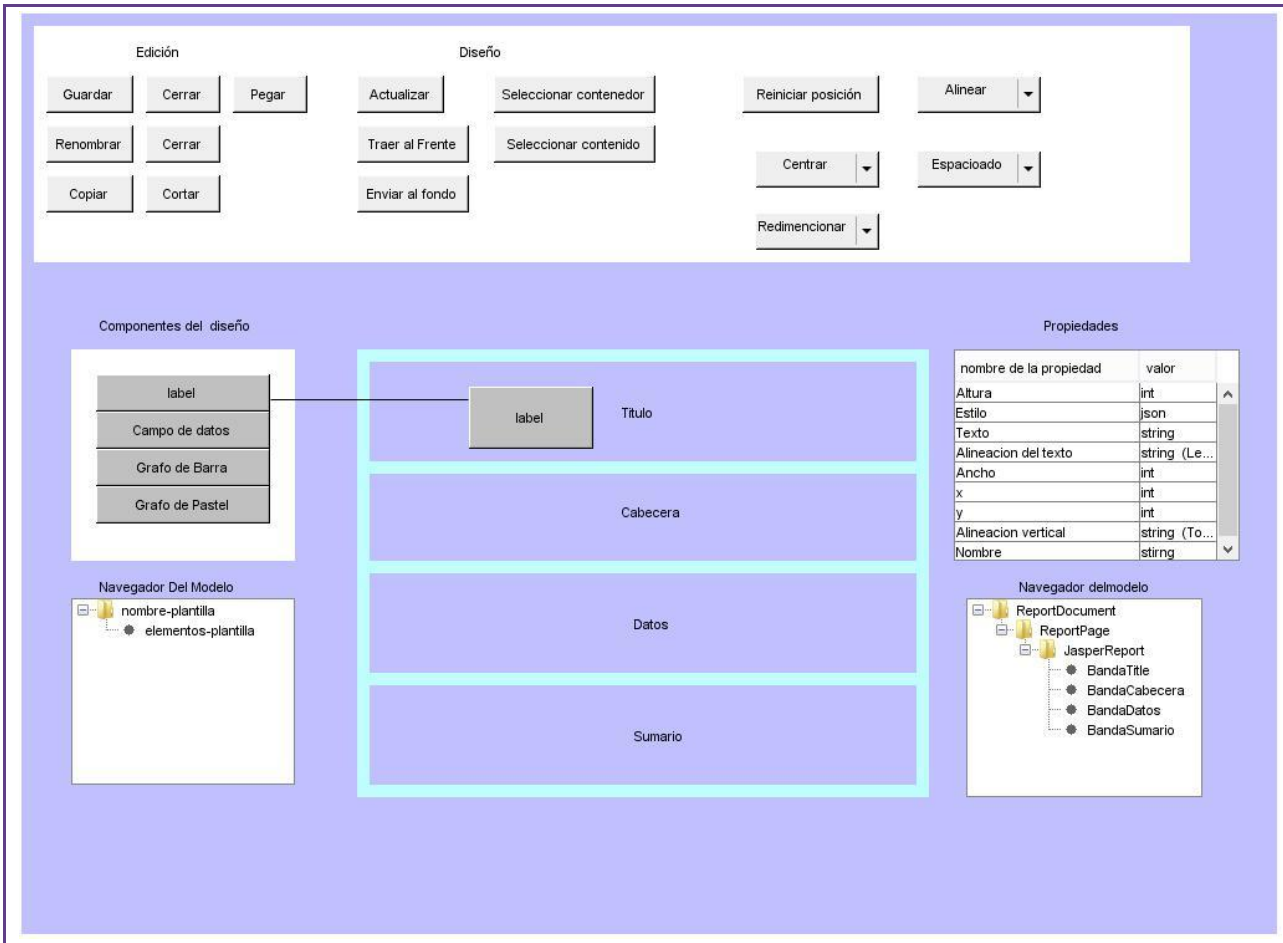
CRUD Completo

Descripción: El patrón CRUD Completo consiste en un caso de uso para administrar la información (CRUD Información). Permite modelar las diferentes operaciones para administrar una entidad de información. Este patrón se usa cuando todas las operaciones contribuyen al mismo valor de negocio. Se utiliza el patrón de casos de uso CRUD completo, para modelar los casos de uso Gestionar Reporte, Realizar diseño y Gestionar Reporte Publicado (34).

2.3.3. Descripción textual del Caso de Uso: Realizar Diseño

Tabla 2 Descripción de Caso de Uso

Caso de Uso:	Realizar Diseño.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario está autenticado en SIGDAT y presiona la pestaña Reportes.
Precondiciones:	Tener acceso al SIGDAT.
Referencias	RF6, RF7, RF8, RF9, RF10, RF11, RF12, RF13, RF23
Prioridad	Alta
Flujo Normal de Eventos	
Sección 1 "Adicionar componentes de diseño al reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario escoge un componente de la paleta de componentes y lo arrastra al área de diseño.	4. El sistema actualiza el área de diseño.
Prototipo de Interfaz	



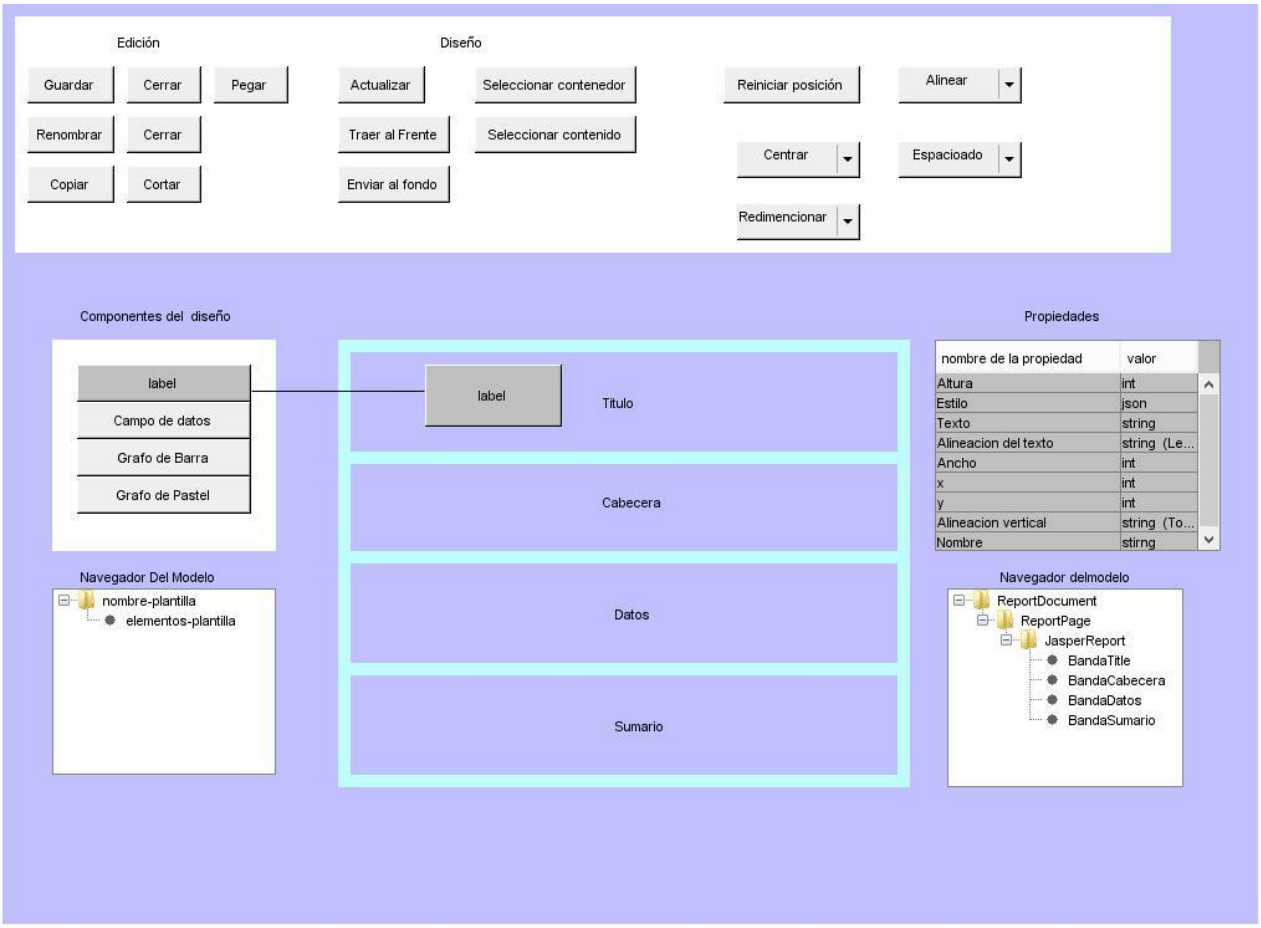
Poscondiciones	El componente queda adicionado al área de diseño.

Sección 2 “Modificar componentes de diseño del reporte”

Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.

3. El usuario selecciona un componente del área de diseño.	4. El sistema activa el menú que muestra las opciones de modificación a los componentes del área de diseño.
5. El usuario selecciona la modificación deseada, como son redimensionar y espaciado.	6. El sistema actualiza el área de diseño.

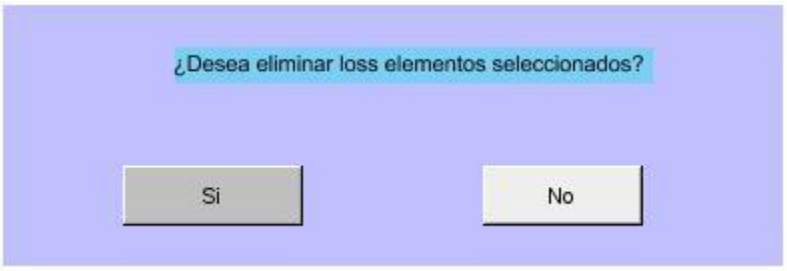
Prototipo de Interfaz



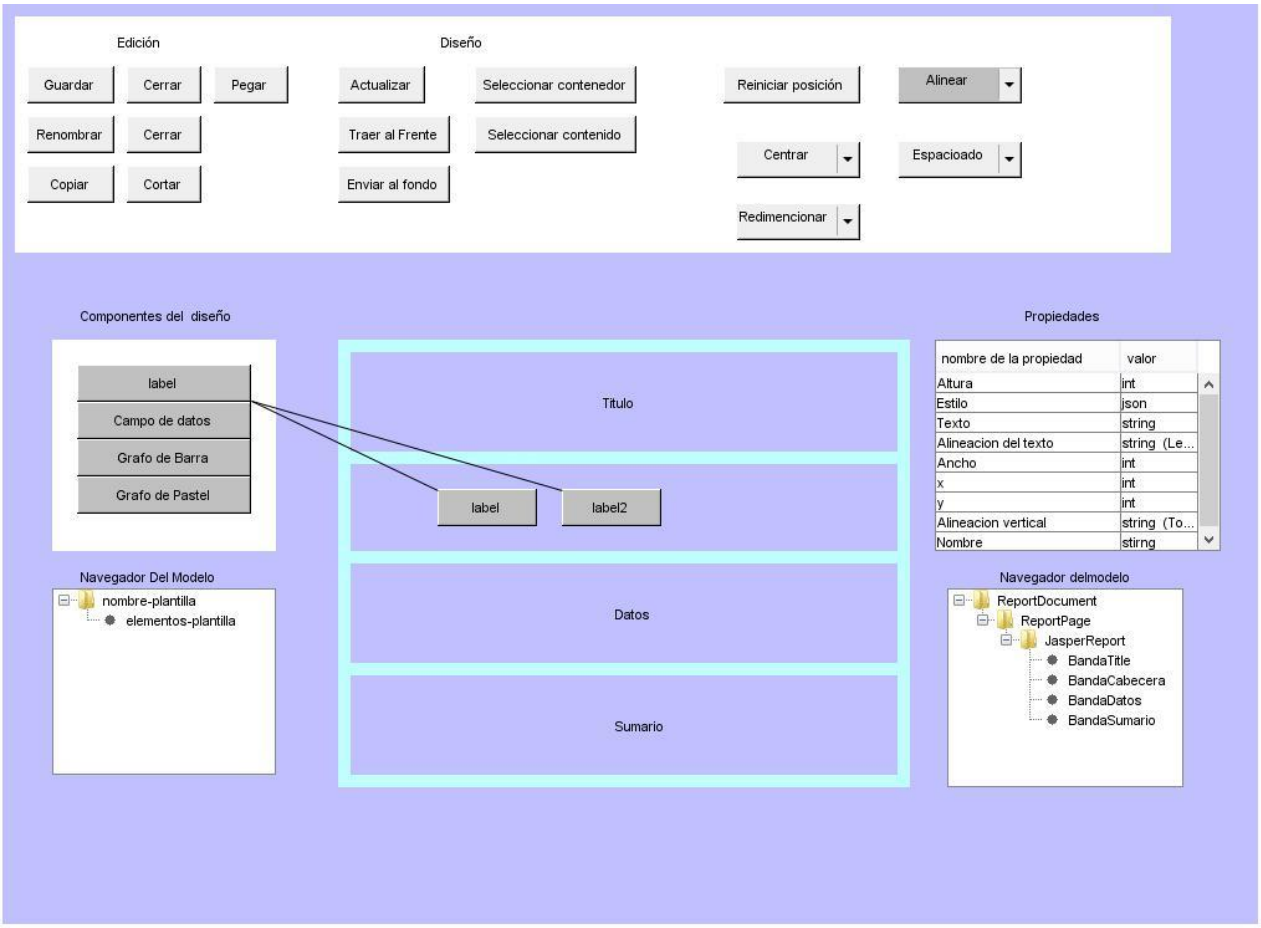
Poscondiciones	El componente queda modificado en el área de diseño.

Sección 3 “Eliminar componentes de diseño del reporte”

Acción del Actor	**Respuesta del Sistema**

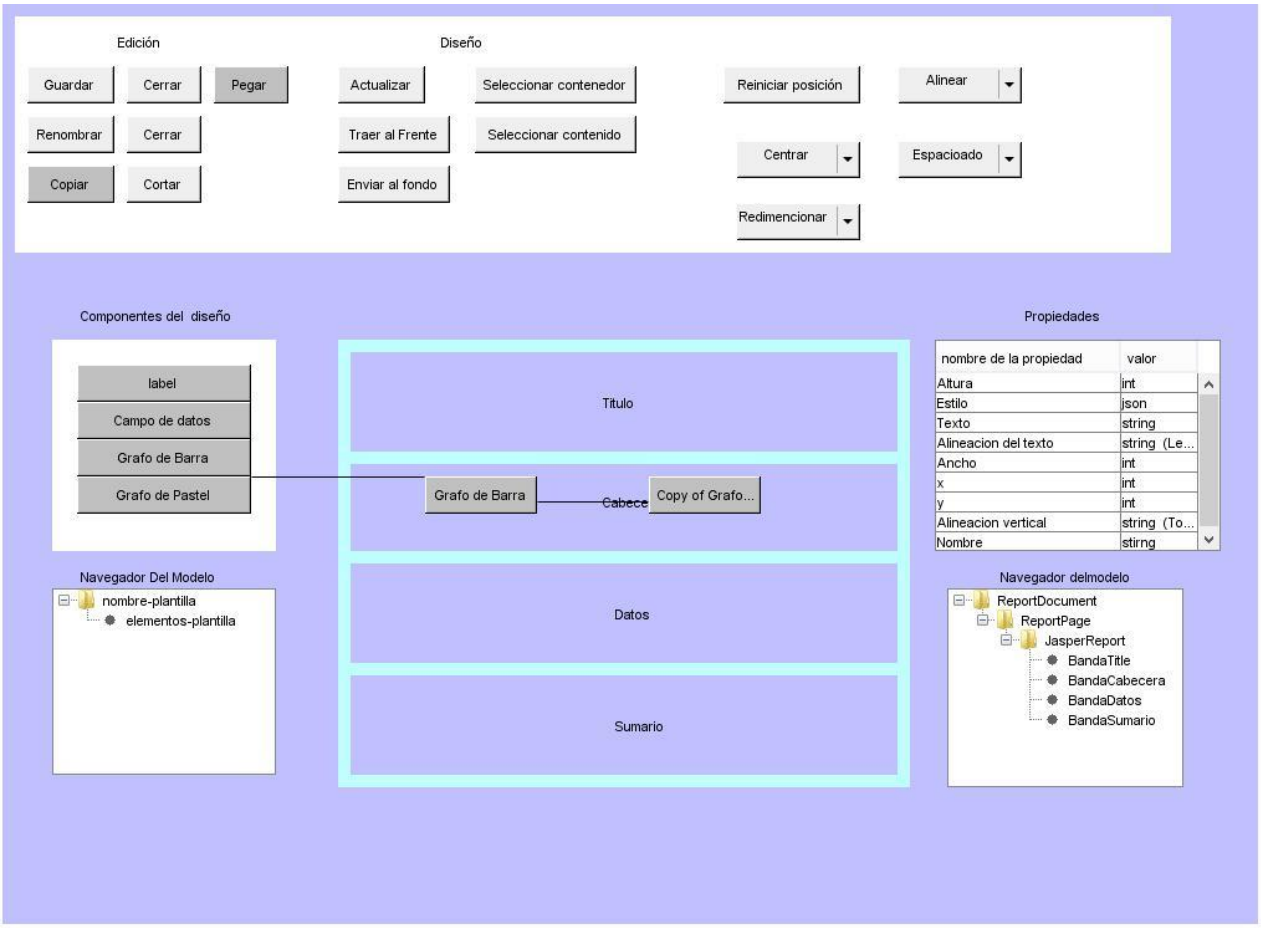
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario selecciona un componente del área de diseño y selecciona la opción eliminar.	4. El sistema muestra un mensaje de confirmación con la opción **"SI"** y **"No"**.
4. El usuario selecciona la opción deseada.	5. Si el usuario selecciona la opción **"Si"** el sistema elimina el componente del área de diseño. Si el usuario selecciona la opción **"No"**, ver flujo alterno.
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	a. Si el usuario selecciona la opción **"No"** el sistema vuelve al área de diseño.
Prototipo de Interfaz 	
Poscondiciones	El componente queda eliminado del área de diseño.
Sección 4 "Alinear componentes de diseño del reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario selecciona uno o más componentes del área de diseño a alinear.	4. El sistema muestra un menú con opciones de alineación, a la izquierda a la derecha, al centro, superior.

Prototipo de Interfaz

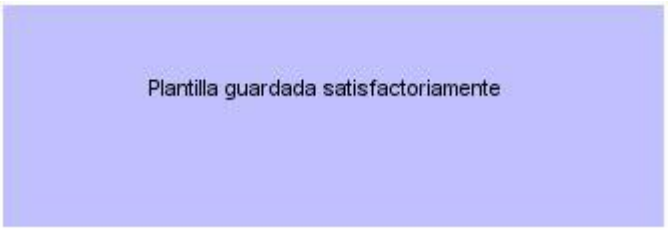


Poscondiciones	Los elementos quedan alineados de la manera deseada.
Sección 5 "Duplicar componentes de diseño del reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario selecciona un componente del área de diseño, el cual desea duplicar.	4. El sistema actualiza el área de diseño.

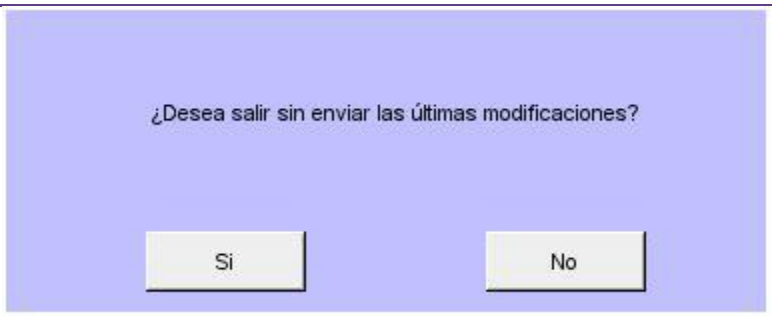
Prototipo de la Interfaz



Poscondiciones	Queda duplicado el componente en el área de diseño.
Sección 6 "Guardar cambios sobre el diseño del reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.

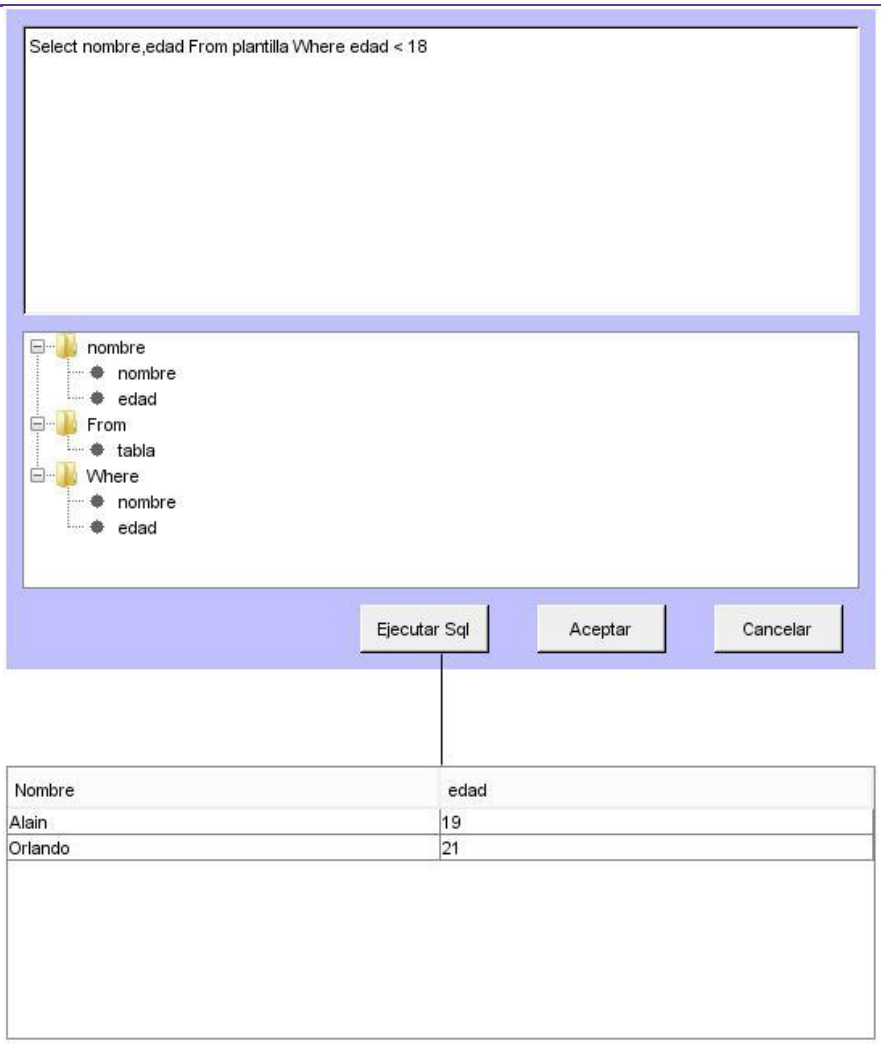
3. El usuario presiona el botón Guardar.	4. El sistema muestra un mensaje diciendo que la acción se realizó satisfactoriamente.
Prototipo de la Interfaz 	
Poscondiciones	El diseño queda guardado.
Sección 7 "Vista previa del diseño del Reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados a ella y selecciona la opción diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario selecciona el botón Vista Previa.	4. El sistema muestra una ventana con el reporte diseñado.
5. El usuario selecciona la opción cerrar	6. Se cierra la ventana y se vuelve al área de diseño.
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	a. Si el usuario selecciona la opción **"No"** el sistema vuelve al área de diseño.
Prototipo de la Interfaz	

Sección 8 "Cerrar el área del diseño del reporte"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario presiona el botón Cerrar.	4. El sistema muestra un mensaje de confirmación con la opción **"SI"** y **"No"**.
5. El usuario selecciona la opción deseada.	6. Si el usuario selecciona la opción **"Si"** el sistema cierra el área de diseño. Si el usuario selecciona la opción **"No"**, ver flujo alterno.
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	7. Si el usuario selecciona la opción **"No"** el sistema vuelve al área de diseño.
Prototipo de la Interfaz	



Poscondiciones	El área de diseño se cierra.

Sección 9 “Crear Consulta SQL”	
Acción de Actor	**Respuesta del Sistema**
1. El usuario presiona la pestaña Reportes, selecciona una plantilla publicada de la que se muestran todos los reportes relacionados con ella y escoge nuevo reporte o diseñar reporte.	2. El sistema muestra la interfaz del Diseñador de reporte.
3. El usuario presiona en la paleta de propiedades del área de diseño la propiedad Consulta SQL.	4. El sistema muestra la interfaz para realizar la consulta. Con las opciones **Ejecutar Consulta**, **Aceptar** y **Cancelar**.
5. El usuario selecciona la opción deseada.	6. Si el usuario selecciona la opción **Ejecutar Consulta,** el sistema muestra una interfaz con la respuesta de la petición. 7. Si el usuario selecciona la opción **Aceptar,** el sistema guarda la consulta SQL en la propiedad que corresponde a la consulta en el área de diseño. 8. Si el usuario selecciona la opción **Cancelar,** el sistema cierra la interfaz
Prototipo de Interfaz	



Poscondiciones	Queda guardada la consulta referente al diseño del reporte.

2.4 Modelo de Diseño

Es una simplificación utilizada para comprender mejor el problema. Encamina el proceso de producción. Brinda una solución de software que satisface los requisitos para darle solución al sistema.

El modelo de diseño está compuesto por:

- Diagramas de clases del diseño y diagramas de interacción (colaboración y/o secuencia) del diseño.
- Paquetes y subsistemas de diseño representados en una jerarquía y una breve descripción de ellos.
- Clases, interfaces y relaciones contenidas en los paquetes y una breve descripción de ellos (35).

2.4.1. Diagrama de Paquetes

Muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido (36). (Ver figura 3).

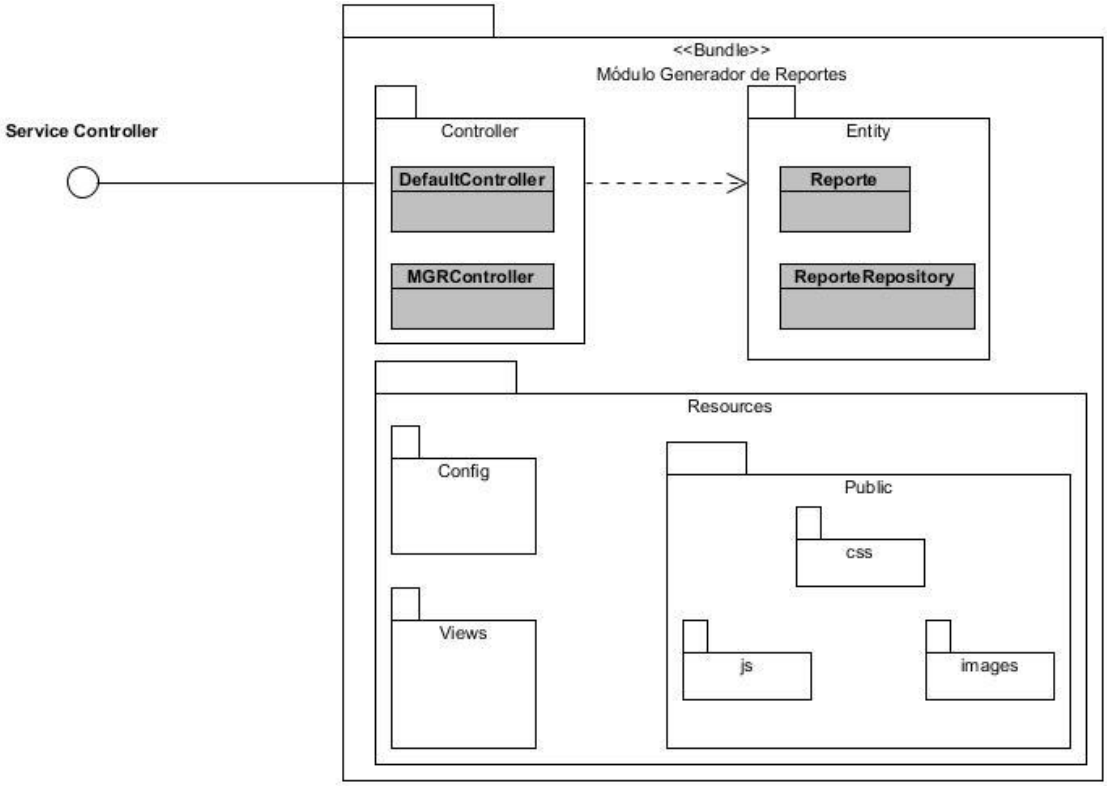


Fig. 3 Diagrama de Paquetes

El MGRBundle correspondiente a la aplicación estará compuesto por las carpetas Controller, Entity y Resources. En el Controller se almacenarán todos los controladores del sistema, los que harán uso del repositorio y entidad en dependencia de las acciones que se realicen. El repositorio estará contenido dentro de la carpeta Entity y se encargarán de organizar las sentencias del lenguaje de consulta Doctrine que utiliza el lenguaje DQL (del inglés *Doctrine Query Language*) para realizar consultas a la BD de una entidad en cuestión.

La entidad se almacenará dentro de la carpeta Entity y se encarga de la abstracción de la lógica relacionada con los datos. Hace que las vistas y las acciones sean independientes del tipo de gestor de base de datos. Las entidad con la que se trabajará es Reporte, en dicha clase se encuentran sus atributos y algunos métodos para el trabajo con ellos.

Otra de las carpetas con las que cuenta el proyecto MGRBundle es Resources la cual está compuesta por las carpetas view, config, y public. Dentro de la carpeta view se encuentra la plantilla Twig, encargada de mostrar los diferentes componentes del módulo. En la carpeta public se encuentran diferentes recursos como hojas de estilos, imágenes y archivos JavaScript, cada uno en su carpeta correspondiente. Para configurar los diferentes servicios que se serán utilizados en el sistema se contará con el archivo services.yml el cual se ubicará dentro de la carpeta config.

La vista del módulo se implementará utilizando Ext JS 3.4, es por ello que es necesario definir una organización para los componentes como se muestra (ver figura 4).

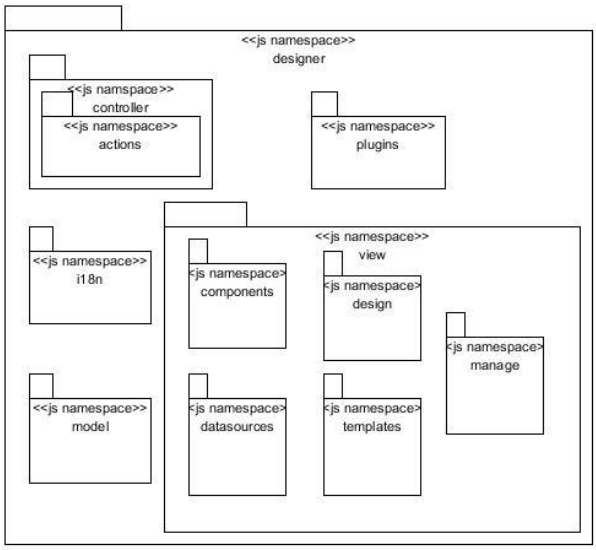


Fig. 4 Organización de Componentes

Las operaciones a realizar en el sistema se encontrarán almacenadas en la carpeta actions. Los stores y record, encargados de realizar las diferentes peticiones a los controladores en la parte del servidor, se encontrarán en la carpeta model. Las vistas del sistema se almacenarán en la carpeta view, según el nombre de las vistas así será el de su carpeta correspondiente.

2.4.2. Diagrama de clases del diseño

Contiene las definiciones de las entidades de software, describe gráficamente las especificaciones de las clases de software y las interfaces en una aplicación. Posee información acerca de clases, asociaciones, atributos, interfaces, métodos, navegabilidad y dependencias (37).

2.4.3. Diagrama de clases del diseño del caso de uso Realizar Diseño

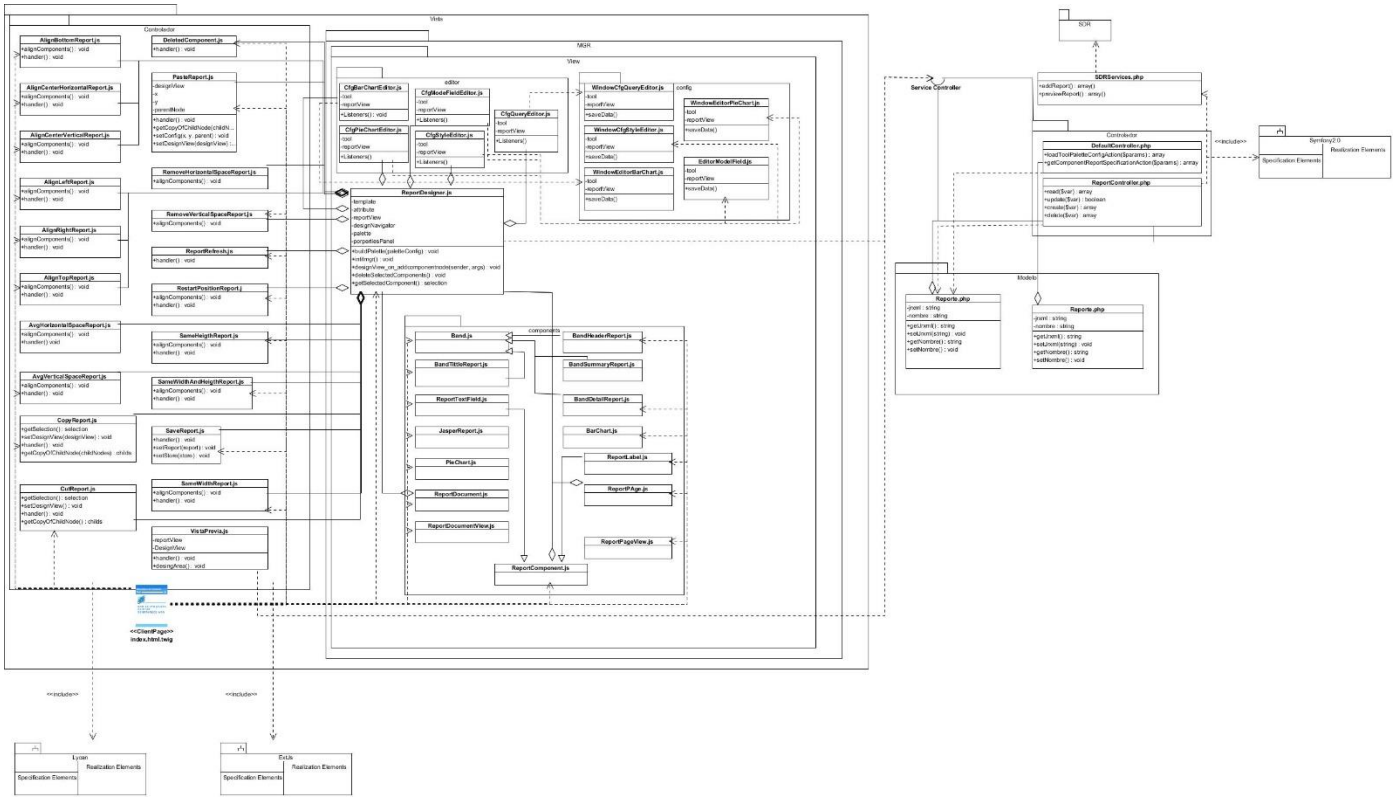


Fig. 5 Diagrama de Clases del Diseño

Descripción del diagrama de clases del diseño del caso de uso Realizar Diseño.

Las clases que se muestran a la izquierda (ver figura 5) representan las clases de la vista, las de la derecha

superior la controladora y la de la derecha inferior las del modelo, evidenciándose así el patrón Modelo-Vista-Controlador. La clase *ReportDesigner* es la encargada del área de diseño en la cual se adicionarán los componentes del diseño como son *label*, *textfield*, campos especiales y gráficos, los cuales se emplearán en la confección del diseño del reporte. Esta área de diseño está conformada por la clase *ReportDocument*, la cual contiene *ReportePage*. *ReportComponent* es la clase de la cual heredan los componentes y esta hereda de *Lycan*. La clase *ReportPage* es la que representa el panel dentro del área de diseño en la cual se adicionarán los componentes, también es un tipo de componente. Los componentes campos especiales y gráficos se les asocian un modelo de datos lo cual se realiza a través de una interfaz gráfica, las clases encargadas de la creación y manejo de dichas interfaces son *CfgModelFieldEditor*.js para el componente campos especiales, *CfgBarChartEditor*.js y *CfgPieChartEditor*.js para los gráficos de barra y pastel respectivamente. EL área de diseño también presenta una serie de acciones, las cuales heredan de la clase *designContextAction* de Lycan como son copiar, pegar, renombrar, cortar, cerrar, vista previa, guardar. Las acciones de alinear heredan de la clase *abstractAlignAction* de Lycan a las cuales se les pasa el área de diseño mediante el método *Reportmgr*(),este es el encargado de tomar el área de diseño y realizar las acciones a los componentes existentes en dicha área, en caso de no existir ningún componente se muestran ocultas; el método para activar dichas acciones es *TestEnableCondition*(), este comprueba que existan componentes seleccionados mediante el método del área de reporte *GetSelectedComponents*(), que es el encargado de devolver los componentes seleccionados. La acción eliminar invoca al método *DeleteSelectedComponent*() de la clase *ReportDesigner* que es el encargado de eliminar el componente seleccionado. La acción guardar invoca al método getReport() el cual a su vez hace uso del método *reloadReport*(), que es el encargado de modificar atributo diseño del reporte, siendo este un objeto de java (json) y lo guarda en la base de datos en la tabla Reporte en el campo *disennoreporte* como un *string,* convirtiéndolo con la función Json_*encode*() de php en el controlador. Las acciones alinear para que se activen deben estar al menos dos componentes seleccionados, esto ejecuta la acción mediante el método *align*() que se redefine para cada una de las clases correspondientes a las acciones alinear como son: *AlignTop*, *AlignBotton*, *AlignLeft*, *AlignRight*, entre otras descritas en el diagrama. El método *LoadToolPaletteConfig* de la clase *DefaultController* es el encargado de cargar la especificación de cada uno de los componentes. La clase controladora *ReportController*.php es la encargada de la comunicación entre el módulo y el SDR, servicio utilizado para mostrar una vista previa del diseño del reporte por la acción de mismo nombre.

2.5 Patrones de Diseño

Patrones GRASP

- **Controlador**

El uso de este patrón se manifiesta en los controladores utilizados por el sistema tales como MGRController y DefaultController, estos se encargan de interactuar con el modelo y enviar una respuesta a la vista.

- **Experto**

Se evidencia en la relación de las clases *GridReportePublicados.js* y *ReportPublisherTemplateStore.js*, indicando que la respondabilidad de acceder a los reportes publicados recae sobre GridReportPublicados.js, pues es la clase que contiene la información para realizar esta acción (ver figura 6).

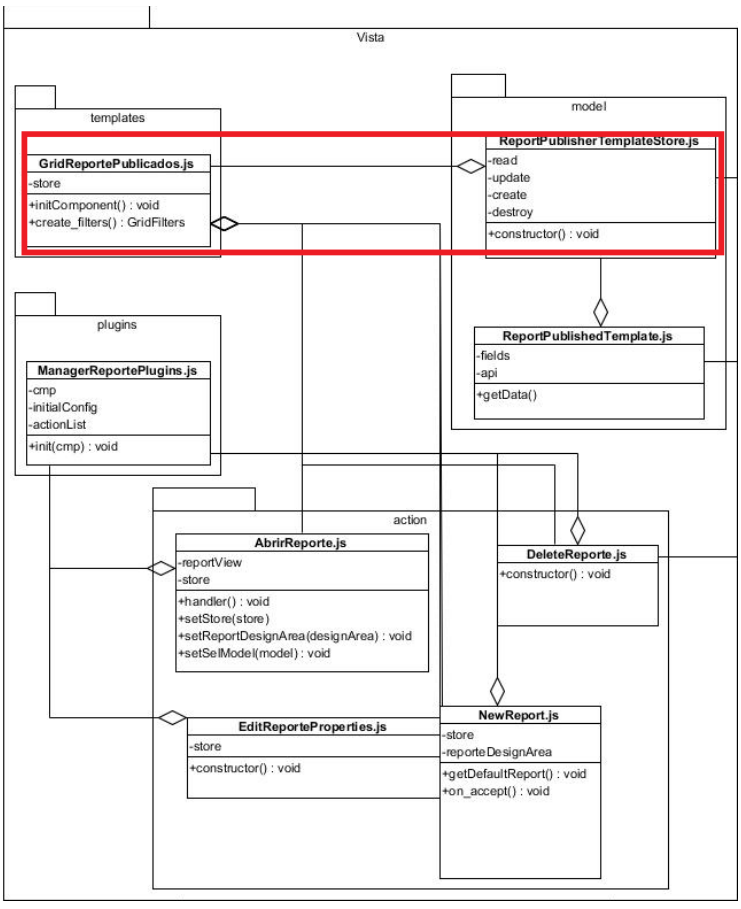


Fig. 6 Evidencia del patrón experto

- **Creador**

Se utiliza cuando se crea un nuevo reporte, donde el controlador de los reportes **MGRController** es el encargado de crear una instancia de la entidad Reporte, esta asigna atributos al nuevo reporte (ver figura 7).

```
$reporte=$var["items"];
$now = new \DateTime("NOW");
$update= new Reporte();
$em=$this->getDoctrine()->getEntityManager();
$plantilla=$em->getRepository("DesignerBundle:PublishedSurveyTemplate")
$update->setPlantillasPublicadas($plantilla);
$update->setNombreReporte($reporte["nombrereporte"]);
$update->setDescripcionReporte($reporte["descripcionreporte"]);
$update->setDisennoReporte(json_encode($reporte['disennoreporte']));
$update->setModeloReporte( json_encode($reporte["modeloreporte"]));
$update->setFechaActualizacionReporte($now);
$update->setFechaCreacionReporte($now);
$update->setEstadoReporte($reporte["estadoreporte"]);
```

Fig. 7 Evidencia del patrón creador

- **Alta cohesión**

Se ve evidenciado el uso de este patrón en la clase del modelo Reporte, pues posee la estructura necesaria para almacenar la información de forma coherente y de acuerdo a la responsabilidad que tenga asignada cada clase logrando una mayor capacidad de reutilización.

- **Bajo acoplamiento**

En la solución se observa el uso de este patrón en la clase Reporte, perteneciente al modelo, pues esta es una clase independiente que reduce el impacto de los cambios de otros componentes y fácil de reutilizar.

Patrones GOF:

- **Fachada**

Este patrón se evidencia en la clase MGR.js, la cual funciona como interfaz principal del módulo y a la vez

es la responsable de mandar a crear las demás interfaces de acuerdo a la acción que el usuario realice.

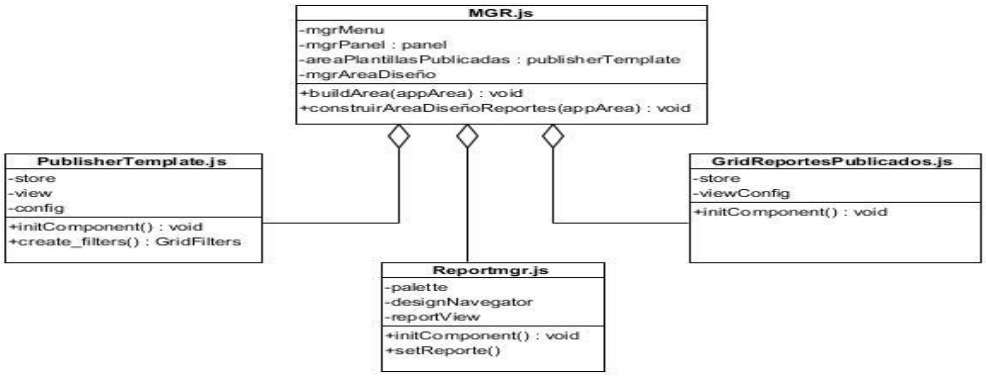


Fig. 8 Evidencia del patrón fachada

- **Command**

Permite encapsular la funcionalidad deseada en un objeto reutilizable. Define una clase por cada tarea que implementa una interfaz común. Se ve evidenciado en la clase NewReport.js, ReportDetail.js y FormReporte.js, puesto que ReportDetail.js es una clase común que es mostrada por dos acciones (ver figura 9).

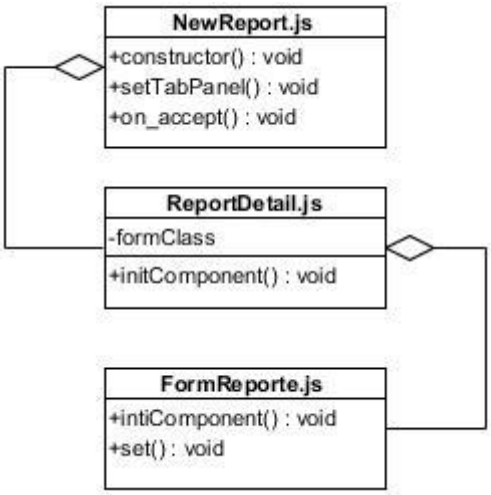


Fig. 9 Evidencia del patrón Command

2.6 Diagramas de Interacción

Se utilizan para modelar los aspectos dinámicos de los sistemas. Muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los diagramas de interacción están conformados por los diagramas de secuencia y los diagramas de colaboración. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Estos diagramas no son solo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa (38).

2.6.1 Diagrama de Secuencia

El diagrama de secuencia describe la dinámica del sistema. Muestra la secuencia de acontecimientos entre los actores. En la mayoría de los casos que se intentan modelar, la dinámica completa se representa utilizando diversos diagramas de secuencia, cada uno de ellos vinculado a una subfunción del sistema. El objetivo que se persigue con estos diagramas es identificar los acontecimientos y operaciones que resuelven un requisito funcional. La creación de estos diagramas es posterior a la descripción de los casos de uso. En resumen, un diagrama de secuencia muestra, para un escenario particular de un caso de uso:

- Los acontecimientos generados por los actores.
- Su orden.
- Los acontecimientos internos en el sistema que resultan de la invocación (39).

2.6.2 Diagrama de Secuencia del Caso de Uso Realizar Diseño sección Eliminar Componentes de Diseño del Reporte

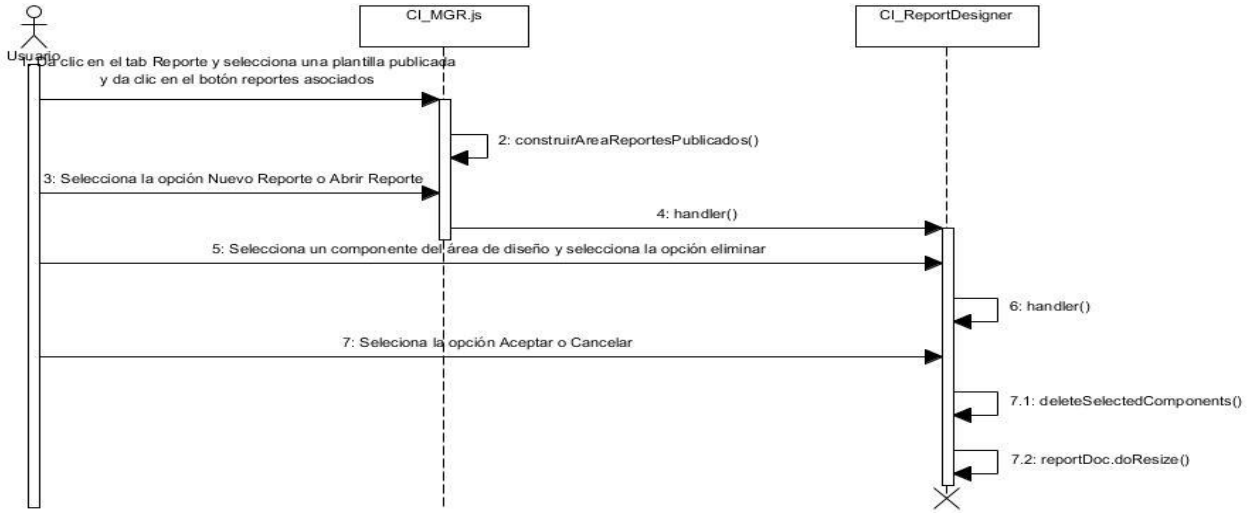


Fig. 10 Diagrama de Secuencia

En el diagrama de secuencia (ver figura 10) el actor usuario presiona la pestaña reportes y esta muestra un listado con las plantillas publicadas. El usuario selecciona una plantilla y se activa el botón reportes asociados, el cual muestra los reportes asociados a dicha plantilla mediante el método *handler*() del botón, el cual activa el *grid* de reportes publicados y este mediante los métodos de su api lista los reportes asociados a la planilla seleccionada. El usuario selecciona la acción abrir reporte, el sistema muestra el *tab* del área de reportes publicados donde se muestran un grupo de acciones en el menú, las cuales se activan, mediante el método *checkEnableRequirements*() si existe algún componente en el área de diseño. Para eliminar un componente se selecciona el componente a eliminar en el área de diseño, invocándose el método *deleteSelectedComponents*(), el cual antes de eliminar el componente seleccionado invoca al método *getSelectedComponent*() y elimina el componente.

2.7 Modelo de Despliegue

Constituye una representación física de las relaciones que existen entre los componentes de hardware y software de un sistema. Permite a los usuarios obtener un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Se compone por nodos, dispositivos y asociaciones (ver figura 11).

Nodos: Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.

Dispositivos: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo (40).

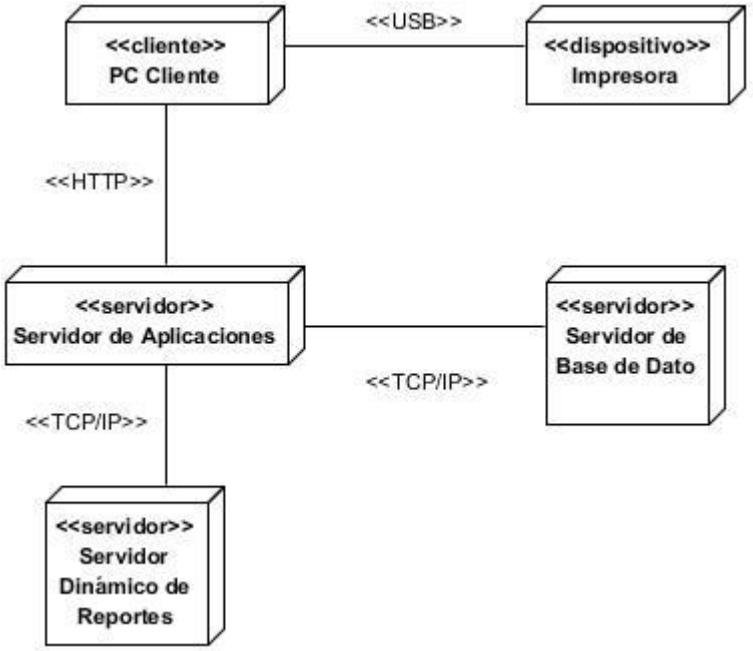


Fig. 11 Diagrama de Despliegue

Descripción del Diagrama de Despliegue

El sistema quedará desplegado de la siguiente manera. La aplicación se montará en un servidor el cual tendrá acceso mediante los protolos TCP/IP a dos servidores, el Servidor de Base de Dato y el Servidor Dinámico de Reportes. El usuario podrá acceder al sistema mediante una Computardora por el protocolo HTTP y si lo desea puede conectar una Impresora a su máquina permitiéndole realizar impresiones de los reportes realizados y previamente exportados a los formatos de su selección (Pdf,Word,Excel).

2.8 Conclusiones parciales del capítulo

En el capítulo se realizó una representación visual de los conceptos del mundo real relacionados con el dominio donde se desarrolla el Módulo de Reportes. Se realizó el levantamiento de los requisitos funcionales

y no funcionales para garantizar el funcionamiento correcto de la aplicación. Posteriormente se reflejó el diagrama de casos de uso para representar la relación existente entre el actor del sistema y los cinco casos de uso que quedaron identificados. Se ilustró el diagrama de clases del diseño del caso de uso Realizar Diseño para garantizar una representación de la estructura del sistema. Se representó el diagrama de secuencia para la sección del caso de uso Eliminar Componentes de Diseño del Reporte. También se analizaron los patrones de diseño para identificar los que se aplican al módulo, además de realizar el diagrama de despliegue para describir la distribución física del sistema.

Capítulo 3: Implementación y prueba del Módulo de Reportes para SIGDAT.

Introducción

En este capítulo se realizará la implementación del módulo, así como las pruebas funcionales de este para garantizar que se corresponda el sistema desarrollado con los requisitos de software definidos. Se realiza el modelo de implementación que muestra el diseño de la solución, así como el diagrama de componentes de la extensión. Se especifican además las pruebas realizadas a la extensión, con el objetivo de comprobar sus funcionalidades en los diferentes escenarios y de esta forma verificar en todos los casos que los resultados de las pruebas sean los esperados.

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 encuentran datos, clases, 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 (41).

3.2 Diagrama de Componentes

El diagrama de componentes es usado para estructurar el modelo de implementación. Muestra como el sistema está dividido en componentes y las dependencias entre ellos. Proveen una vista arquitectónica de alto nivel del sistema. Ayuda a los desarrolladores a visualizar el camino de la implementación. Permite tomar decisiones respecto a las tareas de implementación. Un componente representa un elemento físico que forma parte del sistema, o sea una unidad autónoma que provee una o más interfaces, se pueden representar por archivos, clases, código fuente más cabecera, librerías compartidas, ejecutables y paquetes (42).

Seguidamente se expone el diagrama de componentes del Caso de Uso Realizar Diseño el cual cuenta con tres paquetes de implementación básicos. Estos paquetes son:

- Paquete de Clases Vista, que agrupa los componentes que permiten la interacción directa con el usuario final del sistema, mostrando y recogiendo información.
- Paquete de Clases Controlador, que agrupa las clases que manipulan los eventos del usuario y se apoya en el Paquete de Clases Modelo para dar respuesta a las peticiones de las vistas.
- Paquete de Clases Modelo, que agrupa las clases que interactúan con la base de datos.

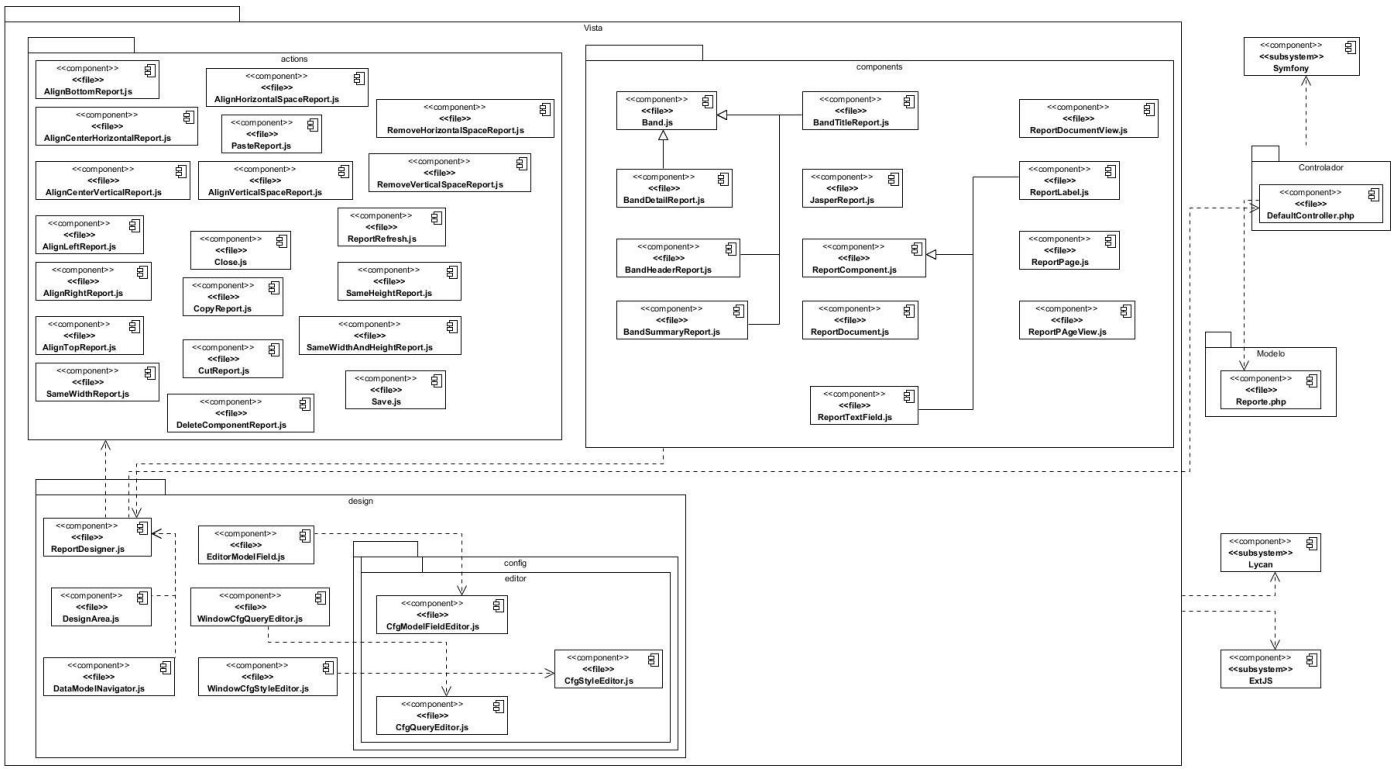


Fig. 12 Diagrama de Componentes

3.3 Código Fuente

El código fuente es texto simple capaz de ser leído y comprendido por cualquier programador que conozca el idioma utilizado. En el están escritas las instrucciones que deberá realizar la computadora, según las sintáxis de un lenguaje de programación. El lenguaje utilizado para la redacción del código fuente es lo que se denomina lenguaje de alto nivel, el programador puede expresarse con cierta facilidad por ser el más próximo al lenguaje natural. El código fuente no es ejecutable directamente por la computadora, la máquina solo comprende el lenguaje de máquina, este desentendimiento lo resuelven programas o herramientas como compiladores, ensambladores e intérpretes (43).

3.3.1 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Los programadores deben implementar un estándar de forma prudente, tendiendo 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 (44).

Estilo de codificación utilizado

- La nomenclatura utilizada será CamelCase, las variables y métodos estarán escritos con el estándar lowerCamelCase y la declaración de las clases con el estándar UpperCamelCase.
- Se utilizarán llaves de apertura y cierre para aumentar la legibilidad y disminuir la probabilidad de errores. Además las etiquetas PHP deben ser completas (<? php ?>)... no reducidas (<? ?>).
- Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.
- Las variables locales serán inicializadas en el momento de su declaración o en caso contrario cuando su valor inicial dependa de una llamada a otro método en determinado momento de ejecución de un método.

Fragmento de Código Fuente

Se muestra a continuación un fragmento de código fuente del módulo de reportes para SIGDAT, evidenciándose el uso del estilo de codificación definido. Este fragmento es extraído del método construirAreaReportesDisennados de la clase MGR.js.

El método construirAreaReportesDisennados(), es el encargado de crear el área de reportes diseñados, donde se muestra el listado de los reportes que han sido diseñados. Este método es el encargado también de administrar las acciones que se realizan sobre los reportes, como son diseñar reporte, nuevo reporte, eliminar reporte, propiedades del reporte y publicar reporte.

```
construirAreaReportesDisennados: function(appArea) {
    var pluginReportesDesigns = new dydamsi.mgr.plugins.ManageReportePlugin({
        createAction: this.getAction('newReporte'),
        searchAction: this.getAction('searchReportes'),
        propertiesAction: this.getAction('editReporteProperties'),
        openAction: this.getAction('abrirreporte'),
        publishAction: this.getAction('publicarReporte'),
        destroyAction: this.getAction('deleteReportes'),
    });

    // area de reportes disennados
    this.areaReportesDisennados = new dydamsi.mgr.view.templates.GridReportesDisennados({
        id: 'area-reportes-disennados',
        itemId: 'area-reportes-disennados',
        title: this.i18n.Labels.REPORTE_DESIGN,
        openAction: this.getAction('abrirreporte'),
        plugins: [pluginReportesDesigns]
    });

    this.areaReportesDisennados.disable();
},
```

Fig. 13 Ejemplo del código fuente

3.4 Pruebas de Software

El único instrumento adecuado para determinar el nivel de calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (45).

Las pruebas de software son aplicadas en diferentes momentos del ciclo de vida del software para demostrar hasta qué punto el software tiene un buen funcionamiento. Existen diferentes niveles de pruebas como desarrollador, independiente, unidad integración, sistema y aceptación. En el desarrollo de la fase de pruebas del módulo de reportes para SIGDAT se aplicarán las pruebas a nivel de desarrollador e integración para probar que el sistema cumpla con los requisitos funcionales previamente definidos en la fase de análisis.

3.4.1 Pruebas de desarrollador

Al ir incorporando nuevas líneas de código, aumenta la posibilidad de añadir nuevos errores al sistema. Durante la etapa de implementación del software, el desarrollador realiza un conjunto de pruebas con el objetivo de validar que el sistema funcione correctamente antes de avanzar a próximos niveles de pruebas. Para ello es necesario identificar qué tipos, métodos y herramientas de pruebas pueden ser adecuados aplicar en este primer momento de revisión del software.

Como tipo de prueba se aplicarán las pruebas funcionales. Tienen como objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. Es común que este tipo de pruebas sean desarrolladas por analistas con el apoyo de algunos usuarios finales (46). Están basadas en el análisis de los datos ya sean de entrada o salida. Se diseñarán y utilizarán los casos de prueba como herramienta de apoyo al proceso de revisión. Estos tienen el objetivo de detectar errores por lo que definen los datos de entrada a utilizar, el proceso que se debe seguir en el sistema así como el resultado esperado. Por cada caso de uso se debe realizar un caso de prueba dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. Para cada escenario se conformarán matrices de datos con el uso de valores válidos e inválidos, evidenciándose el uso de la técnica partición de equivalencia del método de prueba de caja negra.

Las pruebas de caja negra se centran en los requisitos funcionales, se llevan a cabo sobre la interfaz del software y se enfocan en las entradas y salidas y no en el código fuente (47).

Ejecución de las pruebas funcionales

Se realizaron cuatro iteraciones de pruebas funcionales y se aplicaron los cinco casos de prueba diseñados arrojando un total de nueve NC (No Conformidades). En la primera iteración se detectaron cuatro NC, de ellas una de código, dos de interfaz y una de ortografía. En la segunda iteración se detectaron tres NC, de ellas una de código, una de interfaz y una de ortografía. En la tercera iteración se detectaron dos NC, las dos de ortografía. En la cuarta iteración se comprobó que las NC de las iteraciones anteriores estuvieran resueltas y no se detectaron nuevas NC. Estas pruebas permitieron comprobar el correcto funcionamiento del módulo y la correcta validación de los campos, verificando que solo acepten los caracteres válidos. A continuación se muestra una imagen (ver figura 14) con el número de iteraciones realizadas y la cantidad de NC detectadas en cada iteración.

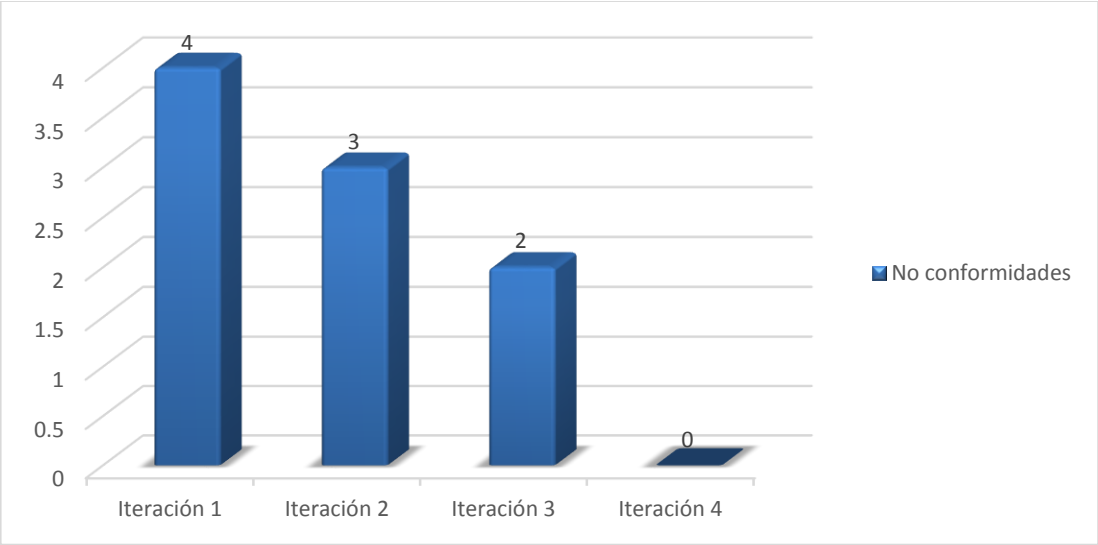


Fig. 14 Gráfica de no conformidades de las pruebas funcionales a nivel de desarrollador

3.4.2. Pruebas a nivel de Integración

Las pruebas de integración es la fase en pruebas en el que los módulos de software individuales se combinan y se ensayan como un grupo. Se produce después de las pruebas unitarias y antes de las pruebas de validación. Las pruebas de integración demuestran como los módulos de un sistema, que han sido probados como unidad, interactúan de manera correcta. Ofrece como salida el sistema integrado (46).

Una vez terminada la revisión del módulo a nivel de desarrollador se procede a la integración en la herramienta SIGDAT. Para evaluar el comportamiento del módulo una vez integrado se realizarán pruebas funcionales y de rendimiento. Para realizar las pruebas funcionales se mantiene la utilización del mismo tipo de prueba, método y herramienta utilizados en las pruebas a nivel de desarrollador. Estas pruebas tienen el objetivo de verificar si durante la integración en la herramienta SIGDAT las funcionalidades presentan alguna dificultad para su correcta ejecución.

El rendimiento de un sistema es un valor tanto o más importante que la propia funcionalidad del sistema. Se puede diseñar un software con una funcionalidad novedosa, desarrollado bajo las más estrictas pruebas funcionales y con la interfaz más intuitiva y amigable, pero si este no responde de manera adecuada en el

ambiente para el que ha sido diseñado, su calidad general será bastante peor de la esperada. Dentro de las pruebas de rendimiento, estas se pueden dividir a su vez en distintos tipos:

Pruebas de carga:

Su principal objetivo es comprobar el comportamiento del sistema frente a un uso intensivo del mismo, por ejemplo, un gran número de usuarios concurrentes. Durante las pruebas de carga, se pueden obtener tiempos de respuesta del sistema según el número de usuarios, carga de las máquinas y uso de memoria.

Pruebas de estrés:

Se utilizan en conjunto con las pruebas de carga. Estas consisten en aumentar el número de usuarios concurrentes o peticiones (en función del diseño del sistema) que se utilizarán en las pruebas de carga. Generalmente, este aumento se hace doblando en cada iteración el número de peticiones. Con estas pruebas se pueden obtener el máximo uso admisible por el sistema. También se podría determinar qué elemento es el que lo limita para estudiar una futura escalabilidad si es hardware o un rediseño si es software.

Pruebas de estabilidad:

Determinan la fiabilidad del sistema en uso continuo. Comprueban la correcta gestión de memoria, gestión de conexiones en la red, gestión de ficheros de log, consultas a BD con gran cantidad de datos. Si algo funciona 10 minutos no quiere decir que vaya a funcionar igual después de 10 días (48).

Para la ejecución de dichas pruebas se utilizará la herramienta de código abierto JMeter, desarrollada para realizar pruebas de rendimiento principalmente en aplicaciones web.

Ejecución de las pruebas

Se realizaron tres iteraciones de pruebas funcionales para garantizar una correcta integración del Módulo de Reportes con la herramienta SIGDAT. Se aplicaron los cinco casos de pruebas detectándose un total de seis NC. En la primera iteración se detectaron tres NC, de ellas dos de errores ortográficos y una de código. En la segunda iteración se detectaron tres NC, de ellas una de código y dos de interfaz. En la tercera

iteración se comprobó que las no conformidades detectadas en las etapas anteriores estuvieran resultas, por lo que no se detectaron nuevas NC. Se demuestra que el sistema está libre de errores (ver figura 15).

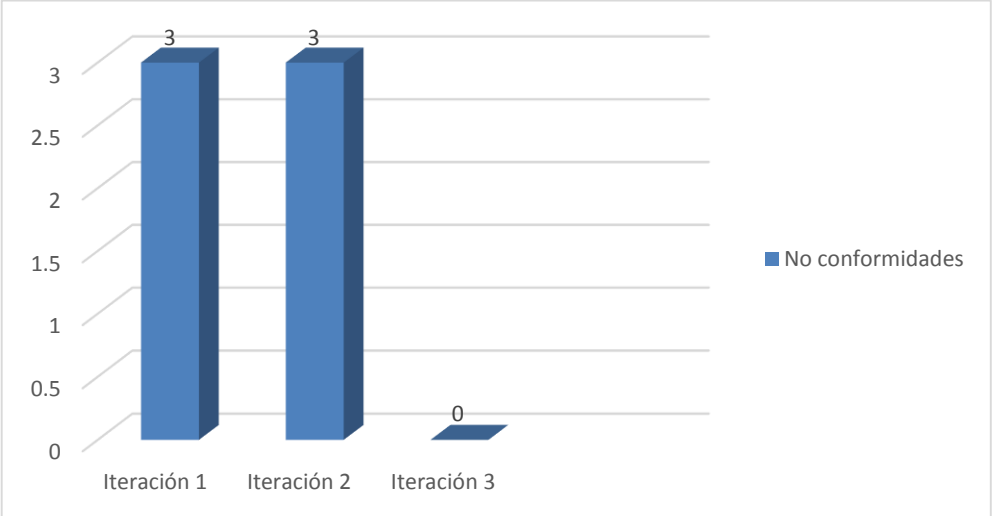


Fig. 15 Gráfica de no conformidades de las pruebas funcionales a nivel de integración

Se realizaron las pruebas de rendimiento con el uso de la herramienta JMeter 2.9, simulando peticiones a la aplicación de varios usuarios conectados concurrentemente. Para ello se pobló la BD y se midió el tiempo de respuesta de todas las funcionalidades haciendo énfasis en las de mayor nivel de complejidad. Estas son las asociadas al caso de uso Realizar Diseño, debido a la complejidad en la programación; ocasionando que el tiempo de respuesta de la aplicación acreciente. En la tabla se muestran los tiempos medios de respuesta resueltos por la herramienta en segundos para distintas cantidades de usuarios.

Tabla 3 Datos obtenidos por la herramienta JMeter

Cantidad de usuarios	Tiempo medio
30	2.3
50	4.7
100	6.8

Como resultado se alcanzó que la aplicación responda en 4.7 segundos a las peticiones de los 50 usuarios concurrentes definidos en los RNF; siendo este número óptimo, los tiempos de respuesta deben estar por

debajo de los ocho segundos para que sean permisibles. Teniendo en cuenta que la cifra de 50 usuarios es elevada. Además se evaluó el tiempo de respuesta de la aplicación para una mayor cantidad de usuarios en línea; obteniéndose que responde adecuadamente a una demanda excesiva de hasta 100 usuarios conectados concurrentemente.

3.5 Resultados Obtenidos

Como resultado del desarrollo de la investigación se obtuvo el Módulo de Reportes para la herramienta SIGDAT. El cual brindó la posibilidad de realizar reportes personalizados sobre los datos recogidos por las encuestas mediante una interfaz amigable y de fácil uso. Automatizó el proceso de asignación del modelo de datos a los componentes del diseño y posibilitó exportar los reportes a diversos formatos (Pdf, Word, Excel) convirtiéndose en una herramienta útil para el apoyo a la toma de decisiones según las necesidades de los usuarios. En la siguiente imagen se muestra un ejemplo de un reporte realizado mediante el Módulo de Reportes y exportado al formato Pdf.

Fig. 16 Ejemplo de Reporte

3.6 Conclusiones Parciales del capítulo

A lo largo del presente capítulo se realizó el modelo de implementación del módulo con el objetivo de mostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes. Se utilizaron estándares de codificación para lograr un estilo claro y organizado del código durante la fase de implementación. Para verificar la calidad de la aplicación se validó la completitud de los requisitos con las pruebas funcionales al software. Se calculó el rendimiento del sistema a través de las pruebas de carga y estrés, con el apoyo de la herramienta JMeter. Se identificaron un total de 15 NC, de ellas cuatro de código, cinco de interfaz y seis de ortografía las cuales fueron resueltas paulatinamente obteniéndose un producto libre de errores y listo para su ejecución.

Conclusiones Generales

Al terminar la investigación se puede afirmar que se proporcionó cumplimiento a los objetivos planteados, llegando a las siguientes conclusiones:

- Se realizó un estudio de los principales conceptos relacionados con las herramientas para la gestión de reportes, dentro y fuera de la Universidad, permitiendo sentar las bases para el desarrollo del Módulo de Reportes para SIGDAT.

- Se realizó el análisis y diseño Módulo de Reportes, alcanzándose como resultado los diagramas y artefactos necesarios para guiar el desarrollo del mismo.

- Se realizó la implementación del Módulo de Reportes cumpliendo con los 23 requisitos funcionales identificados.

- Se diseñaron y ejecutaron pruebas a nivel de desarrollador y de integración, las cuales permitieron comprobar el correcto funcionamiento del Módulo de Reportes.

- Como resultado se obtuvo el Módulo de Reportes para SIGDAT, que permite realizar reportes personalizados dentro de la herramienta SIGDAT, agilizando el proceso de análisis de la información.

Recomendaciones

Luego de haber cumplido con los objetivos trazados y partir de los resultados obtenidos con el presente trabajo de diploma, se proponen las siguientes recomendaciones que permitirán dar continuidad a la investigación desarrollada:

- Incorporar los restantes componentes de diseño que soporta el Servidor Dinámico de Reportes (SDR).
- Incorporar al editor de consultas del área de diseño del reporte nuevas funcionalidades que le permitan a los usuarios realizar consultas complejas de forma visual.

Referencias

1. Arrighi, H. Michael. www.epidemiolog.net. [En línea] 11 de noviembre de 2008. [Citado el: 13 de 11 de 2013.] http://www.epidemiolog.net/es/endesarrollo/GestionYanalisisDeDatos.pdf.

2. Yanet R. Morales, Claudia N.Sanz, Alberto M. Garnache, Adaily H. Carballe. *Sistema Integrado de Gestión de Datos.* 2011.

3. Revelles, Jesus Salinas. Introducción a Jasper Report. [En línea] 14 de octubre de 2013. [Citado el: 15 de noviembre de 2013.] http://www.slideshare.net/EmatizTecnologia/introduccin-a-jasperreports.

4. Microsoft Developer Network. MSDN. [En línea] 2008. [Citado el: 15 de mayo de 2014.] http://msdn.microsoft.com/es-es/library/ms225593%28v=vs.90%29.aspx.

5. Alexander Tzyganenko. Fast Reports. [En línea] 31 de marzo de 2014. [Citado el: 16 de mayo de 2014.] http://www.fast-report.com/es/.

6. GDR. Publicasiones uci. [En línea] 2012. [Citado el: 28 de noviembre de 2013.] https://publicaciones.uci.cu/index.php/SC/search/advancedResults(GDR)..

7. Águila, Yoandry Pacheco. *SDR.* 17 de 05 de 2014.

8. Roque, Alberto M. Garnache y Diana Monné. *Procesos de Desarrollo de Software.* 2011.

9. Computación., Metodología y Tecnología de la Programación. Departamento de Sistemas Informáticos. *Introducción a Herramientas CASE y SystemArchitect.*

10. Lennis Zamudio, Luis Ramirez, Manuel Alvarez, Anaiz Rodriguez. Teoría y Administración de Base de Datos. [En línea] Universidad Yacambú, 2009. [Citado el: 18 de mayo de 2014.] http://www.oocities.org/es/avrrinf/tabd/Foro/Foro_UML.htm.

11. Visual Paradigm. Sitio oficial Visual Paradigm. [En línea] 2011. [Citado el: 27 de noviembre de 2013.] http://www.visual-paradigm.com..

12. IDE. [En línea] 2009. [Citado el: 03 de diciembre de 2013.] http://www.doguezmen.wordpress.com/2013/01/25/entorno_de_desarrollo_integrado/.

13. Manuel López Michelone. NetBeans 7.4 final. [En línea] 22 de octubre de 2013. [Citado el: 2 de diciembre de 2013.] http://www.unocero.com/2013/10/22/sale-netbeans-7-4-final/.

14. Chawla, Shekhar S. Sistema de Gestión de Bases de Datos. [En línea] 2008. [Citado el: 06 de 12 de 2013.] http://www.um.es/geograf/sigmur/sigpdf/temario_9.pdf.

15. Rafael Martinez. PostgreSQL. [En línea] 2 de octubre de 2010. [Citado el: 06 de diciembre de 2013.] http://www.postgresql.org.es/sobre_postgresql.

16. Arpug. Grupo de usuarios PostgreSQL de Argentina. [En línea] 2009. [Citado el: 19 de mayo de 2014.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

17. Definición de. [En línea] 2008. [Citado el: 10 de noviembre de 2013.] http://definicion.de/lenguaje-de-programacion.

18. php. ¿Qué es PHP? [En línea] 2009. [Citado el: 1 de diciembre de 2013.] http://php.net/manual/es/intro-whatis.php.

19. Damián Pérez Váldes. Maestros del Web. [En línea] 3 de julio de 2007. [Citado el: 19 de 05 de 2014.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

20. Javier J. Gutierrez. Framework. [En línea] 2011. [Citado el: 29 de 11 de 2013.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

21. Symfony. Sitio oficial de Symfony. [En línea] 2013. [Citado el: 27 de 11 de 2013.] http://www.symfony.org.

22. José Hernandez. ExtJs.mx. [En línea] 27 de noviembre de 2011. [Citado el: 16 de 05 de 2014.] http://www.extjs.mx/2011/11/27/post-con-cursos-generales/.

23. Lobo, Ing. Armando Robert. *Lycan IDE Documento Visión.* 07/10/210.

24. Eguiluz, Javier. *Desarrollo web ágil con Symfony2.* 2011.

25. Juán Pavón Mestras. Servidores Web - Apache. [En línea] diciembre de 2012. [Citado el: 19 de 05 de 2014.] http://www.fdi.ucm.es/profesor/jpavon/web/31-ServidoresWeb-Apache.pdf.

26. Marcello Visconti y Hernán Asstudillo. Patrones de Diseño. [En línea] 2012. [Citado el: 29 de 11 de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

27. Alonso G. Casaty. Patrones Arquitectónicos. [En línea] 2007. [Citado el: 19 de 05 de 2014.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.

28. MVC. Patrón de Arquitectura Modelo Vista Controlador. [En línea] 2013. [Citado el: 19 de 05 de 2014.] http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html.

29. Universidad Politécnica de Madrid. Patrones del Gang of Four. [En línea] 2011. [Citado el: 19 de mayo de 2014.] http://is.ls.fi.upm.es/docencia/proyecto/docs/patrones_gof.pdf.

30. Facultad de Cs. Físicas y Matemáticas. Cursos Ingeniería de Software. [En línea] 2005. [Citado el: 3 de 12 de 2013.] https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&ved=0CDwQFjAD&url=https%3A%2F%2Fwww.u-cursos.cl%2Fingenieria%2F2005%2F2%2FCC51A%2F1%2Fmaterial_docente%2Fobjeto%2F76454&ei=k2qfUu_LDILwyQGDs4HYBA&usg=AFQjCNFXzAMV6zkXbb2CJ9IddPprj8.

31. Larman, Craig. *UML y Patrones 2da Edición.*

32. Lis Mirabal. slideShare. [En línea] 21 de octubre de 2012. [Citado el: 19 de 05 de 2014.] http://www.slideshare.net/Lismirabal/requerimientos-funcionales-y-no-funcionales.

33. Armando Cabrera. slideShare. *Fundamentos de Ingeniería de Software.* [En línea] 2008. [Citado el: 20 de 05 de 2014.] http://www.slideshare.net/ktyk/uml-casos-de-uso.

34. Patrones Caso Uso. [En línea] febrero de 2012. [Citado el: 21 de mayo de 2014.] http://blog.micayael.com/2013/12/02/patrones-de-casos-de-uso-object-manager/.

35. Omar Beltrán. slideshare. *Desarrollo de Software orientado a objetos.* [En línea] 19 de octubre de 2013. [Citado el: 21 de mayo de 2014.] http://www.slideshare.net/omarbeltrancelismendoza/05-modelo-dediseno?qid=b9da3fd1-ec6c-4952-82a8-e442d50fe25c&v=default&b=&from_search=5.

36. Andrés Cofrán. slideshare. *Diagrama de Paquetes.* [En línea] 28 de junio de 2011. [Citado el: 22 de mayo de 2014.] http://www.slideshare.net/andrescofran/diagrama-paquetes-colaboracion-y-componetes-6738524?qid=cfdf8442-8dea-4249-8855-42b34b359a01&v=qf1&b=&from_search=1..

37. Lis Mirabal. slideshare. *Diagrama de Clases.* [En línea] 16 de octubre de 2013. [Citado el: 22 de mayo de 2014.] http://www.slideshare.net/Lismirabal/diag-de-clases.

38. Andres Macea Tirado. slideshare. *Diagramas de Interacción.* [En línea] 17 de abril de 2013. [Citado el: 28 de febrero de 2014.] http://www.slideshare.net/andresmacea31/diagramas-de-interaccion-cun-monteria?qid=e384e0eb-480c-4c3f-8665-96fe3015821d&v=qf1&b=&from_search=1.

39. Daniel Santiago Martinez. slideshare. *Diagrama de Secuencia.* [En línea] 13 de marzo de 2013. [Citado el: 03 de marzo de 2014.] http://www.slideshare.net/DaniSantia/diagrama-de-secuencia-17178366?qid=7fd9e8d5-9451-42f6-8740-1e11ff86f805&v=default&b=&from_search=1.

40. Jose Rafael Estrada. slideshare. *Modelo de Despliegue.* [En línea] 15 de junio de 2014. [Citado el: 23 de mayo de 2014.] http://www.slideshare.net/JoseRafaelEstrada/modelos-de-despliegue.

41. Leovigilda Hernandez. Modelo de Implementación. [En línea] junio de 2013. [Citado el: 15 de marzo de 2014.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

42. Eduardo Rivera Alba. Arquitectura de Software. [En línea] 2008. [Citado el: 10 de 04 de 2014.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.

43. Maria Elena Casañas. Software Libre. [En línea] 2009. [Citado el: 15 de marzo de 2014.] http://www.casanas.com.ar/attachments/Que_es_-_A_-_Conc_tecnicos.pdf.

44. Estándar Codificación. [En línea] 2011. [Citado el: 24 de mayo de 2014.] https://docs.google.com/document/d/1rbxDFM0zsbFDNRZeM2FoXfRDbYSiSt6tCdbYPA0qdzs/edit?hl=en_US&pli=1.

45. Alexander B. Oré. Quality Assurance and Software Testing. [En línea] 2009. [Citado el: 25 de mayo de 2014.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

46. Gestión de Calidad y Pruebas de Software. [En línea] [Citado el: 25 de abril de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

47. Andrés José Sebastián Rincón González. slideshare. *Pruebas de caja blanca y negra.* [En línea] 20 de octubre de 2012. [Citado el: 01 de 05 de 2014.] http://www.slideshare.net/rinconsete/pruebas-de-caja-blanca-y-negra.

48. tupakamaru. slideshare. *Pruebas de Rendimiento.* [En línea] 27 de enero de 2011. [Citado el: 05 de 05 de 2014.] http://tupakamaru.wordpress.com/2011/01/27/pruebas-de-rendimiento.