Universidad de las Ciencias Informáticas

Facultad 6

Título: “Administrador de Reportes para la versión 2.0 del Generador Dinámico de Reportes”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Lianet Salazar Labrada

Tutores: Ing. Marleysi López Duque

Ing. Raidel Oceguera Ravelo

La Habana junio de 2012
“Año 54 de la Revolución”

DECLARACIÓN DE AUTORÍA

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

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

Lianet Salazar Labrada	**Marleysi López Duque**	**Raidel Oceguera Ravelo**
_____	_____	_____
Firma del Autora	Firma del Tutor	Firma del Tutor

DATOS DE CONTACTO

AUTORA:

Lianet Salazar Labrada.
Universidad de las Ciencias Informáticas,
Ciudad de la Habana, Cuba
E-mail: lslabrada@estudiantes.uci.cu.

TUTORES:

Ing. Marleysi López Duque.
Universidad de las Ciencias Informáticas,
Ciudad de la Habana, Cuba
E-mail: mduque@uci.cu.

Ing. Raidel Oceguera Ravelo.
Universidad de las Ciencias Informáticas,
Ciudad de la Habana, Cuba
E-mail: roceguera@uci.cu.

AGRADECIMIENTOS

A las dos personas más importantes en mi vida, mi mamá y mi papá, los cuales desde el día en que nací dejaron de vivir para ellos y comenzaron a vivir para mí, que siempre han estado a mi lado y me han apoyado en todo. Gracias por todos los consejos y por las palabras de ánimo que me dan cada vez que me siento mal y acudo a ustedes, que son mi mayor tesoro y el regalo más hermoso que tengo.

A mis abuelos Edelmira, María y Reynaldo gracias por confiar en mí y estar seguros de que este momento llegaría, y a mi abuelo Eduardo Gil, que aunque no esté presente físicamente yo sé que estaría muy orgulloso de mí.

A Juan Carlos, por el apoyo diario, por estar a mi lado en los momentos más difíciles, por escuchar mis lamentos, por estar siempre pendiente de mí, por comprenderme, soportarme y quererme.

A mis tutores Marleysi y Raidel gracias por ser cómplices de este gran esfuerzo, por el apoyo brindado, por ofrecerme sus conocimientos, por sus sacrificios en algún tiempo incomprendido, por sus ejemplos de superación incasable, su comprensión y confianza, porque sin su apoyo no hubiera sido posible la culminación de mi carrera profesional y sobre todas las cosas por su esmerada dedicación.

A mi hermana Dayma que siempre ha sido un ejemplo para mí. Gracias por todo el cariño y el amor que me has dado aunque estés lejos, siempre estás presente y sé que estás muy orgullosa de mí.

A mis tías(os), primas(os) y suegros por su apoyo y cariño en todo momento. En especial a mi tía Elaine, gracias por todo lo que has hecho por mí. Muchas gracias a Angel Parra que ha sido como un padre para mí, gracias por todo tu cariño y tu amor.

A mis amigos, por los buenos momentos que pasamos juntos y los recuerdos que quedaron grabados para siempre, en especial a Yanel y Ariadna gracias por tantas experiencias vividas juntas en estos cinco años que nunca olvidaré. A Doble, Lincon, Katy, Reisel, Pepe, Julio César, Reynaldo Niebla, Lianet Cruz y Garnache, gracias por toda su ayuda, su paciencia y comprensión. Además para todos mis amigos de la universidad, en especial a los que nos mantuvimos desde el inicio y llegamos al final, que no me alcanzaría el papel para nombrarlos.

DEDICATORIA

A mi eterno amor, mi papá, por guiar cada uno de mis pasos, por brindarme siempre su amor a cambio de nada, sus enseñanzas, sus años de sacrificio, su confianza, sus consejos y apoyo en cada momento, por ser el papá más lindo y especial de este mundo, por confiar tanto en mí y estar seguro que no lo defraudaría. Te quiero mucho.

A la mujer más linda del mundo, mi mamá, por todo lo que has hecho en la vida por mí, solo de darme la posibilidad de venir al mundo fue lo más hermoso que cualquier ser humano puede pedir, por su amor, confianza, dedicación, apoyo, por ser guía ejemplar y constante, por ser una madre excepcional y no te comparo con nada en este mundo, sin ser perfecta me has guiado por el buen camino. Te quiero mucho.

RESUMEN

La creación de la Universidad de las Ciencias Informáticas (UCI) le permite al país insertarse en el vertiginoso desarrollo del software. En dicha institución se desarrollan gran cantidad de proyectos, los cuales se dividen en centros de desarrollo. El Centro de Tecnología de Gestión de Datos (DATEC) es uno de estos centros, dedicado a gestionar los proyectos y soluciones especializadas en el análisis y almacenamiento de datos. Entre los proyectos que se desarrollan se encuentra el Generador Dinámico de Reporte (GDR), el cual es una herramienta que permite la construcción de reportes a partir de la información almacenada en una base de datos. El presente trabajo tiene como objetivo desarrollar el módulo Administrador de Reportes para el sistema Generador Dinámico de Reportes V (2.0), permitiendo la entrega de los reportes de forma dinámica, mediante la creación de suscripciones las cuales pueden ser de tipo Correo o FTP. Para desarrollar el módulo Administrador de Reportes se utilizó la metodología OpenUP, el *framework* Symfony para la aplicación del lado del servidor y para el lado del cliente el *framework* ExtJS. Para el modelado de la solución se utilizó Visual Paradigm y como entorno integrado de desarrollo NetBeans. Se realizaron pruebas unitarias y funcionales para comprobar el correcto funcionamiento del módulo. Obteniéndose como resultado el Administrador de Reportes con las nuevas funcionalidades incluidas, las cuales son de gran importancia para el GDR.

PALABRAS CLAVES

Administrador de reportes, base de datos, generador dinámico de reportes, suscripciones

TABLA DE CONTENIDOS

INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTO TEÓRICO 4
1.1 Motores de Reportes 4
1.1.1 Jasper Reports 4
1.1.2 Crystal Reports 5
1.1.3 Active Reports 5
1.1.4 Microsoft SQL Server 2005 Reporting Services 6
1.1.5 Selección del Motor de Reportes 6
1.2 Tecnologías a Utilizar 8
1.2.1 Java 8
1.2.2 XML 8
1.2.3 JavaScript 9
1.2.4 CSS 9
1.2.5 PHP5 10
1.2.6 AJAX 10
1.3 Marcos de Trabajo 11
1.3.1 Symfony 1.4.8 11
1.3.2 ExtJS 3.3.0 12
1.4 Entorno de Desarrollo 13
1.4.1 NetBeans 7.0 13
1.5 Formato Alternativo de Envío y Recepción de Datos 14
1.5.1 JSON 14
1.6 Gestor de Base de Datos 14
1.6.1 PostgreSQL 9.0 15
1.7 Administrador de Bases de Datos 15
1.7.1 pgAdmin III 16
1.8 Mapeo de Objetos a Bases de Datos 16
1.8.1 Propel 17
1.9 Lenguaje de Modelado 17

1.10 Herramienta de Modelado 18
1.10.1 Visual Paradigm 6.4 18
1.11 Metodología de Desarrollo 19
1.11.1 OpenUP 19
1.12 Patrones de Diseño 20
1.12.1 Patrones GOF (Gang of Four “Banda de los Cuatro”) 20
1.12.2 Patrones GRASP (General Responsabilty Assignment Software Patterns, “Patrones de Software para la Asignación General de Responsabilidad”) 22
1.13 Patrones de Arquitectura 23
Conclusiones del Capítulo 25
CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN 26
2.1 Modelo de Dominio 26
2.2 Requisitos de Software 27
2.2.1 Requisitos Funcionales 27
2.2.2 Requisitos no Funcionales 31
2.3 Modelo de Casos de Uso del Sistema 32
2.4 Patrones 40
2.4.1 Patrones de Diseño GRASP 40
2.4.2 Patrones de Diseño GOF 41
2.4.3 Patrones de Arquitectura 41
2.5 Modelo de Diseño 42
2.6 Diagramas de Clases del Diseño 42
2.7 Diagramas de Interacción 44
2.7.1 Diagrama de Secuencia 45
2.8 Modelo de Datos 48
2.9 Modelo de Despliegue 49
Conclusiones del Capítulo 49
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS 50
3.1 Diagrama de Componentes 50
3.2 Mecanismos de Implementación 52

3.3 Implementaciones Significativas....52
3.4 Pruebas....54
3.4.1 Nivel de Prueba....54
3.4.2 Tipos de Pruebas....54
3.4.3 Método de Prueba....55
3.4.4 Casos de Pruebas....56
Conclusiones del Capítulo....61
CONCLUSIONES....62
RECOMENDACIONES....63
REFERENCIAS BIBLIOGRÁFICAS....64
BIBLIOGRAFÍA....66
ANEXOS....68
GLOSARIO....71

INTRODUCCIÓN

En la era de la información y de la explosión de sus tecnologías la humanidad ha alcanzado un desarrollo imprevisible. Resulta evidente el progreso de las sociedades industriales y es el conocimiento uno de los factores esenciales para dicho desarrollo. La sociedad de la información, con organizaciones basadas en el aprendizaje, se sustenta en un desarrollo tecnológico sin precedentes. Las grandes compañías planifican sus productos en función de la gestión del conocimiento y de la viabilidad para su obtención. En este contexto, las tecnologías de la información y las telecomunicaciones no son más que un medio para transmitir y gestionar datos, información y conocimiento.(1)

La información es un elemento fundamental para la organización y planificación de las empresas, influyendo en la correcta toma de decisiones relacionadas con los procesos e investigaciones que éstas desarrollan. La información debe ser concreta, precisa, coherente y adaptada a las necesidades que se requieren satisfacer. Con el transcurso de los años la gestión de la información se ha convertido en un elemento fundamental para el desarrollo, ocupando un espacio mayor en la economía de los países. La gestión de la información es un elemento vital para cualquier empresa o institución porque influye de manera directa en la forma en que éstas operan. Un sistema de información está conformado por un conjunto de componentes capaces de realizar operaciones de procesamiento con datos para generar información.(1)

Las bases de datos constituyen una forma de almacenar grandes cantidades de información organizadas siguiendo un determinado modelo que facilite su recuperación y modificación. En todo sistema de información es necesario disponer de una herramienta complementaria cuya función sea crear reportes a partir de la información almacenada en una base de datos. Precisamente, los generadores de reportes cumplen con esta función. Los mismos utilizan una especie de lenguaje transparente para el usuario, por medio del cual este realiza consultas a la base de datos y obtiene información de ella en forma de reporte.(2)

La mayoría de las organizaciones utilizan reportes para registrar y visualizar análisis y resultados. De esta manera, los directivos de las empresas pueden evaluar la trayectoria del negocio a partir de los reportes de información. Los cuales permiten identificar nuevos negocios o servicios que beneficien la empresa. Como consecuencia, los reportes son considerados una necesidad principal en Inteligencia de Negocio.

La Universidad de las Ciencias Informáticas (UCI), tiene como principales metas informatizar el país y desarrollar la industria del software para contribuir al desarrollo económico del mismo. Su modelo de producción está concebido en Centros de Desarrollo vinculados a las facultades y especializados en áreas temáticas. Dichos centros vinculan la producción, la investigación y la formación de los estudiantes en profesionales de la informática. Uno de los centros productivos de la UCI es el Centro de Tecnologías de Gestión de Datos (DATEC), especializado en tecnologías de Gestión de Datos, la cual está relacionada con la planificación, control y entrega de recursos de información. Incluye el diseño, implementación y soporte de las bases de datos así como las tecnologías de reporte y análisis. (3)

Entre las tecnologías que se desarrollan en este centro se encuentra la Plataforma de Ayuda a la Toma de Decisiones y Soluciones Integrales (PATDSI) y dentro de esta plataforma se incluye el sistema Generador Dinámico de Reportes (GDR). Actualmente, el GDR (versión 1.8) está compuesto por 6 módulos los cuales cuentan con una serie de características importantes para construir y guardar reportes; además permiten generar reportes con datos de diversos gestores de bases de datos.

Dentro de los módulos que contiene se encuentra el Administrador de Reportes, el cual brinda la posibilidad de administrar los reportes y plantillas previamente creadas en el GDR. Este trabajo se realiza a partir de la recomendación de *realizar la implementación de la interfaz del módulo Administrador de Reportes* (4). Además el módulo carece de una de las funcionalidades más solicitadas por los clientes las suscripciones, la cual permite recibir de forma automática un reporte, sin necesidad de consultar la aplicación. Dicha suscripción tiene asociada una frecuencia, la cual especifica la periodicidad con la que se desea que el reporte llegue al correo o a la dirección FTP. Actualmente el módulo necesita de nuevas funcionalidades que permitan administrar los reportes y plantillas existentes.

Debido a la necesidad de dar solución a la situación planteada, se identifica como ***problema científico:*** ¿cómo garantizar la administración de reportes para la versión 2.0 del Generador Dinámico de Reportes?

Este problema determinó que el ***objeto de estudio*** de esta investigación sea: proceso de desarrollo de software en los sistemas generadores de reportes, y se precisa como ***campo de acción:*** proceso de desarrollo de software del Administrador de Reportes en el Generador Dinámico de Reportes V2.0.

Para dar solución al problema planteado se define como ***objetivo general:*** desarrollar el módulo Administrador de Reportes para el sistema Generador Dinámico de Reportes V2.0.

Este objetivo general se desglosa en los siguientes ***objetivos específicos:***

- Definir las funcionalidades que debe tener el módulo Administrador de Reportes.

- Diseñar el módulo que brinde solución al problema planteado.
- Implementar las funcionalidades definidas.
- Validar el correcto funcionamiento del módulo implementado.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes ***tareas:***

- Revisión de los motores de reportes existentes para escoger el que se utilizará en la implementación del módulo.
- Definición de los requisitos funcionales y no funcionales para el correcto funcionamiento del módulo.
- Selección de patrones a emplear para el desarrollo del Administrador de Reportes.
- Realización del diseño de clases para facilitar la implementación del módulo.
- Selección de los marcos de trabajo a utilizar para la implementación.
- Implementación del diseño realizado para satisfacer las funcionalidades definidas.
- Aplicar pruebas de caja negra para validar el módulo implementado.
- Diseñar los casos de pruebas para determinar el correcto funcionamiento de los requisitos.
- Aplicar pruebas unitarias para validar el código de las clases del módulo.

El trabajo ha sido desarrollado en 3 capítulos, los cuales se detallan a continuación:

Capítulo 1: Fundamento Teórico: En éste capítulo se hace referencia a los diferentes motores de reportes que existen en el mundo. Se especifican las herramientas, tecnologías y metodología que se utilizarán para implementar el módulo Administrador de Reportes, utilizando las definidas por el equipo de arquitectura del GDR. Además se mencionan los patrones de diseño (GRASP y GOF) y de arquitectura.

Capítulo 2: Análisis y Diseño de la Solución: En éste capítulo se especifican los requisitos funcionales y no funcionales que el sistema debe cumplir. Además se generan los artefactos que pertenecen a la etapa de análisis y diseño. Dentro de los cuales se encuentran el diagrama de casos de uso del sistema, clases del diseño, secuencia, despliegue y modelo de datos. Se identifican los patrones de diseño (GRASP y GOF) y de arquitectura que se utilizarán.

Capítulo 3: Implementación y Pruebas: En éste capítulo se generan los artefactos asociados a la etapa de implementación y pruebas, dentro de los cuales se encuentran el diagrama de componentes. Se mencionan los mecanismos de implementación utilizados. Se especifican los tipos de pruebas y métodos que se utilizarán para validar el correcto funcionamiento del módulo. Además se muestra el código de alguna de las implementaciones más significativas.

CAPÍTULO 1: FUNDAMENTO TEÓRICO

En este capítulo se realiza un estudio de los diferentes motores de reportes que existen para seleccionar el que se utilizará para la versión 2.0 del GDR. Guiados por los documentos "*Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0*" y "*Pautas de arquitectura para el desarrollo de la versión 2 del GDR*" se describen las herramientas, metodología de desarrollo de software, marcos de trabajo y tecnologías que se utilizarán durante el ciclo de desarrollo del módulo Administrador de Reportes.

1.1 Motores de Reportes

En el mundo existen diferentes motores de reportes, para seleccionar el que se utilizará en la versión 2.0 del Generador Dinámico de Reportes se realizó un estudio sobre algunos motores. Dentro de los cuales se encuentran Jasper Reports, PHP Reports, Crystal Reports, Exchange Reporter Plus, Active Reports y Microsoft SQL Server 2005 Reporting Services. A continuación se explican las principales características de cada uno de ellos.

1.1.1 Jasper Reports

Jasper Reports es una librería de código abierto para la creación de informes en Java, la misma es poderosa y flexible para la generación y gestión de reportes. Permite la creación de reportes que pueden ser presentados en la web o se pueden crear ficheros en diversos formatos, como por ejemplo, PDF, HTML, XLS, ODT y RTF para que los datos puedan ser procesados con otras aplicaciones. También soporta diferentes fuentes de datos: postgresql, mysql, oracle, mssql, odbc, interbase, informix, xml, csv. Este motor brinda además, la posibilidad de generar textos o imágenes de fondo para ser utilizados como marcas de agua por motivos de seguridad o simplemente para su identificación. Soporta varios tipos de gráficos. Admite la creación de reportes dentro de reportes, permite añadir enlaces en los reportes, lo que haría más fácil y rápida la navegación entre las diferentes secciones del informe cuando este es muy largo. *Jasper Reports* se usa comúnmente con *iReport*, un front-end gráfico de código abierto para la edición de reportes. Con una solución de este tipo, se dispone en el tiempo deseado de los datos indispensables para lograr una gestión eficaz. También pueden seguirse fácilmente los resultados obtenidos y la manera en que la actividad progresa en función de los objetivos fijados. Pueden descubrirse las tendencias y los factores claves que afectan a la actividad y los resultados de la empresa. Las principales características de *Jasper Reports* son:

- Formato y layout flexible.
- Informes y subinformes con múltiples niveles de anidamiento.
- Librería de cálculos predefinidos.
- Internacionalización.
- Herramientas para diseño gráfico de los informes.
- Fácil integración en aplicaciones.

Sistema Operativo: Se puede utilizar en cualquier sistema operativo o entorno, siempre y cuando exista una implementación de la máquina virtual de Java.

Licencia: Se distribuye bajo los términos de la Licencia Pública para Librerías GNU (GNU Library Public License), por lo que es software libre y está respaldado por una gran comunidad internacional de desarrollo. (5)

1.1.2 Crystal Reports

Crystal Reports es un producto de alta tecnología, además de ser una herramienta potente es fácil de usar para el diseño y generación de reportes a partir de datos almacenados en una base de datos u otra fuente de información. *Crystal Reports* es una solución de informes poderosa, dinámica y procesable que ayuda a diseñar, explorar, visualizar y entregar informes por medio de la web. Permite transformar rápidamente cualquier fuente de datos en contenido interactivo, integrar estrechamente capacidades de diseño, modificación y visualización en aplicaciones .NET, Java o COM, y además permite a los usuarios finales acceder e interactuar con los reportes a través de portales Web, dispositivos móviles y documentos de Microsoft Office.

Sistema Operativo: Se puede utilizar en sistemas operativos como, Microsoft Windows XP con Service Pack (SP) 2, Windows Server 2003 con SP 1 o posterior.

Licencia: Se distribuye bajo los términos de la EULA (End User Licensing Agreement), por lo que es software privativo y de uso restringido mediante el pago de patente. (6)

1.1.3 Active Reports

Active Reports es un componente de informes .NET, entre las características claves de este componente figuran la personalización, rendimiento rápido, alta calidad y numerosas prestaciones. *Active Reports* admite exportaciones de datos a todos los formatos de archivo habituales, como PDF, Excel, RTF, TIFF.

Incluye un diseñador de informes Visual Studio .NET fácil de usar y una potente API. También ofrece un despliegue de tiempo de ejecución armonizado y libre de derechos.

Sistema Operativo: En tiempo de diseño para utilizarlo se requiere tener instalado Microsoft Visual Studio.NET 2003, Visual Studio 2005, o Visual Studio 2008 sobre Windows NT 4.0, Windows 2000, Windows XP, Windows Vista o Windows 2003 Server así como Microsoft .NET Framework v1.1, v2.0, v3.0, o v3.5.

Licencia: Se distribuye bajo licencia privativa perteneciente a Grape City. (7)

1.1.4 Microsoft SQL Server 2005 Reporting Services

Reporting Services es una plataforma que permite definir, administrar y distribuir distintos formatos de reportes dentro de una organización o a través de múltiples organizaciones. *SQL Server Reporting Services* otorga todas las facilidades necesarias para que podamos cubrir cada uno de los aspectos asociados a la creación, administración y distribución de los reportes. Otra de las grandes características es que puede exportar los reportes en distintos formatos, como hojas de excel, PDF, texto, XML.

Sistema Operativo: Se puede utilizar en sistemas operativos como Windows, preferiblemente en sus versiones para servidores.

Licencia bajo la que se libera: Se distribuye bajo licencia privativa perteneciente a Microsoft Corporation, dicha licencia forma parte de la licencia de Microsoft SQL Server 2000 ó 2005. (8)

1.1.5 Selección del Motor de Reportes

Existen múltiples herramientas cuyo objetivo principal es generar reportes, para seleccionar el que se utilizará hay que tener en cuenta las necesidades de los usuarios. La versión 1.8 del Generador Dinámico de Reportes utiliza el motor *PHP Reports*, para la versión 2.0 se desea cambiar de motor. Dentro de las deficiencias que presenta se encuentran: contiene limitados formatos de salida, ausencia de seguridad sobre el documento final, no permite la visualización de informes que contengan un conjunto de reportes, tampoco permite rango de visualización por páginas. Para seleccionar el que se utilizará se realizó una comparación entre diferentes motores de reportes, teniendo en cuenta los formatos de salida y fuentes de datos que soporta, sistema operativo en los que se puede utilizar y si es libre.

Aspectos	Jasper Reports	PHP Reports	Crystal Reports	Active Reports	Microsoft SQL
Formatos de salida	HTML, PDF, Excel, CSV, ODT, RTF, XLS	HTML, PDF, Excel, CSV	HTML, PDF, Excel, CSV	RTF, PDF, Excel, HTML, TIFF	HTML, PDF, TIFF, Excel, CSV
Fuentes de datos	postgresql, mysql, oracle, mssql, odbc, interbase, informix, xml, csv	postgresql, mysql, oracle, mssql, odbc, interbase, informix	postgresql, mysql, oracle, mssql, odbc, interbase, informix	postgresql, mysql, oracle, mssql, odbc, interbase, informix	postgresql, mysql, oracle, mssql, odbc, interbase, informix
Libre	Si	Si	No	No	No
Sistema Operativo	Multiplataforma	Multiplataforma	Windows	Windows	Windows

Tabla 1: Comparación entre diferentes motores de reportes.

Como resultado del estudio realizado se decide escoger el motor de reportes *Jasper Reports*, por ser libre y multiplataforma. Además es uno de los que más formatos de salida y fuentes de datos soportan, permite generar sub-reportes para facilitar el diseño, brinda la posibilidad de generar textos o imágenes de fondo para utilizarlo como marcas de agua con el propósito de identificar el reporte o simplemente por motivos de seguridad. Ofrece diferentes componentes dentro de los cuales se encuentran tablas de contingencia dinámica, etiquetas, grid, dial, 2D, 3D. Por todo lo anteriormente planteado y por la superioridad que presenta frente a los demás motores se selecciona *Jasper Reports* para desarrollar la versión 2.0 del Generador Dinámico de Reportes. Una vez seleccionado el motor, se identifican las tecnologías con las que se trabajará.

1.2 Tecnologías a Utilizar

1.2.1 Java

Java es un lenguaje moderno de alto nivel, que recoge los elementos de programación que típicamente se encuentran en todos los lenguajes de programación, permitiendo la realización de programas profesionales. Se utiliza la programación orientada a objetos, pues los objetos son los elementos básicos para modelar los datos sobre los que trabaja un programa. Algunas de las características que presenta son:

- **Lenguaje simple:** permite aprender de manera muy rápida.
- **Orientado a objetos:** soporta las características principales del paradigma de la programación orientado a objetos: encapsulación, herencia y polimorfismo.
- **Distribuido:** proporciona una colección de clases para su uso en aplicaciones de red, facilitando así la creación de aplicaciones distribuidas.
- **Robusto:** elimina el uso de apuntadores para referenciar las áreas de memoria, además libera al desarrollador de la necesidad de desalojar la memoria que la aplicación ya no usa.
- **Seguro:** se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.
- **Indiferente a la arquitectura:** está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo y sobre arquitecturas distintas o con sistemas operativos diversos.
- **Multiplataforma:** funciona en cualquier sistema operativo que tenga instalada la máquina virtual de Java. (9)

1.2.2 XML

Es el estándar de Lenguaje de Etiquetado Extensible (*Extensible Markup Language*), conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación. Un lenguaje de marcas es un mecanismo para identificar estructuras en un documento. La especificación XML define una manera estándar de añadir etiquetas a los documentos. Algunas de las ventajas de XML son las siguientes:

- Las aplicaciones se pueden generar rápidamente y su mantenimiento es sencillo.
- La información es más accesible y reutilizable.
- Separa los datos de la presentación y del proceso, lo que permite mostrar y procesar los datos.
- Ofrece un formato para la descripción de datos estructurados, facilitando que las declaraciones de los contenidos sean más precisas y los resultados de las búsquedas sean más significativos en varias plataformas. (10)

1.2.3 JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas, ésta es una de las principales ventajas que presenta sobre el HTML, además es muy fácil de aprender. Es técnicamente un lenguaje de programación interpretado por lo que no es necesario compilar los programas para ejecutarlos. Permite integrarlo directamente en páginas HTML. Los programas escritos en este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, no guarda ninguna relación directa con el lenguaje de programación Java. Algunas de las principales características son las siguientes:

- Es interpretado por el navegador del cliente.
- Está basado en objetos. Por lo cual emplea herencia, típicas de la Programación Orientada a Objetos (POO).
- Su código se integra en las páginas HTML, incluido en las propias páginas.
- No es necesario declarar los tipos de variables que van a utilizarse.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila. (11)

1.2.4 CSS

Es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS permite separar los contenidos y su presentación, es necesario para la creación de páginas web complejas. Este lenguaje se utiliza para definir el aspecto de todos los contenidos, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista.

1.2.5 PHP5

PHP5 utiliza la interpretación y compilación para ofrecer a los programadores mejor rendimiento y flexibilidad. Compila para el código una serie de instrucciones siempre que estas son accedidas. Las instrucciones son ejecutadas una por una hasta que el script termine. Esto es diferente a la manera convencional de compilación de lenguajes como C++. Es recompilado cada vez que se solicita un script.

Se usa principalmente en interpretación del lado del servidor para la generación de páginas Web dinámicas. Las principales características de PHP5 son: rapidez, facilidad de aprendizaje, soporte multiplataforma tanto de diversos Sistemas Operativos, como servidores HTTP y bases de datos, además se distribuye de forma gratuita bajo una licencia abierta. Algunas de sus principales ventajas son:

- Tiene manejo de excepciones.
- Es completamente expandible. Está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código.
- Capacidad de conexión con la mayoría de los motores de base de datos, destaca su conectividad con MySQL y PostgreSQL.
- Permite aplicar técnicas de programación orientada a objetos.
- Es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.
- Es de código abierto, lo cual le permite al usuario realizar cambios para lograr una mejor eficiencia en lo que necesite sin necesidad de depender de una compañía. (12)

1.2.6 AJAX

El término AJAX es un acrónimo de *Asynchronous JavaScript* + XML, que se puede traducir como "JavaScript asíncrono + XML". AJAX no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de nuevas formas. Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- *XMLHttpRequest,* para el intercambio asíncrono de información.
- *JavaScript,* para unir todas las demás tecnologías.

En las aplicaciones web tradicionales, las acciones del usuario en la página desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al

navegador del usuario. AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, el intercambio de información con el servidor se produce en un segundo plano. Las aplicaciones construidas con AJAX permiten mejorar la respuesta de la aplicación mediante la creación de un elemento intermedio entre el usuario y el servidor. (13)

1.3 Marcos de Trabajo

Un marco de trabajo facilita la programación de aplicaciones, pues encapsula operaciones complejas en instrucciones sencillas. Permite simplificar el desarrollo de las aplicaciones, mediante la utilización de los patrones para resolver diferentes tareas. Además proporciona una estructura para el código fuente, permitiendo que sea más legible. (14) El equipo de arquitectura del Generador Dinámico de Reportes definió que se utilizará el *framework* Symfony para la aplicación del lado del servidor y para el lado del cliente el *framework* ExtJS.

1.3.1 Symfony 1.4.8

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web de acuerdo a algunas de sus características. Primeramente, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Presenta varias herramientas y clases que permiten reducir el tiempo de desarrollo de las aplicaciones web complejas. Symfony está desarrollado completamente en PHP 5. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows. Alguna de las características que presenta son:

- Sigue las mejores prácticas de patrones de diseño para la web.
- Código fácil de leer y permite un mantenimiento sencillo.
- Independiente del sistema gestor de bases de datos.
- Utiliza programación orientada a objetos.
- Fácil de instalar y configurar en plataformas como Windows y Linux.
- Posee una potente línea de comandos que facilitan la generación de código, lo cual permite ahorrar tiempo de trabajo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Utiliza la arquitectura Modelo – Vista – Controlador (MVC) como la capa de abstracción de base de datos, el controlador frontal y las acciones.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo:

- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- La capa de presentación utiliza plantillas y diseños que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del *framework*. Las ayudas incluidas permiten minimizar el código utilizado en la presentación, pues encapsulan grandes bloques de código en llamadas simples a funciones.
- Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.

Las interacciones con AJAX son tan fáciles de implementar mediante las ayudas que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código. (15)

1.3.2 ExtJS 3.3.0

ExtJS es una librería *JavaScript* que se utiliza para la creación de aplicaciones enriquecidas del lado del cliente. Esta librería incluye:

- Componentes Interfaces de Usuario (UI) del alto performance y personalizables.
- Modelo de componentes extensibles.
- Una librería (API) fácil de usar.
- Licencias Open Source (GPL) y comerciales.

Usar ExtJS permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor: la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona: en este tipo de aplicación el motor renderizado que puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente lo note.

- Eficiencia de la red: el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

ExtJS es uno de los *framework* que además de flexibilizar el manejo de componentes de la página como el DOM y peticiones AJAX, tiene la gran funcionalidad de crear interfaces de usuario funcionales.

Ventajas

- Posibilita crear aplicaciones complejas utilizando componentes predefinidos.
- Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores (*Firefox, Safari, Opera*).
- El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- Se distribuye la carga de procesamiento permitiendo que el servidor pueda atender más clientes al mismo tiempo. (16)

1.4 Entorno de Desarrollo

En el mundo existen diferentes entornos integrados de desarrollo (IDE) como por ejemplo Zend Studio, Eclipse y NetBeans, las características que tienen en común son: multiplataforma, soportan diversos lenguajes de programación y brindan la posibilidad de exportar e importar proyectos. Se decide utilizar NetBeans 7.0, el cual permite integrar Symfony y ExtJS como marcos de trabajo, los cuales se utilizarán para el desarrollo del módulo Administrador de Reportes y además fueron definidos por la arquitectura del GDR.

1.4.1 NetBeans 7.0

La plataforma NetBeans es un entorno integrado de desarrollo (IDE) libre y gratuito sin restricciones de uso, cuya función es proporcionar un marco de trabajo fácil y amigable. Diseñado principalmente para el lenguaje de programación Java y para lenguajes dinámicos como *PHP*, *JavaScript* and *Groovy*. Permite editar programas en java, compilarlos, ejecutarlos, depurarlos y construir rápidamente la interfaz gráfica de una aplicación. Además las aplicaciones son desarrolladas a partir de un conjunto de componentes de software a los que se le llama módulos.

NetBeans es un proyecto de código abierto de gran éxito, con una comunidad de usuarios que está en constante crecimiento. Ofrece una amplia documentación y recursos de capacitación. Es además una plataforma de ejecución de aplicaciones, esto significa que facilita la escritura de aplicaciones Java,

proporcionando una serie de servicios comunes, que a su vez están disponibles a través del IDE. Se utiliza para crear aplicaciones de escritorio, empresariales y web. NetBeans es fácil de instalar y se puede ejecutar tanto en Windows, Linux, Mac OS X y Solaris. (17)

1.5 Formato Alternativo de Envío y Recepción de Datos

1.5.1 JSON

Es un formato sencillo para intercambiar datos. Consiste básicamente en un arreglo asociativo de *JavaScript* que se utiliza para incluir información del objeto. JSON ofrece dos grandes ventajas para las interacciones AJAX: es muy fácil de leer en *JavaScript* y puede reducir el tamaño en bytes de la respuesta del servidor. Es muy sencillo de usar, especialmente como alternativa a XML. Una de sus ventajas sobre XML como formato de intercambio de datos es que prevalece un formato mucho más sencillo a la hora de escribir un analizador semántico del mismo. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de lenguajes C, C++, C#, Java, JavaScript, Perl, Python. JSON está constituido por dos estructuras:

- Una colección de pares de nombre: valor y propiedades: valor.
- Una lista ordenada de valores.

Estas son estructuras universales; que todos los lenguajes de programación las soportan de una forma u otra. Es un formato de intercambio de datos que es independiente del lenguaje de programación y se basa en estas estructuras. (18)

1.6 Gestor de Base de Datos

Los Sistemas Gestores de Bases de Datos son una herramienta efectiva, la cual permite que varios usuarios puedan acceder a los datos. Brindan un grupo de funciones con el objetivo de garantizar la confidencialidad, calidad, seguridad e integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos. Existen diferentes gestores de bases de datos dentro de los cuales se encuentran MySQL y PostgreSQL. La principal ventaja que tiene PostgreSQL sobre MySQL es que es libre.

1.6.1 PostgreSQL 9.0

PostgreSQL es una poderosa fuente, se utiliza para abrir el objeto-relacional de bases de datos del sistema. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad de datos y corrección. Se ejecuta en todos los principales sistemas operativos, incluyendo Linux, UNIX y Windows. Cuenta con características avanzadas tales como Multi-Versión Control de Concurrencia (MVCC), punto en el tiempo de recuperación, replicación asincrónica, puntos de retorno. Es altamente escalable, tanto en la gran cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede acomodar.

Lo mejor de todo, es que el código fuente está disponible bajo una licencia de código abierto. Esta licencia da la libertad de usar, modificar y distribuir PostgreSQL en cualquier forma, de código abierto o cerrado. Es además una poderosa plataforma de desarrollo. Es una herramienta de trabajo muy importante en la comunidad de software libre, que ha sido muy utilizada alrededor del mundo. Su objetivo principal es manejar de manera clara, sencilla y ordenada los conjuntos de datos que se convierten en información relevante para una organización. Entre sus principales ventajas se destacan las siguientes:

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Posee gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que atienden un gran número de solicitudes.
- Posee estabilidad y confiabilidad.
- Extensible a través del código fuente disponible.
- Multiplataforma, disponible en la mayoría de los sistemas operativos.
- Permite implementar reglas, vistas, disparadores, subconsultas y funciones.

En la versión 9.0 se incorporan notables mejoras como por ejemplo: la restauración de base de datos en forma paralela para acelerar la recuperación de las copias de seguridad, se ha mejorado el control en datos sensibles e incluso la base de datos PostgreSQL es más útil en entornos de varios idiomas. (19)

1.7 Administrador de Bases de Datos

Se decide utilizar como Administrador de Base de Datos pgAdmin III. Una característica interesante de pgAdmin III es que cada vez que realiza una modificación en un objeto, escribe la sentencia SQL correspondiente, lo que hace que sea una herramienta muy útil y a la vez didáctica.

1.7.1 pgAdmin III

Es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia *Open Source*. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. Es desarrollado en todo el mundo por una comunidad de expertos de PostgreSQL y está disponible en más de una docena idiomas. Es de software libre liberado bajo la licencia de PostgreSQL. Tiene una enorme variedad de características entre las que encontramos:

- Multiplataforma.
- Diseñado para las versiones de PostgreSQL múltiples y derivados.
- Una amplia documentación.
- Acceso a los datos.
- Acceso a todos los objetos PostgreSQL.
- Soporta la mayoría de las codificaciones del lado del servidor. (20)

1.8 Mapeo de Objetos a Bases de Datos

Para acceder a las bases de datos como si fueran orientadas a objetos, es necesaria una interfaz que traduzca la lógica de cada objeto a la lógica relacional. A este proceso se le llama Mapeo de Objetos a Bases de Datos (ORM). Actualmente existen numerosos sistemas o librerías para incluir en proyectos desarrollados bajo cualquier lenguaje de programación y que en ámbitos generales nos aportan numerosas ventajas. Dentro de las cuales encontramos:

- **Rapidez de desarrollo:** la mayoría de las herramientas ORM disponibles, permiten la creación del modelo a través del esquema de la base de datos, es decir, se crea la base de datos y la herramienta automáticamente lee el esquema de tablas y relaciones y crea un modelo ajustado.
- **Abstracción del motor de base de datos:** todo sistema ORM debe generar de forma automática las consultas a la base de datos para convertir los registros en objetos (y viceversa) y éstas deben poder adaptarse a los distintos proveedores (MySQL, Oracle, PostgreSQL).
- **Lenguaje propio para consultas a la base de datos:** otra característica importante es la exposición de clases y métodos que dan las herramientas ORM para poder extraer los datos de la forma que se necesite (filtros, ordenaciones, agrupaciones). (21)

El framework Symfony permite integrar los ORM Propel y Doctrine, se decide utilizar Propel pues es el que está por defecto integrado con Symfony 1.4.8. La principal ventaja que tiene Propel sobre Doctrine es que no es necesario tener tanto conocimiento sobre sentencias SQL para realizar una consulta. Además es uno de los ORM que tiene más autocompletación de código en los IDE's, principalmente en el NetBeans, el cual se utilizará para el desarrollo del módulo. Otra ventaja es que Propel crea 4 clases por cada tabla de la base de datos, las de acceso a datos y las de abstracción de datos. Cuando se realiza un cambio en la base de datos sólo se generan nuevamente las clases de abstracción de datos y no se perderían los métodos que se encuentran en las de acceso a datos, sin embargo Doctrine crea sólo una clase y cada vez que se realiza un cambio en la base de datos se genera nuevamente el esquema.

1.8.1 Propel

Propel es un proyecto de software libre, es una de las mejores capas de abstracción de objetos/relacional disponibles en PHP5. Permite acceder a la base de datos utilizando un conjunto de objetos, que proporciona una interfaz sencilla para almacenar y recuperar datos. Realmente es una forma muy cómoda de manipular bases de datos en nuestras aplicaciones. Solo define la estructura de la base de datos en un archivo XML y Propel solo gestionará las bases de datos. De esta manera es posible definir el esquema de la base de datos y siempre se va a tener el control sobre el mismo. Propel es uno de los más antiguos utilizados por Symfony, además que tiene buena documentación. Brinda al desarrollador de aplicaciones web, las herramientas para trabajar con bases de datos de la misma manera se trabaja con otras clases y objetos en PHP. Le facilita a la base de datos una API bien definida. (22)

1.9 Lenguaje de Modelado

Lenguaje Unificado de Modelado (UML por sus siglas en inglés, *Unified Modeling Language*) en la actualidad es uno de los lenguajes de modelado de sistemas de software más conocido y utilizado, está respaldado por el OMG (*Object Management Group*). Es un lenguaje de modelado para especificar y no para describir métodos o procesos. Permite visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. UML estandariza diferentes tipos de diagramas para representar gráficamente un sistema desde distintos puntos de vista. El hecho de

que UML sea un lenguaje de propósito general proporciona gran flexibilidad y expresividad a la hora de modelar sistemas. (23)

1.10 Herramienta de Modelado

A pesar de que existen diferentes herramientas que permiten modelar el proceso de desarrollo de software, se decide utilizar Visual Paradigm versión 6.4 como herramienta CASE, el cual permite representar alrededor de 13 diferentes tipos de diagramas, posibilita modelar procesos de negocios y bases de datos. Brinda la posibilidad de generar código (PHP, Java, Delphi, Python, C++, C#, Perl, Ruby) a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código. Permite exportar los diagramas en imagen, Excel o formato XML. Genera documentación en formatos HTML y PDF.

1.10.1 Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Se caracteriza por:

- Software libre.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar, común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones para cada necesidad.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web.
- Fácil de instalar y actualizar. (25)

1.11 Metodología de Desarrollo

Es una colección de documentos que muestran los procesos, políticas y procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de desarrollo de software. No es más que una guía que muestra paso a paso las tareas y actividades que se deben realizar para obtener un producto informático con buena calidad. Existen dos tipos de metodologías: ágiles y tradicionales (pesadas). La principal diferencia entre ambas es que las metodologías tradicionales se enfocan en el plan de proyecto. En tenerlo todo bien especificado antes de comenzar, en seguir el camino planificado y documentar exhaustivamente todo lo realizado. Por otra parte las metodologías ágiles no generan gran cantidad de artefactos, pues se enfocan en satisfacer las necesidades del cliente, adaptándose a sus siempre cambiantes necesidades.

A pesar de que XP es una metodología ágil al igual que OpenUp y genera menos artefactos, no se utiliza pues se basa sólo en la programación, donde el usuario es parte del proceso de desarrollo del software, por lo cual se basa en mantener una comunicación directa con el usuario, al cual debe entregársele en cada fase una parte funcional de la aplicación. Sin embargo en el GDR el usuario es un agente externo que no está involucrado con el equipo de desarrollo, además se necesita documentar todo el proceso de análisis y diseño para que sea más fácil programar a los desarrolladores y para mantener la calidad del software. Por lo tanto se decide utilizar OpenUP como metodología de desarrollo, por todas las características que presenta y además es la definida por el equipo de arquitectura del GDR.

1.11.1 OpenUP

OpenUP es un marco de trabajo para procesos de desarrollo de software *Open Source* que cubre un amplio sistema de necesidades para los proyectos de desarrollo. Es un proceso iterativo para el desarrollo de software que es:

Mínimo: solo incluye el contenido del proceso fundamental.

Completo: puede ser manifestado como proceso entero para construir un sistema.

Extensible: puede ser utilizado como base para agregar o para adaptar más procesos.

Algunas de las características generales que presenta son:

- Preserva la esencia del *Unified Process.*
- Desarrollo iterativo e incremental.
- Desarrollo dirigido por Casos de Uso.
- Centrado en la Arquitectura.

- Sólo lo fundamental está incluido, sin dejar de ser completo y extensible.
- Se utiliza en proyectos de corta duración y equipos de trabajo pequeños.

Fases del OpenUP

1. Concepción: es la primera fase, se basa en el entendimiento de los propósitos y objetivos, obteniendo suficiente información para confirmar que debe hacer el proyecto. El objetivo de esta fase es capturar las necesidades de los usuarios interesados en el entorno de desarrollo (stakeholder), en los objetivos del ciclo de vida para el proyecto.

2. Elaboración: es la segunda fase del ciclo de vida de OpenUP donde se trata los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la arquitectura base del sistema.

3. Construcción: fase está enfocada en el diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito es completar el desarrollo del sistema basado en la arquitectura definida.

4. Transición: es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, evalúa la funcionalidad y rendimiento del último entregable de la fase de construcción. (26)

1.12 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Además son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Algunas de las ventajas de los patrones de diseño:

- Facilitan la localización de los objetos que formarán el sistema.
- Facilitan la determinación de la granularidad adecuada.
- Especifican interfaces para las clases.
- Especifican implementaciones (al menos parciales).
- Facilitan el aprendizaje y la comunicación entre programadores y diseñadores.

1.12.1 Patrones GOF (Gang of Four “Banda de los Cuatro”)

Los patrones de diseño GOF dan una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular. Se clasifican en:

- **Creacionales:** resuelven problemas relativos a la creación de objetos.
- **Estructurales:** resuelven problemas relativos a la composición de objetos.

- **De Comportamiento:** resuelven problemas relativos a la interacción entre objetos.

Patrones Creacionales

- ***Abstract Factory:*** proporciona una interfaz para crear familias de objetos 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 crea 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

- ***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 los objetos individuales y 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

- ***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:*** 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.
- ***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.
- ***Template Method:*** 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. (27)

1.12.2 Patrones GRASP (General Responsabilty Assignment Software Patterns, "Patrones de Software para la Asignación General de Responsabilidad")

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales que son: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador. Además de otros patrones adicionales que son: Fabricación Pura, Polimorfismo, Indirección.

- **Bajo Acoplamiento:** debe haber pocas dependencias entre las clases. De manera que en caso de producirse una modificación en alguna de las clases se tenga la mínima repercusión posible en el resto de las clases. Estas clases no se afectan por los cambios en otros componentes, fáciles de entender por separado y de fácil reutilización.
- **Alta Cohesión:** cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La información que almacena una clase debe ser coherente y estar en mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño.
- **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. Es el principio básico de asignación de responsabilidades.
- **Creador:** identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento.
- **Controlador:** asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.
- **Polimorfismo:** cuando se identifican variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos.
- **Fabricación Pura:** cuando los problemas se complican se deben construir clases que se encarguen de construir los objetos adecuados en cada momento (factorías).
- **Indirección:** crear clases intermedias para desacoplar clientes de servicio y servicios. (28)

1.13 Patrones de Arquitectura

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes con sus responsabilidades, relaciones y formas en que colaboran.

- **Arquitectura orientada a servicios:** define la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios, lo cual facilita la interacción entre diferentes sistemas propios o de terceros. Proporciona una metodología y

un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

- **Arquitectura dirigida por eventos:** es un patrón que promueve la producción, detección, consumo y reacción a eventos.
- **Monolítica:** el software se estructura en grupos funcionales muy acoplados.
- **Arquitectura en pipeline (basada en filtros):** consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior. Esta arquitectura es muy común en el desarrollo de programas para el intérprete de comandos porque se pueden concatenar comandos fácilmente con tuberías. También es una arquitectura muy natural en el paradigma de programación funcional.
- **Cliente – Servidor:** es un modelo de aplicación distribuida en el que las tareas se reparten entre los servidores y los clientes. Un cliente realiza peticiones al software y el servidor da la respuesta. Los cuales no tienen que ejecutarse en la misma computadora.
- **Modelo–Vista–Controlador:** el patrón proporciona grandes ventajas como la organización de código, reutilización y flexibilidad. La programación es más organizada pues separa los datos de la aplicación, la interfaz de usuario y la lógica de los datos en tres componentes diferentes, en el cual cada capa se especializa en una función específica.
  - **Modelo:** representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
  - **Vista:** muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual de los datos representados por el Modelo.
  - **Controlador:** responde a eventos e invoca cambios en el modelo y probablemente en la vista.

  Ventajas del Modelo – Vista – Controlador:

  - Soporte para múltiples vistas pues no existe dependencia directa entre la Vista y el Modelo.
  - Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.
  - Mayor soporte a los cambios, pues los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vista al sistema generalmente no afectan al modelo. (29)

Conclusiones del Capítulo

Luego de los documentos consultados y el estudio realizado se decide utilizar las siguientes tecnologías, herramientas, motor de reportes y metodología para el desarrollo de la versión 2.0 del Generador Dinámico de Reportes, las cuales son de gran utilidad y contribuyen a mejorar el proceso de desarrollo del módulo Administrador de Reportes. OpenUP como metodología de desarrollo, el IDE de desarrollo *NetBeans* 7.0 al cual se puede integrar Symfony 1.4.8 y ExtJS 3.3.0 como marcos de trabajo. PostgreSQL 9.0 como gestor de base de datos y pgAdmin III para administrar este gestor. Para modelar los diagramas generados durante las fases de desarrollo del módulo, se utilizará Visual Paradigm 6.4 utilizando como lenguaje de modelado UML.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

Introducción

En este capítulo se describe el proceso llevado a cabo durante la fase de análisis y diseño. En el cual se definen los requisitos funcionales y no funcionales del software, este paso es de gran importancia, pues el éxito del software se define a partir de una buena identificación de los requerimientos. A partir de los requisitos funcionales se identifican y describen los casos de uso. Como parte del proceso que se realiza durante esta fase se generan los diferentes diagramas y artefactos correspondientes para dar cumplimiento a los objetivos trazados. Además se especifica la estructura física de la solución, se realiza una descripción de las principales clases del sistema, así como los principales componentes existentes en el Administrador de Reportes del Generador Dinámico de Reportes (V2.0).

2.1 Modelo de Dominio

El modelo de dominio es utilizado por el analista para comprender el sector de negocios al cual el sistema va a servir. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Puede ser tomado como el punto de partida para el diseño del sistema, por lo que el mapa de conceptos del modelo de dominio constituye una primera versión del sistema. (30)

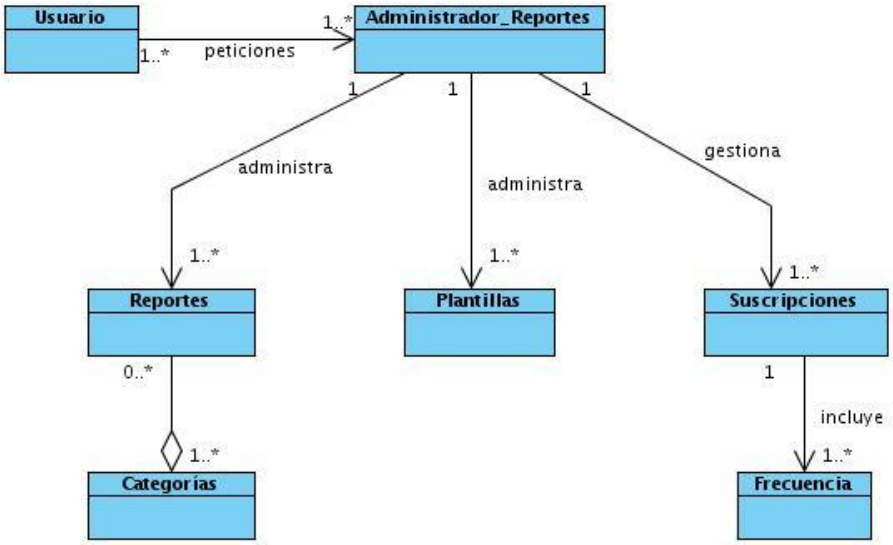


Figura 1: Modelo de Dominio

- **Usuario:** representa los usuarios que accederán a la aplicación.
- **Administrador Reportes:** recibe las peticiones del usuario, las procesa, analiza y determina que camino debe tomar para finalmente responder la petición.
- **Reportes:** contiene una lista de reportes a los que el usuario tiene acceso, le permite mover, copiar, renombrar, eliminar o mostrar las propiedades del reporte seleccionado.
- **Plantillas:** contiene una lista de plantillas a las que el usuario tiene acceso, le permite renombrar o eliminar una plantilla, además permite ver las propiedades de la plantilla seleccionada.
- **Suscripciones:** permite a los usuarios crear todas las suscripciones que deseen y escoger de qué tipo sería, de Correo o FTP, además permite crear, editar o eliminar una suscripción.
- **Frecuencia:** permite a los usuarios escoger con qué frecuencia la suscripción creada debe llegar al Correo o a la dirección FTP. La frecuencia puede ser diario, semanal o mensual, además de escoger a qué hora les debe llegar.
- **Categorías:** permite a los usuarios crear, editar o eliminar todas las categorías que deseen, además se pueden escoger cuáles reportes pertenecerán a esa categoría.

2.2 Requisitos de Software

Son un punto clave en el desarrollo de las aplicaciones informáticas. Un gran número de proyectos de software fracasan debido a una mala definición, especificación o administración de requisitos. Factores tales como requisitos incompletos o mal manejo de los cambios de los requisitos llevan a proyectos completos al fracaso total. (31)

2.2.1 Requisitos Funcionales

Los requisitos funcionales son condiciones que el sistema debe cumplir. El módulo Administrador de Reportes debe efectuar los requisitos funcionales que a continuación se describen:

RF 1: Actualizar la lista de plantillas existentes en el sistema

- **Descripción:** el sistema debe permitir actualizar la lista de las plantillas registradas. (Refrescar).
- **Entrada:** solicitud de actualización de la lista de plantillas registradas.
- **Salida:** lista de plantillas registradas en el sistema.

RF 2: Buscar plantilla registrada en el sistema

- **Descripción:** el sistema debe permitir buscar una plantilla por su nombre, escribiendo una parte

del nombre o el nombre completo.

- **Entrada:** criterios de búsqueda nombre de la plantilla (cadena de texto).
- **Salida:** listado de plantillas.

RF 3: Eliminar plantilla

- **Descripción:** el sistema debe permitir eliminar la plantilla seleccionada.
- **Entrada:** identificador de la plantilla que se desea eliminar (entero).
- **Salida:** mensaje (cadena de texto).

RF 4: Renombrar plantilla

- **Descripción:** el sistema deberá permitir cambiar el nombre a una plantilla previamente definida.
- **Entrada:** nombre de la plantilla (cadena de texto).
- **Salida:** nombre de la plantilla modificada (cadena de texto).

RF 5: Mostrar las propiedades generales de la plantilla

- **Descripción:** el sistema debe permitir, una vez seleccionada la plantilla, visualizar un conjunto de información asociada a la configuración básica de la plantilla, así como sus fechas de creación y actualización.
- **Entrada:** solicitud de visualización de las propiedades generales de la plantilla.
- **Salida:** nombre de la plantilla (cadena de texto), fecha de creación y modificación de la plantilla (fecha).

RF 6: Actualizar la lista de reportes existentes en el sistema

- **Descripción:** el sistema debe permitir actualizar la lista de reportes. (Refrescar).
- **Entrada:** solicitud de actualización de la lista de reportes registrados.
- **Salida:** lista de reportes registrados en el sistema.

RF 7: Eliminar reporte

- **Descripción:** el sistema debe permitir eliminar el reporte seleccionado.
- **Entrada:** identificador del reporte que se desea eliminar (entero).
- **Salida:** mensaje (cadena de texto).

RF 8: Renombrar reporte

- **Descripción:** el sistema deberá permitir cambiar el nombre a un reporte previamente definido.
- **Entrada:** nombre del reporte (cadena de texto).

- **Salida:** nombre del reporte modificado (cadena de texto).

RF 9: Copiar reporte hacia una categoría

- **Descripción:** el sistema debe permitir realizar copia idéntica de un reporte hacia una categoría especificada.
- **Entrada:** ubicación hacia la que se desea copiar el reporte (cadena de texto) e identificador del reporte que se desea copiar (entero).
- **Salida:** lista de reportes actualizada (lista).

RF 10: Mover reporte hacia una ubicación

- **Descripción:** el sistema debe permitir la copia de un reporte hacia una ubicación destino especificado y la eliminación del reporte en la ubicación origen.
- **Entrada:** ubicación hacia la que se desea copiar el reporte (cadena de texto) e identificador del reporte que se desea copiar (entero).
- **Salida:** lista de reportes actualizada (lista).

RF 11: Buscar reporte registrado en el sistema

- **Descripción:** el sistema debe permitir buscar un reporte por su nombre, escribiendo una parte del nombre o el nombre completo.
- **Entrada:** criterios de búsqueda (nombre del reporte), cadena de texto.
- **Salida:** listado de reportes.

RF 12: Mostrar las propiedades generales del reporte

- **Descripción:** el sistema debe permitir, una vez seleccionado el reporte, visualizar un conjunto de información asociada a la configuración básica del reporte, así como sus fechas de creación y actualización.
- **Entrada:** solicitud de visualización de las propiedades generales del reporte.
- **Salida:** nombre del reporte (cadena de texto), fecha de creación y modificación del reporte (fecha).

RF 13: Actualizar la lista de suscripciones existentes en el sistema

- **Descripción:** el sistema debe permitir actualizar la lista de suscripciones registradas. (Refrescar).
- **Entrada:** solicitud de actualización de la lista de suscripciones registradas.
- **Salida:** lista de suscripciones registradas en el sistema.

RF 14: Crear suscripción

- **Descripción:** el sistema debe permitir crear una tarea programa de ejecución en el sistema para el envío automático de reportes a través de correo electrónico o almacenamiento en un servidor de ficheros externo
- **Entrada:**

  Recurso de entrega (lista desplegable),

  Datos de configuración para correo electrónico: para, asunto, texto (cadena de texto).

  Datos de configuración para servidor de ficheros: ruta FTP, usuario y contraseña (cadena de texto).

  Condición de ejecución de la tarea programada: fecha y hora de inicio, frecuencia (tipo fecha y hora).

  Frecuencia diaria: ciclo de repetición diario.

  Frecuencia semanal: ciclo de repetición semanal y lista de días de la semana a ejecutarse.

  Frecuencia mensual: lista de meses, días del mes (lista).

  Datos de configuración avanzada: hora de inicio y fin (tipo hora).
- **Salida:** lista de suscripciones actualizada (lista).

RF 15: Editar suscripción

- **Descripción:** carga la vista de suscripción registrada en el sistema con la posibilidad de actualizar los valores anteriores por nuevos ingresados por el actor.
- **Entrada:** valores configurados en la suscripción.
- **Salida:** suscripción actualizada.

RF 16: Eliminar suscripción

- **Descripción:** el sistema debe permitir eliminar la suscripción seleccionada.
- **Entrada:** identificador de la suscripción que se desea eliminar (entero).
- **Salida:** mensaje (cadena de texto).

RF 17: Buscar suscripción registrada en el sistema

- **Descripción:** el sistema debe permitir buscar una suscripción por su nombre, escribiendo una parte del nombre o el nombre completo.
- **Entrada:** criterios de búsqueda (nombre de la suscripción), cadena de texto.
- **Salida:** listado de suscripciones.

RF 18: Adicionar categoría

- **Descripción:** el sistema debe permitir la adición de una nueva categoría de reportes que dará soporte a una mejor organización de los reportes.
- **Entrada:** nombre y descripción de la categoría (cadena de texto).
- **Salida:** lista de categoría actualizada (lista).

RF 19: Eliminar categoría

- **Descripción:** el sistema debe permitir eliminar la categoría seleccionada.
- **Entrada:** identificador de la categoría que se desea eliminar (entero).
- **Salida:** mensaje (cadena de texto).

RF 20: Editar categoría

- **Descripción:** el sistema debe permitir la actualización de los datos de una categoría registrada.
- **Entrada:** nombre y descripción de la categoría (cadena de texto).
- **Salida:** lista de categoría actualizada (lista).

2.2.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. El Administrador de Reportes cuenta con los siguientes requerimientos no funcionales:

RNF 1: Tipo de usuario final: el usuario que interactúe con la aplicación debe ser una persona autorizada. Además debe poseer conocimientos que le permitan manejar la aplicación y la computadora.

RNF 2: Finalidad: el objetivo que persigue la aplicación es generar de forma sencilla reportes dinámicos de una base de datos según las necesidades del usuario.

RNF 3: Software requerido para desplegar y utilizar la aplicación

Servidor para instalar la aplicación: el servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos de software:

- Sistema Operativo (SO): GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
- Paquetes: apache2, php5, libapache2-mod-php5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite.
- Usuario con privilegios de administración del SO.

Servidor para instalar la de Base de Datos: el servidor donde se instalará la Base de Datos debe cumplir con los siguientes requisitos de software (puede ser el mismo donde estará la aplicación):

- SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.

- PostgreSQL versión 8.3 o superior.
- pgAdmin III o algún administrador para PostgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por md5.

RNF 4: Hardware requerido para desplegar y utilizar la aplicación

PC Cliente

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 256MB.

PC Servidor

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 1Gb.
- Disco Duro con 10Gb de capacidad para instalar el sistema.

RNF 5: Lenguaje y marco de trabajo para el desarrollo del sistema del lado del servidor: el sistema debe ser implementado en el lenguaje de programación PHP versión 5.2 o superior. Como *framework* de desarrollo Symfony, el cual propone una arquitectura en tres capas: el modelo, la vista y el controlador.

RNF 6: Lenguaje y marco de trabajo para el desarrollo del sistema del lado del cliente: Como *framework* para el diseño de las interfaces ExtJS, la cual es una librería JavaScript que contiene interfaces de usuario personalizable usando metodologías como AJAX y peticiones DOM.

RNF 7: Eficiencia: El tiempo máximo para obtener un reporte generado a alguno de los diferentes formatos no debe sobrepasar los 5 minutos. El tiempo promedio para ver las propiedades de los reportes y las plantillas es de 10 segundos. El sistema debe permitir que existan al menos 100 usuarios conectados de forma simultánea. Sin embargo la eficiencia depende de la velocidad con la que se pueda conectar a la base de datos y la cantidad de información que contenga.

2.3 Modelo de Casos de Uso del Sistema

El diagrama de casos de uso documenta el comportamiento de un sistema desde el punto de vista del usuario. Por tanto los casos de uso determinan los requisitos funcionales del sistema, los cuales representan las funcionalidades que un sistema puede ejecutar (32). Luego de realizar el análisis se definieron 20 requisitos funcionales agrupados en 4 casos de uso, los cuales el módulo Administrador de

Reportes del Generador Dinámico de Reportes V (2.0) debe cumplir. Los patrones de Casos de Uso que están presentes son CRUD (son las siglas de Create, Read, Update y Delete) Total y Parcial. El primero se evidencia en los casos de uso Gestionar Categoría y Gestionar Suscripción, pues en ambos casos están presentes las cuatro funciones. El segundo se evidencia en Administrar Plantilla y Administrar Reportes, debido a que falta alguna de las 4 funciones mencionadas anteriormente.

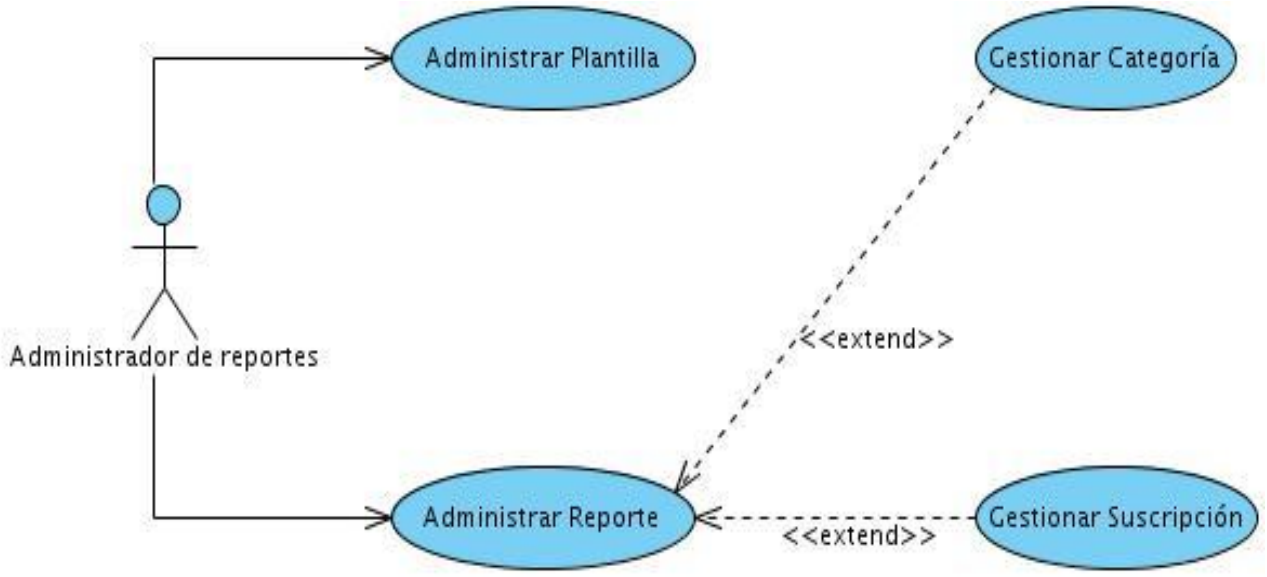


Figura 2: Diagrama de Casos de Uso del Sistema

- **Gestionar Categoría**: permite adicionar, eliminar y modificar categorías, lo cual posibilita organizar los reportes por categorías.
- **Gestionar Suscripción:** permite adicionar, eliminar, buscar y modificar suscripciones a un reporte creado.
- **Administrar Plantilla:** permite renombrar, buscar y eliminar una plantilla del sistema.
- **Administrar Reporte:** permite renombrar, buscar, copiar, mover y eliminar un reporte del sistema.

Matriz de Trazabilidad

Requisitos Funcionales	Gestionar Categoría	Gestionar Suscripción	Administrar Plantilla	Administrar Reporte
Adicionar categoría	X			
Editar categoría	X			
Eliminar categoría	X			
Adicionar suscripción		X		
Editar suscripción		X		
Eliminar suscripción		X		
Buscar suscripción		X		
Actualizar listado de suscripciones		X		
Buscar plantilla			X	
Eliminar plantilla			X	
Renombrar plantilla			X	
Actualizar listado de plantillas			X	
Mostrar propiedades de la plantilla			X	
Buscar reporte				X
Eliminar reporte				X
Renombrar reporte				X
Copiar reporte				X
Mover reporte				X
Actualizar listado de reportes				X
Mostrar propiedades de la reportes				X

Tabla 2: Matriz de Trazabilidad

La matriz de trazabilidad permite conocer cuáles son los requisitos funcionales que pertenecen a cada caso de uso. A continuación se muestra la especificación del Caso de Uso Gestionar Suscripción, se puede consultar el expediente de proyecto del GDR versión 2.0.para ver las especificaciones de los demás casos de uso.

Especificación del Caso de Uso: Gestionar Suscripción.

Objetivo	Gestionar Suscripción
Actores	Administrador del Reporte
Resumen	El caso de uso se inicia cuando el actor solicita gestionar las suscripciones de los reportes. El caso de uso permite crear nueva suscripción, editarla y eliminarla. Finaliza al crear, editar o eliminar una suscripción.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Existe al menos un reporte.
Postcondiciones	Son gestionadas las suscripciones de los reportes.

	Actor	**Sistema**
1.	Presiona clic derecho sobre alguna suscripción.	
2.		Muestra un menú contextual, con tres opciones. • Adicionar (Correo o FTP) • Editar (Correo o FTP) • Eliminar
3.	Selecciona una de las opciones.	
4.		En caso que el actor deje de hacer foco sobre el menú contextual, ver paso 1 del Flujo Alterno. Permite: Adicionar una nueva suscripción de reporte, ver Sección: Adicionar suscripción. Editar una suscripción existente, ver Sección: Editar

		suscripción. Eliminar una suscripción: ver Sección: Eliminar suscripción.
5.		Termina el caso de uso.
Flujos Alternos		
	Actor	**Sistema**
1.		Se cierra el menú contextual.
Sección 1: "Adicionar Suscripción"		
Flujo básico		
	Actor	**Sistema**
1.		Muestra una interfaz para introducir los datos (nombre de la suscripción, tipo de suscripción, seleccionar el reporte y el formato, si escoge suscripción de correo entonces debe introducir el correo, asunto y un mensaje, si es suscripción de FTP entonces introduce ip, usuario y contraseña) de la nueva suscripción que se desea adicionar. Permite: Presionar el botón de "Cancelar", ver Flujo Alterno 1.
2.	Registra los datos en la interfaz y presiona el botón "Aceptar".	
3.		Valida que los campos obligatorios no estén vacíos. En caso de que algún campo esté vacío, ver Flujo Alterno 2.
4.		Registra en el sistema la nueva suscripción.
5.		Actualiza el listado de las suscripciones con la nueva suscripción registrada.
Flujos alternos		
	Actor	**Sistema**

1.		Cancela la solicitud de adicionar una nueva suscripción y cierra la interfaz.
2.		Muestra un mensaje al actor que el campo es obligatorio.
Sección 2: "Editar Suscripción"		
Flujo básico		
	Actor	**Sistema**
1.		Muestra una interfaz cargando los datos asociados a la suscripción Permite: Presionar el botón de "Cancelar", ver Flujo Alterno 1.
2.	Actualiza los datos en la interfaz y presiona el botón "Aceptar".	
3.		Valida que los campos obligatorios no estén vacíos. En caso de que algún campo esté vacío, ver Flujo Alterno 2.
4.		Registra en el sistema la suscripción con los datos actualizados.
5.		Actualiza el listado de las suscripciones con la nueva suscripción registrada.
Flujos Alternos		
	Actor	**Sistema**
1.		Cancela la solicitud de editar suscripción y cierra la interfaz.
2.		Muestra un mensaje al actor informando que el campo es obligatorio.
Sección 3: "Eliminar Suscripción"		
Flujo básico		

	Actor	Sistema
1.		Muestra un mensaje solicitando confirmación al actor sobre la eliminación de una suscripción y todo su contenido. Permite: Seleccionar el botón “Cancelar”, ver paso 1 del Flujo Alterno.
2.	Selecciona eliminar la suscripción.	
3.		Elimina la suscripción de la lista de suscripciones existentes en el sistema.
4.		Actualiza el listado de suscripción con la eliminación de la suscripción.
Flujos alternos		
	Actor	**Sistema**
1.		Ignora la acción de eliminación de la suscripción.
Sección 4: “Buscar Suscripción”		
Flujo básico		
	Actor	**Sistema**
1.		Muestra una interfaz para introducir los datos (nombre de la suscripción) de la suscripción que desea buscar.
2.		Busca en la lista de suscripciones registradas en el sistema, la suscripción por su nombre. En caso de que no exista se muestra una lista vacía.
Flujos alternos		

Interfaces de Usuario del Caso de Uso Gestionar Suscripción

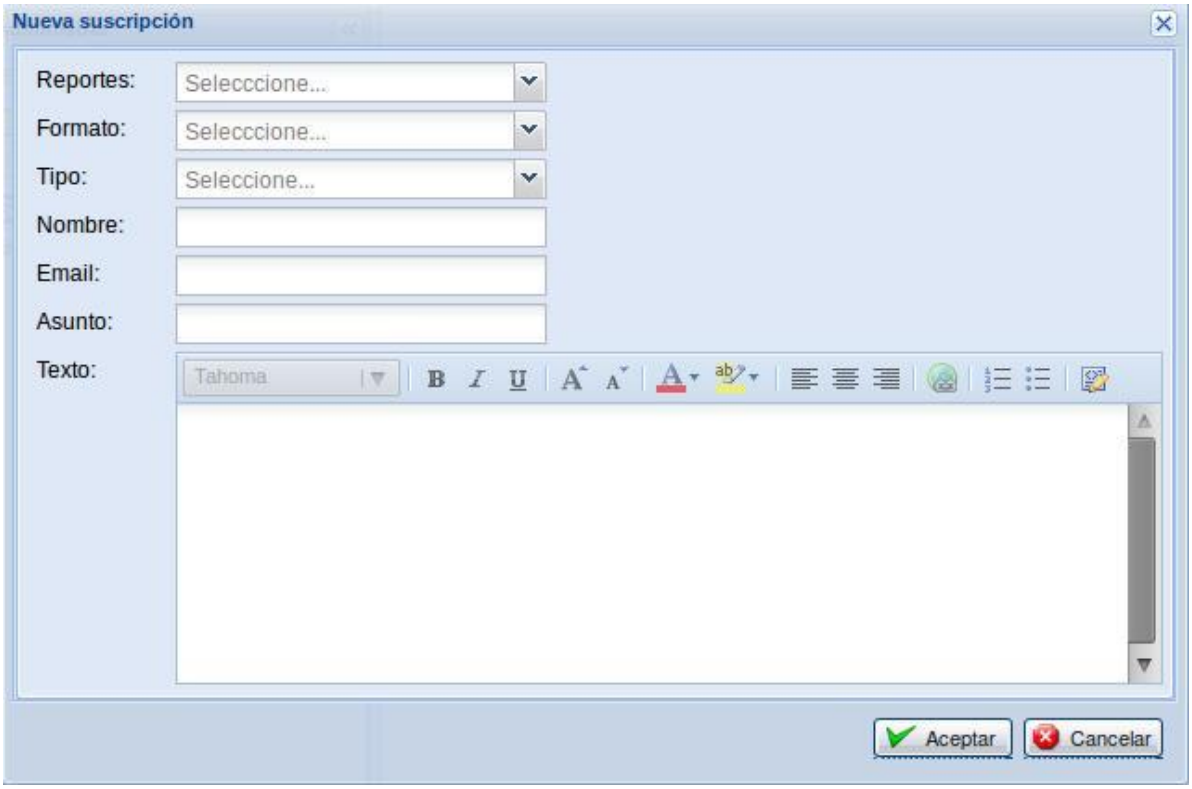


Figura 3: Adicionar suscripción de correo

Figura 4: Adicionar suscripción de FTP

Figura 5: Adicionar frecuencia

2.4 Patrones

2.4.1 Patrones de Diseño GRASP

Controlador: Symfony aplica este patrón, pues mantiene una estructura organizada desde el report_generator.php hasta las actions. Cada clase es esta capa tiene su responsabilidad y es única. En el módulo se utiliza este patrón, por ejemplo la clase addCategory tiene la responsabilidad de controlar el flujo de eventos asociado a adicionar una categoría.

Bajo Acoplamiento: se evidencia en la capa Modelo, pues existe poca dependencia entra las clases de acceso a datos y las de abstracción de datos, lo que permite mayor reutilización. En el módulo se pueden modificar las clases del modelo sin que se afecten las del controlador y viceversa porque éstas son acciones independientes.

Alta Cohesión: Symfony organiza el trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases que tienen una alta cohesión. Por ejemplo la clase addSuscription tiene la responsabilidad de definir las acciones para adicionar suscripciones y a su vez colabora con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las suscripciones.

Experto: este patrón se evidencia en la capa modelo donde existen dos tipos de clases, las de abstracción de datos y las de acceso a datos. Symfony utiliza Propel como ORM, el cual crea 4 clases. Las de acceso a datos que trabajan directamente con la base de datos utilizando Propel y las de abstracción de datos que son las que tienen los atributos necesarios para realizar dicha función. Por tanto deben implementar la responsabilidad de realizar las acciones directamente con la base de datos y ahí es donde se aplica el patrón. Por ejemplo la clase baseSuscription no conoce los atributos para comenzar a interactuar con la base de datos, tan sólo le avisa a la baseSuscriptionPeer lo que tiene que hacer, pues esta si tiene todos los datos necesarios para ejecutar la acción.

2.4.2 Patrones de Diseño GOF

Front Controller (Controlador frontal): se evidencia en el report_generator.php, pues es el punto de entrada a la aplicación que recibe todas las peticiones y decide el flujo a seguir, carga la configuración y determina la acción a ejecutarse. Es único para cada aplicación y las acciones que incluyen el código específico del contenido de cada página.

Decorador: está presente en la layout, la cual contiene todo el código HTML y a su vez decora todas las plantillas a usar, en dependencia de la petición del usuario.

Command (Orden): en el framework Symfony se utiliza este patrón, el cual está representado por las acciones, por tanto cada acción, encapsula una petición en un objeto.

2.4.3 Patrones de Arquitectura

Modelo-Vista-Controlador: este patrón se utiliza para modelar los diferentes diagramas los cuales están agrupados por paquetes que representan las capas, en la que cada una realiza una función específica. En la Vista se encuentran todas las interfaces con las que el usuario puede interactuar. En el Controlador se encuentran todas las acciones, con las cuales se le da respuesta a las peticiones del usuario que llegan a través del controlador frontal. En el Modelo se encuentran las tablas de la base de datos, en este caso se

usa Propel como ORM, el cual crea cuatro clases por cada tabla. Este patrón se evidencia en la imagen del diagrama de clases del diseño del Caso de Uso Gestionar Suscripción.

Cliente – Servidor: este patrón se evidencia, pues se separa al cliente del servidor. Lo que significa que el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

2.5 Modelo de Diseño

El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución que se prepara para la implementación y prueba del sistema. Tiene como finalidad:

- Transformar los requisitos en un diseño del sistema.
- Evolucionar una arquitectura sólida para el sistema.
- Adaptar el diseño para que se ajuste al entorno de implementación.

2.6 Diagramas de Clases del Diseño

Un diagrama de clases describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los cuales son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema. Además de los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

La siguiente imagen muestra el diagrama de clases del diseño del Caso de Uso Gestionar Suscripción. En el expediente de proyecto del GDR versión 2.0 y en los anexos se encuentran los restantes diagramas.

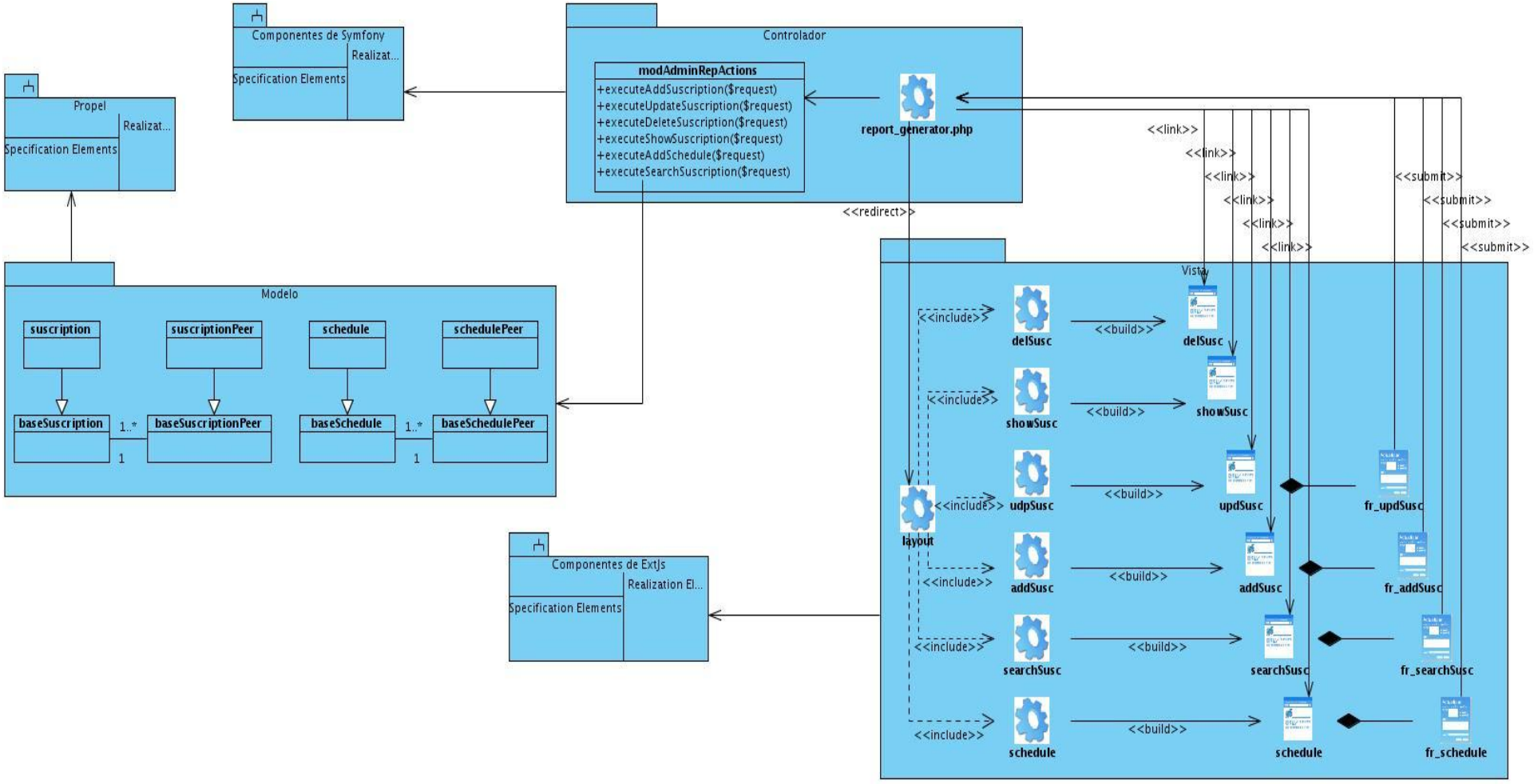


Figura 6: Diagrama de Clases del Diseño del Caso de uso Gestionar Suscripción

- **Controlador:** esta capa es la que recibe las entradas, traducidas a solicitudes de servicios para el modelo. Es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario.
- **modAdminRepActions:** esta clase contiene todas las acciones en dependencia del caso de uso que se trate, en este caso contiene todas las acciones referentes al Caso de Uso Gestionar Suscripción.
- **report_generator.php:** esta clase es el punto de entrada a la aplicación, es la que recibe las peticiones y decide el flujo que debe seguir para dar respuesta a la petición del usuario.
- **Modelo:** esta capa contiene todas las clases de la base de datos, como se utiliza Propel como ORM, se obtienen 4 clases por cada tabla de la base de datos. Cada una con sus funciones específicas, en este caso serían suscription, suscriptionPeer, baseSuscription y baseSuscriptionPeer. Además de las clases para que el usuario seleccione con que frecuencia desea que las suscripciones lleguen al correo o FTP, las cuales son shedule, shedulePeer, baseShedule y baseShedulePeer.
- **suscription, suscriptionPeer, shedule, shedulePeer:** son clases de acceso a datos, las cuales son responsables de interactuar con las clases de abstracción de datos, devuelven los objetos que necesitan los controladores en su forma original.
- **baseSuscription, baseSuscriptionPeer, baseShedule, baseShedulePeer:** son clases de abstracción de datos, las cuales son encargadas de la abstracción de datos y responsables de realizar las operaciones con la base de datos.
- **Vista:** en esta capa se encuentran las interfaces de usuario de la aplicación, con las cuales los usuarios pueden interactuar.

2.7 Diagramas de Interacción

Los diagramas de interacción modelan el comportamiento dinámico del sistema y el flujo central en una operación. Describe la interacción entre objetos, los cuales interactúan a través de mensajes para cumplir ciertas tareas. Existen dos tipos de diagramas de interacción: secuencia y colaboración.

2.7.1 Diagrama de Secuencia

Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. El diagrama de secuencia contiene detalles de la implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos. (23)

Las imágenes siguientes muestran los diagramas de secuencia del Caso de Uso Gestionar Suscripción. En el expediente de proyecto del GDR versión 2.0 se encuentran los restantes diagramas de secuencia.

El escenario eliminar suscripción representa las acciones que se ejecutan para realizar dicha acción. El usuario selecciona la suscripción que desea eliminar, luego el sistema consulta la base de datos para ver si existe esa suscripción, en caso de no existir u ocurra algún problema se muestra un mensaje al usuario indicándole lo sucedido. En caso de que exista en la base de datos se elimina la suscripción.

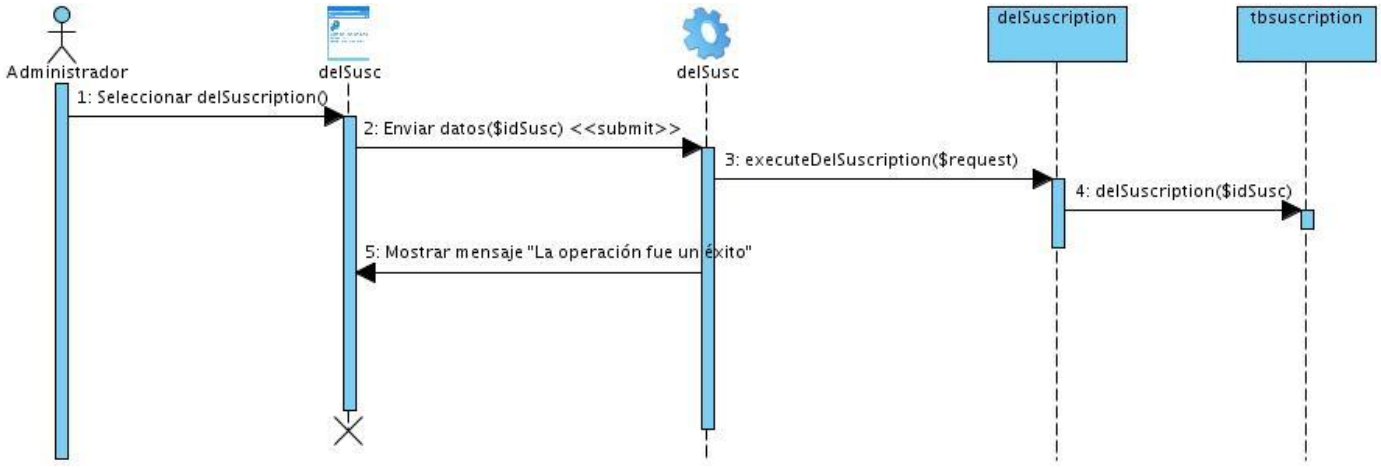


Figura 7: Diagrama de Secuencia de Eliminar Suscripción del Caso de Uso Gestionar Suscripción

El escenario adicionar suscripción representa las acciones que se ejecutan para realizar dicha acción. El usuario introduce los datos del formulario, luego el sistema valida los datos y verifica que no exista ningún campo vacío. En caso de que ocurra alguno de estos casos muestra un mensaje al usuario indicándole lo sucedido. Luego el sistema consulta la base de datos para ver si existe esa suscripción, en caso de no existir se adiciona la suscripción, en caso contrario se muestra un mensaje al usuario.

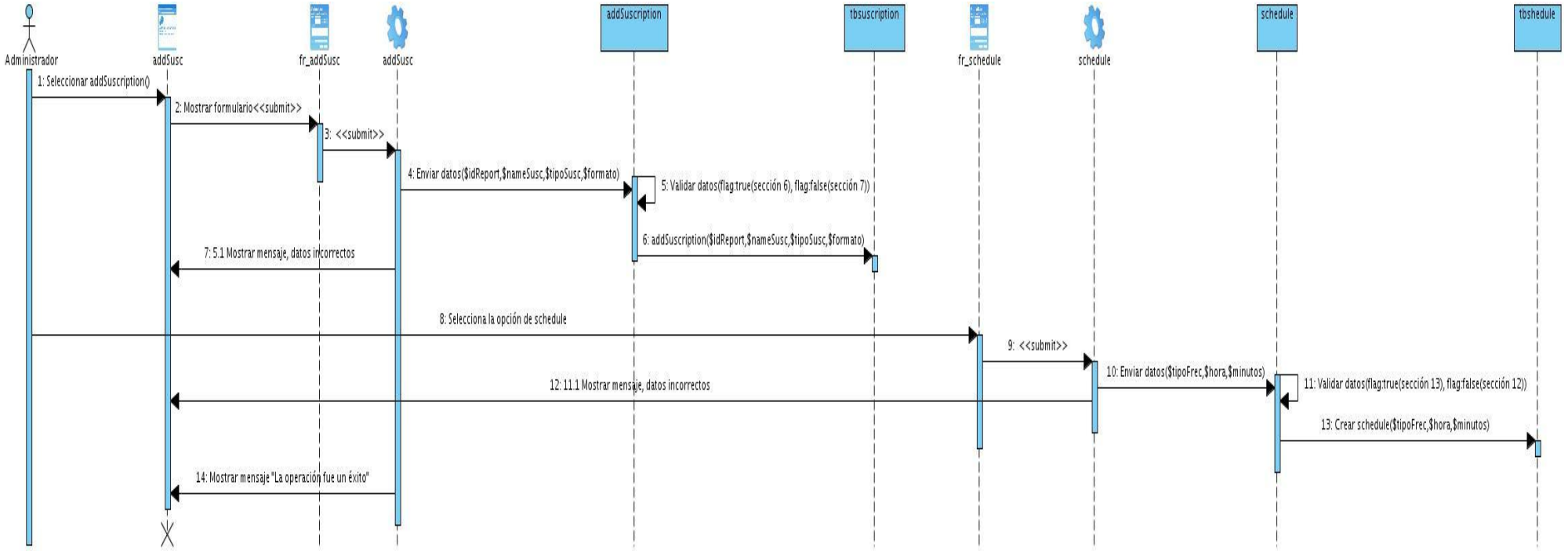


Figura 8: Diagrama de Secuencia de Adicionar Suscripción del Caso de Uso Gestionar Suscripción

El escenario editar suscripción representa las acciones que se ejecutan para realizar dicha acción. El usuario introduce los datos que desee modificar en el formulario, luego el sistema valida los datos y verifica que no exista ningún campo vacío. En caso de que ocurra alguno de estos casos muestra un mensaje al usuario indicándole lo sucedido. Luego el sistema modifica la suscripción en la base de datos.

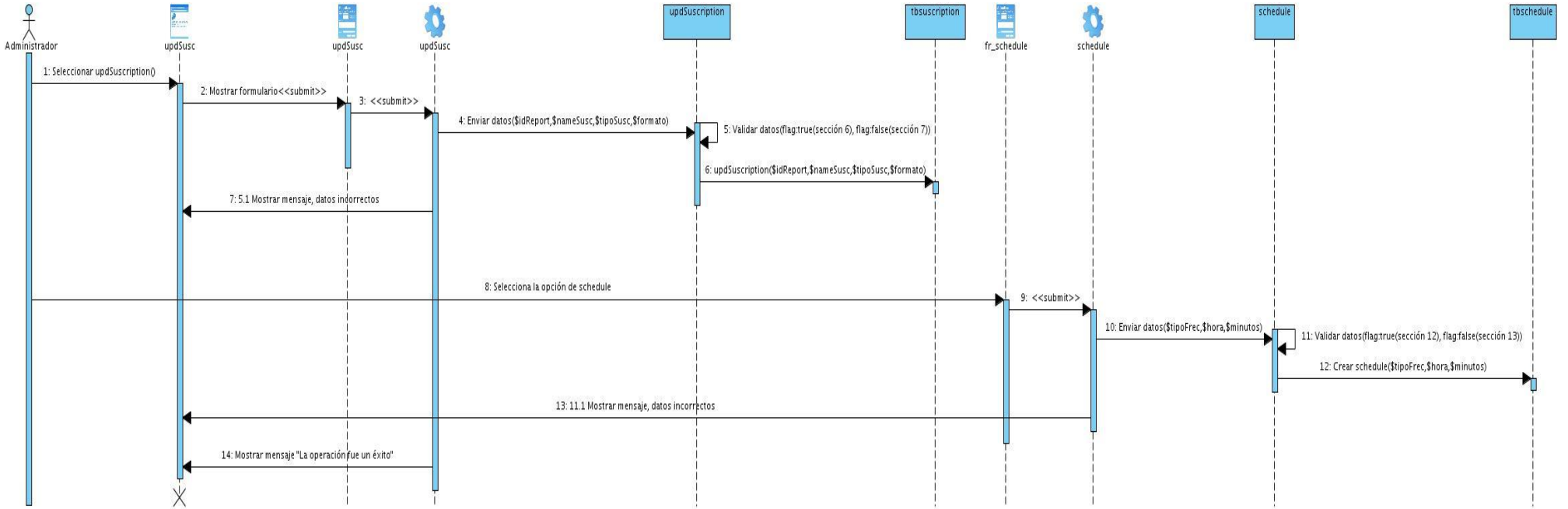


Figura 9: Diagrama de Secuencia de Editar Suscripción del Caso de Uso Gestionar Suscripción

2.8 Modelo de Datos

Esta vista presenta el modelo de datos utilizado con la descripción de las tablas más importantes que componen la base de datos del Generador Dinámico de Reportes.

Módulo Administrador de Reportes: presenta las entidades relacionadas con las suscripciones y programación de entrega de los reportes, información de los reportes y las planillas que se encuentran en cada categoría.

- **tbSubscription**: almacena la información de las suscripciones para entregar los reportes.
- **tbSchedule**: almacena la información de las tareas programadas.
- **tbCategory**: almacena las categorías creadas a las que se asociarán los reportes.
- **tbReport**: almacena la definición de los reportes diseñados.
- **tbTemplate**: contiene la definición de las plantillas diseñadas.

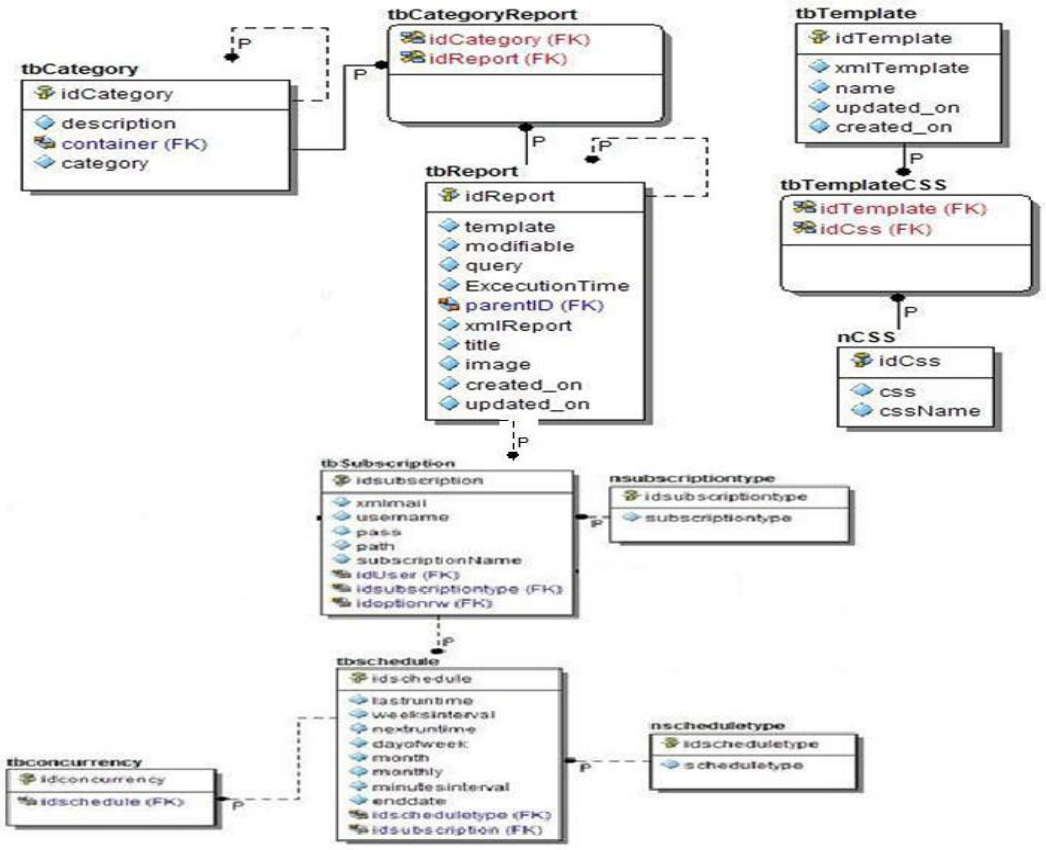


Figura 10: Modelo de Datos

2.9 Modelo de Despliegue

El Diagrama de Despliegue es un diagrama que se utiliza para modelar el hardware utilizado en la implementación del sistema y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. (33)

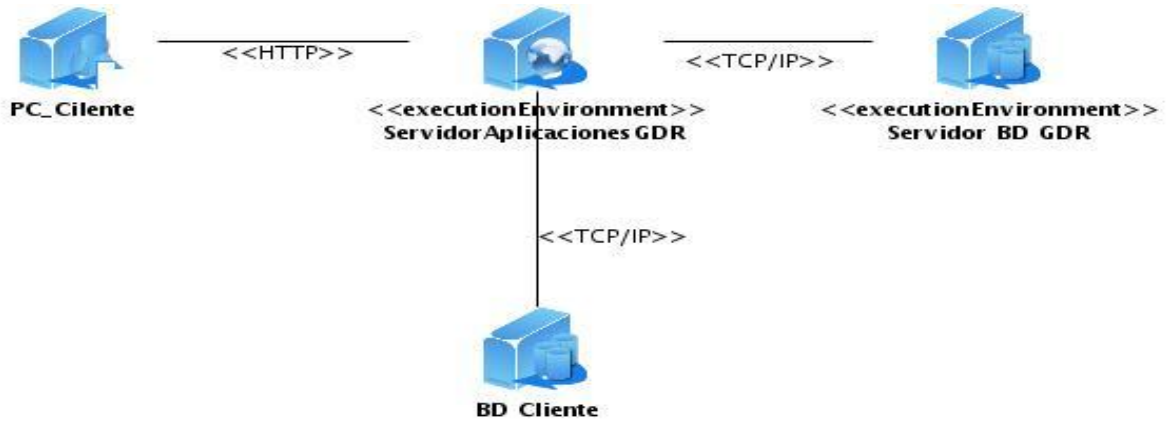


Figura 11: Modelo de Despliegue

- **PC_Cliente:** computadora desde donde se ejecuta la url, donde está publicada la aplicación.
- **Servidor Aplicaciones GDR:** computadora donde está publicada la aplicación.
- **Servidor BD GDR:** contiene toda la información del GDR.
- **BD Cliente:** contiene toda la información del cliente.
- **HTTP:** se utiliza para conectar la computadora del cliente y el servidor donde está la aplicación.
- **TCP/IP:** protocolo para conectar el servidor de aplicaciones con las bases de datos.

Conclusiones del Capítulo

En este capítulo se definieron 20 requisitos funcionales y 7 no funcionales, representando las funcionalidades que el sistema debe tener y cumplir. Los requisitos funcionales se agruparon en 4 casos de uso, los cuales se muestran en el diagrama de casos de uso del sistema. Se definieron los diferentes patrones de diseño (GRASP y GOF) y de arquitectura (M-V-C y Cliente Servidor) que se utilizarán. Se representó la estructura estática y dinámica del sistema, a través del diagrama de clases del diseño y el diagrama de secuencia. Además se muestra el diagrama de despliegue para representar el hardware utilizado en la implementación del módulo y el modelo de datos el cual contiene las tablas de la base de datos asociadas al módulo Administrador de Reportes.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

Luego de obtener el resultado del diseño, en este capítulo se realizan las actividades que se llevan a cabo durante la fase de implementación y pruebas. Entre las cuales se encuentran generar los artefactos pertenecientes a ambas fases. Se modela el sistema en términos de componentes, se especifican los tipos, niveles, métodos y casos de pruebas que se realizarán al módulo. Además se muestran ejemplos de las implementaciones más relevantes.

Objetivos de la Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño. Tiene como finalidad:

- Definir la organización del código en términos de los subsistemas de implementación, organizados en capas.
- Implementar los elementos del diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- Probar y desarrollar componentes como unidades.
- Integrar los resultados producidos por los implementadores individuales o equipos en un sistema ejecutable.

3.1 Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema, muestran y organizan las relaciones de dependencia entre los componentes y subsistemas que lo conforman. Los componentes representan los elementos de un modelo dentro de un paquete, como son las clases en el modelo de diseño. Normalmente los diagramas de componentes contienen: componentes, interfaces, relaciones de dependencia, generalización, asociación y realización, paquetes y subsistemas.

La siguiente imagen muestra el diagrama de componentes del Caso de Uso Gestionar Suscripción. En el expediente de proyecto del GDR versión 2.0 y en los anexos se encuentran los restantes diagramas.

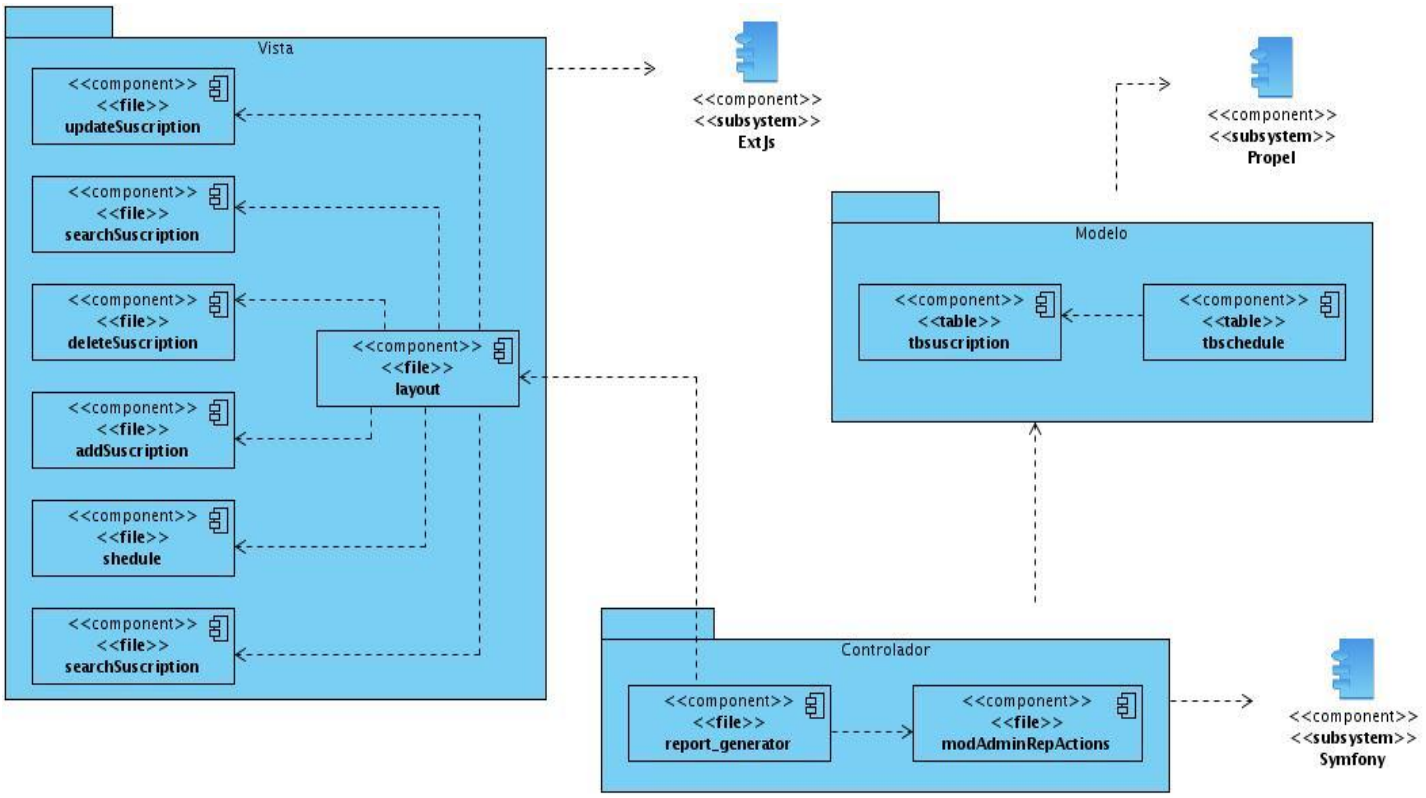


Figura 12: Diagrama de Componentes del Caso de Uso Gestionar Suscripción

- **Vista:** contiene las interfaces de usuario asociadas al caso de uso Gestionar Suscripción, convertidas en componentes de tipo <<file>>. La Vista tiene relación de dependencia con el <<subsystem>> ExtJS.
- **Controlador:** contiene las acciones asociadas al caso de uso Gestionar Suscripción, con las cuales se le da respuesta a las peticiones del usuario que llegan a través de report_generator, convertidas en componentes de tipo <<file>>. El Controlador tiene relación de dependencia con el <<subsystem>> Symfony.
- **Modelo:** contiene las tablas de la base de datos asociadas al caso de uso Gestionar Suscripción, convertidas en componentes de tipo <<table>>. El Modelo tiene relación de dependencia con el <<subsystem>> Propel.
- **ExtJS:** paquete que se utiliza en la capa Vista para el diseño de las interfaces, convertido en un componente de tipo <<subsystem>>.

- **Symfony:** paquete que se utiliza en la capa Controlador para la implementación de las acciones, convertido en un componente de tipo <<subsystem>>.
- **Propel:** paquete que se utiliza en la capa Modelo para convertir las tablas de la base de datos en clases, convertido en un componente de tipo <<subsystem>>.

3.2 Mecanismos de Implementación

- El sistema se desarrollará utilizando la arquitectura Modelo-Vista-Controlador.
- La parte cliente se implementará haciendo uso generalizado del *framework* de desarrollo de interfaces de usuario con *JavaScript* ExtJS, versión 3.3.0.
- La parte del servidor se implementará en el lenguaje de programación PHP5, utilizando Symfony 1.4.8 como *framework* de desarrollo para dicho lenguaje.
- La capa del modelo (acceso a datos) se genera utilizando Propel como ORM incluido en la versión de Symfony a utilizar.
- El intercambio de información entre el cliente y el servidor se hace únicamente en formato JSON.

3.3 Implementaciones Significativas

La imagen muestra el código para eliminar una suscripción dado el identificador. Si encuentra la suscripción en la base de datos la elimina, en caso contrario muestra un mensaje al usuario para indicarle que no se pudo eliminar.

```php
<?php
class deleteSubscriptionAction extends sfAction
{
    public function execute($request)
    {
        try
        {
            $id = $request->getParameter('id');
            $suscription = SubscriptionPeer::retrieveByPK($id);
            $suscription->delete();
                return $this->renderText("{success:true}");
        }
        catch (Exception $e)
        {
            return $this->renderText("{success:false,Message:'". $e->getMessage()."'}");
        }
    }
}
?>
```

Figura 13: Acción Eliminar Suscripción

La imagen muestra el código para mostrar las suscripciones, lo cual permite que el usuario pueda ver todas las suscripciones existentes en el sistema. Cada vez que se adiciona una suscripción se actualiza la lista de suscripciones existentes por tanto se ejecuta este método.

```php
<?php
class showSuscriptionAction extends sfAction {
    public function execute($request) {
        $suscriptions = Subscription::getAll();
        $json = array();
        foreach ($suscriptions as $suscription) {
            $array = array();
            $array['id'] = $suscription->getIdsubscription();
            $array['text'] = $suscription->getSubscriptionname();
            $array['xmlmail'] = $suscription->getXmlmail();
            $array['idReport'] = $suscription->getReport();
            $array['allowDrag'] = false;
            $array['allowDrop'] = false;
            $array['leaf'] = true;
            $array['iconCls'] = 'rss';
            $json[] = $array;
        }
            return $this->renderText(json_encode($json));
    }
}
?>
```

Figura 14: Acción Mostrar Suscripciones

La imagen muestra el código para mostrar las plantillas, lo cual permite que el usuario pueda ver todas las plantillas existentes en el sistema. Cada vez que se adiciona una plantilla se actualiza la lista de plantillas existentes por tanto se ejecuta este método.

```php
<?php
class showTemplateAction extends sfAction{

    public function  execute($request) {
        $templates = Template::getAll();
        $json = array();
        foreach ($templates as $template) {
            $array = array();
            $array['id'] = $template->getIdtemplate();
            $array['text'] = $template->getName();
            $array['allowDrag'] = false;
            $array['allowDrop'] = false;
            $array['leaf'] = true;
            $array['iconCls'] = 'template';
            $json[] = $array;
        }
            return $this->renderText(json_encode($json));
```

Figura 15: Acción Mostrar Plantillas

3.4 Pruebas

La prueba es un elemento crítico para la garantía de la calidad del software, es el proceso que permite verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una de las fases del desarrollo de software consistente en probar las aplicaciones construidas. Las pruebas de software se integran dentro de las diferentes fases del ciclo del software dentro de la Ingeniería de Software. (34)

3.4.1 Nivel de Prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas: Prueba de desarrollador, unidad, integración, sistema y aceptación (35). En este caso se aplicarán las pruebas de desarrollador, la cual está diseñada e implementada por el equipo de desarrollo, además incluye a las pruebas unitarias y funcionales.

3.4.2 Tipos de Pruebas

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el módulo Administrador de Reportes cumple con todas los requisitos identificados durante el proceso de análisis. Dentro de los tipos de pruebas se encuentran las pruebas unitarias, funcionales, regresión, aceptación e integración. A continuación se explican en qué consiste cada una de ellas para luego decidir las que se aplicarán al módulo.

Pruebas unitarias: se encargan de probar una clase en concreto, testeando cada uno de sus métodos y comprobando si dado unos parámetros de entrada, la salida es la esperada.

Pruebas funcionales: como su nombre indica, prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario.

Pruebas de regresión: son aquellas pruebas cuyo objetivo es comprobar por qué ha dejado de funcionar algo que ya funcionaba. El objetivo de las pruebas de regresión es no tener que “volver atrás”.

Pruebas de aceptación: son pruebas funcionales, pero vistas directamente desde el cliente. Digamos que son aquellas pruebas que demuestran al cliente que la funcionalidad está terminada y funciona correctamente.

Pruebas de integración: conjunto de pruebas unitarias, funcionales, regresión y aceptación que se realizan para probar el software. Incluye también comprobar que lo programado por los diferentes desarrolladores funciona en un entorno real. (36)

Se decide realizar las pruebas de tipo unitarias y funcionales. Las pruebas funcionales tienen como objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. Al realizar este tipo de pruebas, se pretende usar el sistema como lo usaría el usuario. Las pruebas unitarias tienen como objetivo asegurar la calidad del código entregado, además de que ayuda a definir los requerimientos y responsabilidades de cada método en cada clase probada.

3.4.3 Método de Prueba

Existen dos métodos de pruebas para realizar las pruebas funcionales: el método de caja negra y de caja blanca. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Las cuales se pueden aplicar mediante los Casos de Pruebas basados en los Casos de Uso. La prueba de la caja blanca comprueba los caminos lógicos del programa, las condiciones y bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado (35). El método de prueba seleccionado es pruebas de caja negra.

Las pruebas de Caja Negra permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Algunas ventajas de este tipo de pruebas:

- La prueba termina siendo imparcial porque el que diseñó el software y el que lo prueba son independientes.
- La persona que lo prueba no necesita conocimientos de programación.
- Las pruebas se realizan desde el punto de vista del usuario y no como programador, tomando en cuenta todas las funciones de cómo luce y su usabilidad. (37)

3.4.4 Casos de Pruebas

Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Caso de Prueba del Caso de Uso: Gestionar Suscripción

Descripción general: el caso de uso se inicia cuando el actor solicita gestionar las suscripciones de los reportes. El caso de uso permite crear una nueva suscripción, editarla y eliminarla. Finaliza al crear, editar o eliminar una suscripción.

Condiciones de ejecución: Exista al menos un reporte.

A continuación se muestra el conjunto de datos que se utilizó para realizar la prueba de caja negra a la interfaz adicionar suscripción asociada al caso de uso Gestionar Suscripción. En el expediente de proyecto del GDR versión 2.0 se encuentran los restantes casos de pruebas.

Escenario: Adicionar Suscripción

Escenario	Descripción	Formato	Reporte	Nombre	Tipo	Email	Asunto	Texto	Ip	Usuario	Contraseña
EC 1.1 Adicionar suscripción satisfactoriamente.	Se llenan todos los campos correctamente y todos son válidos.	HTML, PDF V	Seleccionar un reporte. V	Suscripción 1 V	Correo, FTP V	lslabrada@estudiantes.uci.cu V	Reporte V	Recibió el reporte 123 V	10.56.20.24 V	Administrador V	Administrador V
EC 1.2 Adicionar suscripción con campos en blanco.	Se dejan campos vacíos en la ventana de adicionar suscripción.	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
EC 1.3 Adicionar suscripción con campos no válidos.	Los campos introducidos no son válidos.	NA	NA	NA ¡”·$%&/(=)?¿	NA	NA 123.com	NA ¡”·$%&/(=)?¿	NA ¡”·$%&/(=)?¿	NA Mundo123	NA	NA
EC 1.4 Cancelar la opción de suscripción categoría.	Cancela la opción de adicionar suscripción.	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

FDiaria	Día semana	Día del mes	Mes del año	Hora	Minutos	Fecha Inicio	Fecha Fin	Inter valo	Resp. Sistema	Flujo Central
Frecuencia diaria, semanal o mensual	Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo	1 al 31	Enero, Febrero, Marzo, Abril, Mayo, Junio, Julio, Agosto, Septiembre, Octubre, Noviembre, Diciembre	13	34	03/09/2012	10/09/2012	3	Se adiciona la suscripción satisfactoriamente.	Se selecciona la opción Adicionar suscripción, se llenan los campos correctamente, luego se selecciona la opción Aceptar.
NA	NA	NA	NA	NA	NA	NA	NA	NA	Muestra un mensaje donde se indica que hay campos vacíos.	Se selecciona la opción Adicionar suscripción, se llenan los campos, pero se dejan campos vacíos y entonces se muestra un mensaje a al usuario para indicarle que hay campos vacíos y que son obligatorios.
NA	NA	NA	NA	NA	NA	NA	NA	NA	Muestra un mensaje indicando que los campos no son válidos.	Se selecciona la opción Adicionar suscripción, se llenan los campos, pero los campos son inválidos, se muestra un mensaje a al usuario para indicarle que los campos no son válidos.
NA	NA	NA	NA	NA	NA	NA	NA	NA	Cancela la opción.	Se selecciona la opción Adicionar suscripción, se llenan los campos, luego se selecciona la opción Cancelar.

Para validar el correcto funcionamiento del software se realizaron las pruebas de caja negra a través de los casos de pruebas basados en los casos de uso. Se ejecutaron 4 casos de pruebas y 3 iteraciones, en total se identificaron 10 no conformidades. La tabla muestra un resumen de las deficiencias encontradas luego de realizar el caso de prueba Gestionar Suscripción, en el cual se verificó que el sistema permite adicionar, eliminar y editar una suscripción, validando que solo se acepten los caracteres válidos para cada campo.

Fecha	Iteración	Caso de Prueba	Tipo de error	Descripción
17/05/2012	1	CU Gestionar Suscripción	Validación	El campo dirección de correo acepta caracteres no válidos
17/05/2012	1	CU Gestionar Suscripción	Validación	El campo ip acepta letras
17/05/2012	1	CU Gestionar Suscripción	Recomendación	Cambiar el nombre de la ventana adicionar suscripción
24/05/2012	2	CU Gestionar Suscripción	Recomendación	Cambiar la interfaz asociada a la frecuencia
24/05/2012	2	CU Gestionar Suscripción	Ortografía	Le falta la tilde al campo día de la semana
31/05/2012	3	CU Gestionar Suscripción	-	-

Tabla 3: No conformidades asociadas al caso de uso Gestionar Suscripción

La tabla muestra la cantidad de no conformidades que se identificaron en cada iteración asociadas a cada caso de uso. En la primera iteración se identificaron 7, en la segunda iteración 3 y en la tercera iteración no se encontraron no conformidades.

Casos de Uso	Iteración 1	Iteración 2	Iteración 3
Gestionar Categoría	1	-	-
Gestionar Suscripción	3	2	-
Administrar Reporte	1	-	-
Administrar Plantilla	2	1	-

Tabla 4: Cantidad de no conformidades por casos de uso en cada una de las iteraciones

Se realizaron pruebas unitarias a las 20 clases que contiene el módulo, para verificar que los datos de entrada de cada uno de los métodos que se probaron, devuelven la salida esperada. Para realizar este tipo de pruebas se usó el *framework* Lime que incluye Symfony, el cual está escrito en PHP, además no depende de ningún archivo para ejecutarse y los resultados que devuelve son muy fáciles de leer. La imagen muestra la respuesta obtenida al ejecutar la prueba al método createSuscription de la clase addSuscription. En la que se muestra que los datos de entrada: identificador del reporte, identificador del formato, nombre de la suscripción, tipo de suscripción, asunto y texto del correo, usuario y contraseña de la computadora son datos válidos que devuelven una nueva suscripción, la cual se almacena en la base de datos.

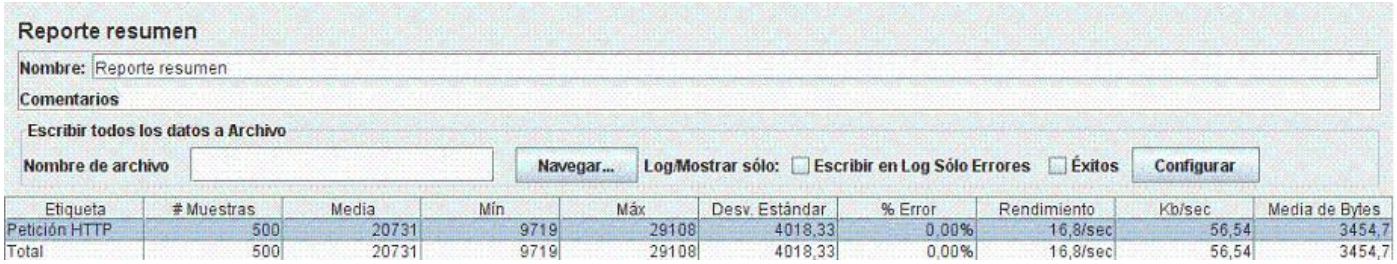
Resultado de una prueba unitaria realizada a la clase addSuscription

```
1..8
# Comienza la prueba para la clase addSuscription
ok 1 - Comparando id del reporte
ok 2 - Comprarando id del formato
ok 3 - Comparando nombre de la suscripción
ok 4 - Comparando tipo de suscripción
ok 5 - Comparando asunto del correo
ok 6 - Comparando texto del correo
ok 7 - Comparando usuario de la PC
ok 8 - Comparando contraseña de la PC
Looks like everything went fine.
```

Figura 16: Imagen del resultado de las pruebas unitarias

Para validar el requisito no funcional 7 se realizaron las pruebas de carga y estrés, las cuales son pruebas de rendimiento que se realizan para comprobar la velocidad con la que un sistema ejecuta una tarea. Para ejecutar este tipo de prueba se usó la herramienta JMeter, la cual es una aplicación Java de escritorio,

multiplataforma, extensible y de código abierto. La imagen muestra el resultado obtenido al ejecutar la prueba, se observa que se realizó sin errores. Esto se deduce de la columna del tanto por ciento de errores para cada una de las peticiones. El rendimiento muestra que para una simulación de 500 usuarios es capaz de aceptar una media de 16,8 peticiones por segundo. La latencia (entendida como el tiempo en obtener respuesta del servidor) para cada conjunto de pruebas no supera el valor de 25441 milisegundos.



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Petición HTTP	500	20731	9719	29108	4018,33	0,00%	16,8/sec	56,54	3454,7
Total	500	20731	9719	29108	4018,33	0,00%	16,8/sec	56,54	3454,7

Figura 17: Imagen del resultado de la prueba de carga y estrés

Conclusiones del Capítulo

En el presente capítulo se realizó la descripción de la implementación del módulo Administrador de Reportes del GDR. Se representó el diagrama de componentes asociado al caso de uso Gestionar Suscripción, con el cual se trabajó desde el inicio del trabajo. Se mostraron imágenes con el código de alguna de las funcionalidades de la aplicación, así como los mecanismos de implementación utilizados. Se realizaron pruebas unitarias y funcionales para validar que los requisitos fueron implementados correctamente, se utilizó el método de la caja negra a través de los casos de prueba basados en los casos de usos.

CONCLUSIONES

Una vez culminado el trabajo es posible afirmar que se cumplieron los objetivos trazados para el mismo:

- Se realizó el análisis y diseño del módulo Administrador de Reportes para administrar los reportes, plantillas y suscripciones.
- Se realizó la implementación del módulo Administrador de Reportes, él cual cumple con los requisitos identificados.
- Se realizaron pruebas funcionales y unitarias para validar el correcto funcionamiento del módulo Administrador de Reportes.

RECOMENDACIONES

Luego del cumplimiento de los objetivos trazados, se recomienda:

- Agrupar las plantillas y suscripciones por categorías, de la misma manera que se encuentran agrupados los reportes, para lograr mejor organización.

REFERENCIAS BIBLIOGRÁFICAS

1. Gestión de la información, del conocimiento y de la calidad. [En línea] 10 de Mayo de 2005. [Citado el: 20 de Septiembre de 2011.] http://scielo.sld.cu/scielo.php?pid=S1024-94352002000500004&script=sci_arttext.
2. Silva Hernández, Iliana Amabely. Generador Automático de Reportes Dinámicos. [En línea] [Citado el: 20 de Septiembre de 2011.] http://www.cs.cinvestav.mx/TesisGraduados/2003/resumenSilvaHernandez.html.
3. *Caracterización del Centro de Tecnologías de Gestión de Datos.*
4. Hernández Carvajal, Juan José. *Desarrollo del módulo Administrador de Reportes del Generador Dinámico de Reportes.* 2011.
5. Cívica. [En línea] [Citado el: 5 de Octubre de 2011.] http://www.civica-soft.com/tecnologia/jasperreports.html.
6. Sitio EcuRed. [En línea] [Citado el: 5 de Octubre de 2011.] http://www.ecured.cu/index.php/Crystal_Reports.
7. ComponenteSource. [En línea] [Citado el: 2 de Octubre de 2011.] http://www.componentsource.com/products/activereports-6/summary-es.html.
8. SQL Max Connections. [En línea] 2001. [Citado el: 2 de Octubre de 2011.] http://www.sqlmax.com/reportin_services1.asp.
9. Allende, Jesús S. Programación en Java. [En línea] 2009. [Citado el: 25 de Septiembre de 2011.] http://www.lab.dit.upm.es/~fprg/asignatura/java.htm.
10. Walsh, Norman. A Technical Introduction to XML. [En línea] 1998. [Citado el: 5 de Noviembre de 2011.] http://www.xml.com.
11. Eguíluz Pérez, Javier. Introducción a JavaScript. [En línea] 2008. [Citado el: 26 de Octubre de 2011.] http://www.librosweb.es.
12. Rosalba. Arquitectura del computador. [En línea] 11 de Julio de 2010. [Citado el: 12 de Octubre de 2011.] http://rosalba-rouse1.blogspot.com/.
13. Eguíluz Pérez, Javier. Introducción a AJAX. [En línea] 2009. [Citado el: 8 de Noviembre de 2011.] http://www.librosweb.es.
14. ecuRed. [En línea] [Citado el: 21 de Mayo de 2012.] http://www.ecured.cu/index.php/Symfony.
15. Rovelo, Efren A. symfony. [En línea] [Citado el: 15 de Octubre de 2011.] http://www.symfony-project.org/.
16. EcuRed. [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.ecured.cu/index.php/Sencha_Ext_JS.
17. NetBeans. [En línea] [Citado el: 10 de Octubre de 2011.] http://netbeans.org/community/releases/70/index.html.
18. JSON. [En línea] [Citado el: 17 de Octubre de 2011.] http://www.json.org/json-es.html.
19. PostgreSQL. [En línea] [Citado el: 8 de Noviembre de 2011.] http://www.postgreSQL.org.
20. pgAdmin PostgreSQL Tools. [En línea] [Citado el: 5 de Noviembre de 2011.] http://www.pgadmin.org/features.php.
21. Guardado, Iván. web.Ontuts. [En línea] 5 de Mayo de 2010. [Citado el: 19 de Noviembre de 2011.] http://web.ontuts.com/.
22. Propel. [En línea] [Citado el: 20 de Octubre de 2011.] http://www.propelorm.org/.

23. Popkin Software and Systems. Modelado de Sistemas com UML. [En línea] [Citado el: 4 de Marzo de 2012.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.
24. Yanes León, Vania Elena. *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* 2011.
25. EcuRed. [En línea] [Citado el: 12 de Noviembre de 2011.] http://www.ecured.cu/index.php/Visual_Paradigm.
26. Jordana, Garcilaso. Introducción Open UP. [En línea] 2008. [Citado el: 10 de Noviembre de 2011.]
27. Mühlrad, Daniel. Patrones de diseño. [En línea] 2008. [Citado el: 11 de Octubre de 2011.]
28. Hernández, Pedro Veloso. Uso de patrones de arquitectura. [En línea] [Citado el: 11 de Noviembre de 2011.]
29. Ivanek. [En línea] [Citado el: 10 de Noviembre de 2011.] http://ivanex.wikidot.com/patron-arquitectura.
30. EcuRed. [En línea] [Citado el: 2 de Marzo de 2012.] http://www.ecured.cu/index.php/Modelo_de_dominio.
31. EcuRed. [En línea] [Citado el: 2 de Marzo de 2012.] http://www.ecured.cu/index.php/Requisitos_de_Software.
32. Cáceres Tello, Jesús. [En línea] [Citado el: 4 de Marzo de 2012.] http://www2.uah.es/jcaceres/capsulas/DiagramaCasosDeUso.pdf.
33. Scribd. [En línea] [Citado el: 5 de Marzo de 2012.] http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.
34. Elizabeth Arrieche. Scribd. [En línea] Enero de 2011. [Citado el: 5 de Marzo de 2012.] http://es.scribd.com/doc/50741997/Pruebas-de-Software.
35. PRESSMAN, ROGER S. *Ingenieria de Software Un Enfoque Práctico.*
36. Luis Artola . Tipos de pruebas automatizadas de software . [En línea] 2009. [Citado el: 7 de Marzo de 2012.] http://www.programania.net/diseno-de-software/tipos-de-pruebas-automatizadas-de-software/.
37. BuenasTareas.com. [En línea] 2012. [Citado el: 22 de Marzo de 2012.] http://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html.

BIBLIOGRAFÍA

Allende, Jesús S. Programación en Java. http://www.lab.dit.upm.es/~fprg/asignatura/java.htm.
Alexander Oré B. CalidadySoftware.com.
http://www.calidadysoftware.com/testing/pruebas_funcionales.php.
BuenasTareas.com. http://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html.
Cáceres Tello, Jesús. http://www2.uah.es/jcaceres/capsulas/DiagramaCasosDeUso.pdf.
Cívica. http://www.civica-soft.com/tecnologia/jasperreports.html.
ComponenteSource. http://www.componentsource.com/products/activereports-6/summary-es.html.
Daniel Mühlrad, Patrones de diseño.
EcuRed. http://www.ecured.cu/index.php/Modelo_de_dominio.
EcuRed. http://www.ecured.cu/index.php/Requisitos_de_Software.
EcuRed. http://www.ecured.cu/index.php/Sencha_Ext_JS.
EcuRed. http://www.ecured.cu/index.php/Visual_Paradigm.
EcuRed. http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.
Eguíluz Pérez, Javier. Introducción a AJAX. http://www.librosweb.es.
Eguíluz Pérez, Javier. Introducción a JavaScript. http://www.librosweb.es.
Elizabeth Arrieche. Scribd. http://es.scribd.com/doc/50741997/Pruebas-de-Software.
Enerys Mesa Morales, Yoandry Martínez Rodríguez, Irina Elena Argota Vega, Sistema para el análisis cuantitativo de los riesgos para los proyectos de producción de software.
http://semanatecnologica.fordes.co.cu/ocs-2.3.2/public/site/150.pdf
http://www.xml.com.
http://www.symfony-project.org/
Choucair Cárdenas, Maria Clara. Pruebas de software ¿la salvación, un proceso sin utilidad, trivial, simplemente una moda?
http://www.librosweb.es/
Hernández Carvajal, Juan José. Desarrollo del módulo Administrador de Reportes del Generador Dinámico de Reportes. 2011.
François Zaninotto y Fabien Potencier, Symfony 1.2, la guía definitiva.
Félix Prieto, Patrones de diseño.
García, Joaquín, Patrones de diseño.
Gestión de la información, del conocimiento y de la calidad. http://scielo.sld.cu/scielo.php?pid=S1024-94352002000500004&script=sci_arttext.
Guardado, Iván. web.Ontuts. http://web.ontuts.com/.
Hernández, Pedro Veloso. Uso de patrones de arquitectura.
Ivanek. http://ivanex.wikidot.com/patron-arquitectura.
Iván Guardado, Introducción a ORM, 5/Mayo/2010.
Jordana, Garcilaso. Introducción Open UP.
JSON. http://www.json.org/json-es.html.
Juan Murua Olaide, Metodologías para desarrollo de software.
Luis Artola . Tipos de pruebas automatizadas de software http://www.programania.net/diseno-de-software/tipos-de-pruebas-automatizadas-de-software/.
Mühlrad, Daniel. Patrones de diseño.
NetBeans. http://netbeans.org/community/releases/70/index.html.

PBWorks. http://isg2.pbworks.com/w/page/7624280/Pruebas%20del%20Software.
pgAdmin PostgreSQL Tools. http://www.pgadmin.org/features.php.
PostgreSQL. http://www.postgreSQL.org.
Popkin Software and Systems. Modelado de Sistemas com UML. http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.
Rosalba. Arquitectura del computador. http://rosalba-rouse1.blogspot.com/.
Rovelo, Efren A. symfony. http://www.symfony-project.org/.
PRESSMAN, ROGER S.*Ingenieria de Software Un Enfoque Práctico.*
Propel. http://www.propelorm.org/.
Rumbaugh, Jim, Jacobson, Ivar y Booch, Grady.UML el lenguaje unificado de modelado. http://www.uml.org/.
Scribd. http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.
Silva Hernández, Iliana Amabely. Generador Automático de Reportes Dinámicos. http://www.cs.cinvestav.mx/TesisGraduados/2003/resumenSilvaHernandez.html.
Sitio EcuRed. http://www.ecured.cu/index.php/Crystal_Reports.
SQL Max Connections. http://www.sqlmax.com/reportin_services1.asp.
Shaw, Mary y Garlan, David.*Software Architecture.* 1996.
UML el lenguaje unificado de modelado. Grady Booch, Jim Rumbaugh, Ivar Jacobson http://www.uml.org/
Web.Ontuts, http://web.ontuts.com/tutoriales/introduccion-a-object-relational-mapping-orm/
Walsh, Norman. A Technical Introduction to XML.
Yanes León, Vania Elena. Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0. 2011
Prieto, Félix. Patrones de diseño, Curso 2008/09
López Quesada, Juan Antonio. Pruebas del software
Fabien Potencier, François Zaninotto. La guía definitiva de Symfony
Fabien Potencier. Tutorial de Jobbet
Shea Frederick, Colin Ramsay, Steve Cutter Blades. Learning ExtJS
Stoyan Stefanov. Object Oriented JavaScript
Jorge Ramon. ExtJS CookBook
Jordi Lonch. Curso framework Symfony
James Hart, Dave Writz, Eric Jung, Steven Brodhead. Java y XML
Walter Celiano, Tituaña Cumbal, Edwin Jesús Torres Cañizares. Elaboración de un manual de plataforma NetBeans IDE
Esteban Saavedra López. Revista de Software Libre
Teodor Danciu, Lucian Chirita. The Definitive guide to Jasper Reports
http://www.calidadysoftware.com/testing/pruebas_funcionales.php.
EcuRed. http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.
Rodríguez, Jorge. *Pruebas Unitarias.*
Hernández Hernández, Yasmany. Pautas de arquitectura para el desarrollo de la versión 2 del GDR.
Yanes León, Vania Elena. Definición de la arquitectura de software para el Generador Dinamico de Reportes en su versión 2.0

ANEXOS

Diagrama de Clases del Diseño del Caso de Uso Gestionar Categoría

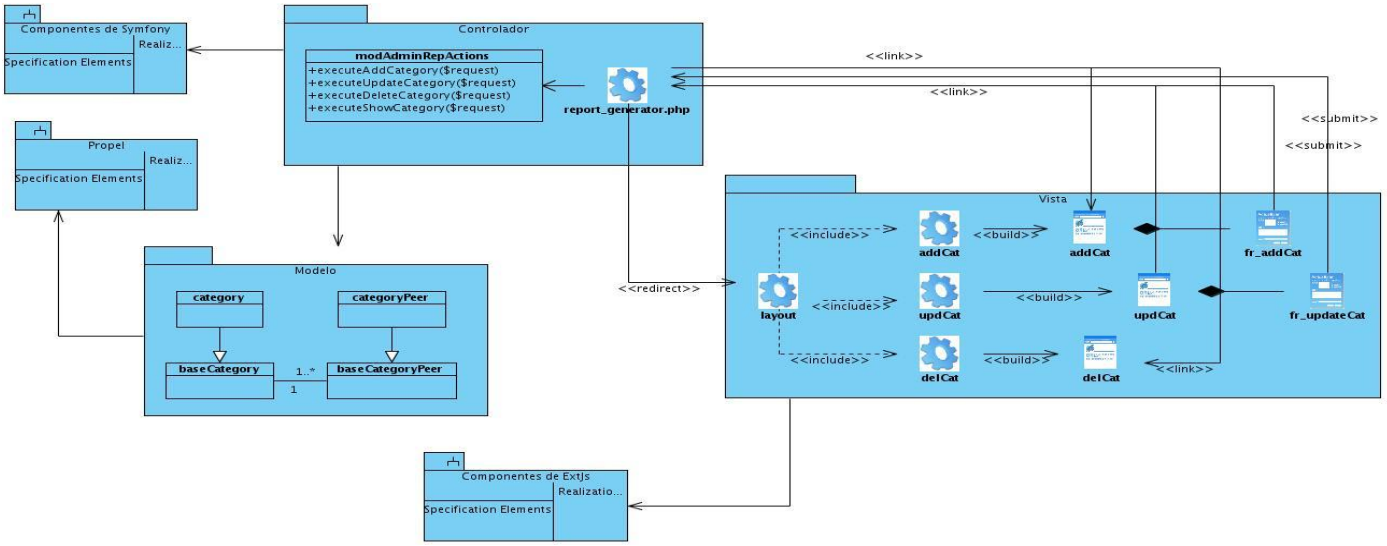


Diagrama de Clases del Diseño del Caso de Uso Administrar Plantilla

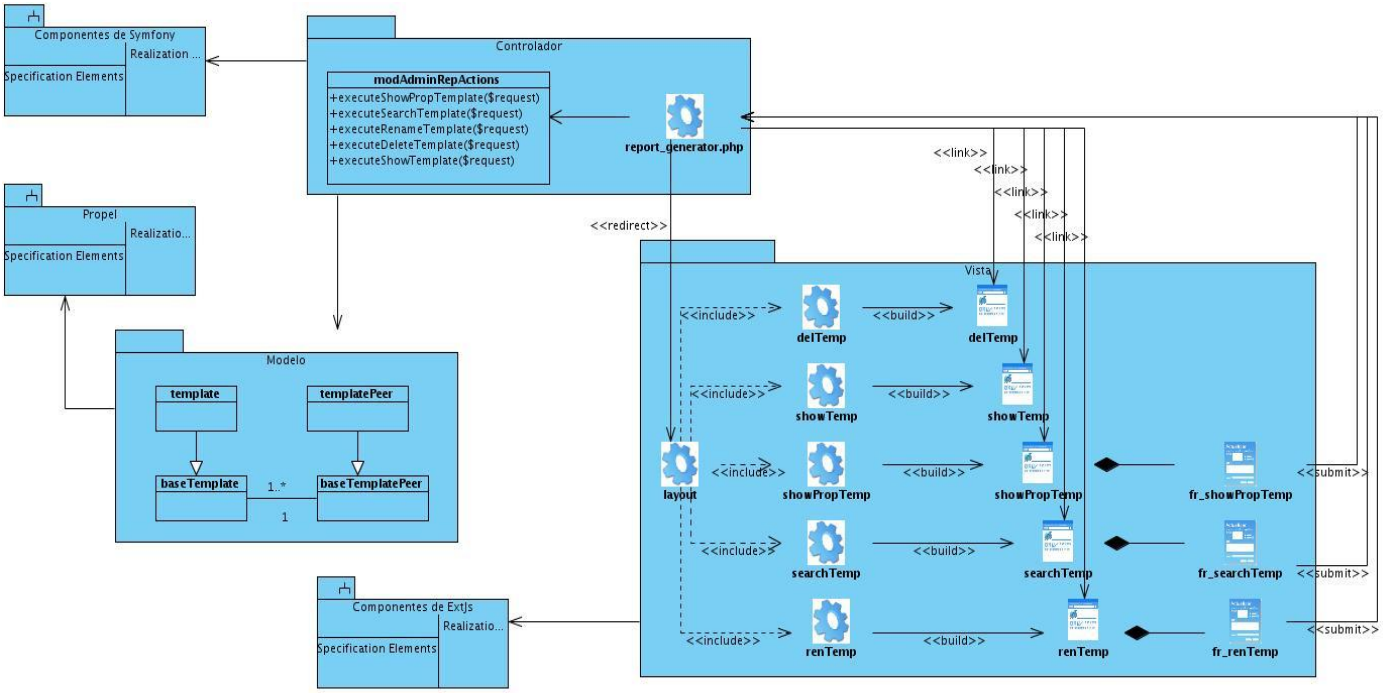


Diagrama de Clases del Diseño del Caso de Uso Administrar Reporte

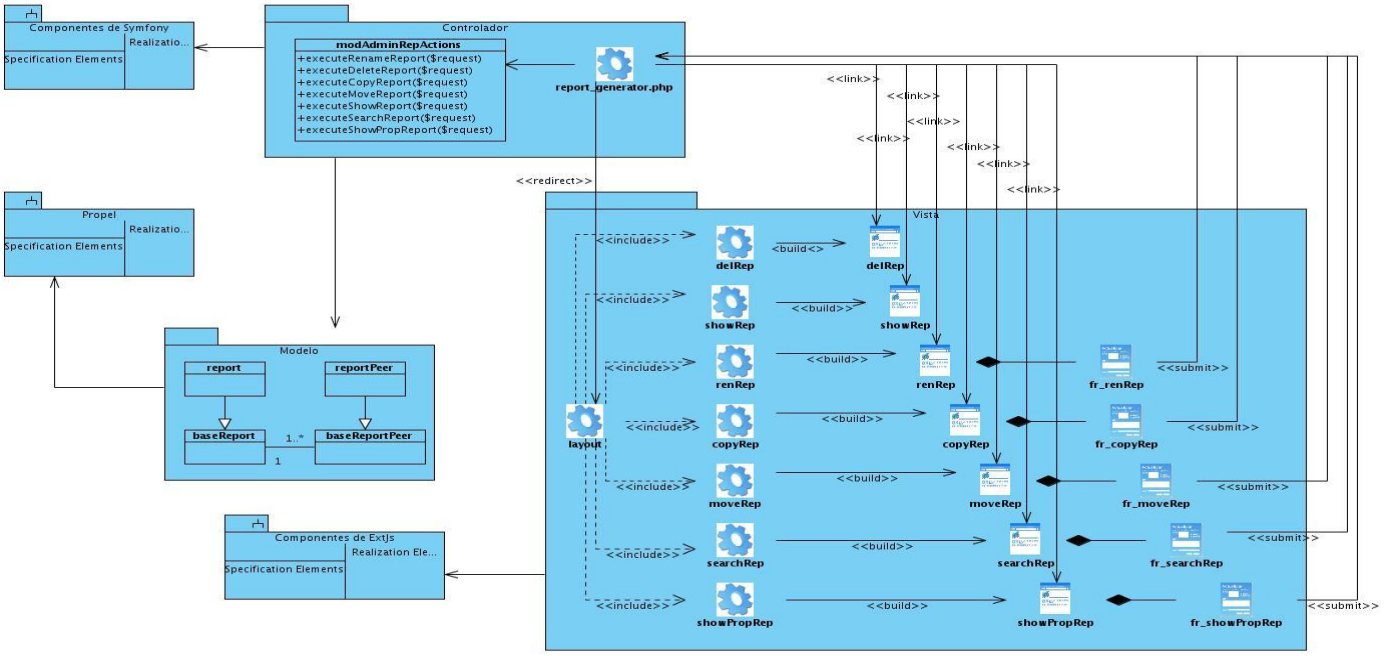


Diagrama de Componentes del Caso de Uso Gestionar Categoría

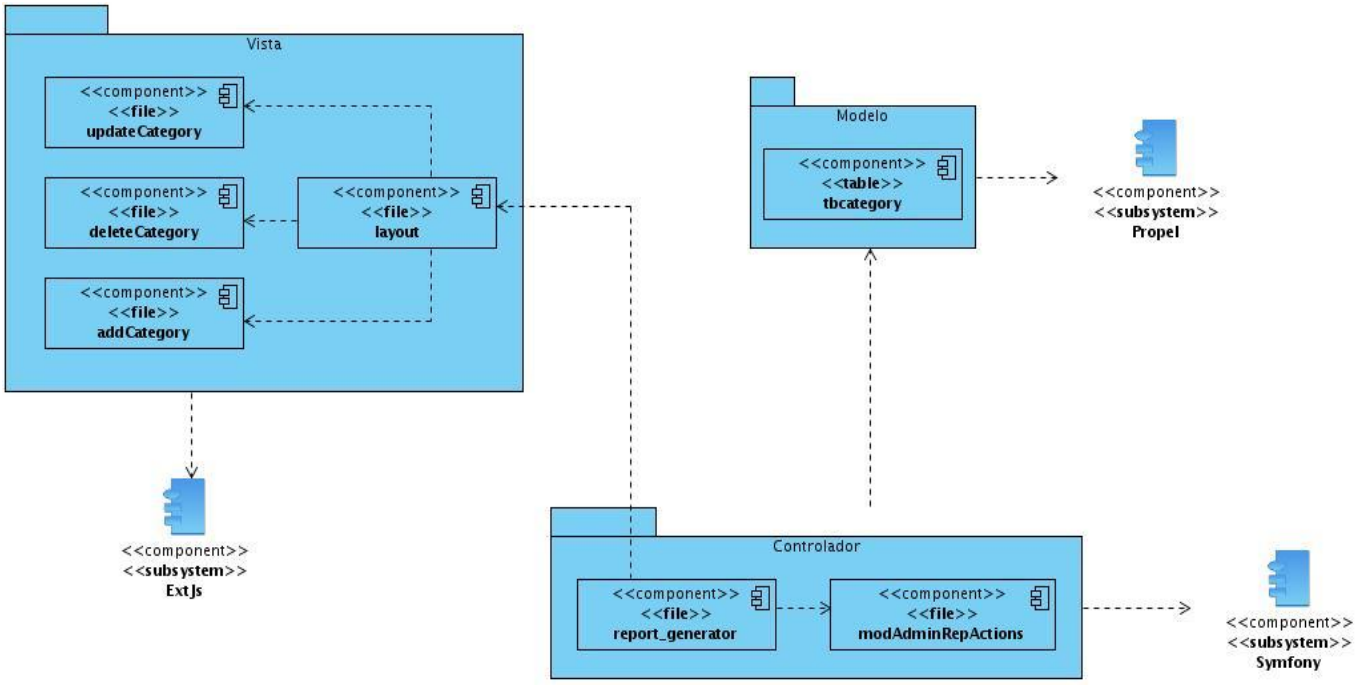


Diagrama de Componentes del Caso de Uso Administrar Reporte

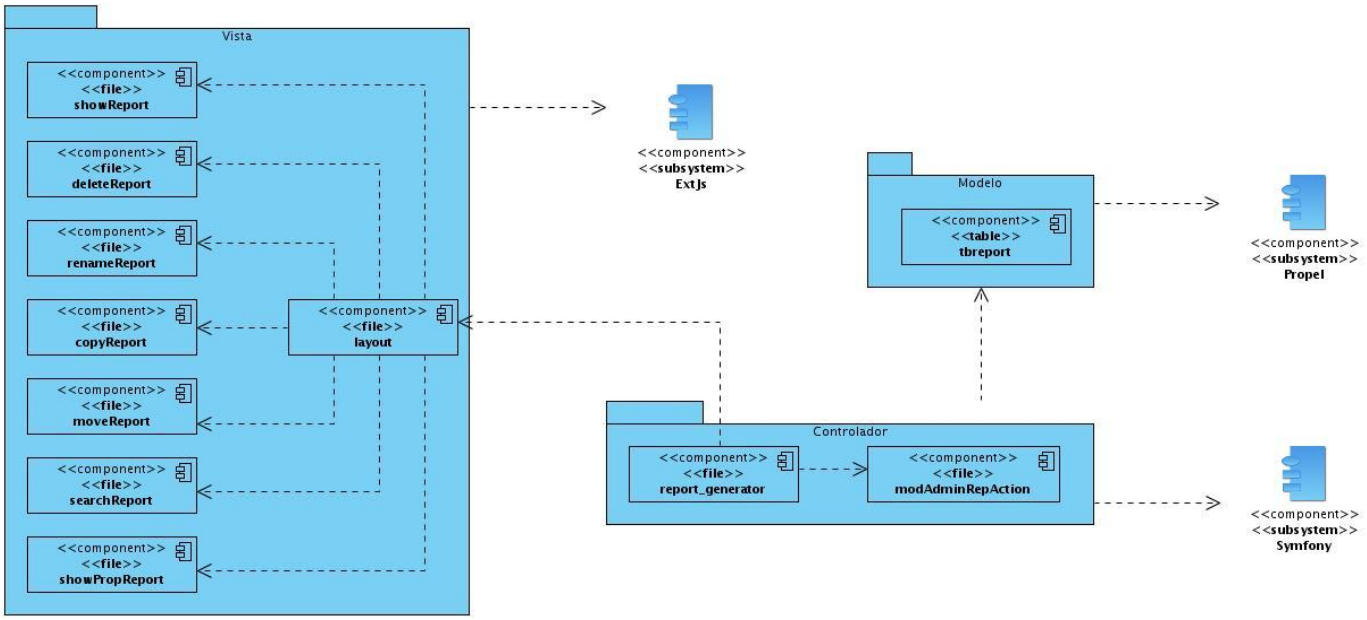
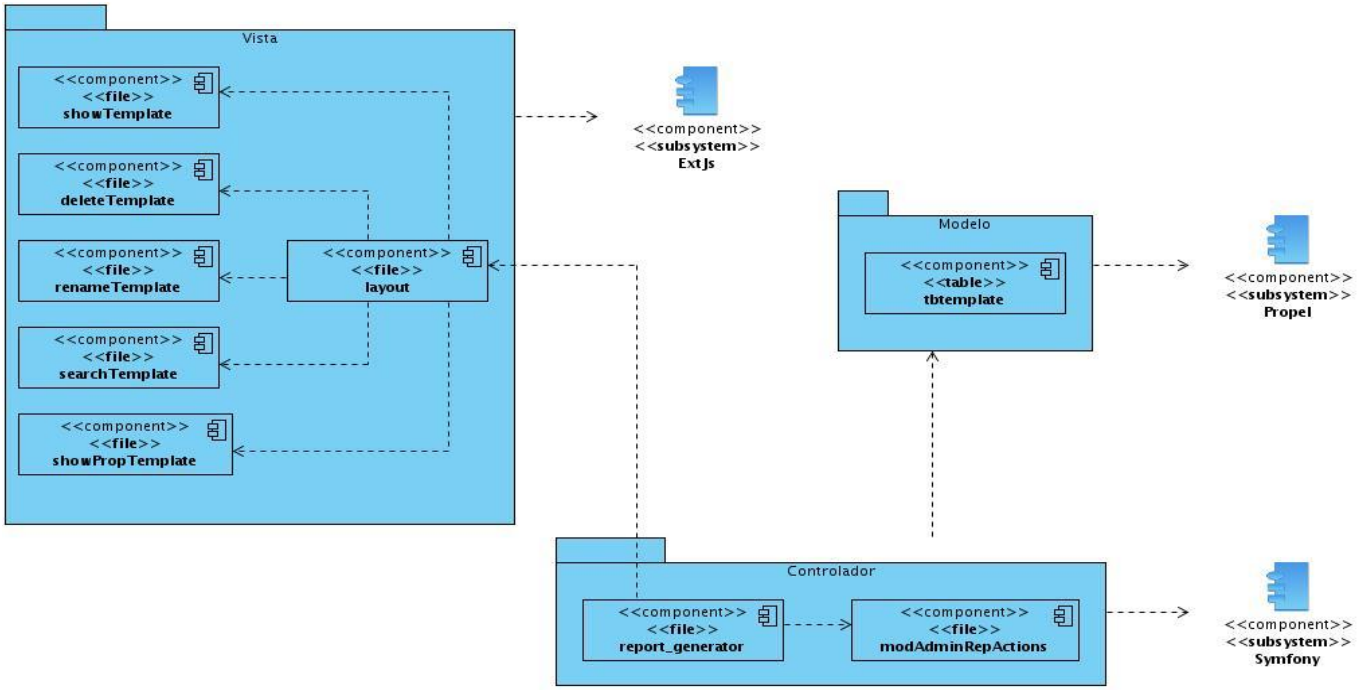


Diagrama de Componentes del Caso de Uso Administrar Plantilla



GLOSARIO

AJAX: JavaScript Asíncrono + XML

CASE: Ingeniería de Software Asistida por Computación

CU: Caso de Uso

DATEC: Centro de Tecnologías de Gestión de Datos

EULA: Acuerdo de Licencia de Usuario Final

Framework: Marco de Trabajo

GDR: Generador Dinámico de Reportes

GOF: Banda de los Cuatro

GPL: Licencias Open Source

GRASP: Patrones de Software para la Asignación General de Responsabilidad

IDE: Entorno Desarrollo Integrado

JSON: Notación de Objetos de JavaScript

MVC: Modelo – Vista – Controlador

MVCC: Multi-Versión Control de Concurrencia

OpenUp: Metodología para el proceso del desarrollo del software

ORM: Mapeo de Objetos a Bases de Datos

PATDSI: Plataforma de Ayuda a la Toma de Decisiones y Soluciones Integrales

POO: Programación Orientada a Objetos

SOA: Arquitecturas Orientada a Servicios

SO: Sistema Operativo

UCI: Universidad de las Ciencias Informáticas

UI: Interfaces de Usuario

UML: Lenguaje Unificado de Modelado

XML: Lenguaje de Marcas Extensible