

**Universidad de las Ciencias Informáticas**  
**Facultad 6**



**“Diseñador gráfico de reportes para el Reporteador  
Dinámico de la plataforma alasGRATO”.**

Trabajo de Diploma para optar por el título de  
Ingeniero Informático

**Autores:** Mayelín Lluch Hernández

Edilberto Bravo Campo

**Tutores:** MsC. Aurelio Antelo Collado

Ing. José Rolando Lafaurié Olivares

Ing. Julio Antonio Villaverde Martínez

Junio, 2010

*“La grandeza de un hombre no se mide por el terreno que ocupan sus pies, sino por el horizonte que descubren sus ojos”.*

*José Martí*

## **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 \_\_\_\_\_.

**Mayelín Lluch Hernández**

\_\_\_\_\_  
Firma del Autor

**Edilberto Bravo Campo**

\_\_\_\_\_  
Firma del Autor

**MsC. Aurelio Antelo Collado**

\_\_\_\_\_  
Firma del Tutor

**Ing. José Rolando Lafaurié Olivares**

\_\_\_\_\_  
Firma del Tutor

**Ing. Julio Antonio Villaverde Martínez**

\_\_\_\_\_  
Firma del Tutor

## DATOS DE CONTACTO

### Tutores:

**Aurelio Antelo Collado:** Ingeniero Industrial, Máster en Matemática Aplicada, Profesor Asistente con 4 años de experiencia. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

E-mail: [aaantelo@uci.cu](mailto:aaantelo@uci.cu)

**José Rolando Lafaurié Olivares:** Ingeniero en Ciencias Informáticas. Profesor en Adiestramiento. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

E-mail: [jrlafaurie@uci.cu](mailto:jrlafaurie@uci.cu)

**Julio Antonio Villaverde Martínez:** Ingeniero en Ciencias Informáticas.

E-mail: [jvillaverde@uci.cu](mailto:jvillaverde@uci.cu)

## AGRADECIMIENTOS

*Mayelín:*

*A mis padres por ser los mejores y por guiarme ante la vida con sabiduría y amor.*

*A mi novio por quererme tanto y darme los días más felices...*

*A mi compañero de tesis por su paciencia.*

*A mis tutores por su apoyo.*

*A mi familia por su cariño.*

*A mis amigos y compañeros de aula por el tiempo compartido.*

*A todos los que hicieron posible este gran sueño. Gracias.*

*Edilberto:*

*En especial quisiera agradecer a mi madre.*

*A toda mi familia por darme su apoyo incondicional.*

## DEDICATORIA

*A nuestros padres, hermanos, familiares y amigos.*

## **RESUMEN**

En la Universidad de las Ciencias Informáticas (UCI) se desarrolló un Reporteador para el proyecto productivo alasGRATO de la facultad 6, se trata de un sistema de generación de reportes referente a información sobre la estructura y actividad biológica de diferentes compuestos. El mismo presenta una serie de características importantes para construir y guardar reportes, además puede generar informes con información extraída de diversos gestores de bases de datos.

Sin embargo, el sistema creado no cuenta con la posibilidad de que el usuario pueda decidir la apariencia de su informe, y se obtienen actualmente reportes basados en plantillas definidas previamente por el desarrollador de la aplicación. En aras de dar solución a este problema se realiza este trabajo de diploma cuyo principal objetivo es implementar un diseñador de reportes como parte del Reporteador Dinámico, que permita la configuración de las plantillas de los reportes de una forma rápida y sencilla. Esto contribuirá significativamente a que los clientes se sientan satisfechos con la presentación de sus informes.

En el presente documento se describe la fundamentación teórica, las características del sistema, y en capítulos posteriores se define el diseño e implementación.

## **PALABRAS CLAVE**

Reportes, generador de reportes, diseño, plantilla.

## **ABSTRACT**

At the University of Informatics Sciences (UCI) has developed a report for the alasGRATO productive project of faculty 6, is a report generation system of the information concerning about the structure and biological activity of different compounds. It presents a number of important features to build and save reports, can also generate reports with information from various database managers. However, the system has not created the possibility that the user can decide the appearance of your report, and are currently produced reports based on predefined templates for the application developer. In the interest of resolving this problem is made this dissertation whose main objective is to implement a report designer as part of dynamic reports that allow the configuration of the templates of reports quickly and easily. This will contribute significantly to the customers feel satisfied with the presentation of their reports.

This paper describes the theoretical basis, the characteristics of the system, and in later chapters defines the design and implementation.

## **KEYWORDS**

Reports, Reports generator, design, templates.



## TABLA DE CONTENIDOS

<b>AGRADECIMIENTOS</b> .....	I
<b>DEDICATORIA</b> .....	II
<b>RESUMEN</b> .....	III
<b>ABSTRACT</b> .....	IV
<b>INTRODUCCIÓN</b> .....	1
<b>CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA</b> .....	4
<b>1.1 Introducción</b> .....	4
<b>1.2 Generalidades sobre los generadores de reportes</b> .....	4
<b>1.3 Diseño de reportes</b> .....	5
<b>1.4 Diseñadores de reportes</b> .....	6
1.4.1 NCRReport .....	6
1.4.2 Report Manager .....	7
1.4.3 Kugar .....	8
1.4.4 Matrix .....	9
1.4.5 Pentaho Reporting .....	10
1.4.6 iReports .....	11
<b>1.5 Comparativa entre los diseñadores de reportes estudiados</b> .....	13
<b>1.6 Herramientas y metodologías utilizadas</b> .....	13
1.6.1 Metodología: OpenUP .....	14
1.6.2 Lenguaje de modelado: UML .....	14
1.6.3 Herramienta CASE: Visual Paradigm .....	14
1.6.4 Lenguaje de programación: Java .....	15
1.6.5 Entorno Integrado de Desarrollo: Eclipse .....	15
1.6.6 Librería JFreeChart .....	16
1.6.7 Librería JasperReport .....	16
<b>1.7 Conclusiones</b> .....	16
<b>CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA</b> .....	17
<b>2.1 Introducción</b> .....	17
<b>2.2 Breve descripción del sistema</b> .....	17
<b>2.3 Modelo de Dominio</b> .....	17
<b>2.4 Especificación de Requerimientos de Software</b> .....	18
2.4.1 Requerimientos Funcionales (RF) .....	19
2.4.2 Requerimientos No Funcionales .....	19
<b>2.5 Actores y Casos de Uso del Sistema</b> .....	20

2.5.1 Actores del Sistema .....	20
2.5.2 Casos de Uso del Sistema .....	20
2.5.3 Diagrama de Casos de Uso del Sistema .....	21
<b>2.6 Patrones de CU utilizados .....</b>	<b>22</b>
2.6.1 Concordancia (Commonality). Reusabilidad .....	22
2.6.2 CRUD (Creating, Reading, Updating, Deleting). Completo.....	22
<b>2.7 Descripción textual de los Casos de Uso del Sistema .....</b>	<b>23</b>
2.7.1 Caso de Uso: Crear plantilla.....	23
2.7.2 Caso de Uso: Modificar plantilla. ....	24
2.7.3 Caso de Uso: Gestionar elementos de la plantilla.....	27
<b>2.8 Conclusiones.....</b>	<b>28</b>
<b>CAPÍTULO 3: DISEÑO DEL SISTEMA .....</b>	<b>29</b>
<b>3.1 Introducción.....</b>	<b>29</b>
<b>3.3 Patrones de Diseño.....</b>	<b>31</b>
3.3.1 Patrón Observador.....	31
3.3.2 Patrón Singleton .....	31
3.3.3 Patrón Experto.....	32
3.3.4 Patrón Creador .....	33
<b>3.4 Modelo de diseño .....</b>	<b>33</b>
3.4.1 Diagrama de subsistemas .....	34
3.4.2 Diagrama de paquetes .....	34
3.4.3 Diagramas de Clases del Diseño .....	35
<b>3.5 Diagramas de Interacción .....</b>	<b>38</b>
3.5.1 Diagramas de Secuencias.....	38
<b>3.6 Diagrama de Despliegue .....</b>	<b>41</b>
<b>3.7 Conclusiones.....</b>	<b>42</b>
<b>CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA.....</b>	<b>43</b>
<b>4.1 Introducción.....</b>	<b>43</b>
<b>4.2 Diagrama de Componentes .....</b>	<b>43</b>
4.2.1 Diagrama de despliegue de los componentes .....	44
<b>4.3 Código fuente de los principales métodos y su descripción.....</b>	<b>44</b>
<b>4.4 Pantallas principales .....</b>	<b>49</b>
<b>4.5 Modelo de pruebas .....</b>	<b>51</b>
4.5.1 Casos de pruebas .....	51
4.5.1.1 Crear plantilla.....	52
4.5.1.2 Modificar plantilla. ....	53
4.5.1.3 Gestionar elementos de la plantilla.....	54
<b>4.6 Conclusiones.....</b>	<b>55</b>
<b>CONCLUSIONES .....</b>	<b>56</b>
<b>RECOMENDACIONES .....</b>	<b>57</b>

<b>REFERENCIAS BIBLIOGRÁFICAS .....</b>	<b>58</b>
<b>GLOSARIO DE TÉRMINOS .....</b>	<b>61</b>

## INTRODUCCIÓN

Un problema recurrente en los sistemas se presenta a la hora de mostrar la información resultante de los procesos de las aplicaciones y/o del día a día, principalmente, cuando esto implica tomar decisiones comerciales o gerenciales. Por lo general, esta información se encuentra almacenada en bases de datos o, en su efecto, en archivos planos. Es posible realizar consultas a las bases de datos, leer estos archivos, diseñar y codificar ventanas o interfaces de usuarios, para interactuar con esa información. Pero este proceso se torna complicado y tedioso, sobre todo si la aplicación tiene varias interfaces o si queremos imprimir la información bajo determinadas condiciones.

A partir de entonces, toma importancia la generación de reportes, tanto para obtener dinamismo en las consultas, como para lograr múltiples vistas e, incluso, para facilitar el mantenimiento posterior y su extensibilidad.

Los generadores de reportes o reporteadores se han creado con el objetivo de poder realizar, de forma transparente, consultas a las bases de datos y obtener la información en un determinado formato. Por lo cual, el acceso a la misma es más fácil y en períodos de tiempo relativamente cortos.

La gestión de reportes representa actualmente una necesidad real para las instituciones, ya que estos software ofrecen al usuario final un acercamiento a la información contenida en las bases de datos, que se requiere en las empresas para facilitar el manejo de dichos documentos. Por otro lado, la importancia de hacer reportes se hace lógica ya que estos son el reflejo del comportamiento interno de cada empresa.

La gran mayoría de estos reporteadores presentan la funcionalidad del diseño de reportes o sea, de configurar la estructura de los informes en forma de plantilla, la misma contendrá la estructura y apariencia antes de incluir los datos de las consultas y así el reporte saldría con la presentación deseada por el cliente. Esto constituye una versátil e intuitiva herramienta para usuarios no técnicos que deseen explotar los datos que su empresa almacena, por medio de reportes. Facilitando así una mejor visión de la información en apoyo a la toma de decisiones, ya que el usuario puede escoger la manera en que quiere visualizar el contenido del reporte.

En este marco de desarrollo la Universidad de Ciencias Informáticas cumple un papel fundamental, en cumplimiento a uno de los objetivos principales por los cuales fue creada: impulsar la producción de software y servicios informáticos en el país. Nuestra facultad está directamente relacionada con el desarrollo de productos para el campo de la Bioinformática, tal es el caso del “Reporteador Dinámico” del proyecto alasGRATO, el cual es un software para la predicción de actividad farmacológica usando la

teoría de grafos. Este cuenta con una serie de características importantes a la hora de construir y guardar reportes.

Sin embargo, actualmente dicho Reporteador no brinda al usuario la posibilidad de diseñar la estructura de su plantilla en el momento de la generación del reporte; debido a que el mismo construye los reportes a partir de una plantilla estática que el usuario no tiene la libertad de editar. Esto impide obtener informes con plantillas personalizadas. Trayendo como resultado que los usuarios no puedan proporcionarle a sus reportes un aspecto conveniente y atractivo de acuerdo a sus necesidades o preferencias.

Lo anteriormente planteado conduce a la formulación del siguiente **problema científico**:

¿Cómo permitir al usuario no especializado la personalización del diseño de los reportes generados por el Reporteador Dinámico de la Plataforma alasGRATO?

Por lo que se define como **objeto de estudio**:

Las herramientas utilizadas para diseñar reportes.

Enmarcado en el **campo de acción**:

Herramientas que diseñan reportes empleando la plataforma JAVA.

Para dar respuesta al problema antes mencionado se plantea, como **objetivo general**:

Desarrollar un módulo para la plataforma alasGRATO capaz de confeccionar el diseño visual de los reportes a generar.

Para dar cumplimiento a este objetivo general se definieron los siguientes **objetivos específicos**:

- Diseñar un módulo que permita confeccionar el diseño de los reportes de forma visual.
- Implementar el módulo diseñado.
- Integrar el módulo al Reporteador Dinámico de la plataforma alasGRATO.
- Realizar pruebas exploratorias.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes **tareas científicas**:

- Análisis del estado del arte acerca de aplicaciones para diseñar reportes.
- Definición de los requisitos funcionales y no funcionales de la aplicación.

- Elaboración del modelo conceptual.
- Desarrollo del diagrama de casos de uso del sistema.
- Desarrollo de los diagramas de clases del diseño.
- Implementación de los diagramas de clases del diseño.
- Implementación de la aplicación.
- Realización de pruebas exploratorias al sistema.

El presente documento está estructurado en cuatro capítulos:

### **Capítulo 1: Fundamentación Teórica.**

En este capítulo se lleva a cabo un estudio y análisis de varias herramientas para el diseño de reportes en el mundo y se selecciona la metodología de desarrollo a utilizar. Además, se presenta el lenguaje de programación y las herramientas empleadas en el desarrollo de la solución.

### **Capítulo 2: Características del Sistema.**

En este capítulo se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el diagrama de caso de usos del sistema, la descripción de los mismos y el modelo de dominio.

### **Capítulo 3: Diseño del Sistema.**

En este capítulo se describe el estilo arquitectónico del sistema, haciendo énfasis en los patrones de diseño utilizados. Además, se desarrolla el diseño del sistema.

### **Capítulo 4: Implementación y pruebas del sistema.**

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los casos de uso. Se incluirá el modelo de despliegue.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En este capítulo se lleva a cabo un estudio y análisis de varias herramientas para el diseño de reportes en el mundo y se selecciona la metodología de desarrollo a utilizar. Además, se presenta el lenguaje de programación y las herramientas empleadas en el desarrollo de la solución.

## 1.2 Generalidades sobre los generadores de reportes

Hace años los reportes se confeccionaban en formato duro, pues no existía una herramienta capaz de realizarlos. En la actualidad, el empleo del formato digital ha venido captando el interés de todos por las facilidades que ofrece y la presencia de aplicaciones que son capaces de generarlos automáticamente. Dichos programas, denominados generadores de reportes o reporteadores, permiten a los usuarios obtener con facilidad datos de archivos o bases de datos.

Los generadores de reportes están compuestos principalmente por dos elementos básicos, un diseñador o editor de informes y un motor de generación. [1]

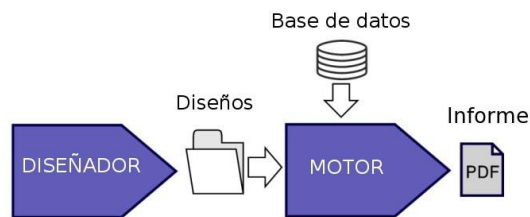


Figura 1.1: Estructura general de un generador de reportes. Esquema de funcionamiento.

El diseñador visual es un programa que ayuda a los usuarios y desarrolladores para diseñar reportes visualmente. A través de una interfaz rica y simple de usar, este provee las funciones más importantes para crear reportes amenos en poco tiempo.

Una vez concluida la etapa de diseño, entra en juego el motor, donde ocurre la generación del reporte completándolo con el contenido directamente de la base de datos y elaborando el informe final con la apariencia previamente diseñada.

El diseñador de reportes es de vital importancia ya que es una herramienta capaz de llevar a cabo la personalización del diseño, brindando a los usuarios una interfaz visual para construir sus reportes, abstrayéndolo de un conocimiento profundo para crearlo.

Se conoce que varios generadores de informes presentan solo el componente de generación o motor generador, y la manera que tienen para diferenciar la apariencia visual de un reporte a otro es modificando las plantillas, a las cuales les definen una estructura y formato por defecto. Esta técnica es muy usada pero es un poco engorrosa puesto que no se ve el diseño de cómo se desea el reporte final.

### 1.3 Diseño de reportes

El diseño de reportes o informes representa una plantilla que va a ser utilizada por el motor para armar el reporte de acuerdo con su estructura, y completándolo con la información obtenida desde la base de datos. De manera que el usuario en tiempo de diseño puede agregar componentes y demás características, en las distintas secciones, que se irán almacenando en la plantilla para en el momento de la generación del reporte el mismo presente el diseño personalizado que se desea y además con los datos dispuestos en su interior.

Las plantillas generalmente contienen las siguientes secciones o bandas:

title	Título del reporte
pageHeader	Encabezado del documento
columnHeader	Encabezado de las columnas
detail	Detalle del documento. Cuerpo
columnFooter	Pie de columna
pageFooter	Pie del documento
summary	Resumen. Cierre del documento

Tabla 1: Secciones del documento.

El significado de cada banda es el siguiente:

- **Title:** Esta banda se mostrará sólo una vez al principio del informe tenga las páginas que tenga el mismo.
- **PageHeader:** Esta banda es la cabecera de la página; se repite cada vez que se pinta una página nueva.



- **ColumnHeader:** Esta banda es la cabecera de las columnas.
- **Detail:** Esta banda es la encargada de mostrar los elementos que tienen alguna repetición.
- **ColumnFooter:** Pie de la columna. Su comportamiento es análogo a **ColumnHeader**.
- **PageFooter:** Pie de página, se repite una vez por página. Su comportamiento es análogo a **PageHeader**.
- **Summary:** Sólo se repite una vez por informe en la última página del mismo. Su comportamiento es análogo a **Title**.

## 1.4 Diseñadores de reportes

A continuación se presenta un estudio de algunas herramientas para el diseño de reportes, con el objetivo de obtener ideas y conceptos importantes para el desarrollo de este trabajo.

### 1.4.1 NCReport

Es un generador de informes que a su vez permite el diseño visual de las plantillas a generar. Tiene una apariencia visual amigable permitiendo, desde el menú principal, la utilización de los distintos componentes en la paleta de herramientas. Ejemplos de los componentes de los cuales dispone están campo de expresión, número de página, texto y etiquetas, imágenes, hipervínculo y autoforma (línea, cuadro) y tabla de referencia cruzada.

Los reportes diseñados son almacenados en archivos con formato XML permitiendo la lectura del mismo por el motor generador para la generación del reporte deseado. Está implementado sobre C++ y Qt y bajo licencia GPL lo cual permite su uso y modificación en caso de interés para cualquier persona o entidad. [2]

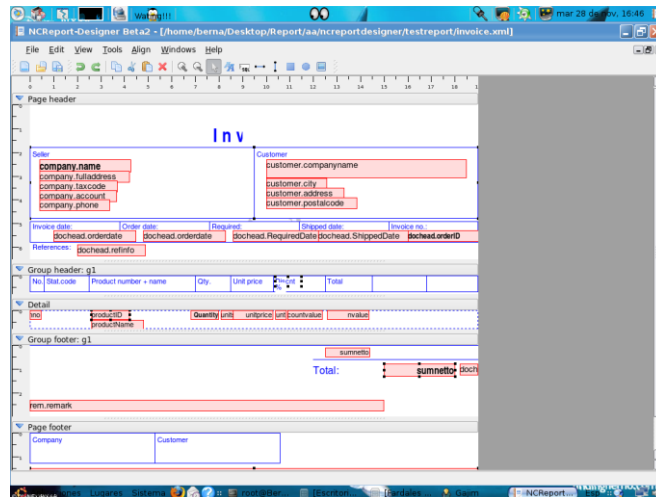


Figura 1.2: Ambiente de edición del NCRReport.

## 1.4.2 Report Manager

Es una aplicación de generación de reportes (Report Manager Designer). Posee una interfaz visual con buena apariencia y permite insertar componentes tales como número de página, campo de expresión, texto y etiquetas, imágenes y gráficos de barras. Incluye una librería dinámica estándar con funciones que proporcionan una interfaz con distintos lenguajes de programación como GNU C. Este sistema es un proyecto de software libre bajo el modelo MPL (incluye permiso de uso en aplicaciones GPL), por lo que puede ser usado en distintas aplicaciones incluyendo las que son de carácter comercial, pero cualquier mejora introducida en el motor de impresión debe ser publicada bajo esta licencia. Corre tanto en GNU/Linux como en Microsoft Windows, y dispone de generación en varios formatos de salida como son PDF, HTML y XSL. [3]

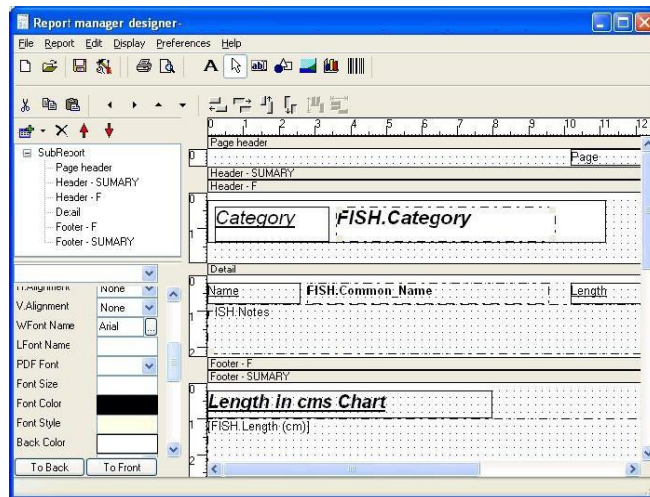


Figura 1.3: Pantalla de la aplicación Report Manager Designer.

### 1.4.3 Kugar

Kugar (KOFFICE.ORG) es una herramienta del entorno de escritorio gráfico KDE desarrollada en Qt™ para generar informes que pueden ser vistos en pantalla o impresos. No es más que uno de los componentes de KOffice, suite ofimática de KDE. Incluye un diseñador de plantillas de informes (Kudesigner), un motor, una pieza (Kpart) de Konqueror para la inspección previa fácil del informe y un grupo de ejemplos. Esto significa que en cualquier aplicación KDE puede incluirse la funcionalidad de presentación de informes, y así, estos pueden ser vistos usando Konqueror (navegador Web y de archivos).

El diseñador de informes de Kugar es una aplicación del tipo WYSIWYG (What You See Is What You Get) así el tamaño de la página del informe define las dimensiones del informe en la pantalla. Mediante el uso de este diseñador se logra la creación y modificación de las plantillas de informes, y la colocación interactiva de las secciones de informe y de los elementos sobre dichas secciones. Este trae consigo toda una gama de plantillas para a partir de ellas empezar a trabajar además de poner a disposición del usuario diferentes menús para facilitar el trabajo. La mayoría de estas opciones están disponibles en barras de herramientas para su acceso de forma visual.

Kugar se discontinúa en la serie de KOffice2 y se reemplazará por Kexi y su generador del informe en el futuro.

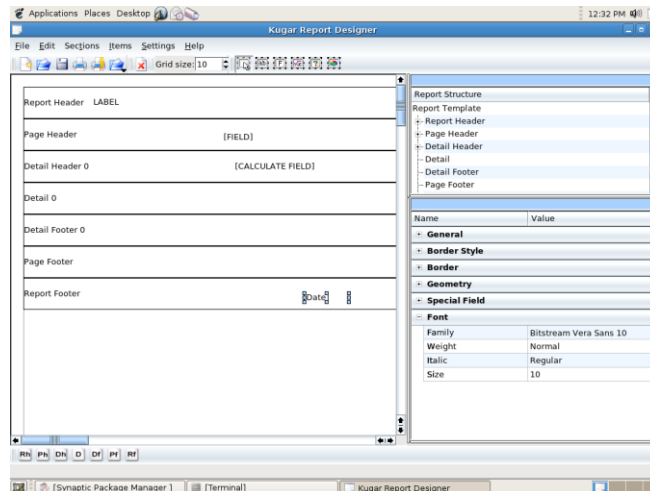


Figura 1.4: Vista diseño de Kugar.

#### 1.4.4 Matrix

Módulo para el diseño de reportes gerenciales del Reporteador Matrix. Permite al usuario diseñar varios tipos de reportes con cientos de opciones habilitadas. Cuenta con múltiples secciones que le permite personalizar el reporte a las exigencias más estrictas. Incluye un generador de fórmulas. En él se pueden realizar reportes de más de 65,000 líneas, con la inclusión de imágenes asociadas a líneas del reporte. Admite la creación de múltiples encabezados/detalles. Permite visualizar los reportes mientras aún no se han generado completamente. [4]

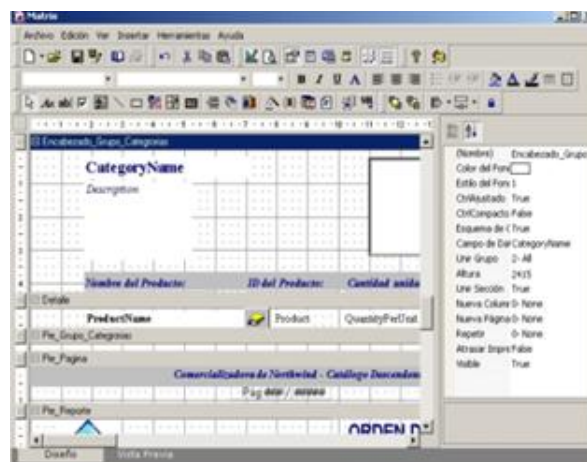


Figura 1.5: Vista del diseñador de reportes del Reporteador Matrix.

### 1.4.5 Pentaho Reporting

Pentaho Reporting es la solución proporcionada por pentaho e integrada en su suite para el desarrollo de informes. Originariamente el proyecto se llamaba JFreeReports hasta la adquisición por parte de pentaho. Existen tres productos diferentes con enfoques distintos.

Por una parte, existe un editor basado en Eclipse con prestaciones profesionales de customización de informes denominado Report designer destinado a desarrolladores de informes. Diseñador gráfico basado en “arrastrar y soltar” (drag & drop) que provee completo control de acceso a los datos, agrupaciones, cálculos, gráficas, formato, etc. para reportes de alta resolución. Posee opciones de salida flexibles incluyendo los populares formatos PDF, HTML, Microsoft Excel, entre otros [5]. Está estructurado de forma que los desarrolladores pueden acceder a sus prestaciones de forma rápida. Además, incluye un editor de consultas para facilitar la confección de los datos que serán utilizados en un informe. Forma parte de la Suite para Inteligencia de Negocios (BI) de Pentaho. [6]

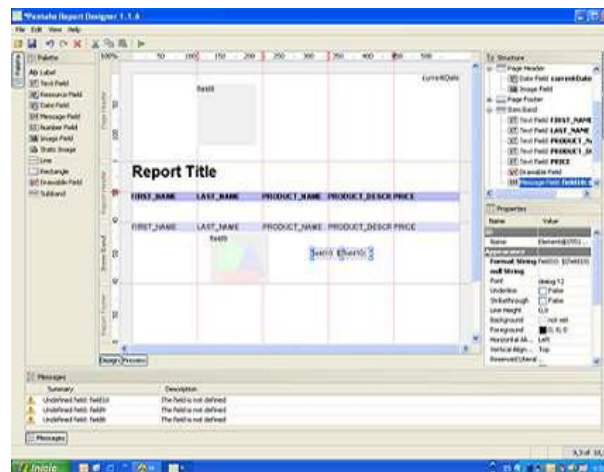


Figura 1.6: Vista del Report designer.

Por otro lado, existen un par de herramientas destinadas a usuarios con menos conocimientos técnicos basadas en wizards. Una como aplicación cliente llamada Report design wizard:

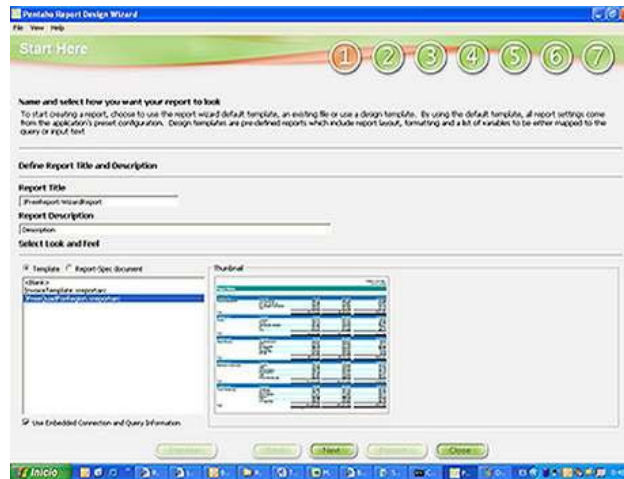


Figura 1.7: Vista del Report design wisard.

Y otra vía web, llamada web ad-hoc reporting:



Figura 1.8: Vista del Web ad-hoc reporting.

## 1.4.6 iReports

La herramienta iReport es un constructor / diseñador de informes visual muy poderoso para JasperReports escrito 100% en Java y además código abierto y gratuito. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. Está integrado con JFreeChart, una de las bibliotecas gráficas de código abierto más difundida para Java. Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de datos, cartas, subreportes. Soporta internacionalización nativamente. Tiene asistentes para las plantillas. [7]

iReport provee a los usuarios de JasperReports una interfaz visual para construir reportes, generar archivos “jasper” y “print” de prueba. Puede utilizarse como una herramienta de oficina para adquirir datos almacenados en una base de datos, sin pasar a través de alguna otra aplicación. [8]

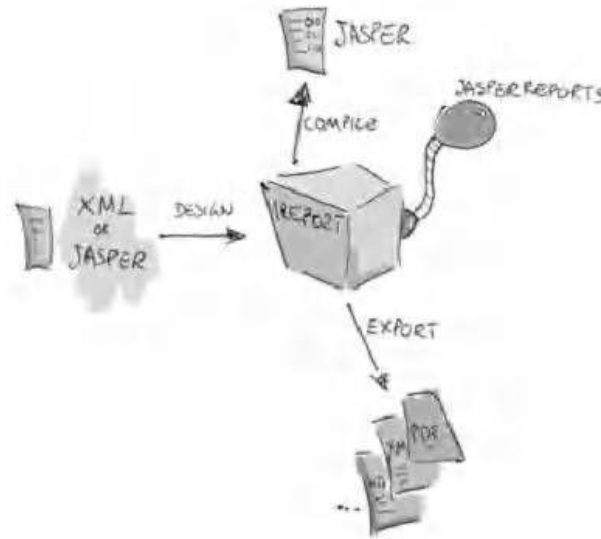


Figura 1.9: Funcionamiento de iReport.

Puede leer y modificar ambos tipos de archivo, XML (que por convención tienen la extensión .jrxml, los cuales contienen el diseño del reporte, ver figura 1.10) y jasper (es el compilado de un código .jrxml, denominado .jasper). A través de JasperReports, es capaz de compilar XML a archivos jasper y “ejecutar reportes” para llenarlos usando varios tipos de fuentes de datos (JRDataSource) y exportar el resultado a PDF, HTML, XLS, CSV, etc.

```

<!DOCTYPE jasperReport PUBLIC "-//JasperReports//DTD JasperReport//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="report name" pageWidth="595" pageHeight="842" columnWidth="535"
leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
<parameter name="count" class="commonj.sdo.DataObject"/>
<background> <band/> </background>
<title> <band height="79"> ... </band> </title>
<pageHeader> <band height="35"> ... </band> </pageHeader>
<columnHeader> <band height="61"> ... </band> </columnHeader>
<detail> <band height="125"> ... </band> </detail>
<columnFooter> <band height="45"> ... </band> </columnFooter>
<pageFooter> <band height="54"> ... </band> </pageFooter>
<summary> <band height="42"> ... </band> </summary>
</jasperReport>

```

Figura 1.10: Muestra del código para un JRXML.





Marrero [9]. Es por ello que en este epígrafe sólo se abordarán las herramientas y la metodología definidas para esa arquitectura.

### **1.6.1 Metodología: OpenUP**

OpenUP (Open Unified Process) [10], es un proceso unificado que se aplica a enfoques iterativo e incremental en un ciclo de vida estructurado. Abarca una filosofía ágil que se centra en la naturaleza de colaboración de desarrollo de software. Se puede ampliar para hacer frente a una amplia variedad de tipos de proyectos. Está organizada en micro-incrementos. Estos representan las unidades cortas de trabajo que producen un ritmo sostenido y mensurable del avance del proyecto, y ofrece un ciclo de retroalimentación muy corto que impulsa las decisiones de adaptación dentro de cada iteración. El ciclo de vida de un proyecto en OpenUP está estructurado en cuatro fases: inicio, elaboración, construcción y transición.

### **1.6.2 Lenguaje de modelado: UML**

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes metodologistas en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson. No es un lenguaje de programación, sino un lenguaje de propósito general para el modelado orientado a objetos. También puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes. Se ha convertido en un estándar, es la herramienta ideal para atacar el ciclo de vida de un proyecto de software utilizando la tecnología Orientada a Objetos.

### **1.6.3 Herramienta CASE: Visual Paradigm**

Visual Paradigm es una herramienta CASE (Computer-Aided Software Engineering) que utiliza "UML": como lenguaje de modelaje. Está diseñada para una amplia gama de usuarios que incluye a: ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, interesados en la creación de grandes sistemas de software de manera confiable, a través del paradigma Orientado a Objetos. Ofrece amplias características de modelado de casos de uso incluyendo la función completa de UML. Produce la documentación del sistema en formato PDF, HTML y MS Word. Se integra con varias

herramientas como son: Eclipse, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic. Además, genera código Java. [11]

#### **1.6.4 Lenguaje de programación: Java**

Es un lenguaje de programación desarrollado por Sun Microsystems a principios de los 90. Está diseñado para ser lo suficientemente simple permitiendo que los programadores puedan lograr fluidez con el lenguaje. Trabaja con sus datos como objetos y con interfaces a estos, además proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando. [12] Sus principales características son:

Orientado a objetos: soporta las características esenciales del paradigma de la programación orientado a objetos: encapsulación, herencia y polimorfismo.

Robusto: elimina el uso de apuntadores para referenciar áreas de memoria, además libera al desarrollador de la necesidad de desalojar la memoria que la aplicación ya no usa. Además, requiere la declaración explícita tanto de los tipos de datos como de los métodos.

Multiplataforma: el mismo código Java que funciona en un sistema operativo, funciona en cualquier otro sistema operativo que tenga instalada la máquina virtual de Java.

Multitareas: A pesar de que las capacidades multitarea que pueden ser implementadas en Java dependen en gran parte del sistema operativo en el cual se ejecuten, digamos Windows o Unix, dichas capacidades superan en gran manera a los entornos de flujo único que ofrecen otros lenguajes de programación. Permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución.

#### **1.6.5 Entorno Integrado de Desarrollo: Eclipse**

En su web oficial se define a Eclipse como “An IDE for everything and for nothing in particular” (un IDE para todo y para nada en particular). Eclipse es una de las herramientas que se engloban bajo el denominado Proyecto Eclipse. Es una plataforma basada en Java y de tipo código abierto. Este IDE ofrece el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Su misión consiste en evitar tareas repetitivas, facilitar la escritura correcta del código, disminuir el tiempo de depuración e incrementar la productividad del desarrollador.

### **1.6.6 Librería JFreeChart**

JFreeChart es una librería para gráficos escrita 100% en Java que facilita mostrar gráficos de calidad profesional en las aplicaciones, ya sean Web o de escritorio. Entre las características principales de esta biblioteca están: un API consistente y bien documentada con soporte para un amplio rango de tipos de gráficas, un diseño flexible fácilmente entendible, y la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) y de cliente (Swing, por ejemplo), soporte para varios tipos de salida, incluyendo componentes Swing, archivos de imagen como PNG y JPEG, y formatos gráficos de vectores (incluyendo PDF, EPS y SVG), es código abierto, más específicamente, Software Libre, está distribuido bajo la licencia LGPL, que permite el uso en aplicaciones propietarias. [13]

### **1.6.7 Librería JasperReport**

JasperReports es una herramienta de creación de informes Java libre que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. Está escrito completamente en Java y puede ser usado en gran variedad de aplicaciones de Java, incluyendo J2EE o aplicaciones web, para generar contenido dinámico.

Su propósito principal es ayudar a crear documentos de tipo páginas, preparados para imprimir en una forma simple y flexible. Se usa comúnmente con iReport, un front-end gráfico de código abierto para la edición de informes.

## **1.7 Conclusiones**

En este capítulo se realizó un estudio de los diseñadores de reportes en el mundo decidiéndose hacer uso de la herramienta iReports debido a que ofrece potentes funcionalidades de diseño. Además es muy poderosa y de código abierto. Su uso es adoptado por una amplia gama de usuarios en el mundo y existe una amplia documentación libre al respecto. La tendencia ha sido reutilizar la herramienta, tomando los componentes necesarios, haciéndole las modificaciones pertinentes y la personalización que se ajuste al Reporteador Dinámico. Además se definió el uso de las librerías JFreeChart-1.0.13 y JasperReport-3.7.1, poderosas herramientas para generar reportes en Java. Se decidió utilizar como metodología de desarrollo OpenUp, Visual Paradigm v6.4 como herramienta CASE, lenguaje de programación JAVA y como Entorno Integrado de Desarrollo Eclipse v3.3.

## **CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA**

### **2.1 Introducción**

En este capítulo se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el diagrama de caso de usos del sistema, la descripción de los mismos y el modelo de dominio.

### **2.2 Breve descripción del sistema**

El diseñador de reportes permitirá al usuario poder configurar una plantilla para el reporte a generar, permitiéndole elegir la apariencia de sus informes tal y cual desee, con un área de trabajo agradable, lo cual facilita la interacción de los usuarios con la aplicación, sin que estos tengan grandes conocimientos informáticos.

### **2.3 Modelo de Dominio**

El modelo del dominio recoge los tipos de objetos –las clases- más importantes. Como objetos importantes, se establece la clasificación siguiente:

- Objetos del negocio.
- Objetos del mundo real y conceptos relacionados con éstos.
- Acontecimientos.

Para realizar el modelo de dominio, se utiliza el diagrama de clases del UML.

Al modelar el contexto, hay que tener presente que se trata de hacer un modelo del entorno del software, y no del software. [14]

A continuación se muestra el modelo conceptual de la problemática a resolver, identificando los objetos fundamentales y sus relaciones.

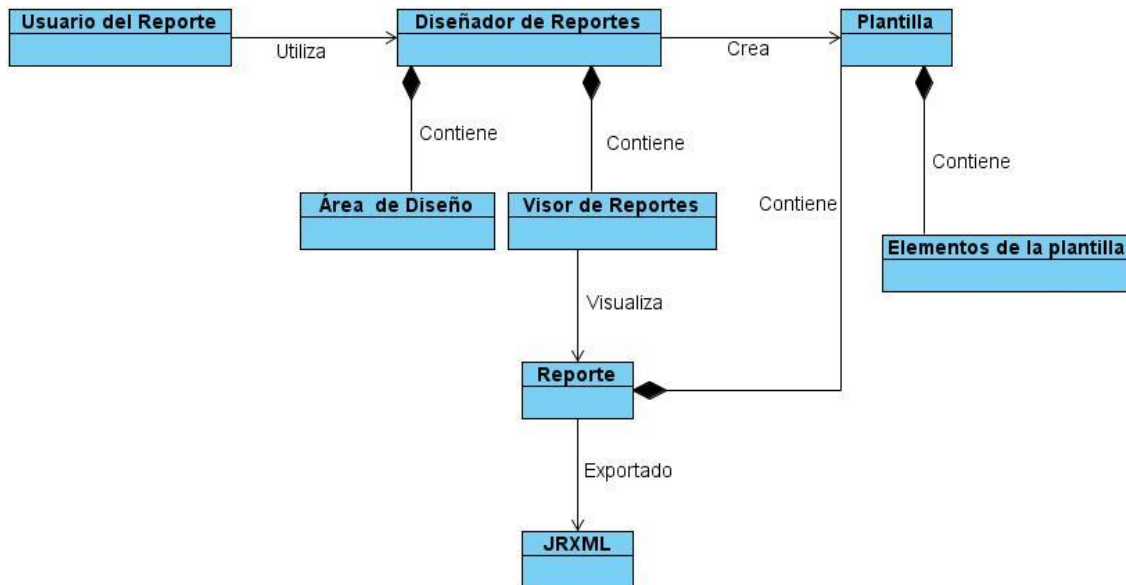


Figura 2.1 Diagrama del Modelo de Dominio.

Para una mejor comprensión, a continuación se describen cada una de las clases que intervienen en el diagrama mostrado.

**Usuario del Reporte:** Cualquier persona que interactúe con la aplicación y necesite diseñar reportes.

**Diseñador de Reportes:** Aplicación visual para el diseño de reportes.

**Área de Diseño:** Región del Diseñador de Reporte que permite la creación y edición de plantillas.

**Plantilla:** Modelo que define la presentación de los datos en los reportes.

**Visor de Reportes:** Área de visualización del informe en tiempo de diseño.

**JRXML:** Es un documento XML que contiene la definición del diseño del informe.

**Elementos de la plantilla:** Elementos que se insertan en la plantilla del reporte a generar.

**Reporte:** Información de interés para el usuario.

## 2.4 Especificación de Requerimientos de Software

La IEEE (Institute of Electrical and Electronics Engineers) define un requerimiento como una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Un requerimiento no es más que esa condición que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. Por lo tanto, los requerimientos de software le brindan al usuario la posibilidad de que el sistema cumpla las condiciones

que le interesen, siendo estos una manera de verificar la calidad del producto. Por esto, el levantamiento de requerimientos se hace fundamental a la hora de desarrollar cualquier sistema.

Estos requerimientos se dividen en dos grupos, los requerimientos funcionales, que son las capacidades o condiciones que debe cumplir el sistema, y los no funcionales que son cualidades o propiedades que el producto debe tener.

### **2.4.1 Requerimientos Funcionales (RF)**

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. La aplicación a desarrollar debe cumplir los siguientes requisitos:

RF1 Crear plantilla.

RF2 Definir el formato de la plantilla.

RF3 Gestionar elementos de la plantilla.

RF3.1 Insertar elemento de la plantilla.

RF3.2 Eliminar elemento de la plantilla.

RF3.3 Modificar elemento de la plantilla.

RF4 Crear JRXML.

RF5 Visualizar el reporte que se está diseñando.

RF6 Abrir fichero.

RF7 Modificar plantilla.

### **2.4.2 Requerimientos No Funcionales**

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, y que harán del mismo un sistema confiable y seguro.

#### **Requisitos de Usabilidad**

- La interfaz de usuario podrá ser utilizada por usuarios con pocos o ningún conocimiento informático.

#### **Requisitos de Software**

- Máquina virtual de Java (JDK) 1.5 o superior.
- Sistema Operativo Microsoft Windows 95 o superior.
- Sistema Operativo Linux con ambiente gráfico KDE o GNOME.

### Requisitos de Hardware

- Se requiere un mínimo de 256 Mb de RAM, 512 Mb recomendado.
- 1.3 GHz de velocidad de procesamiento.

### Requerimientos de Apariencia e Interfaz Externa

- La interfaz de usuario debe ser agradable y sencilla para el diseño de reportes.

### Requerimientos de Portabilidad

- La aplicación debe correr tanto en el sistema operativo Windows como en Linux.

## 2.5 Actores y Casos de Uso del Sistema

La forma en que los actores usan el sistema es representada a través de los casos de usos. Estos últimos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor.

### 2.5.1 Actores del Sistema

Se le llama actor a toda entidad externa al sistema que guarda una relación con este y que le demanda una funcionalidad. En el sistema se identificó el siguiente actor:

<b>Actor</b>	<b>Descripción</b>
Usuario del Reporte	Representa al usuario que va a hacer uso del sistema, y quien tiene la posibilidad de interactuar con todas las funcionalidades de este para diseñar reportes.

Tabla 2: Actor del Sistema y su descripción.

### 2.5.2 Casos de Uso del Sistema

La forma en que los actores usan el sistema es representada a través de los casos de usos. Estos últimos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. Los casos de usos identificados en el presente trabajo son enunciados a continuación:

Orden	Nombre	Prioridad	Breve descripción
1	Crear plantilla.	Crítico	El CU inicia cuando el Usuario del Reporte interactúa con el sistema para crear la plantilla del reporte a generar, gestiona los elementos de la misma, indica visualizar el reporte en tiempo de diseño y guarda la plantilla con el nombre y el destino elegido finalizando así el CU.
2	Modificar plantilla.	Crítico	El CU inicia cuando el Usuario del Reporte abre la plantilla a modificar en el sistema, cambia las propiedades deseadas, indica visualizar el reporte en tiempo de diseño y guarda la plantilla finalizando así el CU.
3	Gestionar elementos de la plantilla.	Crítico	El CU inicia cuando el Usuario del Reporte decide gestionar los elementos de la plantilla: insertar elemento, eliminar elemento, modificar elemento, finalizando así el CU.

Tabla 3: Casos de Uso del Sistema y su descripción.

### 2.5.3 Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso del sistema sirve para especificar la comunicación y el comportamiento de un sistema, mediante su interacción con los usuarios y/u otros sistemas. Proporcionan un modo claro y preciso de comunicación entre cliente y desarrollador. A continuación se muestra, en la figura 2.2, el diagrama de casos de uso del sistema del presente trabajo.

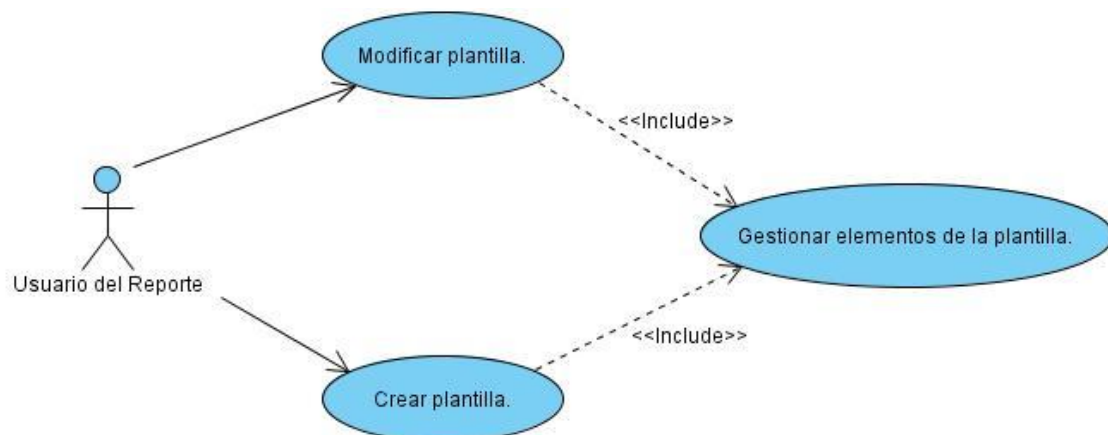


Figura 2.2. Diagrama de Casos de Uso del Sistema.

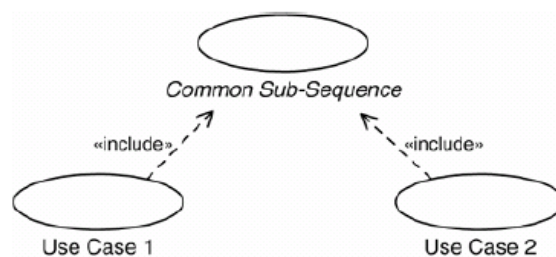


## 2.6 Patrones de CU utilizados

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. A continuación se describen los patrones de casos de uso utilizados en la modelación del sistema.

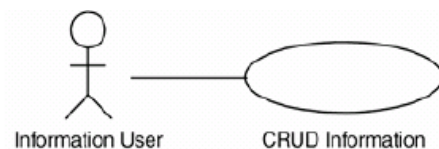
### 2.6.1 Concordancia (Commonality). Reusabilidad

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado. Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos. [15]



### 2.6.2 CRUD (Creating, Reading, Updating, Deleting). Completo

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual. Consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.



## 2.7 Descripción textual de los Casos de Uso del Sistema

### 2.7.1 Caso de Uso: Crear plantilla.

<b>Caso de Uso:</b>	Crear plantilla.
<b>Actores:</b>	Usuario del Reporte (inicia).
<b>Resumen:</b>	El CU inicia cuando el Usuario del Reporte interactúa con el sistema para crear la plantilla del reporte a generar, elige gestionar los elementos de la misma, indica visualizar el reporte en tiempo de diseño y guarda la plantilla con el nombre y el destino elegido finalizando así el CU.
<b>Precondiciones:</b>	Reporteador en funcionamiento.
<b>Referencias:</b>	RF1, RF2, RF3, RF3.1, RF3.2, RF3.3, RF4, RF5.
<b>CU Asociados:</b>	
<b>Prioridad:</b>	Crítico
<b>Flujo Normal de Eventos</b>	
Acción del Actor	Respuesta del Negocio
1. El Usuario del Reporte elige la opción “Nueva Plantilla”.	1.1 La aplicación muestra el cuadro de diálogo “Propiedades del Reporte”, para la creación de una nueva plantilla.
2. Entra los datos requeridos: - “Nombre de la plantilla”. - “Formato de la plantilla”. Y da clic en el botón “Aceptar”.	2.1 La aplicación muestra la interfaz correspondiente para el diseño de la plantilla y ofrece la opción de visualizar la plantilla mientras se va diseñando, descrito en la sección 1.
3. Procede a gestionar los elementos de la plantilla, procedimiento descrito en el CU Gestionar elementos de la plantilla.	3.1 La aplicación realiza los cambios según la acción del Usuario del Reporte.
4. Solicita guardar la plantilla mediante la opción “Guardar”.	4.1 La aplicación muestra el cuadro de diálogo “Guardar Plantilla” que da la opción de elegir nombre y destino de la plantilla a guardar.
5. Elige la dirección e inserta el nombre de la	5.1 Guarda la plantilla creada con el nombre y el

plantilla y da clic en el botón “Guardar”.	destino elegido, en formato JRXML y finaliza el CU.
<b>Flujo Alternativo 1</b>	
2. Elige la opción “Cancelar”.	2.1 La aplicación cierra el cuadro de diálogo y finaliza el CU.
<b>Flujo Alternativo 2</b>	
5. Elige la opción “Cancelar”.	5.1 La aplicación cierra el cuadro de diálogo y finaliza el CU.
<b>Sección 1: “Visualizar Reporte”</b>	
1. Si ha guardado la plantilla selecciona la opción “Vista Previa”.	1.1 La aplicación muestra en una ventana la vista previa del reporte que se está diseñando.
<b>Flujo Alternativo 3</b>	
	1.1 La aplicación indica guardar la plantilla, ir al paso 4.1.
	1.2 La aplicación muestra en una ventana la vista previa del reporte que se está diseñando.
<b>Pos condiciones</b>	Se ha creado, guardado la plantilla, y visualizado el reporte en tiempo de diseño.
<b>Interfaces</b>	Ver figura 4.6.

Tabla 4. Descripción textual del Caso de Uso Crear plantilla.

### 2.7.2 Caso de Uso: Modificar plantilla.

<b>Caso de Uso:</b>	Modificar plantilla.
<b>Actores:</b>	Usuario del Reporte.
<b>Resumen:</b>	El CU inicia cuando el Usuario del Reporte abre la plantilla a modificar en el sistema, cambia las propiedades deseadas, indica visualizar el reporte en tiempo de diseño y guarda la plantilla finalizando así el CU.
<b>Precondiciones:</b>	El Usuario del Reporte debe haber cargado el fichero que desea modificar en la

	aplicación.
<b>Referencias:</b>	RF3, RF3.1, RF3.2, RF3.3, RF4, RF5, RF6, RF7.
<b>CU Asociados:</b>	
<b>Prioridad:</b>	Crítico
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Negocio</b>
1. El Usuario del Reporte selecciona la opción “Abrir”	1.1 La aplicación muestra un cuadro de diálogo “Cargar JRXML”, que da la opción de elegir la dirección del fichero a cargar.
2. El Usuario del Reporte busca y elige el fichero a cargar y presiona el botón “Abrir”.	2.1 Lee la dirección del fichero señalado.
	2.2 Comprueba la validez de los datos del fichero y si el fichero aun no se ha cargado.
	2.3 Si son correctos los datos y el fichero no existe aún en el sistema, procede a cargarlo.
3. Necesita realizar alguna de estas operaciones para modificar la plantilla: - “Gestionar elementos de la plantilla”. - “Modificar estructura de la plantilla”.	3.1 El sistema ejecuta alguna de las siguientes acciones:  - “Gestionar elementos de la plantilla”, descrito en el CU Gestionar elementos de la plantilla. - “Modificar estructura de la plantilla”, descrito en la sección 1.
4. Solicita guardar la plantilla con los cambios efectuados mediante la opción “Guardar”.	4.1 La aplicación muestra el cuadro de diálogo “Guardar Plantilla” que da la opción de elegir nombre y destino de la plantilla a guardar.
5. Elige la dirección e inserta el nombre de la plantilla y da clic en el botón “Guardar”.	5.1 Guarda la plantilla con el nombre y el destino elegido en formato JRXML y finaliza el CU.
<b>Flujo Alterno 1</b>	
	2.2.1 Si los datos no son correctos o el fichero ya

	está cargado, el sistema muestra un mensaje indicándolo.
3 Indica abrir otro fichero.	3.1 El sistema va al paso 1.1 del flujo normal de eventos.
<b>Flujo Alterno 2</b>	
5. El Usuario del Reporte presiona el botón “Cancelar”.	5.1 La aplicación cierra el cuadro de diálogo.
<b>Sección 1: “Modificar estructura de la plantilla”</b>	
1. El Usuario del Reporte da clic en la opción “Propiedades de la banda”, selecciona la banda a modificar y cambia sus propiedades.	1.1 La aplicación ejecuta los cambios en dependencia de la acción que haga el actor y da la opción de visualizar la plantilla con los nuevos cambios que se van realizando, esto se describe en la sección 2.
<b>Sección 2: “Visualizar Reporte”</b>	
1. Si ha guardado la plantilla selecciona la opción “Vista Previa”.	1.1 La aplicación muestra en una ventana la vista previa del reporte que se está diseñando.
<b>Flujo Alterno</b>	
	1.1 La aplicación indica guardar la plantilla, ir al paso 4.1.
	1.2 La aplicación muestra en una ventana la vista previa del reporte que se está diseñando.
<b>Pos condiciones</b>	Se ha modificado y guardado la plantilla con los nuevos cambios, además se ha visualizado el reporte en tiempo de diseño.
<b>Interfaces</b>	Ver figura 4.7.

Tabla 5. Descripción textual del Caso de Uso Modificar plantilla.

### 2.7.3 Caso de Uso: Gestionar elementos de la plantilla.

<b>Caso de Uso:</b>	Gestionar elementos de la plantilla.
<b>Actores:</b>	Usuario del Reporte.
<b>Resumen:</b>	El CU inicia cuando el Usuario del Reporte decide gestionar los elementos de la plantilla: insertar elemento, eliminar elemento, modificar elemento, finalizando así el CU.
<b>Precondiciones:</b>	Que exista al menos una plantilla abierta en la aplicación.
<b>Referencias:</b>	RF3, RF3.1, RF3.2, RF3.3.
<b>CU Asociados:</b>	Modificar plantilla (Inclusión), Crear plantilla (Inclusión).
<b>Prioridad:</b>	Crítico
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Negocio</b>
1. El Usuario del Reporte puede realizar varias acciones para gestionar los elementos de la plantilla: - “Insertar los elementos de la plantilla”. - “Eliminar los elementos de la plantilla”. - “Modificar los elementos de la plantilla”.	1.1 La aplicación ejecuta una de las siguientes secciones:  - “Insertar los elementos de la plantilla”, ir a la sección 1. - “Eliminar los elementos de la plantilla”, ir a la sección 2. - “Modificar los elementos de la plantilla”, ir a la sección 3.
<b>Sección 1: “Insertar elementos de la plantilla”</b>	
1. El Usuario del Reporte selecciona el elemento y lo inserta en la plantilla.	1.1 La aplicación muestra el elemento insertado en la plantilla y finaliza el CU.
<b>Sección 2: “Eliminar elementos de la plantilla”</b>	
1. El Usuario del Reporte selecciona en la plantilla el objeto a eliminar, y presiona la tecla suprimir.	1.1 La aplicación elimina el objeto seleccionado de la plantilla y finaliza el CU.
<b>Sección 3: “Modificar elementos de la plantilla”</b>	
1. El Usuario del Reporte selecciona el elemento a	1.1 La aplicación muestra las propiedades del

modificar.	elemento seleccionado.
2. El Usuario del Reporte modifica las propiedades deseadas del elemento.	2.1 La aplicación guarda los cambios, muestra el elemento según las nuevas propiedades y finaliza el CU.
<b>Pos condiciones</b>	En dependencia de la acción del Usuario del Reporte se ha: <ul style="list-style-type: none"> <li>- Insertado elemento en la plantilla.</li> <li>- Eliminado elemento en la plantilla.</li> <li>- Modificado elemento en la plantilla.</li> </ul>
<b>Interfaces</b>	Ver figura 4.7.

Tabla 6. Descripción textual del Caso de Uso Gestionar elementos de la plantilla.

## 2.8 Conclusiones

En el presente capítulo se definió el modelo conceptual para identificar los conceptos más significativos del dominio del problema. Se analizaron y aprobaron los requisitos funcionales necesarios para obtener un sistema eficiente, se definieron las responsabilidades del usuario de la aplicación mediante un diagrama de casos de uso del sistema dando cumplimiento a las necesidades de los requisitos funcionales propuestos y se realizó además la descripción de los casos de uso involucrados en el desarrollo de la aplicación.

## CAPÍTULO 3: DISEÑO DEL SISTEMA

### 3.1 Introducción

En este capítulo se describe el estilo arquitectónico del sistema, haciendo énfasis en los patrones de diseño utilizados. Además, se desarrolla el diseño del sistema.

### 3.2 Estilo de arquitectura

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software [16]. El estilo de arquitectura empleado es dentro de los de llamada y retorno, el Modelo Vista Controlador (MVC). Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. [17]

**Modelo:** Esta es la 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:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

**Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Los elementos dispuestos en estos componentes tuvieron modificaciones (no están presentes todos los elementos del iReport), con el objetivo de integrar el sistema al Reporteador Dinámico y cubrir solamente los requisitos propuestos para la solución, como muestra la figura 3.1.



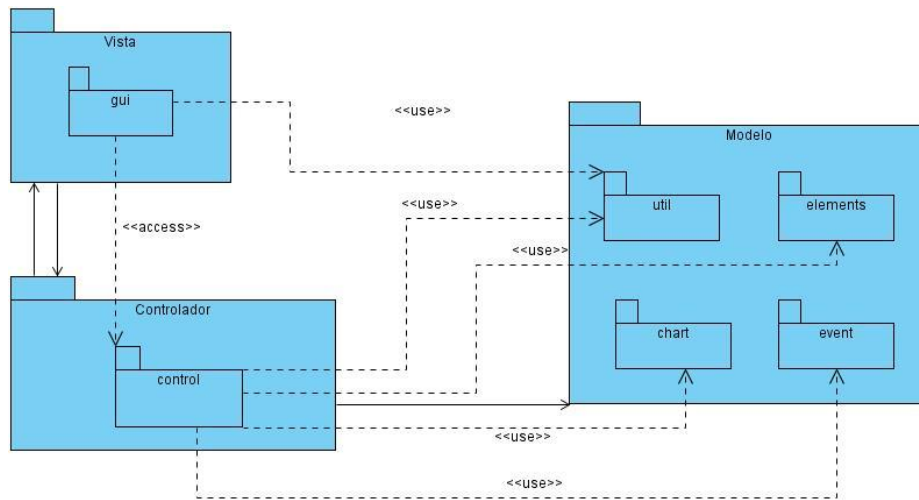


Figura 3.1. Ejemplo de aplicación del patrón MVC.

En el componente **Modelo** se usaron algunas de las entidades definidas por el iReport debido a que no todas son necesarias para el sistema, por ejemplo, no se tuvieron en cuenta las clases que modelan la conexión a la base de datos, así como las consultas y su respectiva ejecución. Además se incorporaron nuevas funcionalidades a algunas entidades que se tomaron, como *TextFieldReportElement.java* que modela al elemento *campo* ( $\$F\{field\}$ ) y *ChartReportElement2.java* que modela al elemento *gráfico* (Chart), para así lograr obtener los datos que provee el Reporteador Dinámico, a través de la interfaz *JRDataSource*, y así permitir que al gestionar alguno de estos elementos sea de forma dinámica con la información de la base de datos.

En el componente **Vista** se utiliza gran parte de la interfaz definida por la herramienta mencionada anteriormente, pero se introdujeron cambios en su filosofía puesto que se organizaron en distintas clases los componentes visuales debido a que estos estaban agrupados en una sola clase (*MainFrame.java*), logrando así una mejor integración, pues el Reporteador Dinámico tiene su interfaz definida y el módulo implementado debe seguir el mismo estilo visual. Un ejemplo de esto, es la creación de la clase *ToolBar.java*, la cual tiene la responsabilidad de indicarle al controlador qué acción se desea hacer con los elementos.

En el componente **Controlador** se incluyeron nuevas entidades, ejemplo *ReportDesignerController.java*, la cual controla los componentes sobre los cuales el usuario interactúa, además se encarga de crear los distintos reportes a generar (ver figura 3.5).

### 3.3 Patrones de Diseño

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos.

Los patrones utilizados en la solución del sistema se describen en las subsecciones siguientes.

#### 3.3.1 Patrón Observador

Problema: Se desea observar un objeto sujeto en una aplicación y notificar sus cambios. Los componentes deben notificar que se ha realizado una acción sobre ellos.

Solución: Suscribirse uno o más observadores a uno o más eventos del objeto sujeto. Cuando el evento es ejecutado los observadores son notificados. Cada observador implementa su propia función según su meta.

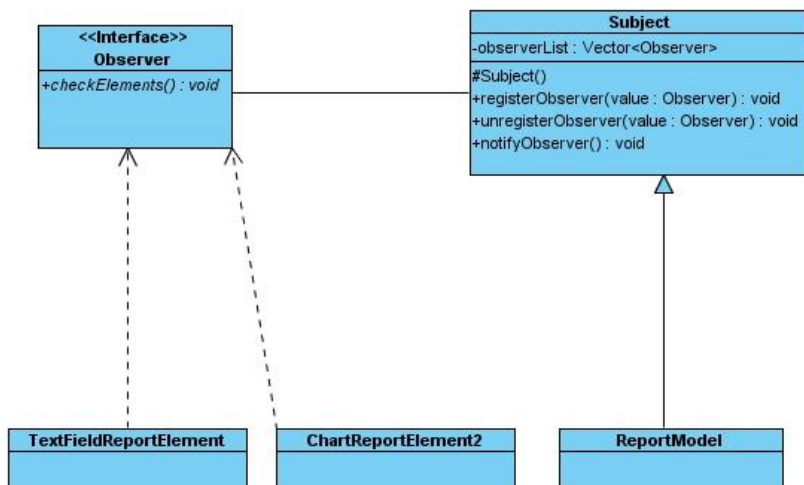


Figura 3.2. Ejemplo de aplicación del patrón Observador.

#### 3.3.2 Patrón Singleton

Problema: No deben existir varias instancias de la clase ReportDesignerController. No tiene sentido que existan varias instancias de esta clase y con un único objeto se puede acceder a todas sus acciones. Lo mismo ocurre en el caso de la clase ErrorReporter y esta solamente brinda una funcionalidad y no tiene sentido instanciarla para acceder a estos métodos.

Solución: Utilizar el patrón Singleton para restringir la creación de objetos pertenecientes a esa clase.

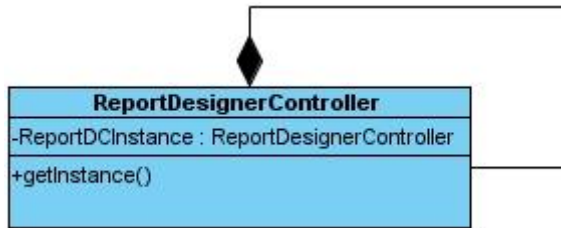


Figura 3.3. Ejemplo de aplicación del patrón Singleton.

### 3.3.3 Patrón Experto

Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. El patrón Experto asigna responsabilidades a las clases que tienen la información necesaria para cumplir con la responsabilidad.

Este patrón es utilizado en varias clases del sistema, ejemplo en la clase Report, puesto que contiene toda la información requerida al crear un reporte, por tanto, se le asigna la responsabilidad de crear los reportes.

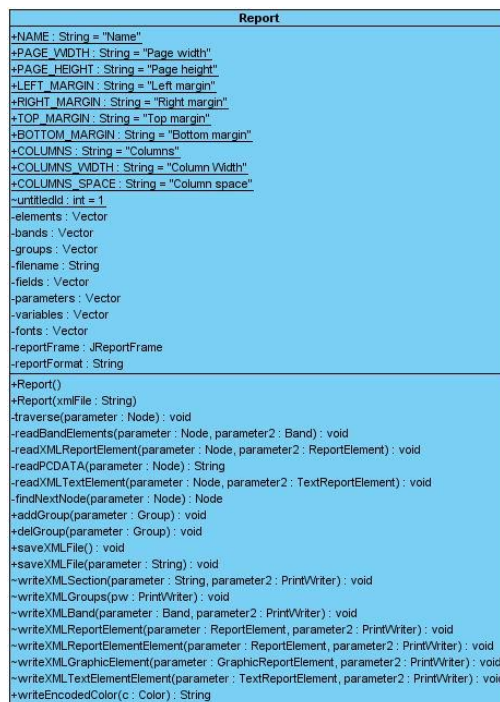


Figura 3.4: Ejemplo de aplicación del patrón Experto.

### 3.3.4 Patrón Creador

Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. Lo que define este patrón es que una instancia de un objeto la tiene que crear el objeto que tiene la información para ello. ¿Qué significa esto?, pues que si un objeto A utiliza específicamente otro B, o si B forma parte de A, o si A almacena o contiene B, o si simplemente A tiene la información necesaria para crear B, entonces A es el perfecto creador de B.

La clase ReportDesignerController contiene objetos JMDIDesktopPane, JReportFrame, ToolBar, Report, GroupsDialog y BandDialog; por tanto, este patrón sugiere que ReportDesignerController es idónea para asumir la responsabilidad de crear instancias de las clases mencionadas anteriormente.

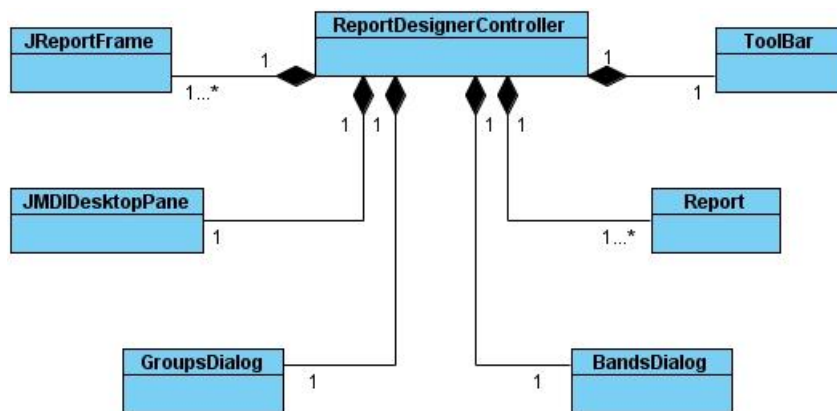


Figura 3.5: Ejemplo de aplicación del patrón Creador.

### 3.4 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización de casos de uso, y sirve como una abstracción del modelo de implementación y su código fuente. Es una solución amplia, que abarca todos los artefactos compuestos: clases de diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos.

A continuación se presentan los temas relacionados con el diseño del Diseñador de Reportes.

### 3.4.1 Diagrama de subsistemas

En la figura 3.6, se muestran los módulos o subsistemas que se relacionan con el sistema. De ellos sólo se explicará el módulo “Diseñador de Reportes”, debido a que representa la construcción de la solución propuesta. Los módulos “Reporteador Dinámico” e “iReport” no aparecen descritos, pues el primero ya ha sido desarrollado y detallado en el trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas de los autores José Rolando Lafaurié Olivares y Yanoski Agneri Martínez Hernández; mientras que en el caso del segundo puede visitarse la web oficial de iReport: <http://jasperforge.org/projects/ireport> para profundizar sobre el tema.

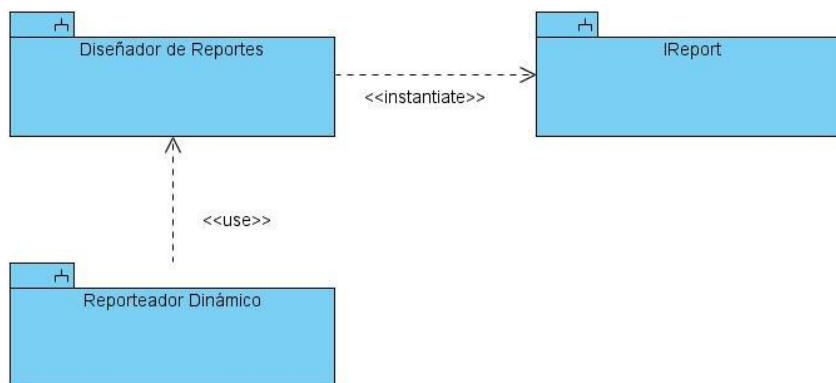


Figura 3.6. Vista general de los subsistemas y sus relaciones.

### 3.4.2 Diagrama de paquetes

En esta sección se explica el módulo “Diseñador de Reportes”, con el diagrama de paquetes que lo componen, teniendo en cuenta su relación, acompañado de una breve descripción.

El objetivo principal del Diseñador de Reportes es proporcionar al usuario no experimentado un ambiente amigable para la creación y diseño de las plantillas de reportes que desee, o abrir y modificar una ya existente.

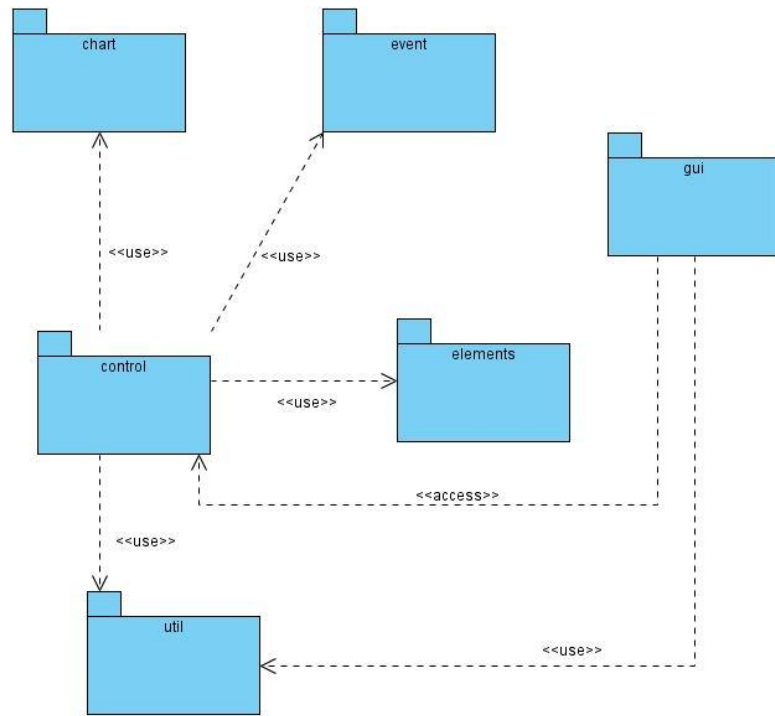


Figura 3.7. Paquetes relevantes para la arquitectura del Diseñador de Reportes.

### Descripción de los paquetes del Diseñador de Reportes

- **chart:** contiene las clases que definen los tipos de gráficos existentes en la plantilla: gráficos de barras, circular, etc.
- **control:** contiene las clases controladoras del sistema.
- **utils:** contiene las clases que permiten modelar las propiedades de los elementos visuales.
- **event:** contiene las clases que definen los eventos del sistema.
- **elements:** contiene las clases que definen los elementos disponibles para el diseño del reporte.
- **gui:** contiene las clases visuales del sistema.

### 3.4.3 Diagramas de Clases del Diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información:

- clases, asociaciones y atributos
- interfaces, con sus operaciones y constantes







### 3.5 Diagramas de Interacción

Un diagrama de interacción explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de estas. El punto de partida de las interacciones es el cumplimiento de las poscondiciones de los contratos de operación. [18]

El UML define dos tipos de estos diagramas; ambos sirven para expresar interacciones semejantes o idénticas de mensaje:

1. diagramas de colaboración
2. diagramas de secuencia

Los diagramas de secuencia dan una descripción gráfica de las interacciones del actor y de las operaciones a que da origen. Describen en el curso particular de los eventos de un caso de uso, los actores externos que interactúan directamente con el sistema (como caja negra) y con los eventos del sistema generados por los actores.

A continuación se describen los principales diagramas de interacción del sistema.

#### 3.5.1 Diagramas de Secuencias

##### Caso de Uso: Crear plantilla.

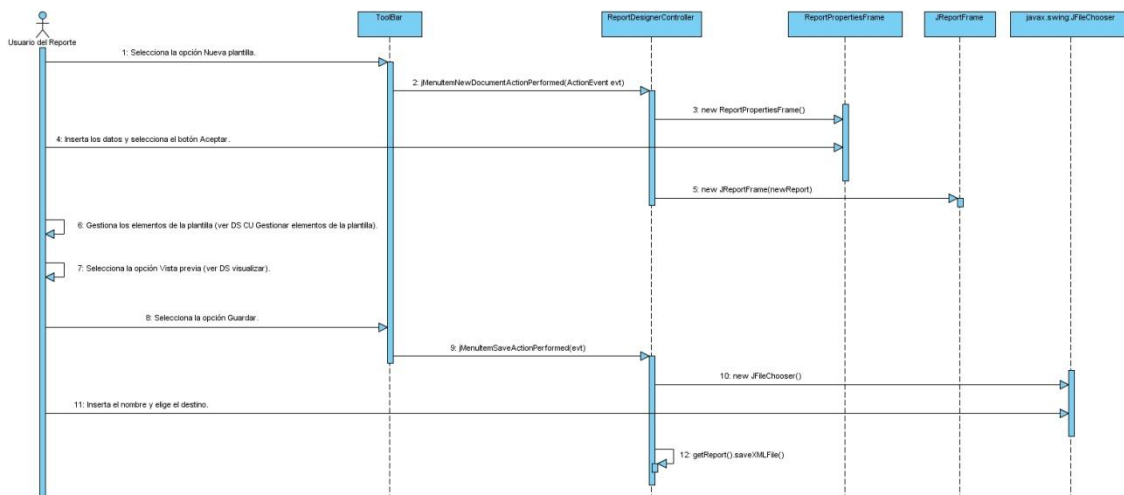


Figura 3.12. Diagrama de secuencia CU Crear plantilla, “flujo normal de eventos”.

## Caso de Uso: Modificar plantilla.

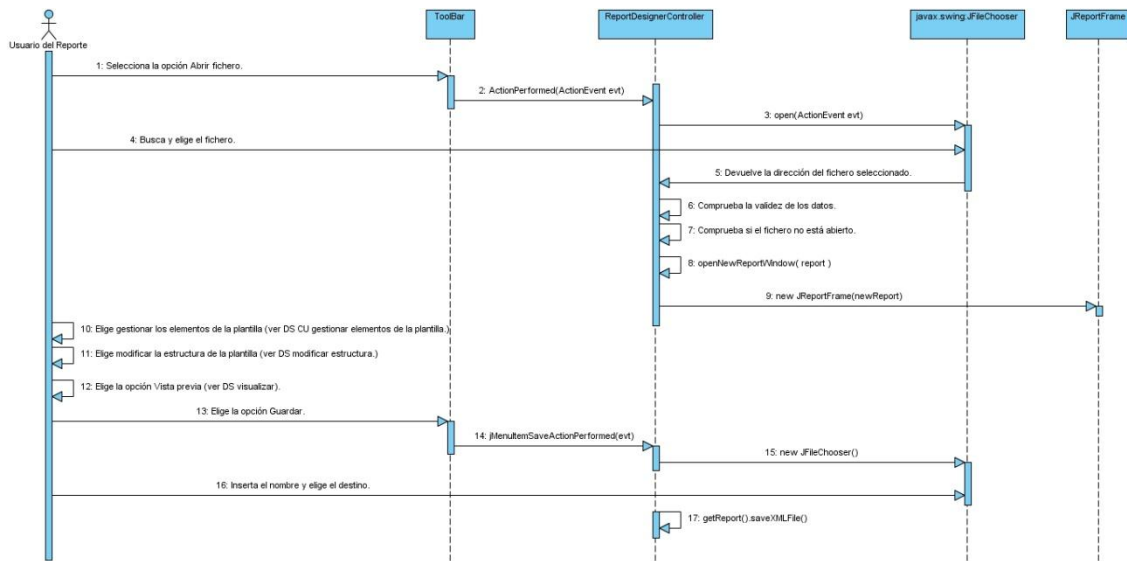


Figura 3.13. Diagrama de secuencia CU Modificar plantilla, flujo normal de eventos.

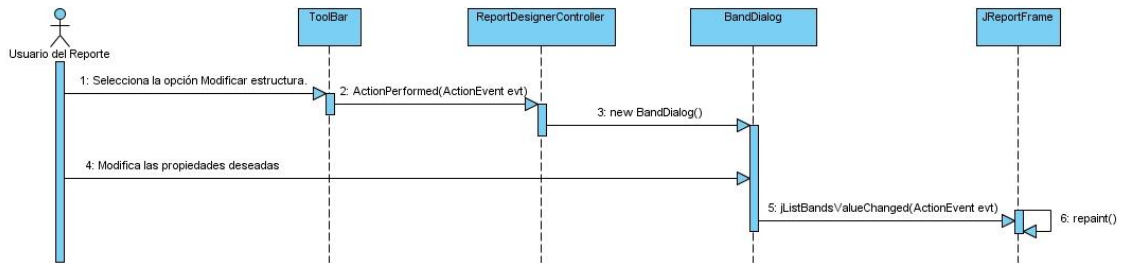


Figura 3.14. Diagrama de secuencia CU Modificar plantilla, sección "Modificar estructura de la plantilla".

### Caso de Uso: Gestionar elementos de la plantilla.

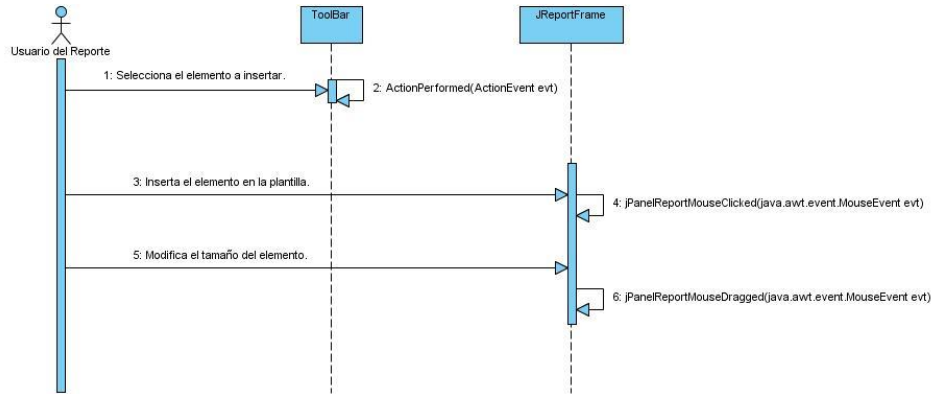


Figura 3.15. Diagrama de secuencia CU Gestionar elementos de la plantilla, sección "Insertar elemento de la plantilla".

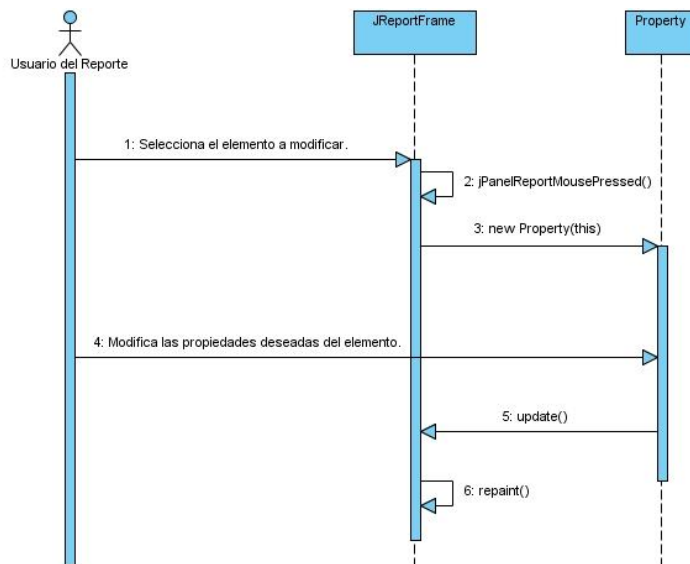


Figura 3.16. Diagrama de secuencia CU Gestionar elementos de la plantilla, sección "Modificar elemento de la plantilla".

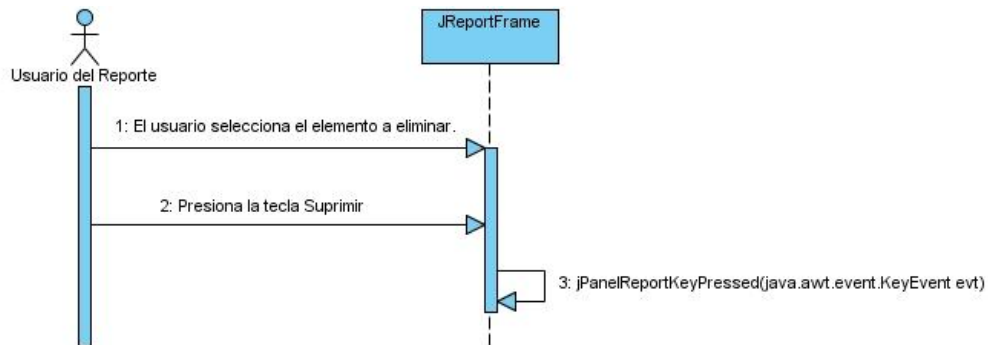


Figura 3.17. Diagrama de secuencia CU Gestionar elementos de la plantilla, sección “Eliminar elemento de la plantilla”.

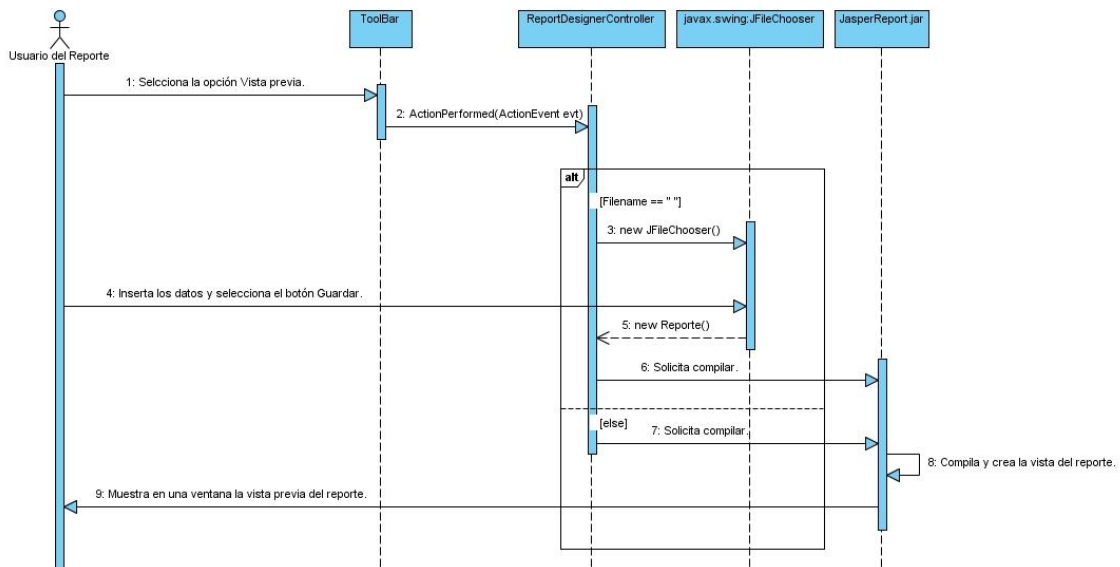


Figura 3.18. Diagrama de secuencia CU Crear plantilla y CU Modificar plantilla, sección “Visualizar”.

### 3.6 Diagrama de Despliegue

Los diagramas de despliegue muestran a los nodos procesadores la distribución de los procesos y de los componentes. [18]

A continuación se muestra el diagrama de despliegue del sistema, (se mantiene el mismo del Reporteador Dinámico), pues la solución implementada es un módulo.

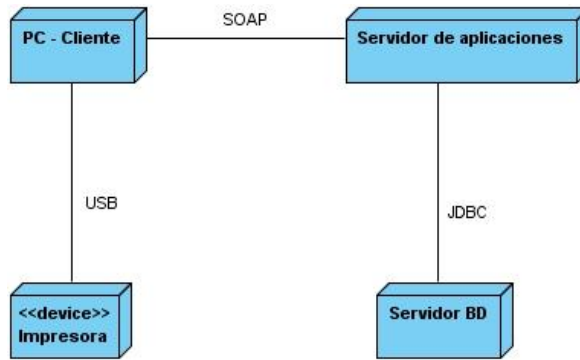


Figura 3.19. Diagrama de despliegue del sistema.

### 3.7 Conclusiones

Como resultado de este capítulo se obtuvo la arquitectura del sistema sustentada en el patrón Modelo Vista Controlador. Se describió el diseño del sistema desde la estructura de paquetes y subsistemas de diseño hasta los diagramas de clases del diseño. También se mostraron los diagramas de interacción de los principales escenarios del sistema. Además se mostró el diagrama de despliegue.

## CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

### 4.1 Introducción

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los casos de uso. Se incluirá el modelo de despliegue.

### 4.2 Diagrama de Componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Es un grafo donde los componentes están unidos por medio de relaciones de dependencia: compilación y ejecución. Son utilizados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

En la siguiente figura se representa el diagrama de componentes del sistema:

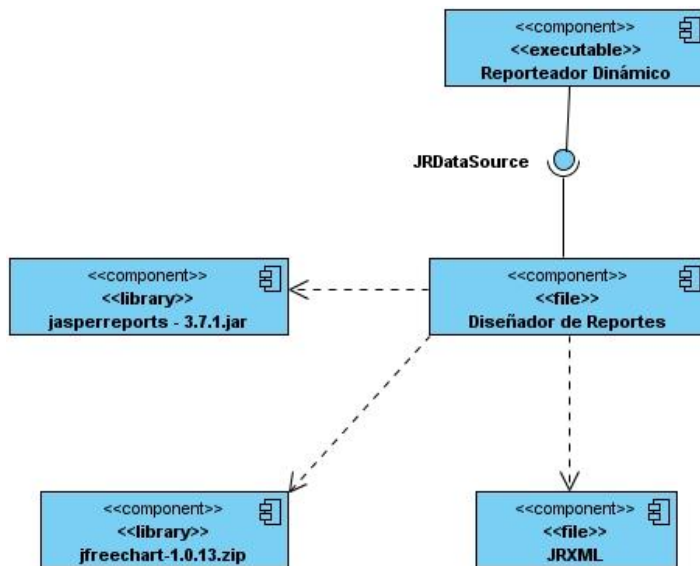


Figura 4.1: Diagrama de componentes.

En la figura 4.1 se muestran los componentes importantes del sistema y la interfaz a la que se accede para interactuar con el Reporteador Dinámico. El componente Diseñador de Reportes es el principal, pues representa el módulo desarrollado que implementa la interfaz `JRDataSource` para obtener los datos del

componente ejecutable Reporteador Dinámico, garantizando así la integración, usando además los componentes de tipo librería, jasperreports-3.7.1 y jfreechart-1.0.13 para realizar sus funcionalidades. El componente Diseñador de Reportes crea un fichero JRXML que define la estructura de los reportes diseñados.

#### 4.2.1 Diagrama de despliegue de los componentes

En este diagrama se muestra la distribución de los componentes por los distintos nodos del despliegue.

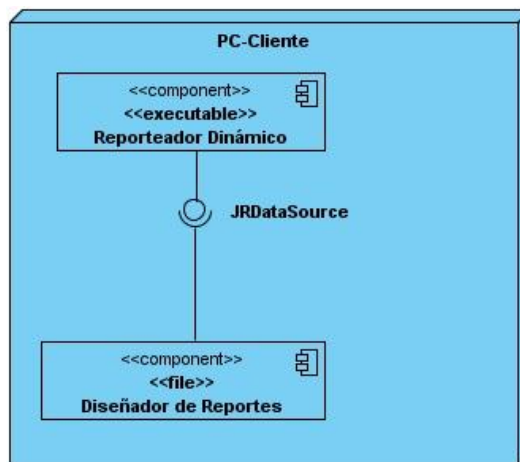


Figura 4.2: Ubicación de los componentes en el nodo del despliegue.

#### 4.3 Código fuente de los principales métodos y su descripción

A continuación se describen algunos de los principales métodos que fueron desarrollados para conformar la aplicación. Las descripciones se centran principalmente en los métodos que permiten dar cumplimiento a los requisitos funcionales.

El siguiente fragmento de código ilustra el método `newDocument()` de la clase `ReportDesignerController`, utilizado para crear el reporte a generar.

```
public void newDocument () {  
    ReportPropertiesFrame rpf = new ReportPropertiesFrame ();  
    rpf.setModal (true);  
    String name = getFirstNameFree ();  
    rpf.setReportName ( name );  
}
```

```

rpf.show();
if (rpf.getDialogResult() == javax.swing.JOptionPane.OK_OPTION)
{
    Report newReport = new Report();
    newReport.setWidth(rpf.getReportWidth());
    newReport.setHeight(rpf.getReportHeight());
    newReport.setOrientation(rpf.getOrientation());
    newReport.setName(rpf.getReportName());
    newReport.setTopMargin(rpf.getTopMargin());
    newReport.setLeftMargin(rpf.getLeftMargin());
    newReport.setRightMargin(rpf.getRightMargin());
    newReport.setBottomMargin(rpf.getBottomMargin());
    newReport.setColumnCount(rpf.getColumns());
    newReport.setColumnWidth(rpf.getColumnsWidth());
    newReport.setColumnSpacing(rpf.getColumnsSpacing());
    newReport.setIsSummaryNewPage(rpf.isSummaryOnNewPage());
    newReport.setIsTitleNewPage(rpf.isTitleOnNewPage());
    newReport.setWhenNoDataType(rpf.getWhenNoDataType());
    newReport.setEncoding(rpf.getXmlEncoding());
    newReport.setPrintOrder(rpf.getPrintOrder());
    newReport.setReportFormat(rpf.getReportFormat());
    openNewReportWindow( newReport );
}
}

```

**Resultado:**



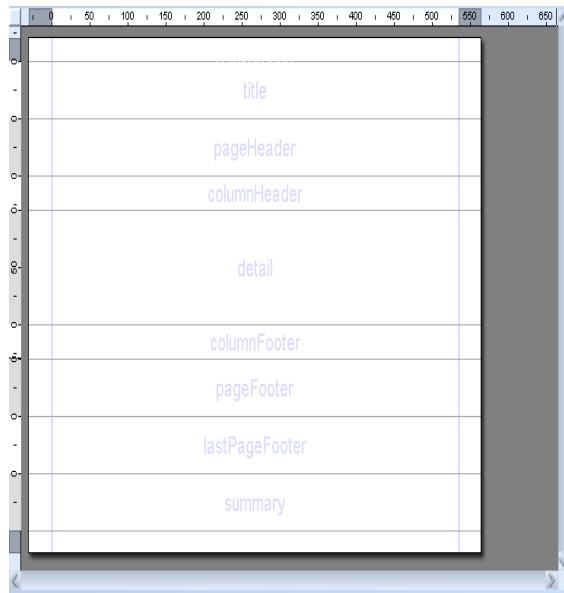


Figura 4.3: Resultado 1.

El siguiente código brinda la funcionalidad a la aplicación de salvar el reporte, en la clase ReportDesignerController.

```

public void save() {
    if (jMDIDesktopPane.getSelectedFrame() != null &&
        jMDIDesktopPane.getSelectedFrame().instanceof JReportFrame)
    {
        JReportFrame jrf = (JReportFrame)jMDIDesktopPane.getSelectedFrame();
        if (jrf.getReport().getFilename() == null ||
            jrf.getReport().getFilename().trim().equals(""))
        {
            javax.swing.JFileChooser jfc = new javax.swing.JFileChooser();
            jfc.setDialogTitle("Guardar reporte como archivo JRXML");
            jfc.setFileFilter( new javax.swing.filechooser.FileFilter() {
                public boolean accept(java.io.File file) {
                    String filename = file.getName();
                    return (filename.endsWith(".jrxml") || file.isDirectory()) ;
                }
                public String getDescription() {
                    return "JasperReports XML *.jrxml";
                }
            });
            jfc.setMultiSelectionEnabled(false);
            jfc.setAcceptAllFileFilterUsed(false);
        }
    }
}

```

```

jfc.setDialogType( javax.swing.JFileChooser.SAVE_DIALOG);
if (jfc.showSaveDialog( null) != javax.swing.JOptionPane.OK_OPTION) return;

        String file = jfc.getSelectedFile().getPath();
        if(!file.endsWith(".jrxml"))
            file += ".jrxml";
            boolean save = true;
JInternalFrame[] jrfs = getJMDIDesktopPane().getAllFrames();
    for (int j = 0; j < getJMDIDesktopPane().getAllFrames().length; j++) {

        if((JReportFrame)jrfs[j]).getReport().getFilename().equals(file))
            {
                save = false;
JOptionPane.showMessageDialog((JReportFrame)jrfs[j], "El sistema no puede asignar a un
documento el nombre de otro documento abierto.", "Error",JOptionPane.ERROR_MESSAGE);
                break;
            }
        }
        if(save){
            jrf.getReport().setFilename(file);

jrf.getReport().setName(jfc.getSelectedFile().getName());
            jrf.getReport().saveXMLFile();
            getJTreeFiles().updateUI();

        }

        }
        if(jrf.getReport().getFilename()!="")
jrf.getReport().saveXMLFile();
        getJTreeFiles().updateUI();

    }
}

```

Resultado:

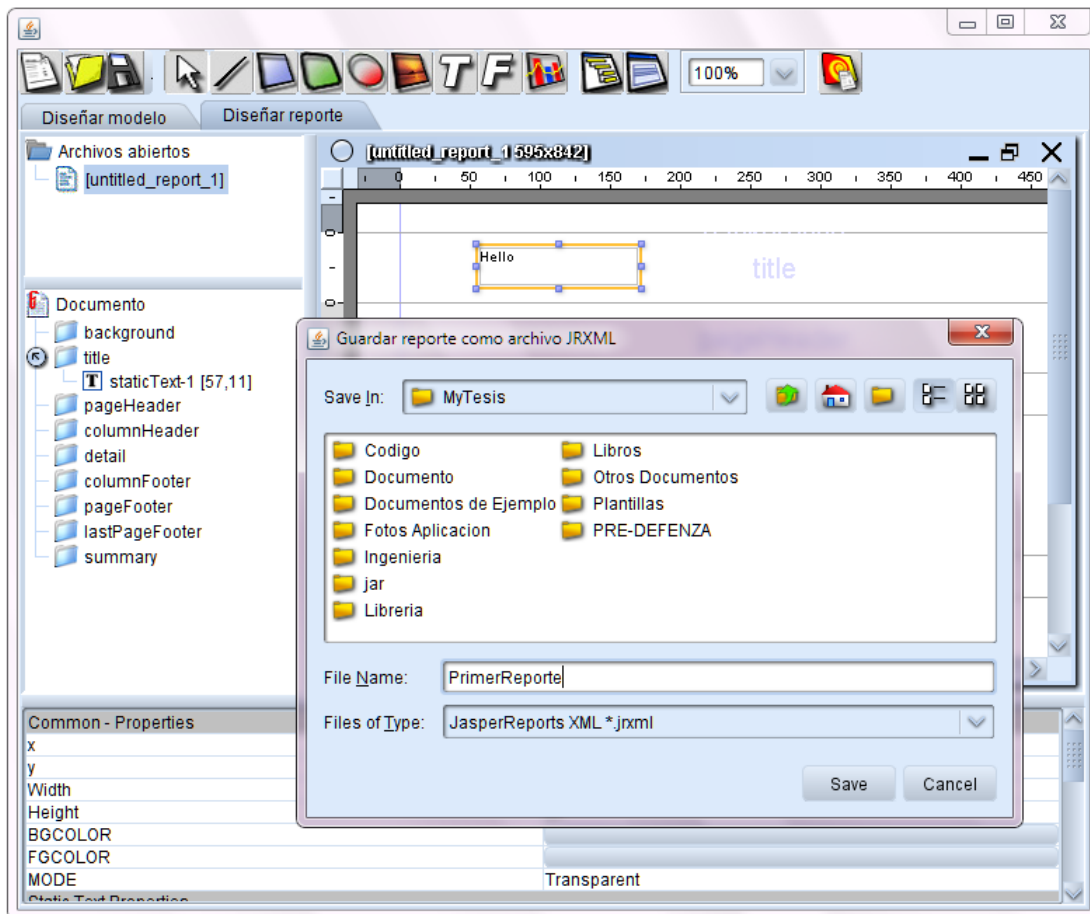


Figura 4.4: Resultado 2.

Se muestran fragmentos de código pertenecientes a los métodos `getTableModel()`, el cual devuelve el modelo proporcionado por el Reporteador Dinámico y `createReportDatasource()`, que tiene como función crear la fuente de datos (`DataSource`) con el modelo, implementados en la clase controladora `ReportDesignerController`.

```

public ReportArrayListTableModel getTableModel(){
    ReportArrayListTableModel model = null;
    if(clientReportControl.getCurrentModel().getListSelectedTable().size() > 0){
        try {
            model = new ReportArrayListTableModel(clientReportControl.getResult().getData(),
            clientReportControl.getResult().getColumnName());
        } catch (Exception e) {
            return null;
        }
    }
    return model;
}

private JRDataSource createReportDatasource(){
    ReportArrayListTableModel model = null;
    try {

        if(clientReportControl.getCurrentModel().getListSelectedTable().size() == 0)
            return new JREmptyDataSource();

        model = new ReportArrayListTableModel(clientReportControl.getResult().getData(),
        clientReportControl.getResult().getColumnName());
    } catch (Exception e) {
        e.printStackTrace();
    } catch (RemoteException e) {
        e.printStackTrace();
    }
    JRTableModelDataSource dataSource = new JRTableModelDataSource(model);

    return dataSource;
}

```

Figura 4.5. Fragmento de código de la clase ReportDesignerController.

#### 4.4 Pantallas principales

En la siguiente imagen se muestra la ventana principal donde el usuario diseñará sus reportes.

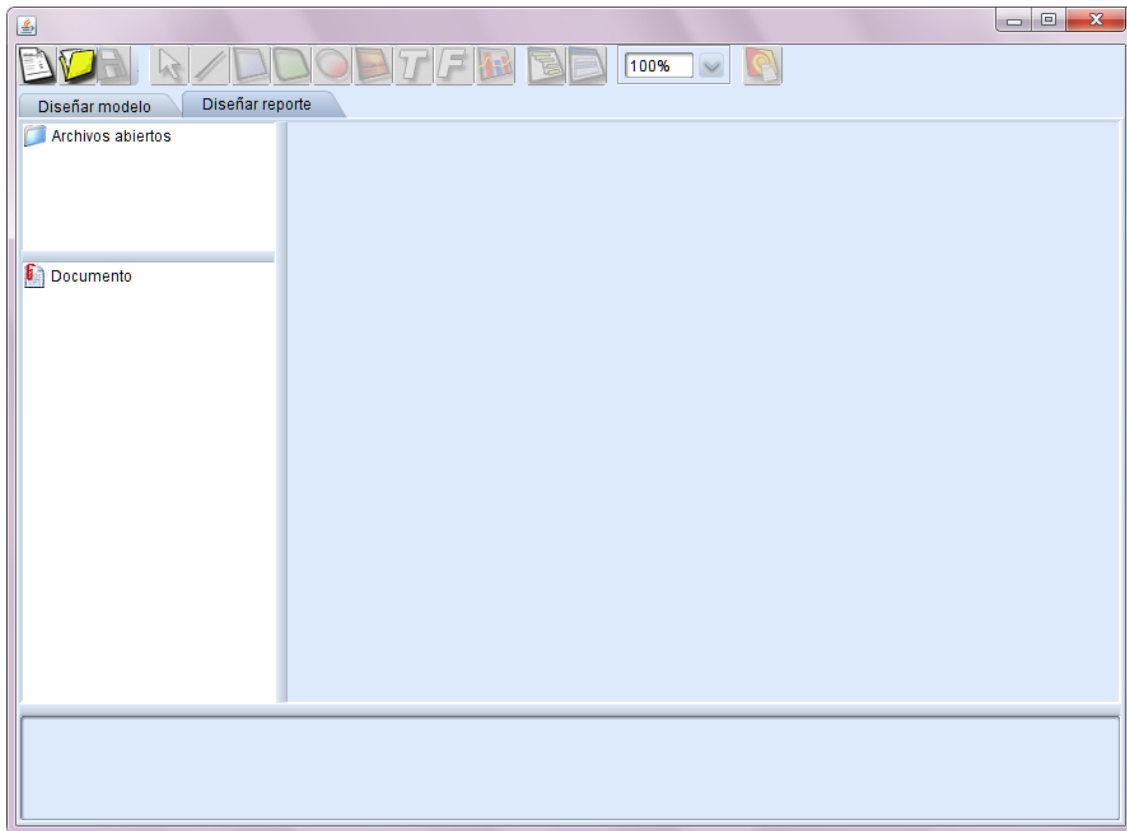


Figura 4.6: Interfaz principal.

En la siguiente imagen se representa el proceso de diseño de una plantilla.

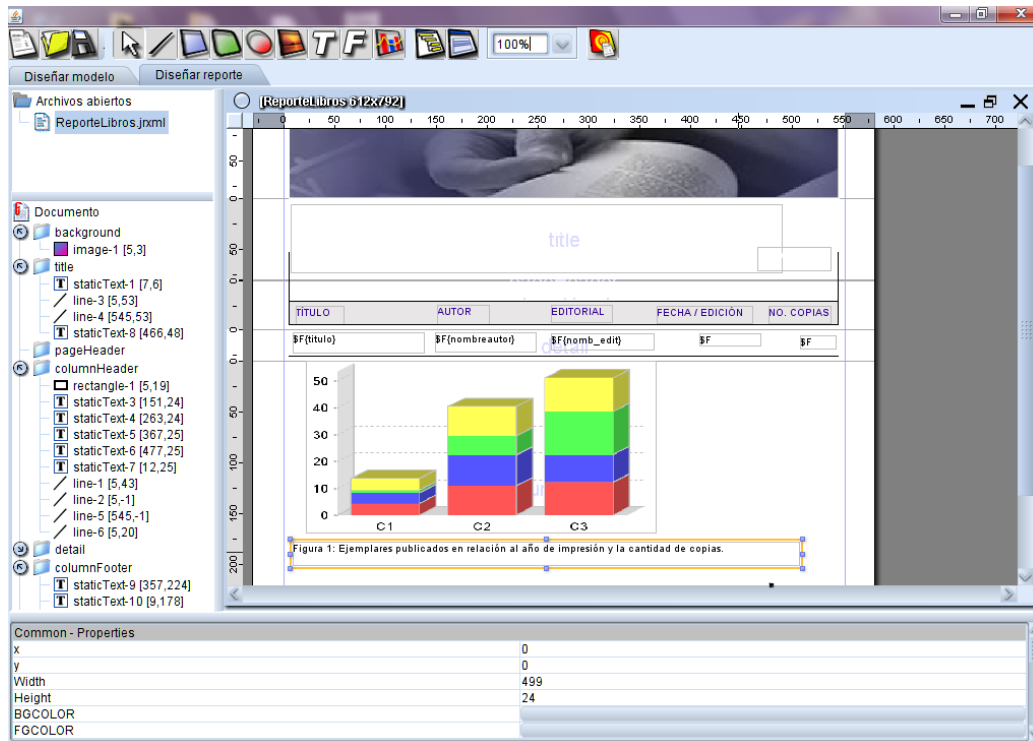


Figura 4.7: Vista que representa el proceso de diseño de la plantilla.

## 4.5 Modelo de pruebas

El flujo de trabajo de pruebas tiene como objetivo principal evaluar o valorar la calidad del producto a través de la búsqueda y documentación de errores, validando el cumplimiento de requerimientos, el desempeño y dando una indicación de calidad. La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Las pruebas de caja negra son las que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código). Para validar las principales funcionalidades del sistema se utilizaron las Pruebas de Caja Negra.

### 4.5.1 Casos de pruebas

Los escenarios de los casos de usos principales fueron probados para detectar no conformidades. En las siguientes subsecciones se representan estos casos de prueba separados por escenario.

#### 4.5.1.1 Crear plantilla.

<b>Id del escenario</b>	<b>Escenario</b>	<b>Nombre</b>	<b>Dirección</b>	<b>Respuesta del Sistema</b>	<b>Resultado de la prueba</b>
EC1	Cancelar crear plantilla.			El sistema cierra la ventana.	Cierra la ventana.
EC2	Crear plantilla.	Reporte1.jrx ml		El sistema crea la plantilla y la muestra.	El sistema crea la plantilla y la muestra.
EC3	Faltan datos obligatorios.	“ ”		El sistema no permite realizar ninguna operación hasta que el usuario haya introducido el dato obligatorio.	El sistema no permite realizar ninguna operación hasta que el usuario haya introducido el dato obligatorio y la plantilla no se crea.
EC4	Guardar plantilla.	NuevoReporte.jrx ml	D:\tesis\Nuevo Reporte.jrx ml	El sistema guarda en un archivo (.jrx ml) la estructura de la plantilla.	El sistema guarda la plantilla en formato (.jrx ml).
EC5	Cancelar guardar plantilla.			El sistema cierra la ventana.	El sistema cierra la ventana.
EC6	Visualizar reporte guardado.	ReportePoblación.jrx ml	D:\tesis\ReportePoblación.jrx ml	El sistema muestra una interfaz con la vista previa del reporte.	El sistema muestra una vista previa del reporte.

EC7	Visualizar reporte sin guardarlo.	ReportePoblación.jrxml	“ ”	El sistema indica guardar el reporte y luego muestra una interfaz con la vista previa del reporte.	El sistema indica guardar la plantilla para poder visualizarla y luego la visualiza.
-----	-----------------------------------	------------------------	-----	--	--

Tabla 14: Matriz de Casos de Prueba con los valores de los datos: Crear plantilla.

#### 4.5.1.2 Modificar plantilla.

<b>Id del escenario</b>	<b>Escenario</b>	<b>Nombre</b>	<b>Dirección</b>	<b>JRXML</b>	<b>Respuesta del sistema</b>	<b>Resultado de la prueba</b>
EC1	Cancelar abrir fichero.				El sistema cierra la ventana.	El sistema cierra la ventana.
EC2	Abrir fichero.	Reporte2.jrxml	D:\tesis\Reporte2.jrxml		El sistema carga la estructura de la plantilla del reporte con los datos contenidos en el archivo y los muestra.	El sistema carga el fichero correctamente.
EC3	Abrir fichero sin seleccionarlo.	“ ”	“ ”		El sistema no permite hacer nada hasta que el usuario no haya elegido la ruta del fichero a cargar.	El sistema no carga el fichero sin haber elegido la dirección.
EC4	Guardar fichero con el mismo nombre de alguno abierto en el sistema.	Reporte2.jrxml	D:\tesis\Reporte2.jrxml		El sistema muestra un mensaje de error.	El sistema muestra un mensaje de error.



EC5	Datos incorrectos.	Reporte2.jrx ml	D:\tesis\Report e2.jrxml	Estructura inválida.	El sistema emite un mensaje indicando que los datos son incorrectos.	El sistema indica con un mensaje que los datos están incorrectos.
EC6	Modificar estructura de la plantilla.	Reporte2.jrx ml			El sistema modifica la estructura de la plantilla y la muestra con los cambios efectuados.	El sistema modifica la estructura y muestra la plantilla con los nuevos cambios efectuados.

Tabla 15: Matriz de Casos de Prueba con los valores de los datos: Modificar plantilla.

#### 4.5.1.3 Gestionar elementos de la plantilla.

Id del escenario	Escenario	Campo	Dirección	Respuesta del sistema	Resultado de la prueba
EC1	Insertar campo.	$\$F\{id\_biblioteca\}$		El sistema muestra el campo insertado por el usuario con el valor indicado.	El sistema inserta y muestra el campo elegido por el usuario en el área indicada con el valor de la base de datos (de la consulta).

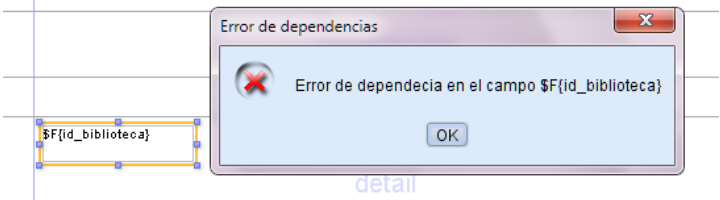
EC2	Eliminar valor del campo.	“ ”		El sistema muestra un mensaje de error y no deja generar el reporte hasta solucionar el problema.	El sistema muestra un mensaje de error.
					
EC3	Insertar imagen.		D:\imagenes\libro.jpg	El sistema inserta y muestra la imagen que ha sido indicada por el usuario.	El sistema inserta y muestra la imagen que ha sido indicada por el usuario.
EC4	Modificar elemento.			El sistema brinda una interfaz determinada según el elemento seleccionado y muestra el elemento con las nuevas propiedades que el usuario modificó.	El sistema brinda una interfaz determinada según el elemento seleccionado y muestra el elemento con las nuevas propiedades.
EC5	Eliminar elemento.			El sistema elimina el elemento de la plantilla.	El sistema elimina el elemento de la plantilla.

Tabla 16: Matriz de Casos de Prueba con los valores de los datos: Gestionar elementos de la plantilla.

## 4.6 Conclusiones

Como resultado de este capítulo se obtuvo el diagrama de despliegue de los componentes del sistema, se ilustraron algunas implementaciones importantes y se mostraron las pruebas de caja negra realizadas con algunos de los resultados obtenidos.

## CONCLUSIONES

Luego de todo lo expuesto en el presente trabajo, es posible concluir que:

- Se realizó el diseño e implementación de un módulo del Reporteador Dinámico para diseñar visualmente los reportes, reutilizando algunos componentes de la herramienta iReport.
- Se realizó la evaluación de las principales funcionalidades del sistema haciendo uso de las pruebas de Caja Negra.

## RECOMENDACIONES

- Continuar el desarrollo de la herramienta con el objetivo de incorporar nuevas funcionalidades que permitan el diseño de reportes de mayor complejidad.

## REFERENCIAS BIBLIOGRÁFICAS

1. Leiva Marrero, Ernesto. Implementación del módulo de diseño de reportes para el SCADA Guardián del ALBA; Ciudad de la Habana: s.n., 2009.
2. NCRReport. [En línea] 2010. [Citado el: 12 de enero de 2010.]  
<http://sourceforge.net/projects/ncreport/>
3. ReportManager. [En línea] 2010. [Citado el: 13 de enero de 2010.]  
<http://reportman.sourceforge.net/indexes.html>
4. Matrix. [En línea] 2010. [Citado el: 12 de enero de 2010.]<http://www.deinsa.com/matrix/principal.htm>
5. Pentaho. [En línea] 2010. [Citado el: 15 de enero de 2010.]  
[http://www.telefonica.net/web2/todobi/Oct07/Reporting\\_OS.pdf](http://www.telefonica.net/web2/todobi/Oct07/Reporting_OS.pdf)
6. Pentaho Report Designer. [En línea] 2010. [Citado el: 12 de enero de 2010.]  
<http://www.pentaho.com/products/reporting/>
7. iReport. [En línea] 2010. [Citado el: 12 de enero de 2010.] <http://jasperforge.org/projects/ireport>
8. Introducción a jasperreports e ireport (primera parte). [En línea] 2010. [Citado el: 14 de enero de 2010.]  
[http://www.mygnet.net/articulos/java/introduccion\\_a\\_jasperreports\\_e\\_ireport\\_primera\\_parte.301/Pagina/2](http://www.mygnet.net/articulos/java/introduccion_a_jasperreports_e_ireport_primera_parte.301/Pagina/2)
9. Marrero López, Yadira y Rosales García, Adonis R. Propuesta del diseño arquitectónico de la Plataforma bioGRATO, Universidad de las Ciencias Informáticas, 2007.

10. Per Kroll. OpenUp in a Nutshell. [En línea] 2010. [Citado el: 8 de marzo de 2010.]  
<http://www.ibm.com/developerworks/rational/library/sep07/kroll/>
11. Visual Paradigm. [En línea] 2010. [Citado el: 18 de enero de 2010.] <http://www.visual-paradigm.com/product/vpum/>
12. Lenguaje de programación. [En línea] 2010 [Citado el: 18 de enero de 2010].  
<http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php>
13. JFreeChart. [En línea] 2010 [Citado el: 9 de marzo de 2010]  
<http://www.jfree.org/jfreechart/>
14. Campderrich Falgueras, Benet. Ingeniería del software. Pág. 113. [En línea] 2010 [Citado el: 12 de marzo de 2010].  
[http://books.google.com/cu/books?id=tKTpr4Ah88C&pg=PA113&lpg=PA113&dq=que+es+modelo+de+dominio+%2B+ingenieria&source=bl&ots=RtlUn0FlwO&sig=73MccvBJVEmMtlp8L7f0Lig\\_FZY&hl=es&ei=rKCasS7XrGIL78AaWm4GiDg&sa=X&oi=book\\_result&ct=result&resnum=9&ved=0CCcQ6AEwCA#v=onepage&q=&f=false](http://books.google.com/cu/books?id=tKTpr4Ah88C&pg=PA113&lpg=PA113&dq=que+es+modelo+de+dominio+%2B+ingenieria&source=bl&ots=RtlUn0FlwO&sig=73MccvBJVEmMtlp8L7f0Lig_FZY&hl=es&ei=rKCasS7XrGIL78AaWm4GiDg&sa=X&oi=book_result&ct=result&resnum=9&ved=0CCcQ6AEwCA#v=onepage&q=&f=false)
15. Clase Teórico Práctica 1 de Ingeniería de Software I: “Profundización en la disciplina de Requisitos “. Curso 2009-2010.
16. Canal Velasco, Carlos. Tesis doctoral: Un Lenguaje para la Especificación y Validación de Arquitecturas de Software. Dic. 2000. Málaga. [En línea] 2010 [Citado el: 20 de abril de 2010].  
[http://www.lcc.uma.es/~canal/papers/tesis/canal\\_tesis.pdf](http://www.lcc.uma.es/~canal/papers/tesis/canal_tesis.pdf)
17. Conferencia 2 de Ingeniería de Software II: “ Arquitectura y Patrones de diseño”. Curso 2009-2010.
18. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. PRENTICE HALL, México, 1999.

## BIBLIOGRAFÍA

1. JasperReports vs JFreeReport. [En línea] 2010 [Citado el: 2 de mayo de 2010]. [http://www.jroller.com/chenihsu/entry/jasperreports\\_vs\\_jfreereport\\_p\\_a](http://www.jroller.com/chenihsu/entry/jasperreports_vs_jfreereport_p_a)
2. Jasper Reports. [En línea] 2010 [Citado el: 5 de mayo de 2010]. [http://el-directorio.org/Jasper\\_Reports](http://el-directorio.org/Jasper_Reports)
3. Manual for iReport. [En línea] 2010 [Citado el: 8 de mayo de 2010]. <http://ireport.sourceforge.net/manual0.2.0.html#1.0>
4. Getting Started. [En línea] 2010 [Citado el: 8 de mayo de 2010]. [http://jasperforge.org//website/ireportwebsite/IR%20Website/ir\\_getting\\_started.html?header=project&target=ireport](http://jasperforge.org//website/ireportwebsite/IR%20Website/ir_getting_started.html?header=project&target=ireport)
5. NCRReport. [En línea] 2010 [Citado el: 15 de abril de 2010]. <http://www.nocisoft.com/index.php/ncreport/articles/ncreport.html>
6. NCRReport. [En línea] 2010 [Citado el: 16 de abril de 2010]. <http://www.openden.com/opensource/singlelink/id/3985190/page/3/pid/30227129/>
7. Report Manager. [En línea] 2010 [Citado el: 25 de abril de 2010]. <http://reportman.sourceforge.net/indexes.html>
8. iReport. [En línea] 2010 [Citado el: 2 de mayo de 2010]. [http://www.amazon.com/Definitive-Guide-iReport-Experts-Voice/dp/1590599284#reader\\_1590599284](http://www.amazon.com/Definitive-Guide-iReport-Experts-Voice/dp/1590599284#reader_1590599284)
9. La Plataforma Pentaho Open Source Business Intelligence. [En línea] 2010 [Citado el: 27 de abril de 2010]. <http://pentaho.almacen-datos.com/>

## GLOSARIO DE TÉRMINOS

**CSV:** (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.

**HTML:** siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web.

**J2EE:** (Plataforma Java 2, Enterprise Edition o J2EE hasta la versión 1.4) es una plataforma de programación -parte de la Plataforma Java- para desarrollar y ejecutar software de aplicaciones en lenguaje de programación con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

**PDF:** Formato de Documento Portable o en inglés Portable Document Format es el formato de archivos desarrollado por Adobe Systems y creado con los programas Adobe Acrobat Reader, Acrobat Capture, Adobe Distiller, Adobe Exchange, y el plugin Amber de Adobe Acrobat.

**Reporte:** Informe (Sinónimo), documento caracterizado por contener información u otra materia reflejando el resultado de una investigación adaptado al contexto de una situación y de una audiencia dadas.

**RTF:** (formato de texto enriquecido) es un formato de archivo informático para el intercambio de documentos multiplataforma. La mayoría de procesadores de texto son capaces de leer y escribir documentos RTF.

**XML:** siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

**XLS:** Extensión para los archivos de hoja de cálculo de Microsoft Excel.