Universidad de las Ciencias Informáticas

Facultad 5

Título: Implementación de un módulo de generación de reportes para sistemas de supervisión, control y adquisición de datos.

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

Autor(es): Bernardo Zaragoza Hijuelos

Raudi Agdel Bacallao Sánchez

Tutor(es): MSc. Jaime Fardales Pérez.

Ing. Amado Espinosa Hidalgo.

Ciudad de la Habana, 2008

DECLARACIÓN DE AUTORÍA

Por este medio se declara que somos los únicos autores de este trabajo y se autoriza a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.

Para que así conste se firma la presente a los __ días del mes de junio del 2008.

Firma del Autor	Firma del Autor
(Bernardo Zaragoza Hijuelos)	(Raudi Andel Bacallao Sánchez)
Firma del Tutor	Firma del Tutor
(MSc. Jaime Fardales Pérez)	(Ing. Amado Espinosa Hidalgo)

Datos de Contacto

MSc. Jaime Fardales Pérez

Graduado de Ingeniero en automático en el año 1997 y Master en automática mención en Sistemas Computacionales en el año 2004. Especialista Superior de la Empresa de Automatización Integral (CEDAI), con nueve años de experiencia en la especialidad.

Ing. Amado Espinosa Hidalgo

Graduado de Ingeniero Informático en el 2004 y profesor instructor con cuatro años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y 4 en el desarrollo de software.

Agradecimientos

De Bernardo:

De Raudi:

RESUMEN

En el marco de los convenios PDVSA-ALBET existentes entre Cuba y la República Bolivariana de Venezuela y con el objetivo de minimizar la dependencia tecnológica, así como aumentar la soberanía de ambos países en materia informática y tecnológica, se desarrolla en la Universidad de las Ciencias Informáticas un proyecto para el desarrollo de un sistema de supervisión, control y adquisición de datos (*SCADA*) *por sus siglas en inglés*. Este sistema está formado por un conjunto de módulos, entre los cuales se encuentra el de Reportes, cuyo objetivo fundamental es permitir la generación de reportes de forma automática. En el presente documento se exponen los trabajos desarrollados para implementar un subsistema de generación de reportes utilizando herramientas libres, que permita generar los reportes requeridos interactuando con los demás subsistemas del SCADA.

Para ello se realiza un estudio de los principales algoritmos utilizados en los diferentes generadores de reportes para aplicarlos en la generación de reportes en el SCADA. Además se analizan las tendencias y tecnologías existentes, proponiéndose las más adecuadas para el desarrollo. Se presentan también, las pruebas realizadas al subsistema para validar el acogimiento del código a la satisfacción de las funcionalidades exigidas al mismo.

PALABRAS CLAVES

Sistema, Software, SCADA, Reporte, Diseño, XML, HTML, CSV.

Tabla de contenidos

Índice de Figuras

INTRODUCCIÓN

Históricamente, la necesidad de aumentar los niveles de eficiencia, minimizar los costos y optimizar los procesos de producción, ha resultado sumamente importante para cualquier industria. Actualmente, uno de los mecanismos más importantes para dar solución a estas necesidades, es la automatización de los distintos procesos que componen la cadena de producción, por tal motivo la necesidad de automatizar operaciones resulta prioritaria para cualquier industria donde se hayan invertido importantes recursos.

Los sistemas de supervisión, control y adquisición de datos, mas conocidos por su acrónimo en ingles: SCADA (Supervisory Control and Data Acquisition) constituyen el núcleo de la mayoría de estos sistema automatizados, por tal motivo son se erigen como sistemas altamente estratégicos para cualquier industria. Un sistema SCADA es una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador. Además, provee a diversos usuarios, toda la información que se genera en el proceso productivo, control de calidad, supervisión, mantenimiento. **(SCADA 2005)**

En la Universidad de la Ciencias Informáticas existen proyectos productivos cuyo objetivo fundamental es lograr la automatización de empresas e instituciones tanto del país como en el exterior, tal es el caso de la empresa venezolana PDVSA (Petróleos de Venezuela). Uno de los proyectos vinculados a tan importante empresa, es el que se lleva a cabo en la Facultad 5 “SCADA”, cuyo objetivo fundamental es desarrollar un sistema de supervisión, control y adquisición de datos (SCADA); sobre la plataforma GNU/Linux el cual debe ser instaurado en todas las infraestructuras productivas de PDVSA además de existir el interés ministerial por parte del MIC (Ministerio de Informática y Comunicaciones de Cuba) de replicar estos trabajos para las infraestructuras productivas de nuestra nación.

PDVSA en sus instalaciones cuenta con SCADAs para operar confiable y eficientemente sus procesos tecnológicos. Estos SCADAs han sido suministrados por compañías extranjeras, lo que ha provocado una total dependencia de las mismas y grandes dificultades para mantenerlos en operación. Algunas de estas

compañías se plegaron, y apoyaron con la utilización de estas tecnologías negándole el soporte en medio del “Paro petrolero” que afectó a PDVSA a fines del año 2002, que posibilitaron la operación remota de las instalaciones afectadas por personal no autorizado para ello. De esta manera resultó evidente lo vulnerable de estos sistemas, lo que unido a la dependencia tecnológica para el mantenimiento de empresas de servicios que se plegaron a otros intereses, provoca una compleja situación para mantener funcionado y expandiendo los sistemas SCADAs existentes en la actualidad y un riesgo potencial para la estabilidad y seguridad de los objetivos tecnológicos operados mediante los mismos.

Debido a la alta dependencia tecnológica inherente a los productos existentes en el mercado, La dirección tecnológica de PDVSA toma la decisión de desarrollar un SCADA para sus instalaciones y que pudiera utilizarse también en otras instalaciones industriales determinándose para dicho desarrollo el acogimiento a los preceptos de software libre.

Entre las funcionalidades básicas de los SCADAs está la de proporcionar toda la información que se genera en el proceso productivo a diversos usuarios relacionados directamente con el control y supervisión de dichos procesos así como a otros pertenecientes a niveles superiores dentro o fuera de la empresa control de calidad, supervisión, mantenimiento, etc. En las bases de datos históricas del SCADA se registran los valores instantáneos de las variables de proceso, los eventos y la configuración del sistema, entre otras. Para muchos usuarios del SCADA, interesados en comportamientos más globales del sistema como estadísticas del proceso, detección de fugas, deterioro de indicadores productivos, etc., esta basta información carece de valor, a menos que sea procesada y presentada de forma más adecuada a sus necesidades. En este punto surge la necesidad de emitir informes que consoliden la información adquirida para entregarla en un formato determinado, de tal manera que sea útil al usuario que va dirigida. Todo sistema SCADA cuenta con mecanismos para la generación de informes, los cuales son más o menos flexibles en dependencia de la complejidad des SCADA en cuestión o del empeño puesto para su desarrollo. Sin duda la generación de informes ha constituido un buen quebraderos de cabeza en muchas industrias, ya que se dedican largas horas ante la pantalla para realizar agrupaciones, saltos de página, y conseguir que todo salga correctamente impreso, de esta afirmación no escapa la realidad presente hoy en los sistemas instaurados en PDVSA. De aquí la

necesidad de que en el proyecto “SCADA” se desarrolle un módulo para la configuración y generación de informes.

De aquí la necesidad de que en el proyecto “SCADA” se desarrolle un módulo para la configuración y generación de reportes para solventar el problema de la inexistencia de una herramienta capaz de generar reportes, que sea lo suficientemente flexible como para procesar de forma automática los datos existentes en las bases de datos históricas pertenecientes al SCADA. Añadiéndose a esto las deficiencias en el proceso de elaboración y generación de reportes empleados actualmente en los SCADAs usados por PDVSA al involucrar varias herramientas propietarias, ya que no existía un diseñador que permitiera definir plantillas para los reportes a generar ni un generador que empleara dichas plantillas, siendo así muy difícil el proceso de configuración y parametrización de los reportes. Además los reportes obtenidos no podían ser accesibles por navegadores Web tendencia ampliamente difundida en la actualidad.

De todo lo expuesto anteriormente se presenta como **problema científico** la siguiente interrogante:

¿Cómo generar y presentar la información gestionada por un sistema SCADA utilizando herramientas y soluciones libres?

Teniendo como **objeto de estudio** los generadores de reportes y como **campo de acción** los generadores de reportes con herramientas libres aplicados específicamente al los SCADAs.

El **objetivo** del trabajo expuesto en este informe es implementar el módulo para la generación de reportes usado en el proyecto SCADA, utilizando herramientas libres y que interactuando con el resto de los módulos del SCADA permita configurar y emitir los reportes necesarios.

Como **objetivos específicos** se identificaron:

- Desarrollar la implementación de las funcionalidades diseñadas en el módulo para la generación de reportes del SCADA.
- Desarrollar pruebas de unidad al módulo de generación de reportes.

Para ello se propusieron las siguientes **tareas:**

- Revisar bibliografía en lo referente el estado del arte del tema, así como, las herramientas, estándares y métodos para implementar este tipo de aplicación para SCADAs.

- Revisar bibliografía en lo referente el estado del arte del tema, así como, las herramientas, estándares y métodos para implementar este tipo de aplicación para SCADAs.
- Investigar sobre las diferentes bibliotecas para el desarrollo de aplicaciones sobre plataformas libres.
- Desarrollar respetando la arquitectura propuesta para lograr una implementación robusta que permita la reutilización de sus componentes.
- Desarrollar una aplicación flexible que sea capaz de soportar el constante crecimiento.
- Realizar las pruebas de unidad que permitan validar el trabajo desarrollado
- Estudiar y seleccionar herramientas automatizadas para el desarrollo de pruebas unitarias.

Esperando obtener como posibles resultados los enunciados a continuación:

- Disponer de un módulo para la generación de reportes que cumpla con los requisitos pactados con el cliente.
- Con el diseño y desarrollo de las prueba se espera un módulo que cumpla con los objetivos para los cuales fue desarrollado, además de un código con la calidad requerida.

Para la presentación de los resultados obtenidos en este trabajo, el presente informe consta de 3 capítulos estructurados de la siguiente forma: en el Capítulo 1 se realiza un esbozo teórico centrado en temáticas como sistemas SCADAs, generadores de informes, etc., que sirven de apoyo a todo el trabajo desarrollado. En el Capítulo 2 se analizan la arquitectura propuesta y se explican las principales funcionalidades. En el Capítulo 3 se dedica a las pruebas de unidad realizadas para garantizar la integridad de la aplicación desarrollada.

CAPÍTULO 1

FUNDAMENTACIÓN DEL TEMA

En el presente capítulo se introducen las principales características y conceptos asociados a los generadores de reportes vinculados a los sistemas SCADAs. Para ello se abordan definiciones generales de los sistemas SCADAs. Además se exponen los sistemas de visualización, la descripción de las principales tecnologías que se emplean en la construcción de estos tipos de software y finalmente se explica la selección de las bibliotecas gráficas, así como las alternativas de desarrollo.

1.1 SCADA, una breve revisión

Los primeros SCADA eran simplemente sistemas de telemetría que proporcionaban reportes periódicos de las condiciones de campo vigilando las señales que representaban medidas y/o condiciones de estado en ubicaciones de campo remotas. Estos sistemas ofrecían capacidades muy simples de monitoreo y control, sin proveer funciones de aplicación alguna. La visión del operador en el proceso estaba basada en los contadores y las lámparas detrás de paneles llenos de indicadores. Mientras la tecnología se desarrollaba, los ordenadores asumieron el papel de manejar la recolección de datos, disponiendo comandos de control, y una nueva función - presentación de la información sobre una pantalla de CRT. Los ordenadores agregaron la capacidad de programar el sistema para realizar funciones de control más complejas.

Un sistema de control y supervisión generalmente se define “como una forma de control remoto con disponibilidad para el control selectivo de las unidades remotas”. Estos sistemas “permiten un gobierno total de la planta aunando las ventajas de seguridad de autómatas industriales y el control y gestión mediante computadoras que permiten interfaces gráficas muy amigables e intuitivas para el operario”. Otra posible definición de un SCADA es la denominación que en general recibe el conjunto formado por una “*aplicación de software especialmente diseñada para funcionar sobre computadoras de control de producción, con acceso a la planta mediante comunicación digital con los reguladores locales básicos, e*

interfaces gráficas de alto nivel con usuario mediante pantallas táctiles, ratones o cursores, lápices ópticos, etc., los programas necesarios, y en su caso el hardware adicional que necesiten" . Otra definición un poco más abarcadora es la siguiente: "un sistema basado en aplicaciones de software diseñada *para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo (Controladores Lógicos Programables- PLC, Unidad Terminal Remota- RTU, etc.) y controlando el proceso de forma automática desde la pantalla del computador. Permitiendo realizar a distancia operaciones de control, supervisión y registro de datos de cualquier proceso industrial. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. De igual forma, ya que cuenta con información (alarmas, históricos, paradas, etc.) de primera mano de lo que ocurre u ocurrió en el proceso, permite la integración con otras herramientas del negocio como lo son intranets, ERP, etc".*

Un SCADA por otra parte, generalmente cubre áreas geográficas más grandes, y normalmente depende de una variedad de sistemas de comunicación menos confiables que una LAN. Entonces, qué es SCADA? Se utiliza para vigilar y para controlar la planta industrial o el equipamiento. El control puede ser automático, o iniciado por comandos de operador. La adquisición de datos es lograda en primer lugar por los RTU's que exploran las entradas de información de campo conectadas con ellos (pueden también ser usados PLC's - Programmable Logic Controllers). Esto se hace generalmente a intervalos muy cortos. La MTU entonces explorará los RTU's generalmente con una frecuencia menor. Los datos se procesarán para detectar condiciones de alarma, y si una alarma estuviera presente, sería catalogada y visualizada en listas especiales de alarmas.

Los datos pueden ser de tres tipos principales:

- Datos analógicos (por ejemplo números reales) que quizás sean presentados en gráficos.
- Datos digitales (on/off) que pueden tener alarmas asociadas a un estado o al otro.
- Datos de pulsos (por ejemplo conteo de revoluciones de un medidor) que serán normalmente contabilizados o acumulados.

La interfaz primaria al operador es una pantalla que muestra una representación de la planta o del equipamiento en forma gráfica. Los datos vivos (dispositivos) se muestran como dibujos o esquemas en

primer plano (foreground) sobre un fondo estático (background). Mientras los datos cambian en el campo, el foreground es actualizado (una válvula se puede mostrar como abierta o cerrada, etc.). Los datos analógicos se pueden mostrar como números, o gráficamente (esquema de un tanque con su nivel de líquido almacenado). El sistema puede tener muchas pantallas, y el operador puede seleccionar las más relevantes en cualquier momento.

1.1.1 Módulos del SCADA.

Los módulos o bloques software que permiten las actividades de adquisición, supervisión y control son los siguientes:

- **Seguridad**: Provee las funcionalidades necesarias para garantizar el trabajo autorizado por usuarios, además brinda las herramientas para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos tales como fallas de corriente, problemas de red o servidores entre otros.

- **Manejadores**: El módulo de manejadores de dispositivos asegura la comunicación del SCADA con los dispositivos de campo. Esta comunicación de realiza a través de una interfaz genérica única para todos los protocolos y bibliotecas dinámicas (Manejadores o drivers) que implementan esa interfaz genérica en cada caso específico.

- **Configuración**: Este módulo se encarga de almacenar, persistir y suministrar, la Información base para el funcionamiento de los demás módulos del SCADA(BDH, BDTR, HMI entre otros), cada módulo del sistema posee un agente (biblioteca), que permite establecer las comunicaciones con el servidor de configuración, a través de la capa de conectividad, permitiendo crear eliminar y modificar los recursos configurables del SCADA.

- **Visualización (HMI)**: Este módulo es el encargado de permitir representar gráficamente de manera fidedigna los procesos operacionales con los que los usuarios interactúan. A través de este módulo se pueden visualizar condiciones operacionales, valores de variables, visualizar alarmas, navegar entre despliegues, enviar comandos entre otros.

- **Base de datos Histórica (BDH)**: La BDH contiene toda la información persistente de alarmas, eventos, bitácoras y datos adquiridos permitiendo acceder a todo lo que ha ocurrido en el sistema durante un período determinado, este tiempo es configurable de acuerdo a las necesidades de los usuarios.

- **Capa de conectividad(Middleware)**: Este módulo es la capa de software, que se encarga de la comunicación entre los diferentes procesos distribuidos de medio y alto nivel, que forman parte del SCADA , y el principal elemento que caracteriza su complejidad es la gestión de las comunicaciones.

- **Núcleo de procesamiento de datos (Recolector)**: El núcleo de procesamiento de datos es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.

- **Generación de reportes**: El módulo de generación de reportes en un SCADA es la herramienta que permite emitir reportes en un formato personalizado de tal manera que le sea útil al personal al cual va dirigido

1.2 Software Libre

Con el progresivo avance de la tecnología y para su posterior explotación se obtiene la necesidad de crear programas de computadoras (Software) que estén cada vez más adaptados al desarrollo actual. Muchos de estos programas creados son pertenecientes a empresas como software propietarios, donde para poder adquirir estos productos y sus licencias hay que pagar grandes sumas de dinero. Esto ha permitido que otros desarrolladores de software se interesen en distribuir de forma libre y gratuita productos de gran importancia encargados de lograr un mejoramiento de estos para su posterior utilización y un mejor funcionamiento, de aquí que comienza el surgimiento del Software Libre.

Software libre: Es un software, cuyo secreto de fabricación (el llamado "código fuente", algo así como los "planos" del programa) es conocido y difundido libremente. El autor del software concede a cualquiera el derecho a usar su obra, a modificarla y a adaptarla a sus necesidades específicas. Este derecho de libre uso a veces se otorga sin restricciones (licencias tipo BSD) y otras veces con la única condición de que

toda mejora se distribuya con las mismas condiciones de uso que tiene originalmente y por tanto siga estando libremente disponible para todo el mundo (licencias tipo copyleft, como la GPL de la Free Software Foundation). A diferencia de lo que sucede con el software propietario cuyo único objetivo es la rentabilidad económica y no el hacer buenas herramientas, la comunidad del software libre se empeña en la búsqueda de una buena adecuación entre las necesidades y el propio producto, esto es, busca a la vez la calidad y la eficiencia social que otorga la libertad de uso. El software libre, incluso en los casos en que se comercializa, siempre es más barato que sus equivalentes propietarios, y casi siempre acaba estando disponible gratuitamente (ALBERTO MOLPECERES TOURIS 18/08/2002).

El sistema de software libre puede ser la solución para muchos problemas en el mundo empresarial, para las organizaciones gubernamentales y, por supuesto, para el usuario particular.

El desarrollo de un sistema SCADA sobre plataforma de software libre contribuye a la obtención de un software de calidad el cual brinda la libertad de poder copiarlo, distribuirlo y modificarlo y así poder obtener un software superior y de un mejor funcionamiento.

1.3 Generalidades sobre los generadores de reportes.

De forma general un generador de reportes se compone de dos elementos básicos, un diseñador de reportes y un motor de generación como se muestra en la figura.

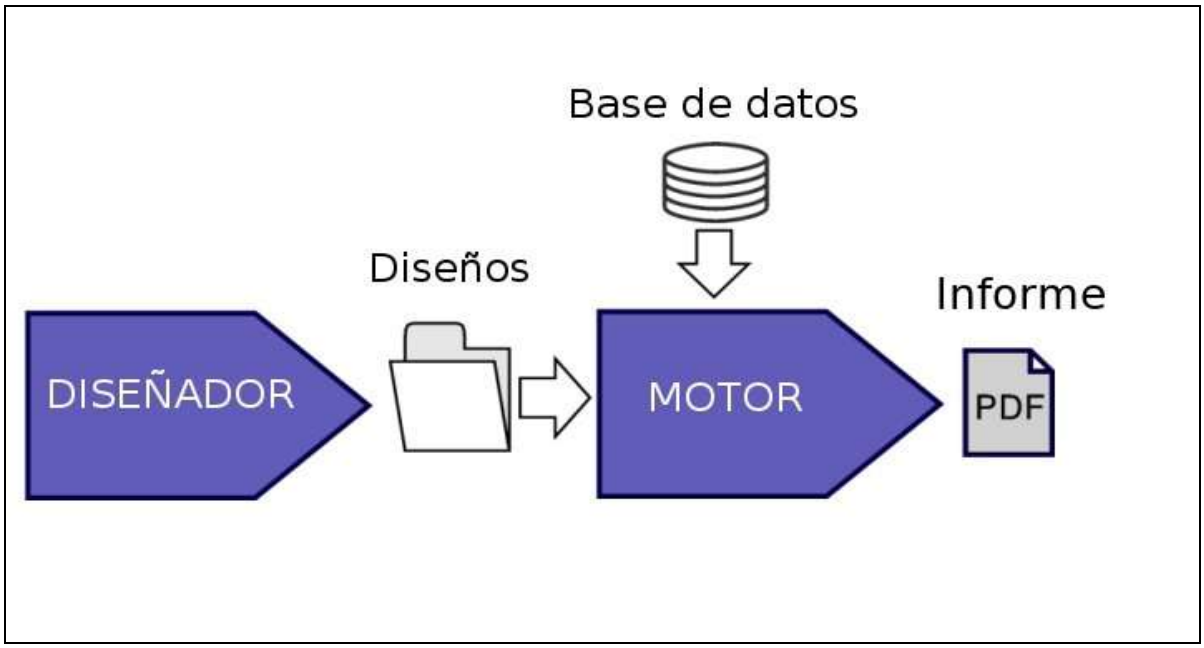

Fig. 1: Estructura general de un generador de reportes. Esquema de funcionamiento.

De aquí se puede deducir que las principales tareas que debe realizar un generador de reportes son las siguientes:

- Diseñar la apariencia que van a tener los reportes.
- Extraer la información y volcarla de forma ordenada con la apariencia diseñada.

Haciendo uso del diseñador se obtienen plantillas de diseños que posteriormente son empleadas por el motor de generación para volcar datos provenientes de almacenes de datos y obtener el reporte final con la apariencia preestablecida en el diseño.

Existen algunos proyectos que solo desarrollan el motor de generación. En este caso también se considera la aplicación como un generador de reportes, pero se debe tener bien presente la necesidad de brindar el diseño al motor en algún formato entendible por este, lo cual, de no contarse con un diseñador, puede resultar un proceso muy engorroso. De aquí que la presencia de un diseñador que brinde flexibilidades para diseñar los reportes que posteriormente se generará sea muy deseable en una aplicación de este tipo.

1.4 Generación de reportes en los SCADAs.

Es cada vez más común la tendencia a complementar las funcionalidades de adquisición, registro de datos y la generación de alarmas con la capacidad de generar información capaz de ayudar en la toma de decisiones.

- Por ejemplo, será interesante disponer de información referente a:
- Situación de la planta (estado, incidencias).
- Producción en tiempo real.
- Generación de alarmas.
- Adquisición de datos para el análisis histórico, control de calidad, calculo de costes mantenimiento preventivo.
- Gestión de almacén, producción y mantenimiento.

Mediante las herramientas SQL es posible realizar extractos de los archivos, los registros de las bases de datos del sistema, realizar operaciones de clasificación o valoración sin afectar los datos originales. También permiten presentar los archivos en forma de reportes o transferirlos a otras aplicaciones mediante las herramientas de intercambio disponibles, por ejemplo, se pueden transferir datos de una tabla de una base de datos a una hoja de cálculo, etc.

1.4.1 Situación actual.

En la actualidad existe una gran cantidad de proyectos enfrascados en el desarrollo de generadores de reportes. Aunque ninguno de los encontrados se enfoca directamente en la generación de reportes para un SCADA ya que no cuentan con el manejo de la seguridad necesaria para manejar la integridad de la información obtenida, ni cuentan con mecanismos para la generación automática programada previamente por un operador entro otras , no es menos cierto que este es uno de los procesos inherentes al SCADA que más se aproxima a aplicaciones informáticas estándares, pues es básicamente un sistema de gestión, aunque con algunas peculiaridades propias de esta aplicación. Concretamente, sirve de puente entre la BDH del SCADA y los diseños de reportes que el usuario previamente configura de acuerdo a sus necesidades para volcar, filtrar y analizar datos. De aquí, que un estudio de los resultados alcanzados en estos proyectos resulta de mucha utilidad para desarrollar el generador de reportes necesario para el proyecto SCADA, incluso teniendo presente la posible adopción de algunos resultados de estos proyectos como soluciones para el que ocupa.

A pesar de la abundante existencia de proyectos sobre software libre para el desarrollo de generadores de reportes, no deben descartarse las ideas que pudieran tomarse de algún software propietario bien asentado en el mercado.

Todo esto lleva a la necesidad de implementar una solución que resuelva las funcionalidades requeridas para un SCADA:

1. Realizar un análisis de las tecnologías y bibliotecas existentes para el manejo de reportes así como los principales algoritmos utilizados en el manejo de grandes flujos de datos.

Este punto conllevaría a crear las bases para la programación de un módulo de reportes que resulte en una herramienta capaz de crear un sistema de configuración y visualización de reportes sencillo, con múltiples escenarios de visualización, potente y fiable en aras de facilitar toda la información necesaria para el desarrollo del SCADA utilizando herramientas libres.

1.5 TENDENCIAS Y TECNOLOGIAS

1.5.1 Tecnologías libres más utilizadas

En el mundo del software libre se emplean varias tecnologías que permiten a los desarrolladores utilizarlas dependiendo de las necesidades y de las características de cada sistema que se construya. Los sistemas libres están adentrándose con creces en el ambiente de las interfaces gráficas de usuario, dejando atrás el mito del sistema operativo en modo texto, para ello emplean diferentes paquetes de desarrollo y además variadas tecnologías para la representación de los elementos gráficos, buscando elevar la eficiencia y calidad en el renderizado de las imágenes.

1.5.1.1 Biblioteca Qt

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Fue creada por la compañía noruega Trolltech. Qt es utilizada fundamentalmente en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Utiliza el lenguaje de programación C++ de forma nativa y además existen *bindings* para C, Python (PyQt), Java (Qt Jambi), Perl (PerlQt) y Ruby (QtRuby) entre otros.

El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

1.5.1.2 Biblioteca GTK+

GTK+ son las siglas de GIMP Toolkit, es una biblioteca que permite crear interfaces gráficas de usuario, se distribuye bajo la licencia pública general (GPL), lo que hace de GTK+ un producto completamente libre.

GTK esta construido encima de GDK (GIMP Drawing Kit) que básicamente es un recubrimiento de las funciones de bajo nivel que debe haber para acceder al sistema de ventanas sobre el que se programe. Se llama el GIMP toolkit porque fue escrito para el desarrollo del General Image Manipulation Program (GIMP), pero ahora GTK se utiliza en un gran número de proyectos de programación, incluyendo el proyecto GNU Network Object Model Environment (GNOME). GTK está construido encima de GDK (GIMP Drawing Kit) que básicamente es un recubrimiento de las funciones de bajo nivel que deben haber para acceder al sistema de ventanas sobre el que se programe (Xlib en el caso de X Microsoft Windows).

GTK+ depende de otras bibliotecas que hacen de GTK+ un éxito total:

- Glib, una biblioteca de propósito general, no destinada a interfaces gráficas en sí, provee tipos de datos, macros y utilidades de conversión, tratamiento de cadenas y abstracciones muy útiles.
- Pango, se encarga de la manipulación de los textos internacionalizados, provee wigdets que se encargan de la representación de los textos.
- Atk, es el paquete de accesibilidad, provee un conjunto de interfaces que permiten a las interfaces de usuario interactuar con las tecnologías de accesibilidad.
- Gdk, es la capa de abstracción que permite a GTK+ ser portable a múltiples plataformas.

1.5.1.3 Biblioteca Cairo

Cairo es una biblioteca de gráficos 2D completamente libre, con soporte para múltiples dispositivos de salida, entre ellos X Window System, Win32, PostScript, PDF, archivos SVG y otros en estado de prueba como OpenGl, Quartz y XCB.

Cairo está diseñado para producir salidas consistentes en todos los dispositivos mientras emplea al máximo la aceleración del hardware de video si está disponible. La API *por su acrónimo en ingles Application Programming Interface* de Cairo ofrece operaciones similares a las de dibujado empleadas en las tecnologías PostScript y PDF. Las operaciones en Cairo incluyen pinceladas, rellenos, curvas Bézier, variadas transformaciones y representación de textos, todas las operaciones de dibujo se pueden lograr mediante las transformaciones básicas (rotar, escalar, mover).

Cairo está escrita en el lenguaje C, pero posee implementaciones para Lisp, Java, Mono, Perl, PHP, Pyton, Ruby y otros.

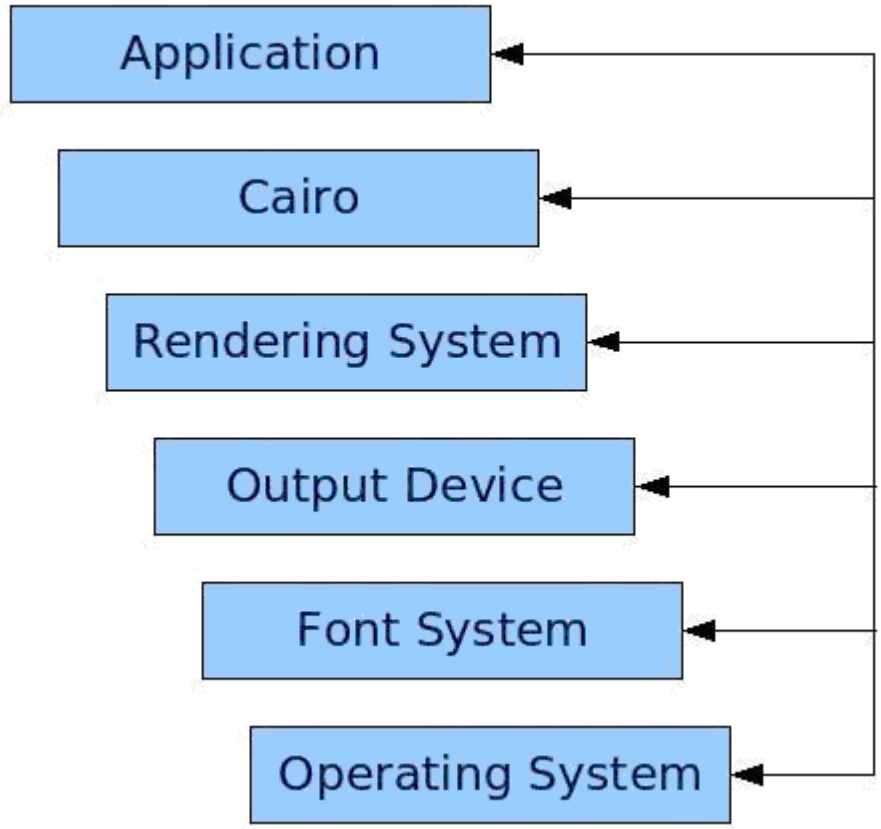

Fig. 2: Arquitectura de Cairo

1.5.2 Biblioteca Boost.

La librería boost cuenta con una alta calidad ya que es desarrollada por algunos miembros del comité de estandarización del C++, es totalmente gratuita lo que permite un desarrollo libre en cualquier proyecto que se utilice. Incluye código fuente y abundante documentación. Permite el desarrollo de las siguientes funcionalidades:

- Cadenas y procesamiento de texto
- Contenedores
- Iteradotes
- Algoritmos
- Programación genérica

- Meta programación con plantillas
- Meta programación de procesos
- Programación concurrente
- Análisis matemático y numérico
- Corrección y prueba
- Estructura de datos
- Entrada/salida
- Soporte para metalenguaje
- Serialización

Esta biblioteca es una especie de antesala del estándar C++, ya que algunos de sus componentes son analizados y evaluados por los miembros del comité como candidatos a ser incluidos en futuras revisiones del estándar C++.

1.5.3 C++

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes como ROOT (enlace externo).

Las principales características del C++ son el soporte para programación orientada a objetos, el soporte de plantillas o programación genérica (*templates*), la portabilidad, brevedad, programación modular, compatibilidad con C y velocidad. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además, se trata de un lenguaje de programación estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar C++ que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel entre las cuales se destacan la posibilidad de redefinir los operadores (sobrecarga de operadores) y la identificación de tipos en tiempo de ejecución (*RTTI*).

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

1.5.4 Estudio de IDEs

Los entornos integrados de desarrollo (IDE, del inglés *Integrated Development Environment)*) actualmente están considerados como un conjunto de herramientas indispensables para un desarrollador de software, y a la vez, pueden ser utilizados para uno o varios lenguajes de programación, de aquí podemos decir que un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Además, los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Hemos visto que los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Visual Basic, Object Pascal, etc. Otra característica importante para un IDE es que puede funcionar como un sistema en tiempo de ejecución en algunos lenguajes de programación.

En la actualidad existe una gran variedad de IDEs. Entre los más usados por los desarrolladores de GNU/Linux tenemos:

KDevelop: realizado para sistemas GNU/Linux y otros sistemas Unix, publicado bajo licencia GPL y se destaca por poseer un entorno de desarrollo muy avanzado y por soportar muchos lenguajes y características avanzadas. A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio para producir código binario, por lo que depende de una serie de compiladores que se consideran estándar para los sistemas operativos derivados de UNIX, que también poseen licencia GPL. Su última versión se encuentra actualmente bajo desarrollo y funciona con distintos lenguajes de programación como C, C++, Java, Ada, SQL, Python, Perl y Pascal. Además de estas características también permite el control de versiones, mantiene el completado automático de código en C y C++, tiene asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación, posee un navegador entre clases de aplicación, tiene un editor de código fuente con

destacado de sintaxis e identado automático y está integrado con Qt para realizar interfaces gráficas para aplicaciones multiplataforma.

Eclipse: es uno de los entornos integrados de desarrollo (IDE) de código abierto y extensible mas utilizados en los últimos tiempos. Este incluye varias características únicas, como la refactorización de código, la actualización/instalación automática de código (mediante Update Manager), una lista de tareas, soporte para unidades de test con JUnit, e integración con la herramienta de construcción de Jakarta: Ant.

A pesar del gran número de características estándar, Eclipse se diferencia de los IDEs tradicionales en varias cosas fundamentales. Quizás lo más interesante de Eclipse es ser completamente neutral a la plataforma - y al lenguaje.

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad. Este mecanismo de módulos es una plataforma ligera para componentes de software. Por otra parte permite a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Phyton y trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como la gestión de la configuración. Se provee soporte para Java y el sistema de control de versiones en el SDK de Eclipse.

Glade: es otra de las herramientas (IDE) de desarrollo visual de aplicaciones, encargada fundamentalmente de manejar lenguajes de programación y su principal función esta dada por el desarrollo de entornos gráficos. Esta herramienta le permite al usuario crear entornos ajustados a las necesidades, utilizando para ello, lenguajes de programación específicos como C, C++ y Ada 95, así como también, hacer modificaciones desde el código fuente. Además es capaz de generar código para implementar la interfaz visual, que se describe en XML, en C, C++, Ada 95, Perl y Eiffel. El código que genera se compila e instala con: configure, make, make install, entre otros. Entre sus paletas de componentes destacan las GTK y GNOME encargadas de construir interfaces graficas de usuario.

Aunque tradicionalmente se ha utilizado de forma independiente, Glade se encuentra incorporado en el programa Anjuta, el cual es un entorno integrado de desarrollo para programar. En su interior contiene una librería nombrada (Libglade) que mejora el uso de los archivos XML y permite una mayor flexibilidad a la hora del trabajo y trabaja sobre la licencia GPL.

1.5.5 Alternativas de desarrollo.

En el mundo existen dos tendencias bien identificadas en el desarrollo de herramientas de este tipo:

1. Integrar la herramienta a aplicaciones específicas (fundamentalmente ofimáticas –Word Report, Generador de Reportes de Access-).
2. Desarrollar aplicaciones independientes totalmente independientes de los entornos de explotación (Cristal Reports, Agata, etc.)

Se pueden valorar tres posibles alternativas de desarrollo para la herramienta descrita:

- **Desarrollo Autónomo**: Desarrollar íntegramente un generador de reportes a la medida haciendo uso de los componentes básicos que pongan a disposición las bibliotecas que se decidan para el proyecto.
- **Desarrollo usando generador de reportes externo**: Usar algún generador de reportes existente que permita explotar sus funcionalidades desde una aplicación externa, la cual se tendría que desarrollar y que permitiera ajustar sus características a las necesidades del SCADA.
- **Desarrollo usando aplicaciones ofimáticas**: Usar alguna aplicación ofimática como el Cal o el Writer para diseñar y generar los reportes y dejar a la aplicación que se desarrolle las tareas de consultar las bases de datos y procesar los datos que entregará a la aplicación ofimática.

1.5.5.1 Alternativa Autónoma.

Esta consiste en desarrollar todo el sistema, partiendo solo de los componentes básicos que a disposición pongan las bibliotecas que se decida emplear para el proyecto. En este caso se dispondría de todo el código desarrollado y documentado por el mismo equipo de desarrollo, lo cual le daría homogeneidad a la solución total. No obstante, parece la variante más trabajosa dada la necesidad de realizar un diseño e implementación de muchas componentes o módulos que en las demás alternativas ya estrían desarrolladas y por ende no implicarían esfuerzos de desarrollo e implementación, aunque sí de investigación de los mecanismos de comunicación que como interfaces externas ponen a disposición de otras aplicaciones. De la anterior idea pudiera inferirse que para desarrollar esta variante no hay que investigar sobre temas muy similares a los planteados en las otras dos alternativas y que posteriormente

se analizarán, cosa que realmente no es así e incluso pudiera ser mayor el esfuerzo en este punto, pues siempre será necesario estudiar las clases que existen en las bibliotecas disponibles para no tener que "inventar la rueda".

1.5.6 Propuesta.

Luego del análisis llevado a cabo, se puede plantear una propuesta que consiste en desarrollar la aplicación del generador de informes sobre el software SCADA, que siga el modelo de los generadores de informes existentes y que incluya además otras funcionalidades, utilizará como lenguaje de programación a C++ dada su portabilidad y eficiencia, como IDE para desarrollo se utilizara Eclipse por ser de código abierto, extensible y porque incorpora funcionalidades tan interesantes como la refactorización de código, para el desarrollo de las interfaces de usuario se utilizara el IDE Glade.

Será desarrollado sobre software libre por las ventajas que trae esto consigo de reutilización de código, independencia tecnológica entre otras.

1.6 Conclusiones.

En este capítulo se realizó un esbozo de las principales características de los sistemas SCADA, los módulos que lo componen además se hace un análisis de las tecnologías a utilizar en el desarrollo de la propuesta de solución, así como algunos conceptos y tendencias que esta debe adoptar para sí. A partir de estos puntos se comenzará el desarrollo de la propuesta de sistema.

CAPÍTULO 2

Construcción de la solución propuesta.

En el presente capítulo se diseña la propuesta de solución, primeramente se incluye un epígrafe donde se explica la arquitectura donde se retoman algunos puntos de arquitectura e implementación para dar una mejor visión de la solución construida. Posteriormente se hace una descripción de las principales funcionalidades que se implementaron en la construcción del software a si como las principales clases del diseño que se utilizaron en la implementación de la interfaz hombre máquina .También se muestran algunos fragmentos de códigos de la implementación de cada funcionalidad expuesta.

1.7 Funcionamiento.

El funcionamiento del sistema a desarrollar puede dividirse en dos:

1. Por un lado tendremos un diseñador de informes, que proporciona al usuario una forma sencilla de diseñar una plantilla o diseño de informe. En esta plantilla el usuario tendrá la posibilidad de introducir imágenes, etiquetas de texto, componentes gráficos, y cajas de texto para indicar el origen de los datos procedentes de distintas bases de datos.
2. Por otro lado tendremos un motor de generación, que se encargará de extraer los datos de una o varias bases de datos, después enlazará estos datos con el diseño realizado antes, y finalmente generará un documento en base al diseño de informe obtenido con el diseñador. El documento resultante es un archivo que incluye todo lo citado anteriormente (diseño más datos), el cual podremos visualizar en pantalla o despachar a distintos medos como la impresora o el disco duro en forma de archivo con un determinado formato para ser accedido mediante los servicios que se establezcan en el SCADA.

Por demás al formar parte esta aplicación de un SCADA, debe brindar la posibilidad de comunicación entre el motor de generación y el SCADA para que este último pueda invocar la generación de informes a partir de diseños previamente desarrollados.

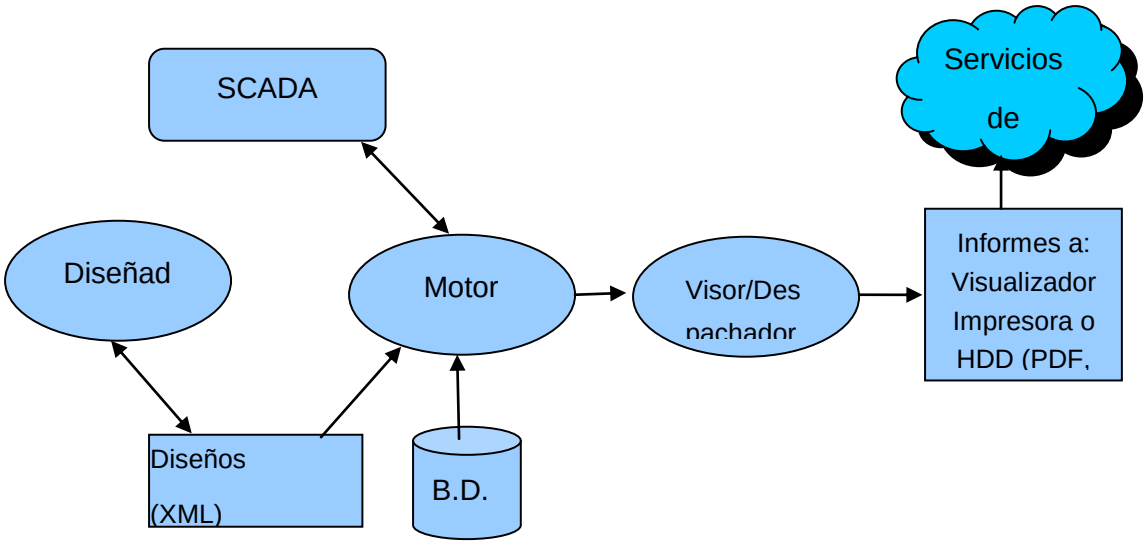

Fig. 3 Esquema funcional del sistema propuesto.

Este funcionamiento queda representado de forma gráfica en la figura que se muestra a continuación:

Como se puede apreciar en este esquema los diseños persistirán en un formato estándar (XML) para ser tomados lo mismo por el motor de generación, para generar el informe ante solicitudes provenientes del SCADA, que por el diseñador para ser usados como diseños preelaborados.

1.8 Patrones.

"En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. (…) Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería de software. Todo lo contrario: intentan codificar el conocimiento, las expresiones y los principios ya existentes: cuanto más trillados y generalizados, tanto mejor." (LARMAN 2004)

Patrones arquitecturales: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas software.

Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.

Dada la similitud existente entre el subsistema que se necesita y una aplicación de gestión se propone una arquitectura en Vista Modelo Controlador, la cual da una gran flexibilidad a la aplicación dado el bajo acoplamiento funcional que se logra con la misma, permitiendo a la aplicación adaptarse de forma relativamente fácil a cambios (adiciones, modificaciones o eliminaciones).

Es el diseño de la aplicación se aplicaron todos los patrones de Grasp son Patrones Generales de Software para Asignación de Responsabilidades considerados como "Buenas Prácticas" en el diseño de software.

1.9 Arquitectura Vista Model Controlador (MVC).

La arquitectura que se ha propuesto persigue que el módulo Reportes pueda dar respuesta a los requisitos de portabilidad, comprensibilidad, extensibilidad y eficiencia.

Se propone una solución orientada al patrón arquitectónico Vista-Modelo-Controlador, el cual es ampliamente utilizado por soluciones computacionales afines al área objeto. Es de destacar que el patrón MVC no se aplica de forma tradicional donde son identificables los paquetes correspondientes al Modelo, la Vista o el Control. Esto se deriva de los escenarios donde se requiere sea desplegada la aplicación. Esta situación contrasta con el escenario de una aplicación de escritorio donde la filosofía de las bibliotecas gráficas buscan que cada componente sea la unidad responsable de la captura de los eventos que genera el usuario (equivalentes al Control) sino que también tienen la responsabilidad de representar la información a través de su área de dibujo. En este contexto es común enlazar el Modelo a través de callbacks con el componente y a partir de estos, generar las actualizaciones de la vista. La asimetría de las implementaciones hace necesario establecer un conjunto de abstracciones que permitan generalizar ambos enfoques para poder dar solución a los requisitos de portabilidad de una forma lo más natural posible. El primer paso en ésa dirección corresponde a la identificación y selección de conceptos que no intersecan las funcionalidades de Vista y Control. Estos conceptos serán agrupados en lo que se denominará el subsistema. Por otra parte las funcionalidades que semánticamente involucren lógica de Vista o de Control serán implementadas particularmente para cada escenario.

El subsistema de reportes del SCADA se divide en dos aplicaciones fundamentales:

2. Ambiente de Ejecución (AE).
3. Ambiente de Edición (AER).

El AE es la aplicación encargada de visualizar los reportes para monitorear y supervisar los procesos y el AER es la aplicación que posibilita la edición de las plantillas de reportes del SCADA.

Es importante destacar que existe una relación de extensión entre el AE y el AER, es decir, hay un número de funcionalidades importante que comparten, como las relacionadas con el concepto motor de generación, y los objetos gráficos, entre otros. Debido ha esta relación se diseñó una estrategia de desarrollo que permitiera la reutilización de los elementos comunes para evitar re-trabajo.

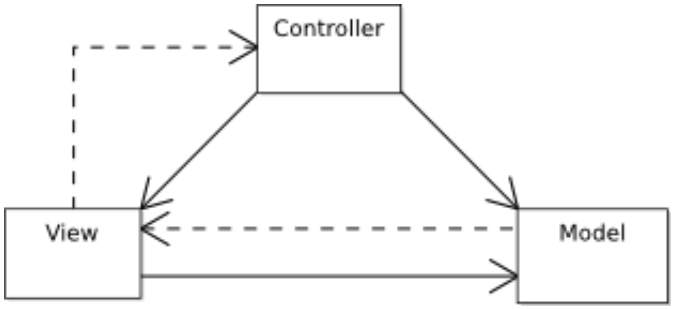

Fig. 4 Arquitectura MVC

1.10 Principales funcionalidades. Diagramas de clases y diagramas secuencia para los principales escenarios de los casos de uso.

A continuación se presentan la explicación de la principales funcionalidades del generador de reportes para ello se muestran los diagramas de clases y de secuencia de dichas funcionalidades y se da una breve explicación de las clases mas importantes que y como apoyo se colocaron fragmentos de código utilizados en las mismas

1.10.1 Desplegar Reporte (Previsualizar).

Permite al usuario la visualización de un reporte determinado el usuario escoger la opción de generar reporte. El Generador de informes haciendo uso de la clase *"Report"* hace una llamada al método *"generate"* con sus respectivos parámetros, dándole la orden la instancia de la clase *"Engine"* que contiene de generar un informe mediante el método *"renderToPreview"* el cual es el encargado de generar solamente la pagina que solicito el usuario para optimizar la memoria.

Para ello las principales clases involucradas se muestran en el siguiente Diagrama de Clases.

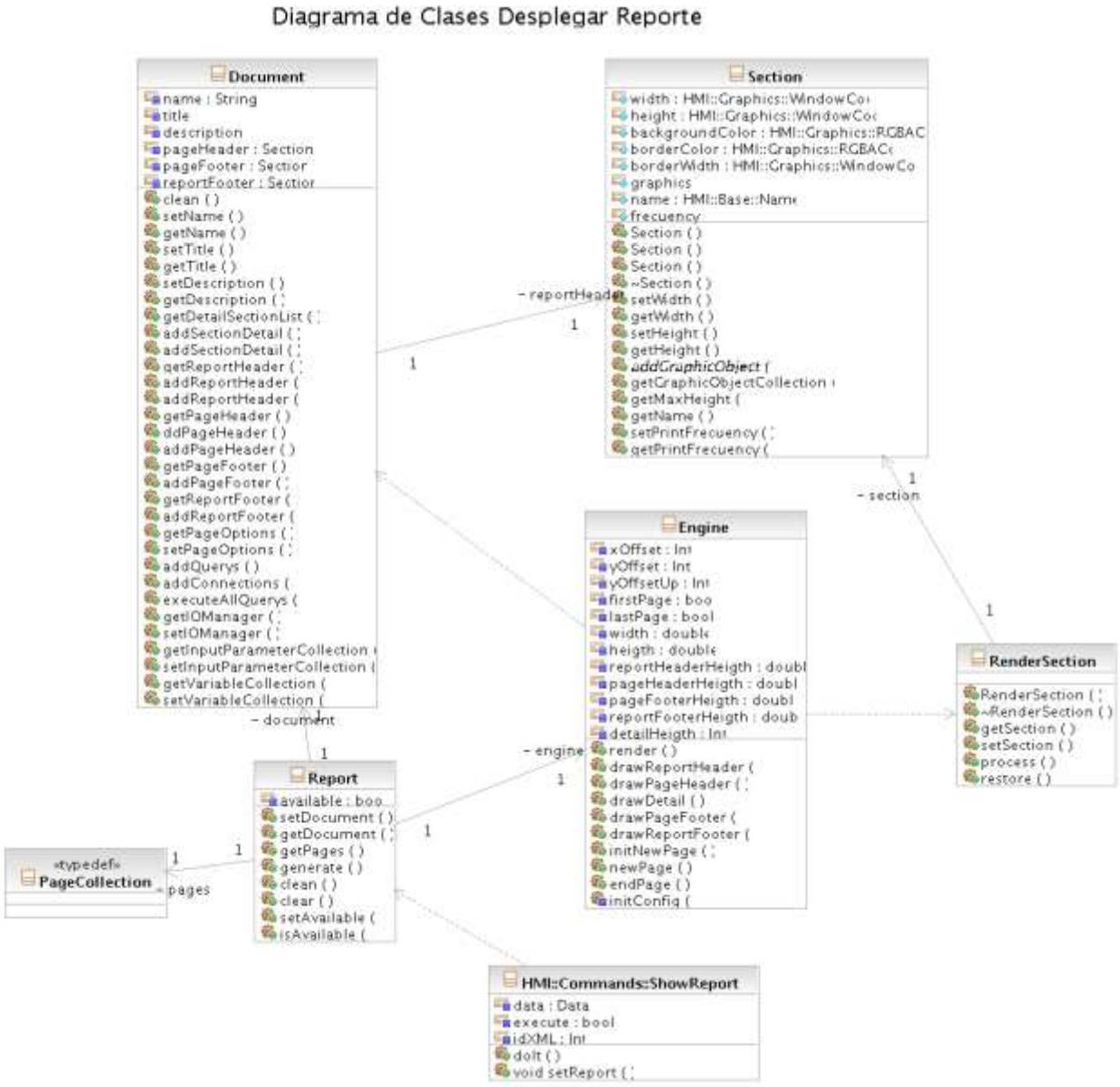

Fig. 5 Diagrama de Clases Desplegar Reporte

Para una mejor comprensión de que es lo que hace esta funcionalidad se muestra a continuación un diagrama de secuencia que muestra la interacción entre dichas clases para llevar a cabo la realización de la funcionalidad de Desplegar Reporte.

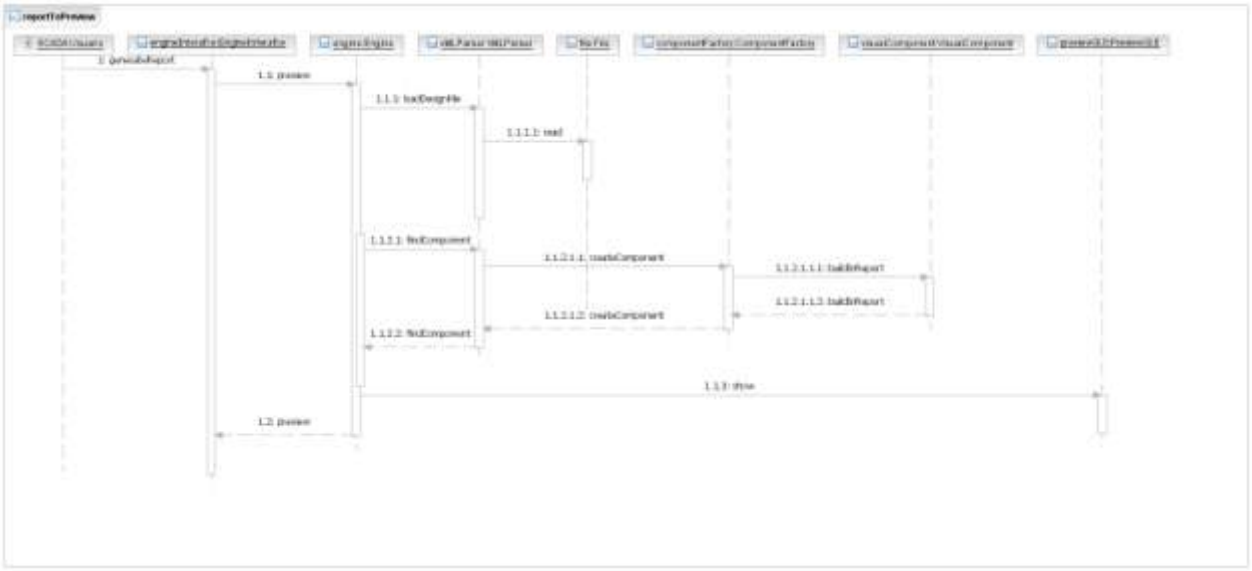

Fig. 6 Diagrama de secuencia que representa la previsualización de un reporte

1.10.1.1 Descripción de la Principales Clases

Nombre: Document	
Descripción: Es la clase principal que contiene todos los objetos gráficos, gestiona la asignación de ellos y el manejo de los datos de entrada y salida a través del IOManager.	
Tipo de clase: Controladora	
Atributo	**Tipo**
Name	String
Title	String
description	String
details	DetailCollection
reportHeader	Section*
pageHeader	Section*
pageFooter	Section*
reportFooter	Section*
pageOptions	Graphics::PageOptions
queryManager	IO::IOManager
inputParameterCollection	IO::InputParameterCollection

variableCollection	Graphics::VariableReportCollection
Para cada responsabilidad:	
Nombre:	void clean()
Descripción:	Limpia todos los datos de entrada y salida
Nombre:	Asigna un nuevo nombre al documento
Descripción:	void setName(String reportName)
Nombre:	String getName()
Descripción:	Devuelve el nombre del documento
Nombre:	void setTitle(String reportTitle)
Descripción:	Asigna nuevo titulo al documento
Nombre:	String getTitle()
Descripción:	Devuelve el titulo del documento
Nombre:	void setDescription(String reportDescription)
Descripción:	Asigna nueva descripción al documento
Nombre:	String getDescription()
Descripción:	Devuelve la descripción del documento
Nombre:	DetailCollection& getDetailSectionList()
Descripción:	Devuelve la lista de sección de detalles del documento
Nombre:	void addSectionDetail()
Descripción:	Adiciona una nueva sección de detalles no inicializada
Nombre:	void addSectionDetail(Graphics::SectionDetail* sectionDetail)
Descripción:	Adiciona una nueva sección de detalles pre-inicializada
Nombre:	Section* getReportHeader()
Descripción:	Devuelve el encabezado de reporte del documento
Nombre:	void addReportHeader(Section * pReportHeader)
Descripción:	Asigna nuevo encabezado de reporte al documento
Nombre:	void addReportHeader()
Descripción:	Adiciona un nuevo encabezado de reporte no inicializado
Nombre:	Section* getPageHeader()
Descripción:	Devuelve el pie de página del documento
Nombre:	void addPageHeader(Section * pPageHeader)
Descripción:	Asigna nuevo encabezado de página al documento
Nombre:	void addPageHeader()
Descripción:	Adiciona un nuevo encabezado de página no inicializado
Nombre:	Section* getPageFooter()
Descripción:	Devuelve el pie encabezado página del documento
Nombre:	void addPageFooter(Section * pPageFooter)
Descripción:	Asigna nuevo pie de página al documento
Nombre:	void addPageFooter()
Descripción:	Adiciona un nuevo pie de página no inicializado
Nombre:	Section* getReportFooter()
Descripción:	Devuelve el pie de reporte del documento
Nombre:	void addReportFooter(Section * pReportFooter)
Descripción:	Asigna nuevo pie de reporte al documento
Nombre:	void addReportFooter()
Descripción:	Adiciona un nuevo pie de reporte no inicializado
Nombre:	Graphics::PageOptions& getPageOptions()

Descripción:	Devuelve la configuración o opciones de una página
Nombre:	void setPageOptions(Graphics::PageOptions& options)
Descripción:	Asigna una nueva configuración o opciones de página
Nombre:	void addQuerys(IO::Query* query)
Descripción:	Adiciona una nueva consulta para manejo de entrada de datos
Nombre:	void addConnections (IO::Connection* connection)
Descripción:	Adiciona una nueva conexión para manejo de entrada de datos
Nombre:	void executeAllQuerys()
Descripción:	Ejecuta todas las consultas y abre cada conexión
Nombre:	IO::IOManager& getIOManager()
Descripción:	Devuelve el controlador de datos de entrada-salida
Nombre:	void setIOManager(IO::IOManager manager)
Descripción:	Asigna nuevo controlador de datos de entrada-salida.
Nombre:	IO::InputParameterCollection& getInputParameterCollection()
Descripción:	Devuelve la colección de parámetros de entrada
Nombre:	void setInputParameterCollection(inputParameterCollection)
Descripción:	Asigna una nueva colección de parámetros de entrada
Nombre:	Graphics::VariableReportCollection& getVariableCollection()
Descripción:	Devuelve la colección de variables.
Nombre:	void setVariableCollection(variableCollection)
Descripción:	Asigna una nueva colección de variables.

Nombre: Section	
Descripción: Clase que representa a un sección dentro de un reporte.	
Tipo de clase: Entidad	
Atributo	**Tipo**
width	WindowCoord
height	WindowCoord
backgroundColor	RGBAColor
borderColor	RGBAColor
borderWidth	WindowCoord
graphics	GraphicCollection
name	Base::Name
frecuency	Utils::PrintFrequency
Para cada responsabilidad:	
Nombre:	void setWidth(WindowCoord)
Descripción:	Asigna nuevo ancho de la sección
Nombre:	WindowCoord getWidth()
Descripción:	Devuelve el ancho de la sección
Nombre:	void setHeight(WindowCoord)
Descripción:	Asigna nuevo largo de la sección

Nombre:	WindowCoord getHeight()
Descripción:	Devuelve el largo de la sección
Nombre:	virtual void addGraphicObject(Graphics::Graphic* object)
Descripción:	Adiciona un nuevo objeto grafico a la sección
Nombre:	GraphicCollection& getGraphicObjectCollection()
Descripción:	Devuelve la colección de gráficos de la sección
Nombre:	double getMaxHeight()
Descripción:	Devuelve el máximo ancho que puede ocupar la sección
Nombre:	Base::Name getName()
Descripción:	Devuelve el nombre de la sección
Nombre:	void setPrintFrecuency(Utils::PrintFrequency frec)
Descripción:	Asigna nueva frecuencia de impresión
Nombre:	Utils::PrintFrequency getPrintFrecuency()
Descripción:	Devuelve la frecuencia de impresión

Nombre: Report	
Descripción: Es la clase que se encarga de controlar y que contiene al documento.	
Tipo de clase: Controladora	
Atributo	**Tipo**
document	Document
engine	Engine
pages	PageCollection
available	bool
Para cada responsabilidad:	
Nombre:	void setDocument(Document& doc)
Descripción:	Asigna un nuevo documento
Nombre:	Document& getDocument()
Descripción:	Devuelve el atributo document
Nombre:	PageCollection& getPages()
Descripción:	Devuelve la colección de páginas
Nombre:	void generate()
Descripción:	Genera los datos del atributo document en la colección de páginas
Nombre:	void clean()

Descripción:	Limpia los datos de entrada-salida del documento
Nombre:	void clear()
Descripción:	Borra todas las páginas en la colección de página
Nombre:	void setAvailable(bool available)
Descripción:	Asigna nuevo valor al atributo available
Nombre:	bool isAvailable()
Descripción:	Devuelve el atributo available

Nombre: Engine		
Descripción: Es la clase principal que se encarga de renderear sus datos de un documento para que sean visualizados en el visualizador del generador de reportes.		
Tipo de clase: Controladora		
Atributo		**Tipo**
firstPage		bool
lastPage		bool
width		double
heigth		double
reportHeaderHeigth		double
pageHeaderHeigth		double
pageFooterHeigth		double
reportFooterHeigth		double
detailHeigth		int
Para cada responsabilidad:		
Nombre:	void initConfig(Document& document)	
Descripción:	Asigna valores a todos los atributos a partir de los datos del document.	
Nombre:	void render(PageCollection& pages, Document& document)	
Descripción:	Renderea document en una colección de pages para ser visualizado.	
Nombre:	void drawReportHeader(Section*, RenderSection&)	
Descripción:	Pinta un objeto de tipo Section en la sección ReportHeader de una página.	
Nombre:	void drawPageHeader(Section*, RenderSection&)	
Descripción:	Pinta un objeto de tipo Section en la sección PageHeader de una página.	
Nombre:	void drawDetail(SectionDetail*, Document&, RenderDetailSection&)	
Descripción:	Pinta un objeto de tipo SectionDetail en la sección Detail de una página,	
Nombre:	void drawPageFooter(Section*, RenderSection&)	
Descripción:	Pinta un objeto de tipo Section en la sección PageFooter de una página	
Nombre:	void drawReportFooter(Section*, RenderSection&)	
Descripción:	Pinta un objeto de tipo Section en la sección ReportFooter de una página	
Nombre:	void initNewPage(PageCollection&, Document&, RenderSection&)	
Descripción:	crea la primera página en la colección de páginas para ser pintada	
Nombre:	void newPage(PageCollection&, Document&, RenderSection&)	
Descripción:	crea la una nueva página en la colección de páginas para ser pintada	
Nombre:	void endPage(Document&, RenderSection&)	
Descripción:	crea la ultima página en la colección de páginas para ser pintada	

Nombre: Page

Descripción: Clase que define y visualiza el reporte, contiene el contexto (Context).	
Tipo de clase: Controladora	
Atributo	**Tipo**
context	Context
width	double
height	double
Para cada responsabilidad:	
Nombre:	void setContext(double w, double h)
Descripción:	Se le asigna un valor de ancho y un largo a la página, (width, height).
Nombre:	Context getContext()
Descripción:	Retorna el contexto.

1.10.1.2 Fragmento de Código

El siguiente fragmento de código muestra el método doit del comando *ShowReport* se da la orden de generar un reporte pasándole previamente el identificador de la plantilla que esta almacenada en el *ReportTemplateCollection*:

```
void ShowReport::doIt( Viewer& viewer )
 {


     if( this->report->getPages().size() != 0 )
     {
         this->report->clean();
         this->report->clear();
     }
     Report::Base::Document doc = viewer.getTemplateReportCollection().at( this->idXML );
     this->report->setDocument( doc );
     Report::Base::ViewParameters* viewParameter = new Report::GTK::ViewParameters();
     viewParameter->setReport(this->report);
     viewParameter->show();
   }


 try
  {
    this->report->generate( SCADA::Report::Utils::renderToPreview, 1 );
    viewer.getReportView().setReport( this->report );
    viewer.getReportView().show();
```

```
    }
   catch (SCADA::Report::Exceptions::Exception* e)
    {
      HMI::System::getInstance()->getMessageWindow().setMessageType( SCADA::HMI::Base::MESSAGE_ERROR );
      HMI::System::getInstance()->getMessageWindow().setButtons( SCADA::HMI::Base::BUTTONS_OK );
      HMI::System::getInstance()->getMessageWindow().setPrimaryText( e->message() );
      HMI::System::getInstance()->getMessageWindow().show();
      HMI::System::getInstance()->getMessageWindow().hide();
    }
 }
```

Luego pasa jugar el papel principal la clase *Engine* que es el motor encargado de la generación del reporte final, haciendo uso de las clases *RenderSection* y *RenderDetailSection* para renderear las secciones del reporte. Este proceso de generación se hace a través del método:

```
void Engine::renderToPreview (int& cantPag, PageCollection& pages, Document& document, int pageToShow )
```

Donde:

cantPag: Almacena la cantidad de páginas generadas.

pages: Es donde se almacenan todas la páginas generadas.

document: Es quien contiene la información que será generada.

PageToShow: Es el numero de la página a generarse.

Este método es el encargado de una vez recibida toda la información del *Document*, de procesarla y generarla, dando como resultado una colección de páginas que luego son visualizadas dando el resultado final del proceso de despliegue de un reporte.

```
void Engine::renderToPreview (int& cantPag, PageCollection& pages, Document& document, int pageToShow )
  {
     this->renderMethod = Utils::renderToPreview;
     this->pagToShow = pageToShow;
     this->preview = true;
```

```
    this->currPage = 0;

      try
      {
        this->initConfig(document);
        if ( this->firstTime == true )
          {
            document.executeAllQuerys();
           }

        RenderSection renderSection;
        RenderDetailSection renderDetailSection;
        RecordSet* rs = 0;
        if(document.getDetailSectionList().front()->getMasterQuery() != 0)
          {
            rs = document.getDetailSectionList().front()->getMasterQuery()->getResult();
            rs->moveFirst();
           }

         if ( this->firstTime == true )
           {
             if( rs->end() == false )
               {
                 if(document.getDetailSectionList().front()->getGroup()!= NULL)
                   {
                     while( rs->end() == false )
                       {
                         if(renderDetailSection.isDetailFinished() == true )
                            {
                              rs->setfirstInGroup( rs->current() );
                            }
                          this->initNewPage( pages, document, renderSection );
                          this->drawDetail(document.getDetailSectionList().front(),
                                document, renderDetailSection );
                        }
                   }
                 else if(document.getDetailSectionList().front()->getGroup() == NULL)
                   {
                     while( rs->end() == false )
                       {
                         this->initNewPage( pages, document, renderSection );
                          this->drawDetail(document.getDetailSectionList().front(),
                                document, renderDetailSection );
                        }
                    }
                 }

                 double val = heigth - this->detailHeigth - this->pageFooterHeigth;
                 this->totalPages += val < this->reportFooterHeigth;
                 cantPag = this->totalPages;
               }

          catch ( Exceptions::Exception* e )
        {
           throw;
        }
    }
```

La función principal de este procedimiento es procesar los datos del Documentos asociado a un Reporte, para ello se utiliza al *RecordSet* para almacenar toda la información de las ejecución de las consultas

maestras, una vez almacenados los datos se procede a llenar la colección de páginas pasada por referencia (*PageCollection& pages),* utilizando en tal proceso a RenderSection y RenderSectionDetail para ir generando cada página, luego se visualiza la página deseada con valor (*pageToShow*).

1.10.2 Imprimir informe.

Después de tener generado un reporte y almacenado en memoria, el generador de reportes haciendo uso de la clase controladora *"PrintReportManager* hace una llamada al método *"print"* pasándole como parámetro la raíz del documento generado dándole la orden al *"Engine"* de procesar dicho reporte y enviarlo hacia un dispositivo, en este caso hacia una impresora. Después de haber creado todo el informe se llama a la interfaz de impresión que se invoca mediante un método de una instancia de *"GtkPrintOperation"* y haciendo uso del método *"begin_print"* comienza el rendereo del reporte haciendo llamadas al método *"draw_page ".*

Las principales clases involucradas son *PrintReportManage*r y *PrintReport* sus relaciones se muestran en el siguiente Diagrama de Clases.

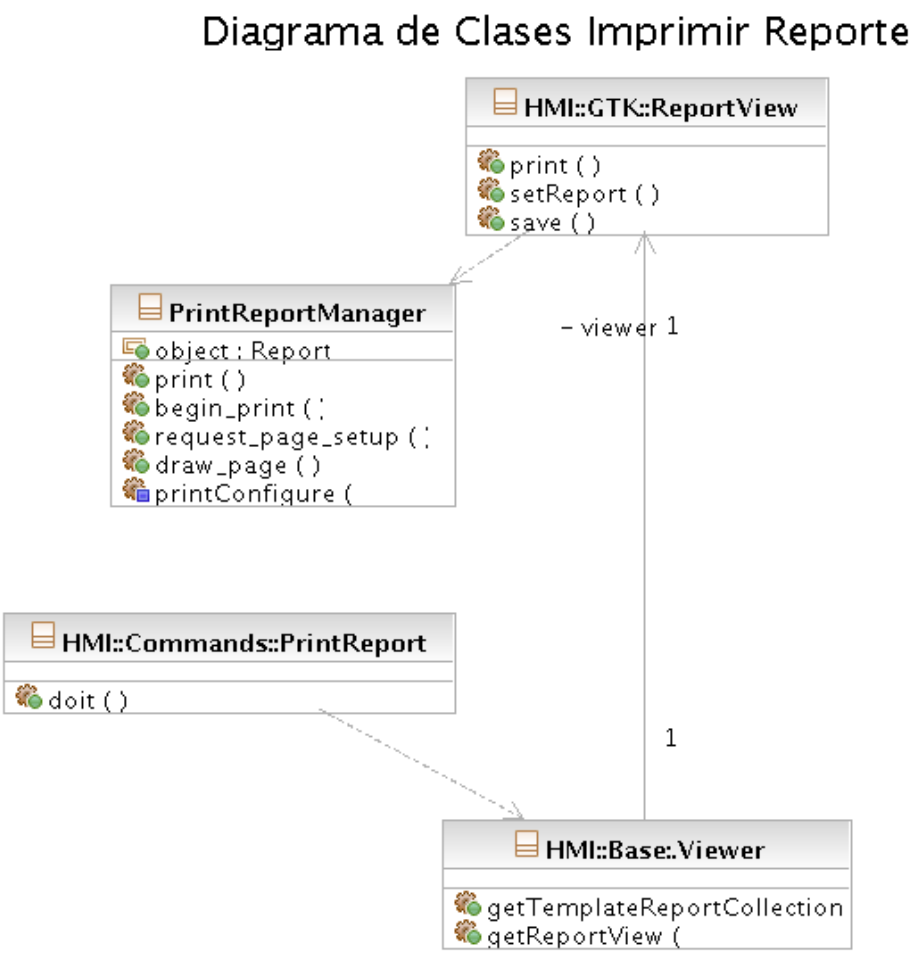

Fig. 7 Diagrama de Clases

Para una mejor comprensión de que es lo que hace esta funcionalidad se muestra a continuación un diagrama de secuencia que muestra la interacción entre dichas clases para llevar a cabo la realización de la funcionalidad de Imprimir Reporte

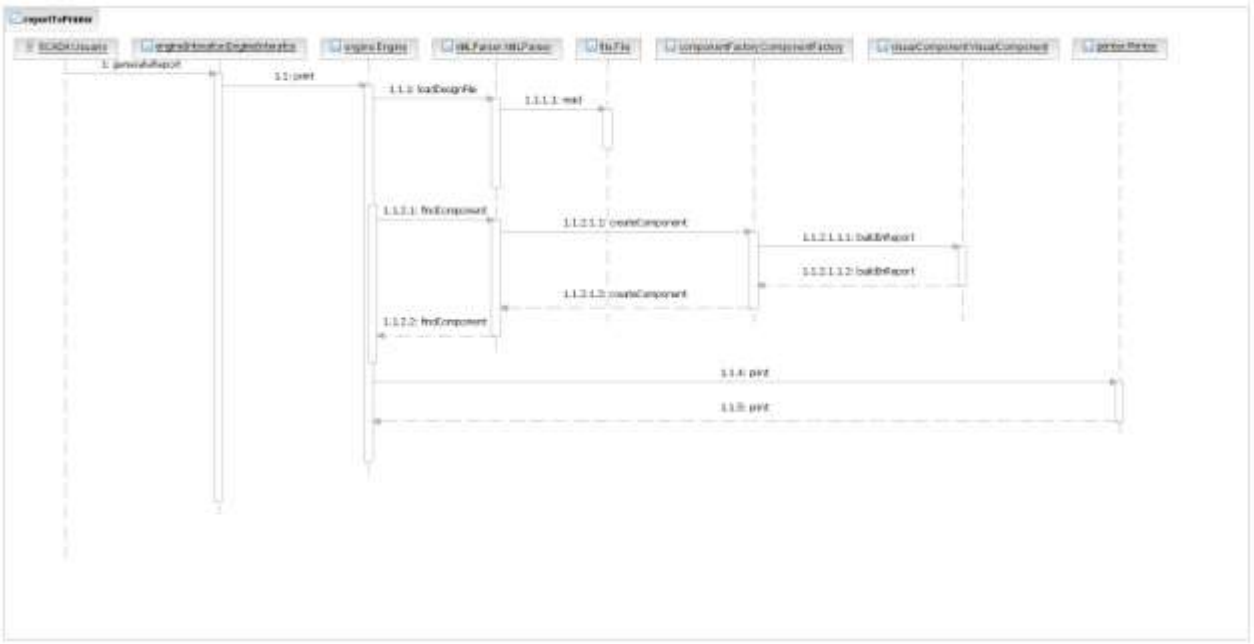

Fig. 8 Diagrama de secuencia que representa la impresión de un informe.

1.10.2.1 Descripción de las Principales Clases

Nombre: PrintReportManager	
Descripción: Clase encargada de la impresión de los reportes.	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	void print(Base::Report* newReportObject)
Descripción:	Método encargado de darle impresión al reporte.
Nombre:	static void begin_print (GtkPrintOperation *,GtkPrintContext *)
Descripción:	Método de Inicio de impresión.
Nombre:	static void request_page_setup (GtkPrintOperation *, GtkPrintContext *, int,..)
Descripción:	Método encargado de abrir y configurar la impresión
Nombre:	static void draw_page (GtkPrintOperation *,GtkPrintContext *,int)
Descripción:	Le da el comportamiento de pintura al reporte al imprimirse.

1.10.2.2 Fragmento de Código

EL *ReportView* se inicia al momento de generar cualquier reporte, en ese momento se crea un *PrintReportManager* que se encarga de imprimir el reporte que este visualizado, cualquier petición de impresión se hace a través del método:

```
void PrintReportManager::print( Base::Report* newDrawableObject )
```

Donde:

newDrawableObject: Es el objeto Reporte que se imprimirá en impresora.

Este método es el encargado de una vez recibido el reporte de preparar la impresora para imprimir la lista de página ya previamente generadas y visualizadas del reporte pasado como parámetro (En el ejemplo siguiente *newDrawableObject*).

```
void PrintReportManager::print( Base::Report* newDrawableObject )
  {
    PrintReportManager::object = newDrawableObject;
    GtkPrintSettings *settings = NULL;
    GtkPrintOperation *print;
    GtkPrintOperationResult res;

    print = gtk_print_operation_new ();
    g_signal_connect (print, "begin_print",
                      G_CALLBACK (PrintReportManager::begin_print),
                      NULL);
    g_signal_connect (print, "draw_page",
                      G_CALLBACK (PrintReportManager::draw_page),
                      NULL);
    g_signal_connect (print, "request_page_setup",
                      G_CALLBACK (PrintReportManager::request_page_setup),
                      NULL);
    if (settings != NULL)
      gtk_print_operation_set_print_settings (print, settings);

    res = gtk_print_operation_run (print,
                                   GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG,
                                   NULL,
                                   NULL);
    if (res == GTK_PRINT_OPERATION_RESULT_APPLY)
      {
        if (settings != NULL)
          g_object_unref (settings);
        settings = gtk_print_operation_get_print_settings (print);
      }
      g_object_unref (print);
}
```

Una vez que se ha configurado la impresora se inicia el proceso de pintura de cada página del reporte en modo impresión, el método usado en esta transformación de página es:

```
void PrintReportManager::draw_page (GtkPrintOperation *operation,GtkPrintContext * context,int page_nr)
```

Donde:

operation: Es el operación a realizar sobre la impresora.

context: Es el contexto sobre el cual se van a pintar la páginas.

page_nr: Es el numero de la página que se desea a imprimir.

Método mediante el cual se pueden ir imprimiendo cada página de la colección de páginas del reporte que este visualizado. Este método es invocado por un objeto de tipo *PrintReport*.

```
void PrintReportManager::draw_page (GtkPrintOperation *operation,GtkPrintContext *
context,int page_nr)
 {
    cairo_t *cr;
    cairo_surface_t *surface;
    gdouble print_width, print_height, scale_x , scale_y;
    gint surface_width, surface_height;
    cr = gtk_print_context_get_cairo_context (context);
    cairo_set_antialias(cr, CAIRO_ANTIALIAS_NONE);
    for(unsigned    int    i    =    1;    i    <=    PrintReportManager::object-
>getReportToSave().size(); i++ )
      {
        cairo_save( cr );
           surface = cairo_get_target(PrintReportManager::object-
>getReportToSave().at( i - 1 )->getContext());
         cairo_surface_reference(surface);
         print_width = gtk_print_context_get_width(context);
         print_height = gtk_print_context_get_height(context);
         surface_width = cairo_image_surface_get_width(surface);
         surface_height = cairo_image_surface_get_height(surface);
         scale_x = (gdouble) print_width / surface_width;
                scale_y = (gdouble) print_height / surface_height;
         cairo_scale(cr, scale_x, scale_y);
         cairo_set_source_surface(cr, surface, 0., 0.);
         cairo_paint( cr );
       cairo_restore( cr );
       cairo_show_page( cr );
      }
```

```
    }
```

En el ejemplo que se muestra se observa que se utilizan las funciones que brinda la librería *CAIRO* por su prefijo ***cairo_,*** siendo el método más sencillo, robusto y fácil de entender, además del atributo estático del reporte *PrintReportManager::object* quien contiene la lista de páginas generadas que están siendo visualizadas, haciendo uso del mismo el procedimiento anteriormente explicado.

1.10.3 Almacenar informe en formato HTML.

Permite al usuario poder exportar un reporte previamente generado en formato HTML para su posterior revisión o transportación del mismo. El Generador después de haber previsualizado el reporte haciendo uso de la clase *"WebExport"* hace una llamada al método *"saveHTML"* con sus respectivos parámetros, dándole la orden al *"Engine"* de generar un informe a formato html que finalmente será almacenado en disco. Para una mejor comprensión se muestra la relación de las principales clases en el siguiente diagrama de clases

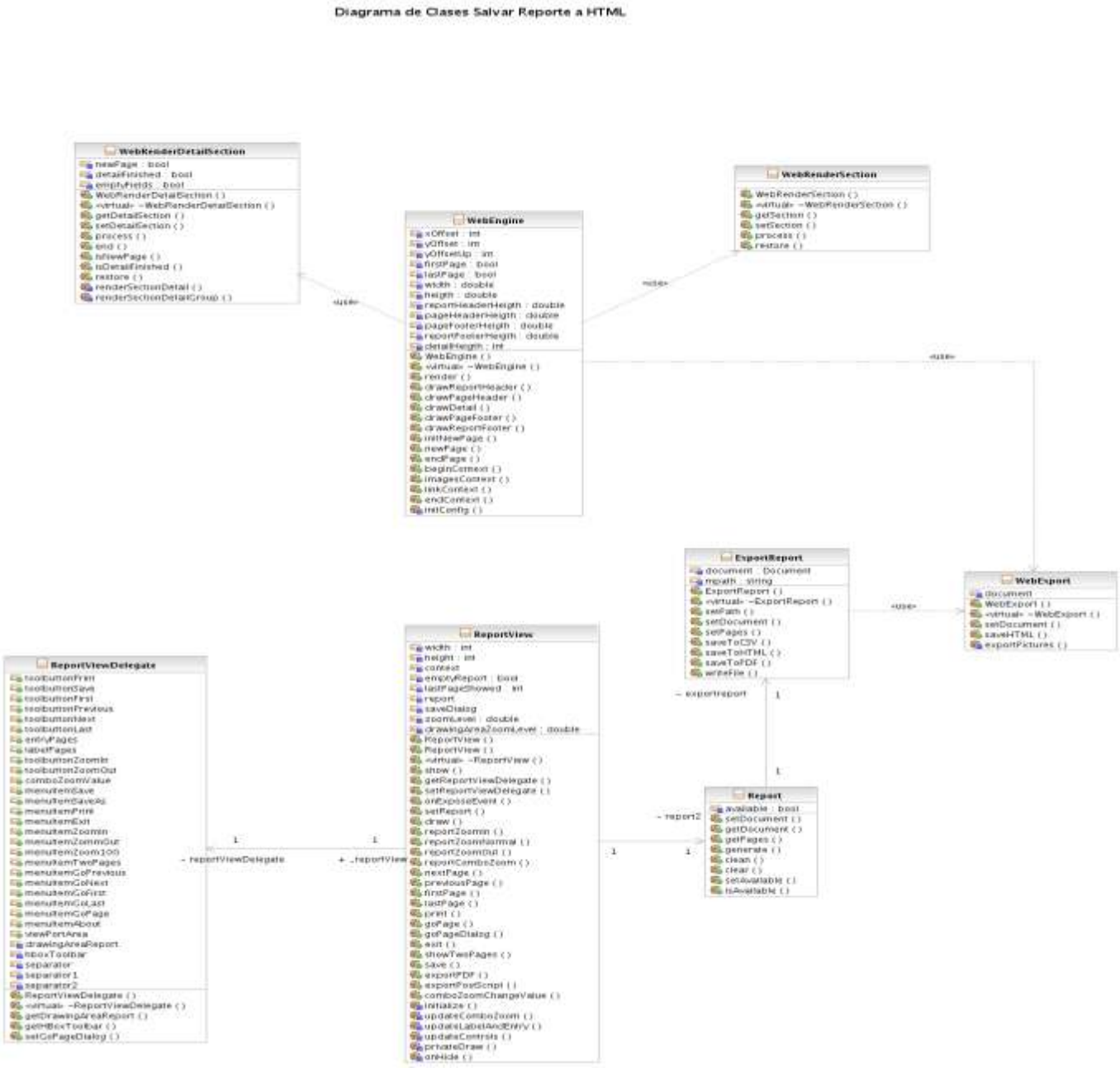

Fig. 9 Diagrama de Clase

Para una mejor comprensión de que es lo que hace esta funcionalidad se muestra a continuación un diagrama de secuencia que muestra la interacción entre dichas clases para llevar a cabo la realización de la funcionalidad de Desplegar Reporte.

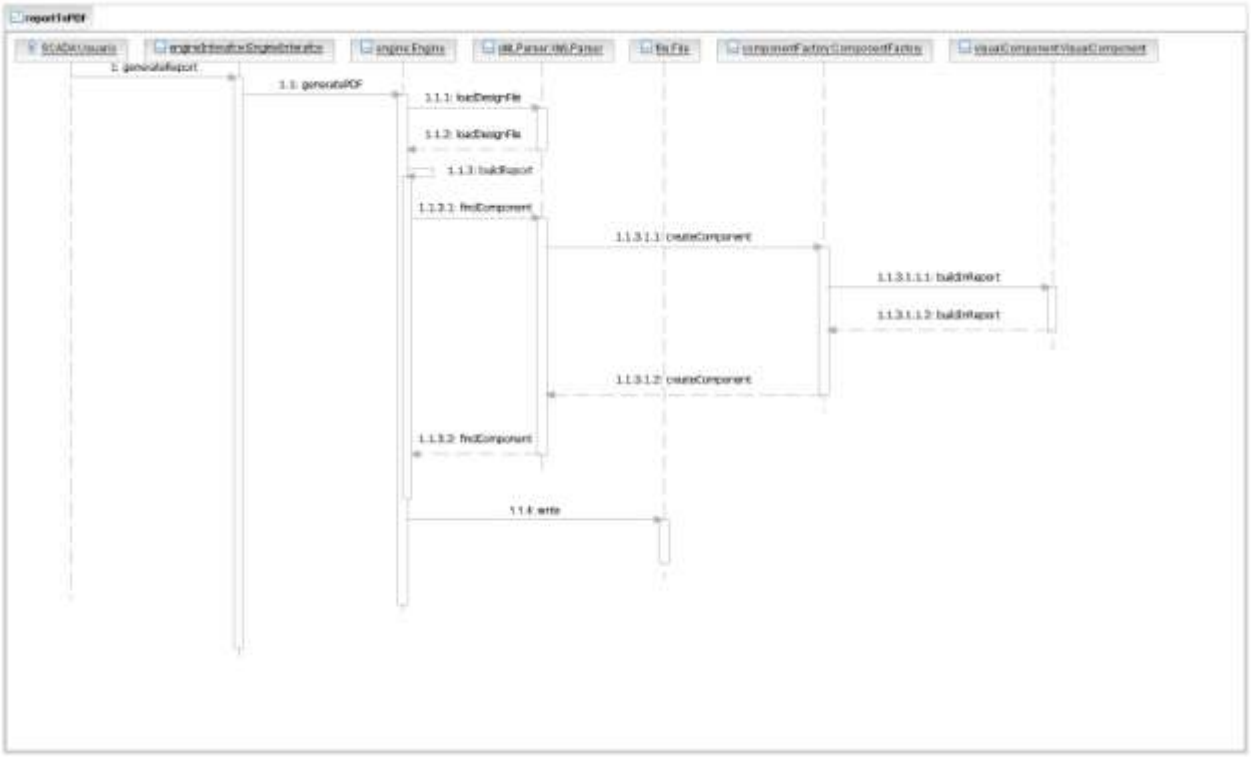

Fig. 10 Diagrama de secuencia que representa la generación y salva de un informe en HTML.

1.10.3.1 Descripción de las principales clases

Nombre: ExportReport		
Descripción: Clase encargada de exportar los reportes a los diferentes formatos, tales como: CSV, HTML y PDF.		
Tipo de clase: Controladora		
Atributo		**Tipo**
mpath		String
document		Document
pages		PageCollection
Para cada responsabilidad:		
Nombre:	void setPath(std::string mpath);	
Descripción:	Asigna nueva directorio a salvar.	
Nombre:	void setDocument(Document& document);	
Descripción:	Asigna nueva documento para salvar.	

Nombre:	void setPages(PageCollection& pages);
Descripción:	Asigna una nueva colección de páginas vacías para salvar a PDF.
Nombre:	void saveToCSV();
Descripción:	Método encargado de salvar los diferentes reportes al formato CSV.
Nombre:	void saveToHTML();
Descripción:	Método encargado de salvar los diferentes reportes al formato HTML.
Nombre:	void saveToPDF();
Descripción:	Método encargado de salvar los diferentes reportes al formato PDF.

Nombre: WebExport	
Descripción: Es la clase principal que se encarga de salvar los datos de un documento a formato HTML.	
Tipo de clase: Controladora	
Atributo	**Tipo**
document	Document
Para cada responsabilidad:	
Nombre:	void setDocument(Document& document)
Descripción:	Asigna valores a todos los atributos a partir de los datos del document.
Nombre:	void saveHTML()
Descripción:	Salva el documento a formato HTML
Nombre:	void exportImage()
Descripción:	Exporta todos los objetos de tipo Picture y Chart a formato PNG.

Nombre: WebEngine	
Descripción: Es la clase principal que se encarga de renderear los datos de un documento para que sea exportado a formato HTML.	
Tipo de clase: Controladora	
Atributo	**Tipo**
firstPage	bool
lastPage	bool
width	double
heigth	double
reportHeaderHeigth	double
pageHeaderHeigth	double
pageFooterHeigth	double
reportFooterHeigth	double
detailHeigth	int

Para cada responsabilidad:	
Nombre:	void initConfig(Document& document)
Descripción:	Asigna valores a todos los atributos a partir de los datos del document.
Nombre:	void render(PageCollection& pages, Document& document)
Descripción:	Renderea document en una colección de páginas HML para ser exportadas.
Nombre:	void drawReportHeader(Section*, WebRenderSection&)
Descripción:	Pinta un objeto de tipo Section en la sección ReportHeader de una página HTML.
Nombre:	void drawPageHeader(Section*, WebRenderSection&)
Descripción:	Pinta un objeto de tipo Section en la sección PageHeader de una página HTML.
Nombre:	void drawDetail(SectionDetail*, Document&, WebRenderDetailSection&)
Descripción:	Pinta un objeto de tipo SectionDetail en la sección Detail de una página HTML.
Nombre:	void drawPageFooter(Section*, WebRenderSection&)
Descripción:	Pinta un objeto de tipo Section en la sección PageFooter de una página HTML.
Nombre:	void drawReportFooter(Section*, WebRenderSection&)
Descripción:	Pinta un objeto de tipo Section en la sección ReportFooter de una página HTML.
Nombre:	void initNewPage(PageCollection&, Document&, WebRenderSection&)
Descripción:	crea la primera página en la colección de páginas HTML para ser pintada
Nombre:	void newPage(PageCollection&, Document&, WebRenderSection&)
Descripción:	crea la una nueva página en la colección de páginas HTML para ser pintada
Nombre:	void endPage(Document&, WebRenderSection&)
Descripción:	crea la ultima página en la colección de páginas HTML para ser pintada
Nombre:	void beginContext()
Descripción:	Escribe los datos de iniciación en el archivo HTML a exportar.
Nombre:	void imageContext()
Descripción:	Escribe los datos de los objetos exportados como imágenes en el archivo HTML.
Nombre:	void linkContext()
Descripción:	Escribe los datos que representan el paginador en la página HTML a exportar.
Nombre:	void endContext()
Descripción:	Escribe los datos de finalización en el archivo HTML a exportar.

1.10.3.2 Fragmento de Código

Para la exportación de los reportes a diferentes formatos se utiliza la clase *ExportReport* donde se validan y exportan los reportes previamente ya generados, además de hacer el uso de excepciones para el seguimiento de errores al guardar estos archivos exportados en disco.

Para los reportes que se deseen exportar a formato HTML se hace el uso del método:

```
void ExportReport::saveToHTML()
```

Ejemplo de una implementación del método *saveToHTML():*

```
void ExportReport::saveToHTML()
{
  std::string pathHTML = mpath;
  writeFile(pathHTML);
  SCADA::Report::Web::Base::WebExport webExport;
  webExport.setDocument( this->document );
  webExport.saveHTML(pathHTML);
}
```

En el método se puede observar que se hace uso del método *writeFile (pathHTML)* el cual crea el directorio donde se va guardar el reporte en formato HTML, una vez validado y creado este directorio se instancia un objeto *WebExport* y para exportar a formato HTML el reporte se hace el uso del método:

```
void WebExport::saveHTML(std::string directory)
```

Donde:

directory: Es el directorio donde se guardará el fichero HMTL.

```
void WebExport::saveHTML(std::string directory)
 {
    std::string parentDirectory = directory;
    std::string imageDirectory  = parentDirectory + "/Image";
    path parentPath(parentDirectory);
    path imagePath(imageDirectory);
    if(exists(parentPath))
     {
       throw  std::logic_error("existe ya ese directorio");
     }
    else
     {
      create_directory(parentPath);
      create_directory(imagePath);
     }

    exportPictures(imageDirectory);
```

```
    PageCollection pageWebCollection;
    WebEngine webEngine;
    std::ostringstream headerText;
    headerText<< webEngine.beginContext(document).str();
    webEngine.render(pageWebCollection,document);
    std::ostringstream footerText;
    footerText<< webEngine.endContext().str();
    PageCollection::iterator it = pageWebCollection.begin();
    int size = pageWebCollection.size();
    int cont = 1;

    while(it != pageWebCollection.end())
      {
        Page* currentPage = &(*((*it).get()));
        std::ostringstream linkText;
        ofstream htmlFile;
        std::string textHTML = parentDirectory + "/page" +
boost::lexical_cast<string>(cont)+".html";
        htmlFile.open(textHTML.data());
        htmlFile <<headerText.str();
        htmlFile <<currentPage->getContext().str();

        if(cont == 1)
          {
            webEngine.linkContext(0,0,2,size,linkText);
          }
        else
         {
           if(cont == size)
             {
               webEngine.linkContext(1,size - 1,0,0,linkText);
             }
           else
             {
                webEngine.linkContext(1,cont - 1,cont +    1,size,linkText);
             }
            }
        htmlFile <<linkText.str();
        htmlFile <<footerText.str();
        htmlFile.close();
        cont++;
        it++;
      }
  }
```

Es válido destacar que para obtener los datos se utiliza un *WebEngine*, que almacena toda la información en una colección de cadenas de caracteres que representa cada página HTML, luego son procesadas y almacenadas en disco.

1.11 Conclusiones

En los sistemas SCADA se utilizan además HMI interactivas, que permiten al operador detectar las anomalías o alarmas en el proceso y mediante la pantalla del ordenador solucionar el problema ejecutado acciones en tiempo real, y algo unas de las funcionalidades principales es la visualización de reportes mediante la adquisición de los datos, representación gráfica de los datos, creación de gráficos (pastel, líneas, puntos etc.) Lo que permite la explotación de los datos para la gestión de la calidad, control estadístico, gestión administrativa.

En este capítulo se ha realizado la descripción de la mayor parte de los elementos implementados, se han seleccionado las funcionalidades más significativas y se ha descrito detalladamente las principales clases que intervienen en su desarrollo

CAPÍTULO 3

Validación de la Solución

Introducción

En cualquier sistema complejo, es crítico probarlo para saber que funcionará una vez que se lance en vivo. Muchos grupos de trabajo se limitan a dedicar una o dos personas a probar el sistema. En este capítulo se realiza el proceso de pruebas para validar el correcto funcionamiento de la solución propuesta. Aunque estas pruebas no garantizan la que la aplicación no tenga de defectos; pueden demostrar que existen errores en el software y así posteriormente poderlos corregir.

1.12 ¿Qué es una Prueba de Unidad?

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código.Las librerías de pruebas de unidad formalizan este trabajo al proporcionar clases para pruebas.

La prueba de unidad ayuda a que el módulo se haga independiente. Esto quiere decir que un módulo que tiene una prueba de unidad se puede probar independientemente del resto del sistema.

Una buena prueba de unidad:

- No requiere de mucho código para prepararla.- Una prueba de unidad cuya preparación toma 3 minutos (conectándose a la base de datos, levantando un servidor de Web, etcétera) se debe considerar demasiado pesada para ejecutarse cada vez que se hagan cambios.
- Prueba sólo un método. Aunque no siempre es necesario hacer "una prueba por método", sí es importante asegurar que cubrimos toda la clase. Muchas veces se requiere más de un método de prueba por cada método en la clase, para poder probar diferentes datos y casos extremos.
- Verifica los resultados con aserciones. De preferencia se debe de chequear sólo un resultado. Se pueden chequear varias partes del mismo resultado, pero diferentes condiciones deben probarse en un método separado.

- Deja los datos de la misma manera en que los encontró. Cuando la prueba comienza, los datos se preparan si es necesario, pero cuando la prueba termina, es importante que el sistema borre los datos después de la verificación. Es por esto que muchas librerías de pruebas unitarias tienen métodos llamados setup() y teardown() o similares.

1.13 Diseño de las pruebas de unidades que permitan validar la solución propuesta

Para la realización de las pruebas se utilizan como recursos físicos: computadoras con un microprocesador Core Duo con una velocidad del CPU de 2.16 Ghz, la memoria RAM que tiene es de 1 Gb y un disco duro con capacidad de 100 Gbytes. Los recursos lógicos con los que realizamos las pruebas fueron: sistema operativo Debian GNU/Linux con núcleo 2.6.21, el IDE de desarrollo es el Eclipse 3.2 con el plug-in CDT 2.1 para programar en C++, y el framework para automatizar las pruebas es el CxxTest, que se integra con el entorno de desarrollo y sirve para el lenguaje que utilizamos. Todos los casos de pruebas que han sido realizados son del tipo Pruebas de Unidad y tienen como objetivo aislar el código fuente y validar el correcto funcionamiento de los métodos que se implementan. Se realizaron por las principales funcionalidades y las clases dentro del código correspondiente.

1.14 Pruebas a las clases dentro del método “Desplegar Reporte”

1.14.1 Clase TestDocument

Prueba unitarias de la clase: ***Document***

Casos de prueba del método: ***void setName (Name name)***
Variables a considerar en el caso de prueba: ***Name name***

Clase de Equivalencia para la variable: ***name***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Verificar que cambie el valor del atributo name.	Válida

1.14.1.1 Casos de Prueba:
1-*void testSetName()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que se pueda cambiar el valor del atributo	Cadena pasada como parámetro.	El la misma cadena pasada en los datos de entrada	El la misma cadena pasada en los datos de entrada	

Casos de prueba del método: ***void addPageHeader(Section * pPageHeader)***
Variables a considerar en el caso de prueba: ***Section * pPageHeader***

Clase de Equivalencia para la variable: ***pPageHeader***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Valores correctos para una objeto de tipo Section.	Válida
2	Valores correctos para una objeto de tipo Section.	Válida

1.14.1.2 Casos de Prueba:
1-*void testAddPageHeader_1 ()*
2-*void testAddPageHeader_2 ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que el método funciona para un puntero de tipo Section instanciado.	Section * section = new Section ();	Asignación correcta de la entrada a otra variable de tipo Section;	Asignación correcta de la entrada a otra variable de tipo Section;	
2	Verificar que el método	Section* section = NULL;	Asignación correcta de la	Asignación correcta de la	

	funciona para un puntero de tipo Section no instanciado.		entrada a otra variable de tipo Section;	entrada a otra variable de tipo Section;	

1.14.2 Clase TestSection

Prueba unitarias de la clase: ***Section***

Casos de prueba del método: ***void setWidth(WindowCoord width)***
Variables a considerar en el caso de prueba: ***WindowCoord width***

Clase de Equivalencia para la variable: ***width***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	El valor de la variable estará entre los valores permisibles de la variable.	Válida

1.14.2.1 Casos de Prueba:

1-*void testSetWidth ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verifica que dado valores que se encuentren dentro de la frontera de validez se devuelvan los datos esperados	75	75	75	

Casos de prueba del método: ***PrintFrequency getPrintFrecuency()***

Variables a considerar en el caso de prueba: ***PrintFrequency printFrecuency***

Clase de Equivalencia para la variable: ***printFrecuency***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Verificar que devuelve el valor correcto del atributo.	Válida

1.14.2.2 Casos de Prueba:
1-***void testGetPrintFrecuency ()***

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que devuelve el valor correcto del atributo.	printFrecuency = *none*	*none*	*none*	

1.14.3 Clase TestReport

Prueba unitarias de la clase: ***Report***

Casos de prueba del método: ***void setAvailable(bool available)***
Variables a considerar en el caso de prueba: ***bool available***

Clase de Equivalencia para la variable: ***available***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia

1	Verificar que cambie el valor del atributo available.	Válida

1.14.3.1 Casos de Prueba:
1-***void testSetAvailable ()***

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verifica que se pueda cambiar el valor del atributo.	false	false	false	

Casos de prueba del método: ***void clean()***
Variables a considerar en el caso de prueba: ***PageCollection pages***

Clase de Equivalencia para la variable: ***pages***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Verificar que la colección de página quede vacia.	Válida

1.14.3.2 Casos de Prueba
1-*void testClean()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Comprobar que la colección de paginas quede vacia.	PageCollection pages	0	0	

1.14.4 Clase TestPageOption
Prueba unitarias de la clase: ***PageOption***

Casos de prueba del método: ***void setFormat(String pageSize)***

Variables a considerar en el caso de prueba: ***String pageSize***

Clase de Equivalencia para la variable: ***pageSize***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Como la función a probar recibe una variable tipo String que puede solo puede tomar valores en el dominio de los tipos de formatos de hoja (A3, Letter, A4,…).	Válida
2	Valores no válidos para el dominio de los tipos de formatos de hoja, valor vacio.	Válida
3	Valores no válidos para el dominio de los tipos de formatos de hoja, valor fuera del dominio.	Inválida

1.14.5 Casos de Prueba:

1-*void testSetFormat_1()*
2-*void testSetFormat_2()*
3-*void testSetFormat_3()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Que se le asigna correctamente el formato pasado como parámetro.	marginLeft = "A3"	Formato con los mismos valores pasados en los datos de entrada.	Formato con los mismos valores pasados en los datos de entrada.	
2	Que se provoca	marginLeft =	Excepción o	Se produce	

	alguna excepción al pasar un valor vacio.	“”	algún tipo de notificación por parte de objeto cuando se le pasa como argumento un valor vacio.	notificación o excepción.	
3	Que se provoca alguna excepción al pasar un valor no valido.	marginLeft = “FQ5”	Excepción o algún tipo de notificación por parte de objeto cuando se le pasa como argumento un valor no valido.	No se produce ninguna notificación ni excepción.	

Casos de prueba del método: ***double getMarginLeft()***

Variables a considerar en el caso de prueba: ***double marginLeft***

Clase de Equivalencia para la variable: ***marginLeft***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Verificar que devuelve el valor del atributo.	Válida

1.14.5.1 Casos de Prueba:

1-*void testGetMarginLeft ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que devuelva el valor del atributo	0.75	0.75	0.75	

1.14.6 Clase TestShape

Prueba unitarias de la clase: ***Shape***

Casos de prueba del método: ***RGBAColor getLineColor()***

Variables a considerar en el caso de prueba: ***RGBAColor lineColor***

Clase de Equivalencia para la variable: ***lineColor***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Como la función a probar recibe una variable tipo RGBAColor pasar valores válidos para el dominio de un objeto de tipo color como pueden se el valor de R, G, B, A los cuales siempre deben estar entre 0 y 255.	Válida
2	Valores no válidos para el dominio de un RGBAColor, al menos uno de los parámetros del color debe estar por debajo de 0 o por encima de 255.	Inválida

1.14.6.1 Casos de Prueba:

1-*void testGetLineColor_1 ()*
2-*void testGetLineColor_2 ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Que se le asigna correctamente el color pasado como parámetro.	RGBA color (255,255,255,0)	Color con los mismos valores pasados en los datos de entrada.	Color con los mismos valores pasados en los datos de entrada.	
2	Que se provoca alguna excepción al pasar un RGBAColor con valores no válidos.	RGBA color (-100,100,260,0)	Excepción o algún tipo de notificación por parte de objeto cuando se le pasa como argumento un valor no válido.	No se produce ninguna notificación ni excepción.	

1.14.7 Clase TestChart

Prueba unitarias de la clase: ***Chart***

Casos de prueba del método: ***void setXIDQuery(unsigned int xIDQuery)***

Variables a considerar en el caso de prueba: ***unsigned int xIDQuery***

Clase de Equivalencia para la variable: ***xIDQuery***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	El valor de la variable estará entre los valores permisibles de la variable.	Válida
2	El valor de la variable estará fuera los valores permisibles de la variable.	Invàlida

1.14.7.1 Casos de Prueba:
1-*void testSetXIDQuery_1 ()*
2-*void testSetXIDQuery_2 ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Que se le asigne correctamente el valor del color pasado como parámetro.	25	25	25	
2	Que se le asigne a la variable el rango un valor dentro del rango	-10	0	187	No se obtuvo la salida esperada ya que la clase no tiene validada la entrada de datos

1.14.8 Clase TestTextComponent

Prueba unitarias de la clase: ***TextComponent***

Casos de prueba del método: ***FontFace getFontFace()***

Variables a considerar en el caso de prueba: ***FontFace fontFace***

Clase de Equivalencia para la variable: ***fontFace***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Se le asignará a la variable un valor de tipo string y se pasará como parámetro.	Válida

1.14.8.1 Casos de Prueba:
1-***void testGetFontFace ()***

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Que se le asigna correctamente el color pasado como parámetro.	fontFace = "Comic Sans"	Fuente con los mismos valores pasados en los datos de entrada.	Fuente con los mismos valores pasados en los datos de entrada.	

1.15 Pruebas a las clases dentro del método "Imprimir Reporte"

1.15.1 Clase TestReportView
Prueba unitarias de la clase: ***ReportView***

Casos de prueba del método: ***void setReport(Report* report)***

Variables a considerar en el caso de prueba: ***Report* report***

Clase de Equivalencia para la variable: ***report***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Valores correctos para una objeto de tipo Report.	Válida

1.15.1.1 Casos de Prueba:
1-*void testSetReport ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que el método funciona para un puntero de tipo Report.	Report * report = new Report ();	Asignación correcta de la entrada a otra variable de tipo Report;	Asignación correcta de la entrada a otra variable de tipo Report;	

1.16 Pruebas a las clases dentro del método “Exportar Reporte a HTML”

1.16.1 Clase TestExportReport

Prueba unitarias de la clase: ***ExportReport***

Casos de prueba del método: ***void setPath(String path)***

Variables a considerar en el caso de prueba: ***String path***

Clase de Equivalencia para la variable: ***path***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Un camino de directorio local de entrada correcto.	Válida
2	Un camino de directorio local de entrada incorrecto.	Vàlida

1.16.1.1 Casos de Prueba:

1-*void testSetPath ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación

1	Que se le asigna correctamente el camino pasado como parámetro.	path = "/home/scada"	Camino con los mismos valores pasados en los datos de entrada.	Camino con los mismos valores pasados en los datos de entrada.	
2	Que se provoca alguna excepción al pasar un camino incorrecto o no accesible.	path = "/c///scada"	Excepción o algún tipo de notificación por parte de objeto cuando se le pasa como argumento un camino no valido.	Se produce notificación o excepción.	

1.16.2 Clase TestWebExport

Prueba unitarias de la clase: ***WebExport***

Casos de prueba del método: ***void setDocument(Document& document)***

Variables a considerar en el caso de prueba: ***Document document***

Clase de Equivalencia para la variable: ***document***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Como la función a probar recibe una variable de tipo Document, se construye y se le pasa como parámetro.	Válida

1.16.2.1 Casos de Prueba:

1-*void testSetDocument()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Verificar que el	Document	Asignación	Asignación	

	método funciona para un objeto de tipo Document.	document;	correcta de la entrada a otra variable de tipo Document.	correcta de la entrada a otra variable de tipo Document.	

1.16.3 Clase TestWebPicture

Prueba unitarias de la clase: ***WebPicture***

Casos de prueba del método: ***void save(String path)***

Variables a considerar en el caso de prueba: ***String path***

Clase de Equivalencia para la variable: ***path***

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Un camino de directorio local de entrada correcto.	Válida

1.16.3.1 Casos de Prueba:

1-*void testSave ()*

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida	Observación
1	Comprobar que no se provocan excepciones.	path = “/home/scada”	No se debe provocar ningún tipo de error.	No se provoca error.	

1.17 Conclusiones

Las pruebas unitarias arrojaron resultados de importancia considerable, en primer lugar ayudaron a identificar algunos errores que pasamos por alto en la codificación, principalmente en elementos que reutilizamos de experiencias anteriores; además pudieron validar el código fuente producido en la etapa de implementación. Debido al tamaño del sistema y a su complejidad se identificaron un grupo de clases que podían ser “probadas” dentro de las principales funcionalidades del sistema para asegurar el correcto funcionamiento de las mismas.

En general las pruebas unitarias ofrecieron valiosa información que ayudó a la corrección de los errores identificados y a implementar un producto con mayor calidad y en condiciones favorables para continuar su implementación y terminación.

Conclusiones

1. A pesar de que los generadores de reportes estudiados son semejantes ninguno llega a cumplir con las expectativas para su uso íntegro en el SCADA.
2. Todos los generadores de informes estudiados usan una filosofía muy similar en su funcionamiento, consistente en la creación de diseños en un ambiente de edición y posterior generación del informe, haciendo uso del diseño creado y los datos necesarios, provenientes de una fuente de datos determinada.
3. El paradigma de software libre facilita en gran medida la reutilización de código y un desarrollo acelerado de aplicaciones, posibilitando el estudio exhaustivo de código existente afín con la aplicación que se diseña.
4. La arquitectura en MVC es muy conveniente para el desarrollo del generador de informes para un SCADA, dado la facilidad de resistir el cambio de los requisitos y el alto desacoplamiento logrado con los demás subsistemas integrantes del SCADA.
5. La realización de la implementación basada en el análisis y diseño que permiten la interacción efectiva con el cliente y la validación de las funcionalidades exigidas en los requisitos.

Recomendaciones.

- Continuar el desarrollo de nuevas funcionalidades del generador de reportes del SCADA.
- Continuar trabajando en el establecimiento de dependencias entre las clases genéricas planteadas en el diseño y las clases concretas proporcionadas por las bibliotecas GTK y Boost que faciliten a los desarrolladores la complementación del mismo.
- Evaluar la posibilidad del uso de la implementación del generador de reportes en otros sistemas en el plano nacional utilizando las funcionalidades descritas en el presente trabajo.

Referencias Bibliográficas.

1. JORGE, OSMANY. Modelación de un generador de informes para sistemas de supervisión, control y adquisición de datos. Ciudad de La Habana, Universidad de las Ciencias Informáticas, Mayo 2007
2. AGATA.ORG. Agata Report. Disponible en: http://www.agata.org.br/us/index.php (11/01/2007)
3. ALBERTO MOLPECERES TOURIS, M. P. M. Arquitectura empresarial y software libre, J2EE, 18/08/2002. [Disponible en: http://www.javahispano.org/articles.article.action?id=70 (05/01/2007)
4. ARINA OYAGA, G. informáticos en tiempo real. Ejemplos prácticos. España, Marzo de 2000. p.
5. BANKHACKER.COM. Web Services. Disponible en: http://web-services.bankhacker.com/ (05/01/2007).
6. CRYSTALREPORTS.ORG. Crystal Reports XI from Business Objects. Disponible en: http://www.businessobjects.com/products/reporting/crystalreports/start_xi.asp (11/01/2007)
7. DISTEFANO., M. Comunicaciones en entornos industriales. . Mendoza. Argentina, Universidad Nacional de Cuyo., 1999. p.
8. FOUNDATION, O. OPC Historical Data Access Specification. . 1.20., V., 2003.
9. GONZÁLEZ, C. D. Curso: Integral de diseño. Programación de sitios dinámicos con MySQL y PHP. Disponible en: http://www.usabilidadweb.com.ar/x_int.php(05/01/2007)
10. Curso: Sitios Web dinámicos con Base de Datos PostgreSQL y PHP. Disponible en: http://www.usabilidadweb.com.ar/postgre.php (05/01/2007).
11. HERNÁNDEZ BASSO, M. ¿Convivencia con lo invisible? 2. Ciudad de la Habana; Cuba, Septiembre 2002. p.
12. IEEE. Tutorial Course. Course Text. Fundamental of supervisory control systems; Engineering Societies Library, 18 de febrero 1982.
13. JAMES JACOBSON, I. B., GRADY Y RUMBAUGH. El Proceso Unificado de Desarrollo de Software. La Habana, 2004. p.
14. JASPERREPORTS. Home. Disponible en: http://jasperreports.sourceforge.net/ (07/01/2007)
15. JFREE.ORG. JFreeReport. Disponible en: http://www.jfree.org/jfreereport/ (07/01/2007)

16. KOFFICE.ORG. The KOffice Project. Disponible en: http://www.koffice.org/kugar/
17. LARMAN, C. UML y patrones. Introducción al análisis y diseño orientado a objetos. 2. 2004. p.
18. MANAGER, R. Report Manager Official page. Disponible en: http://reportman.sourceforge.net/ (07/01/2007)
19. MICROSOFT. Microsoft Office Online: Access. Disponible en: http://office.microsoft.com/es-es/FX010857913082.aspx (11/01/2007)
20. MOLPECERES, A. Procesos de desarrollo: RUP, XP y FDD. Disponible en: http://www.javahispano.org/articles.article.action?id=76 (05/01/2007).
21. OPENOFFICE.ORG. Home. Disponible en: http://www.openoffice.org/ (11/01/2007)
22. OPENREPORT. Open Source and profesional reporting solution. Disponible en: http://openreport.org/ (07/012007)
23. REKALLREVEALED.ORG. Rekall: The database front-end for KDE and the Web. Disponible en: http://www.rekallrevealed.org/kbExec.py
24. SCADA, E. P. Especificación de requisitos de software.: Versión 3.0. Venezuela, 2006.
25. Plataforma SCADA PDVSA. Venezuela, PDVSA, 2005.
26. SCADA, S. Red Nacional de Gasoductos; Sistema SCADA. Colombia, Bucaramanga, 2002.
27. WIKIPEDIA, L. E. L. MySQL. Disponible en: http://es.wikipedia.org/wiki/MySQL (05/01/2007)
28. Oracle. Disponible en: Oracle. http://es.wikipedia.org/wiki/Oracle (05/01/2007)
29. Sistemas Gestores de Bases de Datos. Disponible en: http://es.wikipedia.org/wiki/DBMS (05/01/2007)
30. YUNIER SOTO LÓPEZ, N. M. Y. S. R. Propuesta para un sistema de Catalogación y Recuperación de Recursos de Información. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría" junio 2004.
31. MOYA, R. *Introducción al proyecto GNOME.*, 2004. [Disponible en: http://www.es.gnome.org
32. PDVSA Soberanía plena en soluciones AIT para el sector energético aportando valor social.

Bibliografía.

JAMES JACOBSON, I. B., GRADY Y RUMBAUGH. *El Proceso Unificado de Desarrollo de Software.* La Habana, 2004. p.

LARMAN, C. *UML y patrones. Introducción al análisis y diseño orientado a objetos*. 1. 2004. p.

---. *UML y patrones. Introducción al análisis y diseño orientado a objetos*. 2. 2004. p.

PRESSMAN, R. S. *Ingeniería de Software. Un enfoque práctico*. 5. La Habana, 2005. p.

Anexos
Anexo 1: Evaluación de generadores de informes.

La principal característica a tener en cuenta en el proyecto que ocupa a la hora de reutilizar software desarrollado por otras personas es la liberación del código que acompaña la aplicación de aquí que el análisis que se presenta a continuación se divida en generadores de informes propietarios (no se puede tener acceso a su código) y generadores de informes desarrollados bajo los principios de software libre. Para ambos casos se exponen las funcionalidades que ponen a disposición de los usuarios los distintos generadores de informe, poniendo mayor énfasis en las características de aquellos que están desarrollados siguiendo paradigmas de software libre, con la posibilidad de reutilización de su código.

1.18 Generadores de informes propietarios:

1.18.1 Crystal Reports Profesional

Crystal Reports(CRYSTALREPORTS.ORG) es un programa especializado en la generación de informes. Es una herramienta para volcar, filtrar y analizar datos. Está orientada a usuarios y programadores:

- Para usuarios, es un programa que permite acceder a distintas base de datos y extraer datos.
- Para el programador, ofrece la posibilidad de integrar informes y gráficos desde cualquier lenguaje anfitrión. Muchas versiones de Crystal Reports se han distribuido con entornos de programación de otras compañías; como sucede con dBASE de Borland o Visual Basic de Microsoft. No está atado a ningún tipo de base de datos en particular y proporciona drivers para conectarse a una gran diversidad de ellos.

Crystal Reports Server está compuesto por distintos, aunque interrelacionados, componentes y servicios optimizados para realizar tareas específicas. Estos componentes y servicios incluyen:

- Servicios de datos para un acceso integral y flexible a los datos.
- Herramienta de creación para múltiples posibilidades de formatear datos utilizando Crysttal Reports.
- Servicios de plataforma para publicación, seguridad y procesamientos de informes.
- Herramientas de gestión para administrar los objetos y servicios de Crystal Reports Server.
- Web y Application Services para la integración personalizada de informes con portales y aplicaciones.
- Nivel de usuario para conseguir interacción y visualización de informes.

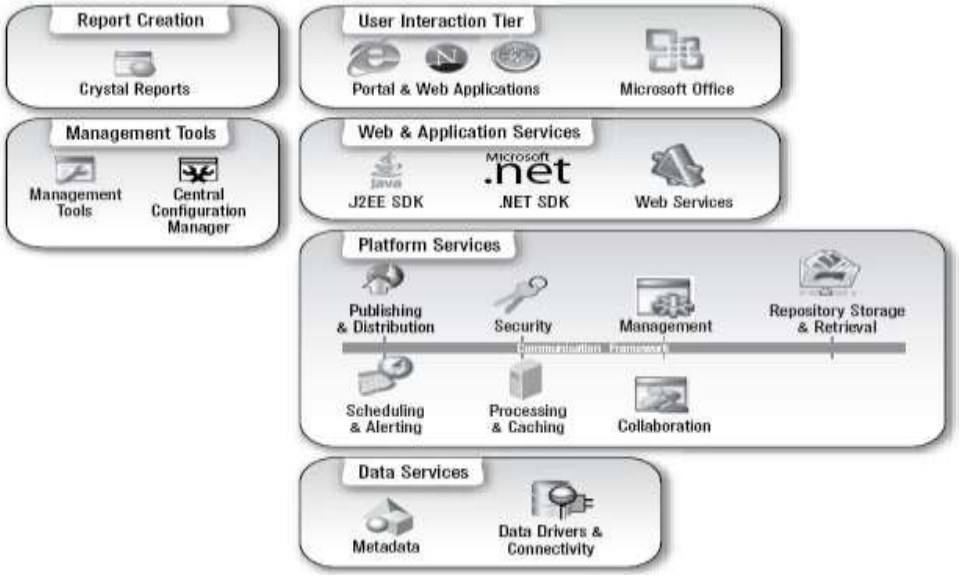

Fig. 11 Arquitectura funcional de Crystal Reports Server.

Es empleado por algunos SCADAs comerciales, Movicon es un ejemplo de ello, como diseñador de informes dada sus reconocidas potencialidades.

Genera distintos tipos de formatos como HTML, PDF, XML, XLS (hoja de cálculo de Microsoft Excel), DOC (documento de Microsoft Word) y es multiplataforma, estando disponible para GNU/Linux, Unix y Microsoft Windows.

Esta aplicación es considerada por muchos una de las herramientas más integrales y flexibles para la generación de informes por lo que resulta un buen paradigma a seguir, incluso referenciada por proyectos de software libre a modo de comparación.

1.18.2 Access de Microsoft Office.

Access(MICROSOFT) es un programa de creación y administración de base de datos. Es una herramienta tanto para desarrolladores como para usuarios. Tiene soporte para varios formatos incluyendo algunos tipos de XML. Entre las herramientas con que cuenta este producto aparece el diseñador de informes, con el que podemos realizar las siguientes tareas:

- Crear un informe con el “Asistente para informes”.
- Previsualizar e imprimir un informe.
- Mover y cambiar el tamaño de un control.

- Modificar las propiedades de formato (fuente, estilo, tamaño de fuente, color, título, etc.).
- Utilizar el Cuadro de controles para agregar controles.
- Utilizar secciones del informe (encabezados, notas a pie, detalles, etc.).
- Esta aplicación sólo es soportada por Microsoft Windows. Puede generar distintos formatos de salida (HTML, XML y XSL).

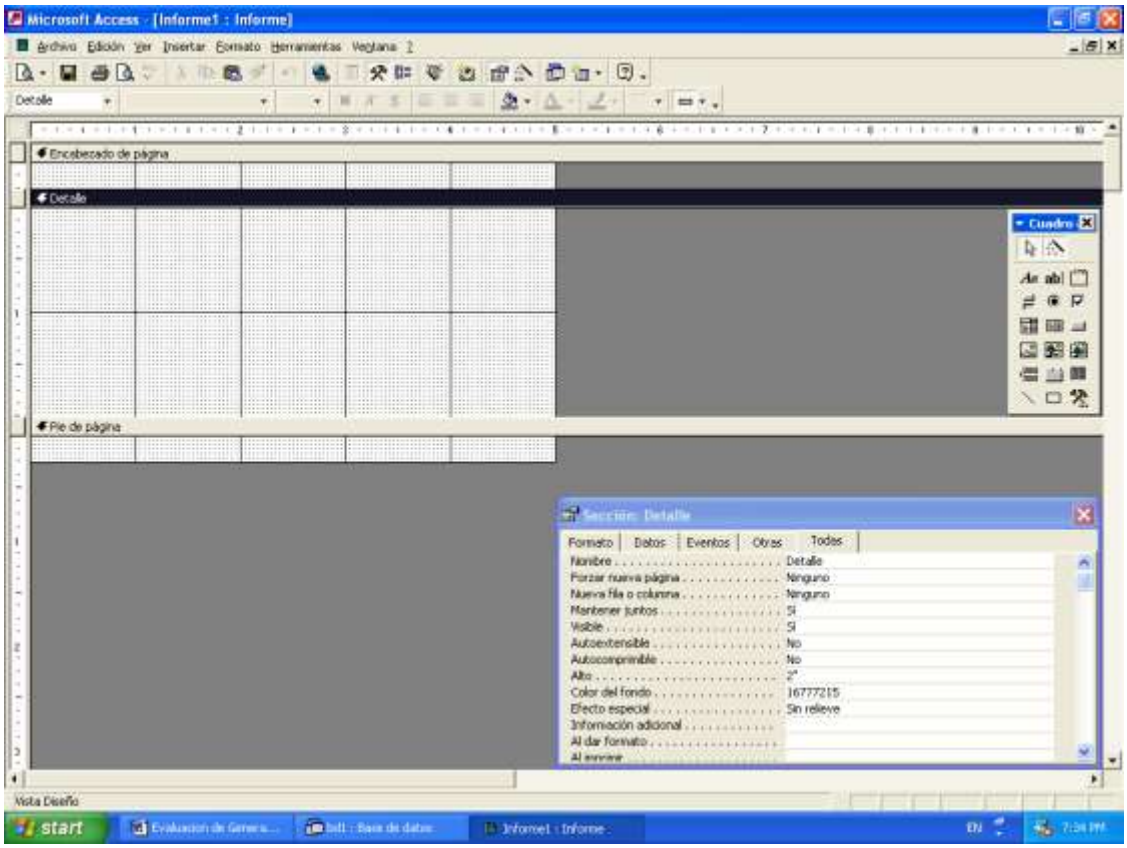

Fig. 12 Vista diseño de Access.

1.19 Generadores de informes de software libre

1.19.1 Report Manager

Report Manager(MANAGER) es una aplicación de generación de informes (Report Manager Designer) y un conjunto de componentes para Delphi, Builder y Kylix. Puede utilizarse desde otros entornos de desarrollo aceptando controles ActiveX (Visual Basic, Visual FoxPro, cualquier lenguaje de Visual Studio.Net). También incluye una librería dinámica estándar con funciones que proporcionan una interfaz con distintos lenguajes de programación como GNU C. Incluye un Servidor de informes TCP a través del cual los clientes obtienen informes procesados en el servidor. Además incluye un servidor Web de informes que genera archivos PDF en tiempo real.

Report Manager es un producto de software libre bajo el modelo MPL (se incluye permiso de uso en aplicaciones GPL), por lo que puede usarse en aplicaciones comerciales, pero cualquier mejora introducida en el motor de impresión debe ser publicada bajo esta licencia. Está soportado por GNU/Linux y Microsoft Windows, y genera varios formatos de salida: PDF, HTML y XSL.

Report Manager dispone de múltiples características, incluyendo algunas exclusivas, como varios subinformes en una página, metaarchivos, fuentes de impresora, secciones externas y subinformes hijos (subinformes en cascada).

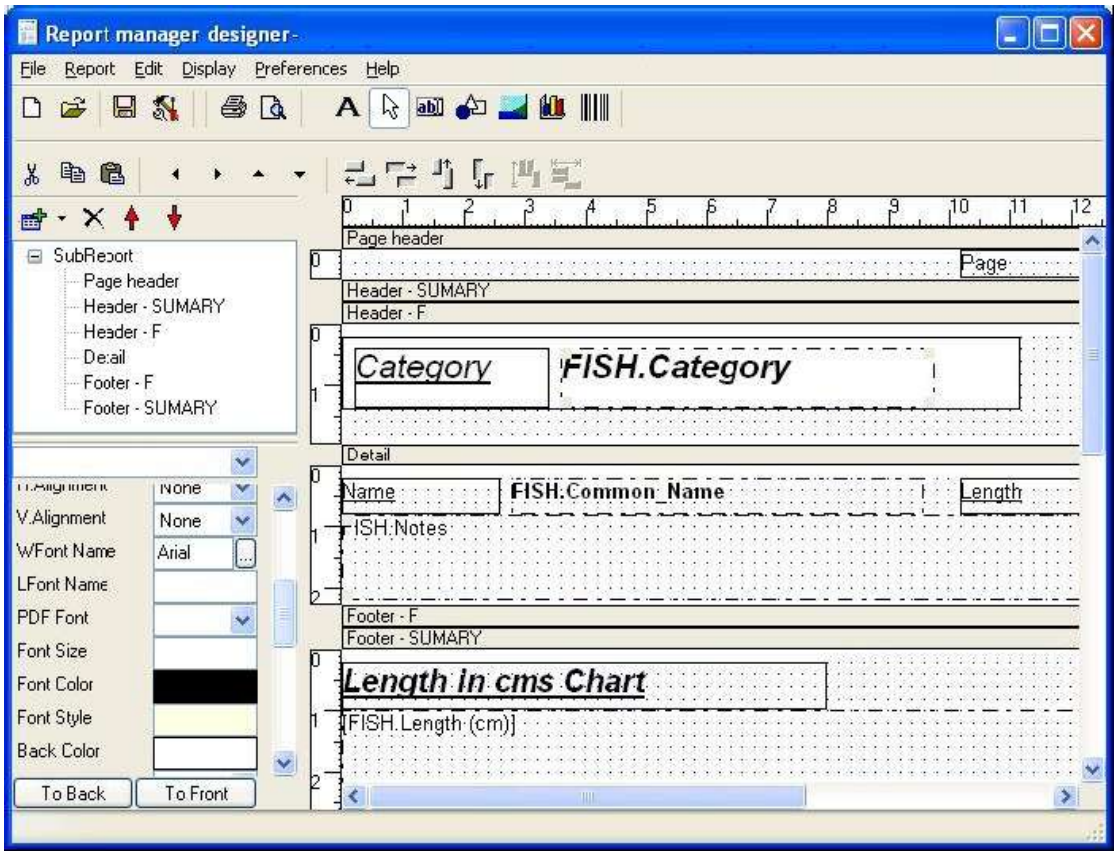

Fig. 13 Pantalla de l a aplicación Report Manager Designer.

Si utiliza Delphi/Kylix/Builder, puede incluir el motor de informes en sus ejecutables, este incluye presentación preliminar, diálogo de impresión, varias opciones para el informe, entre otras, soportando características en relación a otros motores de impresión como:

- Selección de tamaños de papel definidos por el usuario.
- Selección de fuentes de impresora (impresión rápida).
- posibilidad de incluir elementos gráficos y estilos de letra (subrayado, negrita etc.).
- Exportación a PDF escalable y optimizado.
- Soporte de GNU/Linux y Microsoft Windows, Kylix y Delphi.
- Subinformes (impresión de varios informes en la misma página o diferentes páginas).
-

1.19.2 NCReport

NCReport es una herramienta de diseño y generación de informes que permite diseñar de forma visual los informes además de posibilitar su impresión. El ambiente de trabajo para el diseño de informes es bastante amigable al facilitar el acceso a los componentes necesarios para el diseño mediante una barra de componentes y a través del menú. Los componentes que se pueden utilizar son: fecha y hora, campo de expresión, tabla de referencia cruzada, subinforme, número de página, gráfico, texto fijo o etiquetas, imágenes estáticas, hipervínculo y autoforma (línea, cuadro).

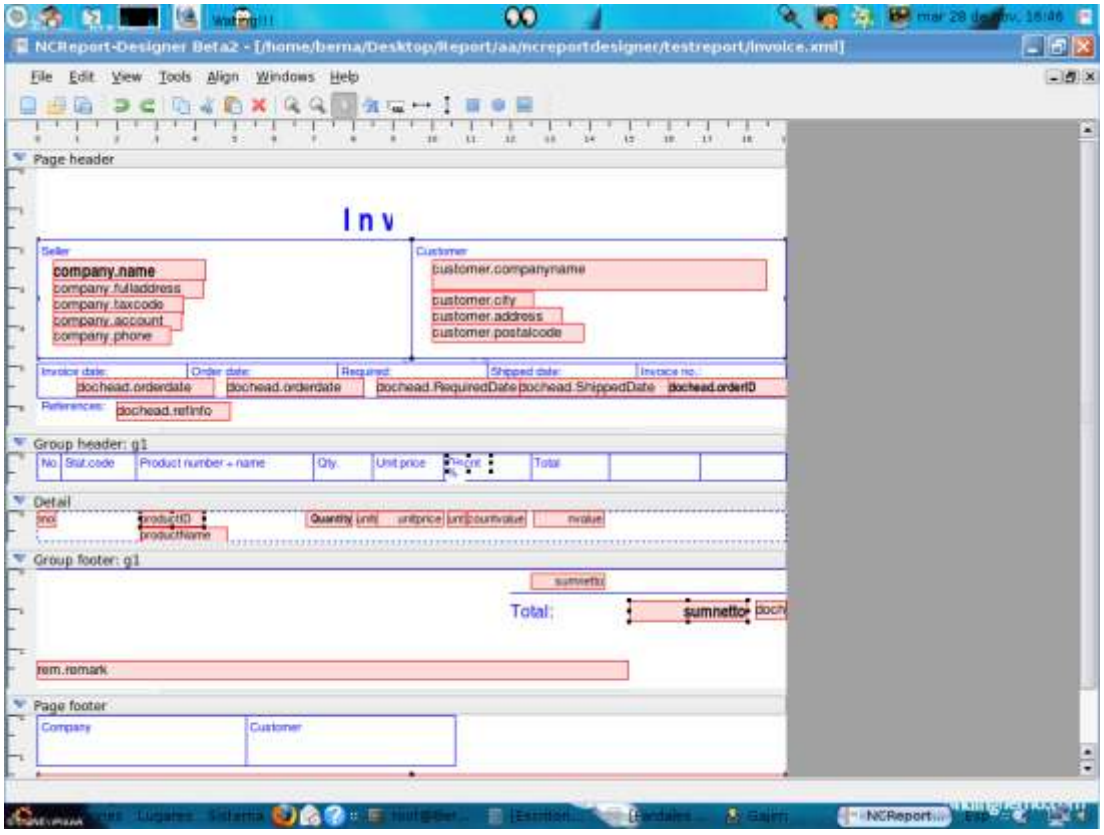

Fig. 14 Ambiente de trabajo en el diseñador de NCReport.

Este generador de informes brinda la posibilidad de conectarse a casi cualquier tipo de base de datos incluyendo MySQL, Oracle, PostgreSQL y SQLite para obtener los datos necesarios para la generación del informe.

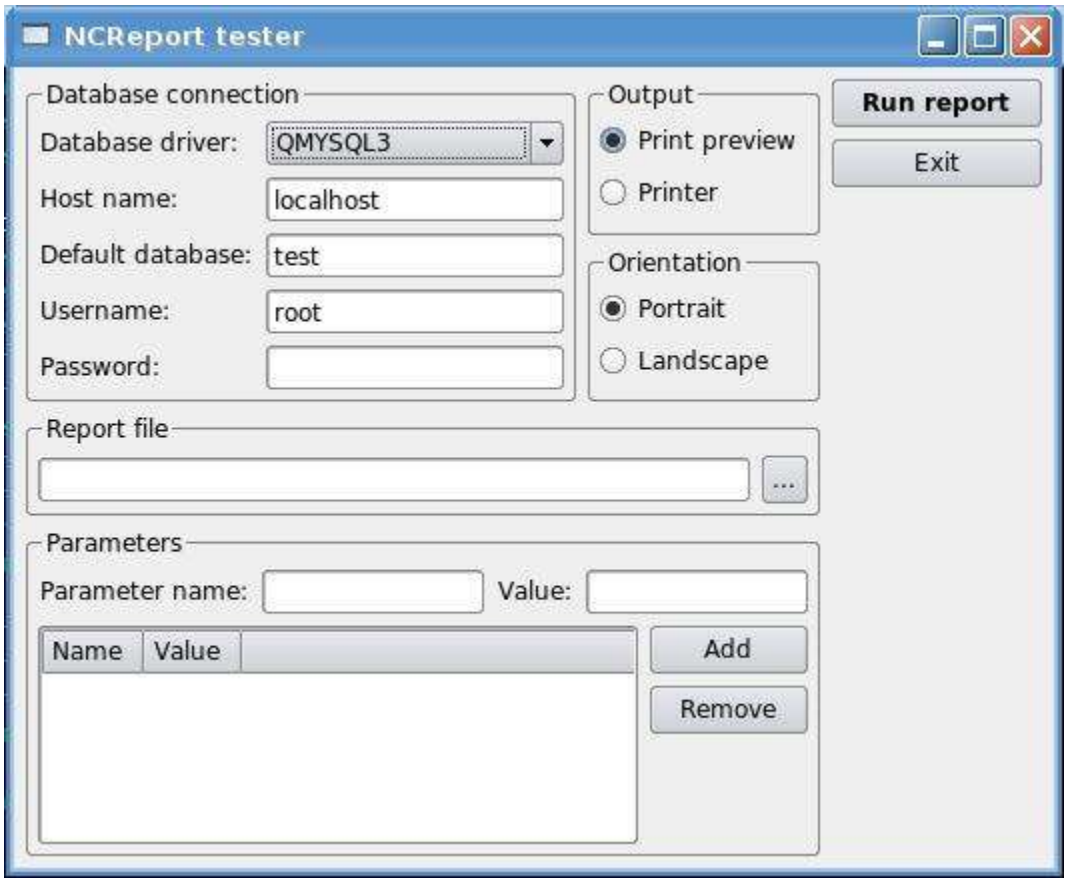

Fig. 15 Pantalla de configuración en NCReport.

Esta es una herramienta muy sencilla que por el momento no permite generar informes hacia ningún formato de salida digital como PDF, ya que solo sirve para diseñar un informe, visualizarlo en vista de impresión e imprimirlo. Los diseños de informes son almacenados en formato XML.

Está desarrollado usando C++ basado en un toolkit de QT, razón por la cual es cualquier programador puede comunicarse con él directamente reutilizando sus clases.

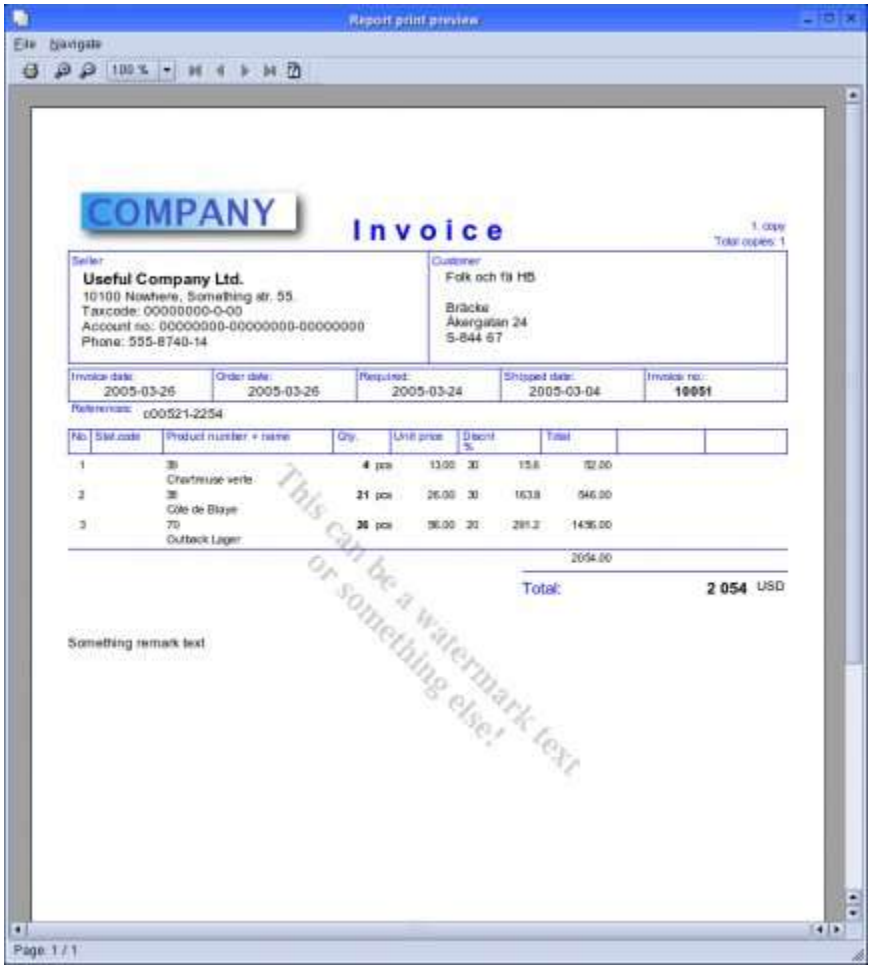

Fig. 16 Ejemplo de informe obtenido con NCReport.

1.19.3 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 mas 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 motor de informe de Kugar trabaja para unir los datos generados con una plantilla o diseño de informe, y producir después el informe final. Tanto los datos como la plantilla son especificados usando XML. Este enfoque implica que las aplicaciones solo necesitan

preocuparse por la exportación de los datos en el formato determinado para ser entendidos por el motor.

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. Pone a disposición del usuario los siguientes menús para facilitar el trabajo, además la mayoría de estas opciones están disponibles en barras de herramientas para su acceso de forma más visual:

- **Archivo** (Nuevo, Abrir, Documentos recientes, Guardar, Guardar como, Importar, Exportar, Imprimir, Impresión previa, entre otros).
- **Edición** (Eliminar, Copiar, Cortar, Pegar).
- **Sección** (Encabezado de página, Encabezado del informe, Encabezado detallado, Detalles, Pie de detalle, Pie de página, Pie del informe).
- **Opciones** (etiquetas, campos, campo calculado, líneas, especiales).
- **Configuración** (barra de herramientas, configuración de acceso directo, configuración de la barra de herramientas).
- **Ayuda** (Manual del diseñador de kugar, Que es esto?, informe de errores, Acerca del diseñador de kugar, Acerca de KDE).

Como en muchas aplicaciones de este tipo, el área de diseño para informes puede verse dividido en varias secciones:

- *Encabezado del informe*, la cual define las secciones de informe que se imprimen generalmente al principio de este.
- *Encabezado de la pagina*, la misma define las secciones de informe que se imprimen generalmente en la parte superior de cada página del informe.
- *Encabezado de detalle*, esta define las secciones del informe que se imprimen ante del detalle en un nivel dado del informe.
- *Detalle*, define las secciones del informe que contienen los datos del informe. El informe puede tener un número ilimitado de detalles.

- *Pie de detalle*, este elemento define las secciones del informe que se imprimen inmediatamente después de detalles de un nivel dado.
- *Pie de la pagina*, esta define las secciones del informe que se imprimen generalmente en el extremo inferior de cada página en el informe
- *Pie del informe* que define la sección del informe que es impresa generalmente al final del informe.

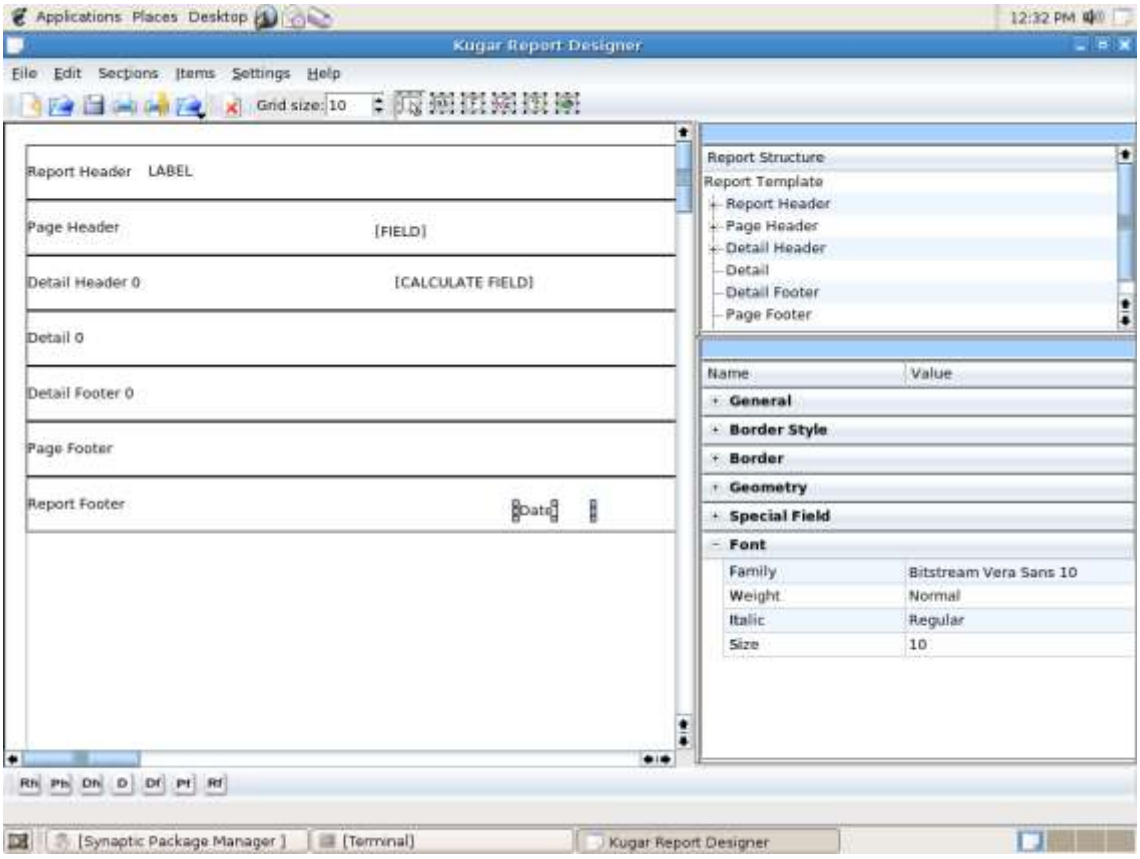

Fig. 17 Vista diseño de Kugar.

Están disponibles para la realización de diseños los siguientes elementos:

- *Etiqueta:* Un área rectangular que puede tener fronteras y se puede llenar por cualquier clase de datos textuales.
- *Campo:* Representan zonas de informaciones; sus valores serán obtenidos y procesados mientras se genera un informe.
- *Campo calculado:* Permite incluir en el informe cuentas, sumas, promedios, etc.
- *Campo especial*: Etiquetas con el texto predefinido, tal como fecha actual o número de página.
- *Líneas:* Permiten refinar el aspecto final del informe.

Tanto los elementos como las secciones tienen sus propiedades características. Esas propiedades definen parámetros geométricos, textuales y de otro tipo. Cada vez que se ubica un elemento, se aplican una serie de propiedades predeterminadas.

Las principales características que presenta la aplicación son:

- Diseñador de informes.
- Impresión de informes en formato Postscript.
- Base de datos neutral, los datos son suministrados al motor de informes en XML.
- Soporte para el acceso directo a base de datos.
- Los diseños de los informes son almacenados en XML.
- Cabeceras y pies de informe.
- Cabeceras y pies de página incluyendo números de página, fecha y hora actual.
- Cálculos de contadores, sumas, medias, varianzas y desviación típica.
- Formato de número basado en valores.
- Formato de fecha y moneda.
- Control de fuentes, colores, alineamiento de texto, etc.
- Dibujo de línea en diferentes estilos.

1.19.4 Agata Report.

Agata Report(AGATA.ORG) es una herramienta de generación de informes, que permite obtener datos desde distintos gestores de base de datos (PostgreSQL, MySQL, SQLite, Oracle, DB2, MSSQL, Informix, InterBase, Sysbase, o FrontBase) y exportarlos a PostScript, texto plano, HTML, XML, PDF y CSV, permitiendo imprimir de una manera muy sencilla los informes obtenidos.

Puede generar gráficos, subtotales, y totales para el informe, introducir los datos en el documento y generar etiquetas de dirección.

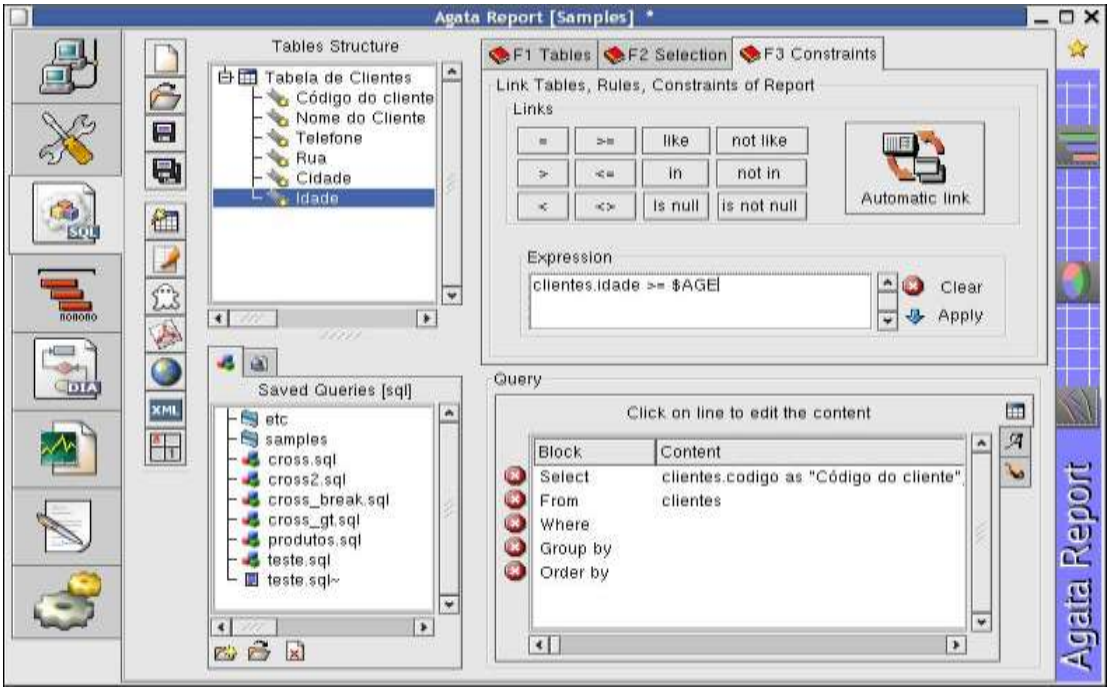

Fig. 18 Conexión a base de datos con Agata Report

Es multiplataforma teniendo soporte en GNU/Linux y Microsoft Windows, y es publicado bajo licencia GPL.

Agata basa su programación en PHP-GTK que es la fusión del lenguaje de script PHP y la librería de objetos GTK+. Con esta librería y junto a la programación sencilla que proporciona el PHP se pueden conseguir aplicaciones GUI (Graphical User Interfaces) muy interesantes y potentes. La programación PHP-GTK se basa totalmente en POO (Programación Orientada a Objetos), ya que los widgets con los cuales se trabaja son clases y se declaran como objetos en el programa. Una de las desventajas es que al ser un lenguaje interpretado se necesita tener el intérprete de PHP, con el consiguiente consumo en tiempo de interpretación.

Este generador de informes provee de una API a programadores para poder manipular la creación, modificación y ejecución de informes, la misma esta disponible solamente para PHP. La forma de ejecutar funciones del Agata desde otro lenguaje es llamarlo como un proceso (consola), utilizando el intérprete de PHP.

1.19.5 Open Report.

Es una solución para la generación de documentos desde diferentes fuentes de datos. La aplicación tiene funciones para usar y presentar gráficos vectoriales, imágenes de mapas de

bits, elementos formateados, plantillas de negocios, datos XML, y une todas estas primitivas para producir un fichero PDF.

Open Report(OPENREPORT) es un paquete formado por tres componentes:

- Tiny RML2PDF es una herramienta para crear documentos PDF. Ésta puede ser usada como una librería Python o como librería binaria independiente. Convierte RML, (dialecto de XML que permite definir la apariencia precisa de un documento que será impreso), a PDF.
- Tiny RML2HTML es una herramienta para convertir el mismo fichero RML a un documento HTML.
- Open reporting server: es una aplicación distribuida que funciona en algunos entornos y puede ser accedida desde diferentes interfaces (SOAP, XMLRPC, NETRPC, PIPE). Ésta se usa para adquirir datos y empleando reglas de formateo generar documentos PDF. El servidor de informes utiliza Tiny RML2PDF y, sin programar, puede usarse para presentar documentos de una determinada compañía, como catálogos automáticamente generados desde una base de datos de productos, facturas creadas en línea, etc. Este producto se distribuye bajo licencia GPL y está soportado por GNU/Linux y Microsoft Windows.

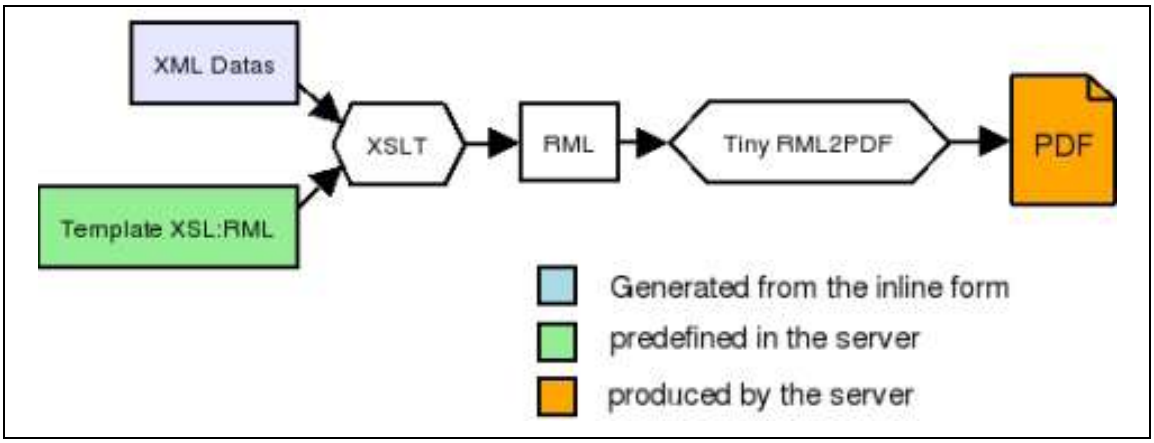

Fig. 19 Proceso de transformación a PDF con OpenReport**.**

1.19.6 Rekall.

Rekall(REKALLREVEALED.ORG) es una interfaz de usuario para bases de datos (Front-End) en entornos KDE similar a Microsoft Access. Sin embargo, no es y no incluye una base de datos. Debido a esto, los datos están almacenados en otra parte como en un servidor SQL. De esta forma Rekall permite gestionar de una forma totalmente gráfica bases de datos (por ahora

MySQL, PostgreSQL, xBase con XBSQL, IBM DB2 y ODBC), permitiendo el diseño de formularios e informes, creación de peticiones a bases de datos, importación y exportación de tablas en diversos formatos para extraer, mostrar, y editar las informaciones contenidas en estas bases de datos, de una forma muy similar al Access de Microsoft, en otras palabras, podemos decir que es una especie de Access para GNU/Linux. La diferencia más importante de Rekall con respecto a Access es que de manera predeterminada no está ligada a ningún motor de base de datos en particular sino que se conecta a varios de los motores de bases de datos existentes.

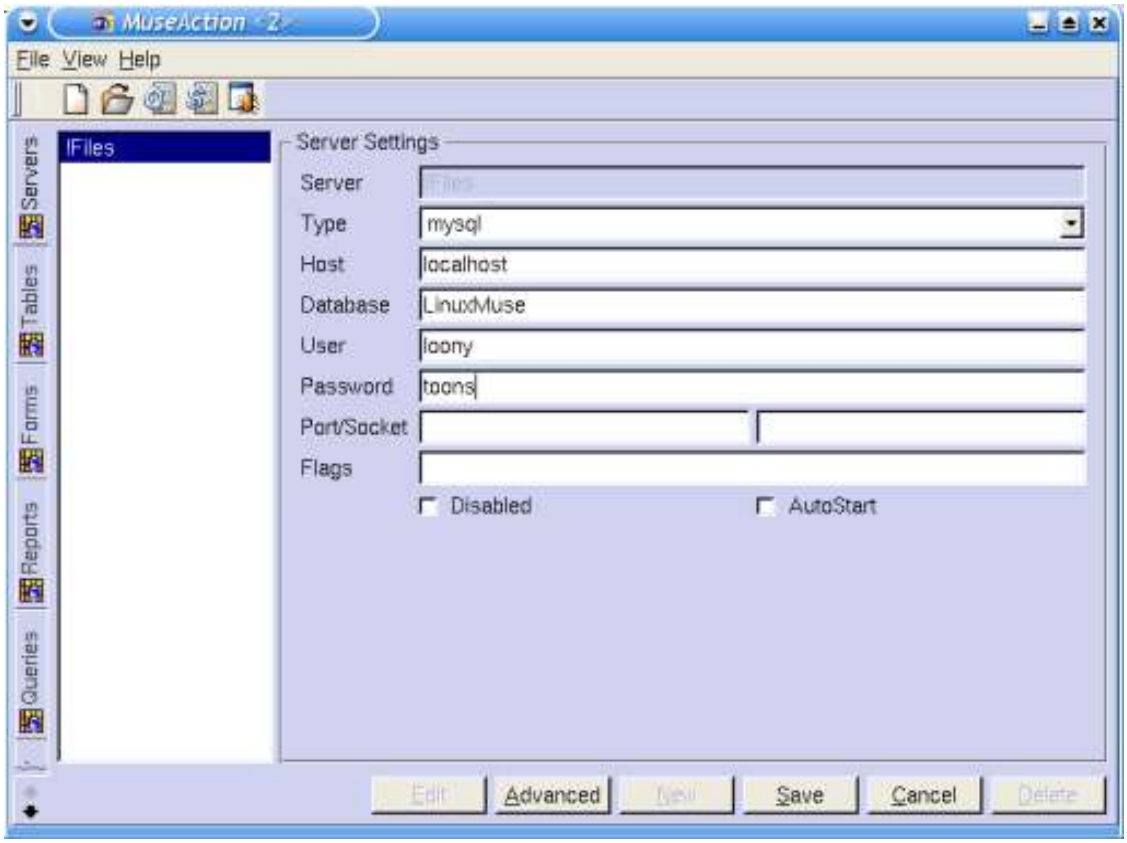

Fig. 20 Creación de una conexión a Base de datos en Rekall.

Rekall también puede diseñar y usar formularios e informes, construir consultas a base de datos, importar y exportar datos. El usuario puede crear componentes reusables y utilizarlos en formularios e informes, para reducir así, el tiempo de desarrollo de la aplicación. Este generador de informes cuenta con funciones tales como constructor de consultas, generador de formularios y asistentes gráficos para generar informes y lo que para Access es VBA (Visual Basic for Aplication), para Rekall lo es Python, es decir los guiones se escriben en este lenguaje.

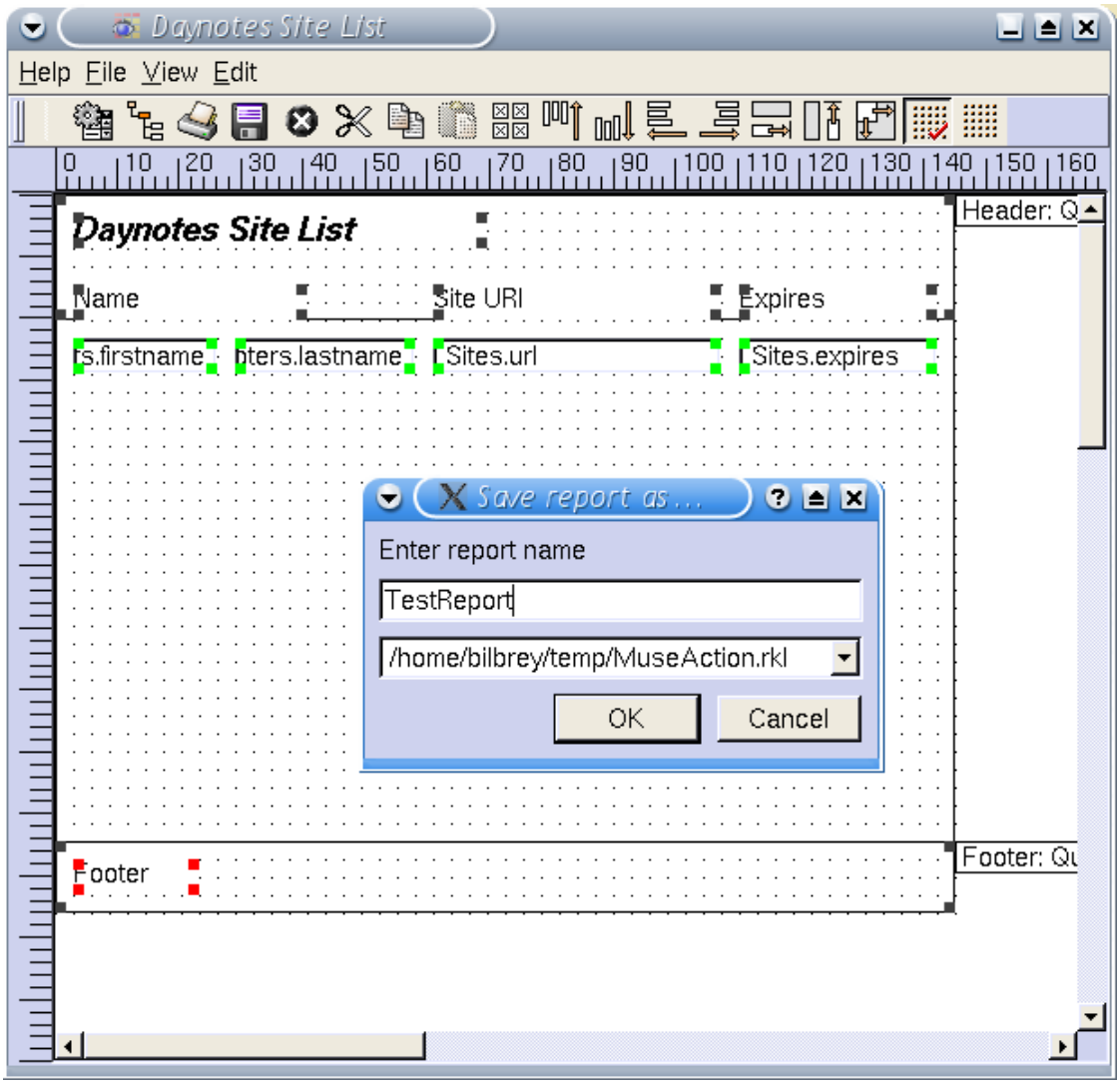

Fig. 21 Ambiente de diseño Rekall.

Rekall tiene un depurador Python integrado, al utilizar un lenguaje con sintaxis de alto nivel proporciona a usuarios la posibilidad de añadir guiones de instrucciones, para después hacer que se ejecuten cuando ocurran determinados eventos (por ejemplo, cuando el usuario cambia el valor de un control). Los guiones pueden ser asociados directamente con el evento, pero también se pueden almacenar en módulos para un uso más general.

Está escrito en C++ / QT, aunque en la página oficial de su creador se aclara que no solo se trabaja en versiones soportadas sobre QT, para hacerlo totalmente independiente de las librerías de KDE. Actualmente está disponible para GNU/Linux y Microsoft Windows y se está trabajando en una versión para Mac OS.

En estos momentos Rekall posee una licencia dual, manteniéndose por la comunidad la versión GPL y por sus propietarios la versión comercial, en este último caso se utiliza una "licencia de uso" que impide la distribución libre de modificaciones del código fuente. Cualquier modificación del código fuente debe ser autorizada por theKompany e incorporada por ésta en el producto.

Fig. 22 Ejemplo de informe generado con Rekall.

1.19.7 JasperReports.

JasperReports(JASPERREPORTS) es una herramienta de fuente abierta para la generación de informes escrita en Java, que puede mostrar contenido de calidad en ficheros PDF, HTML, XLS, CSV (formatos de hoja de calculo) y XML. Puede usarse en aplicaciones embebidas en Java, incluyendo J2EE (plataforma JAVA2 edición empresarial) o aplicaciones Web, para generar contenidos dinámicos.

Esta librería puede usarse en GNU/Linux y en Microsoft Windows, y se distribuye bajo licencia LGPL.

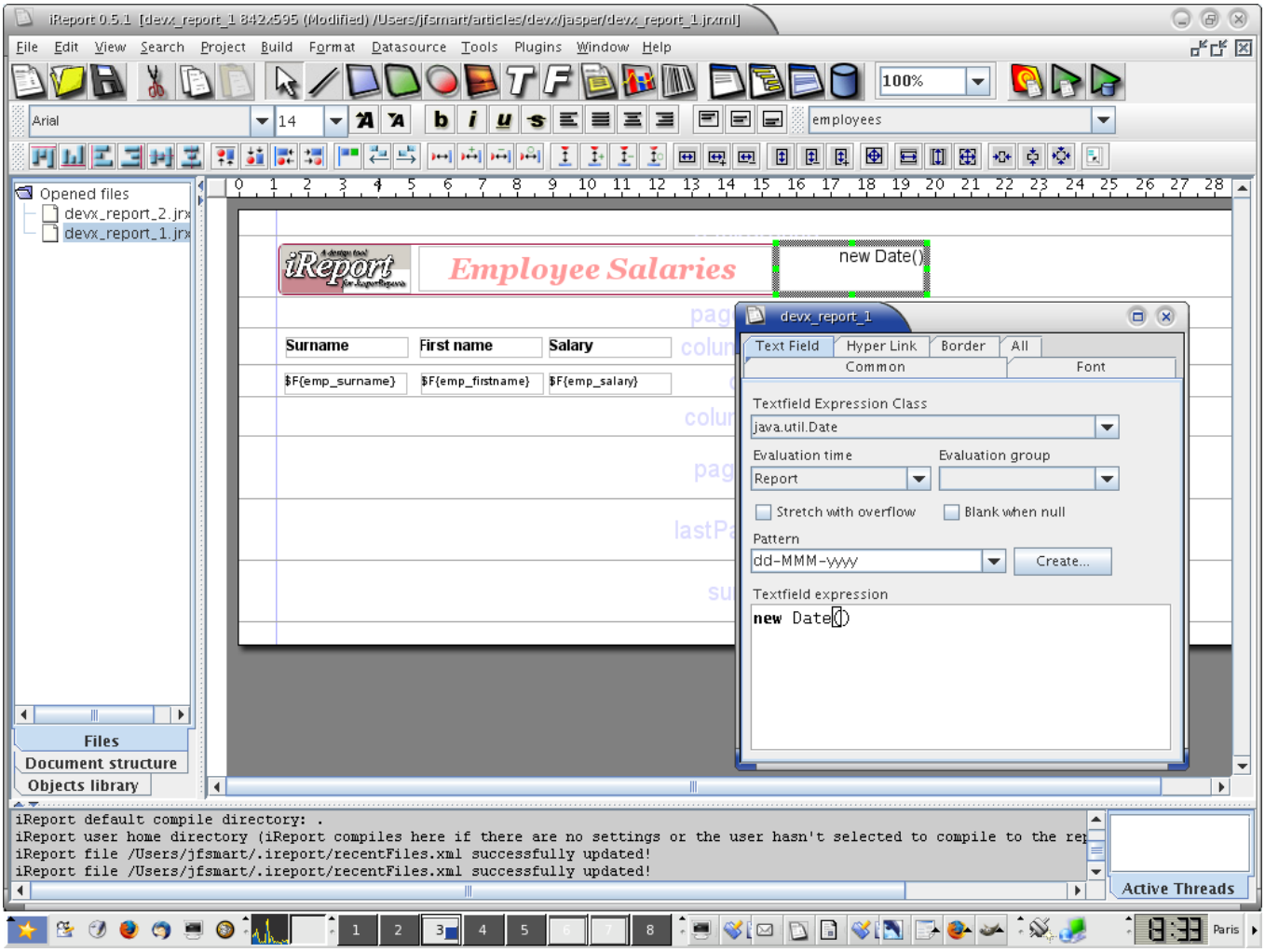

Fig. 23 Ventana de diseño de informe.

1.19.8 JfreeReport

JfreeReport(JFREE.ORG) es una librería para informes escrita en Java. Esta, lee datos de un TableModel y genera salidas formateadas que pueden incluir cabeceras, pies de página, agrupación, totales, medias, imágenes, etc. Los informes pueden ser previsualizados en pantalla o almacenados en PDF, XLS, HTML, XML o texto normal. JFreeDesigner es el editor de informes para JfreeReport.

Al igual que la anterior, al estar escrita en Java es multiplataforma, y también se distribuye bajo licencia LGPL.

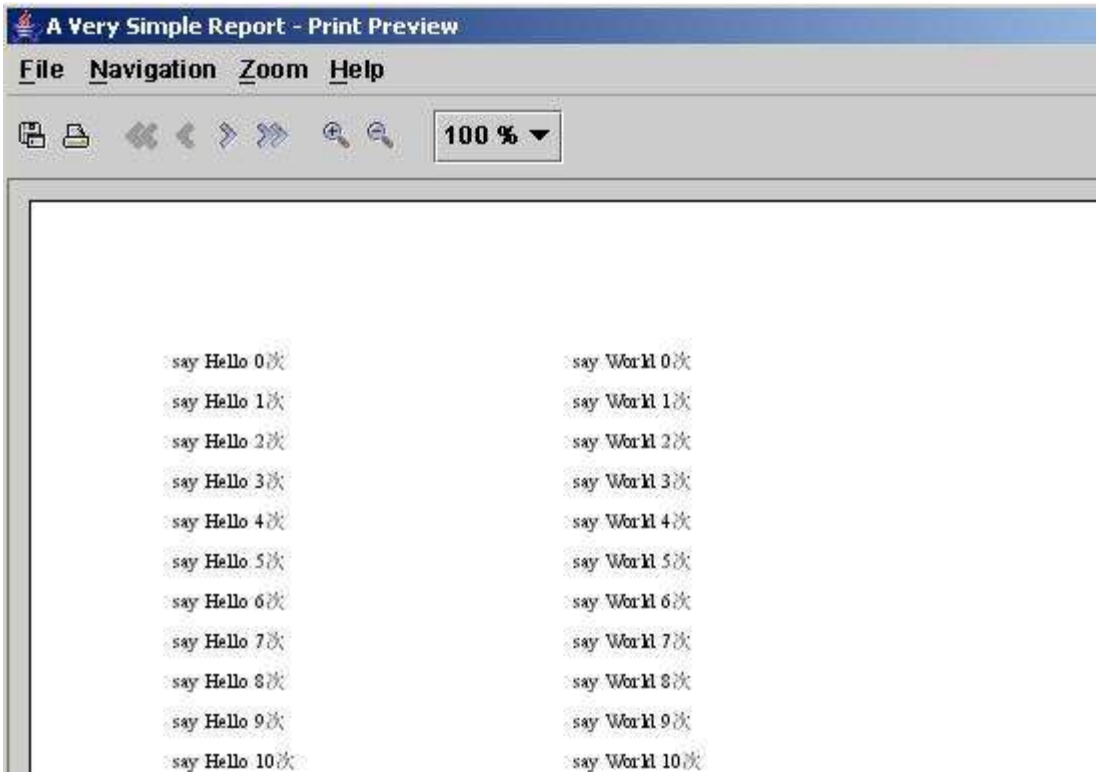

Fig. 24 Ejemplo de informe obtenido con JfreeReport.

1.19.9 DataVision

DataVision es una herramienta de informes para bases de datos similar a Crystal Reports. Los informes pueden ser diseñados usando una interfaz gráfica. Es un producto que disfruta de muchas de las características avanzadas de JasperReports como soporte de JDBC o la disponibilidad de herramientas gráficas para la generación de informes y que a su vez añade alguna característica especial como la posibilidad de exportar los informes a formato DocBook o LaTeX2e.

Su licencia es de tipo Apache 2.0, que es de software libre pero incompatible con la GPL porque tiene un requisito concreto que no tiene la GPL: contempla algunos casos, que la GPL no requiere, en los que puede rescindirse la licencia por problemas de patentes.

DataVision es una de las aplicaciones más usadas en el mundo Java por sus potencialidades como generador de informes, su amigable interfaz gráfica y su capacidad de ser ejecutado en diversos entornos.

DataVision está programado en Java y corre en diversas plataformas. Puede generar informes tomando los datos de bases de datos o archivos de texto cuyas columnas pueden ser separadas por cualquier carácter. Puede trabajar con cualquier base de datos que tenga un controlador JDBC como son Oracle, PostgreSQL, MySQL, Informix, hsqldb, Microsoft Acces y otras.

Las descripciones de los informes son almacenadas como archivos XML. Esto significa que no solo se puede usar la interfaz gráfica de DataVision sino que puede utilizarse cualquier editor de texto para editar los informes.

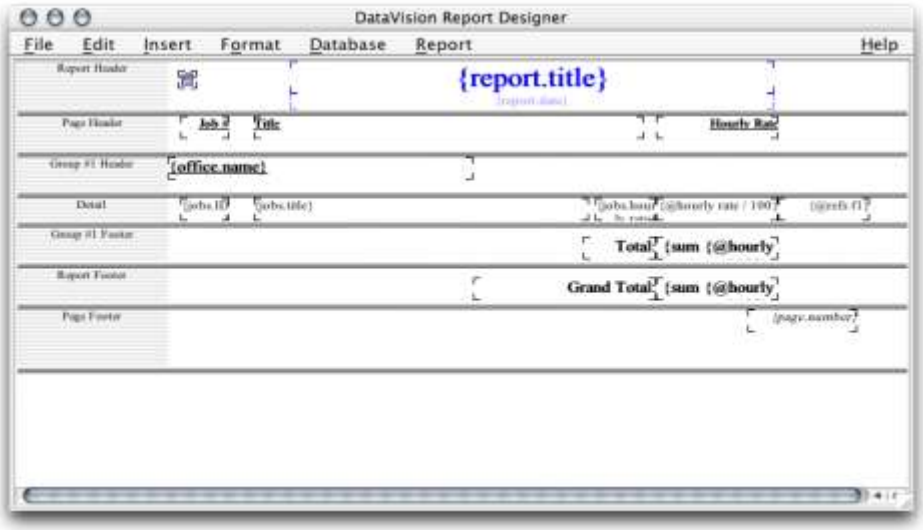

Fig. 25 Ventana de diseño de informe**.**

Como se puede apreciar en la figura, en la filosofía de DataVision, un informe tiene múltiples secciones. En la parte izquierda están los nombres de las secciones. Las largas áreas blancas son secciones que contienen campos del informe: columnas de la base de datos, campos de

texto, fórmulas, agregaciones, y campos especiales como el título del informe o el número de la página.

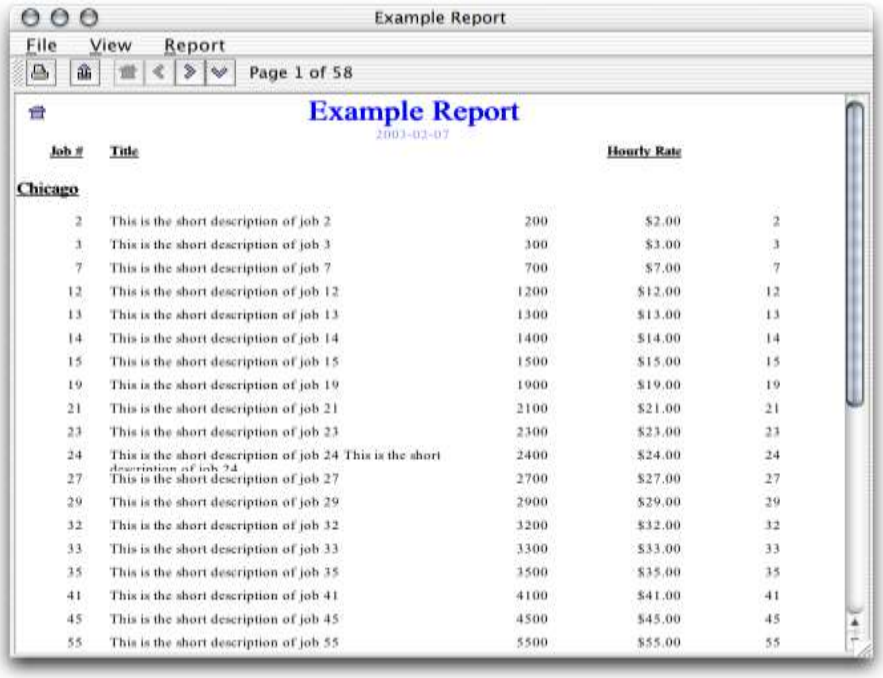

Fig. 26 Ejemplo de informe obtenido con DataVision**.**

Los informes pueden ser ejecutados, vistos e impresos desde la aplicación o exportados como HTML, XML, PDF, LaTex2e, DocBook. Los archivos de salida producidos por LaTex2e y DocBook pueden ser convertidos para producir PDF, texto, HTML, PostScript y más.

De forma general las características más relevantes de este generador de informes son:

- Corre donde quiera que Java corra: Microsoft Windows, GNU/Linux, Mac OS X, BSD, Solaris y otros.
- Trabaja con cualquier base de datos que tenga un manejador JDBC: MySQL, Oracle, PostgreSQL, Informix, hsqldb, Microsoft Acces, DB/2 y otras.
- Genera en el informe cabeceras y pies de páginas.
- Los informes tienen formatos: fuentes, colores, estilos de letras (negrita, itálica, subrayada), bordes de líneas, etc.
- Genera subinformes.
- Agregaciones (sum, min, max, count, average, stddev) por grupos y al finalizar el informe.

- Parámetros en tiempo de ejecución, le pregunta al usuario por los valores cuando el informe está corriendo; si está corriendo desde una línea de comando lee los valores de un fichero XML.
- Utiliza la cláusula SELECT de SQL.
- Oculta columnas y secciones enteras.
- Los informes pueden ser leídos de fuentes de datos. Actualmente, las fuentes de datos están definidas por base de datos y ficheros de textos.
- Permite imprimir informes desde la aplicación.
- Exporta los informes a HTML, XML, PDF, DocBook, Latex, etc.
- Configuración de la interfaz gráfica para diferentes idiomas.
- Informes almacenados en XML que pueden ser leídos fácilmente por las personas.
- DataVision es software libre.
- DataVision puede ser empotrado en otras aplicaciones.

1.19.10 OpenOffice 2 Writer

OpenOffice.org 2.0 Writer(OPENOFFICE.ORG) además de ser un procesador de textos, permite manejar y crear bases de datos sin tener conocimiento de SQL. HSQLDB (motor de base de datos relacional SQL escrito en Java), permite crear documentos a partir de bases de datos. Estos ficheros no requieren un gestor como MySQL.

Toda la información (definiciones de tablas, datos, consultas, formularios, informes) es almacenada en un fichero XML.

HSQLDB tiene un driver JDBC (Java Database Connectivity), el cual no es más que una interfaz de programación de aplicaciones, que permite a programas escritos en Java ejecutar comandos SQL.

OpenOffice2 puede usarse en GNU/Linux y en Microsoft Windows. Actualmente este proyecto está publicado bajo las licencias SISSL y LGPL.

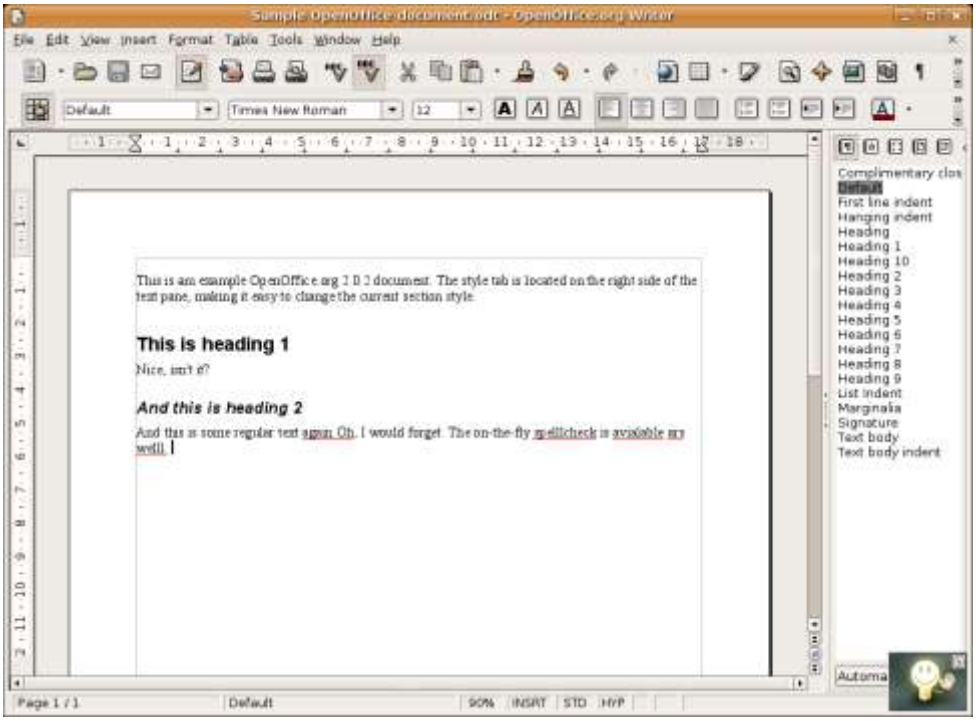

Fig. 27 Ventana de diseño de informe.

Anexo 2: Resumen de las principales características de varios generadores de informe.

Generadores		Plataforma	Formatos de salida	Diseñador Gráfico	Gestor de base de datos	Conexión a bases de datos	Orientado a	Lenguaje usado en su desarrollo	Licencia
Comerciales	Crystal Report	Linux, Windows	PDF, HTML, XML, XSL (Excel), DOC (Word)	SI	NO	SI	Usuario finales y Programadores		Es propietario
	Access 2003	Windows	HTML, XML, XSL	SI	SI	SI	Usuario finales		Es propietario
Software Libre	Report Manager	Linux, Windows	PDF, HTML, XSL	SI	NO	SI	Usuario finales y Programadores	Delphi/Kylix, Pascal	MPL (incluye permiso de uso aplicaciones G
	Kugar	Linux (KDE)	PostScript	SI	NO	SI	Usuario finales y Programadores		GPL
	Agata Report	Linux, Windows	PDF, HTML, CSV, XML, PostScript, texto plano	NO	NO	SI	Usuario finales y Programadores	PHP	GPL
	Open Report	Linux, Windows	PDF, HTML	NO	NO	NO		Python	GPL
	Rekall	Linux (KDE), Windows	PDF	SI	NO	SI			GPL
	Jasper Reports	Linux, Windows	PDF, HTML, XLS, CSV y XML	SI	NO	SI		Java	LGPL
	JfreeReport	Linux, Windows	PDF, HTML, XSL, XML, texto plano	NO	NO	NO		Java	LGPL
	OpenOffice2	Linux,	PDF, HTML	SI	NO	SI		Java	CISL y GPL
	NCReport	Linux, Windows	PDF	SI	NO	SI	Programadores	C++	GPL
	DataVision	Linux, Windows	HTML, XML, PDF	SI	NO	SI	Usuario finales y Programadores	Java	Apache 2.0

Software libre en el mundo

País	1.7.1999	1.7.2000	1.7.2001	1.7.2002	20.6.2003
Estados Unidos	162	169	256	278	297
Alemania	54	58	101	121	136
Reino Unido	34	34	55	63	75
Australia	23	26	41	49	52
Francia	11	11	24	44	51
Canadá	20	22	41	47	49
España	10	11	25	31	34
Japón	15	15	27	33	33
Italia	9	9	22	26	31
Países Bajos	14	14	27	29	29
Suecia	13	13	20	24	27

Fig. 28 Software libre en el mundo

Glosario de términos.

ActiveX: Lenguaje desarrollado por Microsoft para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma a través, normalmente, de navegadores WWW. Permite dar dinamismo a las páginas web.

Benchmark: Técnica utilizada para medir el rendimiento de un sistema o componente de un sistema, frecuentemente en comparación con algún parámetro de referencia.

BMP (*BitMaP***):** En los sistemas operativos Microsoft Windows, representan la sigla BitMaP, o sea mapa de bits. Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de pixeles.

CSV(en inglés *comma-separated values*): Son un tipo de documento sencillo para representar datos en forma de tabla

Diseño de informe: Apariencia final con la cual será generada el informe.

Framework: En desarrollo de software, es una estructura en la cual pueden ser desarrollados distintos tipos de proyectos software. Normalmente incluye programas de ayuda, librerías de código y lenguajes de programación.

Generador de informes o reportes: Herramienta que permite generar informes, también conocidos como informes a partir de datos primarios.

GIF (***G****raphics* ***I****nterchange* ***F****ormat*): Formato gráfico utilizado ampliamente en la World Wide Web, tanto para imágenes como para animaciones. Es un formato sin pérdida de calidad, siempre que partamos de imágenes de 256 colores o menos. Una imagen de alta calidad, como una imagen de color verdadero (profundidad de color de 24 bits o superior) debería reducir literalmente el número de colores mostrados para adaptarla a este formato, y por lo tanto existiría una pérdida de calidad. GIF llegó a ser muy popular porque podía usar el algoritmo de compresión LZW (Lempel Ziv Welch) para realizar la compresión de la imagen, que era más eficiente que el algoritmo Run-Lenght Encoding (RLE) que usaban formatos como PCX y MacPaint. Por lo tanto, imágenes de gran tamaño podían ser descargadas en un razonable periodo de tiempo, incluso con módems muy lentos.

GNOME: GNU **N**etwork **O**bject **M**odel **E**nvironment es un entorno de escritorio basado en las librerías GTK diseñadas para GIMP para sistemas operativos de tipo Unix bajo tecnología X Window. Al igual que KDE, permite la interacción entre programas. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GPL: Son las siglas de General Public License, Licencia Pública General, definida por la Fundación para el Software Libre (FSF) para proteger los derechos de copia del software libre.

GUI (Graphical User Interfaces): Componente de una aplicación informática que el usuario visualiza y a través de la cual opera con ella. Está formada por ventanas, botones, menús e iconos, entre otros elementos.

HTML (***H****yper****T****ext* ***M****arkup* ***L****anguage*, lenguaje de marcas hipertextuales): Lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

Informe o reporte: Documento contentivo de información personalizada y útil para el destinatario del mismo.

KDE (**K D**esktop **E**nvironment): Es un entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y, en particular, GNU/Linux.

LGPL: Son las siglas de Lesser General Public License o Library General Public License. Esta licencia permite el enlace dinámico de aplicaciones libres a aplicaciones no libres.

Modbus: Es un protocolo de comunicación situado en el nivel 7 del Modelo de referencia de interconexión de sistemas abiertos (OSI, en ingles, Open System Interconection), basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs, en ingles Programmable logic controller).

Multiplataforma: Es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

PDF (*P*ortable *D*ocument *F*ormat, Formato de Documento Portátil): Formato de almacenamiento de documentos multiplataforma (Microsoft Windows, Unix, Mac) desarrollado por la empresa Adobe System. Especialmente ideado para documentos susceptibles de ser impresos.

Plataforma: Base, elemento de apoyo

PLC: Controlador Lógico Programable.

PV (*P*resent *V*alue): Valor que tiene la variable en el momento actual.

Informe persistente: Informe que una vez generado debe almacenarse en medios de almacenamiento de forma permanente en el tiempo.

Informe temporal: Informe generado con carácter temporal, razón por la cual no es necesario su almacenamiento permanente en el tiempo.

RTU: Remote Terminal Unit (Unidad de Terminal Remota) La RTU se conecta al equipo físicamente y puede leer el estado de los datos digital o medidas de datos análogos y envía comandos digitales de salida o puntos de ajuste análogos.

License. Bajo esta licencia los desarrolladores pueden modificar y distribuir código fuente libremente.

TCP (Transmission Control Protocol/Internet Protocol): Conjunto de protocolos de bajo nivel (IP, TCP, UDP, ICP, RARP, etc) que permiten el funcionamiento de Internet.

WYSIWYG es un acrónimo de **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et (en inglés, "lo que vez es lo que obtienes"): Se aplica a los procesadores de texto y otros editores con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso. En el caso de editores de HTML este concepto se aplica a los que permiten escribir la página sobre una vista preliminar similar a la de un procesador de textos, ocupándose en este caso el programa de generar el código fuente en HTML.

XML (e***X****tensible* ***M****arkup* ***L****anguage*, 'lenguaje de marcas extensible'): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG y MathML.