Universidad de las Ciencias Informáticas

Facultad 5

Implementación del Subsistema de Generación de Informes Selux.

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

Autor: Lisandra González Serrano.

Tutores: Ing. Bernardo Zaragoza Hijuelos.

Ing. Raudi Agdel Bacallao Sanchez.

Ciudad de La Habana, julio del 2010

“Año 52 de la Revolución”

Datos de Contacto

Tutor(es):

Ing. Bernardo Zaragoza Hijuelos

Graduado de Ingeniero en Ciencias Informáticas y profesor de la Universidad de Ciencias Informáticas (UCI), con 4 años de experiencia en el desarrollo de software.

e-mail: bzaragoza@uci.cu

Ing. Raudi Agdel Bacallao Sánchez

Graduado de Ingeniero en Ciencias Informáticas y profesor de la Universidad de Ciencias Informáticas (UCI), con 4 años de experiencia en el desarrollo de software.

e-mail: rabacallao@uci.cu

DECLARACIÓN DE AUTORÍA

Por este medio se declara que soy la única autora 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 _____ del 2010.

_____ _____

Firma del Tutor

(Raudi Agdel Bacallao Sanchez)

Firma del Tutor

(Bernardo Zaragoza Hijuelos)

_____

Firma del Autor

(Lisandra González Serrano)

Agradecimientos

En primer lugar quiero agradecer el desarrollo de este trabajo a mi mamá y mi papá, por la preocupación y esfuerzos durante todos estos años, en los que me apoyaron para llegar hasta aquí. Por sus llamadas constantes en espera de noticias y por la confianza y el cariño de siempre.

A mi familia por estar todos pendientes siempre de mí y por contribuir de una forma u otra en mi formación personal.

A mi novio, por su cariño y comprensión durante todo este tiempo y por estar a cada instante a mi lado apoyándome y dándome fuerzas para seguir adelante sin importar las dificultades.

A mis dos tutores por la confianza depositada en mí, especialmente a Raudi, por toda su ayuda y preocupación constante.

A mis amigos de ahora, los de antes y los de siempre, por los buenos y malos momentos que hemos compartido, y por estar siempre a mi lado cuando los he necesitado.

A mis compañeros de la universidad por compartir estos 5 años juntos.

A mis profesores por toda la enseñanza transmitida, que hoy se materializa en el desarrollo de este trabajo.

A la Universidad de Ciencias Informáticas, por permitirme crecer profesionalmente y sobre todo como ser humano.

Lisi

Dedicatoria

A mis padres, Nereyda y Evelio.

A mi novio Alberto.

A toda mi familia.

Lisi

Resumen

Como resultado del convenio entre Cuba y la República Bolivariana de Venezuela surge un proyecto de software nombrado SCADA Guardián del Alba, el cual por sus siglas en inglés (Supervisory Control and Data Adquisition) se trata de un sistema de supervisión, control y adquisición de datos. Para este sistema SCADA fue necesario desarrollar una aplicación para la gestión y configuración de reportes; el cual, a raíz del uso y experiencia de trabajo con el mismo, ha demostrado presentar una serie de problemas que han ido dificultando su uso.

El presente trabajo de diploma se centra en la obtención de una nueva versión del sistema de generación de reportes del SCADA Guardián del Alba, con una arquitectura de software mejorada y que potencie atributos de mayor calidad, obteniendo como resultado un software más robusto.

Para llevar a cabo este objetivo, se realiza un estudio minucioso de los principales sistemas de generación de informes. Además, se efectúa un análisis detallado de las principales tecnologías de desarrollo posibles a utilizar y se hace una propuesta de las más importantes. Se presentan además, las pruebas realizadas al código escrito con el objetivo de que el software cumpla con los principios de calidad pertinentes y tenga un buen estado desde el punto de vista funcional.

Índice de Contenido

Introducción

La información; como recurso intangible, es la mayor fuente de conocimiento y es de dónde a través de los años el hombre dentro de sus generaciones y diversidad de culturas se ha nutrido para lograr el desarrollo que se tiene en la actualidad. Este recurso tan importante, tiene la capacidad de multiplicarse cuando es analizada bajo ciertos parámetros, los cuales la convierten en un recurso extensible de sí mismo.

Tradicionalmente, la información se manejaba a mano haciendo los documentos en papel, y aunque el uso de los mismos hoy en día aún es vigente en algunos lugares, el volumen de datos que se hacía necesario manipular en la mayoría de los casos era considerablemente grande, haciendo este trabajo engorroso y muy complicado de manejar.

Con la aparición de nuevas tecnologías se incrementa la cantidad de datos que es necesario analizar para brindar información útil de los procesos de negocio de las empresas. La informática abrió una puerta que nos guió hacia la búsqueda de soluciones más factibles y mucho más eficientes para la realización de esta labor, si bien es cierto que los avances de las tecnologías facilitan el proceso de recopilación y análisis de los datos manejados todavía los recursos de la información más importantes siguen estando perdidos en un mar de números.

El campo empresarial es uno de los puntos donde el análisis de la información es vital para el desarrollo y sostén de una institución. Debido a esto, en la actualidad; la mayoría de las empresas se unen a la idea de la implantación y fomento del uso de las tecnologías para realizar cada uno de los diversos procesos de la empresa y analizar los datos generados en esos procesos. Las empresas se basan en información oportuna y precisa, para gestionar las operaciones de negocios, identificar los riesgos, mejorar la productividad, y aumentar la rentabilidad.

Con este fin, existen en el mundo diferentes herramientas que viabilizan el proceso de generación de reportes. Sin embargo, para nuestro país constituye una real limitación la adquisición del software legal necesario para el desarrollo e informatización de la sociedad cubana.

Introducción.

Por esta razón, la industria cubana del software impone la búsqueda de nuevas alternativas en la producción del software empresarial, entre ellas el uso de estándares informáticos abiertos en la creación de soluciones propias.

Dentro de la Universidad de las Ciencias Informáticas, específicamente dentro del Centro de Informática Industrial (CEDIN), se ha venido desarrollando hace algunos años un sistema de Supervisión, Control y Adquisición de Datos (SCADA), el cual es una aplicación diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autónomas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. (1) Debido a que una de las principales funcionalidades de un SCADA es su capacidad de proveer toda la información que se genera en el proceso productivo a diversos usuarios, surge la necesidad de desarrollar un módulo para la configuración y generación de informes para el SCADA Guardián del Alba, usado por la empresa PDVSA en la hermana República de Venezuela, sin embargo, hoy se pretende hacer uso del mismo en otras empresas y por tal motivo se requiere de cambios en dicho sistema.

Este sistema se desarrolló como parte del módulo HMI (Human Machine Interface) del SCADA y se implementó usando la biblioteca gráfica GTK cuyas siglas provienen de GIMP Toolkit, la cual es una biblioteca que permite crear interfaces gráficas de usuario, y que en su momento fue la solución más adecuada.

En el constante avance y el aumento de las necesidades de los clientes se hizo necesaria una reevaluación de la tecnología. El hecho de que el generador de reportes se encontraba empotrado dentro del módulo HMI; traía consigo un gran consumo de recursos, lo que hacía que el sistema fuese un poco lento y además, se dificultaba la portabilidad, ya que el sistema dependía de un grupo de componentes que eran propios del HMI. Por otro lado, al estar implementado sobre la biblioteca gráfica GTK, dificultaba el soporte, la corrección de errores, así como la incorporación de nuevos recursos para el desarrollo. En la búsqueda de soluciones se llegó a la conclusión de que GTK no podía ser la tecnología usada en la nueva versión ya que las API de GTK son muy complejas y las herramientas de diseño son muy difíciles de manejar, añadiéndole a esto que la documentación de la misma no es abundante.

Debido a la existencia de estas dificultades se presenta la siguiente **situación problémica**:

Introducción.

El sistema de generación de reportes del SCADA, es un sistema que al estar vinculado al módulo HMI consume gran cantidad de recursos y dificulta la portabilidad. Además, al estar implementado sobre la biblioteca gráfica GTK dificulta el soporte y la corrección de errores, así como la realización de mejoras en el diseño del mismo.

Ante esta situación se definió el siguiente **Problema Científico:**

¿Cómo desarrollar una nueva versión del sistema de generación de reportes que mitigue las dificultades del sistema actual y satisfaga todas las necesidades de los usuarios?

Objeto de Estudio:

Los sistemas de generación de reportes.

Campo de Acción:

Los sistemas de generación de reportes sobre herramientas libres.

Como respuesta al problema planteado se determinó como **Objetivo General**:

Disponer de una nueva versión del generador de informes, que sea más rápido, portable, mantenible y con un escenario de visualización más robusto.

Idea a Defender:

Migrando el sistema de generación de reportes, a una biblioteca gráfica más apropiada, es posible lograr mayor rapidez en el sistema, y facilitar la portabilidad, el soporte y corrección de errores del mismo y sobre todo se puede lograr una herramienta de diseño más eficaz.

Para ello se propusieron las siguientes **Tareas Investigativas:**

1. Revisión de Bibliografías para el estudio y análisis de los sistemas de generación de reportes actuales.

2. Investigación sobre las diferentes bibliotecas y herramientas sobre plataformas libres para la selección de las mismas en el desarrollo del sistema.

3. Identificación y fundamentación de patrones arquitectónicos para la implementación de la solución.

4. Implementación de la solución basada en los requerimientos y análisis de la versión anterior y algunas modificaciones del diseño existente para lograr una aplicación robusta.

5. Realización de pruebas a la implementación realizada para corregir posibles errores.

Para dar cumplimiento a las tareas se emplearon los siguientes **métodos científicos** en el proceso investigativo:

Métodos teóricos:

- **Analítico - Síntesis:** Con el objetivo de analizar los documentos ya existentes que abordan el tema de los sistemas de generación de informes dinámicos para la extracción de los elementos más importantes acerca de las funcionalidades y características de dichos sistemas.
- **Histórico - Lógico:** Para determinar las tendencias actuales de los sistemas de gestión de reportes, así como las ventajas y desventajas que tienen cada uno de ellos.
- **Inductivo - Deductivo:** En el estudio y comparación de las distintas librerías gráficas, así como de las ventajas y desventajas, que justifiquen el uso de la librería seleccionada.

Métodos Empíricos:

- **Observación:** Para realizar una valoración a partir de la percepción en la etapa exploratoria de la realidad estudiada para determinar cómo ocurre realmente el proceso de gestión de reportes y cuáles son los principales elementos en los que se debe trabajar en la investigación.
- **Experimento:** A partir del estudio del sistema de generación de reportes existente adaptar las características y funcionalidades del mismo a las condiciones especificadas en la situación problémica.

Este documento está estructurado en tres capítulos, que recogen todo el proceso de desarrollo de este trabajo:

- **CAPÍTULO 1:** Fundamentación teórica y evaluación de las tecnologías.
- **CAPÍTULO 2:** Descripción del Sistema y Propuesta de Solución.
- **CAPÍTULO 3:** Implementación y Validación de la Solución

Capítulo 1
Fundamentación Teórica y Evaluación de las tecnologías.

En el presente capítulo se introducen las principales características y conceptos asociados a los generadores de reportes. Además, se exponen los principales sistemas de visualización utilizados en el mundo, la descripción de las principales tecnologías que se emplean en la construcción de estos sistemas y finalmente se explica la selección de las librerías y lenguajes a utilizar, así como las alternativas de desarrollo.

1.1 Conceptos fundamentales

Reporte

Un reporte es un documento, que nos presenta de manera estructurada y/o resumida, datos relevantes almacenados o generados en el proceso de creación del mismo; de manera que puedan ser analizados o entendidos con gran facilidad.

Reporteador o Sistema de Generación de Reportes.

Un Reporteador o Sistema de Generación de Reportes es un programa que permite crear informes sobre diseños en una amplia variedad de formatos que no son rutinariamente producidos por un sistema de información. Extraen datos de los archivos o de las bases de datos y crean reportes proporcionando más control sobre la información que se presenta, ya que pueden manejar además, datos de cálculos y lógica compleja antes de darle salida al reporte final. (2)

Uno de los aspectos más importantes de este tipo de aplicaciones es su capacidad de convertir los datos complejos en texto simple, atractivo y fácil de comprender por todo tipo de usuario.

1.2 Generalidades de los generadores de reporte.

De forma general un reporteador se compone de dos elementos básicos, el diseñador de reportes y el motor de generación.

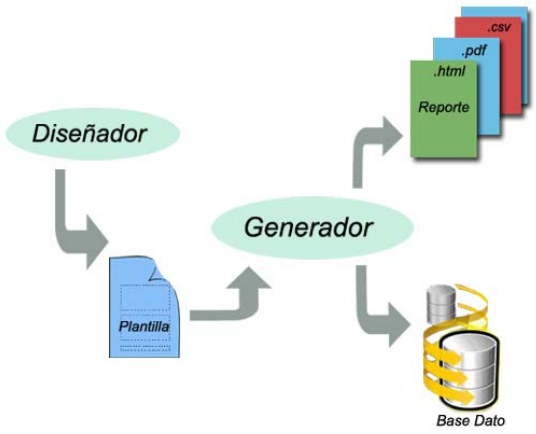

Fig. 1: Esquema de Funcionamiento de un sistema de generación de Reportes

Diseñador: Es una herramienta que permite diseñar y configurar visualmente la apariencia de un reporte mediante la creación de plantillas. En las plantillas se modelan y encierran las características con las cuales el usuario quiere que se genere el reporte. De esta manera una vez que se insertan todos los componentes deseados en tiempo de diseño, en las distintas secciones, persistirá en la plantilla estas características y componentes para que a la hora de generar el reporte este se vea con la apariencia deseada y los datos volcados en su interior. (3)

Generador: Es una herramienta de creación de informes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, CSV, etc. Su propósito principal es obtener los datos necesarios de las fuentes de datos y elaborar el informe final de forma ordenada con la apariencia previamente configurada en plantillas.

El subsistema de generación es una parte primordial de cualquier reporteador, ya que es el responsable de generar lo que los usuarios necesitan observar para poder entender la información mostrada en el reporte de forma sencilla y comprensible. No todos los sistemas de gestión de reportes necesitan contar con un subsistema de diseño, sin embargo, todos sí requieren del subsistema de generación, ya que como se ha planteado con anterioridad este es quien mostrará el resultado final y bien puede funcionar partiendo de un diseño estándar almacenado previamente en alguna plantilla.

1.3 Estado de Arte de los generadores de reporte

En la actualidad existen diversos sistemas de generación de reportes a nivel mundial, una gran parte de estos; solo permiten generar un tipo de reporte determinado en tiempo de desarrollo lo cual obliga al usuario a redefinir los requisitos cada vez que necesite añadir uno nuevo. Otro gran número de estos sistemas permiten la generación de informes de forma dinámica sin embargo la mayoría tiene un elevado costo de adquisición y son por lo general, orientadas al programador y no al usuario.

A continuación se analizarán algunas de las herramientas más utilizadas a nivel mundial.

1.3.1 Crystal Reports

Crystal Reports es una aplicación de inteligencia empresarial, utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos. Se convirtió en el escritor de informes por defecto cuando Microsoft lo liberó con Visual Basic.

Crystal incorpora la posibilidad de crear contenido interactivo con calidad de presentación al entorno de Windows. Además, puede crear informes complejos y profesionales en un programa basado en GUI. Después puede conectar el informe a casi todos los orígenes de base de datos, así como a datos proxy, como un conjunto de resultados (por ejemplo, un ADO.NET DataSet). Los asistentes del diseñador de GUI le permiten establecer fácilmente los criterios de formato, agrupamiento y gráficos, etc.

Puede almacenar el informe en una aplicación Web o para Windows, con uno de los controles de visores de Crystal Reports para Visual Studio 2005. La presentación de informes, tanto en clientes Windows como en HTML 3.2 ó 4.0, es muy interactiva y proporciona funciones como la profundización en gráficos, la exploración de informes y la búsqueda de texto.

Los informes tienen un alto grado de precisión en su diseño y gráficos dinámicos que brindan una gran cantidad de información útil. Además, pueden ser entregados vía web, por correo, en cualquier formato de Microsoft Office, Adobe PDF o embebidos en aplicaciones mayores de gestión. Las tres funciones que constituyen la base de la arquitectura de Crystal Reports son: Crystal Reports Designer incrustado, para el diseño de informes; Controles del visor de informes y Modelos de objetos (Microsoft Corporation).

Una de las desventajas es que sólo utiliza una jerarquía para agrupar los datos, o sea, clasifica los datos según la jerarquía otorgada y así define la estructura de todo el informe. Como consecuencia, si se desea generar reportes con estructura compleja, el usuario está obligado a crear varios subreportes. Además, elabora los reportes en tiempo de diseño ya que está dirigida más bien al programador y no al usuario.

1.3.2 JasperReport

Es el motor de reportes más usado actualmente en el mundo del software libre. Está enteramente escrito en java y puede ser usado en una gran variedad de aplicaciones compatibles con java, incluyendo J2EE o aplicaciones WEB, para generar contenido dinámico. Posee una amplia variedad de formatos de salida y exportación así como una gran comunidad mundial que mantiene y desarrolla la librería. Genera informes para formatos de impresión predeterminados existentes o para reportes continuos a ser visualizados en la web, los reportes pueden ser exportados a formatos como: PDF, XML, HTML, CSV, XLS, RTF, TXT. Su propósito principal es ayudar a crear páginas listas para imprimir documentos de una manera simple y flexible. Soporta a la misma vez varios orígenes de datos incluso aunque sean de tipos distintos.

Es posible pasar parámetros desde una aplicación a JasperReports, esto es muy simple de implementar y brinda una herramienta muy poderosa ya que permite clasificar, restringir o mejorar los datos que son enviados al usuario basándose en las condiciones de tiempo de ejecución. Tiene iReport como herramienta de acompañamiento para diseñar informes.

1.3.3 JasperReport

ActiveReports para .Net 3.0 toma las mundialmente premiadas capacidades de ActiveReports llevándolas al más alto nivel con mejoras significativas que proveen la habilidad de diseñar, crear y desplegar aplicaciones de reportes de forma fácil y rápida.

Características principales:

- Está diseñado teniendo en cuenta a los desarrolladores y sus diversas necesidades. Soporta el uso de componentes .NET en tiempo de diseño
- Escrito completamente en Visual C# y provee una completa integración con Visual Studio .NET.
- Asistente para la conversión de reportes importados desde Microsoft Access.
- Incluye filtros para exportar a los más populares formatos tales como Adobe PDF, Microsoft Excel,

RTF, HTML, texto plano e imágenes TIFF, tanto para aplicaciones de escritorio como para las basadas en la web.

- Diseñador de reportes orientado a usuarios finales, permitiendo su inclusión en las aplicaciones con el fin de que los propios usuarios diseñen y modifiquen sus reportes.

1.3.4 Agata Report

Agata Report es un generador de reportes multi-plataforma, una herramienta de consulta y generación de gráficos como el Crystal Reports que se conecta a varias bases de datos, como PostgreSQL, MySQL, Oracle, DB2, MS-SQL, Informix, InterBase, Sybase, o Frontbase y permite exportar los reportes en formatos como PostScript, TXT, HTML, XML, PDF o CSV (StarCalc, Excel). Permite crear documentos, como cartas y conjugar dinámicamente con los datos provenientes del reporte, así como crear etiquetas de direccionamiento y hasta generar un diagrama ER completo a partir de su banco de datos.

Agata basa su programación en PHP-GTK que es la fusión del lenguaje de script PHP y la librería de objetos GTK+. Este generador de informes provee de una API a programadores para poder manipular la creación, modificación y ejecución de reportes, la misma está 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.

1.3.5 DataVision

DataVision es una herramienta de reportes para bases de datos, similar a Crystal Reports. Los reportes 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.

Su licencia es de tipo Apache 2.0, que es de software libre pero incompatible con la GPL.

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 trabajar con cualquier base de datos que tenga un controlador JDBC como son Oracle, PostgreSQL, MySQL, Informix, Microsoft Acces y otras.

Las descripciones de los reportes 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 reportes.

Los reportes pueden ser ejecutados, vistos e impresos desde la aplicación o exportados como HTML, XML, PDF, DocBook.

1.4 Plataforma de Desarrollo

1.4.1 Software Libre

Debido al constante avance tecnológico que enfrenta la sociedad a nivel mundial; dentro de la industria de software ha tomado un gran impulso lo que se conoce como desarrollo de Software Libre. El software libre es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

El sistema de software libre es sin duda alguna la solución para muchos problemas en el mundo empresarial, por tal motivo el desarrollo de un sistema de generación de informes 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 que se adapte a nuestras necesidades.

1.5 Lenguajes y Tecnologías

1.5.1 Bibliotecas para el Desarrollo de Interfaces Gráficas

1.5.1.1 Qt

Qt es un amplio marco de desarrollo que incluye clases, librerías y herramientas para la producción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas incluyendo los sistemas Unix (Linux, MacOS X, Solaris, etc) o incluso toda la familia de Windows. Con Qt se pueden desarrollar ricas aplicaciones gráficas, incluye soporte de nuevas tecnologías como OpenGL, XML, Bases de Datos, programación para redes y mucho más. Proporciona la mayoría de las entidades gráficas empleadas en una aplicación KDE: menús, botones, regletas, etc.

La interfaz de programación de Qt está totalmente orientada a objetos; añade a C++ sus propias mejoras tales como:

- Un mecanismo de comunicación entre objetos llamado “señales y ranuras” (“signals and slots”).
- Las propiedades de cada objeto se pueden diseñar y consultar.
- Potentes eventos y filtros de eventos.
- Cadenas contextuales de traducción para una mejor internacionalización.
- Un sofisticado soporte de temporizadores que hacen posible integrar elegantemente muchas tareas en una GUI orientada a eventos.
- Un árbol jerárquico de objetos ordenados de un modo natural y con una fácil interpretación.
- Punteros seguros,”QGuardedPtr”, que son automáticamente puestos a “null” cuando el objeto referenciado es destruido, en contraposición con los punteros normales de C++ que no cambian cuando sus objetos son destruidos.

Qt tiene ventajas sobre otras herramientas que proveen un recubrimiento para C++, ya que, al estar implementada directamente en C++, no tiene ninguna dificultad en adaptar los Widgets, la calidad en tiempo de ejecución es más alta, ya que no necesita hacer uso de referencias innecesarias y además, no depende de ninguna otra biblioteca, aparte de sí misma.

Las aplicaciones basadas en Qt tienen una buena respuesta y un uso aceptable de la memoria. El desarrollo con Qt es muy apropiado para proyectos software de larga escala, tanto comerciales como de libre distribución. Por ejemplo, el escritorio KDE, un conocido entorno de escritorio para Linux y FreeBSD, ha sido desarrollado con Qt o bien el paquete de KOffice, una alternativa para el MS Office, también ha sido desarrollado con Qt.

Qt, posee una doble licencia, una GPL para proyectos de libre distribución y otra comercial para aplicaciones con ánimo de lucro, en cuyo caso habría que justificar ante Trolltech S.A. que es la empresa de software con los derechos sobre Qt.

Las principales herramientas que la componen:

- Qt Designer: Permite crear formularios visualmente.

- Qt Assistant: Permite acceso rápido a la documentación.
- QT Demo: Permite acceso rápido gran variedad de ejemplos.
- Qt Linguist: Facilita la traducción rápida de programas.
- Qmake: Simplifica el proceso de construcción de proyectos en las diferentes plataformas soportadas.

1.5.1.2 GTK

GTK+ es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros.

GTK+ está escrito en C y da uso a la teoría de Orientación a Objetos (utilizando la idea de las clases y funciones respuestas, las también llamadas *callback*).Tiene recubrimientos para muchos otros lenguajes, incluyendo C++, Guile, Perl, Python, TOM, Ada95, Objective C, Free Pascal, y Eiffel.

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+ se basa en varias bibliotecas del equipo de GTK+ y de GNOME:

- GLib. Biblioteca de bajo nivel, estructura, básica de GTK+ y GNOME. Proporciona manejo de estructura de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.
- GTK. Biblioteca que realmente contiene los objetos y funciones para crear la interfaz de usuario. Maneja widgets como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.
- GDK. Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.
- ATK. Biblioteca para crear interfaces con características de una gran accesibilidad. Pueden usarse utilerías como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado o mouse.

- Pango. Biblioteca para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+2.

Licenciado bajo los términos de LGPL, GTK+ es software libre y es parte del proyecto GNU, sin embargo GTK tiene como desventaja que su documentación es muy escasa y sus herramientas de diseño son muy difíciles de manejar.

1.5.1.3 wxWidgets

Las wxWidgets son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste.

Las wxWidgets proporcionan una interfaz gráfica basada en las bibliotecas ya existentes en el sistema (nativas), con lo que se integran de forma óptima y resultan muy portables entre distintos sistemas operativos. Están disponibles para Windows, MacOS, GTK+, Motif, OpenVMS y OS/2.

También pueden ser utilizadas desde otros lenguajes de programación, aparte del C++: Java, Javascript, Perl, Python, Smalltalk, Ruby .

Cuenta con una parte denominada wxBase que incluye clases como wxString, clases para el manejo de archivos y directorios de manera independiente del sistema, funcionalidades como: gráficos 2D, 3D con OpenGL, Bases de Datos (ODBC), Redes, Impresión, Hilos, visión e impresión del HTML, un sistema de archivos virtual y cuenta con algunos IDEs.

Entre sus principales desventajas se pueden citar:

- El uso de los componentes gráficos nativos hace que sea más probable que el mismo código se comporte de diferente forma al pasar de una plataforma a otra y también hace que sea más probable que se obtengan errores específicos de la plataforma.
- El diseño orientado a objetos no es bueno. Abusa de la utilización de macros.

- Depuración tediosa.
- Debido a su antigüedad, no soporta características modernas como pude ser el manejo de excepciones de la STL (Standard Template Library).

1.5.2 Bibliotecas para la generación de gráficas

1.5.2.1 KDChart.

KDChart es una herramienta para crear gráficos, está desarrollada sobre Qt y es el componente Qt más poderoso de su tipo. Además de tener todas las características estándares, permite al desarrollador diseñar y gestionar un gran número de ejes y ofrece sofisticadas formas de personalización, ya que no solo permite una detallada y precisa configuración del diseño del gráfico, sino que también es posible complementar el diseño mediante la adición de cajas de texto enriquecido y / o marcos para puntos de datos o posiciones aleatorias. KDChart es fácilmente escalable mediante el ajuste automático de los tamaños de fuente y el diseño, cuando se cambia el tamaño del gráfico. Todo esto hace posible crear rápida y eficientemente programas de fácil uso, que ofrecen un alto nivel de funcionalidad. Un ejemplo de su uso, es la actual versión de la suite de productividad KOffice que utiliza esta librería para crear sus gráficos.

1.5.2.2 JFreeChart

JFreeChart es una librería para gráficos escrita 100% en Java que facilita mostrar gráficos de calidad profesional en nuestras aplicaciones, ya sean web o de escritorio. Entre las características principales de esta biblioteca tenemos:

- Un API consistente y bien documentado con soporte para un amplio rango de tipos de gráficas.

- Un diseño flexible fácilmente extensible, y la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) y de cliente (Swing, por ejemplo).

- Soporte para varios tipos de salida, incluyendo componentes Swing, archivos de imagen como PNG y JPEG, y formatos gráficos de vectores (incluyendo PDF, EPS y SVG).

- JFreeChart es Software Libre, este está distribuido bajo la licencia LGPL, que permite el uso en aplicaciones propietarias.

1.5.3 Lenguajes de Programación

1.5.3.1 C++

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos.

Se suele decir que C++ es un lenguaje de programación multiparadigma ya que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

La principal característica de C++ es que es un lenguaje orientado a objetos, y por tanto soporta las capacidades fundamentales de la Programación Orientada a Objetos: Abstracción, Encapsulamiento, Modularidad, y la Jerarquía. C++ además, brinda soporte para la programación genérica (templates), y tiene una enorme compatibilidad con el C principalmente por la gran cantidad de código C que comparten.

C++ es un lenguaje de programación estandarizado (ISO/IEC 14882:1998), posee una biblioteca estándar, pero además, es un lenguaje muy potente, flexible y eficaz frente al resto de los lenguajes orientados a objetos, características que han hecho que se le considere como lenguaje universal, de propósito general y ampliamente utilizado tanto en el ámbito profesional como en el educativo. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Una de las ventajas que ofrece es que es mucho más sencillo de aprender para los programadores que ya conocen el lenguaje C, sin embargo es a su vez uno de los menos automatizados (obliga a hacerlo casi todo manualmente al igual que C).

1.5.4 Biblioteca para la realización de pruebas

1.5.4.1 CxxTest.

Es una biblioteca utilizada para llevar a cabo las pruebas al código fuente de cualquier proyecto. Está diseñada para ser tan portátil como sea posible. Es muy fácil de usar y de poner en funcionamiento.

Cxxtest utiliza el lenguaje Perl para generar código C++. Analiza el código de las pruebas implementadas por el usuario y genera a un corredor de prueba en C++ que se ejecuta directamente desde las pruebas del usuario. Por su flexibilidad es posible hacer pruebas rigurosas con código sencillo. Su único requerimiento es instalar Perl y que este se ejecute correctamente.

Dentro de sus principales ventajas están:

- No requiere de funciones miembro de plantilla.
- No requiere de manejo de excepciones.
- No requiere ninguna biblioteca externa.
- Es una herramienta multiplataforma y está disponible en los repositorios de varias distribuciones de software libre como es el caso de Debian. Se distribuye en su totalidad como un conjunto de archivos de cabecera que le hace extremadamente portátil y utilizable.

1.5.5 Ambiente de desarrollo

1.5.5.1 Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin

ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Se integra fácilmente con herramientas CASE como Visual Paradigm, con sistemas de control de versiones como Concurrent Versions System (CVS) y Subversion (SVN).

El punto fuerte de Eclipse es su modularidad ya que este IDE, a pesar de haber sido desarrollado inicialmente para el lenguaje Java, actualmente gracias a las facilidades que brinda para la inclusión de plug-in así como para el desarrollo de otros nuevos, posee soporte para casi todos los lenguajes de programación existentes, ya sean compilados o interpretados.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son de software libre, pero son incompatibles con la Licencia Pública General de GNU (GNU GPL). Mike Milinkovich, de la fundación Eclipse comentó que el cambio a la GPL será considerado cuando la versión 3 de la GPL sea liberada.

1.5.5.2 Qt Creator

Qt Creator es un IDE para el desarrollo de aplicaciones con Qt, creado por la compañía de teléfonos NOKIA después de adquirir a Trolltech y sus librerías Qt. Los sistemas operativos que soporta en forma oficial son:

- GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además, hay una versión para Linux con GCC (GNU Compiler Collection) 3.3.
- Mac OS X 10.4 (denominada versión “Tiger”) o superior, requiriendo Qt 4.x .
- Windows XP y Vista, requiriendo el compilador MinGW (Minimalist GNU for Windows) y Qt 4.4.3.

Incluye un avanzado editor de código C + +. Las características del editor incluyen resaltado de sintaxis y de auto completamiento. Qt Creator utiliza el compilador C + + de la colección de compiladores de GNU en Linux.

1.5.5.3 Code::Blocks

Code::Blocks es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C++. Está basado en la plataforma de interfaces gráficas WxWidgets, lo cual quiere decir que puede usarse libremente en diversos sistemas operativos, y está licenciado bajo la Licencia pública general de GNU.

Es un IDE construido como un núcleo altamente expansible mediante complementos (plug-in). Actualmente la mayor parte de la funcionalidad viene provista por los complementos incluidos predeterminadamente. No es un IDE autónomo que acepta complementos, sino que es un núcleo abstracto donde los complementos se convierten en una parte vital del sistema. Esto lo convierte en una plataforma muy dinámica y potente, no solo por la facilidad con que puede incluirse nueva funcionalidad, sino por la capacidad de poder usarla para construir otras herramientas de desarrollo tan solo añadiendo complementos.
Code::Blocks trae integradas plantillas para generar varias clases de programas, ya sea la clásica aplicación de consola, bibliotecas estáticas o dinámicas, o proyectos completos enlazados con populares bibliotecas como OpenGL y SDL.

1.6 Metodología de Desarrollo

OpenUp

OpenUp es un proceso de desarrollo de software propuesto por un conjunto de empresas, quienes lo donaron en el año 2007 a la Fundación Eclipse. La fundación lo ha publicado bajo una licencia libre y lo mantiene como método de ejemplo dentro del proyecto Eclipse Process Framework.

El OpenUP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto no provee lineamientos para todos los elementos que se manejan en un proyecto pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos de OpenUP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

Este proceso de desarrollo unificado está basado en Rational Unified Process (RUP), desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de Adaptación, Importancia a los involucrados e interesados en los resultados del proyecto; Colaboración, Valor a la iteración; y Calidad Continua.

El OpenUP estructura el ciclo de vida de un proyecto en cuatro fases: concepción, elaboración, construcción y transición. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.

1.7 Herramienta Case

Visual Paradigm

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Ordenador, en inglés Computer Aided Software Engineering) que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo de desarrollo de un software, desde la fase de análisis hasta el despliegue del mismo. Permite realizar ingeniería directa o inversa sobre el software, es capaz a partir de un modelo relacional en diferentes SGBD, desplegar todas las clases asociadas a las tablas y soporta múltiples usuarios trabajando sobre el mismo proyecto. Visual Paradigm es una herramienta libre utilizada para el modelado de aplicaciones. Esta herramienta permite construir la aplicación con mayor rapidez, mayor trabajo en equipo, fácil de utilizar, mayor exactitud, además de facilitar la interoperabilidad con otras herramientas CASE, la mayoría de los IDEs principalmente y permitir la integración de todos los componentes.

1.8 Alternativas de desarrollo

Existen dos tendencias en cuanto al desarrollo de sistemas de generación de reportes:

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

Para el desarrollo podemos valorar utilizar las siguientes alternativas:

- **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.

1.9 Conclusiones Parciales.

En este capítulo se realizó un esbozo de las principales características de los sistemas de generación de reportes, y se logró tomar experiencias del comportamiento y las funcionalidades que ofrecen los sistemas de generación de reportes más usados en el mundo, que servirán como base para comenzar el desarrollo de la aplicación enfocándonos además en las necesidades propias del sistema. Luego de un profundo análisis de las tecnologías, lenguajes y bibliotecas posibles a utilizar en el desarrollo de la solución se está en condiciones de decidir cuál de cada una de ellas; de acuerdo con sus beneficios o dificultades de uso; es la más adecuada para el desarrollo de la aplicación propuesta.

Capítulo 2
Descripción del Sistema y Propuesta de Solución

En el presente capítulo se exponen las principales procesos que se llevan a cabo en el Sistema de Generación de Informes y se hace un análisis del funcionamiento de cada uno de estos procesos para realizar la propuesta de la solución al problema plateado. Se describe además la arquitectura que tendrá el mismo, así como el patrón arquitectónico a utilizar.

2.1 Principales características y funcionalidades del Sistema.

El subsistema o módulo para la generación de reportes es el encargado de conformar un nuevo reporte partiendo de un diseño existente. Durante el proceso de creación del nuevo reporte, se ejecutan todas las consultas desde las fuentes de datos existentes y acorde a estas se crean el número de páginas y componentes que serán mostrados en el visor de Reportes. Esta herramienta ha estado integrada hasta el momento en el HMI del SCADA y entre sus principales funcionalidades podemos mencionar; visualizar un reporte, exportar a pdf y HTML, e imprimir el reporte para obtener una copia en formato duro del mismo.

Visualizar_Reporte: Luego de la selección de un reporte previamente creado, posibilita la visualización de una vista previa del mismo.

Generar_Reporte: El sistema se conecta a la fuente de datos recupera la definición del reporte y lo transforma al formato de salida predeterminado.

Exportar_Reporte: Posibilita la exportación del reporte a diferentes formatos de salida: PDF, HTML, CSV

Imprimir: Permite imprimir un reporte en formato duro.

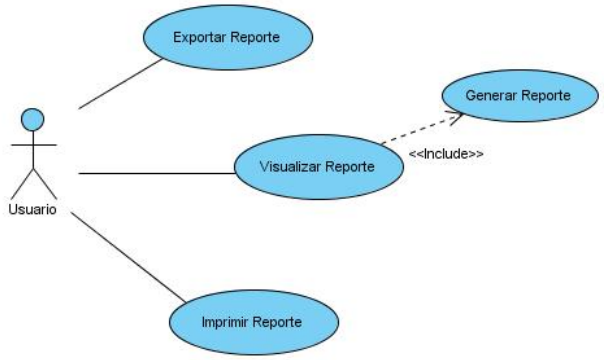

Fig.2: Diagrama de CU del Sistema

2.2 Selección de la Alternativa de Desarrollo

Alternativa Autónoma.

Esta consiste en desarrollar todo el sistema de generación de informes, partiendo del los requerimientos y análisis de la versión anterior del sistema, ya que los requisitos funcionales que determinan el funcionamiento del sistema no varían y de los componentes básicos de los que disponen las librerías que se decidan emplear para el proyecto. En este caso se dispondría de todo el código desarrollado y documentado del sistema anterior, y se estudiarían las clases existentes en caso de poder reutilizar algunos componentes que pueden ya estar bien implementados. El mayor esfuerzo recaería en la implementación de las interfaces gráficas que es uno de los aspectos más importantes del desarrollo de la nueva versión.

2.3 Herramientas y Lenguajes a utilizar.

2.3.1 Lenguaje de Modelado:

- ✓ UML

Lenguaje de Modelado Unificado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema mediante modelos. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y

aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

2.3.2 Metodología de Desarrollo

- ✓ OpenUp

Por ser un procesos de desarrollo de software de código abierto, que aplica un enfoque iterativo e incremental dentro de su estructura del ciclo de vida y que puede adaptarse para desarrollar diversos tipos de proyectos. Es un proceso para equipos de desarrollo pequeños y que le dan valor a la colaboración y a las necesidades de los usuarios.

2.3.3 Herramienta Case:

- ✓ Visual Paradigm

Por ser una herramienta UML de tipo profesional, la cual soporta el ciclo completo de desarrollo de software. Constituye una gran ayuda en la rápida construcción de aplicaciones de calidad a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generador de código desde diagramas y generar documentación. Y además por ser Software Libre.

2.3.4 Lenguaje de Programación:

- ✓ C++

Se utilizará como lenguaje de programación a C++ dada su portabilidad y eficiencia, además de que es el lenguaje en el que mayor experiencia se tiene por parte del equipo de desarrollo del proyecto.

2.3.5 IDE de Desarrollo:

- ✓ Eclipse

Se utilizará Eclipse como IDE de desarrollo por ser de código abierto, extensible y porque incorpora funcionalidades tan interesantes como la refactorización de código. Además que es el IDE de desarrollo que ha utilizado el equipo del proyecto en el desarrollo de otras aplicaciones; y por tanto es en el que más experiencia se tiene.

2.4 Selección de la biblioteca gráfica a utilizar.

Para la selección de la biblioteca gráfica se realizó una comparación entre las tres bibliotecas analizadas en el capítulo anterior, evaluando las principales características que poseen cada unas de ellas, lo cual está reflejado en el Anexo I.

Se puede observar que en sentido general todas estas bibliotecas gráficas son de gran utilidad y facilitan el trabajo de los desarrolladores de software. Si bien es cierto que GTK+ tiene algunas ventajas, el hecho de estar escrito de forma estructurada en lenguaje C, no permitirá hacer uso de los recursos y potencialidades de la programación orientada a objetos, además de que su uso queda descartado debido a todos los problemas que ha demostrado tener en la implementación de la versión anterior. Por otro lado se encuentra la biblioteca wxWidgets, que si es orientado a objeto y utiliza el lenguaje C++, sin embargo su diseño orientado a objetos no es el mejor, comparado con QT. La wxWidgets a veces abusa de la utilización de macros para realizar ciertas operaciones, como las tablas de eventos, etc.; aunque esto hace que codificar sea más fácil, también complica la labor de depuración. Además esta biblioteca no cuenta con soporte para algunas características relativamente modernas con respecto a la publicación del estándar C++ de 1998, como por ejemplo el manejo de excepciones y el trabajo con la Librería Estándar de Plantillas (STL). Otro elemento importante a tener en cuenta es la cantidad de documentación, permitiendo mitigar costos en cuestiones de tiempo de aprendizaje, contrariamente a GTK+ y wxWidgets, Qt consta de excelente documentación, y una amplia comunidad de desarrollo, así como de herramientas que agilizan el tiempo de creación de las aplicaciones. Valorando todo lo anterior se determinó usar Qt para comenzar el desarrollo del sistema.

2.5 Arquitectura del sistema.

La arquitectura que se ha propuesto para el sistema de generación de reportes Selux es implementar una solución orientada al patrón arquitectónico Modelo- Vista-Controlador (MVC), siguiendo la misma estructura que tenia la primera versión de la aplicación, y además porque este es un patrón que es ampliamente utilizado a nivel mundial en el desarrollo de sistemas.

El patrón MVC es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- **Modelo:** Encapsula los datos y las funcionalidades.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

Sin importar el tipo de implementación de MVC que se tenga, el flujo que se sigue generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario.
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

A lo largo de los años, desde la presentación de este patrón a la comunidad científica se han desarrollado tres variantes fundamentales (4), sin embargo en el desarrollo de del sistema Selux nos centraremos en la utilización de la variante inicial del patrón MVC.

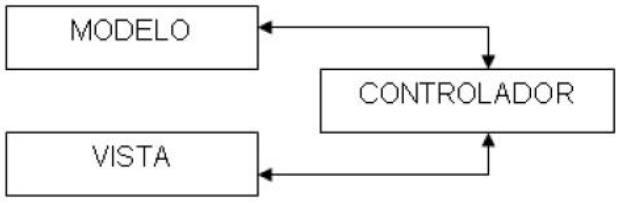

Fig. 3: Variante inicial del patrón MVC. (4)

En esta variante del patrón MVC no existe comunicación entre las vistas y el modelo, sino que todo el flujo de datos que se maneja en el sistema circula a través de las clases que se encuentran en el controlador. En el caso de la aplicación que se propone, los modelos lo constituirán todas las clases que se encuentran dentro de los paquetes Graphics, Draw, e IO, por ser las clases que modelan toda la información que se maneja en el sistema, así como el acceso a las base de datos. Las clases dentro del paquete Base serán las encargadas de llevar el control de la aplicación, accediendo a los modelos para obtener la información que necesitan las clases del paquete View para ser mostradas, ya que las mismas implementaran las vistas o interfaces con las que interactúan los usuarios.

2.6 Propuesta de sistema.

2.6.1 Descripción general de la propuesta de sistema

El sistema que proponemos será una aplicación independiente que dejará de ser una herramienta utilizada dentro del HMI, para ser un módulo independiente que pueda ser usado por cualquier otra aplicación o sistema que necesite la generación de informes. Constará básicamente de un motor de generación, que será el encargado de cargar las plantillas de diseños existentes en memoria y enlazar los datos extraídos de una o varias bases de datos, con dichos diseños que deben haber sido previamente realizados; para finalmente generar un documento de informe a partir de los datos solicitados y el diseño deseado. El documento resultante será un archivo que se podrá visualizar en pantalla o despachar a distintos medios como la impresora o disco duro en forma de archivo con diversos formatos.

El desarrollo de la aplicación estará basado en los requerimientos y análisis de la versión anterior del sistema, debido a que no existe ninguna variación en cuanto a los procesos que se llevan a cabo en la aplicación. Sin embargo se realizarán algunos cambios en el diseño, debido a la existencia innecesaria de un pequeño grupo de clases y atributos. Se redefinirán también varias jerarquías y paquetes de clases para lograr una mejor estructuración de las mismas. Los principales cambios que tendrá el nuevo sistema serán en la forma en que se implementan los procesos que debe llevar a cabo, para disponer de un código más manejable, entendible y que satisfaga las necesidades de los usuarios.

El sistema además de permitir salvar a diversos formatos como PDF, CSV y HTML, navegar entre páginas, acercar o alejar el documento e imprimir; mostrará gráficos 3D, se podrá ejecutar en múltiples plataformas, tendrá un menor consumo de recursos y brindará una mejor calidad en tiempo de ejecución.

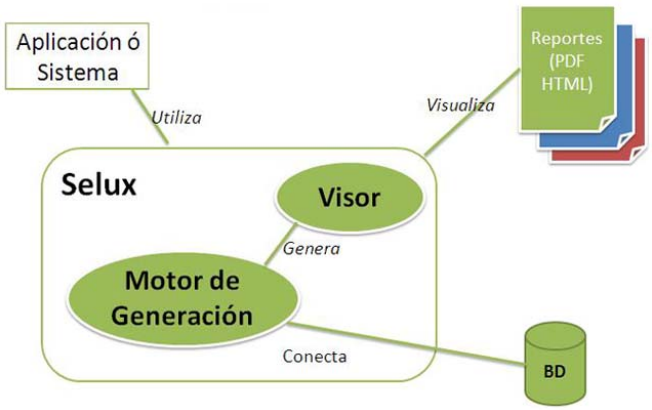

Fig. 4: Esquema de Funcionamiento del Sistema de Generación de Reportes Selux.

1.1.1 Paquetes de diseño más significativos.

Esta sección describe la descomposición del sistema en paquetes. Además para cada paquete su descomposición en clases y la descripción de sus responsabilidades así como las relaciones más importantes, operaciones y sus atributos.

Entre los principales paquetes se encuentran:

- Paquete Base.
- Paquete Graphics.
- Paquete IO.
- Paquete QT.
  - Paquete Draw.
  - Paquete View
- Paquete Types.
- Paquete Web.
  - Paquete WebBase.

- Paquete WebDraw.

1.1.1.1 Paquete Base

En este Paquete se encuentran las clases sobre las cuales se soporta parte del sistema, estas definen las principales interfaces que se implementan la visualización de los reportes, fabricación y renderizado de los objetos y el control de la aplicación en general. Como principales elementos de este Paquete se encuentran las clases Document, Engine y Report.

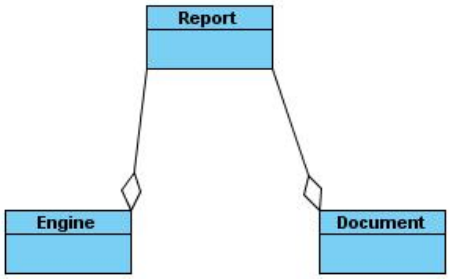

Fig. 5: Diagrama de clases. Paquete Base.

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

Tabla # 1: Descripción de la clase de Diseño "Report".

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	Types::String
title	Types::String
description	Types::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:	void setName(Types::String reportName)
Descripción:	Asigna un nuevo nombre al documento
Nombre:	Types::String getName()
Descripción:	Devuelve el nombre del documento
Nombre:	void setTitle(Types::String reportTitle)
Descripción:	Asigna un nuevo título al documento
Nombre:	Types::String getTitle()

Descripción:	Devuelve el titulo del documento
Nombre:	void setDescription(Types::String reportDescription)
Descripción:	Asigna nueva descripción al documento
Nombre:	Types::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.

Tabla # 2: Descripción de la clase de Diseño "Document".

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 el documento en una colección de páginas 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 una nueva página en la colección de páginas para ser pintada
Nombre:	void endPage(Document&, RenderSection&)
Descripción:	Crea la última página en la colección de páginas para ser pintada

Tabla # 3: Descripción de la clase de Diseño "Engine".

1.1.1.2 Paquete Graphics

En el paquete Graphics se encuentran las clases que representan los principales objetos gráficos, definiendo las características de los mismos y logrando una jerarquía de clases que abarca los elementos más importantes en la confección de un reporte.

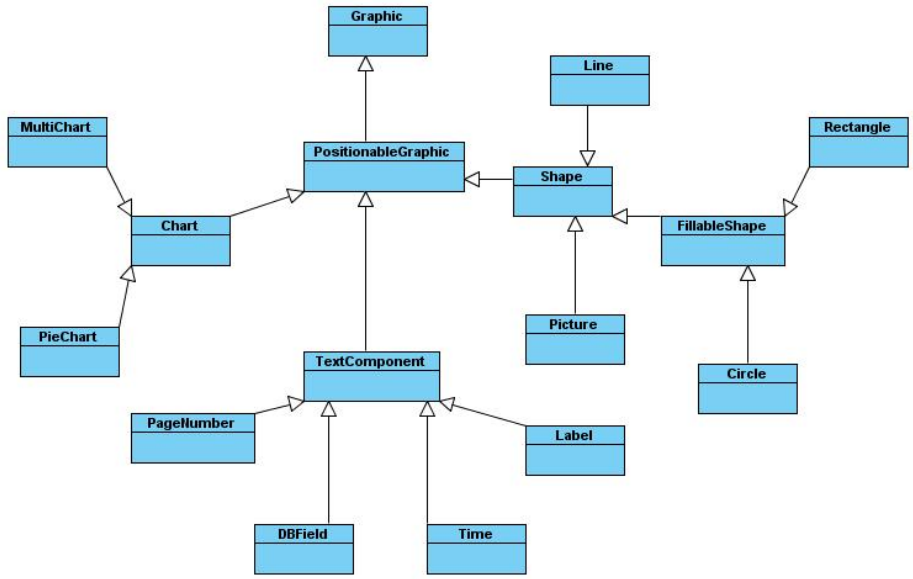

Fig. 6: Diagrama de Clases. Paquete Graphics.

Nombre: Graphic	
Descripción: Clase base que define a cualquier objeto gráfico dentro del reporte.	
Tipo de clase: Entidad	
Atributo	Tipo
name	Types::Name
id	Types::Identifier
Para cada responsabilidad:	
Nombre:	Types::Name getName()
Descripción:	Retorna el nombre del objeto gráfico
Nombre:	Types::Identifier getIdentifier()
Descripción:	Retorna el id del objeto gráfico
Nombre:	void setName(Types::Name name)
Descripción:	Cambia el valor del atributo ¨name¨
Nombre:	void setIdentifir(Types::Identifier id)
Descripción:	Cambia el valor del atributo id

Tabla # 4: Descripción de la clase de Diseño "Graphic".

Nombre: PositionableGraphics	
Descripción: Clase base que define a todos los objeto gráfico que ocupan una posición dentro del reporte.	
Tipo de clase: Entidad	
Atributo	Tipo

rect	Rect
Para cada responsabilidad:	
Nombre:	void setRect(const Rect& rect)
Descripción:	Cambia el valor atributo ¨rect¨
Nombre:	Rect getRect() const
Descripción:	Devuelve el valor del atributo ¨rect¨

Tabla # 5: Descripción de la clase de Diseño "PositionableGraphics".

Nombre: Shape	
Descripción: Clase base que define a todos los objeto formas dentro del reporte.	
Tipo de clase: Entidad	
Atributo	Tipo
pen	Pen
Para cada responsabilidad:	
Nombre:	void setPen(const Pen& pen)
Descripción:	Cambia el valor del atributo ¨pen¨
Nombre:	Pen getPen() const
Descripción:	Devuelve el valor del atributo ¨pen¨

Tabla # 6: Descripción de la clase de Diseño "Shape".

Nombre: Text
Descripción: Clase base que define a todos los objeto textos dentro del reporte.

Tipo de clase: Entidad	
Atributo	Tipo
text	Types::String
font	Font
color	Color
Para cada responsabilidad:	
Nombre:	Types::String getText() const
Descripción:	Devuelve el valor del atributo “text”
Nombre:	void setText(const Types::String text)
Descripción:	Cambia el valor del atributo “text”
Nombre:	Font getFont() const
Descripción:	Devuelve el valor del atributo “font”
Nombre:	void setFont(const Font & font)
Descripción:	Asigna un nuevo valor del atributo “font”
Nombre:	Color getColor() const
Descripción:	Devuelve el valor del atributo “color”
Nombre:	void setColor(const Color & color)
Descripción:	Asigna un nuevo valor al atributo “color”

Tabla # 7: Descripción de la clase de Diseño “Text”.

Nombre: **Chart**
Descripción: Clase base que define a todos los gráfico que puedan generarse dentro del reporte.

Tipo de clase: Entidad	
Atributo	**Tipo**
title	String
dataTable	std::list<Report::IO::FieldCollection>
xData	Report::IO::FieldCollection
yIDQuery, xIDQuery	int
xNameColumnQuery, yNameColumnQuery	Types::String
backgroundColor	Report::Graphics::Color
is3D, atributesVisible	bool
borderWidth	Report::Types::Size
borderColor	Report::Graphics::Color
Para cada responsabilidad:	
Nombre:	String getTitle();
Descripción:	Devuelve el valor del atributo ”title ”
Nombre:	void setTitle(String t);
Descripción:	Asigna un Nuevo valor al atributo “title”
Nombre:	Report::Graphics::Color getBackgroundColor();
Descripción:	Devuelve el valor del atributo ”backgroundColor”
Nombre:	void setBackgroundColor(Report::Graphics::Color c);
Descripción:	Asigna un Nuevo valor al atributo “ backgroundColor”
Nombre:	Report::Graphics::Color getBorderColor();
Descripción:	Devuelve el valor del atributo ”borderColor”

Nombre:	void setBorderColor(Report::Graphics::Color c);
Descripción:	Asigna un Nuevo valor al atributo “ borderColor”
Nombre:	Report::Types::Size getBorderWidth();
Descripción:	Devuelve el valor del atributo ”borderWidth”
Nombre:	void setBorderWidth(Report::Types::Size s);
Descripción:	Asigna un Nuevo valor al atributo “ borderWidth”
Nombre:	int getXIDQuery();
Descripción:	Devuelve el valor del atributo ”xIDquery ”
Nombre:	void setXIDQuery(int id);
Descripción:	Asigna un Nuevo valor al atributo “ xIDquery”
Nombre:	int getYIDQuery();
Descripción:	Devuelve el valor del atributo ”yIDQuery ”
Nombre:	void setYIDQuery(int id);
Descripción:	Asigna un Nuevo valor al atributo “ yIDQuery”
Nombre:	String getXNameColumnQuery();
Descripción:	Devuelve el valor del atributo ”XNameColumnQuery”
Nombre:	void setXNameColumnQuery(String name);
Descripción:	Asigna un Nuevo valor al atributo “ XNameColumnQuery”
Nombre:	String getYNameColumnQuery();
Descripción:	Devuelve el valor del atributo ”yNameColumnQuery ”
Nombre:	void setYNameColumnQuery(String name);
Descripción:	Asigna un Nuevo valor al atributo “ yNameColumnQuery”

Nombre:	void setTreeDVisible(bool visible);
Descripción:	Asigna un Nuevo valor al atributo “ treeDVisible”
Nombre:	bool getTreeDVisible();
Descripción:	Devuelve el valor del atributo ”treeDVisible ”
Nombre:	void setAtributesVisible(bool visible);
Descripción:	Asigna un Nuevo valor al atributo “ atributesVisible”
Nombre:	bool getAtributesVisible();
Descripción:	Devuelve el valor del atributo ”atributesVisible ”
Nombre:	void setXData(Report::IO::FieldCollection);
Descripción:	Asigna un Nuevo valor al atributo “xData”
Nombre:	void setDataTable(Report::IO::FieldCollection);
Descripción:	Asigna un Nuevo valor al atributo “dataTable”

Tabla # 8: Descripción de la clase de Diseño “Chart”.

1.1.1.3 Paquete IO

En este paquete se modela el mecanismo de entrada/salida (Input/Output) del visualizador de reportes. En el mismo se han definido un grupo de entidades de suma importancia en el sistema, ya que proveen de las funcionalidades para obtener la información de las Bases de Datos.

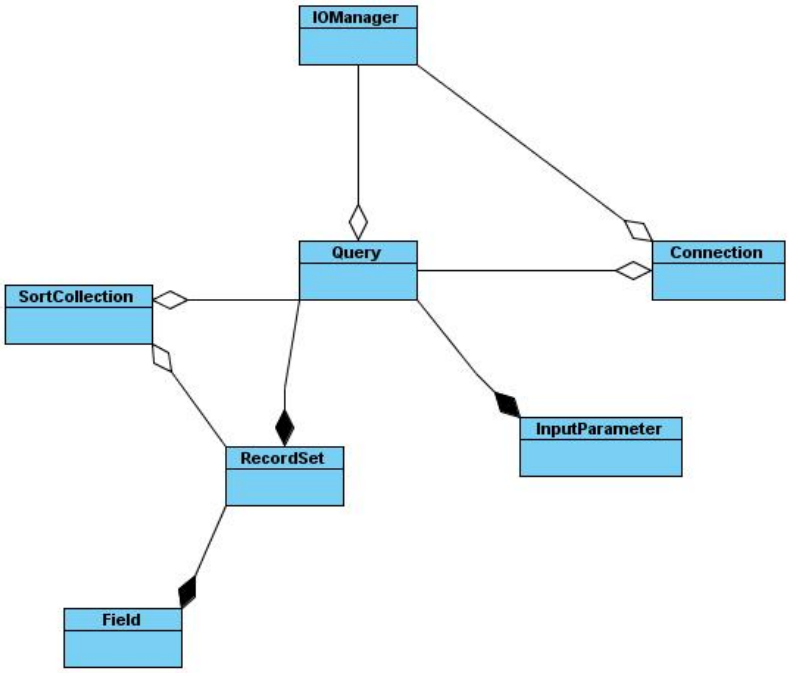

Fig.7: Diagrama de Clases. Paquete IO.

Nombre: IOManager	
Descripción:	
Tipo de clase:	
Atributo:	**Tipo:**
queries	QueryCollection
connections	ConnectionCollection
Para cada responsabilidad:	
Nombre:	void exec()
Descripción:	Ejecuta las consultas existentes

Nombre:	void addQuery(Query* query)
Descripción:	Añade una nueva consulta a través del parámetro “query”
Nombre:	Query* find(Types::Identifier identifier)
Descripción:	Busca y devuelve si existe un consulta dado un “id”
Nombre:	QueryCollection& getQueryCollection()
Descripción:	Devuelve una colección de todas las consultas existentes
Nombre:	ConnectionCollection& getConnectionCollection()
Descripción:	Devuelve una colección de todas las conexiones existentes.
Nombre:	Connection* findConnection(Types::Identifier identifier)
Descripción:	Busca y devuelve si existe una conexión dado un “id”
Nombre:	void addConnection (Connection* connection)
Descripción:	Añade una nueva conexion.
Nombre:	void clean()
Descripción:	Elimina todas las consultas y conexiones existentes.
Nombre:	void insertParametersValuesInQuerys(InputParameterCollection & inputParameterCollection)
Descripción:	Inserta los valores de los parámetros en las consultas.

Tabla # 9: Descripción de la clase de Diseño “IOManager”.

Nombre: Query
Descripción:
Tipo de clase:

Atributo:	**Tipo:**
name	Types::String
id	Types::Identifier
sortCriteria	SortCollection
parameters	InputParameterCollection
recordset	RecordSet*
connectionId	Types::Identifier
connection	Connection*

Para cada responsabilidad:	
Nombre:	Types::String getName()
Descripción:	Devuelve el valor del atributo “name”
Nombre:	void setName(Types::String name)
Descripción:	Asigna un nuevo valor al atributo “name”
Nombre:	Types::Identifier getIdentifier() const
Descripción:	Devuelve el valor del atributo “id”
Nombre:	void setIdentifier(Types::Identifier id)
Descripción:	Asigna un Nuevo valor al atributo “id”
Nombre:	RecordSet* getResult()
Descripción:	Devuelve los datos obtenidos de la ejecución de las consultas
Nombre:	InputParameterCollection& getParameterCollection ()
Descripción:	Devuelve la colección de parámetros
Nombre:	void setConnectionId(Types::Identifier connectionId)

Descripción:	Asigna un valor al atributo “ connectionId”
Nombre:	Types::Identifier getConnectionId ()
Descripción:	Devuelve el valor del atributo “ connectionId”
Nombre:	void setConnection (Connection* connection)
Descripción:	Asigna una nueva conexión
Nombre:	Connection* getConnection ()
Descripción:	Devuelve el valor del atributo “connection”
Nombre:	SortCollection& getSortCriteria ()
Descripción:	Devuelve la colección de criterios de ordenamiento
Nombre:	void clear()
Descripción:	Elimina todos los elementos existentes en el atributo “recordSet”
Nombre:	virtual void execute() = 0
Descripción:	Método virtual puro para ejecutar la consulta en dependencia del tipo de consulta que sea.

Tabla # 10: Descripción de la clase de Diseño “Query”.

Nombre: Connection	
Descripción:	
Tipo de clase:	
Atributo:	**Tipo:**
id	Types::Identifier
Para cada responsabilidad:	

Nombre:	Types::Identifier getIdentifier() const
Descripción:	Devuelve el valor del atributo “id”
Nombre:	void setIdentifier(Types::Identifier id)
Descripción:	Asigna un nuevo “id”
Nombre:	virtual bool isActive() = 0;
Descripción:	Método virtual puro que será implementado por las clases hijas para devolver si la conexión está activa.
Nombre:	virtual void open () = 0;
Descripción:	Método virtual puro que será implementado por las clases hijas para abrir la conexión.
Nombre:	virtual void close () = 0;
Descripción:	Método virtual puro que será implementado por las clases hijas para cerrar la conexión.

Tabla # 11: Descripción de la clase de Diseño “Connection”.

1.1.1.4 Paquete QT

En el paquete Qt se modelan todas las clases que tienen algún tipo de dependencia con la biblioteca Qt utilizada para implementar el visualizador de Reportes, esto se hace para lograr independencia total de la lógica y las entidades del sistema. La información que se maneja dentro de dicho paquete se encuentra organizada en dos paquetes; el paquete Draw, y el paquete View.

1.1.1.4.1 ***Paquete Draw***

En este Paquete se encuentran las clases que implementan el dibujo de los objetos definidos en el paquete Graphics, haciendo uso de la biblioteca seleccionada.

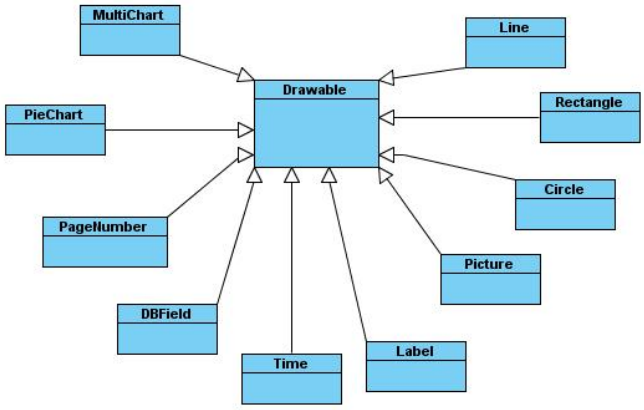

Fig. 8: Diagrama de Clases. Paquete Draw.

Nombre: **Drawable**	
Descripción: Clase base que define a cualquier objeto que pueda ser pintado en el reporte.	
Tipo de clase: Entidad	
Para cada responsabilidad:	
Nombre:	Virtual void draw(Graphics::Scene& scene, int xOffset, int yOffset)=0
Descripción:	Método virtual puro que será implementado por cada unas de las clases hijas para pintar cada objeto en el reporte.

Tabla # 12: Descripción de la clase de Diseño "Drawable".

1.1.1.4.2 *Paquete View*

En este Paquete se encuentran las clases con las que se construyen las interfaces graficas de usuarios. La clase que implementa la interfaz principal ReportView, así como el resto de componentes que colaboran para conformar la interfaz con la que el usuario va a interactuar.

Fig. 9: Diagrama de Clases. Paquete View.

Nombre: **ReportView**	
Descripción: Clase que representa la interfaz principal que va a interactuar con el usuario.	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
ui	Ui::MainPreview
goToPagesDialog	GoToPage *
view	QGraphicsView *
zoomCombo	QComboBox *
lastPageShowed;	int
zoomLevel;	int
emptyReport;	bool
width;	int
height;	int
report;	Report::Base::Report*
Para cada responsabilidad:	
Nombre:	void draw()

Descripción:	Manda a generar y pintar la pagina que el usuario desea ver.
Nombre:	void initialize();
Descripción:	Inicializa todos los componentes que conforman el visor
Nombre:	void showReport();
Descripción:	Inicializa los valores de los atributos una vez que le es asignado un reporte.
Nombre:	void setReport(Report::Base::Report* report);
Descripción:	Asigna un nuevo reporte
Nombre:	void zoomIn();
Descripción:	Aumenta el nivel de zoom del visor para acercar el documento
Nombre:	void zoomOut();
Descripción:	Disminuye nivel de zoom del visor para alejar el documento
Nombre:	void zoomNormal();
Descripción:	Pone el nivel de zoom en estado normal.
Nombre:	void print();
Descripción:	Muestra un cuadro de diálogo con las opciones para imprimir un reporte.
Nombre:	void nextPage();
Descripción:	Muestra la página siguiente.
Nombre:	void previousPage();
Descripción:	Muestra la página anterior.
Nombre:	void firstPage();
Descripción:	Muestra la primera página
Nombre:	void lastPage();

Descripción:	Muestra la última página
Nombre:	void goPage();
Descripción:	Muestra un cuadro de diálogo para que el usuario seleccione la página que desea ver.
Nombre:	void fileSave();
Descripción:	Muestra un cuadro de diálogo con las opciones de salvar el documento en formato HTML o CVS.
Nombre:	void comboZoomChangeValue();
Descripción:	Permite al usuario manejar las opciones de zoom de forma directa.
Nombre:	void exportPDF();
Descripción:	Muestra un cuadro de dialogo con las opciones de salvar el documento en formato PDF.
Nombre:	void updateControls();
Descripción:	Actualiza los valores de los controles con los que interactúa el usuario
Nombre:	void privateDraw();
Descripción:	Dibuja la página actual del documento.

Tabla # 13: Descripción de la clase de Diseño "ReportView".

1.1.1.5 Paquete Web

Este paquete contiene las clases que se encargan de exportar el resultado del reporte obtenido a formato HTML. Para una mejor organización la información de este paquete está dividida en dos subpaquetes.

1.1.1.5.1 Paquete WebBase.

En este paquete se encuentran las clases que controlan todo el proceso de exportar a HTML, de manera similar al proceso de visualizar un reporte en pantalla.

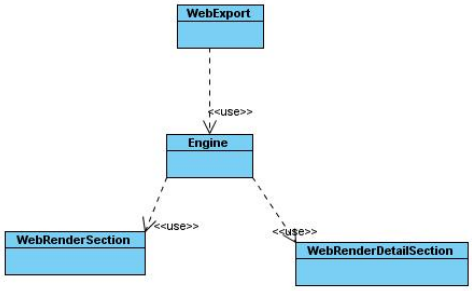

Fig. 10: Diagrama de Clases. Paquete WebBase.

Nombre: WebEngine	
Descripción: Clase que se encarga de renderear los datos de un documento para que guardados en un documento HTML.	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
xOffset, yOffset, yOffsetUp	int
currentPage	Web::Draw::Page*
contextCanvas, contextFinal, contextInitial	Web::Draw::Context
detailHeigth	int
reportFooterHeigth	double
pageFooterHeigth	double
pageHeaderHeigth	double
reportHeaderHeigth	double
Height, width	double
lastPage, firstPage	bool
marginTop, marginLeft, marginBottom, marginRight	int

Para cada responsabilidad:	
Nombre:	void render(PageCollection& pages, Document& document);
Descripción:	Renderea el documento en una colección de páginas para ser visualizado.
Nombre:	void drawReportHeader(Section* reportHeader, WebRenderSection& renderSection)
Descripción:	Pinta un objeto de tipo Section en la sección ReportHeader de una página.
Nombre:	void drawPageHeader(Section* pageHeader, WebRenderSection& renderSection)
Descripción:	Pinta un objeto de tipo Section en la sección PageHeader de una página.
Nombre:	void drawDetail(SCADA::Report::Graphics::SectionDetail* sectionDetail, Document& document, WebRenderDetailSection& renderDetailSection)
Descripción:	Pinta un objeto de tipo SectionDetail en la sección Detail de una página,
Nombre:	void drawPageFooter(Section* pageFooter, WebRenderSection& renderSection)
Descripción:	Pinta un objeto de tipo Section en la sección PageFooter de una página.
Nombre:	void drawReportFooter(Section* reportFooter, WebRenderSection& renderSection)
Descripción:	Pinta un objeto de tipo Section en la sección ReportFooter de una página.
Nombre:	void initNewPage(PageCollection& pages, Document& document, WebRenderSection& renderSection)
Descripción:	Crea la primera página en la colección de páginas para ser pintada
Nombre:	void newPage(PageCollection& pages, Document& document, WebRenderSection& renderSection)
Descripción:	Crea una nueva página en la colección de páginas para ser pintada
Nombre:	void endPage(Document& document, WebRenderSection& renderSection)
Descripción:	Crea una la última página en la colección de páginas para ser pintada

Tabla # 14: Descripción de la clase de Diseño "WebEngine".

Nombre: WebRenderSection	
Descripción: Clase que se encarga de renderear la colección de objetos gráficos asignados a cada sección.	
Tipo de clase: Controladora	
Atributo:	Tipo:
section	Report::Graphics::Section *
Para cada responsabilidad:	
Nombre:	Section* getSection();
Descripción:	Devuelve el valor del atributo section.
Nombre:	void setSection(Section*)
Descripción:	Asigna un nuevo valor al atributo section.
Nombre:	int process(double xOffset, double yOffset, Report::Web::Draw::Page& canvas)
Descripción:	Da la orden de dibujar a cada uno de los objetos gráficos que están en esa sección.
Nombre:	void restore();
Descripción:	

Tabla # 15: Descripción de la clase de Diseño "RenderSection".

1.1.1.5.2 Paquete WebDraw.

En este Paquete se encuentran las clases que representan los objetos que se visualizaran en la pagina HTML, en correspondencia a cada uno de los elementos gráficos definidos previamente en la visualización del reporte (imágenes, textos, campos de la base datos, gráficos, etc.), así como la definición de lo que será la página como tal.

Fig.11: Diagrama de Clases. Paquete WebDraw.

Nombre: Page	
Descripción:	
Tipo de clase:	
Atributo:	**Tipo:**
context	Context
width	double
height	double
Para cada responsabilidad:	
Nombre:	void setContext(Context& context);
Descripción:	Cambia el atributo context por uno nuevo pasado pro paramétro.
Nombre:	Context& getContext();
Descripción:	Devuelve una referencia del atributo context.

Tabla # 16: Descripción de la clase de Diseño "Page" del paquete WebDraw.

Capítulo 3
Implementación y Validación de la Solución

El presente capítulo presenta la implementación del sistema en términos de componentes, con el objetivo de mostrar los componentes que se obtienen de implementar las clases y los paquetes definidos en el diseño. También describe la vista de implantación del sistema y los casos de pruebas desarrollados para validar la solución implementada.

3.1 Implementación del Sistema

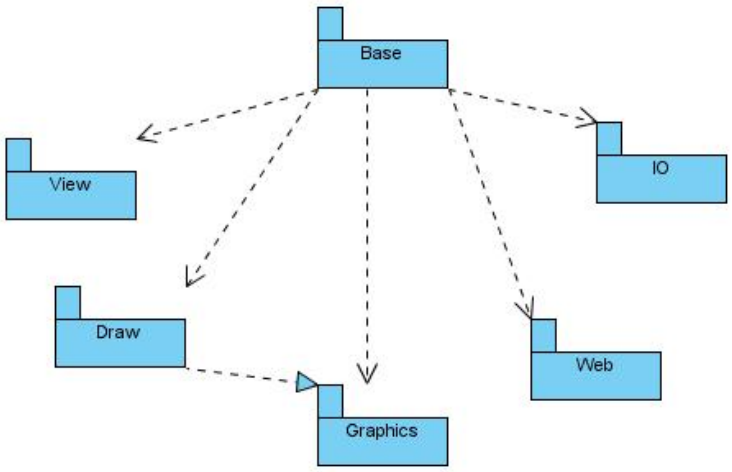

Fig.12: Diagrama de Componentes del Sistema de Generación de Informes Selux.

3.1.1 Componentes agrupados en paquetes.

A continuación se muestran algunos diagramas de los componentes más importantes de la aplicación, agrupados por paquete para un mejor entendimiento del sistema.

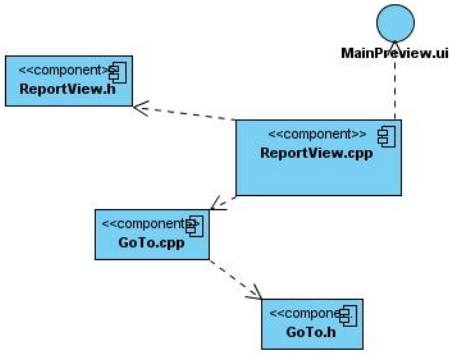

Fig.13: Diagrama de Componentes del paquete View.

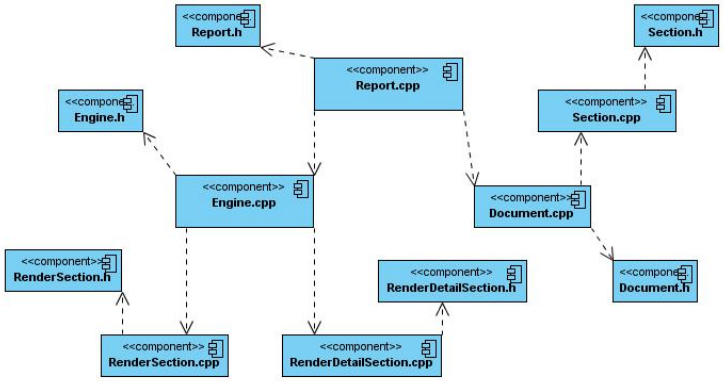

Fig.14: Diagrama de Componentes del paquete Base.

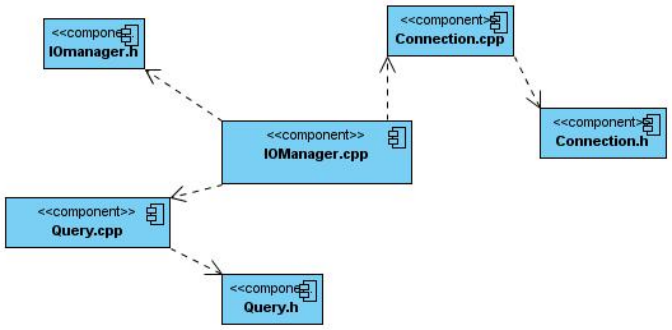

Fig.15: Diagrama de Componentes del paquete Base.

3.1.2 Diagrama de Despliegue

A continuación se muestra el despliegue en nodos físicos identificando los diferentes nodos y la forma de conexión entre ellos. Así como una breve descripción de los mismos y sus conexiones

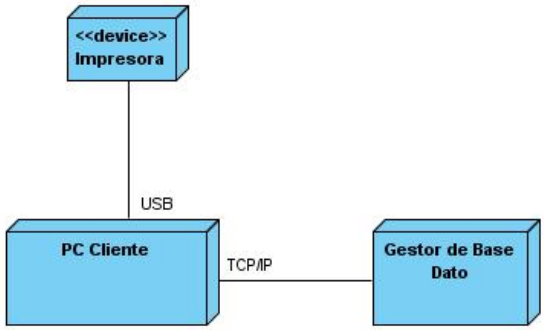

Fig.16: Diagrama de Despliegue del Sistema de Generación de Informes Selux.

PC Cliente: Este nodo se refiere a la PC donde estará instalado el sistema Selux para ser usado por los usuarios que necesiten trabajar con algún tipo de reporte.

Gestor de Base Dato: Este nodo representa la PC donde estará instalado un gestor de Base Dato al que se conectara el Sistema Selux, para extraer los datos que necesita para generar el reporte deseado por el usuario.

Impresora: este es un dispositivo que puede estar o no conectado a la PC donde se encuentre instalado el sistema de Generación de Reportes Selux, en casi de que el usuario quiera imprimir una copia en formato duro del reporte generado.

3.1.3 Estilo de Código Utilizado

3.1.3.1 Definición y Usabilidad de los Nombres

- Los nombres de las clases son sustantivos singulares.
- Los nombres deben reflejan el “qué” y no el “cómo”.

- Nombres lo suficientemente largo para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.
- Evitar nombre que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto de las letras para el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención, con excepción de la primera letra del nombre, la cual debe ser en minúscula.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaciones. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviación debe significar solo una cosa. En general agregar a la documentación las abreviaturas.
- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.

3.1.3.2 Manejo de Errores

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.
- Es buena práctica emplear herramientas para identificar errores en la codificación en caliente.

3.1.3.3 Documentación y Comentarios

- En el código debe documentarse en forma explicativa los pasos que se van ejecutando.
- Emplear oraciones completas al documentar código.
- Documentar mientras se programa.
- Documentar cualquiera cosa que no sea obvia en el código.

- Documentar eliminación de errores y cambios sobre el código.
- Al modificar el código se deben actualizar todos los comentarios y documentación asociada.
- Documentar cada rutina agregando: nombre del desarrollador, fecha, parámetros de entrada, valores de retorno, precondiciones, pos condiciones, dependencia con otros métodos o funciones y descripción general del algoritmo.
- Evitar agregar comentarios al final de líneas de código, salvo en el caso de declaraciones. En este caso tales comentarios deben estar alineados.
- Antes de la entrega de la aplicación, eliminar todos los comentarios superfluos y/o temporales con la finalidad de evitar confusiones en su mantenimiento.

3.1.3.4 Codificación

- Se establece un tamaño de identación estándar de tres (3) espacios, sin tabulaciones. Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Emplear cada variable y rutina solo para un propósito.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear select-case o switch en sustitución de if anidados sobre la misma variable.

- Liberar apuntadores de manera explícita.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Comentar siempre las llaves que cierran.
- Emplear al máximo operadores del tipo: +=, *=, /=, -=, ++, --, etc.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (declaraciones, lazos, etc.).
- Siempre asignar NULL a los apuntadores luego de ser destruidos (solo aplica para C)
- Evitar prácticas que incrementan explosivamente la complejidad, como lo son: objetos y variables globales y saltos tipo goto.

3.1 Realización de Pruebas

Una vez que se ha culminado con la implementación de cualquier sistema, es de vital importancia realizarle algunas pruebas para comprobar su correcto funcionamiento un vez que se ponga a disposición de los usuarios finales. En el presente capitulo se exponen las principales pruebas de unidad realizadas al Sistema de Generación de Informes Selux que validan el correcto funcionamiento del mismo.

3.2.1 Prueba de Unidad

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Las pruebas de la unidad centran su proceso de verificación en la menor unidad de software, el módulo. Una unidad de programa es lo suficientemente pequeña como para que el programador pueda probarla minuciosamente tomando como guía el diseño detallado. A este nivel las pruebas más importantes son del tipo de la caja blanca.

La idea es escribir casos de prueba para cada función o método en el módulo de forma que cada caso sea independiente del resto.

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

1. **Fomentan el cambio:** Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
2. **Simplifica la integración:** Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
3. **Documenta el código:** Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
4. **Separación de la interfaz y la implementación:** Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock (mock object) para simular el comportamiento de objetos complejos.
5. **Los errores están más acotados y son más fáciles de localizar:** dado que tenemos pruebas unitarias que pueden desenmascararlos.

Es válido aclarar que las pruebas no garantizan que la aplicación no tenga de defectos; las pruebas solo pueden demostrar que existen errores en el software y así posteriormente poderlos corregir.

3.2.2 Diseño de las pruebas de unidades que permitan validar la solución propuesta.

Para llevar a cabo las pruebas de unidad del sistema se utilizan un grupo de recursos físicos y lógicos que relacionamos a continuación:

Recursos físicos:

- Una computadora con un microprocesador Dual Core con una velocidad del CPU de 2.4 Ghz, una memoria RAM de 2 Gb y un disco duro con capacidad de 160 Gbytes.

Recursos lógicos:

- Sistema operativo Debian GNU/Linux.
- IDE de desarrollo Eclipse 3.2 con el plug-in CDT 2.1 para programar en C++.
- Framework para automatizar las pruebas; 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.

3.2.1 Casos de Prueba

3.2.1.1 Pruebas a las clases para la funcionalidad "Visualizar Reporte"

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

Caso 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 por parámetro.	La misma cadena pasada en los datos de entrada.	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 un objeto de tipo Section.	Válida
2	Valores correctos para un objeto de tipo Section.	Válida

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	Asignación correcta de la entrada a otra variable de tipo	

			Section.	Section.	
2	Verificar que el método funciona para un puntero de tipo Section no instanciado.	Section* section = NULL;	Asignación correcta de la entrada a otra variable de tipo Section.	Asignación correcta de la entrada a otra variable de tipo Section.	

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

Caso de Prueba:

1. void testSetWidth ()

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

3) Clase TestReport

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

- 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 vacía.	Válida

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 páginas quede vacía.	PageCollection pages	0	0	

4) Clase TestShape

- Prueba unitarias de la clase: Shape

- Casos de prueba del método: Color SetLineColor()
- Variables a considerar en el caso de prueba: Color lineColor
- Clase de Equivalencia para la variable: lineColor

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

Equivalencia		Equivalencia
1	Pasar valores válidos para el dominio de un objeto de tipo Color como pueden ser el valor de Red, Green, Blue, Alpha los cuales siempre deben estar entre 0 y 255.	Válida
2	Valores no válidos para el dominio de un Color, al menos uno de los parámetros del color debe estar por debajo de 0 o por encima de 255.	Inválida

Casos de Prueba:

1. *void testSetLineColor_1 ()*
2. *void testSetLineColor_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.	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 Color con valores no válidos.	Color(-100,100,260,0)	Excepción o algún tipo de notificación por parte del objeto cuando se le pasa como argumento un valor no válido.	No se produce ninguna notificación ni excepción.	

3.2.1.2 Pruebas a las clases para la funcionalidad "Exportar Reporte a HTML"

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.	Inválida

Casos de Prueba:

1. *void testSetPath_1 ()*
2. *void testSetPath_2 ()*

Caso de Prueba	**Objetivo de la Prueba**	**Datos de Entrada**	**Salida Esperada**	**Salida Obtenida**	**Observación**
1	Comprobar que se le asigna correctamente el camino pasado como parámetro.	path = "/home/lisi"	Camino con los mismos valores pasados en los datos de entrada.	Camino con los mismos valores pasados en los datos de entrada.	
2	Comprobar que se provoca alguna excepción al pasar un camino incorrecto o no accesible.	path = "/c///lisi"	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.	

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

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

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

Casos de Prueba:

2. *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/lisi"	No se debe provocar ningún tipo de error.	No se provoca error.	

3.2.2 Conclusiones Parciales

Las pruebas unitarias arrojaron resultados de gran importancia, ya que permitieron validar el código fuente desarrollado durante la etapa de implementación del sistema, facilitando además la corrección de los errores identificados.

Debido a la complejidad y extensión del sistema no se le realizaron pruebas a todas las clases de forma particular; teniendo en cuenta la existencia de un conjunto de clases que constituyen abstracciones que se limitan a definir comportamientos y cualidades comunes de otras clases más especificas, por lo que se identificaron un grupo de clases que serian las que podían ser probadas por tener implementaciones concretas y que además involucraran las funcionalidades más importantes de la aplicación; permitiéndonos corroborar el correcto funcionamiento de las mismas y lograr un producto con mayor calidad.

Conclusiones

Al término de la presente investigación para el desarrollo del Subsistema de Generación de Informes Selux, se concluye que:

- Qt, es sin duda alguna el framework de desarrollo de interfaces gráficas de usuario más usado en el software libre y sus facilidades de uso permitieron desarrollar una aplicación con escenarios de visualización más robustos y manejables.
- El Subsistema de Generación de Informes Selux desarrollado, es una aplicación que puede ser integrado a cualquier sistema que requiera su uso, logrando una independencia total del SCADA.
- La implementación de la solución basada en tecnologías libres representa un gran aporte a la soberanía tecnológica por concepto de compra de licencias de software.
- La realización de pruebas de unidad permitió validar la implementación de las funcionalidades desarrolladas corroborando la calidad de la aplicación y demostrando además que la misma esta lista para su uso.

Recomendaciones

- Desarrollar una funcionalidad que permita al usuario agrupar los datos del reporte por varios criterios.
- Desarrollar la exportación del reporte a otros formatos, como lo puede ser los documentos de texto con extensiones .odt ó documentos Word (.doc).
- Desarrollar un módulo de diseño, para la realización de las plantillas de los reportes de forma visual, que puedan ser exportadas a un archivo XML y cargadas luego por el generador para la visualización del reporte.

Referencias Bibliográficas

1. GD Ingenieria C.A. [En línea] [Citado el: 11 de 05 de 2010.] http://www.gdingenieria.com/SCADA.htm.

2. Cosas Libres. [En línea] [Citado el: 18 de 2 de 2010.] http://www.cosaslibres.com/software.html.

3. **Leyva, Ernesto.** *Implementación del módulo de diseño de reportes para el SCADA Guardián del ALBA.* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2009.

4. **Ciudad Ricardo, Febe Ángel.** *Utilización del Patrón Modelo – Vista – Controlador (MVC).* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n.

Bibliografía Consultada

1. **Ciudad Ricardo, Febe Ángel.** *Utilización del Patrón Modelo – Vista – Controlador (MVC).* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n.

2. SLAM-C++. [En línea] [Citado el: 20 de 03 de 2010.] http://slam-c.nireblog.com/cat/tecnologia.

3. **Zaragoza, Bernardo y Bacallao, Raudi Agdel.** *Implementación de un módulo de generación de reportes para sistemas de supervisión, control y adquisición de datos.* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2008.

4. **Jorge, Osmany.** *Generador de informes para "SCADA Nacional PDVSA".* Ciudad de la Habana, Universidad de las Ciencias Informáticas : s.n., 2007.

5. **Agata.org.** Agata Report. [En línea] [Citado el: 15 de 01 de 2010.] http://www.agata.org.br/us/index.php.

6. **Pressman, R. S.** *Ingeniería de Software. Un enfoque práctico.* La Habana : s.n., 2005.

Glosario de Términos

- **HMI** (Human Machine Interface ó Interfaz Hombre Máquina) se define como un panel a través del cual el operador es capaz de controlar la maquinaria y ver diferentes procesos en una planta.
- **SCADA** (Supervisory Control and Data Adquisition) por sus siglas en inglés; es un sistema completo que incluye HMIs y además es capaz de registrar datos, generar alarmas y administrar un sistema de control distribuido a través de una red de hardware.
- **STL** (Standard Template Library) es una biblioteca de software incluida parcialmente en la biblioteca estándar del C + +. Proporciona contenedores , iteradores y algoritmos.
- **Mac OS X v10.4** denominada Tiger es la quinta versión del sistema operativo de Apple, Mac OS X, para computadoras de escritorio y servidores Macintosh.
- **MinGW** (Minimalist GNU for Windows) es una versión nativa del compilador GCC para la plataforma Windows además de un conjunto de bibliotecas, archivos de encabezado, y herramientas de desarrollo, lo cual proporciona un ambiente completo para programar aplicaciones nativas en Windows.
- **PDF** (*Portable Document Format*, formato de documento portátil) es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.
- **CSV:** (*Comma-Separated Values*) es un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.
- **GUI:** Graphic User Interface o Interfaz Gráfica de Usuario, es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.
- **XML:** Son las siglas de Extensible Markup Language, Lenguaje de marcas extensible, es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos
- **XLS:** Principal del formato de hoja de cálculo que contiene los datos en hojas de cálculo, gráficos y macros.

- **HTML**, siglas de **HyperText Markup Language** (*Lenguaje de Marcado de Hipertexto*), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.
- **RTF:** (Rich Text Format ó Formato de Texto Enriquecido) es un formato de archivo desarrollado por Microsoft en 1987 para el intercambio de documentos multiplataforma. La mayoría de procesadores de texto son capaces de leer y escribir documentos RTF.
- **TXT:** Formato de los archivos de texto plano (en inglés plain text) que están compuestos únicamente por texto sin formato, sólo caracteres.
- **JDBC:** Java Database Connectivity, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.
- **DocBook:** Es un aplicación del estándar SGML/XML e incluye una definición de tipo de documento propia y que se utiliza de manera más destacada en el área de la documentación técnica, permite crear documentos en un formato neutro, independiente de la presentación.
- **FreeBSD:** Es un sistema operativo libre para computadoras basado en las CPU de arquitectura Intel.
- **ECJ:** Es un compilador de código abierto utilizado por el IDE de Java que se entrega como parte del Eclipse.
- **VisualAge:** Nombre de la familia de equipo de Entorno de Desarrollo Integrado de IBM(International Business Machines), que incluye soporte para múltiples lenguajes de programación .
- **OpenGL:** (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
- **SDL:** Simple DirectMedia Layer es un conjunto de bibliotecas desarrolladas con el lenguaje C que proporcionan funciones básicas para realizar operaciones de dibujado 2D, gestión de efectos de sonido y música, y carga y gestión de imágenes.

Índice de Figuras.

Índice de Tablas.

Anexos

Anexo I

1. Cuadro resumen de las características de las bibliotecas gráficas Qt y wxWidgets.

Características	Qt	*wxWidgets*
Objetos	Se parte de la base de que Qt es una herramienta desarrollada por completo en C++, por lo tanto brinda soporte para construcciones totalmente orientadas a objetos sin sacrificar eficiencia en tiempo de ejecución. Evidentemente toma todas las características del lenguaje C++, luego todos los mecanismos inherentes a C++ han sido adoptados por esta biblioteca e incluso mejorados.	Su diseño orientado a objetos no es el mejor que uno pueda ver. A veces abusa de la utilización de macros para realizar ciertas operaciones (como las tablas de eventos), aunque esto hace que codificar sea más fácil, también complica la labor de depuración, especialmente para los mismos desarrolladores de wxWidgets.
Clases	Está implementado Orientado a Objetos por lo que incluye no solo clases que tienen que ver con las GUI sino también estructuras de almacenamiento como son listas, pilas, colas.	Presenta una amplia gama de clases y controles que incluye no solo clases que tienen que ver con la GUI sino otras clases referentes a base de datos y programación multihilos.
Herencia	Los *QObjets* se organizan en árboles de objetos. Cuando se crea un *QObject* con otro objeto como padre se añade al a lista de children() (hijos) del padre y es borrado cuando el padre lo sea. A la vista salta que esta aproximación encaja perfectamente con las necesidades de los objetos GUI. Por ejemplo, un *QAccel* (acelerador de teclado) es un hijo de una ventana importante, luego, cuando el usuario cierra la ventana, el acelerador también es destruido.	Presenta la herencia entre objetos de los diferentes lenguajes que soporta aparte del C++ como son Python , Perl, Java.

Anexos.

Comprobación de Tipos	Para la comprobación de tipos utiliza los distintos mecanismos elaborados por C++ como static_cast, dinamic_cast.	Para la comprobación de tipos utiliza el distinto mecanismo de los lenguajes que soporta.
Privacidad	La privacidad de los datos al igual que en C++ es controlada por el programador pudiendo definir la visibilidad de los objetos: public, protected, prívate.	Es controlada por el programador según el lenguaje soportado.
Señales	Qt implementa un modelo de componentes que permite a los programas cargar dinámicamente objetos en cualquier punto durante el tiempo de ejecución. La comunicación mediante ranuras y señales permite escribir componentes independientemente del resto y unirlos con apenas unas líneas de código.	wxWidgets tiene un **sistema de eventos** similar a los mapas de mensaje de MFC (aunque más elegante y poderoso), que permite que los **eventos** se asocien con **funciones miembro** de manera *estática* (en tiempo de compilación) o *dinámica* (en tiempo de ejecución).
Excepciones	Se lleva a cabo el tratamiento de excepciones mediante clases diseñadas con este fin.	Debido a su antigüedad, no soporta características modernas como pude ser el manejo de excepciones de la STL.
Documentación	Posee una extensa documentación y una amplia comunidad de desarrollo que la mantiene. Además, entre sus herramientas se encuentra el Qt Assistant que permite el acceso rápido a la documentación.	Su documentación no es abundante y la mayoría se encuentra solo en ingles.
Portabilidad	Es un framework multiplataforma, disponible para plataformas como Windows, Mac, GNU/Linux, Solaris, HP-UX, UNIX, etc.	Se integra de forma óptima y resulta muy portable entre distintos sistemas operativos. Está disponible para Windows, MacOS, GTK+, Motif, OpenVMS y OS/2, en plataformas móviles como Microsoft Pocket PC, Palm OS.

Anexos.

Anexo II

Ejemplos de Reportes generados con los reporteadores actuales analizados.

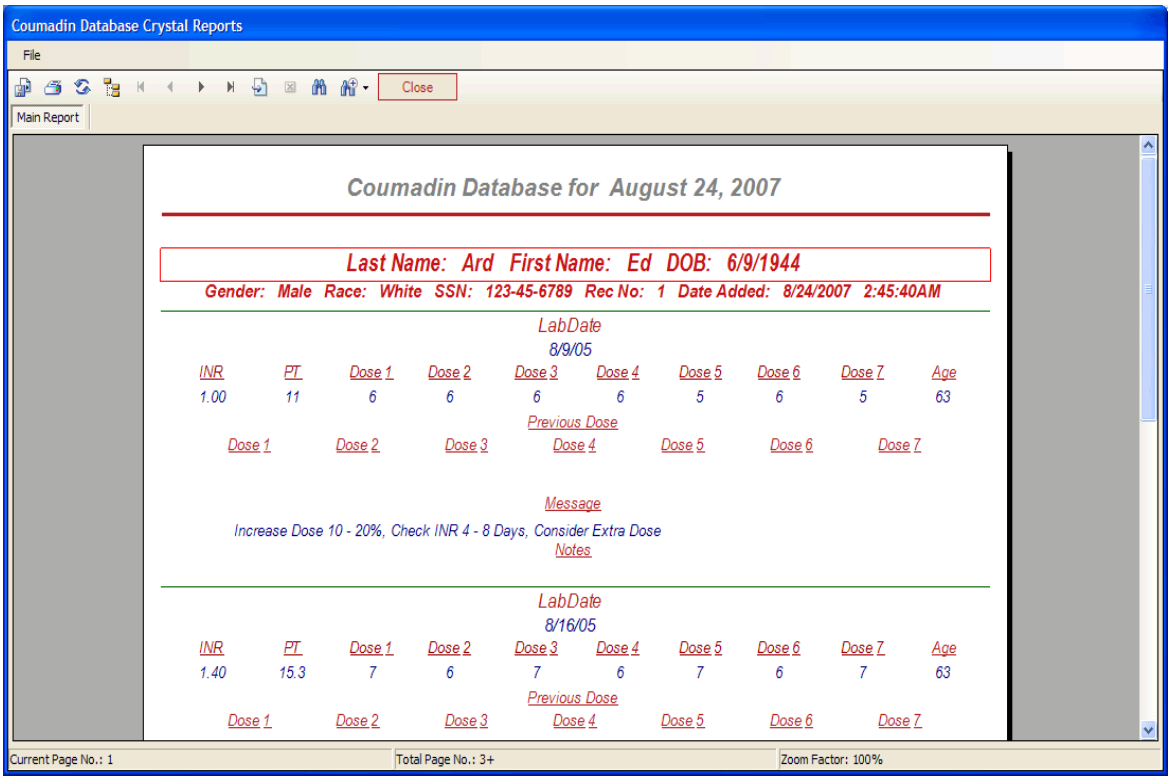

Fig. 16: Ejemplo de reporte generado con Cristal Report

Anexos.

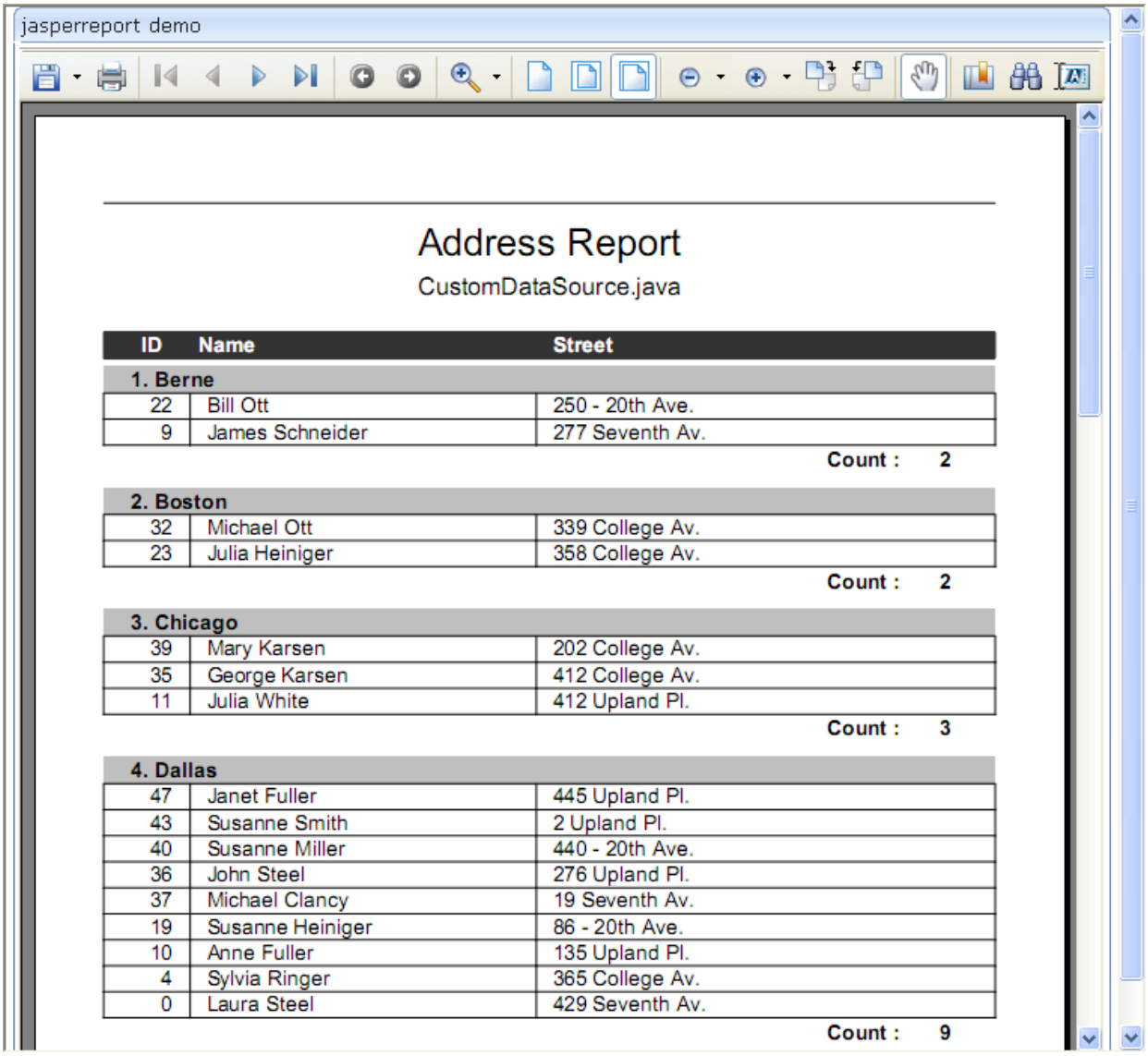

Fig.17: Ejemplo de reporte generado con el JasperReport.

Anexos.

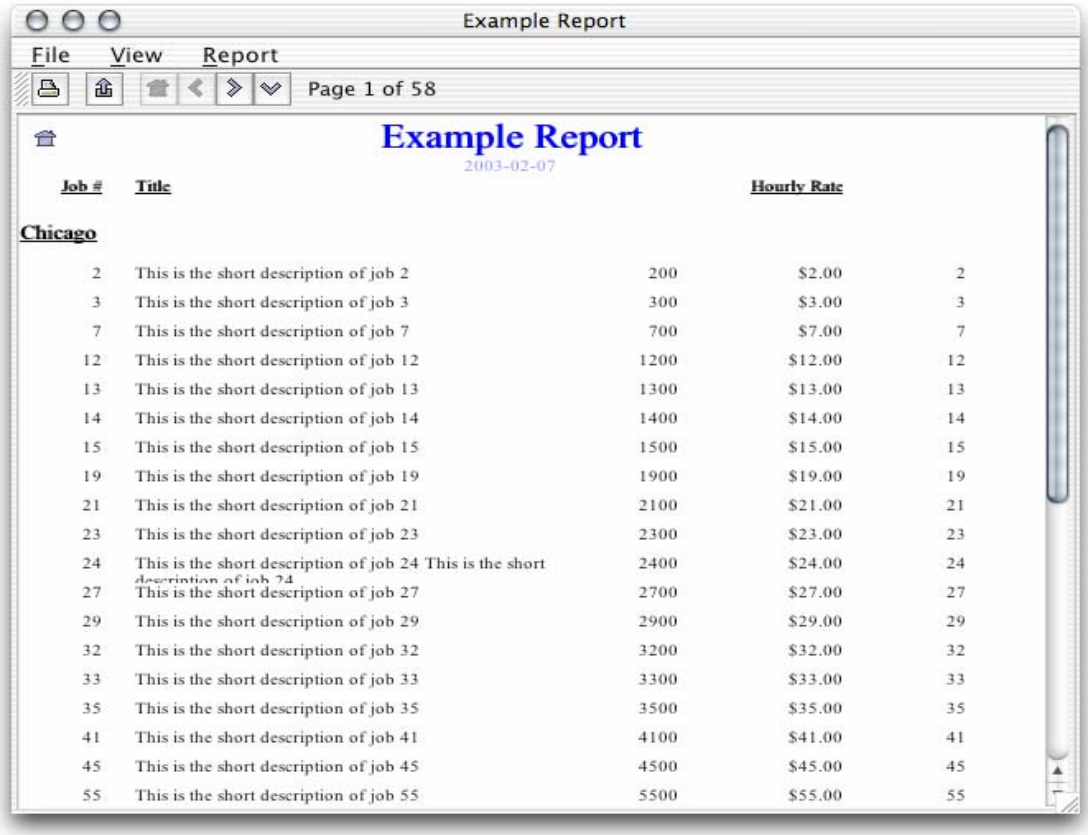

Fig. 18: Ejemplo de reporte obtenido con DataVision.