Universidad de las Ciencias Informáticas

Facultad 5

Título: "Módulo integrador de reportes en el SCADA Guardián del Alba"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Alberto Dávila Morejón

Tutor:

Ing. Ernesto Leyva Barrero

Co Tutores:

Ing. Yorji Pérez Hernández

Ing. Raudi Agdel Bacallao Sánchez

La Habana, junio 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Firma del autor	**Firma del tutor**
Alberto Dávila Morejón	**Ernesto Leyva Barrero**

DATOS DE CONTACTO

Tutor

Ing. Ernesto Leyva Barrero

Graduado de la especialidad de Ingeniero en Ciencias Informáticas desde hace tres años, profesor con categoría docente de instructor con un año de experiencia y con más de cuatro años trabajando en temas relacionados con la presente investigación.

Co Tutores:

Ing. Yorji Pérez Hernández

Graduado de la especialidad de Ingeniero en la Universidad de las Ciencias Informáticas desde hace cuatro años, profesor con categoría docente de instructor con dos años de experiencia y con más de cinco años trabajando en temas relacionados con el presente trabajo.

Ing. Raudi Agdel Bacallao Sánchez

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI), profesor con categoría docente de instructor con dos años de experiencia y con más de cinco años trabajando en temas relacionados con la presente investigación y en el desarrollo de software.

AGRADECIMIENTOS

A mis compañeros y compañeras de aula y apartamento que en el transcurso de estos años de universidad me han brindado su ayuda incondicional, los que están y los que ya se han ido.

A mis compañeros de la línea HMI que me han brindado apoyo y su ayuda en la realización de este trabajo de diploma y en otras actividades relacionadas con el mundo del software.

A mis tutores por haber confiado en mí y haberme guiado en la realización del presente trabajo, siempre buscando mi satisfacción como profesional y el aumento de mis conocimientos.

A mi familia por haberme apoyado siempre e impulsarme a estudiar cada día más y a superarme para formarme como un verdadero profesional.

A mis verdaderos amigos que aunque no son muchos siempre están ahí para lo que necesite y me han brindado su amistad incondicionalmente.

A Yoly por haberme ayudado tanto en los primeros años de universidad y brindarme su apoyo y confianza.

DEDICATORIA

A mi madre, por sacrificarse tanto y hacer de mí un hombre de bien, estando siempre en mi vida y participando en cada una de las decisiones importantes que he tenido que tomar.

A mi familia en general, mis abuelos y abuelas, mi padre, mis hermanas que son 4, mis tíos y tías, mis otras madres, mis primos, vecinos y amigos, mi padrastro, mi suegra, mi familia de Santiago de Cuba.

En especial dedico este trabajo a mi novia Yoly por haberme apoyado tanto y soportarme durante estos cinco largos años, y espero que me soporte por muchos más.

RESUMEN

A partir del convenio entre Cuba y la República Bolivariana de Venezuela y bajo la rectoría de las empresas ALBET-PDVSA surge un proyecto de software nombrado SCADA Guardián del ALBA, un sistema de supervisión, control y adquisición de datos, cuyo principal objetivo es automatizar el proceso productivo de la empresa Petróleos de Venezuela (PDVSA). Para este sistema es necesario tener un módulo que se encargue del diseño y visualización de los reportes relacionados con los datos históricos almacenados en las bases de datos del SCADA. Buscando dar solución a este problema se realizó el presente trabajo de diploma cuyo principal objetivo es implementar un módulo generador de reportes sobre plataforma de software libre, capaz de interactuar con los correspondientes módulos del SCADA y que permita la configuración y visualización de los reportes deseados integrando nuevas tecnologías.

Para lograr alcanzar el objetivo de este trabajo, se realizó un estudio minucioso de las principales herramientas generadoras de reportes desarrolladas usando el software libre como requisito de desarrollo, además se efectuó un análisis detallado de las tecnologías de desarrollo más usadas y las principales metodologías de desarrollo de software.

PALABRAS CLAVE

Diseño, generación, reportes, SCADA, software.

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

Históricamente, la necesidad de aumentar los niveles de eficiencia, minimizar los costos y optimizar los procesos de producción, ha sido un objetivo primordial para cualquier industria. Actualmente, uno de los mecanismos más importantes para dar solución a estas necesidades, es la automatización de los distintos procesos que componen la cadena de producción, por tal motivo la necesidad de automatizar operaciones resulta prioritaria para cualquier industria donde se hayan invertido importantes recursos. El advenimiento de los nuevos avances tecnológicos llevados a cabo en las últimas décadas ha propiciado soluciones eficientes al problema. Los ordenadores personales agregaron la capacidad de programar tareas para realizar funciones más complejas y, a su vez, se han ido perfeccionando y adaptando a las condiciones que imponen el paso de las Tecnologías de la Información y las Comunicaciones (TIC) en el mundo.

Un ejemplo de esto, son los Sistemas de Control Supervisión y Adquisición de datos (SCADA) que en sus inicios fueron diseñados para cumplir funciones específicas como el control de las producciones, interactuando directamente con los dispositivos automáticos (controladores autónomos y autómatas programables), a través de redes LAN o buses especiales, y visualizando en tiempo real el comportamiento del flujo productivo.

En la Universidad de las Ciencias Informáticas se desarrollan numerosos proyectos productivos sobre automatización de empresas e instituciones tanto dentro del país como en el exterior. Tal es el caso del proyecto SCADA Guardián del Alba, desarrollado por el Centro de Informática Industrial (CEDIN) perteneciente a la Facultad 5 para la puesta en funcionamiento en las instalaciones productivas de la empresa venezolana Petróleos de Venezuela (PDVSA).

Son muchas las funcionalidades de estos sistemas y entre las más básicas está la de proporcionar toda la información que se genera en el proceso productivo a diversos usuarios relacionados directamente con el control y supervisión de dichos procesos, así como a otros pertenecientes a niveles superiores dentro y fuera de la empresa como pueden ser control de calidad, supervisión o mantenimiento. En las bases de datos históricas se registran los valores instantáneos de las variables de proceso, los eventos y la configuración del sistema, entre otras. Para muchos interesados en comportamientos más globales del sistema como estadísticas del proceso, detección de fugas y deterioro de indicadores productivos, esta vasta información carece de valor, a menos que sea procesada y presentada de forma más adecuada. En este punto surge la necesidad de emitir reportes que consoliden la información adquirida para entregarla

en un formato determinado, de tal manera que sea útil al usuario que va dirigida. Es una realidad que la generación de reportes constituye una actividad crítica en muchas industrias, ya que se dedican largas jornadas delante del ordenador para organizar la información, o hacer agrupaciones y conseguir que todo salga correctamente impreso.

En el proyecto SCADA Guardián del ALBA se desarrolló un sistema para la gestión de reportes implementado sobre la biblioteca gráfica GTK (GIMP Tolkit), pero es un sistema que consume gran cantidad de recursos, además de no ser capaz de asimilar nuevas funcionalidades como la portabilidad y hacer difícil el soporte, la corrección de errores, así como la realización de mejoras en el diseño. En el proyecto se encuentra en desarrollo un nuevo editor y visualizador de HMI (Interfaz Hombre Máquina) usando la biblioteca gráfica QT (QT Tolkit). Este nuevo software pretende a partir de la utilización del mecanismo de extensiones ser un software dinámico y flexible, posibilitando una fácil configuración de sus módulos y una mayor integración a otros procesos industriales y se hace difícil la integración del generador de reportes existente a este nuevo sistema.

De lo antes mencionado surge la necesidad de desarrollar un módulo capaz de integrar la configuración y visualización de reportes en el actual HMI del SCADA Guardián del Alba y así darle solución a la imposibilidad de conocer la información detallada del proceso industrial en las plantas de PDVSA para la toma de decisiones por parte del personal de la empresa.

Para dar solución a esta problemática se tiene que responder la presente interrogante como **problema científico**: ¿Cómo diseñar un reporte desde el editor de HMI y mostrarlo en el visualizador HMI?

El **objeto de estudio** en la presente investigación es la generación de reportes en procesos industriales, dentro del cual se define como **campo de acción** la generación de reportes usando el mecanismo de extensiones del actual HMI.

El **objetivo** del trabajo es Desarrollar un módulo que permita la integración del diseño y la visualización de las plantillas de reportes en el HMI del SCADA Guardián del Alba.

Teniendo como **idea a defender** la siguiente:

Desarrollando un módulo para la integración de la configuración y visualización de reportes se logrará una mejor visualización de los datos almacenados en la base de datos histórica del SCADA Guardián del ALBA además de lograr una mayor portabilidad de la información de interés para el personal de PDVSA y así contribuir a la toma de decisiones en la empresa.

El desarrollo de este subsistema y su puesta en explotación de conjunto con el sistema SCADA permitirá que se minimice el tiempo de trabajo para llevar a cabo cualquier operación que necesite obtener información del proceso industrial, así como posibilitar una mayor portabilidad de los datos. Además permitirá la configuración personalizada de cualquier reporte, los cuales una vez generados en los formatos pertinentes serán de vital importancia en la toma de decisiones.

De acuerdo al objetivo expuesto se definen las siguientes **Tareas de Investigación**:

- Estudio de diseñadores de reportes existentes sobre plataformas libres para la selección de uno de estos.
- Estudio de las distintas tecnologías de desarrollo sobre software libre y en especial las principales bibliotecas para el desarrollo de aplicaciones para seleccionar una de estas.
- Implementación del módulo de configuración o diseño de reportes y el módulo de visualización siguiendo la arquitectura propuesta y lograr la reutilización de los distintos componentes.
- Estudio y análisis de herramientas para la ejecución de pruebas de aceptación para su posterior aplicación.
- Realización de pruebas de aceptación que permitan definir la robustez del sistema.

Esperando obtener como **resultados**:

- Un módulo que pueda ser utilizado en el SCADA Guardián del Alba para la configuración y visualización de los reportes de producción.

Para el cumplimiento de estos objetivos se utilizan varios **métodos y técnicas en la búsqueda y procesamiento de la información** como son:

A nivel teórico:

- **Método analítico-sintético:** Para el estudio de los conceptos empleados en los sistemas de generación de reportes en los SCADA, analizando todos los documentos elaborados por desarrolladores, para la extracción de los elementos más importantes.
- **Análisis histórico–lógico:** Para la comprensión de los antecedentes y las tendencias actuales referidas a la evolución en el mundo de los sistemas de generación de reportes en los SCADA.

A nivel empírico:

- **Experimento:** Elaboración de un módulo para el HMI que posibilite la configuración y generación de reportes.

El presente documento está estructurado en tres capítulos:

- **Capítulo 1:** Fundamentación del tema.
- **Capítulo 2:** Construcción de la solución propuesta.
- **Capítulo 3:** Validación de la solución.

En cada capítulo se va a tratar acerca de:

- En el **Capítulo 1** se realiza un esbozo teórico centrado en temáticas como generadores de reportes, sistemas de Base de Datos, sistemas SCADA y las principales tendencias y tecnologías actuales que sirven de apoyo a todo el trabajo desarrollado.
- En el **Capítulo 2** se analiza e implementa la arquitectura propuesta y se explican las principales funcionalidades.
- El **Capítulo 3** se dedica a las pruebas realizadas para garantizar la robustez de la aplicación desarrollada.

CAPÍTULO 1

FUNDAMENTACIÓN DEL TEMA

Introducción

Este capítulo expone los conceptos y características generales de los generadores de reportes, y brinda una panorámica de la aplicación de estos a los sistemas SCADA. Además se muestran diferentes tecnologías para la generación de reportes sobre plataformas libres, se abordan temas acerca de las diferentes metodologías de desarrollo de software usadas en la actualidad, se habla acerca de los patrones arquitectónicos más utilizados, así como las distintas herramientas para el modelado y el desarrollo de software existentes.

1.1 Generalidades de los SCADA

Sus siglas son el acrónimo en inglés de Supervisory Control And Data Acquisition (Control, Supervisión y Adquisición de Datos) y se definen como una aplicación de software, diseñada con la finalidad de controlar y supervisar procesos a distancia. Se basa en la adquisición de datos de los procesos remotos, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática. Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento, etc (1).

Estos sistemas no son solamente herramientas operacionales, sino, también un recurso importante de información y esto se pone de manifiesto en los recursos que se exportan a otros sectores como la plataforma web y la tecnología móvil, permitiendo a usuarios externos ver lo que sucede en el proceso de producción, así como obtener distintos reportes relacionados con datos importantes. Es un aspecto principal en este tipo de programas, el tratamiento y manejo de los datos adquiridos, así como su almacenamiento (2). En la imagen se expone un esquema general:

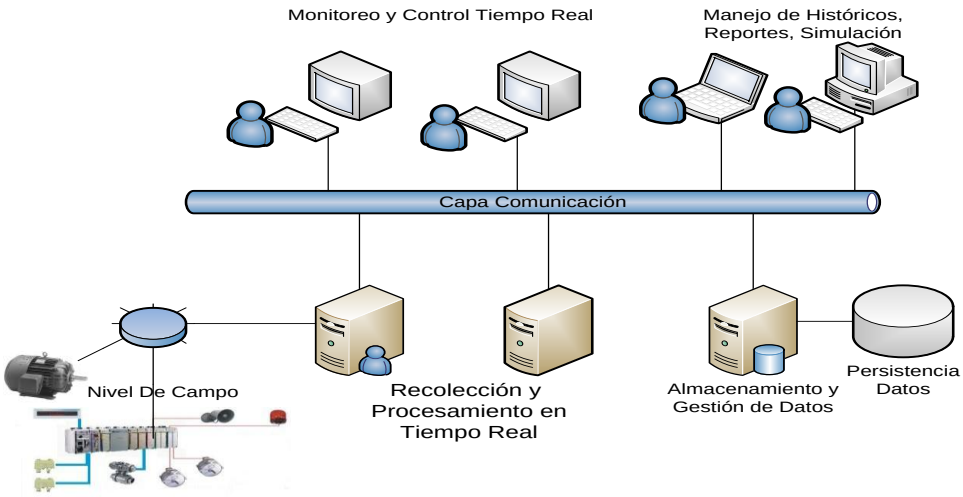

Figura 1. Esquema general de un SCADA

1.1.1 Módulos del SCADA

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

Seguridad: Provee las funcionalidades necesarias para garantizar el trabajo autorizado por usuarios, además brinda las herramientas para la protección contra ataques maliciosos o involuntarios al sistema.

Manejadores: El módulo de manejadores de dispositivos asegura la comunicación con los dispositivos de campo a través de una interfaz genérica única para todos los protocolos y bibliotecas dinámicas.

Configuración: Este módulo se encarga de almacenar, persistir y suministrar, la Información base para el funcionamiento de los demás módulos como BDH (Base de Datos Histórica), Generación de Reportes, HMI entre otros.

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

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

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

Núcleo de procesamiento de datos (Recolector): Es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.

Generación de reportes: Es la herramienta que permite emitir reportes en un formato personalizado de tal manera que le sea útil al personal al cual va dirigido.

1.2 Bases de Datos

Se le llama Base de Datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto. Estas pueden ser de diversos tipos, desde un pequeño fichero casero para ordenar libros y revistas por clasificación alfabética hasta una compleja base que contenga datos de índole gubernamental en un Estado u organismo internacional. Recientemente, el término base de datos comenzó a utilizarse casi exclusivamente en referencia a bases construidas a partir de software informático, que permiten una más fácil y rápida organización de los datos. Las bases de datos informáticas pueden crearse a partir de software o incluso de forma online usando Internet. En cualquier caso, las funcionalidades disponibles son prácticamente ilimitadas. (3)

1.2.1 Sistemas gestores de Bases de Datos

Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS), son esenciales para el adecuado funcionamiento y manipulación de la información contenida en las bases de datos. Se puede definir como: El Conjunto de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos, manteniendo su integridad, confidencialidad y seguridad (4).

Las principales funciones de un SGDB son la descripción, manipulación y utilización de los datos.

Descripción: Incluye la descripción de los elementos de datos, su estructura, sus interrelaciones, sus validaciones. Tanto a nivel externo como lógico global e interno esta descripción es realizada mediante un LDD o Lenguaje de Descripción de Datos.

Manipulación: Permite Buscar, Añadir, Suprimir y Modificar los datos contenidos en la Base de Datos.

La manipulación misma supone definir un criterio de selección, definir la estructura lógica a recuperar, acceder a la estructura física. Esta manipulación es realizada mediante un LMD o Lenguaje de Manipulación de Datos.

Utilización: La utilización permite acceder a la base de datos, no a nivel de datos sino a la base como tal, para lo cual, reúne las interfaces de los usuarios y suministra procedimientos para el administrador.

En la actualidad existen diversos gestores de base datos, privativos y libres, entre los más conocidos se encuentran MySQL, PostgresSQL, Microsoft SQL Server y Oracle Database, aunque existen otros como Adaptive Server Enterprise (ASE), Interbase, Firebird, SQLite, DB2 Express-C,Apache Derby, MariaDB y Drizzle.

1.3 Generalidades sobre los generadores de Reportes

Un sistema de Generación de Reportes es un programa que permite crear reportes 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. (5)

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 (6).

Los generadores de reportes basan su funcionamiento en dos partes fundamentales, un diseñador o editor de reportes y un motor de generación:

***Diseñador*:** Es una herramienta que permite 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, persistirán en la plantilla estas características posibilitando que en el reporte generado se mantenga la apariencia del diseño y contenga los datos en su interior (6).

Motor de Generación: Es una herramienta de creación de reportes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros en formato PDF, HTML, CSV entre otros. 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 (6).

En la siguiente imagen se muestra un esquema general del funcionamiento de un generador de reportes.

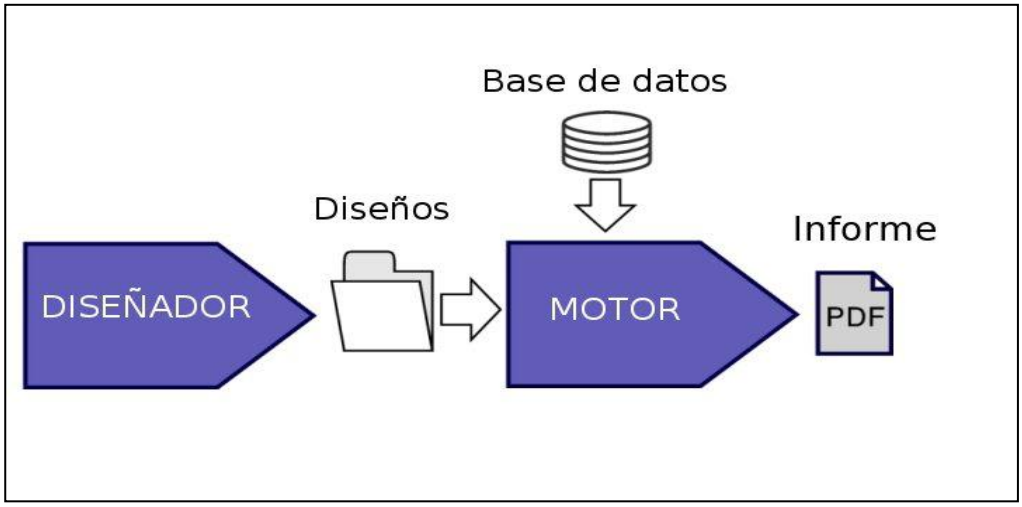

Figura 2. Esquema general de un generador de reportes. Esquema de funcionamiento.

Es necesario destacar que la parte más importante en un generador de reportes es el motor de generación, existiendo actualmente generadores de reportes que no poseen un editor, y el diseño de la plantilla es realizado manualmente, utilizando solamente una plantilla con el formato por defecto, lo cual se convierte en una tarea compleja. Es por ello que se hace necesaria una herramienta que permita realizar diseños fácilmente.

1.4 Los reportes en los SCADA

La generación de reportes en los sistemas SCADA siempre ha sido un punto débil. Muchas veces se hace necesario contar con mecanismo capaces de generar reportes referentes a la adquisición y registro de datos, generación de alarmas, entre otros, siendo esto una parte fundamental para la toma de decisiones en las distintas industrias.

Existen en el mundo una gran diversidad de proyectos inmersos en la creación de generadores de reportes, aunque es cierto que en su mayoría están enfocados en la generación de reportes para todos los sectores y no específicamente en ambientes industriales. Los generadores de reportes aplicados específicamente a los sistemas SCADA deben cumplir ciertos parámetros, entre ellos la seguridad, ya que cada usuario debe ver solamente lo que se le está permitido.

1.5 Tendencias para el desarrollo

Enfrentar el desarrollo de un producto de software no solo consiste en diseñar e implementar una arquitectura robusta y flexible, sino que va más allá, teniendo como punto de partida un estudio minucioso de la tendencia de desarrollo que se debe seguir, apoyándose en las características propias de cada proyecto de software. Una de las tendencias que existe actualmente consiste en incorporar o integrar la solución que se desea realizar a herramientas ya elaboradas aprovechándose de las libertades que otorga el software libre. Otra tendencia que se apoya igualmente en las bondades del software libre, consiste en reutilizar un producto ya existente y adaptarlo a nuestras necesidades, haciéndole las modificaciones pertinentes posibilitando que se disponga de suficiente tiempo como para entender la solución a reutilizar y luego comenzar a incorporarle las nuevas funcionalidades, logrando al final un software estable y con un mayor tiempo de desarrollo. Una última tendencia utilizada en la mayoría de los productos de software se basa en la utilización de las diferentes bibliotecas de desarrollo de software apoyándose en las herramientas básicas que brinda la misma, teniendo con esto todo el código desarrollado, toda la documentación y el derecho de autoría sobre la solución.

Para la implementación del módulo se escoge desarrollar a partir de elementos ya existentes logrando la reutilización de una herramienta que lleva ya algún tiempo de desarrollo y que brinda funcionalidades, a las cuales, si se desarrolla la aplicación completamente sería difícil llegar, además que para desarrollar una aplicación para la generación de reportes se necesitaría un equipo de desarrollo comprometido y con altos conocimientos.

1.6 Generadores de reportes

Resulta muy necesario realizar un estudio minucioso de las distintas herramientas generadoras de reportes con el objetivo de escoger una y reutilizarla en el presente trabajo. El estudio se enmarca en la búsqueda de los generadores realizados sobre plataformas libres y liberados bajo licencias compatibles con el actual desarrollo del proyecto. El generador escogido luego de haber realizado el estudio debe tener la capacidad de tomar la información almacenada históricamente en las distintas bases de datos del SCADA, generar reportes en diversos formatos y adaptarse a los dos ambientes de ejecución del HMI del SCADA, edición y visualización.

1.6.1 Report Manager Designer

Es una aplicación de generación de informes y un conjunto de componentes para Delphi, Builder y Kylix, aunque también puede utilizarse desde otros entornos de desarrollo con el componente ActiveX incluido, como Visual Basic, Visual FoxPro y cualquier lenguaje de Visual Studio.Net. Se proporciona una biblioteca dinámica estándar con funciones para su uso con cualquier lenguaje como GNU C. Es un producto opensource bajo el modelo MPL (Licencia Pública de Mozilla), incluyendo permiso de uso en aplicaciones GPL, por lo que se puede usar en aplicaciones comerciales, pero cualquier mejora introducida en el motor de impresión debe ser publicada bajo esta licencia. Funciona en sistemas operativos como Windows y Linux. Cuenta con un diseñador de reportes y permite el guardado solamente en archivos con formato PDF, HTML y XLS. Dispone de múltiples características como bibliotecas de informes, meta-archivos, fuentes de impresora, secciones externas y subinformes en cascada. Incluye presentación preliminar, diálogo de impresión, opciones del informe entre otras características. Posee conectores a bases de datos como Oracle, SQL Server, MySQL, PostgreSQL, Firebird entre otras (7).

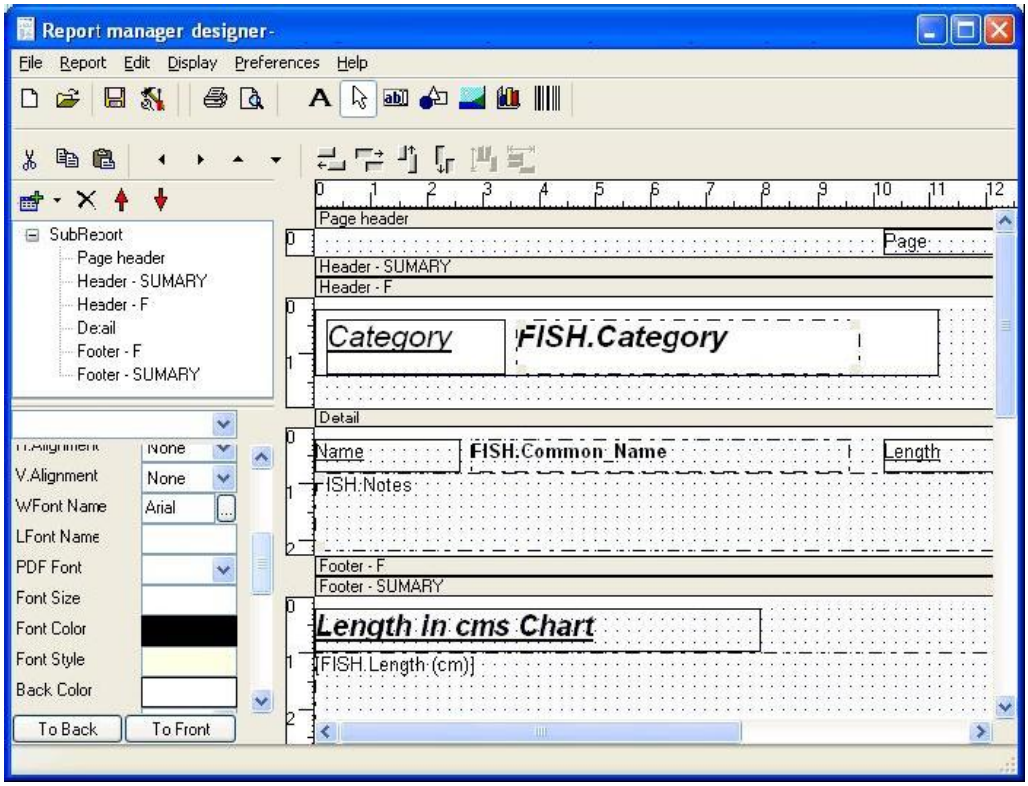

Figura 3. Pantalla de la aplicación Report Manager Designer

1.6.2 Kugar

Es una herramienta KDE para generar reportes de calidad profesional que pueden ser vistos o impresos. Incluye un visor de reportes independientes y un visor de reportes KPart. Esto último significa que cualquier aplicación KDE puede integrar la funcionalidad de visualización de reportes y estos se pueden ver utilizando el navegador Konqueror. Esta herramienta funciona a partir de la fusión de datos con una plantilla, produciendo el informe final, la plantilla se especifica mediante un XML. Kugar posibilita hacer referencia a una plantilla a través de una URL, lo que permite a las empresas crear una gestión centralizada de plantillas. Este software es liberado bajo la licencia GPL (8).

El diseñador de reportes de Kugar, Kudesigner, es una aplicación del tipo WYSIWYG (*what you see is what you get*), lo que ves es lo que obtienes, así el tamaño de la página del reporte define las dimensiones del reporte en la pantalla. Mediante el uso de este diseñador se logra la creación y modificación de las plantillas de reportes, y la colocación interactiva de las secciones de reporte y de los elementos sobre dichas secciones. Este trae consigo toda una gama de plantillas para a partir de ellas empezar a trabajar. Están disponibles para la realización de diseños diferentes elementos como etiquetas, campos calculados y especiales, líneas. Tanto los elementos como las secciones tienen sus propiedades y características. Esas propiedades definen parámetros geométricos, textuales y de otro tipo. Cada vez que se ubica un elemento, se aplican una serie de propiedades predeterminadas (2).

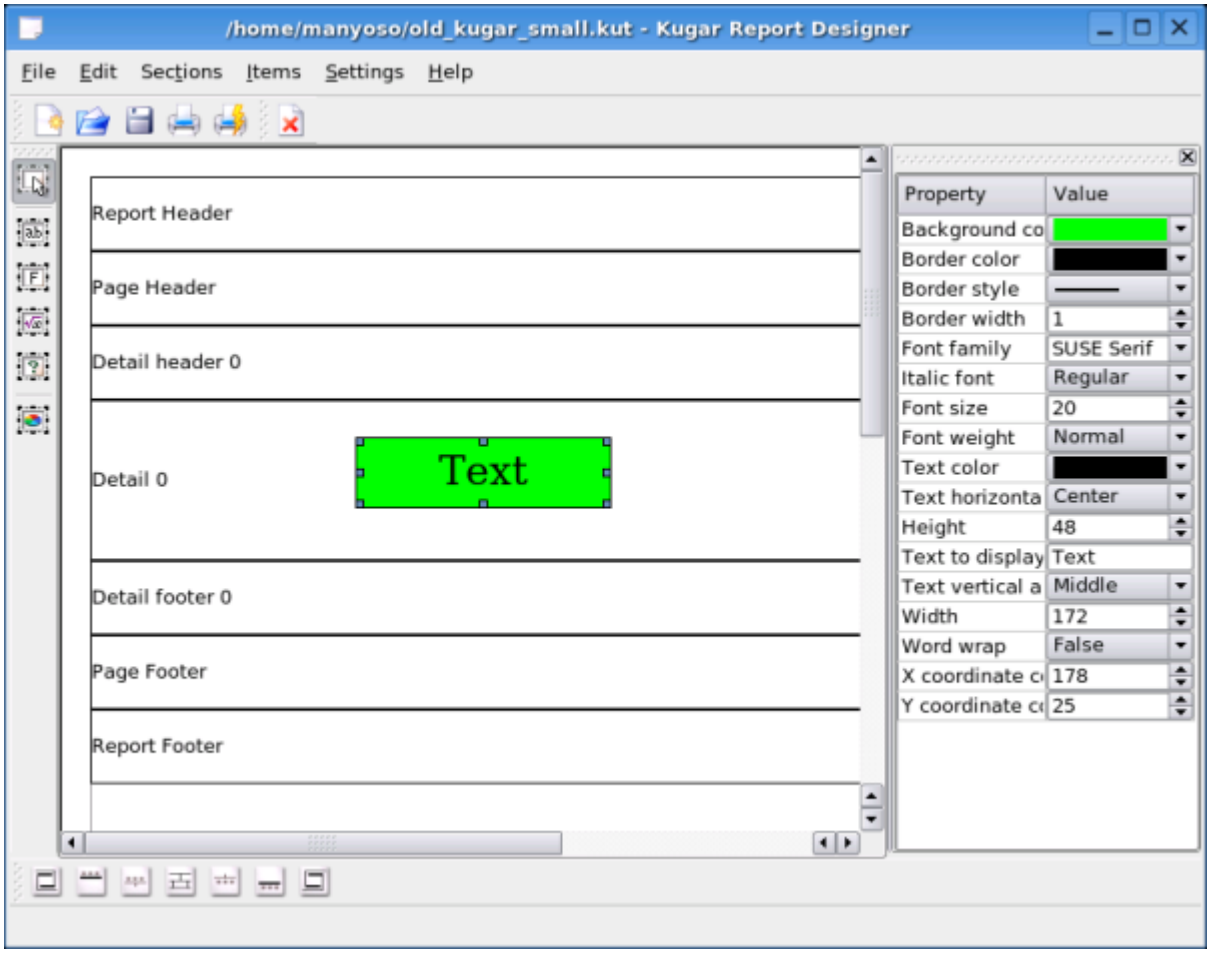

Figura 4. Vista de diseño de Kugar

1.6.3 JasperReports

Es un motor de generación de reportes de código abierto. Está escrito completamente en Java y es capaz de utilizar los datos procedentes de cualquier tipo de fuente de datos y producir documentos que se pueden visualizar, imprimir o exportar en una variedad de formatos de documentos incluyendo HTML, PDF, Excel, OpenOffice y Word (9). Esta biblioteca está liberada bajo la Licencia Pública General Reducida (LGPL).

JasperReports se usa comúnmente con iReport, un editor gráfico de código abierto para la edición de reportes. Este editor se encuentra bajo la Licencia Pública General (GPL), por lo que es Software libre. El diseñador iReport cuenta con excelente apariencia visual y permite insertar numerosos componentes

dadas las facilidades de Java. Tanto JasperReports como iReport pueden usarse en sistemas operativos como Linux y Microsoft Windows.

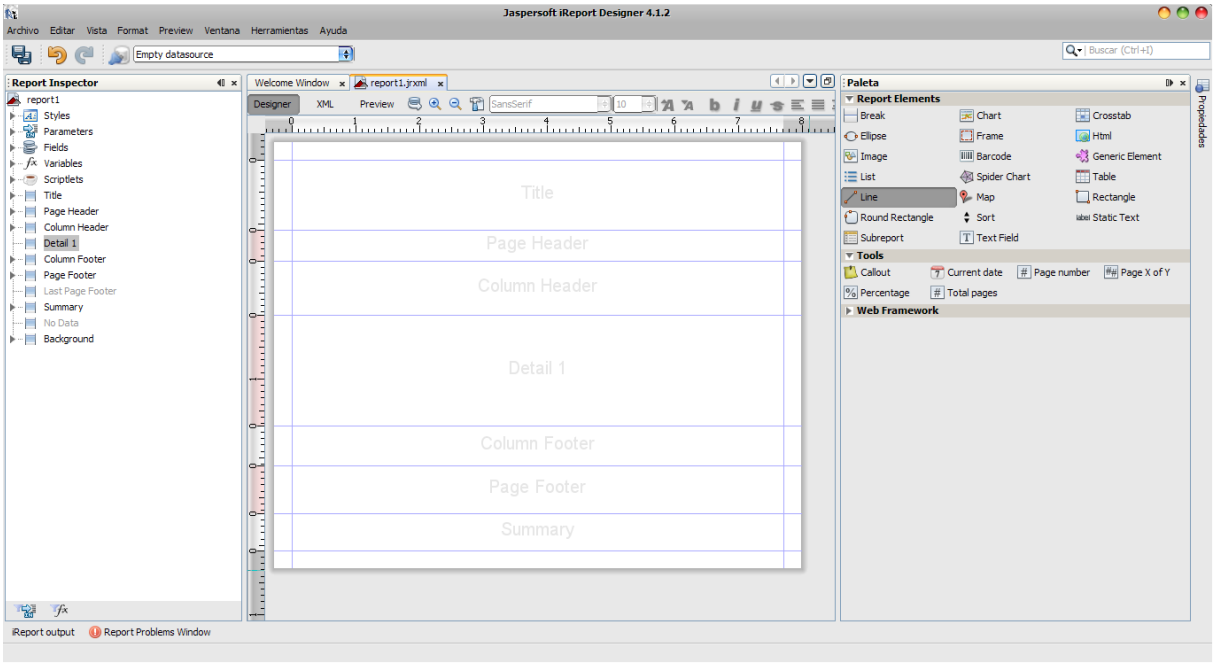

Figura 5. Ventana de diseño de reportes de iReport

1.6.4 Data Vision

Es una herramienta de código abierto similar a Crystal Reports. Los reportes pueden ser diseñados con una interfaz gráfica de usuario arrastrando y soltando los elementos con la posibilidad de visualizarlos, imprimirlos o exportarlos como HTML, XML, PDF, Excel, LaTeX2e, DocBook. Los archivos de salida producidos por LaTeX2e y DocBook a su vez pueden ser utilizados para producir el texto en formato PDF, HTML, PostScript entre otros. DataVision está escrito en Java y funciona en la mayoría de las plataformas. Puede generar informes desde bases de datos o archivos de datos de texto. Posee conectores para base de datos como Oracle, PostgreSQL, MySQL, Informix, HSQLDB, Microsoft Access entre otras. Esta herramienta almacena los reportes como archivos XML, lo que significa que no sólo se puede utilizar el editor gráfico sino que se pueden editar los reportes usando editores de texto. Su licencia es de tipo Apache 2.0, que es de software libre pero incompatible con la GPL (10).

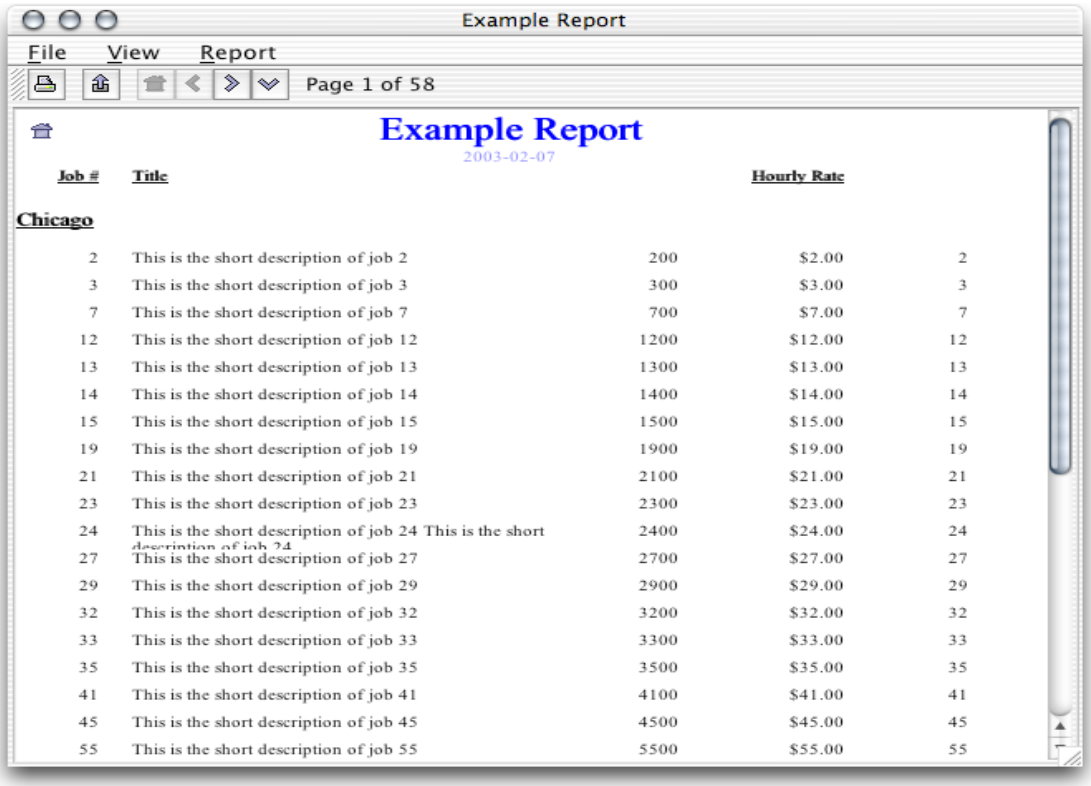

Figura 6. Ventana de visualización de reportes de DataVision

1.6.5 Selux

Es un subsistema de Generación de Reportes desarrollado en la Universidad de las Ciencias informáticas con el objetivo de lograr una nueva versión del sistema generador de reportes del HMI del SCADA Guardián del ALBA desarrollado en GTK. Es una aplicación desarrollada a partir de la utilización de la biblioteca gráfica Qt y es construida como un software externo y no como un módulo de HMI del SCADA antes mencionado.

1.6.6 Olympia

Olympia es una aplicación desarrollada sobre el framework Symfony y escrita en PHP con soporte para imágenes, gráficas y subreportes, así como varios orígenes de datos. Proporciona a los usuarios, entre otras opciones, agilizar la toma de decisiones, generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos. Está compuesto por varias aplicaciones entre las que se encuentran el Visor de reportes, Diseñador de

modelos y el Diseñador de reporte. Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos. Además, su entorno de trabajo está estructurado de forma que es difícil guiarse en la creación y generación de reportes (11).

1.6.7 Akademos

Es una aplicación desarrollada en plataforma .NET, en el 2007, en la Universidad de las Ciencias Informáticas, la cual brinda actualmente importantes servicios académicos tanto a los estudiantes como a los departamentos docentes. Cuenta con una herramienta de generación de reportes, la cual los genera con eficiencia, pero desde el punto de vista del dinamismo de la generación, se encuentra adaptada solamente al negocio de la gestión académica en la UCI, restringiendo que sean elaborados con la información contenida en la base de datos específicamente utilizada por el software, lo cual dificulta que esta herramienta sea aplicada a otros ambientes además de estar diseñada sólo para la Web (12).

1.6.8 Grato Report

Una herramienta desarrollada en el año 2008 por el proyecto alasGRATO de la Facultad 6, en la Universidad de las Ciencias Informáticas, para la generación de reportes referentes a la Química. Fue desarrollado sobre el lenguaje JAVA usando la biblioteca para la generación de reportes Dynamic JasperReports como base para su funcionamiento (11).

1.7 Selección del Generador de Reportes

Para seleccionar uno de los generadores de reportes del epígrafe anterior y reutilizarlo en la solución a desarrollar se estableció una matriz de decisión dando como puntuación mínima y máxima 1 y 5 respectivamente. Los parámetros a medir para cada generador de reportes son:

Formatos de Salida: Cantidad de formatos de salida con los cuenta.

Conexiones con Bases de Datos: Se refiere al número de conectores a bases de datos que posee el generador.

Nivel de Reutilización: Se refiere a la complejidad de reutilizar el generador de reportes en la solución, teniendo en cuenta la tecnología y los mecanismos que usa.

Editor: Cantidad de componentes que permite insertar y nivel de configuración del editor en general.

Plataformas: Sistemas operativos en los cuales puede ejecutarse.

Usabilidad: Nivel de uso para el usuario.

En la presente tabla se muestra el valor asignado a cada uno de los generadores de reportes del epígrafe anterior, además de un total de los puntos otorgados.

Generadores de Reporte	Formatos de Salida	Conexiones con Bases de Datos	Nivel de Reutilización	Editor	Plataformas	Usabilidad	Total
Report Manager Designer	3	3	4	3	2	2	17
Kugar	4	4	3	5	2	5	23
JasperReports	5	5	4	5	5	5	29
Data Vision	3	3	4	3	5	4	22
Selux	1	1	5	2	5	4	18
Olympia	1	1	1	2	5	4	14
Akademos	2	1	1	1	4	4	13
Grato Report	4	4	1	1	5	4	19

Tabla 1. Matriz de Decisión de Generadores de Reportes

La herramienta seleccionada es JasperReports ya que posee en total 29 puntos, siendo la más completa de las estudiadas, además obtuvo altas puntuaciones para cada parámetro con el cual se comparó.

1.8 Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software constituyen un pilar fundamental en el proceso de construcción de un software evitando que este vaya rumbo al fracaso. Se definen como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales (13).

En los últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos consiste en que mientras los métodos pesados intentan conseguir su objetivo común por medio de orden y documentación, los métodos ligeros (también llamados métodos ágiles) tratan de mejorar la calidad del

software por medio de la comunicación directa e inmediata entre las personas que intervienen en el proceso (14).

1.8.1 Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto de software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar (15).

1.8.2 Metodologías Ágiles

Las necesidades de un cliente pueden sufrir cambios importantes del momento de contratación de un software al momento de su entrega; y es mucho más importante satisfacer estas últimas que las primeras. Esto requiere procesos de software diferentes que en lugar de rechazar los cambios sean capaces de incorporarlos. Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables, no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado, que no vaya a variar ni en tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previos a la entrega final, lo cual agradará al cliente. Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas que puede que no curen todos los males pero harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega (15).

1.8.3 Metodologías en la actualidad

El Proceso Unificado de Rational (RUP)

Como sus siglas en inglés lo indican es Rational Unified Process, de las metodologías existentes, una de las más generales y completas. Su surgimiento se remonta a aproximadamente treinta años atrás, lo cual nos da una medida de lo bien estructurada y refinada que está. Fue creada por James Jacobson y se

puede decir que unifica los mejores elementos de las metodologías anteriores. Además está preparado para desarrollar grandes y complejos proyectos. Está enfocado al uso del paradigma orientado a objetos y utiliza a UML como su lenguaje de representación visual. Dentro de sus principales características están, que es guiado por casos de usos, que es iterativo e incremental y que centra en la arquitectura el esqueleto del producto. También es válido destacar que se divide en cuatro fases importantes (Inicio, Elaboración, Construcción y Transición) (2).

Programación Extrema (XP)

Esta metodología llamada Programación Extrema, o Extreme Programming, es otra de las existentes en la actualidad. Mientras que RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción (14).

XP propone la descripción de las funcionalidades del sistema a partir de la utilización de las historias de usuarios (HU), que son escritas por el cliente y describen los diferentes escenarios de funcionamiento del software en construcción, es por ello que se hace necesaria la presencia del cliente en el equipo de trabajo. Las iteraciones en XP generalmente son cortas buscando obtener cuanto antes un software funcional, siempre trabajando sólo en las funcionalidades requeridas para la entrega más cercana.

1.9 Selección de la Metodología

Para la selección de la metodología de desarrollo de software se estableció una matriz de decisión, en la cual se tomaron como parámetros a medir:

1. Tiempo estimado del proyecto.
2. Tamaño del Equipo de desarrollo.
3. Cantidad de artefactos generados.
4. Nivel de uso en la actualidad.

Para cada uno de estos parámetros se estableció una puntuación mínima y máxima de 1 y 5 respectivamente.

Metodologías de Desarrollo de Software	Tiempo Estimado del Proyecto	Tamaño del Equipo de desarrollo	Cantidad de Artefactos Generados	Nivel de Uso	Total
RUP	3	3	2	4	12
XP	5	4	5	4	18

Tabla 2. Matriz de Decisión de Metodologías de Desarrollo

A partir de lo antes expuesto se escoge XP como la metodología de desarrollo de software que va a regir el desarrollo del Módulo integrador de reportes en el SCADA Guardián del ALBA ya que es una metodología ligera hecha especialmente para equipos de desarrollo pequeños buscando una mayor eficiencia, un menor tiempo de desarrollo y una buena calidad del producto final.

1.10 Lenguajes

UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software orientado a objetos (OO). 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 (16).

C++

Es un lenguaje de programación basado en C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de bibliotecas. Nació para añadirle cualidades y características de las que carecía C. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción (17).

Java

Fue diseñado por James Gosling, de Sun Microsystems, en 1990, como software para dispositivos electrónicos de consumo, como calculadoras y microondas. Inicialmente se llamó Oak (roble en inglés), aunque tuvo que cambiar debido a que dicho nombre ya estaba registrado por otra empresa (18).

El lenguaje en sí mismo toma su sintaxis de C y C++ y corrige muchos de los fallos de estos. Tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador.

1.11 Biblioteca Qt

Qt es un framework de desarrollo de software que incluye clases, bibliotecas y herramientas para la producción de aplicaciones usando el lenguaje C++ de forma nativa y puede operar en varias plataformas incluyendo los sistemas Unix (Linux, MacOS X, Solaris) o incluso toda la familia de Windows. Esta poderosa herramienta permite desarrollar ricas aplicaciones gráficas e incluye soporte de nuevas tecnologías como OpenGL, XML, Bases de Datos y programación para redes.

Diversos proyectos de renombre mundial usan este framework entre los que se encuentra, el entorno de escritorio KDE, incluyendo sus aplicaciones, así como el visor de videos VLC (Video Lan). Existen diferentes versiones de Qt que posibilitan su utilización con otros lenguajes de programación como Python (PyQt), Java (Qt Jambi), Perl (PelrQt), entre otros.

1.12 Patrones Arquitectónicos

Los patrones arquitectónicos, o patrones de arquitectura definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema (19).

A continuación se describen algunos patrones arquitectónicos con el fin de utilizarlos en el proceso de desarrollo de software.

Arquitectura de tres capas

Es un estilo de programación cuyo objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos. Entre sus ventajas está, que el desarrollo se puede llevar a cabo en varios niveles y en caso que sobrevenga algún cambio solo se afecta la capa en cuestión. Además permite distribuir el trabajo de creación de una aplicación por niveles; cada grupo de trabajo está totalmente abstraído del resto de los niveles, de forma que basta con conocer la API (Interfaz de Programación de Aplicación) que existe entre niveles (20).

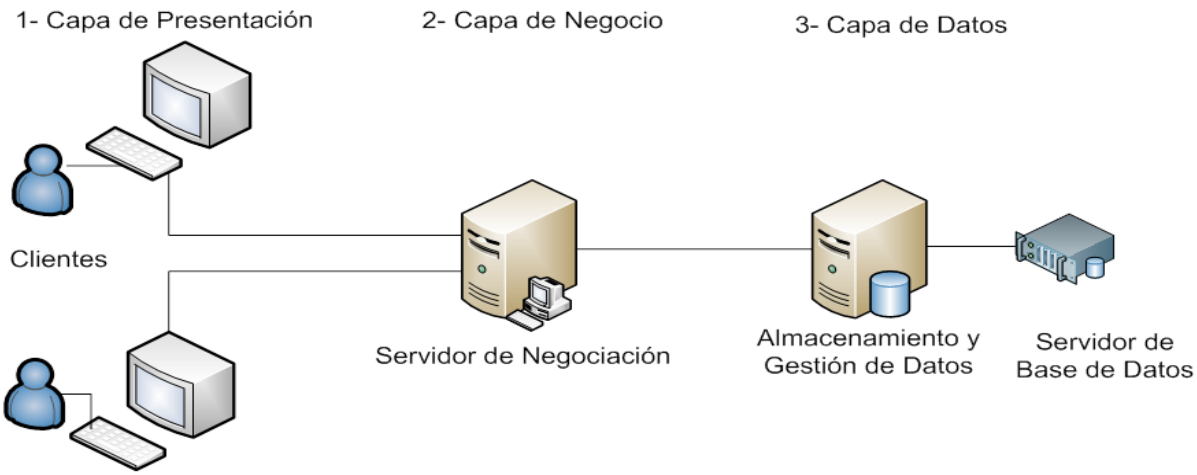

Figura 7. Esquema Arquitectura en capas

Capa de presentación

Es la capa encargada de hacer el pedido y mostrar la información al usuario, es donde se encuentra la interfaz de usuario.

Capa de negocio

Esta capa es la encargada de recibir el pedido del cliente, extrae los datos al interactuar con la capa de almacenamiento y da una respuesta al cliente, es la encargada de implementar las reglas de negocio.

Capa de datos

Almacena los datos que serán utilizados por la aplicación.

Patrón arquitectónico Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista (21).

Los elementos del patrón son los siguientes:

Modelo

- Accede a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas del negocio (las funcionalidades del sistema).
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente.

Controlador

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.

Vista

- Recibe datos del modelo y los muestra al usuario.
- Tiene un registro de su controlador asociado.

Entre las ventajas que proporciona la utilización de este patrón están:

- Es posible tener diferentes vistas para un mismo modelo.
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

1.13 IDEs

Un entorno de desarrollo integrado (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede

dar cabida a varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones. Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic entre otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse (22).

1.13.1 Eclipse

Es un Entorno Integrado de Desarrollo (IDE) de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Con este IDE se han desarrollado importantes aplicaciones 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). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent Azureus. Emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software permitiendo adicionalmente a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Phyton y trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistemas de gestión de base de datos (23).

1.13.2 Netbeans

NetBeans es una plataforma para el desarrollo de aplicaciones de escritorio usando el lenguaje Java y un entorno de desarrollo integrado (IDE) para desarrollar bajo esta plataforma, pero también admite otros lenguajes de programación como C y C++ mediante los cuales se pueden crear aplicaciones gráficas, por ejemplo usando bibliotecas como wxWidgets. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las API de NetBeans y un archivo especial (manifest) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos

pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (24).

1.14 Herramientas de modelado

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML (25).

1.15 Propuesta de solución

A partir de lo expuesto anteriormente, se procede a realizar una propuesta de solución para este trabajo, la cual consiste en desarrollar un módulo que permita integrar la generación de reportes usando la biblioteca JasperReports y su editor iReport, en el actual HMI del SCADA Guardián del Alba. El lenguaje de programación C++ va a ser el utilizado en el desarrollo de la aplicación por su portabilidad y eficiencia, así como por su alto nivel de compatibilidad con los entornos de software libre y las bibliotecas a utilizar, además de ser este el lenguaje de desarrollo del HMI. El IDE que se utilizará para el desarrollo será Eclipse por ser, además de multiplataforma, de código abierto y una potente herramienta, la usada en el proyecto productivo para el desarrollo de software haciendo uso de la extensión Qt C++. Además se utilizará Netbeans para el desarrollo sobre Java a partir de la utilización de la biblioteca JasperReports. Para el desarrollo de interfaces gráficas se usará la biblioteca gráfica Qt en su versión 4.7 usando para su diseño la extensión Qt Designer para Eclipse. Se pretende utilizar además Visual Paradigm para el modelado UML. El sistema operativo donde se realizará la solución es GNU/Linux en su distribución Debian 6.0, por su estabilidad y por la independencia tecnológica que brinda su uso.

Consideraciones Parciales

En este capítulo se hizo un estudio de las características fundamentales de los generadores de reportes, enfocándose en las plataformas de software libre. Se expusieron conceptos, tendencias y tecnologías actuales que conllevan al desarrollo de aplicaciones robustas y estables. Al finalizar el capítulo se hizo una propuesta de solución la cual comenzará a ser implementada en el siguiente capítulo.

CAPÍTULO 2

CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se describen brevemente las funcionalidades del sistema en desarrollo a partir de la propuesta de solución expuesta en el capítulo anterior, así como la estrategia a seguir para la implementación de las mismas, adecuándose a los plazos de tiempo impuestos por el cliente. Además se muestran las historias de usuarios que fueron elaboradas y se definen tres iteraciones por los programadores, elaborándose el Plan de Entrega para cada versión. Se definen las tareas generadas a partir del desarrollo de las historias de usuarios durante las tres iteraciones planificadas y se puntualizan los artefactos que dirigen la descripción del sistema.

2.1 Exploración

Para comenzar la producción de un producto de software es necesario asegurarse de que es factible y posible hacerlo. Se debe tener confianza en que las herramientas seleccionadas ayudarán a la culminación del trabajo y sobre todo se debe creer que una vez que el código se haga, éste puede utilizarse. Cada miembro del equipo debe confiar en sus propias habilidades y en las habilidades de los demás. La fase de exploración como propone XP ayuda a resolver todos estos conflictos. Es en ella donde los clientes plantean a grandes rasgos las historias de usuario que son de interés para el producto. Durante la misma, los desarrolladores interactúan con las herramientas a utilizar durante todo el proceso, exploran activamente las posibilidades de arquitectura y experimentan los límites de las tecnologías a utilizar (26).

Durante la fase de exploración se procedió a realizar un estudio en profundidad en cuanto a las herramientas y bibliotecas de desarrollo seleccionadas para explorar las posibles soluciones a generar.

2.1.1 Actores del Sistema

Actores	Descripción
Usuario	Hace uso de la aplicación beneficiándose de sus funcionalidades.

Tabla 3. Actores del Sistema

2.1.2 Historias de Usuario

Las historias de usuario (HU) son la unidad de funcionalidad en un proyecto XP. Se demuestra el progreso aplicando pruebas e integrando código que implementa la historia. La historia de usuario debe ser fácil de entender tanto por desarrolladores como clientes; posible de probar, de valor para el cliente y lo suficientemente pequeña para que los programadores puedan construir un considerable grupo de ellas en cada iteración (26). En XP la gestión de requerimientos del sistema es extremadamente simple, el cliente describe y prioriza sus necesidades mediante historias de usuario que deben ser descripciones cortas y escritas sin terminología técnica.

Las historias de usuario solamente proporcionarán los detalles sobre la complejidad y cuánto tiempo conllevará la implementación de dicha historia de usuario. El nivel de detalle de las historias de usuario debe ser el mínimo posible, permitiendo hacerse una ligera idea de cuánto costará implementar el sistema. Los programadores estiman el esfuerzo asociado y las dependencias entre ellas. Para planificar el trabajo desde el punto de vista técnico, las HU son divididas en tareas para las cuales también se realiza una estimación. Teniendo en cuenta el esfuerzo asociado a las HU y las prioridades del cliente, se define una versión que sea de valor y que tenga una duración de unos pocos meses.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Crear Módulo de Reportes
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.7
Nivel de Complejidad: Alta	**Puntos Reales:** 0.7
Descripción: El sistema debe permitir al usuario agregar el módulo de reportes.	
Observaciones: Esta acción solo se realiza una vez para cada proyecto ya que un proyecto solo puede contener únicamente un módulo de reportes.	

Tabla 4. HU Crear Módulo de Reportes

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Eliminar Módulo de Reportes
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.2
Nivel de Complejidad: Media	**Puntos Reales:** 0.2
Descripción: El sistema debe permitir al usuario eliminar el módulo de reportes siempre y cuando este exista.	
Observaciones:	

Tabla 5. HU Eliminar Módulo de Reportes

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Añadir Plantilla
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.8
Nivel de Complejidad: Alta	**Puntos Reales:** 0.8
Descripción: El sistema debe permitir al usuario agregar una plantilla de reporte. Los datos asociados son: Nombre, Descripción, Base de Datos(BD), Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave.	
Observaciones:	

Tabla 6. HU Añadir Plantilla

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Eliminar Plantilla
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.3
Nivel de Complejidad: Media	**Puntos Reales:** 0.3
Descripción: El sistema debe permitir al usuario eliminar una plantilla de reporte específica.	
Observaciones: Para eliminar una plantilla el usuario debe seleccionarla previamente.	

Tabla 7. HU Eliminar Plantilla

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Modificar Plantilla
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.2
Nivel de Complejidad: Media	**Puntos Reales:** 0.2
Descripción: El sistema debe permitir al usuario modificar una plantilla de reporte específica. Los datos asociados son: Nombre, Descripción.	
Observaciones: Para modificar una plantilla el usuario debe seleccionarla previamente.	

Tabla 8. Modificar Plantilla

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Añadir Conexión
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.6
Nivel de Complejidad: Alta	**Puntos Reales:** 0.6
Descripción: El sistema debe permitir al usuario agregar una nueva conexión. Los datos asociados son: Nombre, Base de Datos(BD), Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave.	
Observaciones:	

Tabla 9. Añadir Conexión

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Eliminar Conexión
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.3
Nivel de Complejidad: Media	**Puntos Reales:** 0.3
Descripción: El sistema debe permitir al usuario eliminar una conexión.	
Observaciones: Para eliminar una conexión el usuario debe seleccionarla previamente.	

Tabla 10. Eliminar Conexión

Historia de Usuario	
Número: 8	**Nombre Historia de Usuario:** Modificar Conexión
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.2
Nivel de Complejidad: Media	**Puntos Reales:** 0.2
Descripción: El sistema debe posibilitar al usuario modificar una conexión existente. Los datos asociados son: Nombre, Base de Datos(BD), Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave.	
Observaciones:	

Tabla 11. Modificar Conexión

Historia de Usuario	
Número: 9	**Nombre Historia de Usuario:** Editar Plantilla
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.7
Nivel de Complejidad: Alta	**Puntos Reales:** 0.7
Descripción: El sistema debe permitir al usuario abrir una plantilla de reporte para su edición.	
Observaciones:	

Tabla 12. Editar Plantilla

Historia de Usuario	
Número: 10	**Nombre Historia de Usuario:** Importar Plantilla
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.5
Nivel de Complejidad: Media	**Puntos Reales:** 0.5
Descripción: El sistema debe permitir al usuario importar una plantilla de reporte desde una ruta específica. Los datos asociados son: Ruta de la Plantilla.	
Observaciones: La plantilla a importar debe estar en extensión jrxml y tener el formato DTD especificado para la biblioteca JasperReports.	

Tabla 13. Importar Plantilla

Historia de Usuario	
Número: 11	**Nombre Historia de Usuario:** Importar Conexión
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.3
Nivel de Complejidad: Media	**Puntos Reales:** 0.3
Descripción: El sistema debe posibilitar al usuario importar una conexión. Los datos asociados son: Ruta del fichero de Conexión.	
Observaciones: La conexión a importar debe estar en extensión xml y contar con el formato establecido por iReport.	

Tabla 14. Importar Conexión

Historia de Usuario	
Número: 12	**Nombre Historia de Usuario:** Eliminar Plantillas
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Media	**Puntos Estimados:** 0.7
Nivel de Complejidad: Media	**Puntos Reales:** 0.7
Descripción: El sistema debe posibilitar al usuario eliminar todas las plantillas existentes pidiendo confirmación de la acción.	
Observaciones:	

Tabla 15. Eliminar Plantillas

Historia de Usuario	
Número: 13	**Nombre Historia de Usuario:** Eliminar Conexiones
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Media	**Puntos Estimados:** 0.6
Nivel de Complejidad: Media	**Puntos Reales:** 0.6
Descripción: El sistema debe brindar al usuario la posibilidad de eliminar todas las conexiones existentes pidiendo confirmación de la acción.	
Observaciones:	

Tabla 16. Eliminar Conexiones

Historia de Usuario	
Número: 14	**Nombre Historia de Usuario:** Visualizar Reporte
Actor: Usuario	**Iteración Asignada:** 3
Prioridad de Negocio: Alta	**Puntos Estimados:** 1.8
Nivel de Complejidad: Alta	**Puntos Reales:** 1.8
Descripción: El sistema debe brindar al usuario la posibilidad de visualizar un reporte específico.	
Observaciones: Esta funcionalidad solo es ejecutada en modo de visualización.	

Tabla 17. Visualizar Reporte

Historia de Usuario	
Número: 15	**Nombre Historia de Usuario:** Exportar Reporte
Actor: Usuario	**Iteración Asignada:** 3
Prioridad de Negocio: Alta	**Puntos Estimados:** 1.2
Nivel de Complejidad: Alta	**Puntos Reales:** 1.2
Descripción: El sistema debe permitir al usuario exportar el reporte en distintos formatos de salida y en un lugar específico. Los datos asociados son: Ruta de Guardado, Extensión del Fichero.	
Observaciones: Esta funcionalidad solo es ejecutada en modo de visualización.	

Tabla 18. Exportar Reporte

2.1.3 Diseño de Casos de Prueba

XP a diferencia de las metodologías más tradicionales, donde la definición y la implementación de las pruebas se realizan usualmente sobre el final del proyecto propone un modelo inverso, donde lo primero que se escribe son las pruebas por las que el sistema debe pasar (27).

Pruebas de Aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios definidas por el cliente y son consideradas como pruebas de caja negra siendo los clientes los responsables de verificar que los resultados de estas pruebas sean correctos (27). Durante una iteración la HU seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una HU ha sido correctamente implementada. Una HU puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento (28). Una historia de usuario no se puede considerar terminada hasta que no pase correctamente todas las pruebas de aceptación.

Para la realización de las pruebas al sistema se diseñaron 17 casos de pruebas donde se ejecutan paso a paso cada una de las posibles entradas y se evalúa el resultado de las mismas. Los diseños de casos de pruebas se encuentran en el epígrafe 3.1 del capítulo 3 donde además se le incluyen los resultados de la ejecución de las mismas.

2.2 Planificación de la Entrega

El propósito de la fase de planificación es establecer un acuerdo entre los clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de historias de usuario puedan ser realizadas. Para ello se realizó el juego de planeación con el objetivo de maximizar el valor del software producido a partir de la puesta en producción de las características más importantes lo antes posible, en el mismo participan en conjunto desarrolladores y clientes con el fin de determinar rápidamente el alcance de la versión a construir; esto se logra gracias a tener al cliente siempre disponible, lo cual facilita la interacción continua entre los involucrados. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración (27).

En esta fase el cliente establece la prioridad de cada HU y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debe obtenerse en no más de tres meses.

2.2.1 Plan de entrega

Una vez que el cliente culmina la elaboración de las HU, se comienza con la creación del Plan de Entregas. El mismo se hace con la intención de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle, o sea, para fijar el período de tiempo que se puede tardar en la implementación de cada una. La planificación se encuentra expuesta en el Anexo 5, esta cumple con todos los requerimientos que exige XP pero existen riesgos potenciales que atentan contra el proceso de desarrollo del software, por lo que es necesario de alguna forma mitigar estos riesgos, ya que el tiempo es corto y algunas de las historias de usuarios poseen una alta complejidad. Por lo antes mencionado se tomaron medidas que permitirán el desarrollo de la solución de forma segura y eficiente.

- Mantener un control de versiones del desarrollo de la aplicación, que permita regresar a una versión anterior y funcional si ocurriese algún inconveniente.

- Comprobación periódica, que permita probar el funcionamiento correcto de la aplicación y evitar que algunas funcionalidades presenten errores debido a la incorporación de otras funcionalidades o por el constante cambio en el código.

2.3 Diseño del sistema

La metodología XP no requiere la descripción del sistema por medio de diagramas de clase utilizando notación UML, sino que se guía por técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Esto no implica que no se utilicen los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando su complejidad no sea alta y defina información importante (29).

2.3.1 Paquetes del Sistema

Para la construcción del sistema se dividió el mismo en paquetes distribuidos de forma tal que cada uno englobe funcionalidades comunes e independientes de los demás paquetes, lo cual provee un diseño distribuido y bien organizado.

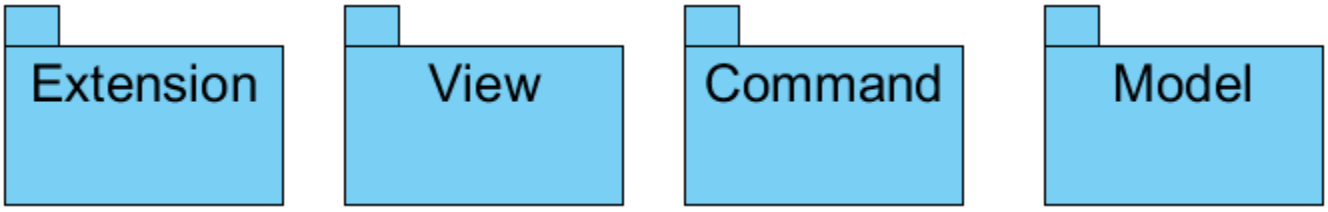

Figura 8. Estructura de Paquetes del Sistema

Los paquetes identificados fueron:

Extension: Contiene las clases que implementan las diferentes interfaces, así como las diferentes fábricas de objetos del sistema.

View: Contiene las interfaces de usuarios del sistema.

Command: Contiene las clases que se encargan de realizar las diferentes tareas en el sistema.

Model: Contiene las clases encargadas de la lógica del negocio.

2.3.2 Funcionamiento del Módulo

El funcionamiento del sistema a desarrollar puede dividirse en dos:

- Un diseñador de reportes, que proporciona al usuario una forma sencilla de diseñar una plantilla donde el usuario tendrá la posibilidad de introducir imágenes, etiquetas de texto, componentes gráficos y cajas de texto para indicar el origen de los datos procedentes de distintas bases de datos.

- Un motor de generación, que se encarga de extraer los datos de una o varias bases de datos, después enlaza estos datos con el diseño realizado y finalmente genera un documento en base al

diseño de reporte obtenido con el diseñador. El documento resultante es un archivo que incluye el diseño más los datos, el cual podremos visualizar en pantalla o exportar a distintos medios como la impresora o el disco duro, en forma de archivo con un determinado formato.

Al formar parte esta aplicación de un SCADA, debe brindar la posibilidad de comunicación entre el motor de generación y los módulos involucrados, en este caso Configuración y Base de Datos Histórico. A continuación se muestra un esquema sencillo donde se explica el funcionamiento básico del sistema, tanto del diseñador como del visualizador.

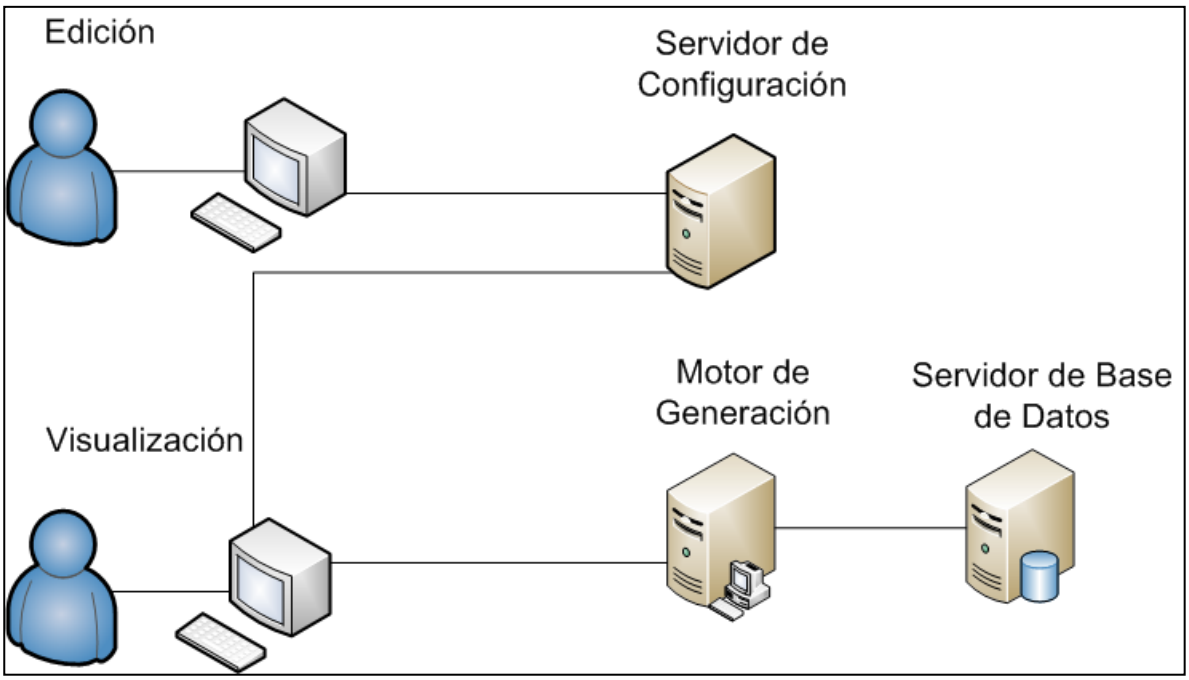

Figura 9. Esquema de interacción del sistema propuesto con el resto de los módulos

El diseñador de reporte guarda la configuración o plantilla del reporte en el Servidor de Configuración, y es de esa configuración de donde se nutre el Motor Generador para mostrar en el Visor el reporte generado con la información proveniente de las distintas bases de datos volcados en su interior y con las mismas características de la plantilla que le dió origen.

2.3.3 Fases de generación de JasperReports

Para un mejor entendimiento del generador de reportes seleccionado JasperReports, se hizo un estudio de las fases o estados por los que pasa un reporte hasta su visualización o impresión para así saber en qué momento intervenir para lograr la integración del mismo a nuestro sistema.

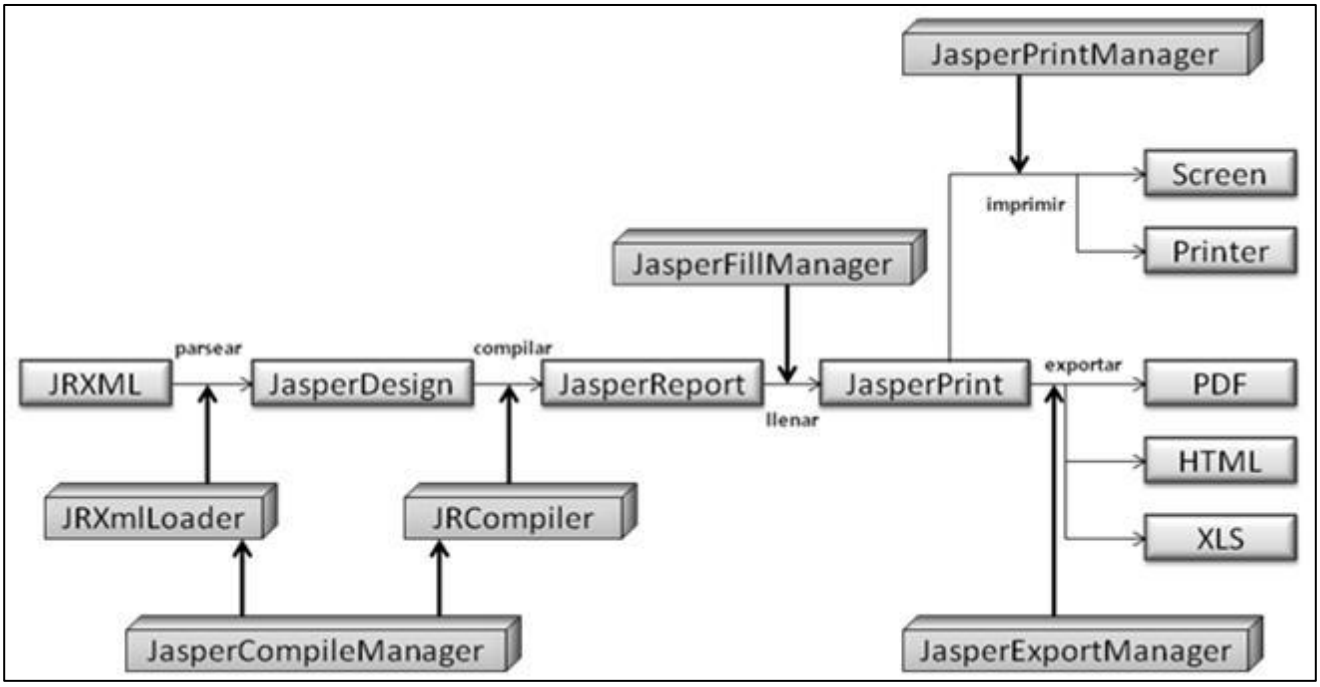

Figura 10. Etapas de generación de un reporte

Como se muestra en la figura anterior, el proceso de generación de reportes parte de la edición de un archivo .jrxml, el cual pertenece a la familia de XML basado en el DTD (definición de tipo de documento) de JasperReports y que contiene el diseño del reporte a generar, este diseño pasa por diferentes etapas, entre ellas una de compilación y llenado la cual consiste en imprimir los datos de la base de datos hacia la plantilla de diseño obteniendo un reporte listo para imprimir o exportar en los diferentes formatos compatibles.

A continuación se ilustra en la imagen los principales elementos de un archivo .jrxml, todos son opcionales con la excepción del elemento raíz <jasperReport>. Cada uno de estos elementos contiene como hijo un elemento <band>.

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport name="report1" pageWidth="595" pageHeight="842">
    <background>
        <band splitType="Stretch"/>
    </background>
    <title>
        <band height="79" splitType="Stretch"/>
    </title>
    <pageHeader>
        <band height="35" splitType="Stretch"/>
    </pageHeader>
    <columnHeader>
        <band height="61" splitType="Stretch"/>
    </columnHeader>
    <detail>
        <band height="125" splitType="Stretch"/>
    </detail>
    <columnFooter>
        <band height="45" splitType="Stretch"/>
    </columnFooter>
    <pageFooter>
        <band height="54" splitType="Stretch"/>
    </pageFooter>
    <summary>
        <band height="42" splitType="Stretch"/>
    </summary>
</jasperReport>
```

Figura 11. Formato Básico del archivo jrxml

Las bandas contienen los datos que se muestran en el reporte. La función de cada uno de los elementos presentados es la siguiente:

Title. Esta es la primera sección del reporte, generada solo una vez durante el proceso de llenado del reporte y ubicada al comienzo del documento resultante.

PageHeader. Aparece en la parte superior de cada página del documento resultante.

ColumnHeader. Aparece al inicio de cada columna definida en el reporte.

Detail. En esta sección es donde se encuentran los datos principales que contendrá el reporte. Para cada tupla del origen de datos, esta sección será generada.

ColumnFooter. Aparece al terminar cada columna.

PageFooter. Aparece en la parte inferior de cada página.

Summary. Esta sección es generada sólo una vez por reporte y aparece al final del documento generado, pero no es necesariamente la última sección.

2.3.4 Arquitectura del Sistema

El diseño de la arquitectura de un sistema es el proceso más complicado en el ciclo de desarrollo de un software, aquí se crea o define una solución para los requisitos técnicos y operacionales del software. Este proceso define qué componentes conforman el sistema, la relación de los mismos además de cómo llevarán a cabo las funcionalidades del sistema mediante su interacción.

La arquitectura propuesta persigue que el producto pueda dar respuesta a los requisitos de comprensibilidad, portabilidad, seguridad, extensibilidad y eficiencia de forma satisfactoria. Para lograr lo antes mencionado se propone la aplicación del patrón arquitectónico Modelo Vista Controlador y en particular una variante del mismo, conocida como Modelo Presentación siguiendo la arquitectura de un sub-módulo del HMI del SCADA, ambiente en el cual se va a encontrar integrado el sistema propuesto.

El uso de esta variante permite tener separadas las responsabilidades dentro de la aplicación en dos dominios, uno sería el llamado Presentación, abarcando la lógica correspondiente a la interacción con el usuario y el otro, el llamado Modelo, en el cual se maneja la lógica perteneciente al negocio (2).

Presentación

En el dominio o capa de Presentación es donde se implementa la interfaz de usuario y las clases que se encargan de enlazar los eventos de la interfaz con manejadores, que a su vez invocarán las funcionalidades en el modelo para su encuesta y modificación.

Modelo

Dentro del Modelo se identifican un conjunto de clases que se corresponden con los conceptos internos de la aplicación. Estos se asocian principalmente a estados de la configuración del reporte y son independientes de cómo son representados al usuario. También se incluyen otro conjunto de clases, las cuales tienen como finalidad la transformación de los estados de las distintas entidades.

Arquitectura Base

A partir de lo antes mencionado se hizo el diseño de una arquitectura base la cual permite incorporar el módulo en desarrollo al actual editor de HMI permitiendo a la extensión operar en dos modalidades, edición y visualización. El módulo cuando se ejecute en el ambiente de edición debe posibilitar la creación o importación de plantillas que sigan el DTD de JasperReports y la edición de estas mediante el editor iReport.

Figura 12. Esquema del Módulo en el ambiente de Edición

Por otra parte cuando se ejecute el módulo en el ambiente de visualización se debe hacer uso de la plantilla de diseño y mediante la biblioteca JasperReports introducir los datos hacia un reporte generado que puede ser visualizado o exportado a otros formatos.

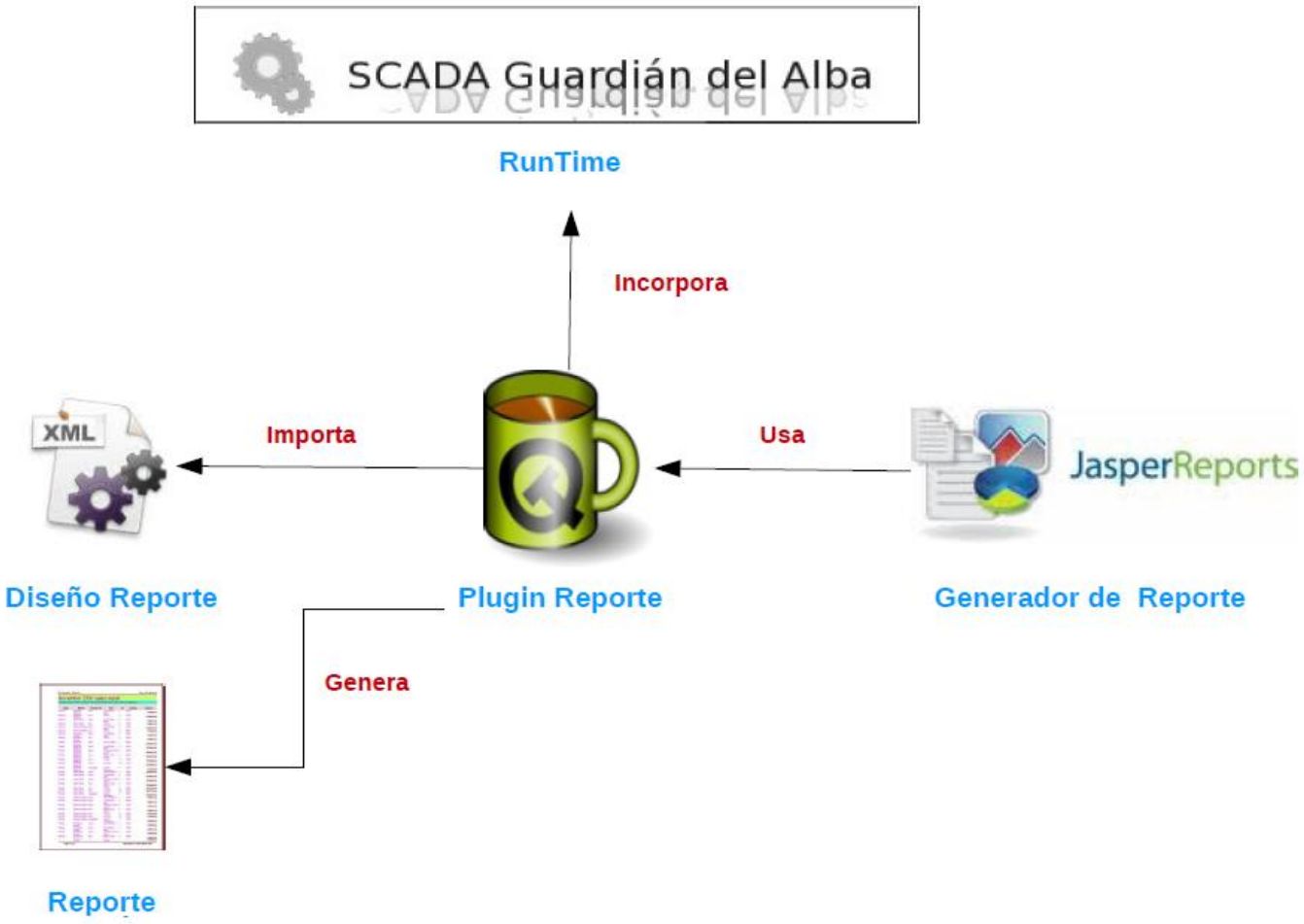

Figura 13. Esquema del Módulo en el ambiente de Visualización

2.3.5 Estructura del Sistema

El sistema en desarrollo debe poseer una estructura compatible con el actual Editor y Visualizador de HMI. La arquitectura actual, a diferencia de las implementadas en los editores y visualizadores más antiguos que separaban en dos proyectos estos ambientes de ejecución, usa en un solo proyecto las dos modalidades. El módulo debe poseer la estructura definida para las extensiones del editor e implementarse con los estándares que se siguen en el desarrollo de otras extensiones.

¿Qué es una extensión?

Es la aplicación que se relaciona con otra para aportarle una función, generalmente muy nueva y específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz que no es más que una clase genérica donde todos sus métodos son virtuales puros. También se lo conoce como *plugin* (enchufable), *addon* (agregado), complemento, conector o extensión (30).

Las extensiones permiten:

- Que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones.
- Reducir el tamaño de la aplicación.
- Separar el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software.

¿Cómo funciona?

La aplicación principal proporciona servicios que el nuevo componente puede utilizar, incluyendo un método para que los complementos se registren a sí mismos y apliquen mecanismos para el intercambio de información. Los complementos dependen de los servicios prestados por la aplicación de acogida y no suelen funcionar por sí solos, por el contrario, la aplicación principal funciona independientemente de ellos, lo que permite a los usuarios finales añadir y actualizar los complementos de forma dinámica sin necesidad de hacer cambios a la aplicación principal.

Las interfaces creadas proporcionan una interfaz estándar (IE), lo que permite a terceros crear extensiones que interactúan con la aplicación principal. Una IE estable permite que extensiones de terceros funcionen como la versión original y amplíen el ciclo de vida de las aplicaciones obsoletas. Las arquitecturas de numerosas aplicaciones suelen utilizar extensiones que permiten a sus editores, ya sean los creadores originales o terceros, agregar funcionalidad al software.

Existen varios tipos de interfaces que ya vienen definidas en el Framework Qt, pero en nuestro caso se declararon otras que facilitan la comunicación entre las diferentes vistas y las diferentes extensiones.

¿Qué funcionalidades provee Qt 4.7 para crear extensiones?

El framework de Qt ofrece a los programadores varias API y MACROS que facilitan el trabajo en la creación e incorporación de extensiones. Para hacer extensible una aplicación a través de extensiones se hace necesario seguir los siguientes pasos:

1. Definir las interfaces usadas para la comunicación.
2. Usar la macro Q_DECLARE_INTERFACE() para decir a la meta-información de objetos de Qt que esta declaración es una interfaz.
3. Usar la función QPluginLoader en la aplicación para cargar las extensiones.
4. Usar qobjet_cast() para probar la validez de las extensiones y hacer conversión de datos.

Para crear una extensión se llevan a cabo estos pasos:

1. Las clases extensiones deben heredar de QObject y de las interfaces que la extensión quiere ampliar.
2. Usar la MACRO Q_INTERFACES() para decir a la meta-información de objetos de QT sobre las interfaz a implementar.
3. Para exportar la extensión debe usar la MACRO Q_EXPORT_PLUGLIN2().

Estructura del Editor:

La aplicación Host o Editor cuenta con cinco vistas principales. Para la comunicación entre las vistas se hace necesario el uso de la biblioteca “libHMICore”. Es esta biblioteca la que permite acceder a todas las extensiones registradas en la aplicación así como a la actualización de cada uno de los modelos de las extensiones y de sus vistas correspondientes.

- *Explorador de Proyecto:* Se encuentra toda la información estructurada del SCADA, este comprende todos los módulos con sus respectivos despliegues de configuración que son los que serán persistentes.
- *Inspector de Propiedades:* Inspecciona las propiedades de los objetos dinámicamente.
- *Paleta de Componentes:* Almacena los componentes gráficos.
- *Historial de Eventos:* Mantiene un historial de todas las acciones que el operador realiza sobre el área supervisada.
- *Área central o área de edición:* Se realiza la edición de los despliegues.

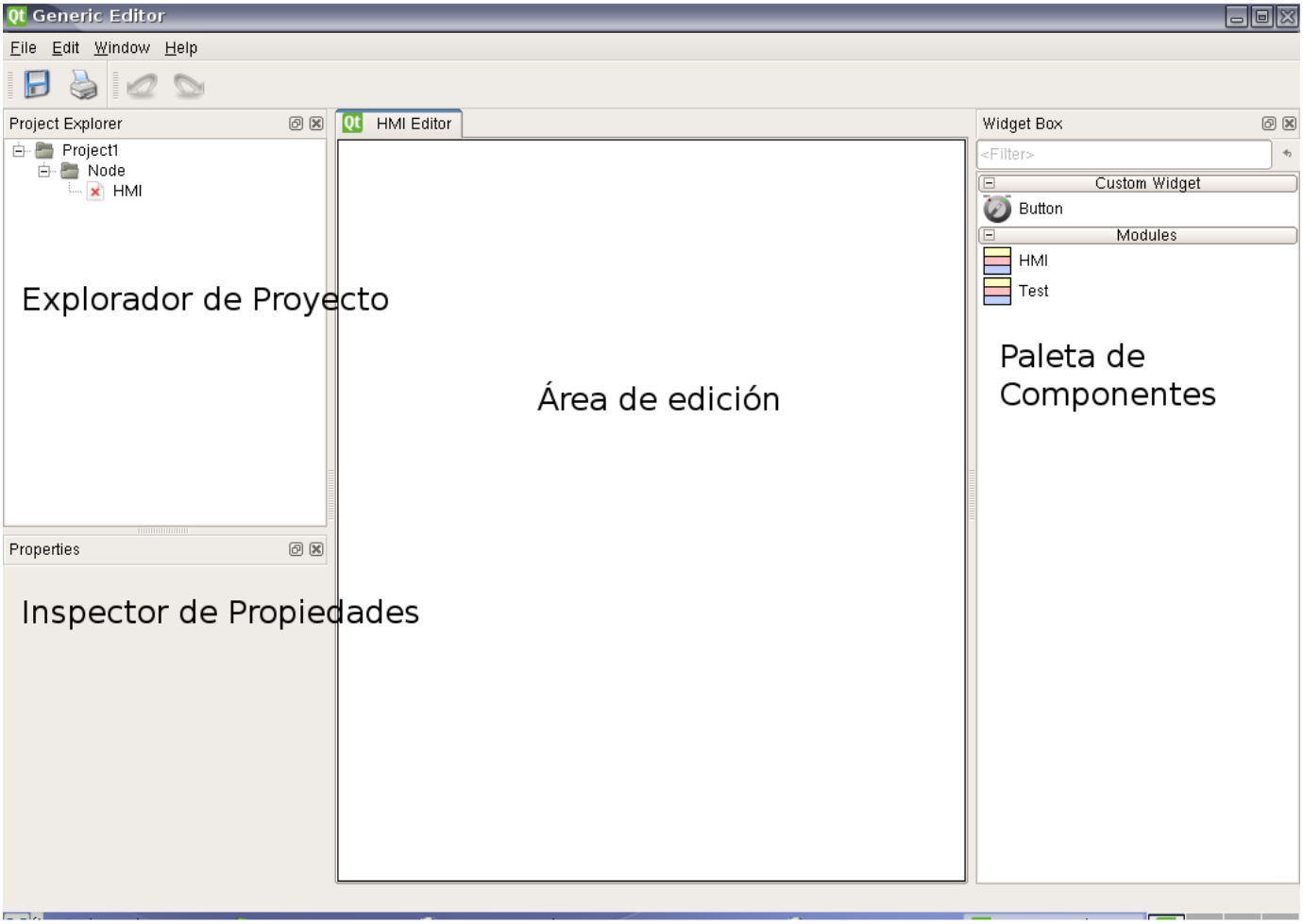

Figura 14. Vistas del Editor genérico

2.3.6 Tarjetas CRC

Las características más sobresalientes de las tarjetas CRC (Clase, Responsabilidad y Colaboración) son su simpleza y ductilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema (31).

Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema y tienen la siguiente estructura.

Clase	
Responsabilidades	Colaboradores

Tabla 19. Modelo de Tarjeta CRC

Tarjetas CRC del Sistema

Para el desarrollo del sistema se hizo necesaria la creación de clases para la representación de las entidades del sistema, las cuales se describen en las siguientes tarjetas CRC. Existen otras clases presentes en el sistema pero por poseer una responsabilidad menor se muestran en las tarjetas CRC del Anexo 4.

ReportModuleExtension	
Contiene la información de los componentes creados en la extensión.	ObjectFactoryExtension ReportFactory CollectionTemplateFactory TemplateFactory

Tabla 20. ReportModuleExtension

ReportFactory	
Contiene aquellos componentes visuales que serán mostrados en una plantilla.	PalettePluginInterface

Tabla 21. ReportFactory

CollectionTemplateFactory	
Contiene todos los objetos creados por nuestro sistema en la modalidad de edición.	ObjectEditorPluginInterface

Tabla 22. CollectionTemplateFactory

TemplateFactory	
Contiene todos los objetos creados por nuestro sistema en la modalidad de visualización.	ObjectEditorPluginInterface

Tabla 23. TemplateFactory

ReportModel	
Contiene todo el módulo de reporte permitiendo la inserción de nuevas plantillas así como su eliminación.	ItemModel

Tabla 24. ReportModel

EditorItemModelTemplate	
Contiene los datos de la plantilla, maneja las acciones que se realizan sobre esta y posee la colección de conexiones.	ItemModelScreen DriverContainer

Tabla 25. EditorItemModelTemplate

EditorItemCollectionTemplate	
Contiene las plantillas del sistema y realiza operaciones sobre estas como creación y eliminación.	ItemCollectionScreen EditorItemModelConnection DriverContainer

Tabla 26. EditorItemCollectionTemplate

EditorItemModelConnection	
Contiene los datos de la conexión y posibilita la realización de acciones sobre estas como modificación y eliminación.	ItemModelScreen

Tabla 27. EditorItemModelConnection

RuntimeItemModelTemplate	
Contiene los datos del reporte ya generado y permite acciones como generar, visualizar y exportar un reporte a diferentes formatos.	ItemModelScreen

Tabla 28. RuntimeItemModelTemplate

RuntimeItemCollectionTemplate	
Contiene los reportes generados del sistema.	ItemCollectionScreen RuntimeItemModelConnection

Tabla 29. RuntimeItemCollectionTemplate

RuntimeItemModelConnection	
Contiene los datos de la conexión para la generación de los reportes en la modalidad de visualización.	ItemModelScreen

Tabla 30. RuntimeItemModelConnection

DocumentReader	
Es el encargado de cargar el reporte generado para su posterior visualización.	Poppler-Qt4

Tabla 31. DocumentReader

RuntimeCentralWidget	
Encargado de visualizar en pantalla el reporte generado y navegar por este además de hacer búsquedas en el mismo.	DocumentReader RuntimeItemModelTemplate

Tabla 32. RuntimeCentralWidget

2.3.7 Patrones de Diseño

Los Patrones de Diseño son soluciones comunes a problemas de diseño de software orientado a objetos y que además poseen ciertas características de efectividad para resolver ese problema. Son reusables ya que pueden ser aplicados en otros diseños o problemas (32).

El módulo al regir su desarrollo mediante un estilo definido en el proyecto productivo debe contar con diferentes patrones de diseño, los cuales son aplicados al código general de SCADA. Entre estos patrones se encuentra Command (Orden), el cual permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma, además que encapsula un mensaje como un objeto, con lo que permite gestionar colas o registros de mensajes y deshacer operaciones. Este patrón soporta restaurar el estado a partir de un momento dado y ofrece una interfaz común que permite invocar las acciones de forma uniforme y extender el sistema con nuevas acciones de forma más sencilla. Por otra parte se hará uso del patrón Abstract Factory (fábrica abstracta) que permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Otro patrón a usar es el nombrado Singleton (instancia única), que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

2.3.8 Estándares de Codificación

Los llamados estándares de codificación son pautas que están enfocadas en la estructura y la apariencia del código dejando a un lado la lógica del programa. Estos estándares tienen como objetivo facilitar la lectura, comprensión y mantenimiento del código (33).

La metodología XP se centra en la comunicación entre los miembros del equipo de desarrollo y en especial trata de lograr una mayor comunicación entre los programadores a través del código. Por lo antes mencionado se hace indispensable que se sigan un grupo de estándares de programación, manteniendo el código legible para los miembros del equipo. Para la implementación del sistema desarrollado se siguieron normas y estándares por los que se rige el proyecto de desarrollo y se exponen en el documento titulado *Estándares de codificación para C++* perteneciente al proyecto SCADA Guardián del ALBA del autor Ariel Chávez Lorenzo.

2.4 Desarrollo de Iteraciones

Esta fase es reconocida como fase principal en el ciclo de desarrollo de XP ya que es aquí donde se desarrollan las funcionalidades, generando al finalizar cada iteración, un entregable funcional que implementa cada una de las historias de usuario que fueron asignadas a la iteración. Las historias de usuario no presentan un nivel de detalle suficiente como para permitir su análisis y desarrollo por lo que al principio de cada iteración se realizan las tareas necesarias de análisis en conjunto con el cliente obteniendo toda la información necesaria. Es por esto que XP necesita tener dentro del equipo de desarrollo al cliente, ya que es de vital importancia su participación activa durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto.

La arquitectura del sistema es establecida en la primera iteración y esta va a ser la utilizada durante el transcurso del proyecto. Para lograr lo anterior se deben escoger las historias que fuercen la creación de esta arquitectura, aunque esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración.

2.5 Implementación

Cada iteración contiene un grupo de HU la cuales son implementadas en el transcurso de la misma. Las HU se descomponen en tareas de desarrollo, asignando a un grupo o una persona como responsable de su implementación. A continuación se muestran detalladamente las tareas de desarrollo realizadas en cada una de las iteraciones.

Iteración 1

La primera iteración tendrá como objetivo darle cumplimiento a las HU 1, 2, 3, 4, 5, 6, 7, 8 y 9 que representan un mayor valor para el cliente, pues con las mismas se conformará la arquitectura base del sistema. Estas recogen funcionalidades de gran importancia para el proyecto, pues a través de ellas se definen aspectos que serán utilizados luego por las demás funcionalidades.

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Crear Módulo de Reportes	0.7	0.7
Eliminar Módulo de Reportes	0.2	0.2
Añadir Plantilla	0.8	0.8
Eliminar Plantilla	0.3	0.3
Modificar Plantilla	0.2	0.2
Añadir Conexión	0.6	0.6
Eliminar Conexión	0.3	0.3
Modificar Conexión	0.2	0.2
Editar Plantilla	0.7	0.7

Tabla 33. Historias de usuarios planificadas para la primera iteración

Para desarrollar las historias de usuario pertenecientes a esta iteración se hizo necesario descomponer las mismas en tareas de implementación, las cuales se listan en el Anexo 1.

Iteración 2

La segunda iteración está centrada en desarrollar las HU 10, 11, 12, 13, las cuales incluyen una menor complejidad de desarrollo, pero son muy importantes para el sistema pues propician la extensibilidad del proyecto mediante archivos externos.

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Importar Plantilla	0.5	0.5
Importar Conexión	0.3	0.3
Eliminar Plantillas	0.7	0.7
Eliminar Conexiones	0.6	0.6

Tabla 34. Historias de usuarios planificadas para la segunda iteración

Luego de relacionar las HU pertenecientes a esta iteración, se procede a la especificación de las principales tareas de desarrollo que se realizaron para cumplir el propósito de las mismas en el Anexo 2.

Iteración 3

Esta iteración se enmarca en los restantes requerimientos, abarcando las HU 14, 15, las cuales se encargan de la visualización y la portabilidad en el sistema.

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Visualizar Reporte	1.8	1.8
Exportar Reporte	1.2	1.2

Tabla 35. Historias de usuarios planificadas para la tercera iteración

Las tareas de desarrollo de las HU pertenecientes a esta iteración, se especifican mediante tareas de desarrollo en el Anexo 3.

Las anteriores iteraciones sobre el ciclo de desarrollo, posibilitaron que al finalizar las mismas se obtuviera un producto con todos los requerimientos y características deseadas por el cliente. El desarrollo de las funcionalidades del sistema se dividió en 3 iteraciones para ordenar la implementación, realizándose pequeñas entregas al cliente al final de cada una de ellas para recibir la aceptación del mismo.

Consideraciones Parciales

Luego de conocer el flujo de trabajo del sistema en cuestión se elaboró una propuesta de la arquitectura, la cual se implementó en el sistema, especificando las diferentes funcionalidades con que debe contar el software, así como la relación entre los diferentes objetos del sistema. Se elaboraron las diferentes historias de usuarios así como su descripción, llevadas a cabo según el orden de prioridad establecido por el cliente para que su implementación siguiera este orden. Las historias de usuarios que en su totalidad son 15 se dividieron en tres grupos los cuales representan las iteraciones por las que pasó el proyecto en su fase de implementación. A partir de cada HU se obtuvieron las diferentes tareas de diseño o implementación que se realizaron para que cada HU quedara completada. En este capítulo quedó conformado el estilo de código por el cual se rige el sistema así como una propuesta de arquitectura en la cual se basa el proyecto.

CAPÍTULO 3

VALIDACIÓN DE LA SOLUCIÓN

Introducción

En el presente capítulo quedan especificados los resultados de la ejecución de las pruebas previamente diseñadas en el capítulo anterior para probar las funcionalidades descritas en el sistema. Además se definen los requerimientos y las actividades necesarias para poner en funcionamiento la herramienta mediante el despliegue de la misma.

3.1 Pruebas de Aceptación

La ejecución de las pruebas permitió evaluar las funcionalidades de la herramienta antes de desplegarlas en un entorno real. Los resultados de las mismas se muestran a continuación:

Caso de prueba de Aceptación	
Número: 1	**Historia de Usuario:** 1
Nombre: Crear Módulo de Reportes	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de crear un nodo que representa el módulo de reportes.	
Condiciones de ejecución: Debe existir un proyecto con un nodo previamente creado.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo del proyecto y en el menú escoge “Crear Módulo” y posteriormente “Reportes”.	
Resultado esperado: El sistema adiciona correctamente el nuevo módulo.	
Evaluación de la prueba: Satisfactorio	

Tabla 36. Caso de Prueba Crear Módulo de Reportes

Caso de prueba de Aceptación	
Número: 2	**Historia de Usuario:** 2
Nombre: Eliminar Módulo de Reportes	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de eliminar el nodo que representa el módulo de reportes.	
Condiciones de ejecución: Debe estar el módulo previamente creado.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo del módulo Reportes el menú escoge “Eliminar Módulo”.	
Resultado esperado: El sistema elimina correctamente el módulo.	
Evaluación de la prueba: Satisfactorio	

Tabla 37. Caso de Prueba Eliminar Módulo de Reportes

Caso de prueba de Aceptación	
Número: 3	**Historia de Usuario:** 3
Nombre: Añadir Plantilla	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de adicionar una nueva plantilla.	
Condiciones de ejecución: El módulo de reportes debe estar previamente creado.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo del módulo de Reportes y en el menú escoge “Añadir Plantilla”. -El sistema muestra el formulario “Nueva Plantilla” con los campos Nombre, Descripción, Base de Datos, Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave. -El usuario introduce los datos requeridos y presiona el botón “Aceptar”. -El sistema adiciona la nueva plantilla y la visualiza en la vista “Explorador de Proyectos”.	
Resultado esperado: El sistema adiciona correctamente la nueva plantilla.	
Evaluación de la prueba: Satisfactorio	

Tabla 38. Caso de Prueba Añadir Plantilla

Caso de prueba de Aceptación	
Número: 4	**Historia de Usuario:** 4
Nombre: Eliminar Plantilla	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” eliminar una plantilla de reportes.	
Condiciones de ejecución: Debe estar la plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo de la plantilla a eliminar y escoge “Eliminar Plantilla”.	
Resultado esperado: El sistema elimina correctamente la plantilla y todas sus conexiones.	
Evaluación de la prueba: Satisfactorio	

Tabla 39. Caso de Prueba Eliminar Plantilla

Caso de prueba de Aceptación	
Número: 5	**Historia de Usuario: 5**
Nombre: Modificar Plantilla	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” modificar una plantilla de reportes.	
Condiciones de ejecución: Debe estar la plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario selecciona la plantilla a modificar en el “Explorador de Proyectos”. -En el “Inspector de Propiedades” el usuario escoge el parámetro a cambiar, teclea su valor y presiona la tecla Enter.	
Resultado esperado: El sistema modifica correctamente la plantilla.	
Evaluación de la prueba: Satisfactorio	

Tabla 40. Caso de Prueba Modificar Plantilla

Caso de prueba de Aceptación	
Número: 6	**Historia de Usuario:** 6
Nombre: Añadir Conexión	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de adicionar una nueva conexión.	
Condiciones de ejecución: Debe haber una plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre la plantilla y en el menú escoge “Añadir Conexión”. -El sistema muestra el formulario “Nueva Conexión” con los campos Nombre, Descripción, Base de Datos, Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave. -El usuario introduce los datos requeridos y presiona el botón “Aceptar”. -El sistema adiciona la nueva conexión y la visualiza en la vista “Explorador de Proyectos”.	
Resultado esperado: El sistema adiciona correctamente la nueva conexión.	
Evaluación de la prueba: Satisfactorio	

Tabla 41. Caso de Prueba Añadir Conexión

Caso de prueba de Aceptación	
Número: 7	**Historia de Usuario:** 7
Nombre: Eliminar Conexión	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” eliminar una conexión.	
Condiciones de ejecución: Debe estar la conexión previamente creada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo de la conexión a eliminar y escoge “Eliminar Conexión”.	
Resultado esperado: El sistema elimina correctamente la conexión.	
Evaluación de la prueba: Satisfactorio	

Tabla 42. Caso de Prueba Eliminar Conexión

Caso de prueba de Aceptación	
Número: 8	**Historia de Usuario:** 8
Nombre: Modificar Conexión	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” modificar una conexión.	
Condiciones de ejecución: Debe estar la conexión previamente creada.	
Entradas/Pasos de ejecución: -El usuario selecciona la conexión a modificar en el “Explorador de Proyectos”. -En el “Inspector de Propiedades” el usuario escoge el parámetro a cambiar, teclea su valor y presiona la tecla Enter.	
Resultado esperado: El sistema modifica correctamente la conexión.	
Evaluación de la prueba: Satisfactorio	

Tabla 43. Caso de Prueba Modificar Conexión

Caso de prueba de Aceptación	
Número: 9	**Historia de Usuario:** 9
Nombre: Editar Plantilla	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” editar una plantilla de reportes.	
Condiciones de ejecución: Debe estar la plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario selecciona la plantilla a editar en el “Explorador de Proyectos”. -Oprime el botón secundario de mouse y en el menú que se despliega escoge “Editar Plantilla”.	
Resultado esperado: El sistema debe abrir el editor externo iReport y permitir la edición de la plantilla, al cerrarlo la plantilla debe haber sufrido los cambios realizados desde iReport.	
Evaluación de la prueba: Satisfactorio	

Tabla 44. Caso de Prueba Editar Plantilla

Caso de prueba de Aceptación	
Número: 10	**Historia de Usuario:** 10
Nombre: Importar Plantilla	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” importar una plantilla.	
Condiciones de ejecución: Debe estar el módulo de reportes previamente creada.	
Entradas/Pasos de ejecución: -El usuario presiona click derecho sobre el módulo de reportes y escoge en el menú desplegado “Importar Plantilla”. -El sistema muestra un cuadro de diálogo para seleccionar la ubicación del fichero jrxml de la plantilla.	
Resultado esperado: El sistema crea una nueva plantilla con los datos que contiene el fichero jrxml.	
Evaluación de la prueba: Satisfactorio	

Tabla 45. Caso de Prueba Importar Plantilla

Caso de prueba de Aceptación	
Número: 11	**Historia de Usuario:** 11
Nombre: Importar Conexión	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” importar una conexión.	
Condiciones de ejecución: Debe existir una plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario presiona click derecho sobre la plantilla reporte y escoge en el menú desplegado “Importar Conexión”. -El sistema muestra un cuadro de diálogo para seleccionar la ubicación del fichero xml de la conexión.	
Resultado esperado: El sistema crea una nueva conexión con los datos que contiene el fichero xml.	
Evaluación de la prueba: Satisfactorio	

Tabla 46. Caso de Prueba Importar Conexión

Caso de prueba de Aceptación	
Número: 12	**Historia de Usuario:** 12
Nombre: Eliminar Plantillas	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” eliminar todas las plantillas creadas.	
Condiciones de ejecución: Debe estar el módulo de reportes previamente creado.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo del módulo de reportes y escoge “Eliminar Plantillas”. -El sistema pide confirmación de la acción en un cuadro de diálogo con los botones “Si” y “No”.	
Resultado esperado: El sistema elimina correctamente todas las plantillas si el usuario escoge “Si”, en caso contrario no se hace nada.	
Evaluación de la prueba: Satisfactorio	

Tabla 47. Caso de Prueba Eliminar Plantillas

Caso de prueba de Aceptación	
Número: 13	**Historia de Usuario:** 13
Nombre: Eliminar Conexiones	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” eliminar todas las conexiones creadas en una plantilla.	
Condiciones de ejecución: Debe estar la plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo de la plantilla de reporte y escoge “Eliminar Conexiones”. -El sistema pide confirmación de la acción en un cuadro de diálogo con los botones “Si” y “No”.	
Resultado esperado: El sistema elimina correctamente todas las conexiones pertenecientes a la plantilla seleccionada en caso de que el usuario haya escogido la opción “Si”, en caso contrario no se hace nada.	
Evaluación de la prueba: Satisfactorio	

Tabla 48. Caso de Prueba Eliminar Conexiones

Caso de prueba de Aceptación	
Número: 14	**Historia de Usuario:** 14
Nombre: Visualizar Reporte	
Descripción: El caso de prueba permite al usuario desde la vista “Inspector de Propiedades” visualizar un reporte a partir de una plantilla.	
Condiciones de ejecución: Debe estar la plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario selecciona la plantilla a visualizar en el “Explorador de Proyectos” y oprime doble click sobre la misma.	
Resultado esperado: El sistema abre el visor de reportes en la vista de “Edición” y visualiza el reporte deseado.	
Evaluación de la prueba:	

Tabla 49. Caso de Prueba Visualizar Reporte

Caso de prueba de Aceptación	
Número: 15	**Historia de Usuario:** 15
Nombre: Exportar Reporte	
Descripción: El caso de prueba permite al usuario desde la vista “Edición” exportar un reporte.	
Condiciones de ejecución: Debe estar la plantilla previamente visualizada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón exportar plantilla situado en la barra de Herramientas del editor. -El sistema muestra un cuadro de diálogo para seleccionar la ruta de guardado y el formato a exportar.	
Resultado esperado: El sistema exporta correctamente el reporte.	
Evaluación de la prueba:	

Tabla 50. Caso de Prueba Exportar Reporte

Caso de prueba de Aceptación	
Número: 16	**Historia de Usuario:** 3
Nombre: Insertar Datos de Plantilla Incorrectamente	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de adicionar una nueva plantilla, esta vez con los datos incorrectos.	
Condiciones de ejecución: El módulo de reportes debe estar previamente creado.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre el nodo del módulo de Reportes y en el menú escoge “Añadir Plantilla”. -El sistema muestra el formulario “Nueva Plantilla” con los campos Nombre, Descripción, Base de Datos, Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave. -El usuario introduce los datos dejando algunos campos vacíos e introduciendo caracteres inválidos como &, luego se presiona el botón “Aceptar”.	
Resultado esperado: El sistema muestra un aviso informando que existen campos incorrectos y seguidamente debe brindar la opción de introducirlos nuevamente.	
Evaluación de la prueba: Satisfactorio	

Tabla 51. Caso de Prueba Insertar Datos de Plantilla Incorrectamente

Caso de prueba de Aceptación	
Número: 17	**Historia de Usuario:** 6
Nombre: Insertar Datos de Conexión Incorrectamente	
Descripción: El caso de prueba permite al usuario desde la vista “Explorador de Proyectos” la posibilidad de adicionar una nueva conexión, esta vez introduciendo datos incorrectos.	
Condiciones de ejecución: Debe haber una plantilla previamente creada.	
Entradas/Pasos de ejecución: -El usuario oprime el botón secundario del mouse sobre la plantilla y en el menú escoge “Añadir Conexión”. -El sistema muestra el formulario “Nueva Conexión” con los campos Nombre, Descripción, Base de Datos, Dirección del Servidor de BD, Tipo de BD, Puerto, Usuario, Clave, Recordar Clave. -El usuario introduce los datos requeridos incorrectamente, dejando campos vacíos e insertando caracteres inválidos, luego presiona el botón “Aceptar”.	
Resultado esperado: El sistema no adiciona la nueva conexión y la visualiza un aviso informando que existen campos incorrectos, posteriormente muestra la ventana para introducir nuevamente los datos.	
Evaluación de la prueba: Satisfactorio	

Tabla 52. Caso de Prueba Insertar Datos de Conexión Incorrectamente

Consideraciones Parciales

En este capítulo se ejecutaron las pruebas de aceptación diseñadas en el segundo capítulo, cada una de estas fue diseñada a partir de las diferentes historias de usuarios y a partir de los resultados arrojados por las mismas se puede llegar a la conclusión de que la herramienta se encuentra lista para su puesta en funcionamiento. Se realizaron 17 pruebas de aceptación, todas arrojando excelentes resultados acerca de la aplicación, confirmando la integridad y buen funcionamiento del módulo desarrollado.

CONCLUSIONES

Una vez finalizada la investigación se puede concluir que:

- Se realizó un análisis de las herramientas y tecnologías utilizadas para la generación de reportes.
- Se logró la generación de los artefactos propuestos por la metodología seleccionada para el desarrollo de la solución.
- Se demostró la factibilidad de la utilización de la biblioteca JasperReports al facilitar el cumplimiento de todas las funciones necesarias en el módulo y permitir la obtención de reportes de un alto nivel con la estructura formal requerida por el cliente.
- Quedó implementada la herramienta informática propuesta.
- Fue probado el sistema, lo que permitió demostrar el cumplimiento de las exigencias del cliente.

RECOMENDACIONES

Concluido el desarrollo de este trabajo se recomienda:

- Incluir mecanismos para la generación periódica de reportes.
- Mejorar la comunicación entre el Módulo Integrador de Reportes y la biblioteca JasperReports con el fin de minimizar tiempo en generación.
- Implementación de un editor de reportes usando el framework Qt con el fin de lograr una mejor integración con el HMI del SCADA Guardián del ALBA.
- Migrar el mecanismo de generación de reportes hacia la web.

REFERENCIAS BIBLIOGRÁFICAS

1. **IEEE.** *Fundamental of supervisory control systems.* s.l. : Engineering Societies Library, 1982.

2. **Barrero, Ernesto Leyva.** *Implementación del módulo de diseño de reportes para el SCADA Guardián del Alba.*

3. Definicion ABC. [Online] [Cited: Junio 15, 2012.] http://www.definicionabc.com.

4. **Rodríguez, Hernan Alberto Silva.** Emagister. [Online] [Cited: Junio 15, 2012.] http://www.emagister.com/curso-procesamiento-datos-oracle/sistema-manejador-base-datos.

5. SLAM-C++. [Online] [Cited: Marzo 20, 2010 .] http://slam-c.nireblog.com/cat/tecnologia.

6. **Serrano., Lisandra González.** *Implementación del Subsistema de Generación de Informes Selux.* La Habana : s.n., 2010.

7. Report Manager Official Page. [Online] [Cited: 6 19, 2010.] http://reportman.sourceforge.net.

8. The Kompany. [Online] [Cited: Junio 15, 2012.] http://www.thekompany.com/projects/kugar/.

9. **JasperSoft.** Jasperforge. [En línea] abril de 2012. http://jasperforge.org/.

10. DataVision, The Open Source Report Writer. [Online] [Cited: Junio 19, 2012.] http://datavision.sourceforge.net/.

11. **Hernández, Yanoski Agneri Martínez and Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO.* Ciudad Habana : s.n., 2008.

12. **Abril, Olivia Rodriguez and Alvarez, Dianly Santiler.** *Analisis y Diseno del Modulo Reportes del Sistema Automatizado para la Gestion Academica Akademos.* Ciudad Habana : s.n., 2007.

13. Alarcos. [Online] [Cited: Junio 19, 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.

14. **Touris, Alberto Molpeceres.** *Procesos de desarrollo: RUP, XP y FDD.* 2002.

15. **Acuña, Karenny Brito.** EUMED.NET. [Online] [Cited: Junio 19, 2012.] http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.

16. **Lafuente, Guillermo Javier.** UML Unified Modeling Languaje. [Online] Febrero 2002. [Cited: Mayo 24, 2011.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html..

17. Zator. [Online] http://www.zator.com/Cpp/E1_2.htm.

18. Ciberaula. [Online] http://java.ciberaula.com/articulo/que_es_java.

19. **pbworks.** Pbworks. [Online] [Cited: Junio 15, 2012.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.

20. SlideShare. [Online] 2008. [Cited: Junio 19, 2012.] http://www.slideshare.net/Decimo/arquitectura-3-capas.

21. **Sebastián, Juan.** Comusoft. [Online] Noviembre 2010. [Cited: Junio 19, 2012.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

22. [Online] [Cited: Junio 19, 2010.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

23. ECLIPSE. [Online] [Cited: Junio 15, 2008.] http://www.eclipse.org.

24. **Ubuntu.** Guía Ubuntu. [Online] [Cited: Junio 15, 2012.] http://www.guia-ubuntu.org/index.php?title=NetBeans.

25. Visual Paradigm for UML. [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

26. **Beck, Kent.** *Extreme Programming Explained.* 1999.

27. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* 2008.

28. **Escribano, Gerardo Fernández.** Introducción a Extreme Programming. [Online] 2002. [Cited: febrero 5, 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf..

29. **Adisleyd.** *Herramienta de Apoyo a la Producción en el CEDIN.* Ciudad Habana : s.n., 2010.

30. **Carrera Ortega, Jorge and Chávez Lorenzo, Ariel.** *Guía para la creación e incorporación de plugins al editor del SCADA Phoenix.* La Habana : s.n.

31. **Casas, Sandra and Reinaga, Héctor.** *Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones.* 2008.

32. **Rojas, Mc Juan Carlos Olivares.** [Online] 2007. http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07b/patrones.pdf.

33. —. *Patrones de Diseño.* 2007.

34. Nocisoft. *Nocisoft.* [Online] http://www.nocisoft.com/ncreport.html.

35. **Penin, Aquilino Rodríguez.** *Sistemas SCADA.* 2006.

36. **M.Romero, Diego.** *Sistemas de Interfaz Humano Maquina(HMI).* 2007.

37. REKALLREVEALED.ORG. *The database front-end for KDE and the Web.* [Online] http://www.rekallrevealed.org/.

38. JASPERREPORTS. [Online] 1 7, 2007. http://jasperreports.sourceforge.net/.

39. **Driggs, Yosell Luis Sehara.** *Implementación de un modelo para la configuración de un sistema SCADA.* 2008.

40. **TOURIS, ALBERTO MOLPECERES.** [Online] 8 2, 2002. http://www.javahispano.org/articles.article.action?id=70.

41. **JACOBSON, JAMES.** *El Proceso Unificado de Desarrollo de Software.* La Habana : s.n., 2004.

42. **Penadés, Patricio Letelier.** *Métodologías ágiles para el desarrollo de software.*

43. **BOOCH, G., JACOBSON, JAMES and I. y RUMBAUGH, J.** *UML Manual de referencia.* 1997.

44. Nokia Corporation. [Online] Nokia, 2008-2010. http://qt.nokia.com/.

45. **Kiccillof, Carlos Reynoso.** *Estilos y Patrones.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.

46. ARQUITECTURA Modelo Vista Controlador. [Online] agosto 2007, 2007. http://exequielc.wordpress.com/2007/08/20/arquitectura modelovistacontrolador.

47. **RUIZ, MARLON.** Monografías.com. [Online] http://www.monografias.com/trabajos34/base-de-datos/base-de-datos.shtml#base.

48. **MEZA, Lic. Luis Enrique MUNIVE.** Scribd. [Online] http://es.scribd.com/luis_meza_95/d/88281916-Base-de-datos-Conceptos-generales.

49. **Galiano, Fernando Berzal.** *Diseño de arquitecturas software.* 2005.

50. **Pressman.** *Ingeniería de Software un enfoque práctico.* EUA : s.n.

GLOSARIO

Software: Programas, procedimientos y reglas para la ejecución de tareas específicas en un sistema de cómputo.

Reporte: Documento contentivo de información personalizada y útil para el destinatario del mismo.

Generador de reportes: Herramienta que permite generar reportes a partir de datos primarios.

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

PDF (P*ortable Document Format*): Formato de documento portátil.

HTML (*Hyper Text Markup Language*): Lenguaje con el que se escriben las páginas web. Permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.

CSV (C*omma Separated Values*): Son un tipo de documento sencillo para representar datos en forma de tabla.

SCADA (*Supervisory Control And Data Acquisition*): Control Supervisor y Adquisición de Datos.

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

Caso de Prueba: Conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

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

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

XML (*eXtensible Markup Language*): Metalenguaje extensible de etiquetas desarrollado por la World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG y MathML.

API: Una interfaz de programación de aplicaciones es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

DTD: Descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD.