Universidad de las Ciencias Informáticas

Facultad 5

Título: Modelación de un generador de informes para sistemas de supervisión, control y adquisición de datos.

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

Autor: Osmany Jorge Riverón.
Hilario Yera Calleja.

Tutores: MSc. Jaime Fardales Pérez.
Ing. Amado Espinosa Hidalgo.

Ciudad de la Habana, Mayo 2007

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los __ días del mes de junio del 2007.

Firma del Autor	Firma del Autor
(Osmany Jorge Riverón)	(Hilario Yera Calleja)
Firma del Tutor	Firma del Tutor
(MSc. Jaime Fardales Pérez)	(Ing. Amado Espinosa Hidalgo)

Datos de Contacto

MSc. Jaime Fardales Pérez

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

Ing. Amado Espinosa Hidalgo

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

Agradecimientos

A nuestros queridos Fidel y Raúl y a la UCI por hacernos participes de este proyecto de la Revolución.

A nuestro tutor: Jaime Fardales Pérez, por dedicar tantas noches a nuestra tesis y habernos impulsado desde el inicio.

A nuestros padres y abuelos, por confiar en nosotros y guiarnos durante toda la vida.

A nuestros amigos y compañeros que vienen con nosotros desde primer año.

A todos aquellas personas que confiaron en nosotros desde un principio.

De Osmany:

A mi padre Joaquín y mi madre Sonia, a mis hermanos queridos Cari y Jose, a mis abuelos y toda mi familia, a mis inseparables amigos Ernesto, Cao, Luis Angel, y Evian por haber sido un equipo incondicional que siempre estuvieron a mi lado, a mis compañeros de largos años juntos en la UCI, a mi amigo y compañero de tesis Hilario, a mis amigos del barrio, a los colectivos de Simpro y SCADA, a mi querida Yadith, a mis compañeros de equipo de Karate, a mi entrañable y querido profesor y amigo Millet, a mis amigos Eikel, Reinier y Yoeny, a mi amiga Dayami, al profesor Amado que siempre estuvo guiándome en la producción, al integral amigo Lannie, a Fardales por su entrega en el proyecto y como persona, a mi amigo eterno Ricardo y a todos los que hicieron posible mi ingreso a la Universidad y que hoy haya logrado cumplir esta gran meta.

De Hilario:

Dedico este trabajo realizado a mis padres que tanto me han apoyado, especialmente a mi mama Alina y mi abuelita Oneida y mi abuelo Luís por confiar en mi y darme cada día mas animo a seguir y triunfar en mis estudios, a mi hermanito Rafelito, a mi tutor Fardales por el gran apoyo a mi compañero de tesis Osmany por el gran trabajo realizado en conjunto, a mi papa Hilario, a mi tía Marilú y mi tío Alberto por celebrar mis regresos a la provincia, a mi tío Rolando y mi Tía Eliza por confiar en mi y darme todo su apoyo, a mis abuelos Marcelino y Maria por su gran apoyo a mi primo Rolandito, a mis lindas hermanitas gemelas Oriley y Orilay que tanto las quiero, a mis amigos de Villa Clara Dagmar, Yojanier y Karen a mis amigos del piquete de la estocada final Yadier, Elier, Alain, Yoe, el tico, el rubio, el chiqui, Julio y demás compañeros por compartir todo este tiempo con ellos, a mis amigos del barrio Lester, Reinier, Leslie y demás, a mi querida Yailyn, a mi compañero y profesor Amado, a mis compañeros de Simpro tanto teórico como practico, a mis compañeros de brigada y proyectos anteriores, en fin a todos los que han hecho que este sueño se haga realidad.

RESUMEN

Producto del convenio de colaboración entre Cuba y Venezuela se desarrolla en nuestra Universidad de las Ciencias Informáticas un proyecto para el desarrollo de un sistema de supervisión, control y adquisición de datos (SCADA) dirigido a satisfacer las necesidades de automatización de la empresa Petróleos de Venezuela S.A. (PDVSA). En este SCADA es necesario un modulo que automatice la generación de informes mediante el procesamiento y la presentación de los datos provenientes de bases de datos históricas. Para dar solución a esta problemática se desarrolla el presente trabajo, cuyo objetivo consiste en modelar un subsistema de generación de informes bajo los preceptos de software libre, que interactuando con los demás subsistemas del SCADA permita configurar y emitir los informes necesarios en el SCADA Nacional de PDVSA.

Para ello se hace un estudio teórico de aspectos relacionados con la generación de informes para SCADAs y se evalúan las características esenciales de un conjunto de generadores de informes muy utilizados, principalmente sobre software libre. Además se realiza un análisis de las tendencias y tecnologías existentes, proponiéndose las que se consideran más adecuadas para el desarrollo. Se realiza todo un proceso de ingeniería que comenzando por el levantamiento de requisitos y pasando por la realización de diagramas de casos de uso, especificación de casos de uso, diagramas de secuencia, diagrama de arquitectura, diagrama de clases y prototipos de interfaz de usurario, permite presentar un modelo de la aplicación a desarrollar.

PALABRAS CLAVES

SCADA, Ingeniería, requisitos, software, informe, reporte, diseño, clases, modelo.

Tabla de contenidos

FIGURAS

INTRODUCCIÓN

El desarrollo de las grandes industrias tiene marcada influencia en el futuro económico de los países, determinando en muchos casos el posicionamiento en el mercado de sus economías. En estas se consumen la mayor parte de los recursos energéticos existentes y se transforman o procesan los componentes naturales básicos, proporcionando empleo a grandes masas de trabajadores. De aquí surge la importancia de contar con industrias competitivas y eficientes que potencien la actividad productiva del ser humano para alcanzar elevados niveles de productividad. Si a estos planteamientos adicionamos el gasto de materiales y materias primas por ineficiencias de los procesos productivos industriales se nos presenta el panorama donde la automatización debe entrar a jugar un papel preponderante.

Los sistemas digitales para el control industrial han evolucionado considerablemente en los últimos años debido a los avances alcanzados en la tecnología de integración de circuitos, a la aparición de nuevas arquitecturas de computadoras e interconexión entre ellas y al desarrollo de nuevos lenguajes y sistemas.

En Junio del 2002 tuvo lugar el primer Encuentro Nacional de Automática en nuestro país. En este, el titular del Ministerio de la Informática y las Comunicaciones (MIC), órgano rector de las actividades de automatización en el país, Ignacio González Planas, expresó refiriéndose a la automática: "*Hay que potenciar su uso donde signifique ahorro, productividad y eficiencia, sobre todo asociada a trabajos duros. La automatización –parte inseparable de los procesos industriales y de servicios- ha logrado poner en funcionamiento algunos proyectos con rápida recuperación de sus inversiones, como por ejemplo los relativos al ahorro de energía. La prioridad está en las soluciones automáticas que resuelvan un problema, mejoren la eficiencia, la calidad y recuperen las inversiones en plazos cortos, ya sea en la agricultura, en la industria azucarera o en la básica*." (HERNÁNDEZ BASSO Septiembre 2002.)

En la Universidad de la Ciencias Informáticas existen proyectos productivos cuyo objetivo fundamental es lograr la automatización de empresas e instituciones tanto del país como en el exterior, tal es el caso de la empresa venezolana PDVSA (Petróleos de Venezuela). Uno de los proyectos vinculados a tan importante empresa, es el que se lleva a cabo en la Facultad 5 "SCADA Nacional", cuyo objetivo fundamental es desarrollar un sistema de supervisión, control y adquisición de datos (SCADA); que en sus siglas en inglés es Supervisory Control and Data Adquisition; sobre la plataforma GNU/Linux bajo los preceptos de software libre y código abierto (OpenSource) el cual debe ser instaurado en todas las infraestructuras productivas de PDVSA además de existir el interés ministerial por parte del MIC (Ministerio de Informática y

Comunicaciones de Cuba) de replicar estos trabajos para las infraestructuras productivas de nuestra nación.

En la actualidad es muy común que para el control y procesamiento de la información de las fabricas e industrias se haga uso de sistemas informáticos denominados SCADA, acrónimo que resulta del nombre ingles de estos sistemas (Supervisory Control and Data Adquisition). Un sistema de supervisión, control y adquisición de datos (SCADA) no es más que una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. En este tipo de sistemas usualmente existe un ordenador, que efectúa tareas de supervisión y gestión de alarmas, así como tratamiento de datos y control de procesos. La comunicación se realiza mediante buses especiales o redes LAN. Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

PDVSA en sus instalaciones cuenta con SCADAs para operar confiable y eficientemente sus procesos tecnológicos. Estos SCADAs han sido suministrados por compañías extranjeras, lo que ha provocando una total dependencia de las mismas y grandes dificultades para mantenerlos en operación. Algunas de estas compañías se plegaron, y apoyaron con la utilización de estas tecnologías el “Paro petrolero” que afectó a PDVSA a fines del año 2002, que posibilitaron la operación remota de las instalaciones afectadas por personal no autorizado para ello. De esta manera resultó evidente lo vulnerable de estos sistemas, lo que unido a la dependencia tecnológica para el mantenimiento de empresas de servicios que se plegaron a otros intereses, provoca una compleja situación para mantener funcionado y expandiendo los sistemas SCADAs existentes en la actualidad y un riesgo potencial para la estabilidad y seguridad de los objetivos tecnológicos operados mediante los mismos.

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

Además de estos argumentos asociados al clima político que hoy se respira en la Republica Bolivariana de Venezuela, existen argumentos técnicos sólidos que sustentan lo problemático de la situación presentada. El sistema SCADA OASyS se define como la infraestructura SCADA más grande de PDVSA, sobre la cual se encuentran configuradas el 70% de las señales automatizadas provenientes de los dispositivos de campo (PLC´s, RTU´s), y sobre el cual se maneja una producción aproximada de 1,1 MMBPD (Millones de barriles promedio por

día), distribuida en 9 Unidades de Explotación, 8 Patios de Tanques y 2 Unidades de Servicio. La plataforma computacional que soporta el SCADA OASyS se encuentra actualmente en niveles críticos de capacidad y vida útil, existen problemas serios a nivel de hardware que afectan la disponibilidad y confiabilidad del sistema y que no permiten afrontar el crecimiento y la demanda de nuevos requerimientos para el crecimiento que necesita PDVSA Occidente, PDVSA y Venezuela.

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

De todo lo expuesto anteriormente se presente como **problema científico** la siguiente interrogante ¿Cómo realizar el procesamiento y presentación de los datos existentes en bases de datos históricas pertenecientes al SCADA, mediante la creación de un generador de informes sobre plataforma GNU/Linux?

Teniendo como **objeto de estudio** en la presente investigación la generación de informes basada en software libre para el procesamiento y la presentación de los datos, dentro de lo cual se tiene como **campo de acción** la generación de informes basada en software libre para el procesamiento y la presentación de los datos, específicamente aplicada al SCADA Nacional de PDVSA**.**

Como **objetivo** nos trazamos la modelación de un subsistema de generación de informes bajo los preceptos de software libre que interactuando con los demás subsistemas del SCADA permita configurar y emitir los informes necesarios en el SCADA Nacional de PDVSA.

El desarrollo de este subsistema y su posterior explotación de conjunto con el sistema SCADA permitirá minimizar el tiempo de trabajo necesario para llevar a cabo cualquier operación, así como las pérdidas de datos. Se logrará una mejor organización en los procesos que la aplicación automatiza y permitirá un mejor funcionamiento de la empresa, además de lograr una mayor agilidad en el control de los procesos de producción y permitir un mejor control de las tareas tanto por el personal directo a la producción como por parte de la máxima dirección de la infraestructura productiva.

Para ello debemos cumplir las siguientes **tareas:**

- Recopilar y evaluar información sobre generadores de informes existentes para plataformas en software libre.
- Estudiar y evaluar información sobre SCADAs.
- Realizar un levantamiento de requisitos para el desarrollo de un sistema de generación de informes aplicado a un SCADA.
- Determinación y especificación de los principales casos de uso presentes en el subsistema
- Seleccionar sistemas de generación de informe de acuerdo a la adecuación de sus características a las necesidades del proyecto teniendo en cuenta la posibilidad de reutilización en el desarrollo planteado (benchmark) y estudio de sus interioridades mediante técnicas de ingeniería inversa.
- Desarrollo de artefactos necesarios para el diseño del subsistema de generación de informes.

Para la presentación de los resultados obtenidos en este trabajo, el presente informe consta de 3 capítulos estructurados de la siguiente forma: en el Capítulo 1 se hace un esbozo teórico centrado en temáticas como sistemas SCADAs, generadores de informes, software libre vs. software propietario, etc., que sirven de apoyo a todo el trabajo desarrollado. En el Capitulo 2 se analizan las principales tecnologías a utilizar y se hace un análisis para su posterior uso. En el Capítulo 3 se profundiza en la situación problémica presentándose la Especificación de los requisitos de software así como la definición de los principales casos de uso. Por último en el Capítulo 4 se presenta el diseño del sistema.

CAPÍTULO 1

1 FUNDAMENTACIÓN DEL TEMA

1.1 Introducción:

Con el objetivo de ayudar a la comprensión general del negocio, en el presente capítulo se exponen las principales características de los sistemas SCADAs desde el punto de vista de la generación de informes. Para ello se abordan conceptos relacionados con estos sistemas y se presentan las características esenciales de los generadores de informes que los mismos usan.

1.2 Definición de SCADA.

Las siglas SCADA son un anglicismo que abrevia la denominación de "*Supervisory Control And Data Acquisition*", es decir: Sistema de control y supervisión con adquisición de datos. Se trata de un sistema que permite controlar y monitorear desde un centro de control los procesos de estaciones remotas distantes, empleando diversos tipos de enlaces de comunicación (SCADA, SISTEMA 2002). Un sistema de control y supervisión generalmente se define; *"como una forma de control remoto con disponibilidad para el control selectivo de las unidades remotas"* (IEEE 1982). Estos sistemas *"permiten un gobierno total de la planta aunando las ventajas de seguridad de autómatas industriales y el control y gestión mediante computadoras que permiten interfaces gráficas muy amigables e intuitivas para el operario."* (ARINA OYAGA Marzo de 2000). Otra posible definición de un SCADA es la denominación que en general recibe el conjunto formado por una *"aplicación software especialmente diseñada para funcionar sobre computadoras de control de producción, con acceso a la planta mediante comunicación digital con los reguladores locales básicos, e interfaces gráficas de alto nivel con usuario mediante pantallas táctiles, ratones o cursores, lápices ópticos, etc., los programas necesarios, y en su caso el hardware adicional que necesiten"* (ARINA OYAGA Marzo de 2000). Otra definición un poco mas abarcadora es la siguiente: *"un sistema basado en aplicaciones de software diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo (Controladores Lógicos Programables- PLC, Unidad Terminal Remota-RTU, etc.) y controlando el proceso de forma automática desde la pantalla del computador. Permitiendo realizar a distancia operaciones de control, supervisión y registro de datos de cualquier proceso industrial. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. De igual forma, ya que cuenta con información (alarmas, históricos, paradas,*

etc.) de primera mano de lo que ocurre u ocurrió en el proceso, permite la integración con otras herramientas del negocio como lo son intranets, ERP, etc".

En esta estructura las computadoras se apoyan en las potencialidades de dispositivos locales, uniéndose a ellos mediante líneas de interconexión digital (redes locales) por donde recogen información sobre la evolución del proceso (adquisición de datos), y envían las órdenes o comandos de arranque, parada o cambios de producción para el gobierno de los mismos (control de producción).

1.3 Antecedentes de los sistemas SCADAs.

Los primeros SCADA eran simples sistemas de telemetría que proporcionaban informes periódicos de las condiciones de campo, vigilando las señales que representaban medidas y/o condiciones de estado en ubicaciones de campo remotas, en muchos casos lo que se hacia era imprimir en papel para registrar información de variables y poder llevar un histórico de eventos que ocurrían durante el proceso. Estos sistemas ofrecían capacidades muy simples de monitoreo y control, sin proveer funciones de aplicación alguna. La visión del operador en el proceso estaba basada en los contadores y las lámparas detrás de paneles llenos de indicadores.

Mientras la tecnología se desarrollaba, los ordenadores asumieron el papel de manejar la recolección de datos, disponiendo comandos de control, y una nueva función, presentación de la información sobre una pantalla que para ese momento eran monocromáticas (CRT). Muchas empresas viendo la necesidad y lo rápido que avanzaba el desarrollo de los computadores, fueron realizando programas de aplicación específicos para atender requisitos de algún proyecto particular, de aquí que los ordenadores agregaron la capacidad de programar el sistema para realizar funciones de control más complejas. Como ingenieros de varias industrias asistieron al diseño de estos sistemas, su percepción de SCADA adquirió las características de su propia industria. Proveedores de sistemas de software SCADA, deseando reutilizar su trabajo previo sobre los nuevos proyectos, perpetuaron esta imagen de industria específica por su propia visión de los ambientes de control con los cuales tenían experiencia. Solamente cuando nuevos proyectos requirieron funciones y aplicaciones adicionales, hizo que los desarrolladores de sistemas SCADA tuvieran la oportunidad de desarrollar experiencia en otras industrias. Así nacieron los pequeños SCADAS desarrollados por empresas desarrolladoras de software y una nueva experiencia para muchas de ellos.

Hoy, los proveedores de SCADA están diseñando sistemas que son pensados para resolver las necesidades de muchas industrias con módulos de software disponibles para proporcionar las capacidades requeridas comúnmente. Puesto que los proveedores de SCADA aún tienen tendencia en favor de alguna industria, los compradores de estos sistemas a menudo

dependen del proveedor para una comprensiva solución a su requisito, y generalmente procurar seleccionar un vendedor que pueda ofrecer una completa solución con un producto estándar que esté apuntado hacia las necesidades específicas del usuario final. Si selecciona a un vendedor con experiencia limitada en la industria del comprador, el comprador debe estar preparado para asistir al esfuerzo de ingeniería necesario para desarrollar el conocimiento adicional de la industria requerido por el vendedor para poner con éxito el sistema en ejecución.
La mayoría de los sistemas SCADA que son instalados hoy se está convirtiendo en una parte integral de la estructura de gerencia de la información corporativa. Estos sistemas ya no son vistos por la gerencia simplemente como herramientas operacionales, sino como un recurso importante de información. En este papel continúan sirviendo como centro de responsabilidad operacional, pero también proporcionan datos a los sistemas y usuarios fuera del ambiente del centro de control que dependen de la información oportuna en la cual basan sus decisiones económicas cotidianas. La arquitectura de los sistemas de hoy, integra a menudo muchos ambientes de control diferentes, tales como tuberías de gas y aceite, en un solo centro de control.
Para alcanzar un nivel aceptable de tolerancia de fallas con estos sistemas, es común tener ordenadores SCADA redundantes operando en paralelo en el centro primario del control, y un sistema de reserva del mismo situado en un área geográficamente distante. Esta arquitectura proporciona la transferencia automática de la responsabilidad del control de cualquier ordenador que pueda llegar a ser inasequible por cualquier razón, a una computadora de reserva en línea, sin interrupción significativa de las operaciones.

1.4 Niveles de un sistema SCADA.

Como se muestra en la Fig. 1, un sistema SCADA puede subdividirse para su mejor comprensión en 4 niveles o partes esenciales: (DISTEFANO. 1999.)

1. Nivel de instrumentación: Este nivel toma las variables físicas y las convierte en una señal equivalente que puede ser leída o interpretada por el hombre. El sistema SCADA maneja instrumentación de tipo electrónica donde la variable física se convierte en una señal eléctrica.
2. Nivel de RTU: Las RTU (Remote Terminal Unit) o unidades terminales remotas, son dispositivos inteligentes a microprocesador que recogen, almacenan y procesan la información que viene de la instrumentación de campo y comandan elementos finales de control. Son programables, permitiendo alojar algoritmos de control y poseen un canal serie de comunicación para su interconexión.
3. Nivel de comunicaciones: Es el encargado de tomar la información de las RTUs y transmitirla por el medio escogido hasta el centro de control. Dependiendo del costo,

tamaño del sistema SCADA, distancias a las RTUs, disponibilidad del medio, la velocidad de transmisión y la confiabilidad requerida, se seleccionan los distintos soportes y medios de comunicación más apropiados.

4. Centro de control: Está compuesto por un conjunto de computadoras, periféricos y software que realizan el procesamiento de las señales. Existe al menos una computadora con su hardware y software de base en la cual se mantiene ejecutando un software SCADA de cierta complejidad, que abarca diversas funciones. Usualmente existe también un equipo que sirve como interfaz de comunicaciones, cuya función es recibir la información de los diferentes canales de comunicación, procesarla y agruparla para enviarla a las restantes computadoras mediante una red.

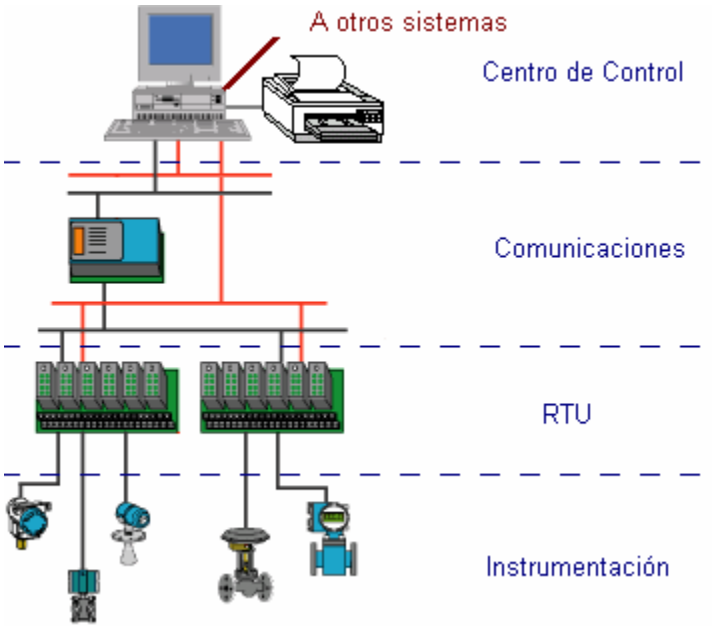

Fig. 1. Niveles de un sistema SCADA.

1.5 Entorno de un SCADA.

Abordando varias cuestiones acerca de los SCADA en los conceptos antes mencionados podemos ver un entorno básico más general acerca de este sistema en la siguiente imagen presentada.

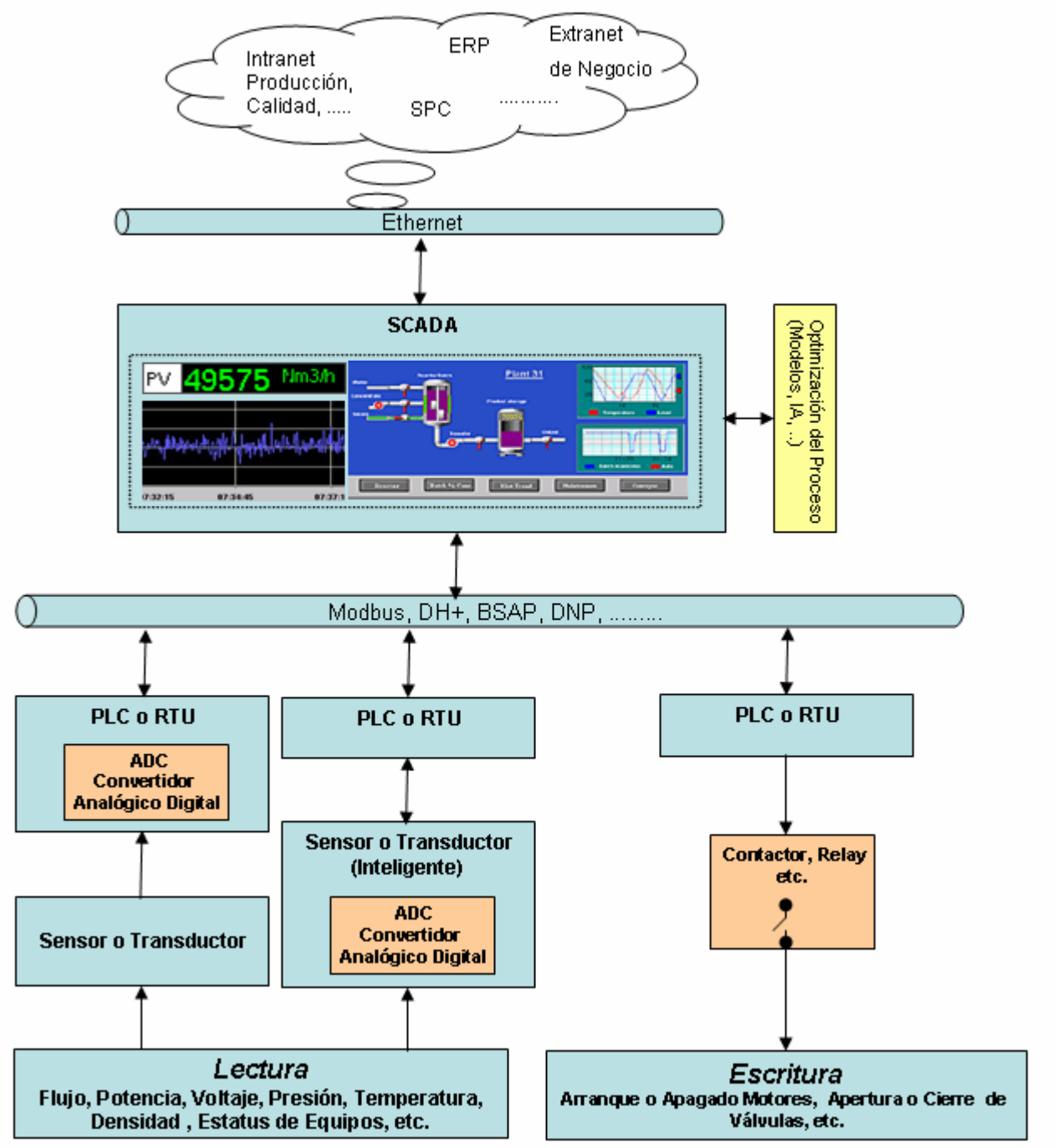

Fig. 2. Entono básico de un SCADA.

1.6 Software Libre

Con el progresivo avance de la tecnología y para su posterior explotación se obtiene la necesidad de crear programas de computadoras (Software) que estén cada vez mas adaptados al desarrollo actual. Muchos de estos programas creados son pertenecientes a empresas como software propietarios, donde para poder adquirir estos productos y sus licencias hay que pagar grandes sumas de dinero. Esto ha permitido que otros desarrolladores de software se

interesen en distribuir de forma libre y gratuita productos de gran importancia encargados de lograr un mejoramiento de estos para su posterior utilización y un mejor funcionamiento, de aquí que comienza el surgimiento del Software Libre.

Software libre: Es un software, cuyo secreto de fabricación (el llamado "código fuente", algo así como los "planos" del programa) es conocido y difundido libremente. El autor del software concede a cualquiera el derecho a usar su obra, a modificarla y a adaptarla a sus necesidades específicas. Este derecho de libre uso a veces se otorga sin restricciones (licencias tipo BSD) y otras veces con la única condición de que toda mejora se distribuya con las mismas condiciones de uso que tiene originalmente y por tanto siga estando libremente disponible para todo el mundo (licencias tipo copyleft, como la GPL de la Free Software Foundation). A diferencia de lo que sucede con el software propietario cuyo único objetivo es la rentabilidad económica y no el hacer buenas herramientas, la comunidad del software libre se empeña en la búsqueda de una buena adecuación entre las necesidades y el propio producto, esto es, busca a la vez la calidad y la eficiencia social que otorga la libertad de uso. El software libre, incluso en los casos en que se comercializa, siempre es más barato que sus equivalentes propietarios, y casi siempre acaba estando disponible gratuitamente (ALBERTO MOLPECERES TOURIS 18/08/2002).

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

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

1.7 Generación de informes en los SCADAs.

En la actualidad existe una gran cantidad de proyectos enfrascados en el desarrollo de generadores de informes. Aunque ninguno de los encontrados se enfoca directamente en la generación de informes para un SCADA, no es menos cierto que este es uno de los procesos inherentes al SCADA que más se aproxima a aplicaciones informáticas estándares, pues el generador de informes en un SCADA es la herramienta que permite emitir informes en un formato personalizado de tal manera que le sea útil al personal al cual va dirigido, igual a cualquier sistema de gestión, aunque con algunas peculiaridades propias de esta aplicación. Concretamente, sirve de puente entre la base de datos histórica del SCADA y los diseños de informes que el usuario previamente configura de acuerdo a sus necesidades para volcar, filtrar y analizar datos. De aquí, que un estudio de los resultados alcanzados en estos proyectos resulta de mucha utilidad para desarrollar el generador de informes necesario para el proyecto

SCADA Nacional, incluso teniendo presente la posible adopción de algunos resultados de estos proyectos como soluciones para el que nos ocupa.

A pesar de la abundante existencia de proyectos sobre software libre para el desarrollo de generadores de informes, no deben descartarse las ideas que pudieran tomarse de algunos software propietarios bien asentados en el mercado. Por estas razones en el presente trabajo se hace un análisis de las principales características de algunos generadores de informes tanto propietarios como de software libre.

En el anexos 1 del presente trabajo se analizan de forma crítica las principales características de un conjunto de generadores de informe ampliamente utilizados, haciendo énfasis en los desarrollados bajo el paradigma de software libre.

Además en el anexo 2 se encuentran resumidas las principales características de los generadores de informe comerciales y no comerciales estudiados, entre las que se destacan si la aplicación es o no un diseñador de informes, si contiene un gestor de base datos, si permite conectar con una o mas bases de datos, las plataformas en las que pueden trabajar y los formatos de salidas de los informes.

Cada uno de estos generadores de informes tiene características muy peculiares que limitan su uso integro para dar solución al problema planteado de generar informes para el SCADA Nacional de PDVSA. Las principales razones que llevan a descartarlos como solución se pueden resumir en:

- Ser software propietario.
- No tener acceso al código de la aplicación.
- No estar desarrollados en el lenguaje de programación exigido para el desarrollo.
- No brindar interfaces para ser usados desde los demás módulos del SCADA.
- No permitir la conexión a bases de datos Postgrees, entre otras

En las cuales no abundamos en estos momentos, pues en el capitulo 3, donde aparecen los requisitos planteados para la aplicación, es que se pueden contrastar las características de cada uno de ellos con los requisitos impuestos para el desarrollo, complementando de esta forma la afirmación de descartarlos como solución al problema planteado.

1.8 Generalidades sobre los generadores de informes.

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

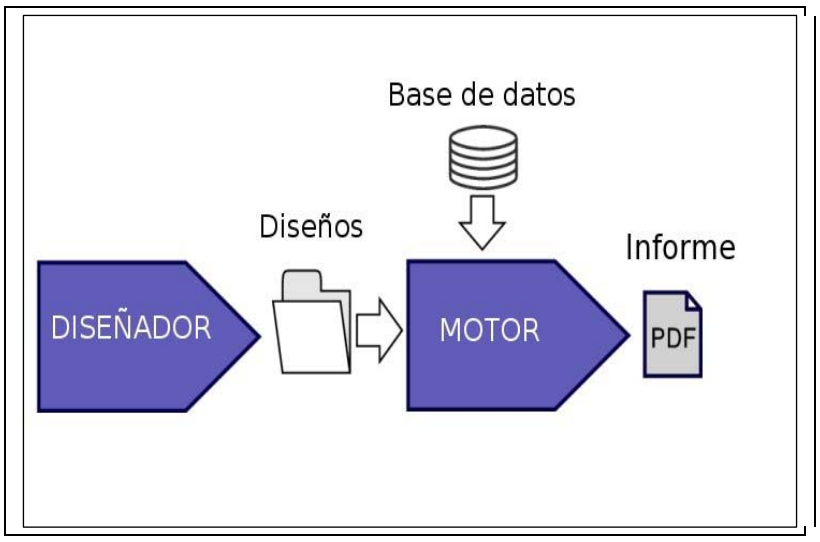

Fig. 3. Estructura general de un generador de informes.

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

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

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

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

1.9 Conclusiones.

En este capítulo se ha descrito el objeto de estudio de este trabajo, centrándose en las peculiaridades de los generadores de informes para SCADAs. Se ha expuesto la necesidad de desarrollar el generador de informes para el SCADA Nacional de PDVSA y se ha brindado una visión general de las principales funcionalidades que debe brindar un generador de informes, partiendo del análisis realizado a varios generadores de informes existentes, los cuales fueron descartados como solución al problema planteado.

CAPÍTULO 2

2 TENDENCIAS Y TECNOLOGÍAS ACTUALES.

2.1 Introducción.

En el presente capítulo se hace un análisis de las tecnologías y tendencias que existen en la actualidad a nivel mundial que pudieran ser útiles en el desarrollo de nuestro trabajo y para un posterior uso. Se tienen en cuenta la política de migración hacia el software libre en el ámbito nacional, las aplicaciones de los generadores de informes en los SCADA, los lenguajes de programación para el SCADA, los sistemas gestores de bases de datos mayormente utilizados, las distintas metodologías de desarrollo de software, basadas principalmente en metodologías ágiles y el RUP, las bibliotecas graficas para la realización del trabajo sobre software libre como QT y GTK, y el estudio de los IDEs (Entorno de desarrollo integrado).

2.2 La política de migración hacia software libre.

Recientemente se ha venido observando una tendencia en la Universidad de las Ciencias Informáticas, y en el país en general, hacia la utilización en grado creciente de software libre. Cada vez se promociona más la migración desde los sistemas con licencia comercial que están en poder de unos pocos monopolios de la rama de la informática y que se reservan el derecho de vender sus productos a quienes consideren pertinente, hacia aquellos que se denominan “libres”, es decir, aquellos cuyo uso por parte de cualquiera que esté interesado en hacerlo está exento de pago.

Como institución de avanzada en el campo de la informática, la UCI está prácticamente obligada a llevar a cabo, y cuánto antes mejor, esta migración. El presente trabajo siendo un modulo del proyecto SCADA, parte de esa premisa y se propone la construcción de un sistema que satisfaga las necesidades que lo originaron, haciendo uso de herramientas y tecnologías libres. Después de un amplio análisis con el cliente y una valoración de las causas que originaron este proyecto, se propuso trasladar todo el sistema SCADA a software libre para a la vez no depender de ninguna compañía extranjera, ni hacer uso de tecnologías por las cuales se tenga que pagar para obtener una licencia de un software. Basado en las entrevistas que se tuvieron con los clientes de la empresa de PDVSA se determino establecer dicho sistemas SCADA bajo la plataforma de GNU/Linux y hacer uso de tecnologías bajo código abierto.

2.3 Aplicaciones de Generadores de Informe en SCADAs.

Dentro del mundo de las aplicaciones de software empresariales, principalmente los SCADA, los generadores de informe constituyen una herramienta fundamental a la hora de fomentar su trabajo. De aquí que la mayor parte de las aplicaciones de empresa requieren la presentación impresa de gran cantidad de datos para que el equipo de análisis del departamento correspondiente pueda sacar sus conclusiones sobre esos datos.

Una vez analizado esto podemos decir que un informe nos es más que la unión de varios listados distintos o varias veces el mismo, y a la vez, de comentarios con el fin de agrupar información que se encuentra almacenada en una base de datos, mas específicamente dentro de este, el listado es una consulta a una determinada base de datos que nos permite obtener la información para su posterior inserción en un informe de una determinada forma, tanto como tabla o como gráfica y un comentario o aclaración es una forma de ver la información que se muestra en el informe, ya que estos pueden ser incluidos como se considere oportuno.

Con el transcurso del tiempo muchos desarrolladores se han dado cuenta de la importancia de tener buenas herramientas para la rápida creación de informes, y aún más importante, para su rápida modificación y mantenimiento.

En la actualidad existen programas comerciales potentísimos capaces de procesar grandes cantidades de datos a una alta rapidez y generar informes de forma atractiva que permitan hacer un análisis posterior de los datos.

2.3.1 C++

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

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

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

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

2.4 Selección del lenguaje a utilizar.

Dentro de los lenguajes de programación que existen, varios son utilizados en el desarrollo de generadores de informes para los SCADAs actuales, en nuestro caso hacemos uso del lenguaje C++ por su sintaxis familiar y por los acuerdos llevados a cabo antes de la realización del software.

2.5 Lenguaje de marcado extensible (Extensible Markup Language, XML).

“XML está revolucionando la comunicación entre aplicaciones o, de forma más general, la comunicación entre equipos, pues ofrece un formato de datos universal que permite adaptar o transformar fácilmente la información.”(BANKHACKER.COM)

XML intenta ser un formato absolutamente genérico, con el que describir cualquier tipo de fichero. Por ejemplo SVG 1.0, Scalable Vector Graphics, formato para representar imágenes vectoriales; DSML 1.0, Directory Services Markup Language, específico para definir directorios y XHTML 1.0, Extensible HyperText Markup Language, formato para representar páginas Web, están basados en XML. Tiene la ventaja de que cualquier información transmitida por un XML está perfectamente estructurada. Para ello se emplea, al igual que el HTML, un lenguaje de marcas, pero las de XML no son fijas, sino variables, lo que constituye la base de su generalidad.

2.6 Sistemas Gestores de Bases de Datos (SGBD).

“Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y Data Base Management System, su expresión inglesa.” (WIKIPEDIA)

En la actualidad existe una gran variedad de SGBD, tanto de tipo comercial como libre. Entre los más usados dentro del grupo de los comerciales se encuentra **Oracle**, el cual es considerado el SGBD más completo que existe. Sus características más destacadas son el soporte de transacciones, su gran estabilidad y seguridad, su escalabilidad, así como que es un sistema multiplataforma, entre otras ventajas. “Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias).” (WIKIPEDIA)

En sus inicios fue muy revolucionario dado que usaba la filosofía de bases de datos relacionales, algo que por los años 70, fecha en que surge Oracle, era todavía desconocido. Hasta hace poco su dominio en el mercado de los servidores de bases de datos empresariales era casi total, pero recientemente está sufriendo la competencia del MS SQL Server de Microsoft y de la oferta de otros SGBD libres.

SQL Server es un potente SGBD que está totalmente habilitado para Web. “Ostenta marcas de referencia en cuanto a escalabilidad y confiabilidad, que son críticas para el éxito de bases de datos de gran tamaño. El SQL Server permite lograr una gran velocidad en el procesamiento de transacciones, y agilidad en todas sus operaciones.” (YUNIER SOTO LÓPEZ junio 2004) A pesar de todas las ventajas que presenta este SGBD, tiene el inconveniente de que, al igual que Oracle, no es un sistema libre.

MySQL “implementa funcionalidades Web que permiten un acceso a los datos, seguro y fácil, desde Internet.” (GONZÁLEZ) “Es uno de los SGBD más populares, desarrollado bajo la filosofía de código abierto.” (WIKIPEDIA)

La licencia GPL de MySQL obliga a distribuir cualquier producto derivado (aplicación) bajo esa misma licencia. Por tanto MySQL tiene sus restricciones: sólo es gratis si se está dispuesto a distribuir la aplicación que se quiere desarrollar bajo esa misma licencia GPL. Si se desea distribuir la aplicación comercialmente, entonces se debe pagar la licencia comercial de MySQL que permite hacer exactamente eso.

MySQL tiene como una de sus principales ventajas la velocidad en la lectura de datos, pero a costa de eliminar un conjunto de facilidades que presentan otros SGBD: integridad referencial, bloqueo de registros, procedimientos almacenados, entre otros. En recientes versiones de MySQL (la versión 4 y la 5) se incluyen algunas de estas características, pero indudablemente esto va en detrimento de la velocidad.

Por otra parte está **PostgreSQL** que está considerado como el SGBD de código abierto más avanzado del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales de alto calibre tales como DB2 u Oracle (GONZÁLEZ).

Es un SGBD objeto-relacional, aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Su avanzada funcionalidad se pone de manifiesto con las consultas SQL declarativas, el control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia y valores no atómicos (atributos basados en vectores y conjuntos).

Es altamente extensible: soporta operadores y tipos de datos definidos por el usuario. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins)

SQL92. Cuenta con una API (del inglés Application Program Interface) flexible lo cual ha permitido dar soporte para el desarrollo con PostgreSQL en diversos lenguajes de programación entre los que se incluyen: Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike. Tiene soporte para lenguajes procedurales internos, incluido un lenguaje nativo denominado PL/pgSQL, el cual es comparable con el lenguaje procedural de Oracle PL/SQL.

A diferencia de MySQL que como se explicó anteriormente, tiene sus restricciones en cuanto al tema de las licencias, PostgreSQL es totalmente libre. Las polémicas entre los partidarios de MySQL y los defensores de PostgreSQL pueden clasificarse como del tipo “Guerra Santa”, junto a otras como GNU/Linux vs. Microsoft Windows, Mac. Vs. PC, etc. Muchos desarrolladores en sus discusiones a través de la Web en torno al tema de qué es mejor: MySQL o PostgreSQL, recomiendan la utilización de PostgreSQL para la elaboración de un sistema robusto y para lograr mayor escalabilidad. La mayoría coincide en que cada SGBD tiene sus ventajas y desventajas, y que la elección de uno de los dos depende de lo que se quiera construir. Se destaca sobre todo que MySQL ha avanzado vertiginosamente comparado con PostgreSQL que ya lleva alrededor de 15 años de desarrollo.

2.6.1 Selección del SGBD a utilizar.

Basado en los argumentos anteriormente expuestos se considera que la mejor variante es la de utilizar PostgreSQL (específicamente la versión 8.1) como SGBD, ya que reúne las dos importantes características de ser un sistema robusto con variadas funcionalidades y de ser un sistema de código abierto.

2.7 Metodologías de Desarrollo de Software.

Uno de los temas más comunes en el mundo de la informática hoy en día es el de las metodologías de desarrollo de software: cómo trabajar eficientemente evitando las catástrofes que conllevan al fracaso de un gran porcentaje de proyectos. Una metodología tiene como objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, por medio de una mayor transparencia y control sobre el proceso; “producir lo esperado en el tiempo esperado y con el coste esperado” (MOLPECERES)

Durante los últimos años se han desarrollado dos corrientes en lo referente a las metodologías de desarrollo de software, las llamadas “pesadas” y las llamadas “ligeras o ágiles”. Las primeras se basan en la idea de conseguir el objetivo común por medio de orden y documentación, mientras que las segundas tratan de lograrlo por medio de la comunicación directa e inmediata entre aquello que intervienen en el proceso. Analizaremos a continuación tres de las más conocidas, sus características, ventajas y desventajas.

2.7.1 Metodologías “Pesadas”.

Las llamadas metodologías pesadas para abordar el desarrollo de software han demostrado ser efectivas y necesarias en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. En estas se lleva a cabo una definición del proceso en cuanto a roles actividades y artefactos. Mediante estas también se maneja una gran cantidad de documentación y actividades extras de la programación.

R-2.7.1.1 El Proceso Unificado de Rational (RUP):

RUP son las siglas en inglés de Rational Unified Process. Esta es una de las metodologías más generales de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto, no sólo de software. Es el producto final de treinta años de desarrollo y uso práctico. Se basa en casos de uso para describir lo que se espera del software y está muy orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML (Unified Modeling Language) como herramienta principal.

RUP “se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes” (JAMES JACOBSON 2004). Cada uno de estos ciclos se divide en cuatro fases (Inicio, Elaboración, Construcción y Transición) en cada una de las cuales se llevan a cabo una o varias iteraciones, cada una de las cuales resulta un incremento de la precedente, por lo que se dice que RUP es iterativo e incremental.

2.7.2 Metodologías “Ágiles”.

Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías ágiles a los desarrolladores. Sin embargo, hay que tener presente una serie de inconvenientes y restricciones para su aplicación, tales como: están dirigidas a equipos pequeños o medianos, el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc.

R-2.7.2.1 Programación Extrema (XP):

La Programación Extrema, o Extreme Programming, es otra de las metodologías de desarrollo de software que existen en la actualidad. “Mientras que el 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.” (MOLPECERES)

Supone la disposición en todo momento, por parte del equipo de trabajo, de un representante competente del cliente, que debe estar en condiciones de dar una respuesta rápida y correcta a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones.

La base para el desarrollo del software que usa esta metodología son las llamadas User Stories, historias escritas por el cliente en las que describe escenarios sobre el funcionamiento del sistema y que no sólo están limitados a la interfaz de usuario, sino que también pueden describir modelos, dominio, etc. Estas User Stories junto a la arquitectura que se persigue, sirve de base para crear un plan de “entregas de software” entre el equipo de desarrollo y el cliente, para cada una las cuales se definen objetivos y las iteraciones (generalmente cortas) necesarias para cumplirlos. Las User Stories y los casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador.

Esta metodología apuesta por iteraciones cortas que generan software que el cliente puede ver.

La codificación del software se realiza siempre en parejas (dos programadores, un ordenador), las cuales no son fijas sino que rotan a lo largo del proyecto, y el código que escriben no les pertenece sólo a ellos sino al equipo completo. El objetivo ideal sería que cada integrante del equipo trabaje al menos una vez con cada uno de los demás integrantes y con cada componente software, de forma que el conocimiento de la aplicación completa lo posea el equipo entero y no unos pocos miembros. Se programa sólo la funcionalidad requerida para la entrega en curso, se trabaja en función de las necesidades del momento, por lo que no se le da importancia al análisis como fase independiente.

R-2.7.2.2 Desarrollo Guiado por la Funcionalidad (FDD).

Se podría considerar que FDD (por las siglas en inglés de Feature Driven Development) está a medio camino entre RUP y XP, aunque en realidad es más bien una metodología ligera. Está pensada para proyectos con un tiempo de desarrollo relativamente corto (menos de un año). Se basa en un proceso iterativo con iteraciones cortas de aproximadamente dos semanas que producen un software funcional, el cual puede ser examinado por el cliente y la dirección de la

empresa. Cada iteración se define en término de funcionalidades (de ahí su nombre) que son pequeñas partes del sistema con significado para el cliente.

Un proyecto que siga esta metodología estará dividido en cinco fases: desarrollo de un modelo general, construcción de la lista de funcionalidades, plan de entregas sobre la base de las funcionalidades a implementar, diseño basado en las funcionalidades e implementación basada en las funcionalidades. Todo el trabajo se realiza en grupo, aunque siempre hay un responsable, que generalmente tiene mayor experiencia, que dice la última palabra en caso de no llegar a un acuerdo.

Las funcionalidades de cada entrega se dividen entre los distintos subgrupos del equipo y se implementan. El código escrito (las clases) tiene propietario, o sea, sólo quien lo crea puede modificarlo, lo que no ocurre en XP. Es por eso que en un subgrupo deben estar todos los propietarios de las clases implicadas, pudiendo un desarrollador pertenecer a varios subgrupos. También se contemplan como parte del proceso de implementación, la preparación y ejecución de pruebas, las revisiones de código y la integración de las partes que componen el software.

2.7.3 Selección de la metodología a utilizar.

Las tres metodologías analizadas tienen pocas similitudes entre si, aunque XP y FDD tienen algunas más al ser ambas del tipo de metodologías ágiles, orientadas al cliente y de iteraciones cortas. Por otra parte, es importante mencionar que algunos autores consideran que dado el carácter general de RUP, todas las otras metodologías son casos particulares de esta. Pero siempre se pueden comparar si se definen determinados criterios.

En cuanto a la forma en que se capturan los requisitos, RUP y XP crean como base los use cases (casos de uso), los cuales describen los requerimientos de la aplicación desde el punto de vista del cliente, definen los requisitos técnicos sin abordar detalles de implementación. FDD, por el contrario, no define la parte del proyecto en la que se obtienen los requisitos y sólo define el proceder a partir del momento que ya estos han sido identificados. La captura de requisitos es una parte vital de cualquier proceso de desarrollo de software, puesto que garantiza la eficacia de éste: que el producto final sea lo que espera el cliente.

Por otra parte, XP es un proceso muy orientado a la implementación, en el que se genera poca documentación y en que la funcionalidad exacta del sistema final no se define nunca formal y contractualmente. Es por eso que este método es más aplicable para desarrollos internos.

En el caso particular de este proyecto, se decidió la utilización de la metodología RUP, por todas las ventajas de organización que brinda y porque las otras metodologías estudiadas presentan ciertas debilidades que, a juicio del equipo de trabajo, representan riesgos considerables, como es el caso de una posible captura de requisitos no adecuada.

2.8 Arquitectura de 3 capas.

Muchas aplicaciones de gestión se pueden desarrollar utilizando una arquitectura de 3 capas. Este modelo propone un ambiente para la construcción y ejecución de aplicaciones de avanzada. Asegura que las aplicaciones puedan correr en ambientes pequeños o grandes, y puedan acompañar el crecimiento de las entidades que las utilizan. En las aplicaciones diseñadas usando un modelo de tres capas, el sistema es dividido en datos, negocio y presentación. La idea de esta arquitectura está basada principalmente en la capacidad de estabilidad que nos ofrece.

Datos: La capa de datos tiene como misión la administración de la información que maneja el sistema. Esto incluye el almacenamiento, la actualización y la consulta de todos los datos contenidos en el sistema, es por esto que contiene a la base de datos y las clases de acceso a la misma.

Negocio: El comportamiento de la aplicación es definido por los componentes que modelan la lógica de negocio. Estos componentes reciben las acciones a realizar a través de la capa de presentación, y llevan a cabo las tareas necesarias utilizando la capa de datos para manipular la información del sistema. Los servicios de esta capa son encapsulados en 2 paquetes, las Entidades del Negocio, que representan objetos que van a ser manejados o consumidos por toda la aplicación, y Lógica del Negocio que contienen las clases principales relacionadas con dicho negocio

Presentación: La capa de presentación representa la parte del sistema con la que interactúa el usuario.

Como se puede ver esta arquitectura permite que tanto la interfaz de usuario, las reglas de negocios y el motor de datos se conviertan en entidades separadas unas de otras, manteniendo bien definidas las interfaces que cada una de estas exponen para comunicarse con la otra. Los componentes y servicios creados según este modelo pueden compartirse y reutilizarse, por lo que las aplicaciones que lo usan, alcanzan una mayor capacidad de crecimiento.

2.9 Biblioteca Gráfica QT.

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

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

Qt cuenta actualmente con un sistema de doble licencia: una GPL para el desarrollo de software de código abierto (*open source*) y software libre, y otra de pago para el desarrollo de aplicaciones comerciales.

Qt facilita la programación orientada a objetos y la creación de componentes, proporcionando sólidos cimientos para la construcción de cualquier tipo de aplicación gráfica.

2.10 Biblioteca Gráfica GTK.

GTK (GIMP Toolkit) es una biblioteca multiplataforma para crear interfaces gráficas de usuario. Su licencia es la LGPL, así que mediante GTK podrá desarrollar programas con licencias abiertas, gratuitas, libres y hasta licencias comerciales no libres sin mayores problemas.

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

2.11 Selección de bibliotecas gráficas a utilizar.

Una vez analizadas las bibliotecas graficas anteriores (**Qt, GTK**) hemos decidido hacer uso la biblioteca Qt. Esta selección viene estrechamente vinculada a que trabaja sobre software libre y está especializada en crear interfaces de usuario, además de estas características posee un framework más completo y con mucha más utilidad a la hora de un desarrollo rápido, hacen uso del lenguaje de programación C++ y posee licencias GPL proporcionando su libre acceso para el trabajo. Por otra parte se puede ver también que los generadores de informes a reutilizar están desarrollados sobre Qt.

2.12 Biblioteca Boost.

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

- Cadenas y procesamiento de texto

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

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

2.13 Estudio de IDEs

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

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

En la actualidad existe una gran variedad de IDEs. Entre los más usados por los desarrolladores de GNU/Linux tenemos el **KDevelop,** realizado para sistemas GNU/Linux y otros sistemas Unix, publicado bajo licencia GPL y se destaca por poseer un entorno de desarrollo muy avanzado y por soportar muchos lenguajes y características avanzadas. A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio para producir código binario, por lo que depende de una serie de compiladores que se consideran estándar para los sistemas operativos derivados de UNIX, que también poseen

licencia GPL. Su última versión se encuentra actualmente bajo desarrollo y funciona con distintos lenguajes de programación como C, C++, Java, Ada, SQL, Python, Perl y Pascal. Además de estas características también permite el control de versiones, mantiene el completado automático de código en C y C++, tiene asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación, posee un navegador entre clases de aplicación, tiene un editor de código fuente con destacado de sintaxis e identado automático y está integrado con Qt para realizar interfaces gráficas para aplicaciones multiplataforma.

Qdevelop por su parte es una plataforma integrada del ambiente de desarrollo dedicado enteramente a Qt4. Para poder compilar con QDevelop se hace necesario tener la version 4 (de Trolltech) o Qt4. Según muchos desarrolladores la mayoría de las distribuciones de GNU/Linux empaquetan normalmente ya la versión Qt4. Principalmente corre sobre las plataformas de GNU/Linux y Microsoft Windows 2000/XP/Vista. Además posee otras características entre las cuales se destaca que posee un completado de código de gran alcance, tiene un buscador que permite al programador que hojee, navegue, o visualice la estructura de clases, de miembros y de funciones de proyectos.

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

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

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

Qtdesigner es un entorno integrado de desarrollo (IDE) que proporciona una herramienta para el diseño de interfaces. En su interior utiliza un documento XML para describir y almacenar la interfaz. Este documento sigue un formato propio denominado UI (User Interface), cuyo esquema establece una serie de elementos XML para los propios componentes,

propiedades, funciones, entre otras características de la interfaz. Qt designer se suele utilizar en conjunto con KDevelop, aunque es multiplataforma. Sirve para generar ficheros .ui, que contienen la interfaz gráfica de un programa que utilice el toolkit Qt (biblioteca grafica). Gracias a las utilidades de la biblioteca Qt, se puede generar código automáticamente a partir de los ficheros creados con Qt designer. Además posee otras características especiales entre las cuales se destacan la generación dinámica de formas, contiene plantilla de formas predefinidas con diferentes estilos en formato XML, tiene un editor de propiedades de widgets y presenta una herramienta de desarrollo visual nombrada WYSIWYG.

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

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

Anjuta es un entorno integrado de desarrollo (IDE) para programar en C y C++ en sistemas GNU/Linux bajo licencia GPL. Su principal objetivo es trabajar con GTK y en el *escritorio* GNOME, además ofrece un gran número de características avanzadas de programación. Anjuta es software libre, liberado bajo la licencia GPL. Además incluye un administrador de proyectos, asistentes, plantillas, depurador interactivo y un poderoso editor que verifica y resalta la sintaxis escrita. A finales de 2004, el código base de Anjuta alcanzó una cierta estabilidad y otros desarrolladores se interesaron otra vez en el proyecto. Ahora muchas de las funcionalidades del entorno IDE están funcionando, mejorando cada día.

Entre sus principales características están que cuenta con un nuevo sistema de extensiones y posee una arquitectura revisada y extensible, posee un nuevo interprete de comandos propio y documentación del API con un nuevo sistema de ayuda, tiene un nuevo administrador de tareas y una nueva extensión para añadir macros e insertar texto predefinido o personalizado,

además contiene un administrador de sesiones de trabajo y otras diversas mejoras o actualizaciones.

2.14 Propuesta.

Luego del análisis llevado a cabo, se puede plantear una propuesta que consiste en desarrollar la aplicación del generador de informes sobre el software SCADA, que siga el modelo de los generadores de informes existentes y que incluya además otras funcionalidades, utilizará como lenguaje de programación a C++ dada su portabilidad y eficiencia. Se propone también la utilización de PostgreSQL como SGBD. Como metodología de desarrollo se utilizará RUP.

2.15 Conclusiones.

En este capítulo se hizo un análisis de las tecnologías a utilizar en el desarrollo de la propuesta de solución, así como algunos conceptos y tendencias que esta debe adoptar para sí. Se expusieron características del lenguaje de programación, el sistema gestor de bases de datos, entorno integrado de desarrollo (IDE) para programar, y la metodología de desarrollo de software; así como el uso de otras tecnologías. A partir de estos puntos se comenzará el desarrollo de la propuesta de sistema.

CAPÍTULO 3

3 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción.

En el presente capítulo se describe la propuesta de solución. Primeramente se desarrolla el modelo de dominio del negocio donde se describen los principales conceptos del mismo. Se plantean los requisitos funcionales y no funcionales de la aplicación a desarrollar y se modela la misma en términos de casos de uso de sistema.

3.2 Modelo del dominio.

En dependencia de la situación o escenario que se presente, hay varias alternativas para expresar el contexto del sistema en una forma entendible para los desarrolladores de software, las principales alternativas están en realizar un modelado del dominio o un modelado del negocio. Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno en el que trabaja el sistema y enlaza estos objetos unos con otros.

Se realizó un modelo del dominio debido a las características de nuestro sistema, y principalmente a el tiempo con se contaba para su desarrollo, además de no presentar procesos de negocios bien definidos.

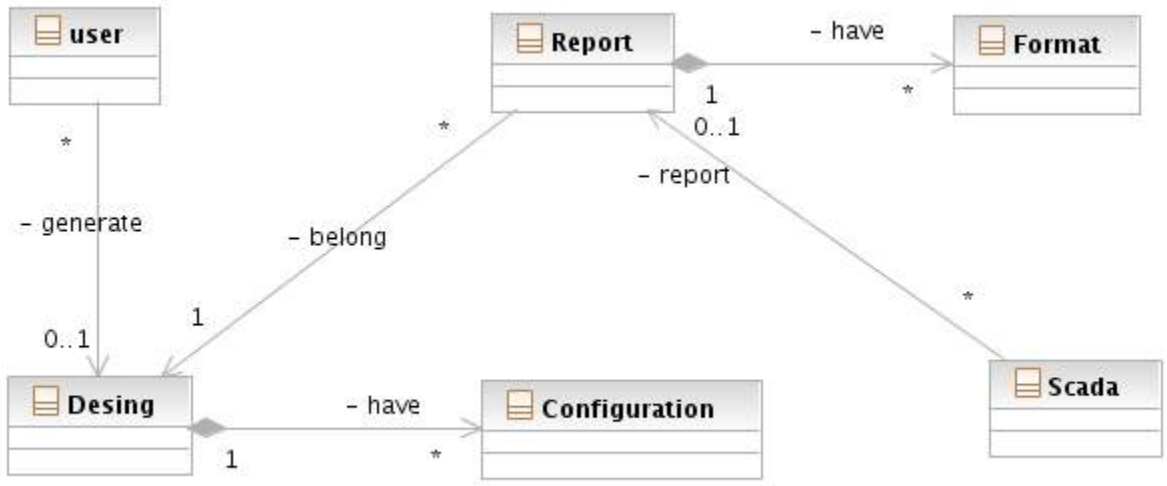

Fig. 4. Diagrama de clase del modelo de dominio.

3.3 Modelo del sistema.

A partir de este punto se comienza a modelar el sistema que se va a construir. Para ello se identifican sus requisitos, tanto funcionales como no funcionales, y se modelan los funcionales en términos de casos de uso del sistema.

3.4 Requerimientos funcionales.

3.4.1 Diseño de informes.

R-3.4.1.1 Reportes a generar.

El sistema debe permitir generar los siguientes reportes (SCADA, EQUIPO PROYECTO 2005):

- Reportes de eventos secuenciales.
- Combinación de datos de proceso de diferentes fuentes (tendencias, valores PV, etc.).
- Reportes de falla.
- Reportes de eventos del sistema.
- Reportes históricos.
- Reportes de libre configuración por el usuario autorizado para tal fin.
- Reportes de todas las alarmas: Un reporte de todas las alarmas existentes.
- Frecuencia de alarmas: Reporte de las 10 alarmas mas frecuentes.
- Alarmas reconocidas: Reporte que lista todas las alarmas que han sido reconocidas en un periodo de tiempo inicio a tiempo fin configurables.
- Alarmas no reconocidas: Reporte que lista todas las alarmas que no han sido reconocidas en un periodo de tiempo inicio a tiempo fin configurables.
- Alarmas deshabilitadas: Reporte que liste todas las alarmas deshabilitadas.

Para lo cual debe proporcionar un conjunto de plantillas a partir de las cuales poder personalizar los diseños.

R-3.4.1.2 Selección de datos a reportar.

El sistema debe posibilitar al usuario la selección de aquellos datos que son necesarios para generar el informe que se desee.

Solo deben estar disponibles para esta acción los datos con permiso de lectura para el perfil del usuario autenticado en el sistema, es decir, solamente deben mostrarse dichos datos y mantener ocultos al resto (no accesibles), facilitando de esta manera la selección de los mismos para incluirlos en el informe.

R-3.4.1.3 Mecanismos de filtraje sobre los datos.

El sistema debe permitir el establecimiento de criterios de filtrado sobre el conjunto de datos que se van a utilizar para generar el informe, de tal manera que sólo se informe la información

realmente valedera para el interesado. Los filtrados a realizar sobre los datos que debe brindar el subsistema son:

- Cota de tiempo a reportar: Se especifica el rango o ventana de tiempo que interesa plasmar en el informe (inicio/fin del tiempo a reportar).
- Filtrado por frecuencia: No se tienen en cuenta todos los valores almacenados del dato en cuestión sino algunos representativos de acuerdo a la frecuencia de remuestreo establecida. (FOUNDATION 2003)
- Filtrado por característica de interés: Sólo se tienen en cuenta aquellos datos que cumplen con alguna o algunas características en particular: tipos de puntos, severidad, área, estados, alarmas, analógicos o digitales, recolección por excepción o por muestreo, por límite de desviación, nivel de acceso, calidad del dato.

R-3.4.1.4 Aplicación de expresiones sobre los datos a procesar.

El objetivo fundamental perseguido en un informe es consolidar datos para convertirlos en información fácil de interpretar por la persona a la cual va dirigido dicho informe. En una infraestructura productiva industrial estas personas pueden agruparse según sus funciones en las siguientes categorías:

- Directivos, jefes en sus distintos niveles y estructuras (directores generales, jefes de turno, responsables de área, jefes técnicos, directores de producción, etc.).
- Personal de asistencia a la producción (laboratorio, calidad, etc.).
- Supervisor técnico o tecnológico (tecnólogo).
- Personal de ingeniería.
- Mantenedores del sistema.
- Operadores.
- Auditores.

Para lograr que un SCADA consolide esta información se debe prever la necesidad de incorporar formulas matemáticas evaluables en los diseños de informes a generar, que permitan personalizar la información a la que se tiene acceso. Dicho de otra manera, el subsistema debe facilitar el procesamiento de los datos para generar información que sea útil a los destinatarios finales de dichos informes. De acuerdo a las características de la información requerida por los grupos de usuarios enunciados anteriormente, las operaciones fundamentales sobre conjuntos de datos que debe brindar el generador de informes son de naturaleza estadística:

- Integración (Totalización).
- Mínimo dentro de un conjunto de valores.

- Máximo dentro de un conjunto de valores.
- Media ponderada en el tiempo.
- Rango de variación.
- Desviación típica.
- Varianza.
- Tiempo fuera de rango de una variable (Tiempo por encima o por debajo de un valor prefijado).
- Tiempo de permanencia de una variable digital en un estado.
- Conteo de sucesos.
- Para operar sobre valores puntuales (no conjuntos de datos), el sistema debe brindar otras operaciones matemáticas clásicas como potenciación, radicalización, trigonométricas, etc.
- Interpolación partiendo de valores relacionados en tablas.

R-3.4.1.5 Incorporación de gráficos estáticos.

En el informe debe brindarse la alternativa de incorporar información gráfica estática. Esto se refiere a la posibilidad de insertar imágenes importadas en formatos gráficos estándares (jpg, gif, bmp, etc.) como pueden ser logotipos, esquemas del sistema y gráficos de tendencia.

R-3.4.1.6 Generación de gráficos a partir de los datos.

A partir de los datos importados para el informe se debe brindar la posibilidad de generar gráficos típicos, que permitan una valoración más visual y comparativa de la información reportada, los posibles gráficos a realizar son:

- Gráficos de barra.
- Gráficos de línea continua.
- Gráficos de pastel.

R-3.4.1.7 Nombres genéricos para los diseños de informes.

Debe establecerse un nomenclador para los diseños de informes que permita identificar características de los mismos como pueden ser tipo, creador, nombre o código de quien lo crea, región geográfica, área o nodo de interés, etc.

Pudiera emplearse un nomenclador similar al propuesto en la “Plataforma SCADA PDVSA. Abril de 2005” (SCADA, EQUIPO PROYECTO 2005) para los puntos de datos, formado por los siguientes elementos claves:

AAAAAAAA-BBBBBB-CCC-DDD

AAAAAAAA Tipo de informe

BBBBBB	Nombre de la Infraestructura de producción.
BBB	Tipo de infraestructura de producción
CCC	Región Geográfica

Ejemplo: Producción Turno-Poso1-Extracción-Maracaibo

R-3.4.1.8 Nivel de privilegio para hacer uso de los diseños de informe.

Al configurar el diseño de informe debe especificarse el o los perfiles de usuario que tendrán acceso al mismo, así como los privilegios que tendrá.

R-3.4.1.9 Destino del informe.

En el diseño del informe debe quedar especificado el destino por defecto que tendrá el informe una vez que sea generado, es decir, el lugar dentro de la estructura de almacenamiento donde se colocará o el medio que se empleará para su propagación o impresión.

3.4.2 Generación de informes.

R-3.4.2.1 Generación automática del informe por criterio temporal.

El sistema debe satisfacer la necesidad de emitir informes de forma automática de acuerdo a configuraciones de tiempo, es decir, llegado un instante de tiempo, previamente establecido en el programador de tareas del SCADA, se debe generar el informe solicitado.

R-3.4.2.2 Generación del informe por demanda:

El sistema también debe permitir la generación de informes debido a la ocurrencia de eventos internos o externos.

Se consideran eventos internos a estos efectos, la ocurrencia de determinadas situaciones de alarma.

Se consideran eventos externos a estos efectos, la solicitud de un usuario.

R-3.4.2.3 Nombres genéricos para los informes.

Los nombres de los informes que persistirán en el disco duro deben ser lo mas intuitivo posibles y tener una estructura genérica de tal forma que una lectura de los mismos puede servir para identificar características generales de los mismos como pueden ser tipo, agente generador, fecha de generación (“time stamp”), etc. Este mismo principio debe seguirse en aquellos informes que se enviarán a los usuarios, sin que sea necesario su persistencia en el disco duro, como son los e-mail, mensajes a móviles y localizadores, etc, de tal forma que el receptor del mismo pueda identificar la naturaleza del mensaje que está recibiendo de forma rápida e intuitiva. Por lo cual debe establecerse un nomenclador para los informes generados.

Estos nombres pudieran tener una estructura como la siguiente:

Nombre diseño de informe –EEEE-YYYY:DD:MM-HH:MM

Nombre diseño de informe

EEEE	Agente generador (Programador de informes periódicos, código del usuario que solicita el informe para el caso de informes generados bajo demanda o excepción que lo invocó).
YYYY-DD-MM-HH:MM	Fecha y hora de solicitud de generación (No son los atributos de tiempo de creación del fichero pues puede suceder que un informe no se genere a la hora exacta que se solicita, pero su contenido si debe corresponderse con la información contenida en este campo).

Ejemplo: Producción Turno-Poso1-Extracción-Maracaibo-Periódico- 2006:20:10-24:00

Informe generado de forma automática debido a una solicitud del programador de tareas del SCADA que tuvo lugar a las 12 de la noche del día 20 de octubre del 2006.

Producción Turno-Poso1-Extracción-Maracaibo-Op15- 2006:20:10-24:00

Informe generado a partir de la petición realizada por el Op15 (Operador #15) a las 12 de la noche del día 20 de octubre del 2006.

R-3.4.2.4 Persistencia de los informes generados.

Con el ánimo de mantenimiento del sistema de archivos esta es una característica que el sistema debe permitir especificar para aquellos informes que se requiere almacenar en disco duro u otro medio de almacenamiento. Es decir, debe brindarse la posibilidad de especificar si los informes persistirán o no en los soportes de almacenamiento, así debe permitir especificar si un informe se solicita con un carácter temporal (susceptible a ser borrado o reemplazado) o si es necesario su almacenamiento de forma permanente.

R-3.4.2.5 Nivel de privilegio sobre los informes generados.

Al generar un informe debe especificarse el o los perfiles de usuario que tendrán acceso al mismo, así como los privilegios que tendrán sobre dicho informe.

R-3.4.2.6 Formato del informe generado.

El sistema debe permitir la generación de informes en los siguientes formatos, además de permitir la visualización del los mismos:

- Texto plano.
- PDF.
- HTML ó XML.
- Impresión en papel.

3.5 Requerimientos no funcionales

3.5.1 Usabilidad.

R-3.5.1.1 Interfaz gráfica.

El sistema realizado deberá ofrecer una interfaz de usuario gráfica con opciones de navegación a través de sus componentes mediante el uso del ratón y el teclado

R-3.5.1.2 Tiempo de familiarización con el sistema.

Lo deseable sería que un usuario con conocimientos básicos de informática pudiera explotar el sistema de acuerdo a sus necesidades después de un periodo de adiestramiento no superior a una jornada laboral.

3.5.2 Confiabilidad.

R-3.5.2.1 Tiempo de accesibilidad al sistema.

El sistema debe mantenerse en funcionamiento y por ende accesible durante todo el tiempo que el software SCADA se encuentre funcionando. Aunque debe estar disponible ante solicitudes esporádicas de sus servicios a pesar de no estar funcionando el SCADA para generar informes bajo demanda del usuario relacionados con los datos históricos captados por el mismo.

R-3.5.2.2 Forma de acceso al sistema.

El sistema debe permitir su acceso de forma independiente, es decir, este subsistema debe ser capaz de generar informes sin que para ello todos los demás subsistemas del SCADA se encuentren en ejecución, solo será imprescindible tener acceso a las fuentes de datos necesarias para obtener la información requerida en la generación del informe y algún mecanismo de autentificación del subsistema de seguridad que certifique el privilegio del usuario para acceder al mismo.

R-3.5.2.3 Tratamiento de las excepciones de seguridad.

El subsistema de generación de informes debe interceptar las excepciones generadas por el subsistema de seguridad que tengan relación con su funcionamiento e interactuar con el usuario informándole de estos eventos. Las excepciones de seguridad debido a intentos de

violación en niveles de privilegio para acceso a funcionalidades relacionadas con este subsistema son:

- Intento de visualización de un informe para el cual no se tiene privilegio de lectura.
- Intento de modificación de diseño de informe sin tener privilegios de escritura sobre el mismo.
- Intento de generación de un informe para el cual no se tiene privilegio.
- Tratamiento de las excepciones en la generación del informe.

De no poderse generar un informe en el momento solicitado, debido a las siguientes situaciones:

- No se encuentra el diseño de informe necesario para la generación del informe solicitado.
- No se tienen acceso a los datos necesarios para la generación del informe.
- No se puede entregar el informe al medio especificado.

Se deben emitir mensajes que impliquen la visualización de la situación al usuario del sistema y el registro como excepción del sistema.

R-3.5.2.4 Performance.

Se deben cumplir los requisitos de despeño generales planteados por el proyecto SCADA Nacional, en el documento de especificación de requisitos. (SCADA, EQUIPO PROYECTO 2006)

R-3.5.2.5 Tiempo de respuesta.

Ante peticiones de informe, el sistema debe responder de inmediato siempre que sea posible nunca atentando contra otros procesos más críticos en cuanto a tiempo de respuesta que coexistirán junto a este subsistema en el SCADA.

R-3.5.2.6 Concurrencia.

El sistema deberá soportar atención concurrente a la demanda de informes. Por ejemplo ante una demanda de informes de un usuario y un informe que se debe generar de forma periódica, o dos informes periódicos que coinciden en hora de generación.

R-3.5.2.7 Mantenibilidad.

El sistema debe ser susceptible a ampliaciones. Por tanto su diseño e implementación deberá realizarse atendiendo a que sea fácilmente mantenible, aplicando para su desarrollo las metodologías internacionalmente estandarizadas de tal manera que la introducción de nuevas funcionalidades no produzca fuertes impactos en el subsistema.

R-3.5.2.8 Mantenimiento independiente al funcionamiento del resto del sistema.

Este subsistema debe permitir su mantenimiento de forma independiente a los demás subsistemas del SCADA, de tal manera que no interfiera en el correcto desempeño del resto del sistema.

3.5.3 Portabilidad.

El sistema debe ser diseñado de manera portable, permitiendo su implementación en diversos lenguajes y plataformas.

Los informes generados deben ser multiplataforma, permitiendo a usuarios de diferentes entornos de trabajo GNU/Linux (Debian, Ubunto, Gentoo) hacer uso de los mismos.

3.5.4 Documentación de usuario y ayuda en línea.

R-3.5.4.1 Ayudas.

El sistema debe proporcionar una ayuda en línea que permita al usuario disponer de información útil para la explotación. Debe recoger conceptos, funcionalidades y mecanismos de interacción con las funcionalidades.

R-3.5.4.2 Manual de documentación del sistema.

Junto al sistema debe entregarse documentación orientada a la explotación del mismo destinada a usuarios pertenecientes a los siguientes perfiles:

- mantenedores del subsistema y diseñadores.
- Explotadores del subsistema.

3.5.5 Interfaces.

R-3.5.5.1 Interfaces de Usuario.

El sistema contará con varias interfaces gráficas de usuario orientadas a las distintas funcionalidades del sistema:

- Interfaz de usuario para la configuración de diseños de informe.
- Interfaz de usuario para la explotación.
- Interfaz de usuario para la gestión de informes.

Todas estas interfaces deben tener características comunes y manejar filosofías de trabajo similares de tal manera que permitan al usuario una rápida familiarización con las mismas. Por demás deben respetar las características generales que para las interfaces gráficas se definan para el SCADA.

R-3.5.5.2 Interfaces de Hardware.

Para lograr su acometido el sistema deberá mantener interfaces con los siguientes dispositivos de hardware.

- Discos duros.
- Impresoras.
- Modems.
- Tarjetas de red.

R-3.5.5.3 Interfaces de Software.

Al ser este un subsistema componente de un sistema SCADA interactuará estrechamente con otros softwares desarrollados en el marco del proyecto. De igual forma interactuará con otros recursos informáticos de amplia difusión entre la comunidad de usuarios de técnicas informáticas como bases de datos, editores y gestores de correo. Por lo cual debe hacer uso de las interfaces de estos subsistemas para acceder a sus servicios y además debe brindar una interfaz de software que permita a otros substistemas explotar sus funcionalidades.

R-3.5.5.4 Interfaces de Comunicaciones.

El subsistema basará sus comunicaciones en estándares de transmisión de información.
El subsistema debe ser capaz de comunicarse con bases de datos remotas alojadas en servidores de datos. Para lo cual debe emplear protocolos estándares de comunicación.
Para el envió de informes fuera del sistema se emplearán los siguientes servicios estándares y/o soportes de comunicaciones: e-mail, http, servicio de localizadores, telefonía móvil y telefonía fija.

3.5.6 Asociados al Licenciamiento.

R-3.5.6.1 Software libre.

El sistema será software libre y, por tanto, cualquier componente software que se utilice también deberá ser libre.

3.6 Diagrama de casos de uso del sistema.

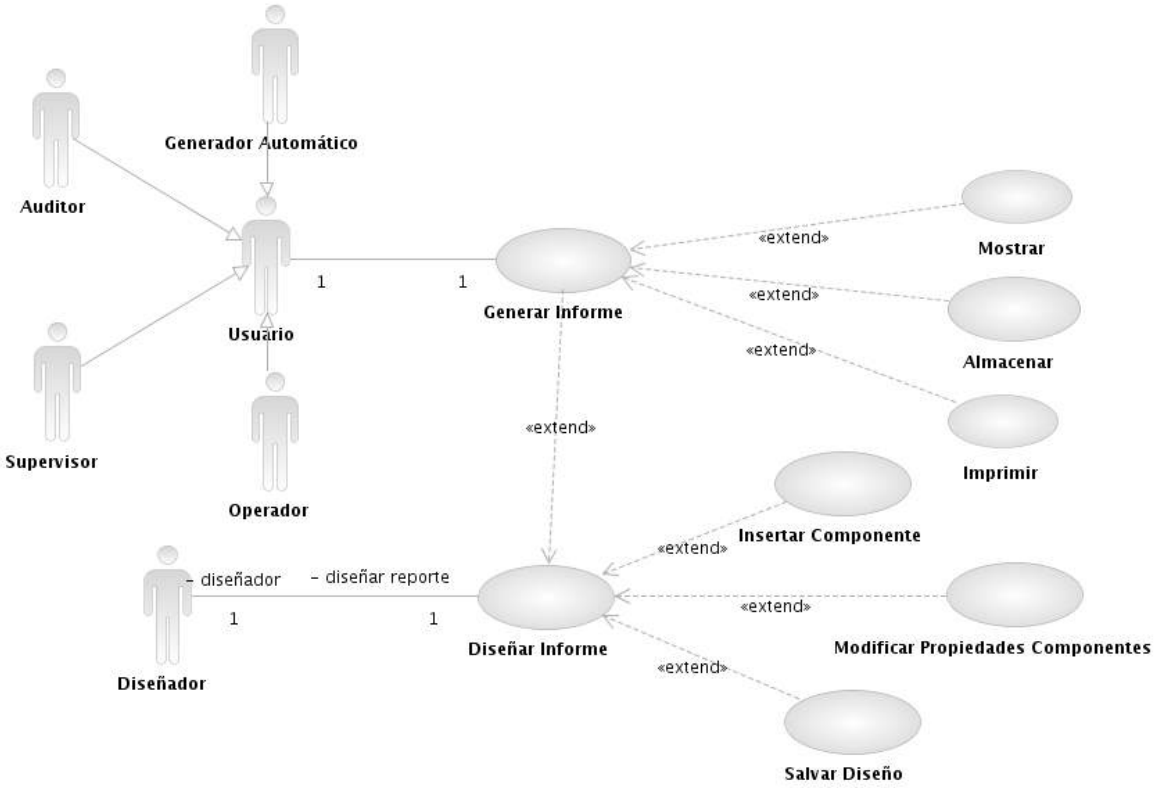

Fig. 5. Diagrama de casos de uso para el subsistema de generación de informes.

3.6.1 Caso de uso "Diseñar informe".

Nombre del Caso de Uso	Diseñar Informe
Actores	Diseñador (inicia)
Propósito	Este caso de uso le permite al Diseñador de informe crear, y abrir los diseños de informes.
Resumen	El Caso de Uso se inicia cuando el diseñador desea utilizar el diseño de un informe, escogiendo entre las opciones de crear un diseño de informe nuevo o recuperar un diseño de informe existente permitiendo crear un diseño a través de componentes gráficos que son salvados en formatos XML, finalizando el caso de uso.
Referencias	R-3.4.1.1
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ El diseñador es autentificado satisfactoriamente por el sistema. ➢ La aplicación debe dar la opción de crear un nuevo diseño de informe. ➢ La aplicación debe dar la opción de recuperar un diseño de informe.

Poscondiciones	➢ Se obtiene un diseño de informe en formato XMl salvado en el disco duro.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El diseñador abre la aplicación de diseño del informe.	1.1 El sistema muestra la interfaz principal de la aplicación.
2. El diseñador del sistema necesita Crear o Abrir un diseño de informe.	2.1 El sistema ejecuta alguna de las siguientes acciones: ➢ Si decide crear un nuevo diseño, ir a la sección “Crear nuevo diseño”. ➢ Si decide abrir un nuevo diseño, ir a la sección "Abrir diseño".
Sección “Crear nuevo diseño”	
1. El diseñador solicita crear un nuevo diseño.	1.1 El sistema le muestra una interfaz con las principales agrupaciones por tipo de informe (informes de alarmas, informes de producción, informes de sistema).
2. El diseñador selecciona uno de los grupos.	2.1 El sistema despliegan todos los tipos de informes pre elaborados en forma de plantilla que pertenecen a este agrupamiento.
3. El diseñador selecciona un tipo de diseño.	3.1 El sistema muestra una interfaz para completar los parámetros necesarios para el diseño seleccionado.
4. El diseñador completa todos los campos solicitados y presiona finalizar.	4.1 El sistema despliega en el área de trabajo el diseño correspondiente con los valores por defecto asignados a las propiedades de los componentes que contiene.
Sección “Abrir Diseño”.	

1. El diseñador selecciona “Abrir Diseño”.	1.1 El sistema muestra un interfaz para permitir entrar la ruta de localización del diseño, muestra aquellos diseños que se encuentra en la localización por defecto usada como almacén de los diseños.
2. El diseñador selecciona un diseño de los mostrados.	2.1 El sistema valida la existencia y estructura del documento XML. 2.2 El sistema carga y muestra el diseño
Puntos de Extensión	
1. El diseñador arrastra un componente al área de diseño.	1.1 Se ejecuta el caso de uso extendido “Insertar componente”.
2. El diseñador intenta modificar las propiedades de un componente.	2.1 Se ejecuta el caso de uso extendido “Modificar propiedades de componente”.
3. El diseñador selecciona salvar diseño con la opción salvar como.	3.1 Se ejecuta el caso de uso extendido “Salvar Diseño”.
Curso Alterno	
Línea 2.2	
I. Si el Diseñador selecciona abandonar el proceso de diseño cerrando la aplicación principal.	II. Si no se han hecho modificaciones en el diseño que se tiene abierto, se cierra el sistema.
	III. Si se han hecho modificaciones en el diseño que se tiene abierto, se ejecuta el caso de uso extendido “Salvar Diseño”.
Prioridad:	Crítico

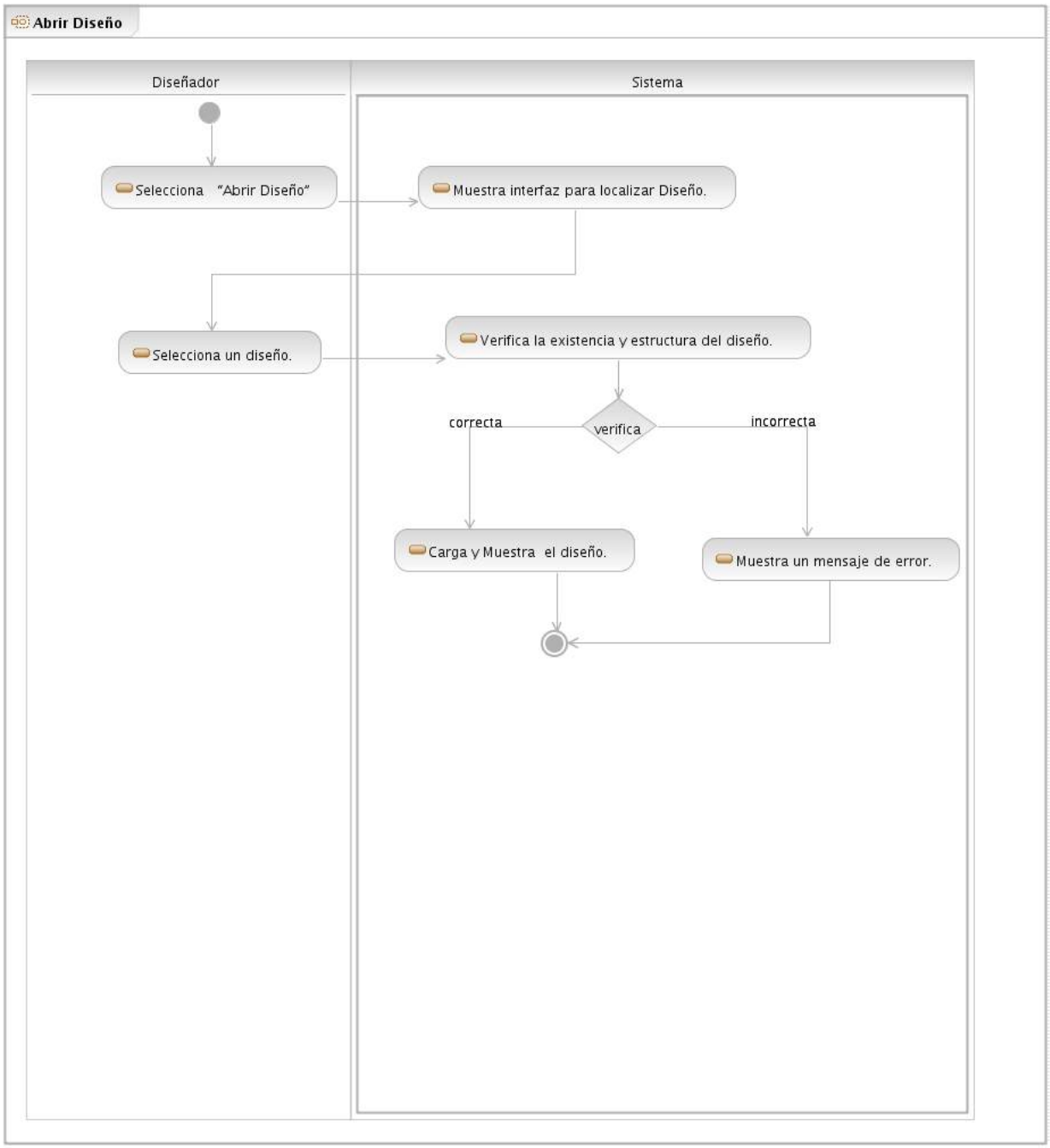

Fig. 6. Diagrama de actividad: Abrir Diseño.

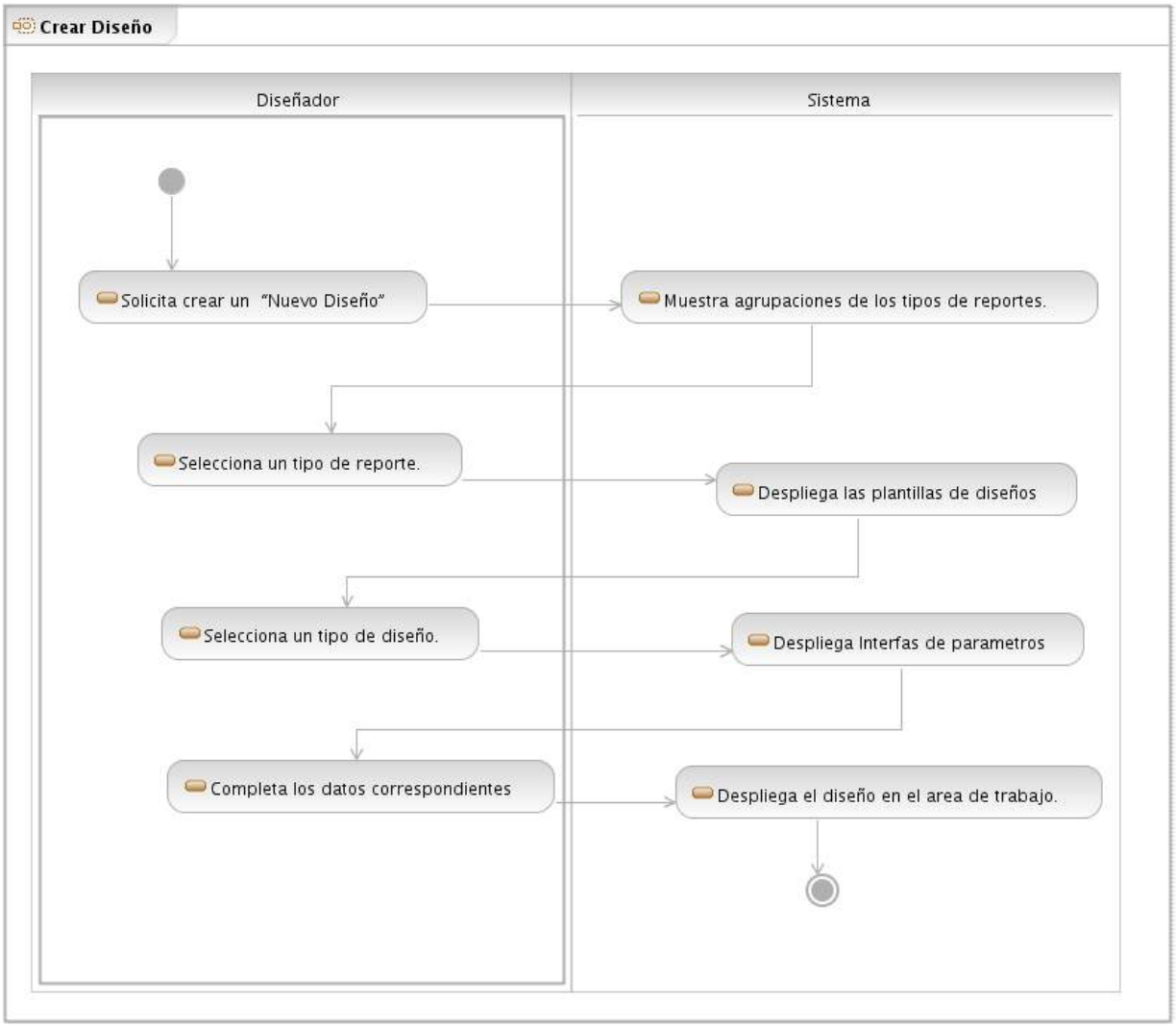

Fig. 7. Diagrama de actividad: Crear diseño.

3.6.2 Caso de uso extendido "Salvar diseño".

Nombre del Caso de Uso	Salvar Diseño
Actores	Diseñador (inicia)
Propósito	Este caso de uso le permite al Diseñador salvar un diseño.
Resumen	El Caso de Uso se inicia cuando el diseñador solicita salvar el diseño. El sistema solicita los datos del informe, finalizando el caso de uso.
Referencias	R-3.4.1.7, R-3.4.1.8
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe dar la opción de guardar el diseño del informe.

	➢ El usuario ha creado un diseño de informe.
Poscondiciones	➢ Se obtiene un diseño de informe en formato XML salvado en el disco duro.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El diseñador solicita salvar el diseño. 2. El diseñador introduce el nombre del informe. 3. El diseñador selecciona el destino de almacenamiento	1.1 El sistema muestra una interfaz que permite al diseñador entrar los datos de configuración del informe.
	3.1 El sistema salva el diseño en formato XML.
Prioridad:	Crítico

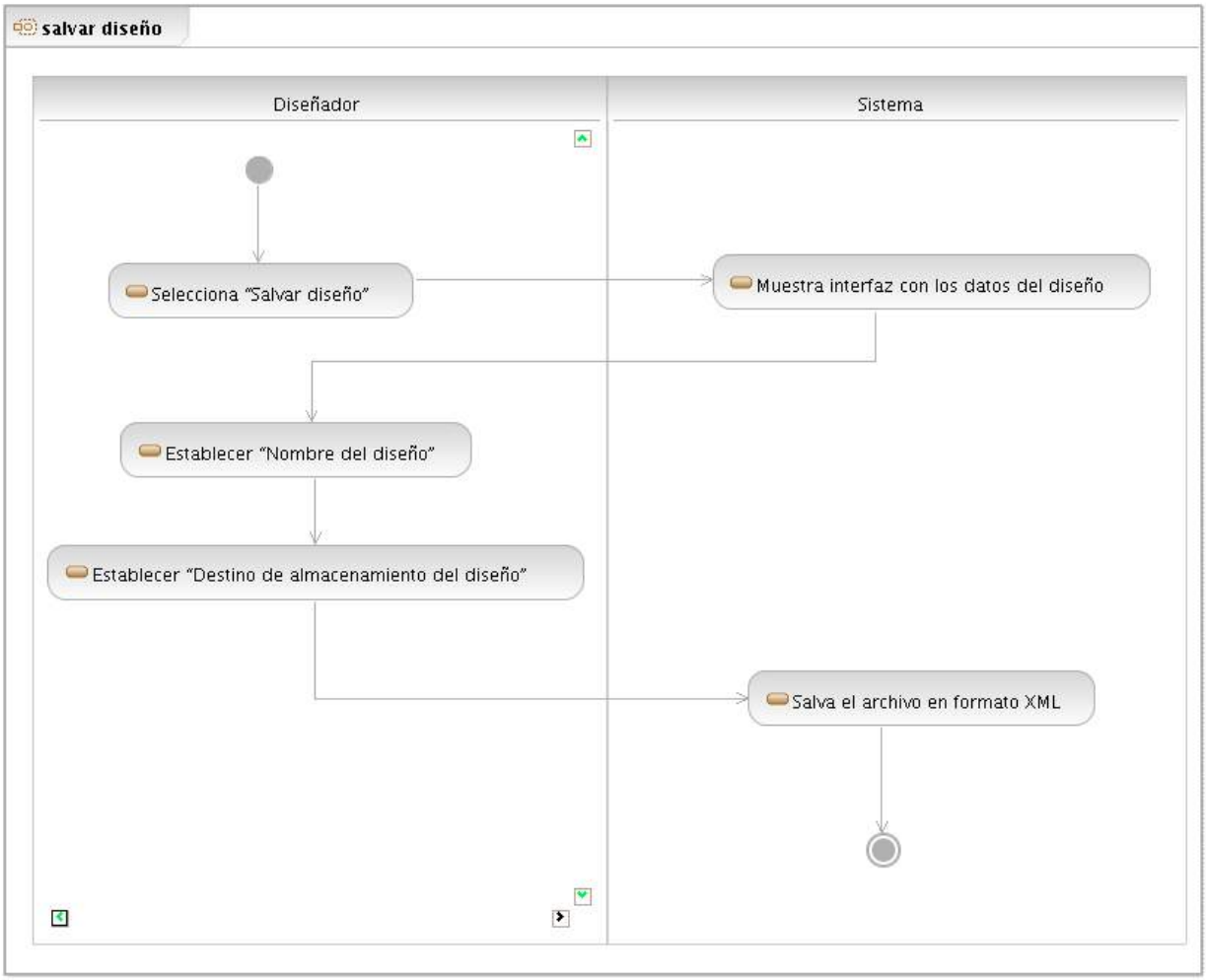

Fig. 8. Diagrama de actividad: Salvar diseño.

3.6.3 Caso de uso extendido “Insertar componente”.

Nombre del Caso de Uso	Insertar componente
Actores	Diseñador (inicia)
Propósito	Este caso de uso le permite al Diseñador añadir un componente gráfico.
Resumen	El Caso de Uso se inicia cuando el diseñador selecciona un componente de la paleta de componentes y lo trata de insertar en el diseño, finalizando así el caso de uso.
Referencias	R-3.4.1.5
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe dar la opción de añadir nuevos componentes gráficos y modificar las propiedades de los que ya existen. ➢ El usuario ha creado un diseño de informe.
Poscondiciones	➢ Se obtiene un diseño de informe en formato XMl con todos los componentes.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un componente de la paleta de componentes.	1.1 El sistema muestra el inspector de propiedades donde aparecen las propiedades por defecto del componente seleccionado.
2. El diseñador mueve el componente seleccionado.	2.1 El puntero del mouse cambia su apariencia a su paso por encima del área de diseño, tomando la apariencia de un marco rectangular con las dimensiones por defecto del componente seleccionado para las zonas donde es posible insertar el componente y otra apariencia para las áreas no permitidas que refleje esta situación.

3. El diseñador intenta ubicar el componente seleccionado.	3.1 El sistema valida la posible ubicación del componente dentro del área de diseño y de ser posible su ubicación en el área seleccionada.
	3.2 El sistema dibujará el componente en el punto de ubicación con las dimensiones por defecto y se establecerán todas las propiedades pertinentes del objeto.
Curso Alterno	
Línea 3.1	
	3.1 Si esta acción se intenta ejecutar en un área sobre la cual es imposible la ubicación del componente, este constituye una acción de liberación del componente seleccionado.
4 Si el diseñador libera el componente seleccionado presiona ESC, elige otra opción u otro componente.	4.1 El puntero del mouse vuelve a tomar la apariencia normal.
	4.2 El sistema mostrará las propiedades del anterior componente insertado o en caso de que este sea el primer componente insertado, se mostrarán las propiedades del área de diseño.
Prioridad:	Crítico

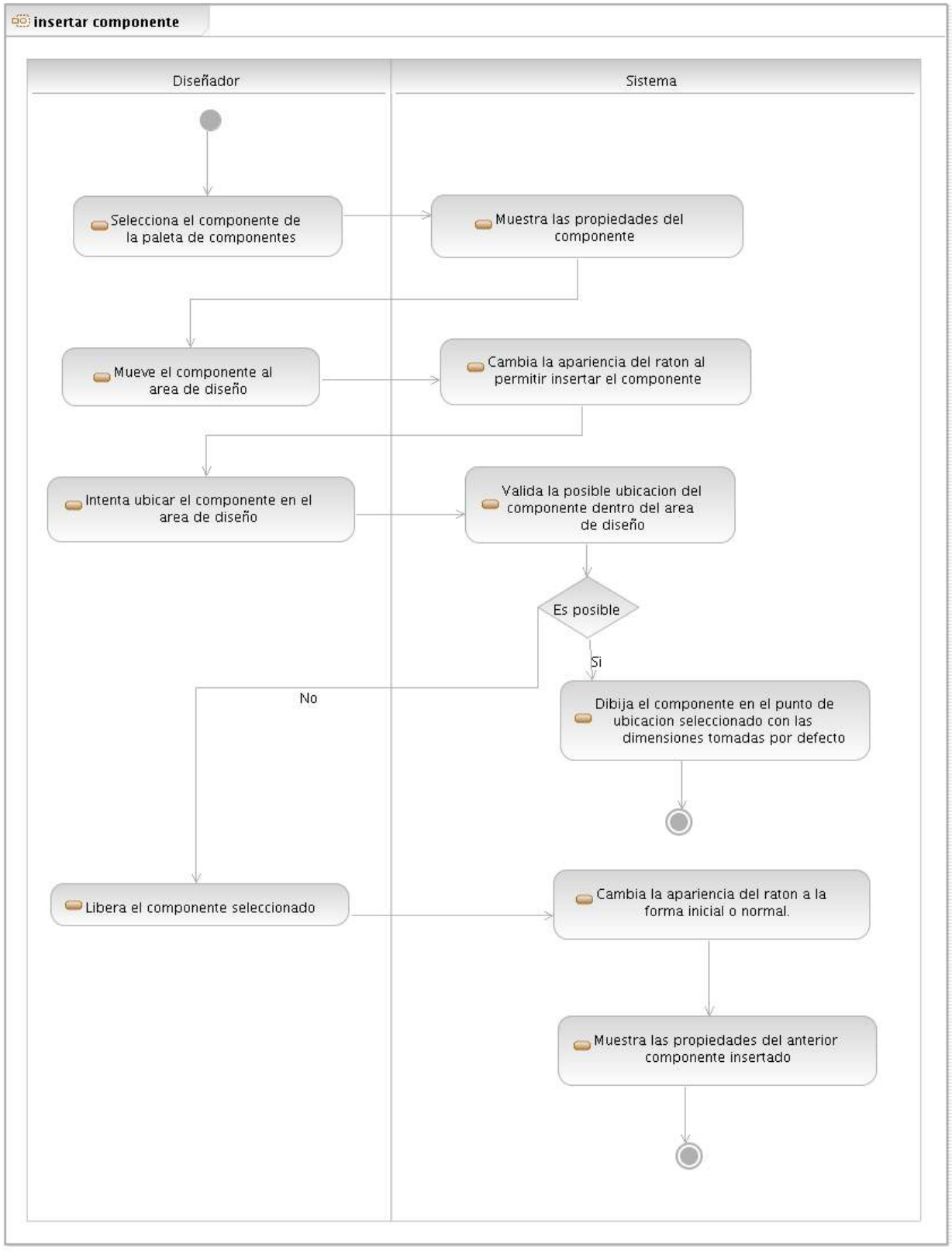

Fig. 9. Diagrama de actividades: Insertar componente.

3.6.4 Caso de uso extendido "Modificar propiedades de componente".

Nombre del Caso de Uso	Modificar Propiedades de componente
Actores	Diseñador (inicia)
Propósito	Este caso de uso le permite al Diseñador modificar las propiedades de un componente al ser colocado en el diseño del informe.
Resumen	El Caso de Uso se inicia cuando el diseñador selecciona un componente del diseño del informe y le modifica sus propiedades, finalizando así el caso de uso.
Referencias	R-3.4.1.2, R-3.4.1.3, R-3.4.1.4
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe dar la opción de modificar las propiedades del componente que forma el diseño del informe. ➢ El usuario ha creado un diseño de componente.
Poscondiciones	➢ Se obtiene un diseño de informe en formato XMl con las propiedades de los componentes modificas.

Curso Normal de los Eventos

Acciones del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un componente.	1.1 El sistema muestra en el inspector de propiedades las propiedades del componente seleccionado.
2. El diseñador selecciona en el inspector de propiedades la propiedad a modificar del componente.	2.1 El sistema activa el cuadro de edición de la propiedad seleccionada para la introducción del nuevo valor de la propiedad.
3. El diseñador modifica la propiedad de un componente.	3.1 El sistema mostrará para aquellas propiedades a las cuales solo son asignables un conjunto de valores, como colores, tamaños, tipos de letras, una lista desplegable con las posibles opciones a seleccionar. 3.2 El sistema valida los valores introducidos en el inspector de propiedades asignado los nuevos valores al componente.

Curso Alterno	
Línea 3.2	
	3.2 En caso de que el valor introducido no sea correcto se mantendrá asignado el valor anterior.
4. El diseñador decide abandonar el proceso de modificación de la propiedad (ESC).	4.1 El sistema volverá a asignar a la propiedad en cuestión el valor que poseía antes de comenzar la edición de la misma.
Prioridad:	Crítico

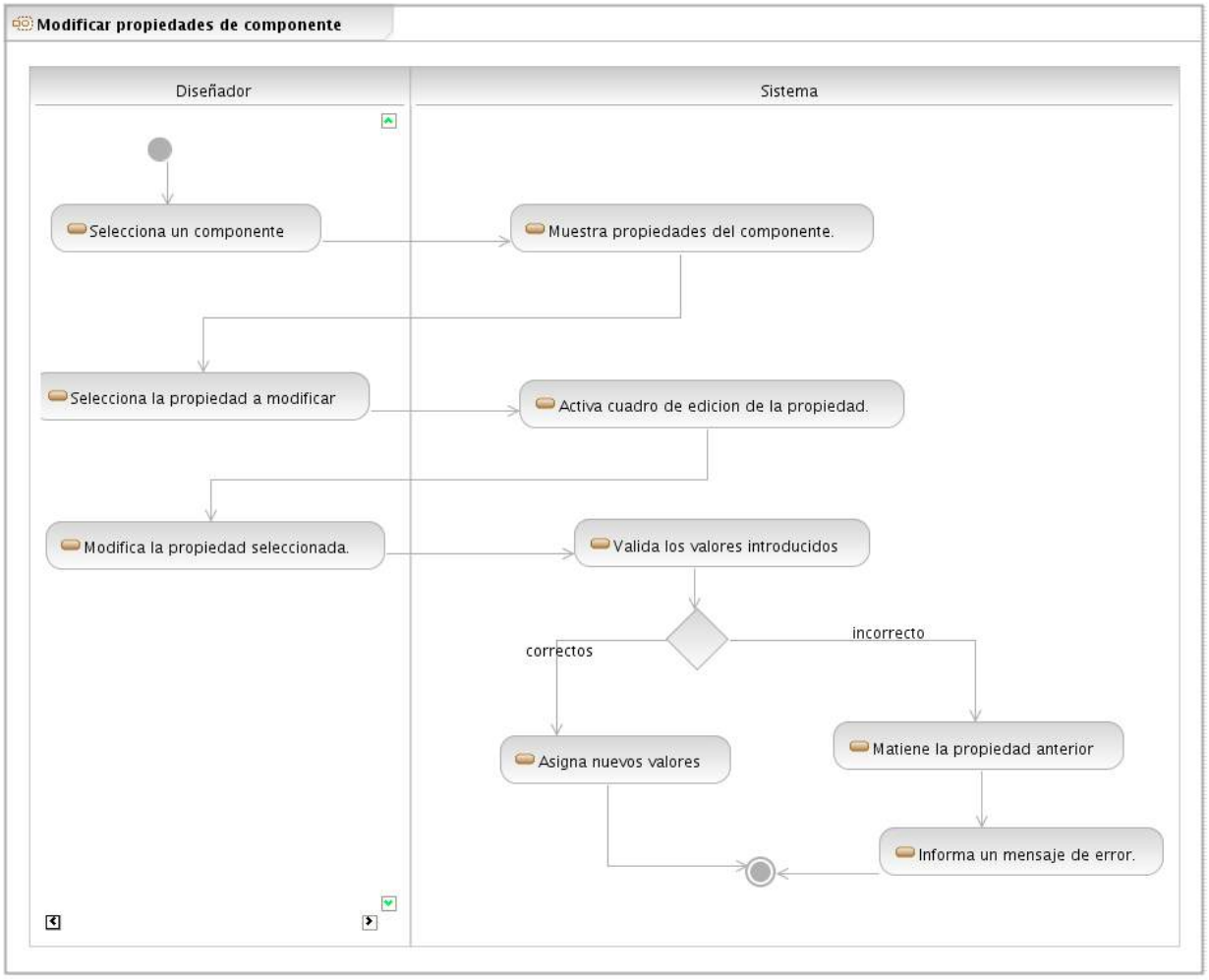

Fig. 10. Diagrama actividad: Modificar Propiedades.

3.6.5 Caso de uso "Generar informe".

Nombre del Caso de Uso	Generar informe
Actores	Usuario (inicia)
Propósito	Este caso de uso le permite al usuario generar un informe con diferentes salidas.
Resumen	El Caso de Uso se inicia cuando el usuario solicita generar un informe y el sistema le brinda las diferentes opciones a elegir, finalizando el caso de uso.
Referencias	R-3.4.1.6, R-3.4.2.1, R-3.4.2.2, R-3.4.2.5
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ El usuario debe estar autentificado. ➢ La aplicación debe dar la opción de generar un informe. ➢ El usuario ha creado un diseño de informe.
Poscondiciones	➢ Se obtiene un informe final en el formato solicitado.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario solicita generar un informe.	1.1 El sistema despliega una interfaz para seleccionar el diseño XML a generar.
2. El usuario solicita despachar el informe.	2.1 El sistema muestra todas las fuentes de despacho posibles.
3. El usuario selecciona el despacho deseado.	3.1 El sistema da salida a la información solicitada.
Puntos de Extensión	
1. El usuario selecciona la opción de Imprimir.	1.1 Se ejecuta el caso de uso extendido "Imprimir Informe".
2. El usuario selecciona la opción de mostrar.	2.1 Se ejecuta el caso de uso extendido "Mostrar Informe".
3. El usuario selecciona la opción de almacenar.	3.1 Se ejecuta el caso de uso extendido "Almacenar Informe".
Prioridad:	Crítico

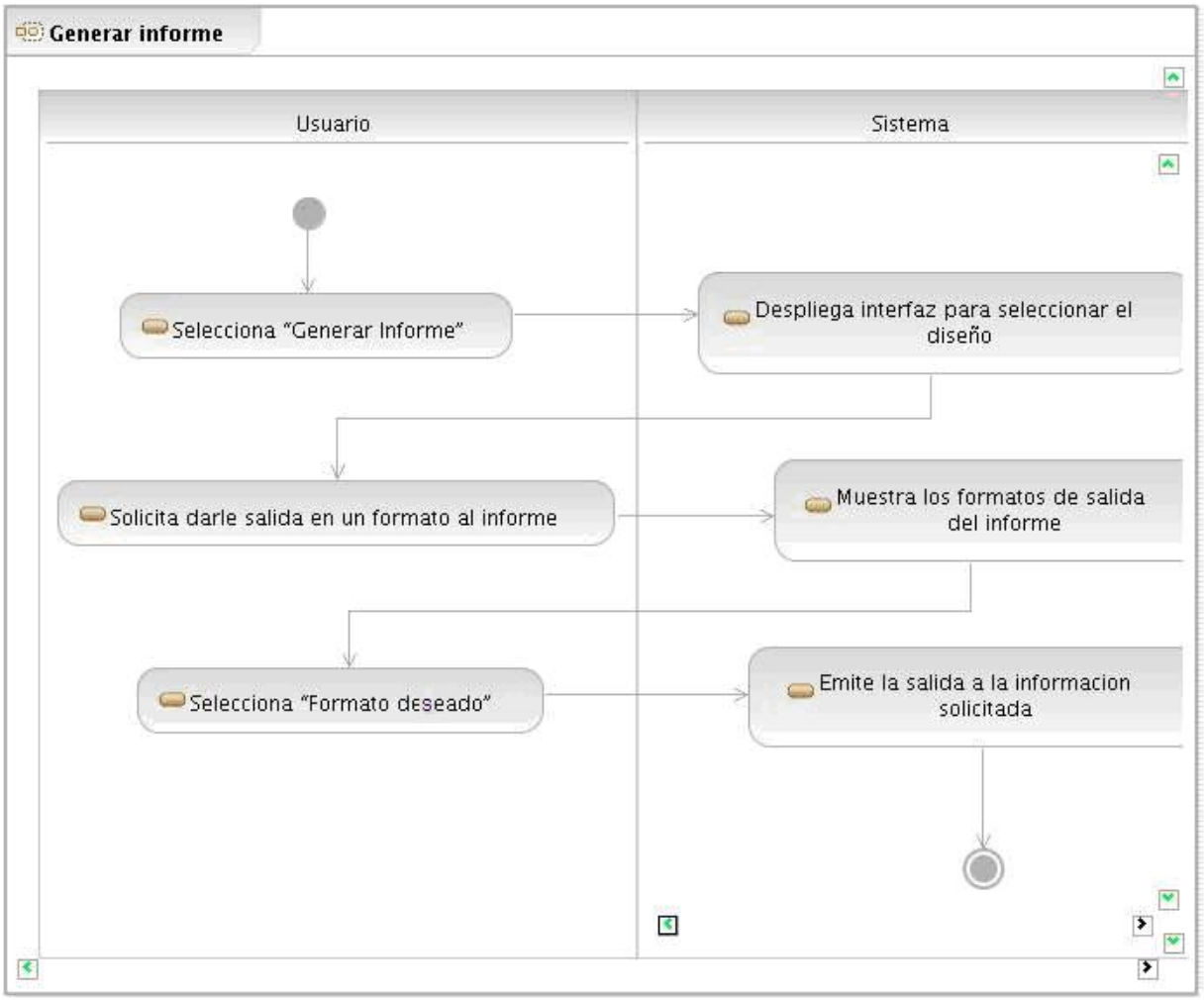

Fig. 11. Diagrama actividad: Generar informe.

3.6.6 Caso de uso extendido "Mostrar informe".

Nombre del Caso de Uso	Mostrar Informe
Actores	usuario (inicia)
Propósito	Este caso de uso permite mostrar un informe al usuario.
Resumen	El Caso de Uso se inicia cuando el usuario solicita mostrar un informe y el sistema le da respuesta, finalizando así el caso de uso.
Referencias	R-3.4.2.6
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe mostrar el informe solicitado.
Poscondiciones	➢ Se obtiene el informe final solicitado.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**

1. El usuario selecciona Mostrar informe.	1.1 El sistema muestra el informe solicitado por el usuario.
Prioridad:	Crítico

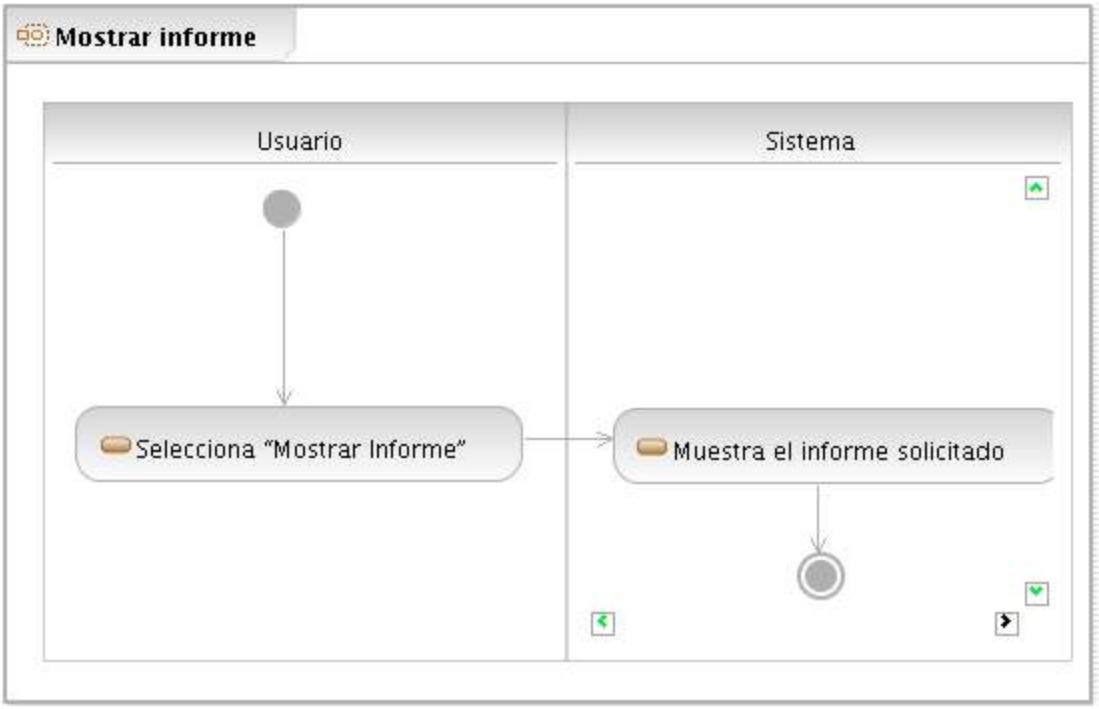

Fig. 12. Diagrama de actividades: Mostrar informe.

3.6.7 Caso de uso extendido "Almacenar informe".

Nombre del Caso de Uso	Almacenar Informe
Actores	usuario (inicia)
Propósito	Este caso de uso permite almacenar un informe en el formato solicitado.
Resumen	El Caso de Uso se inicia cuando el usuario solicita almacenar un informe y el sistema le solicita los datos del informe, finalizando así el caso de uso.
Referencias	R-3.4.1.9, R-3.4.2.3, R-3.4.2.4, R-3.4.2.6
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe almacenar el informe generado.
Poscondiciones	➢ Se almacena el informe en el disco duro.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario selecciona almacenar un	1.1 El sistema muestra una interfaz

informe.	para especificar el nombre del informe.
2. El usuario introduce el nombre del informe. 3. El usuario selecciona el formato del informe.	3.1 El sistema salva el informe en el formato especificado.
Prioridad:	Crítico

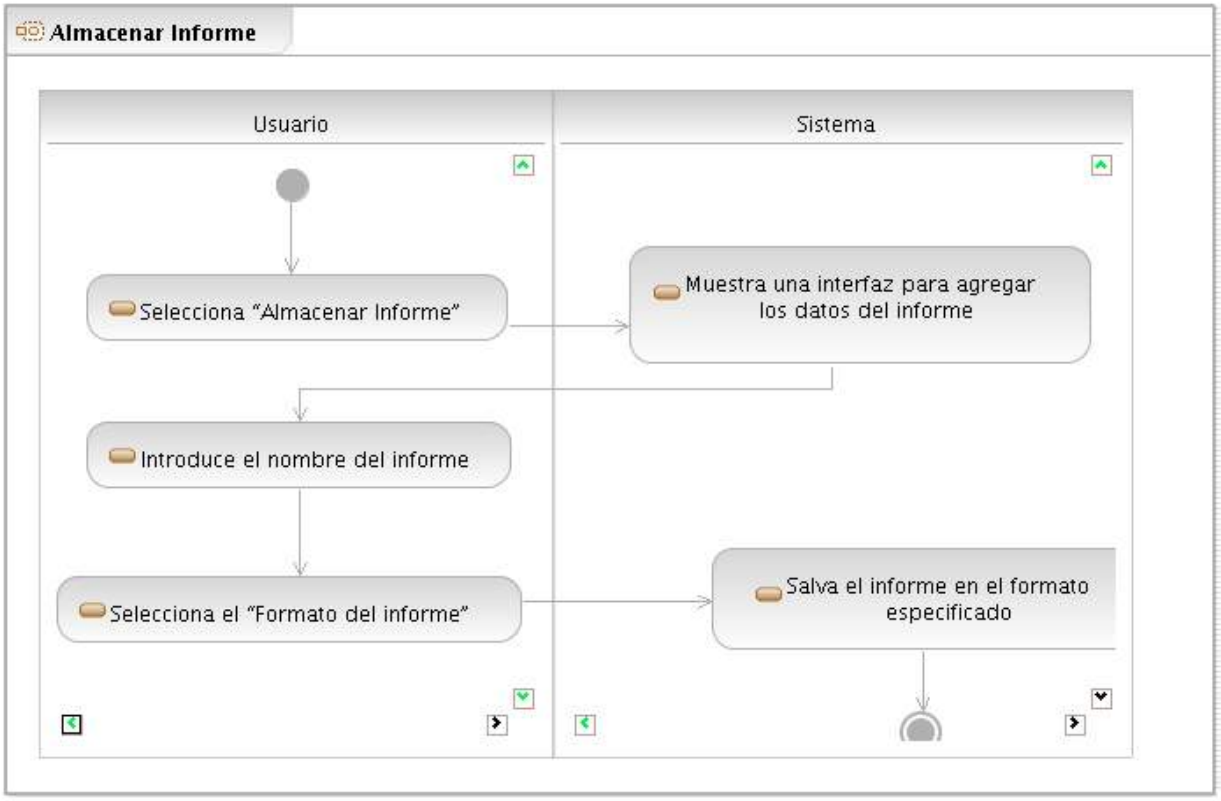

Fig. 13. Diagrama de actividades: Almacenar informe.

3.6.8 Caso de uso extendido "Imprimir informe".

Nombre del Caso de Uso	Imprimir Informe
Actores	usuario (inicia)
Propósito	Este caso de uso permite imprimir un informe.
Resumen	El Caso de Uso se inicia cuando el usuario necesita imprimir un informe con una impresora determinada, finalizando así el caso de uso.
Referencias	R-3.4.2.6
Precondiciones	➢ La aplicación debe funcionar correctamente. ➢ La aplicación debe permitir imprimir un informe.

Poscondiciones	➢ Se imprime el informe con la impresora adaptada al hardware.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario necesita imprimir un informe.	1.1 El sistema muestra una ventana para la configuración y selección de las impresoras declaradas en el sistema operativo.
2. El usuario configura la impresión.	2.1 El sistema imprime el informe haciendo uso de la impresora seleccionada.
Prioridad:	Crítico

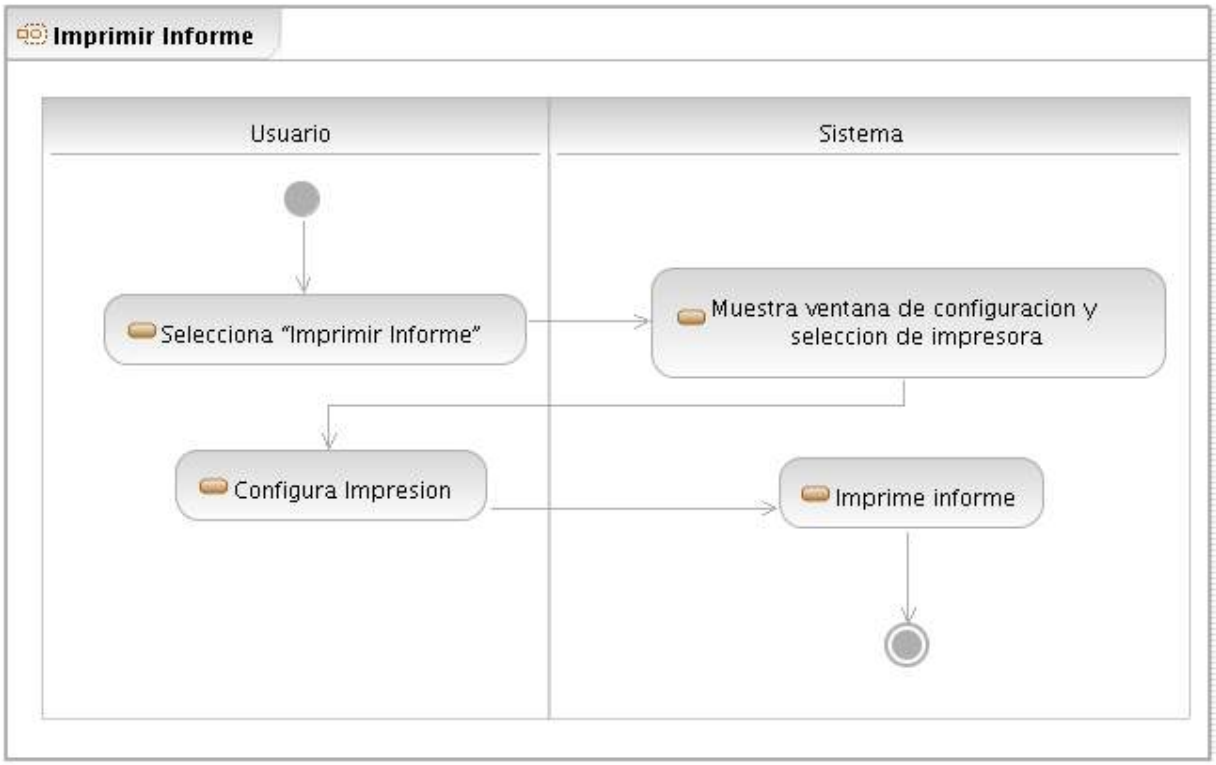

Fig. 14. Diagrama de actividades: Imprimir Informe.

3.7 Conclusiones.

En este capítulo se comenzó a profundizar en la propuesta de solución, obteniéndose las funcionalidades que debe tener el sistema, las cuales fueron representadas mediante un diagrama de casos de uso. Además se describieron todas las acciones que realizan los actores y las respuestas del sistema. Partiendo de los requisitos presentados en este capitulo se modelará el sistema.

CAPÍTULO 4

4 CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción.

En el presente capítulo se diseña la propuesta de solución, primeramente se incluye una pequeña sección de funcionamiento donde se retoman algunos puntos para dar una mejor visión de las solución propuesta. Posteriormente se presentan los diagramas de clases agrupados por las distintas capas de la arquitectura. Además se elaboran los diagramas de secuencia para los principales escenarios de los casos de uso. Obteniéndose finalmente los prototipos de interfaz de usuario del sistema.

4.2 Funcionamiento.

El funcionamiento del sistema a desarrollar puede dividirse en dos:

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

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

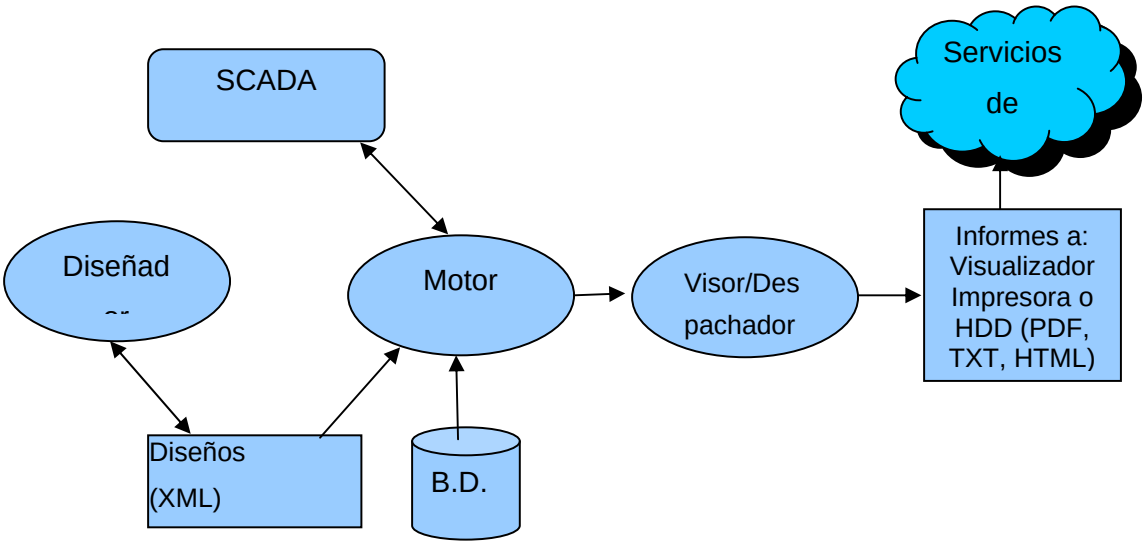

Fig. 15. Esquema funcional del sistema propuesto.

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

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

4.3 Modelo de diseño.

En la fase de diseño se modela el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales, creándose así una entrada apropiada para las actividades de implementación.

4.4 Patrones.

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

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

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

Dada la similitud existente entre el subsistema que se necesita y una aplicación de gestión se propone una arquitectura en capas, la cual da una gran flexibilidad a la aplicación dado el bajo

acoplamiento funcional que se logra con la misma, permitiendo a la aplicación adaptarse de forma relativamente fácil a cambios (adiciones, modificaciones o eliminaciones).

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

4.5 Diagrama de arquitectura para la aplicación.

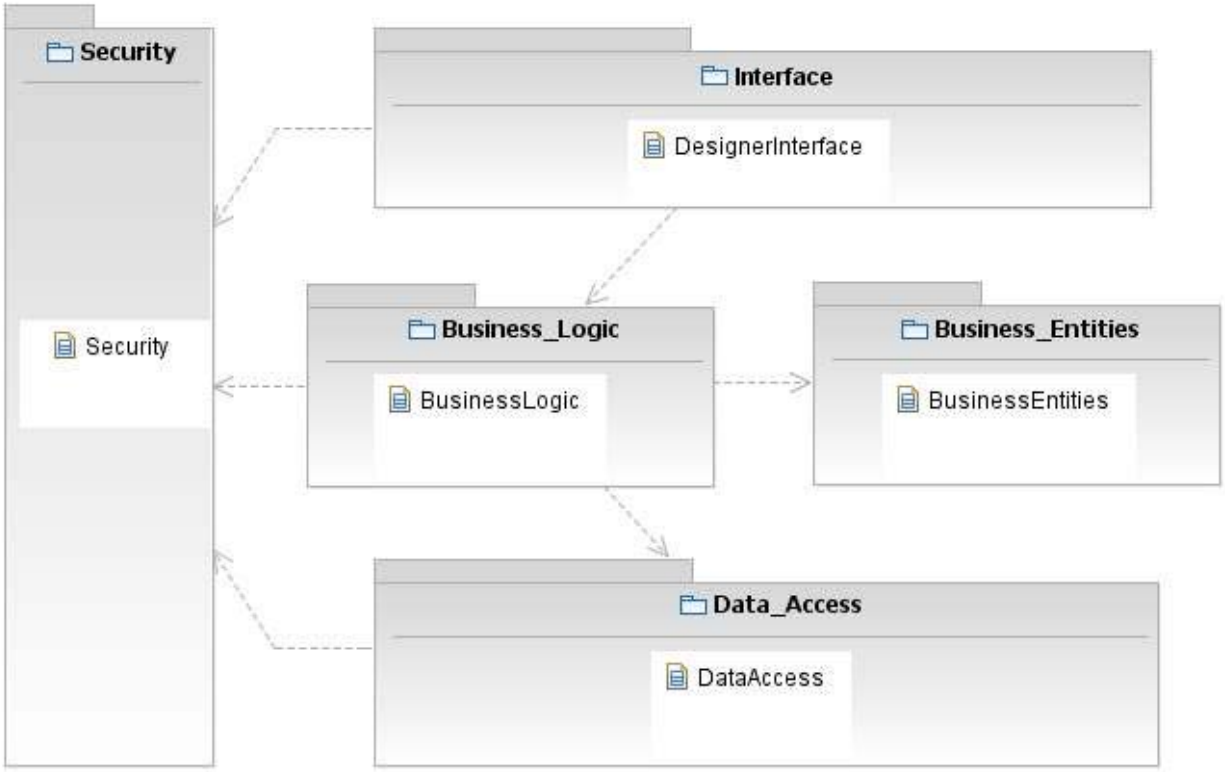

Fig. 16: Arquitectura propuesta para el subsistema de generación de informes.

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

4.6 Diagramas de clases agrupados por las distintas capas de la arquitectura propuesta.

Las entidades se usaran por ambos módulos del subsistema propuesto, es decir por el diseñador y por el motor. Dichas entidades solo variaran su comportamiento en un módulo y en otro, pues para el diseñador es necesario que cada uno de los objetos que se van insertando sean totalmente manipulables por el diseñador, mientras que en el momento de generación estas entidades solo se desplegaran y pintaran sobre las paginas necesarias para tal fin, sin necesidad de ser manipuladas una vez obtenido el informe final. Además de este punto, existe

4.6.1 Entidades del negocio.

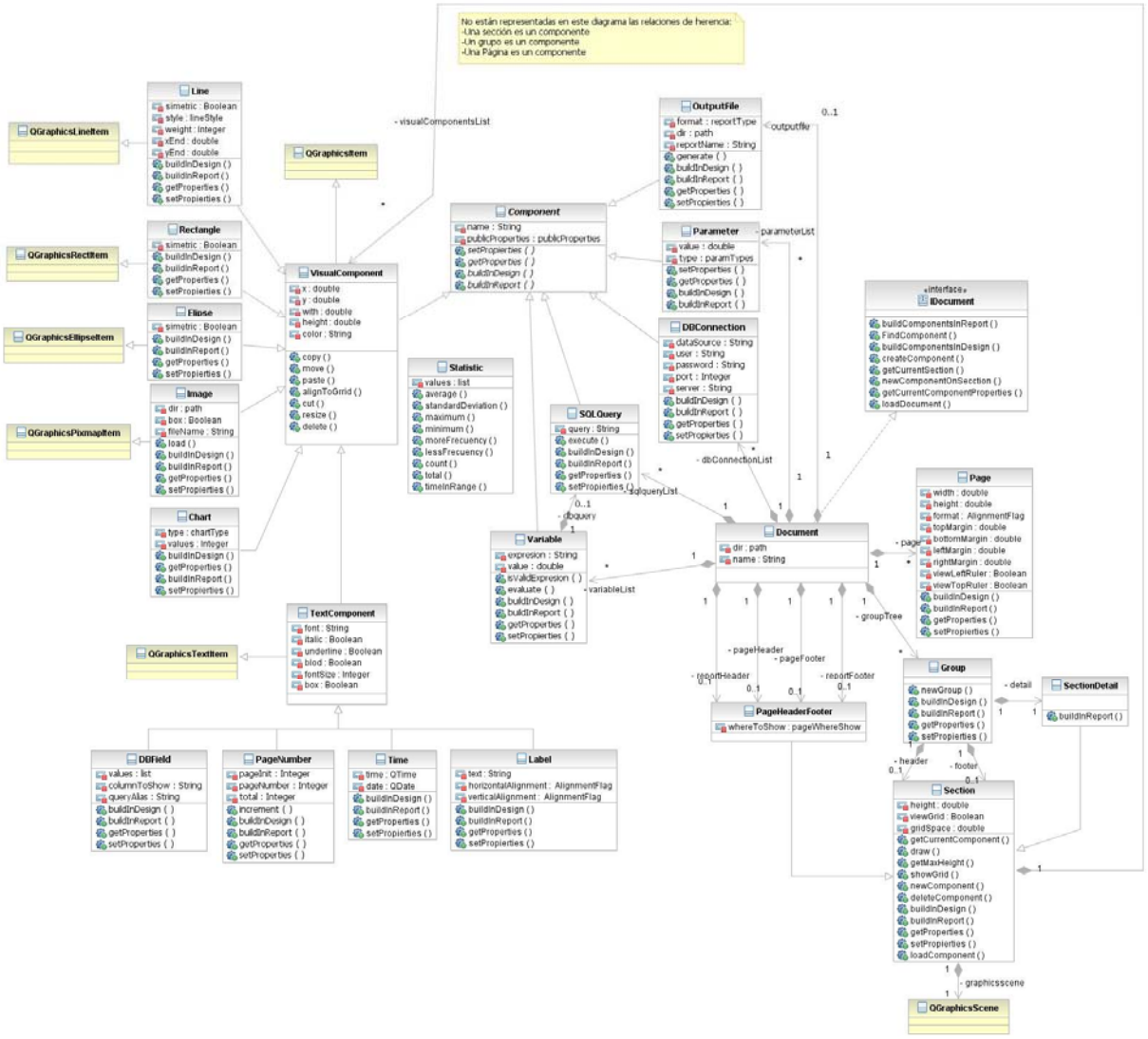

Fig. 17: Diagrama de clases representativo de la capa de entidades del negocio.

otra diferencia en el comportamiento de algunas entidades en el momento de diseño y de generación, en este caso, relacionadas con su apariencia visual. Por ejemplo: una entidad de la clase *"dbField"* en el momento de diseño tendría una apariencia muy similar a una *"label"* pues solo contendría la referencia al campo que resultante de la ejecución de una sentencia SQL se desplegaría en forma de columna en el momento de generación, es decir en el momento de diseño ocupa una sola posición, mientras que el momento de generación ocupa tantas posiciones verticales como datos arroje la consulta. Similarmente ocurre con las instancias de tipo *"chart"* (gráfico de pastel, barras, etc.) las cuales en momento de diseño no pueden mostrar el gráfico final, pues no se cuentan con los datos hasta tanto no se genera el informe. A pesar de esta dicotomía en el comportamiento de las entidades se pude lograr la utilización de las mismas clases en el diseñador y en el generador haciendo uso de las mismas propiedades,

que son las que finalmente se hacen persistir en el fichero XML y representan un determinado diseño. De esta manera se logra el uso de una sola estructura XML.

4.6.2 Lógica de negocios.

El uso de las mismas entidades, tanto para el diseñador como para el generador, hacen posible el uso de una única clase *"XMLParser"* para ambas, la cual está encargada de cargar un fichero XML correspondiente a un diseño y crear los objetos en memoria para que después, haciendo uso de los métodos adecuados, lo mismo las instancias de tipo *"Designer"* que de tipo *"Engine"* puedan construirlos para el diseñador o el generador respectivamente con la apariencia y el comportamiento adecuado.

Para lograr la comunicación con el SCADA se ha incorporado una clase abstracta (*"EngineInterface")* la cual proporciona métodos para la comunicación en ambos sentidos con el SCADA, es decir, vía esta interfaz el SCADA indistintamente puede solicitar de un diseño aquella información que requiere completar para la correcta generación del informe correspondiente (los parámetros establecidos en el momento de diseño), que solicitar la generación de un determinado informe. En ambos casos, el SCADA nunca tendrá acceso directo a ninguna de las clases propias del generador de informes.

La clase *"command"* con sus respectivos descendientes se ha incluido para permitir el control de la secuencia de comandos que el diseñador ejecuta en el momento de diseño, permitiendo las acciones de recuperación o vuelta atrás en los últimos comandos introducidos.

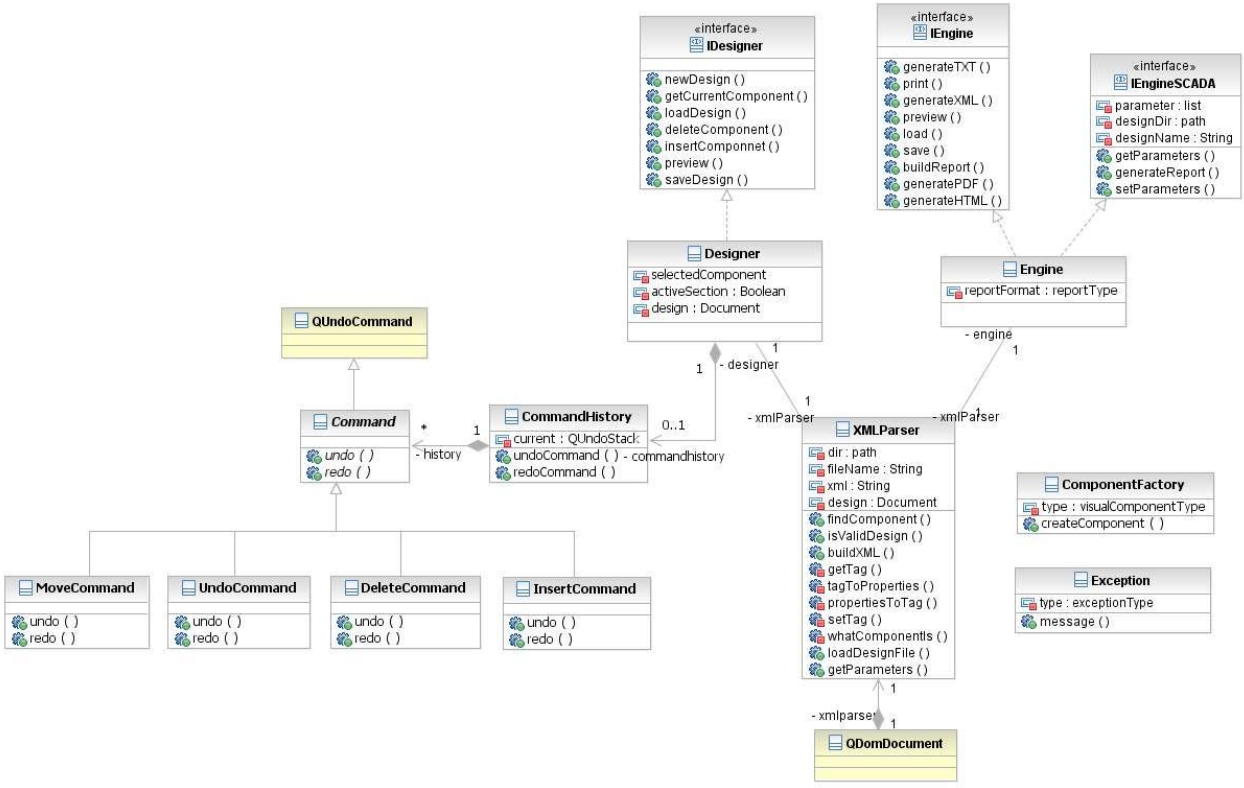

Fig. 18: Diagrama de clases representativo de la capa de lógica del negocio.

4.6.3 Capa de Interfaz.

Esta capa abarca todas las interfaces de comunicación con el usuario. Debido a la dependencia con los prototipos de interfaz propuestos para la aplicación, se limitara a mostrar solamente el diagrama y no describir nada relacionado con el mismo.

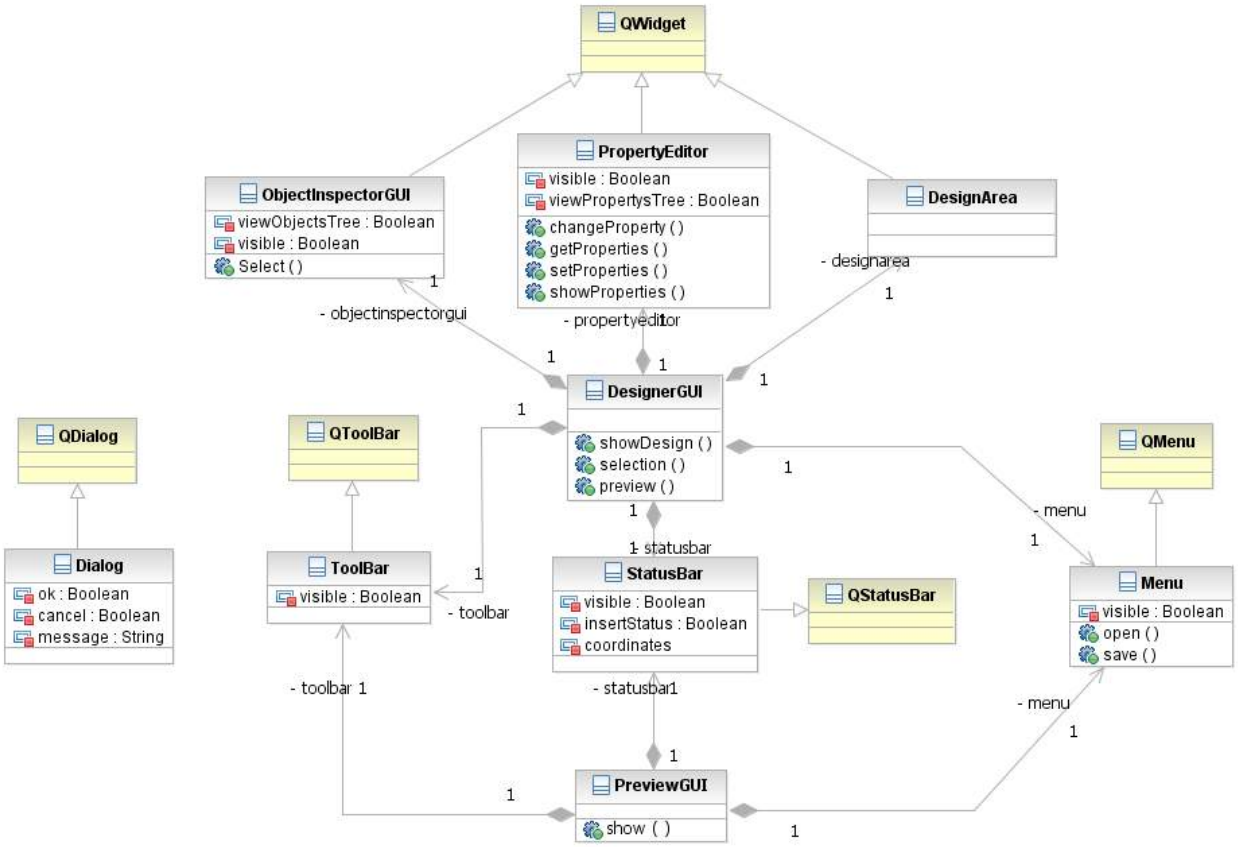

Fig. 19: Diagrama de clases representativo de la capa de interfaz.

4.6.4 Capa de acceso a datos.

En esta capa aparecen aquellas clases que implementan los accesos concretos a los distintos medios usados para el almacenamiento de datos. De forma general se consideran medios de almacenamiento a: las bases de datos, que se gestionaran a través de instancias de las clases *"DBConnection"* y *"DBSQLQuery"* y los archivos que persisten en disco, gestionándose estos últimos a través de las instancias de las clases *"File"* y *"Printer"*.

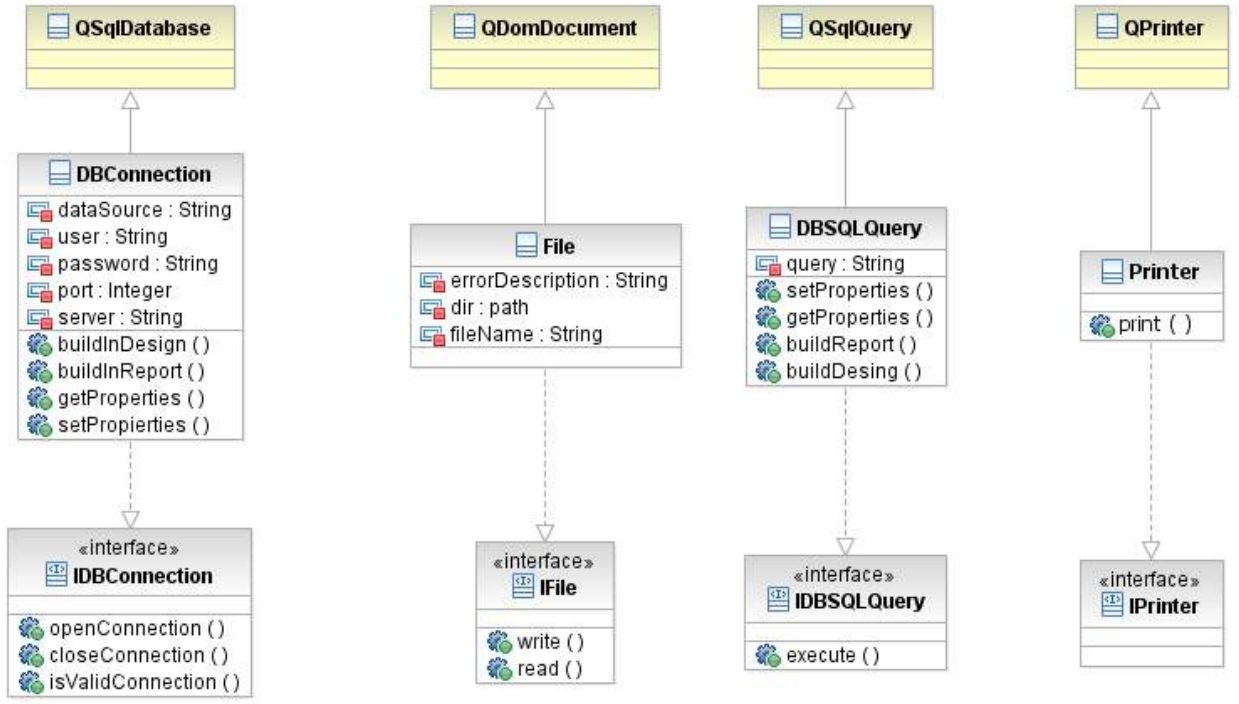

Fig. 20: Diagrama de clases representativo de la capa de acceso a datos.

4.7 Diagramas de secuencia para los principales escenarios de los casos de uso.

A continuación se presentan los diagramas de secuencia que describen los principales escenarios de los casos de uso del sistema, además se muestra la documentación de dichos diagramas de secuencia para su mejor comprensión.

4.7.1 Cargar diseño.

El diseñador decide abrir un diseño a través de la opción abrir de la barra de menú o del botón abrir de la barra de herramientas, esto provoca una llamada al método *"loadDesign"* de la clase *"Designer"*, la cual toma el control y haciendo uso de los métodos de la clase *"XMLParser"* primeramente carga el documento desde el archivo almacenado en disco, construyendo en memoria un árbol de objetos. Luego, siguiendo una lógica de recorrido de la estructura de dependencia de componentes, controlada la clase *"Designer"* se van obteniendo las distintas propiedades de los objetos y construyendo los mismos a través de la factoría implementada para tal fin, por último se le va diciendo a cada una de las instancias creadas que se construyan sobre el diseño. Cuando se termina de ejecutar este lazo, ya se tiene cargado todo el diseño, solo queda mostrarlo, para lo cual se invoca el método "showDesign" de la interfaz gráfica de usuario.

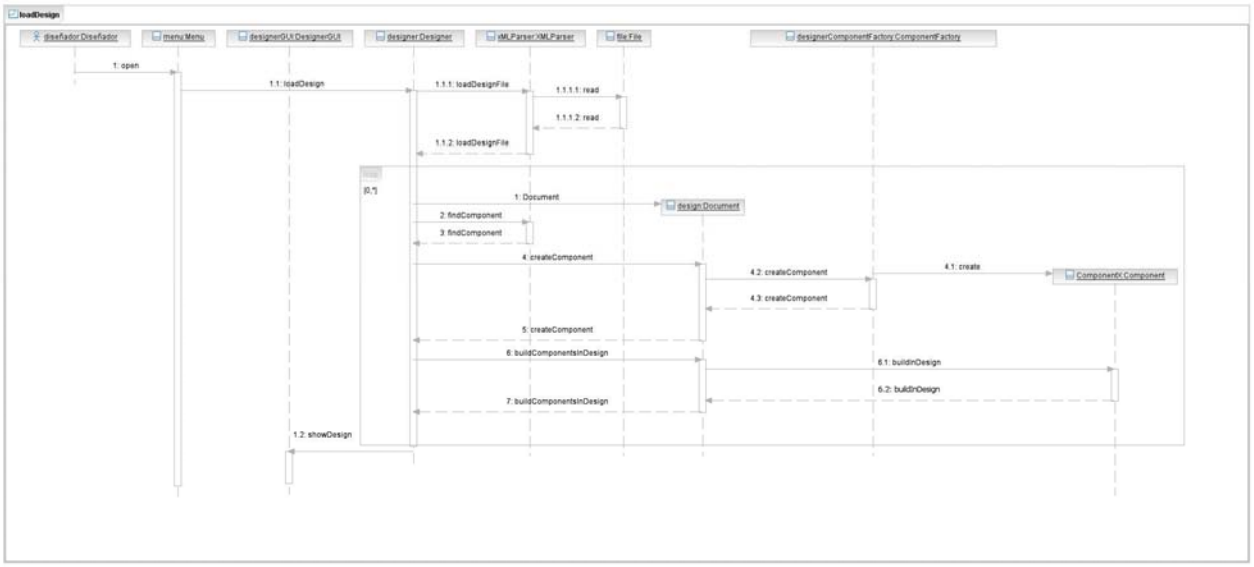

Fig. 21 Diagrama de secuencia que representa la cargar de un diseño.

4.7.2 Salvar un diseño.

El diseñador solicita la opción salvar a través de de la barra de menú o del componente correspondiente de la barra de herramientas, esto desencadena una llamada al método *"saveDesign"* de la clase *"Designer"* la cual toma el control de todo el proceso de salva. Para ello se recorre toda la estructura de dependencia de componentes en postorden, es decir, se comienza recorriendo las listas de componentes de las secciones, luego se sube a las listas de secciones contenidas en los grupos, posteriormente a la lista de grupos contenida en el documento y por último a los componentes de control como (SQLQuery, DBConnection, etc). A cada uno de los componentes que va arrojando el algoritmo de recorrido se le van pidiendo sus propiedades y a través del método *"propertiesToTag"* de la clase *"XMLParser"* se van convirtiendo dichas propiedades a una cadena que finalmente tendrá la estructura del archivo XML que se salva en disco.

Estructura aproximada del XML de un diseño

Document-|
|-ReporHeader
|-PageHeader
|-Parameter
|-DBConection
|-SQLQuery
|-Variable
|-Group - |
|-SectionHeader-|
|-Component-1

|-Component-n

|-SectioDetail-|

|-Component-1

|-Component-n

|-SectionFooter-|

|-Component-1

|-Component-n

|-ReporFooter

|-PageFooter

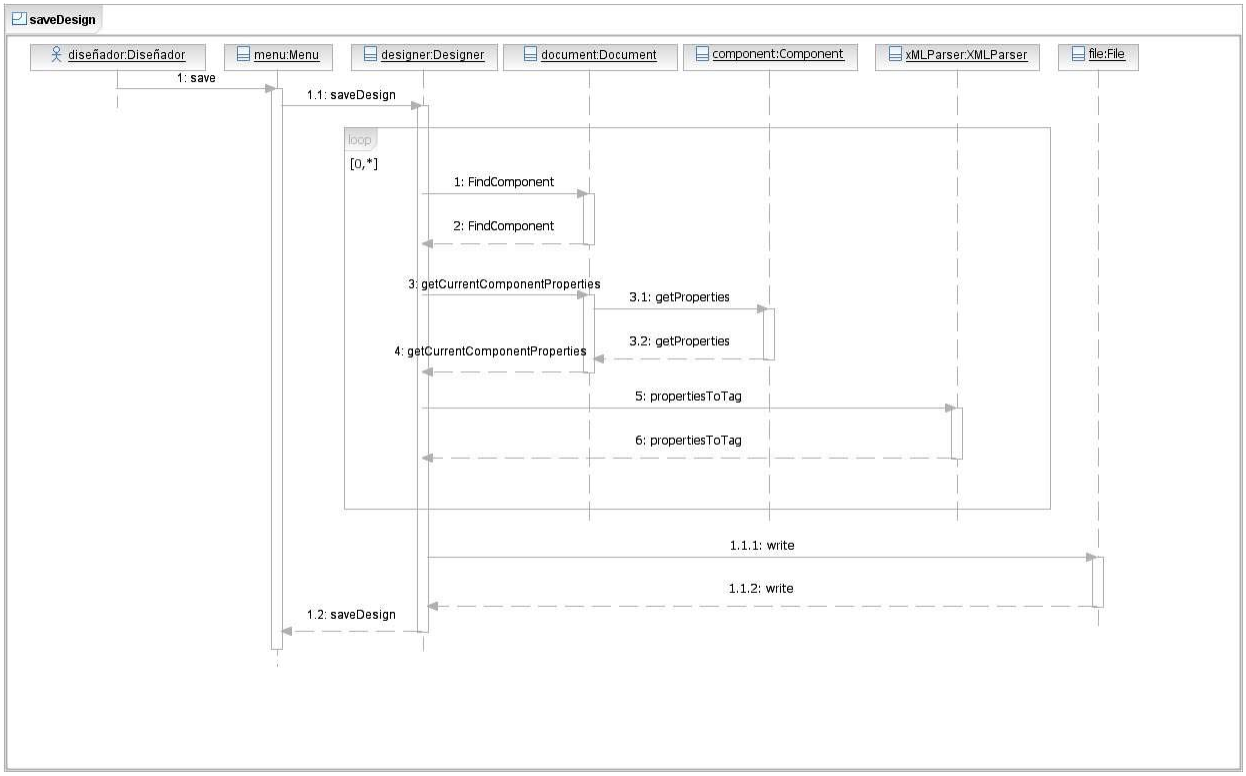

Fig. 22 Diagrama de secuencia que representa la salva de un diseño.

4.7.3 Insertar componente.

El diseñador haciendo uso del botón correspondiente en la barra de herramientas o de la opción insertar de la barra de menú selecciona un componente y decide insertarlo sobre una sección del diseño. La instancia *"design"* controla todo el proceso, para lo cual reconoce cual es la sección que tiene el foco en el momento de inserción (sección sobre la que se hace click para insertar el componente) y a través de la instancia de tipo *"docuement"* creada al crear el diseño se invoca el método *"newComponent"* de la instancia de tipo *"Section"* la cual invoca la construcción del componente haciendo uso de la fábrica de componentes de tipo *"ComponentFactory"* y adicionándolo a la lista de componentes de la sección. Por último, se muestran las propiedades del componente insertado en el inspector de propiedades.

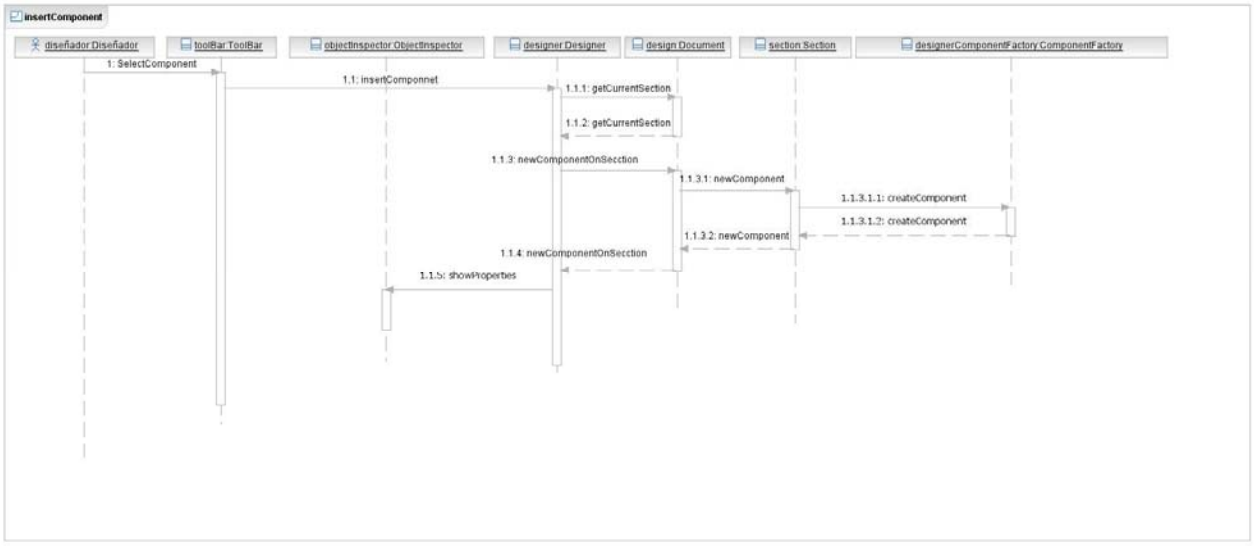

Fig. 23 Diagrama de secuencia correspondiente a la inserción de un componente.

4.7.4 Modificar propiedades de un componente.

El diseñador selecciona un componente haciendo clic sobre el componente en el inspector de objetos, pasándole a la clase *"Designer"* una referencia al componente seleccionado, este le pide las propiedades al componente, las cuales toma y envía al editor de propiedades para que este las muestre, una vez modificados los valores mostrados en el editor de propiedades a través del método *"setProperties"* se invoca el método *"setProperties"* del componente para que los cambios a nivel de instancia del componente se hagan efectivos.

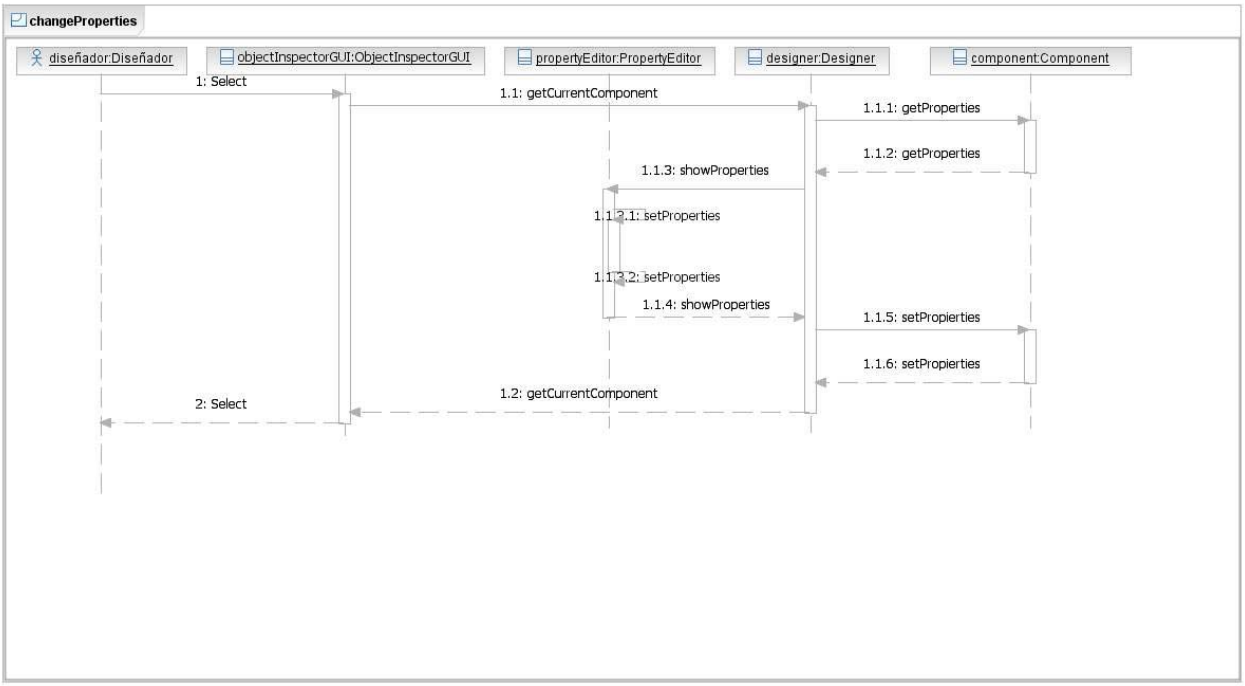

Fig. 24 Diagrama de secuencia que muestra el cambio de las propiedades de un componente.

4.7.5 Solicitud de los parámetros necesarios para la generación de un informe desde el SCADA.

A través del método *"get Parameters"* de una instancia de la clase interfaz *"EngineInterface"* el SCADA solicita los parámetros que se hayan declarado en el diseño de informe que se usará para generar el informe. Mediante esta llamada se invoca el método *"getParameters"* del *"Engine"* el cual carga el diseño haciendo uso del método *"loadDesignFile"* de la clase *"XMLParser"*. Por último haciendo uso del método *"getParameters"* de la clase *"XMLParser"* se obtiene la lista de parámetros que se retorna al SCADA.

Fig. 25 Diagrama de secuencia que representa la solicitud de parámetros desde el SCADA.

4.7.6 Mostrar informe (Previsualizar).

El SCADA haciendo uso de la clase *"EgineInterface"* hace una llamada al método *"generateReport"* con sus respectivos parámetros, dándole la orden al *"engine"* de generar un informe mediante el método *"preview"* el cual llama a su vez al *"loadDesignFile"* con el cual carga el diseño de informe pasado como parámetro desde la llamada del método *"generateReport"* haciendo uso de la clase de acceso a datos *"File"* con el método *"read"*. Después de cargar el diseño se pasa a la creación del mismo mediante el método *"buildReport"* de donde se ejecuta la creación de los componentes concretos llamando al método *"buildComponent"* de la clase controladora *"engine"* que llama a la factoría con el método

"createComponent" para así crear los *"visualComponent"* en el informe haciendo uso del método *"buildInReport"*. Después de haber creado la *"widget Preview"* mediante el método *"show"* y de haber creado todo el informe este es mostrado.

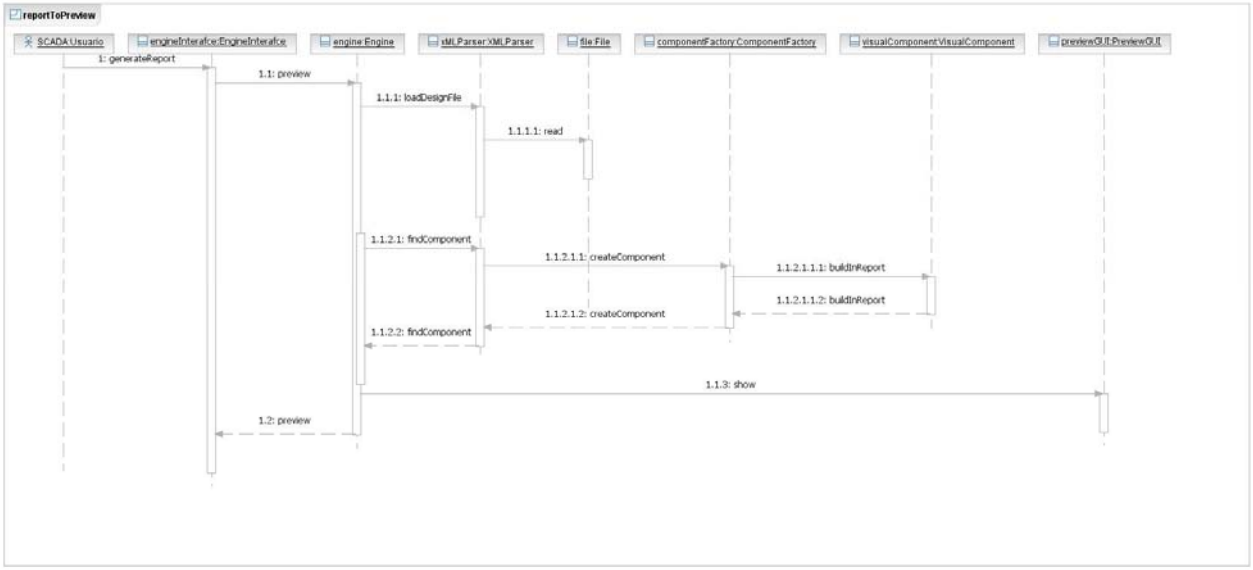

Fig. 26 Diagrama de secuencia que representa la previsualización de un informe.

4.7.7 Imprimir informe.

El SCADA haciendo uso de la clase *"EgineInterface"* hace una llamada al método *"print"* con sus respectivos parámetros dándole la orden al *"engine"* de generar un informe mediante el método *"print"* el cual llama a su vez al *"loadDesignFile"* con el cual carga el diseño de informe pasado como parámetro desde la llamada del método *"generateReport"* haciendo uso de la clase de acceso a datos *"File"* con el método *"read"*. Después de cargar el diseño se pasa a la creación del mismo mediante el método *"buildReport"* de donde se ejecuta la creación de los componentes concretos llamando al método *"buildcomponent"* de la clase controladora *"engine"* que llama a la factoría con el método *"createComponent"* para así crear los "*visualComponent*" en nuestro informe haciendo uso del método *buildInReport".* Después de haber creado todo el informes se llama a la interfaz de impresión que se invoca mediante un método de una instancia de *"Qprinter"* y haciendo uso del método *"setOutputFormat::printer".*

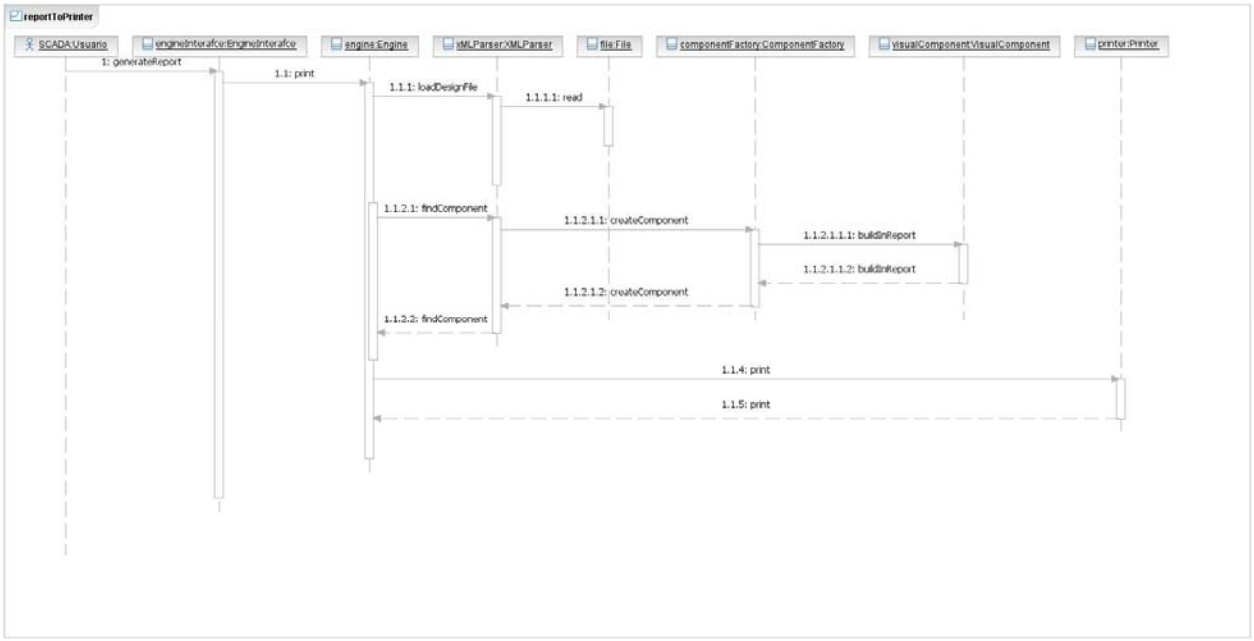

Fig. 27 Diagrama de secuencia que representa la impresión de un informe.

4.7.8 Almacenar informe en formato PDF.

El SCADA haciendo uso de la clase *"EgineInterface"* hace una llamada al método *"generatePDF"* con sus respectivos parámetros, dándole la orden a l *"Engine"* de generar un informe mediante el método *"generatePDF"* el cual llama a su vez al *"loadDesignFile"* con el cual carga el diseño de informe pasado como parámetro desde la llamada del método *"generateReport"* haciendo uso de la clase de acceso a datos *"File"* con el método *"read"*. Después de cargar el diseño se pasa a la creación del mismo mediante el método *"buildReport"* de donde se ejecuta la creación de los componentes concretos llamando al método *"buildcomponent"* de la clase controladora *"Engine"* que llama a la factoría con el método *"createComponent"* para así crear los *"visualComponent"* en el informe haciendo uso del método *"buildInReport"* Después de haber creado todo el informe pasamos a la interfaz de impresión que se llama mediante una instancia de *"Qprinter"* y se hace uso del método *"setOutputFormat::PDF".*

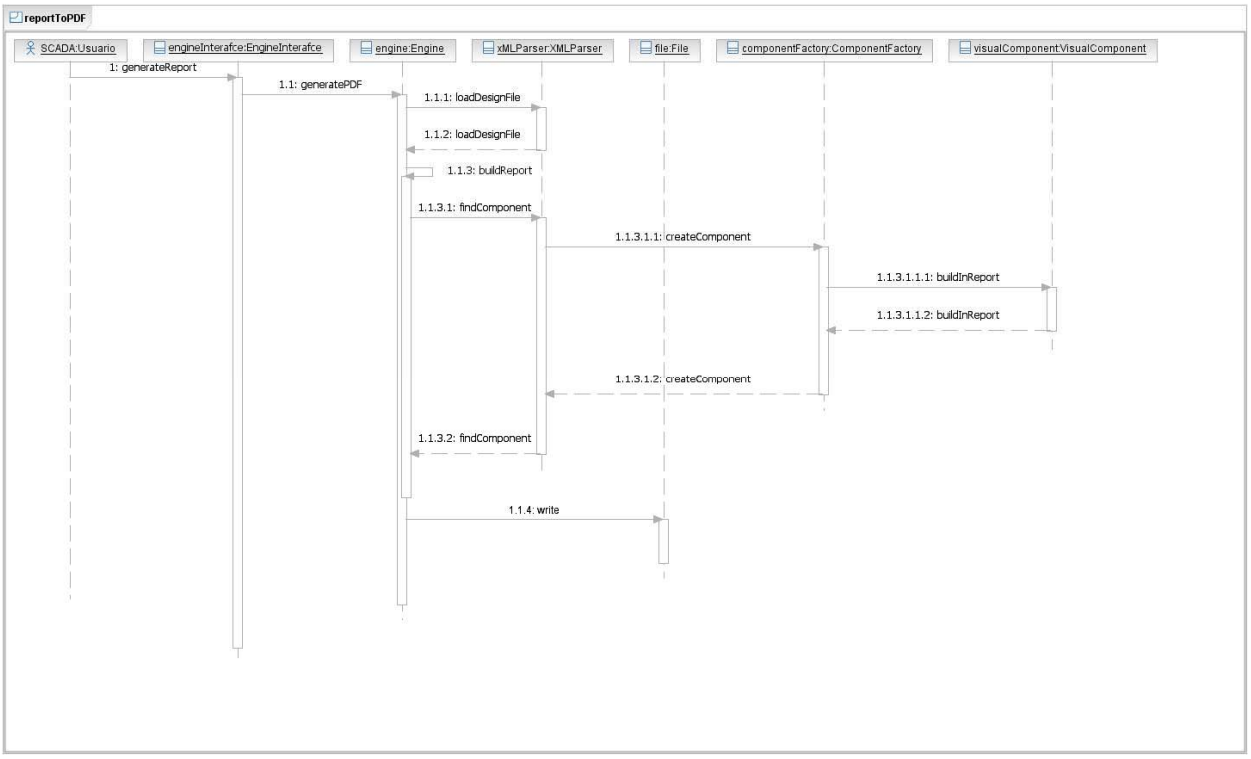

Fig. 28 Diagrama de secuencia que representa la generación y salva de un informe en PDF.

4.8 Principales vistas de los prototipos de interfaz de usuario.

Una aplicación con una interfaz bien diseñada debe tener, además de un buen diseño gráfico, una buena navegabilidad, usabilidad y distribución de los contenidos. Persiguiendo este objetivo se han seguido, para el diseño de la interfaz de usuario del Generador de informe, los siguientes principios:

- Requerir un mínimo proceso de aprendizaje y permitir su utilización desde el primer momento, por cualquier persona que tenga un mínimo dominio de la computación.
- Evitar el uso inadecuado o excesivo de las metáforas, que puede dificultar enormemente el aprendizaje del usuario.
- Garantizar la legibilidad, el color de los textos debe contrastar con el del fondo, y el tamaño de fuente debe ser suficientemente grande.
- Evitar elementos invisibles de navegación que han de ser inferidos por los usuarios, menús desplegables, indicaciones ocultas, etc.
- Requerir de los usuarios un mínimo esfuerzo para alcanzar sus objetivos.
- Evitar las caídas inesperadas de la aplicación y los enlaces rotos.
- Limitar el número de acciones que puede realizar el usuario sobre la aplicación, mostrando sugerencias (opciones) para cada posible acción, evitando así al máximo los errores de usuario.

- Mostrar al usuario solamente aquellas opciones a las que, dado su rol en el negocio, tiene derecho a acceder.
- Mostrar al usuario, siempre que vaya a realizar una acción relevante sobre el sistema, un mensaje de confirmación que le permita asegurarse de que es correcta la opción seleccionada.
- Mostrar la mayor cantidad de información acerca de las opciones brindadas en un momento dado, de modo que el usuario siempre sepa cuáles son las operaciones a las que puede acceder y en qué consiste exactamente cada una.

A Continuación presentamos las principales interfaces de usuario que a modo de prototipo se han desarrollado para la aplicación.

La primera interfaz que se muestra al usuario cuando abre el diseñador es la interfaz de autentificación mediante la cual se autentificará ante el sistema, quedando establecido con este acto los privilegios ante el sistema. Esta interfaz no se mostrará cunado el usuario solicite la ejecución del diseñador desde el SCADA y ya este autentificado ante el sistema con privilegio de mantenedor.

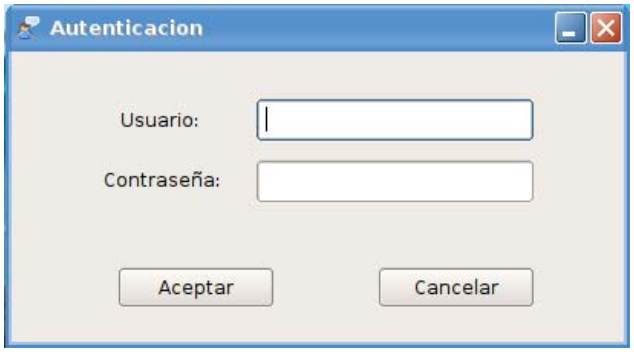

Fig. 29: Interfaz de autentificación ante el sistema.

Una vez autentificado el usuario ante el sistema se mostrará la interfaz principal del diseñador (Fig. 18) mediante la cual tendrá derecho a usar todas las funcionalidades del sistema.

Esta interfaz básicamente está compuesta por:

- Barra de menú que da acceso a todas las opciones del sistema.
- Distintas barras de componentes que agrupan los componentes de acuerdo a sus funcionalidades.
- Inspector de componentes y propiedades donde se muestran todos los componentes que se están usando en el diseño activo, y mediante el cual se pueden gestionar las propiedades del componente seleccionado.
- Área de trabajo, donde se mostraran los diseños abiertos.
- Barra de estado donde se muestra el estado de algunos recursos como la posición en pantalla del puntero del mouse, el estado de insert, etc.

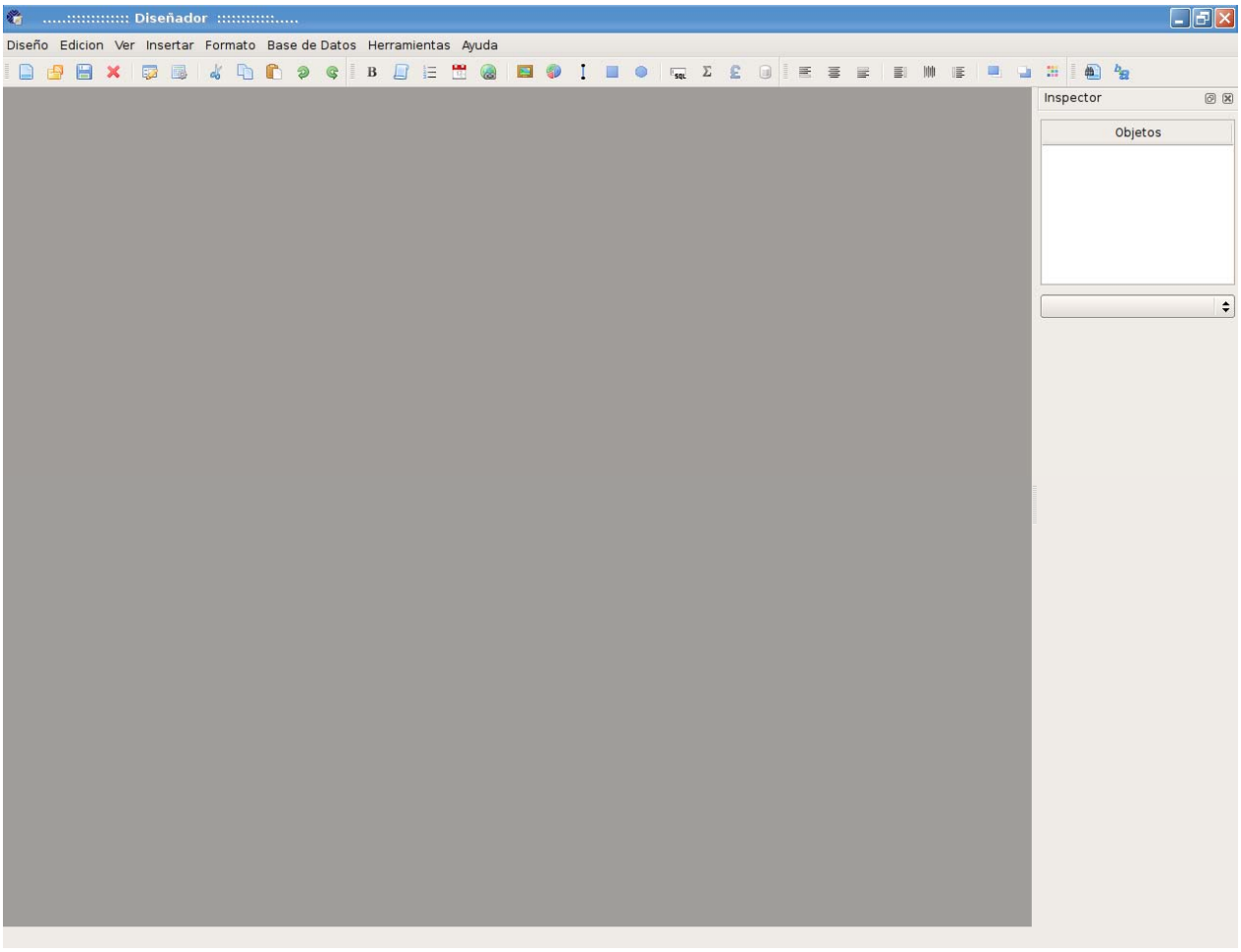

Fig. 30: Interfaz del diseñador, sin diseños abiertos.

Cuando se decide crear un nuevo diseño se hará uso de una serie de ventanas que establecerán un conjunto de requisitos y restricciones para el diseño que se realizará (Ver Fig.19).

Otra de las ventanas con estas características es la que permite establecer el formato de las páginas sobre las cuales finalmente se generará el informe (Fig. 20)

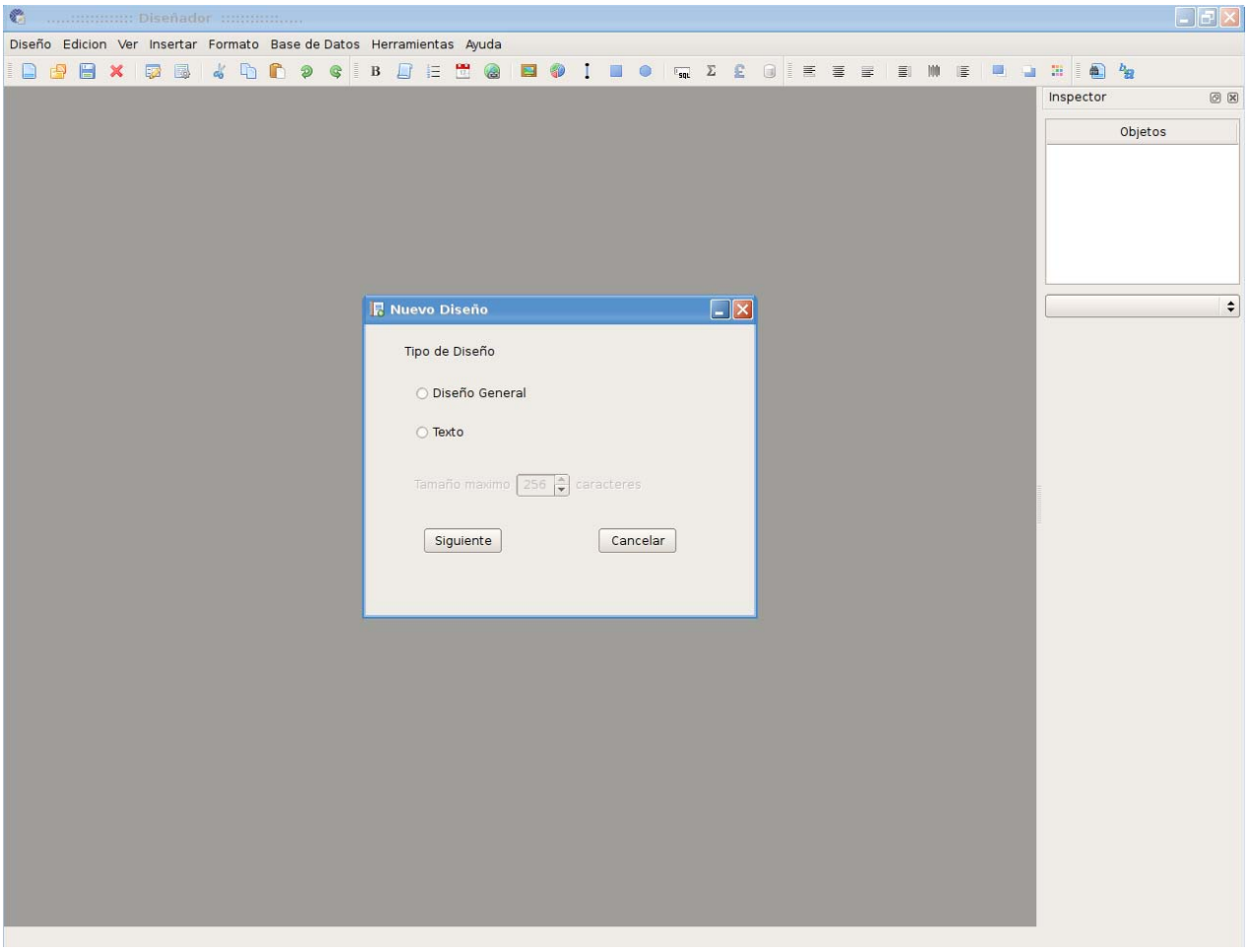

Fig. 31: Interfaz para establecer características esenciales del diseño a desarrollar.

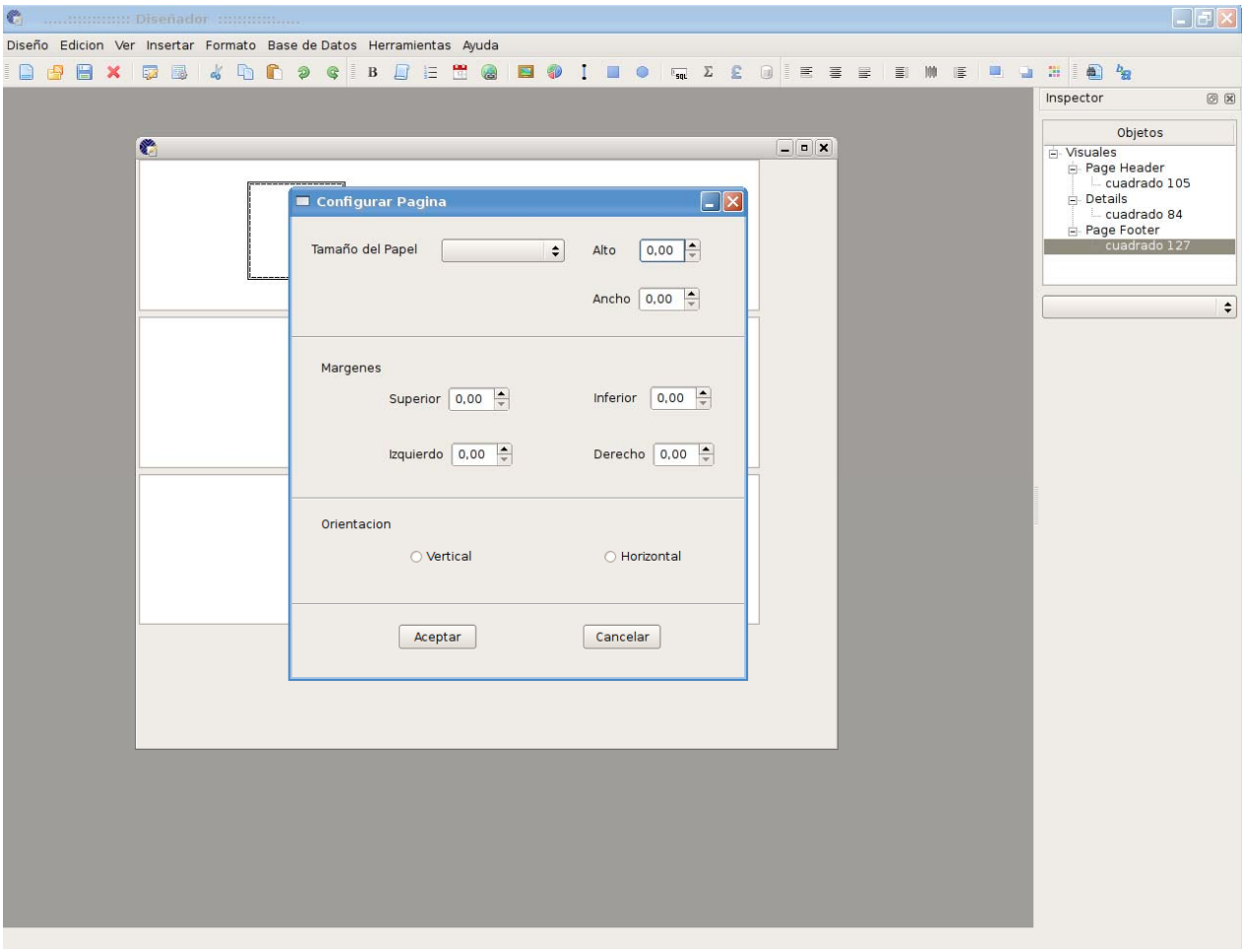

Fig. 32: Interfaz para configurar las características de la pagina donde se generará el informe.

Para facilitar la creación de un nuevo diseño el sistema constará con una serie de diseños previamente elaborados los cuales podrán ser empleados como plantillas sobre las cuales desarrollar el diseño que se desea realizar. El prototipo de interfaz mostrado en la Fig. 21 es el que permitirá elegir estos diseños preelaborados, para lo cual se mostrarán agrupados por características comunes bajo distintas pestañas.

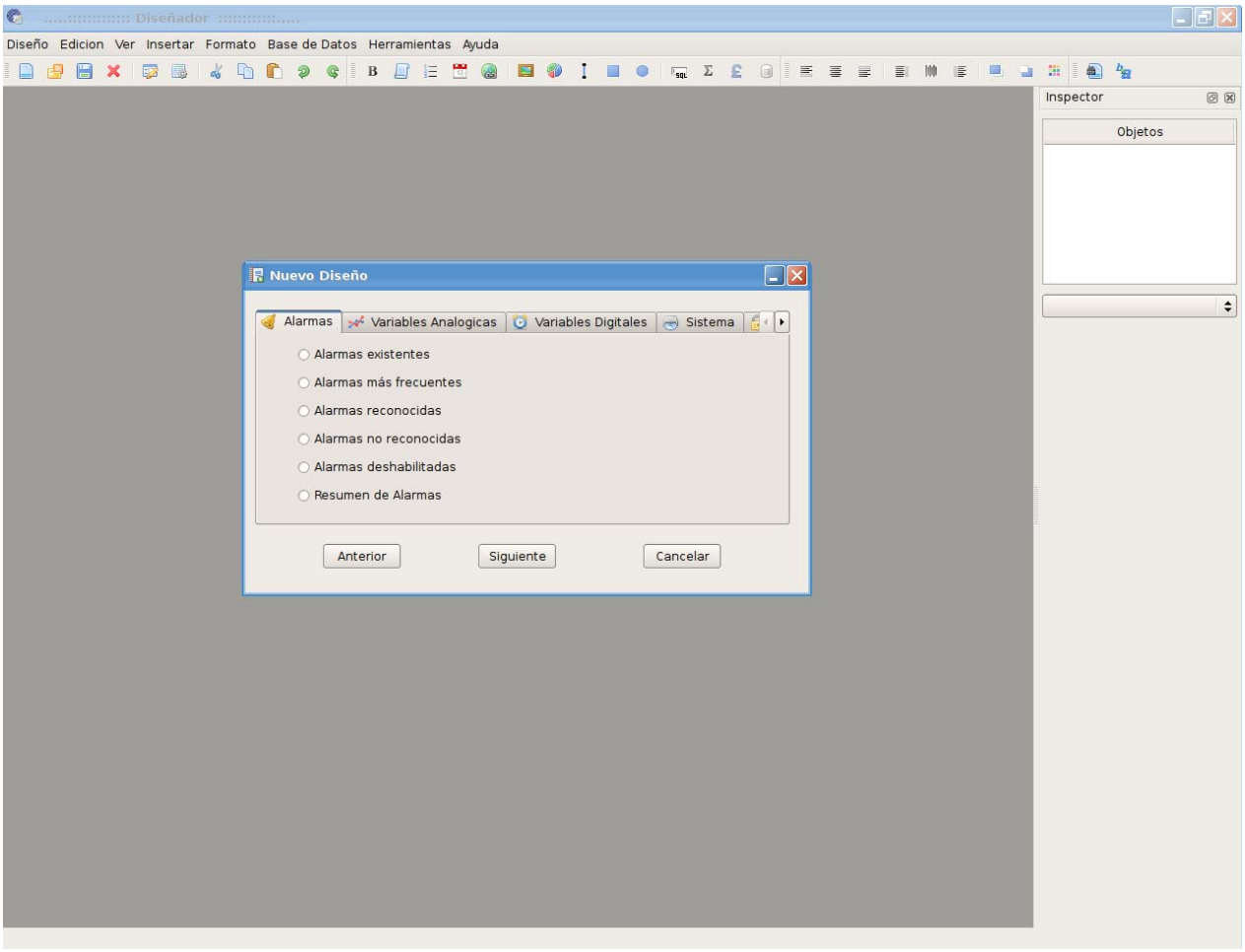

Fig. 33: Interfaz para elegir el tipo de diseño preelaborado sobre el cual se trabajará.

Para la definición de un diseño se presentará sobre el área de trabajo una ventana que contendrá secciones sobre las cuales se irán insertando los distintos elementos de diseño para lograra la apariencia final del informe que se desee. Esto se muestra en la interfaz de usuario que aparece en la Fig. 22, en la misma se puede apreciar también de manera muy reducida el comportamiento del inspector de componentes y el editor de propiedades.

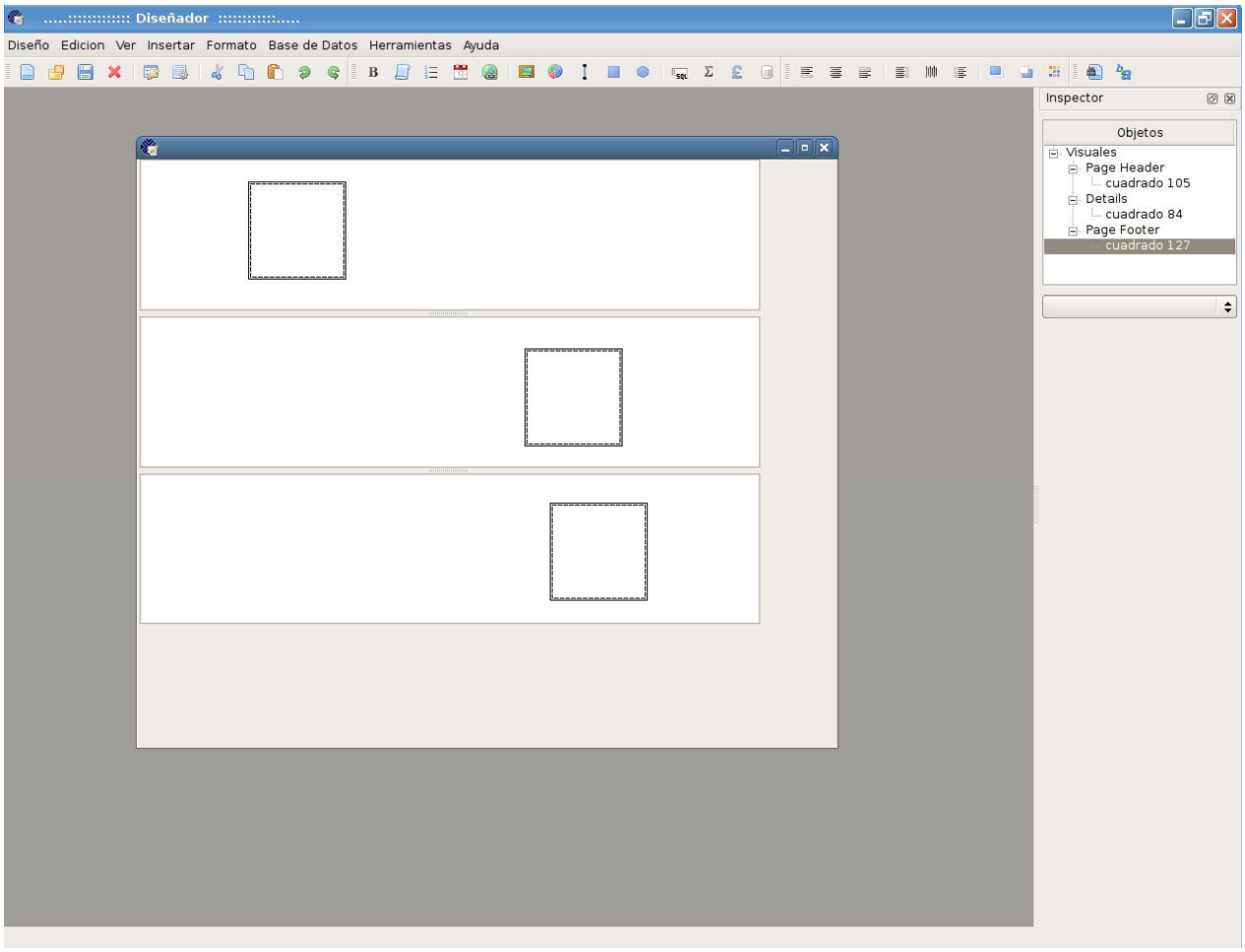

Fig. 34: Interfaz del diseñador. Con un diseño abierto.

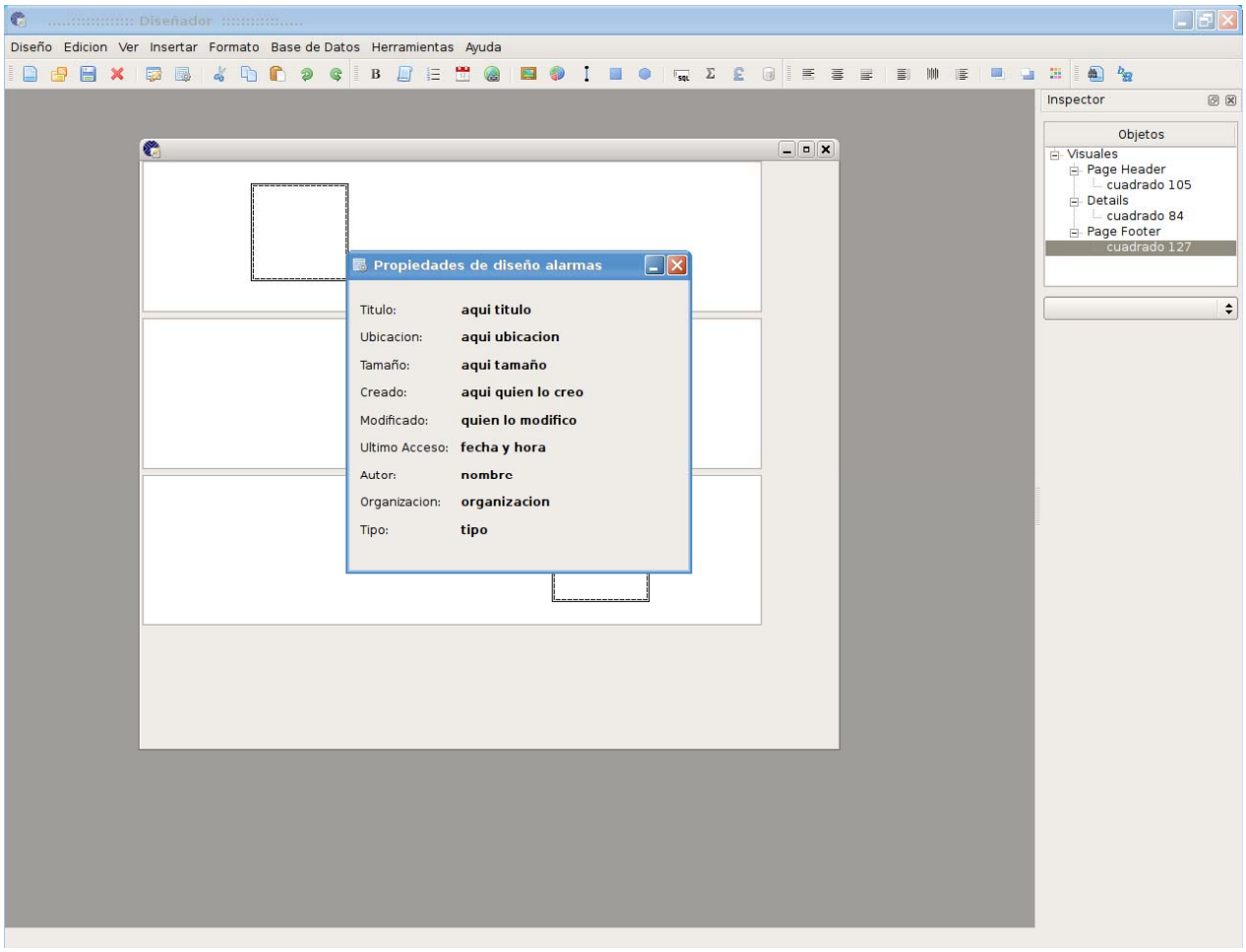

Fig. 35: Interfaz que permite mostrar las principales características de un diseño.

En la figura anterior se muestra una interfaz para informar algunas características fundamentales del diseño.

Una herramienta muy útil para la persona que se encuentra realizando un diseño es el previsualizador del informe, mediante el cual se mostrará el informe generado a partir del diseño que se realiza usando los datos disponibles. En la Fig. 24 se puede observar un prototipo de la interfaz para este previsualizador la cual permitirá ejecutar una serie de acciones sobre el informe obtenido como imprimirlo, guardarlo, navegar por las páginas del mismo, entre otras.

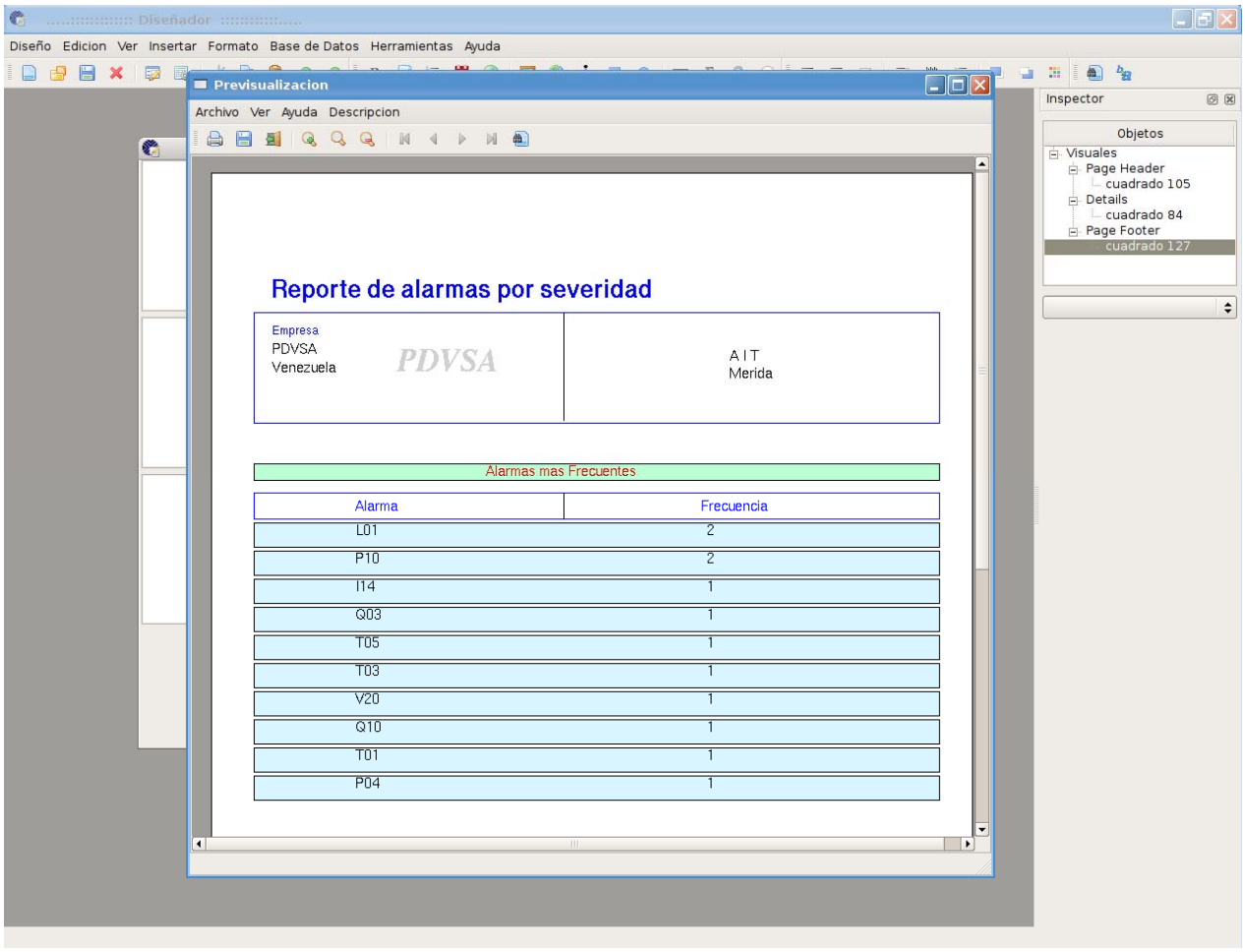

Fig. 36: Interfaz del generador, donde se muestra un informe generado.

4.9 Conclusiones.

Finalmente se ha obtenido una arquitectura base para el sistema, que junto a los diagramas de clases y secuencias, así como los prototipos de interfaz de usuario, constituyen un modelo del generador de informes para un SCADA.

Conclusiones

1. A pesar de las grandes similitudes de los generadores de informes para SCADAs con generadores de informes genéricos, los generadores de informes existentes sobre software libre no cumplen con todos los requisitos exigidos para su uso integro como solución para el desarrollo del generador de informes para el SCADA Nacional de PDVSA.
2. Todos los generadores de informes estudiados usan una filosofía muy similar en su funcionamiento, consistente en la creación de diseños en un ambiente de edición y posterior generación del informe, haciendo uso del diseño creado y los datos necesarios, provenientes de una fuente de datos determinada.
3. El paradigma de software libre facilita en gran medida la reutilización de código y un desarrollo acelerado de aplicaciones, posibilitando el estudio exhaustivo de código existente afín con la aplicación que se diseña.
4. Para realizar un levantamiento de requisitos cunado no se tienen conocimientos sólidos del negocio abordado, es conveniente valerse de varias técnicas que permitan validar cada uno de los requisitos, además, cuando se trata de proyectos donde participan un gran numero de especialistas es muy conveniente hacer uso de herramientas que faciliten la administración de dichos requisitos.
5. La definición de los requerimientos del sistema permite estructurar el modelo de casos de uso del sistema y contribuye en la descripción de los caso de uso para una mejor compresión de la funcionalidad brindada, además de guiar las funcionalidades a plasmar en un prototipo de interacción con el cliente.
6. La arquitectura en capas es muy conveniente para el desarrollo del generador de informes para un SCADA, dado la facilidad de resistir el cambio de los requisitos y el alto desacoplamiento logrado con los demás subsistemas integrantes del SCADA.
7. La realización de prototipos de interfaz de usuario basados en los casos de uso permiten la interacción efectiva con el cliente, si además se incorporan funcionalidades a estos prototipos respetando las clases del diseño desarrollado posibilita hacer pruebas al diseño.
8. La aplicación de patrones en el diseño garantiza su escalabilidad, mantenibilidad y flexibilidad.
9. Los diseños independientes de plataforma favorecen la lógica de comportamiento de un sistema, pero al entregárselo a los desarrolladores es muy conveniente hacerlos dependiente de las bibliotecas elegidas para el desarrollo, lo cual presupone un conocimiento previo de las clases brindadas por la librería en el momento de diseño.

Recomendaciones.

- Continuar usando el diseño obtenido para el desarrollo del generador de informe del SCADA Nacional de PDVSA.
- Capturar y validar posibles nuevos requisitos y extender el diseño adicionándole las nuevas funcionalidades requeridas que permitan la mejora continua y un desarrollo iterativo e incremental de la propuesta obtenida.
- Continuar trabajando en el establecimiento de dependencias entre las clases genéricas planteadas en el diseño y las clases concretas proporcionadas por las bibliotecas QT y Boost que faciliten a los desarrolladores la complementación del mismo.
- Evaluar la idoneidad del diseño propuesto para su uso en otros SCADAs de facturación nacional que adolecen de las funcionalidades descritas en el presente trabajo.

Referencias Bibliográficas.

1 AGATA.ORG. *Agata Report*. Disponible en: http://www.agata.org.br/us/index.php (11/01/2007)

2 ALBERTO MOLPECERES TOURIS, M. P. M. *Arquitectura empresarial y software libre, J2EE*, 18/08/2002. [Disponible en: http://www.javahispano.org/articles.article.action?id=70 (05/01/2007)

3 ARINA OYAGA, G. *informáticos en tiempo real. Ejemplos prácticos.* España, Marzo de 2000. p.

4 BANKHACKER.COM. *Web Services*. Disponible en: http://web-services.bankhacker.com/ (05/01/2007).

5 CRYSTALREPORTS.ORG. *Crystal Reports XI from Business Objects*. Disponible en: http://www.businessobjects.com/products/reporting/crystalreports/start_xi.asp (11/01/2007)

6 DISTEFANO., M. *Comunicaciones en entornos industriales.* . Mendoza. Argentina, Universidad Nacional de Cuyo., 1999. p.

7 FOUNDATION, O. *OPC Historical Data Access Specification.* . 1.20., V., 2003.

8 GONZÁLEZ, C. D. *Curso: Integral de diseño. Programación de sitios dinámicos con MySQL y PHP*. Disponible en: http://www.usabilidadweb.com.ar/x_int.php(05/01/2007)

9 ---. *Curso: Sitios Web dinámicos con Base de Datos PostgreSQL y PHP.* Disponible en: http://www.usabilidadweb.com.ar/postgre.php (05/01/2007).

10 HERNÁNDEZ BASSO, M. *¿Convivencia con lo invisible?* 2. Ciudad de la Habana; Cuba, Septiembre 2002. p.

11 IEEE. *Tutorial Course. Course Text. Fundamental of supervisory control systems;*. Engineering Societies Library, 18 de febrero 1982.

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

13 JASPERREPORTS. *Home*. Disponible en: http://jasperreports.sourceforge.net/ (07/01/2007)

14 JFREE.ORG. *JFreeReport*. Disponible en: http://www.jfree.org/jfreereport/ (07/01/2007)

15 KOFFICE.ORG. *The KOffice Project*. Disponible en: http://www.koffice.org/kugar/

16 LARMAN, C. *UML y patrones. Introducción al análisis y diseño orientado a objetos*. 2. 2004. p.

17 MANAGER, R. *Report Manager Official page*. Disponible en: http://reportman.sourceforge.net/ (07/01/2007)

18 MICROSOFT. *Microsoft Office Online: Access*. Disponible en: http://office.microsoft.com/es-es/FX010857913082.aspx (11/01/2007)

19 MOLPECERES, A. *Procesos de desarrollo: RUP, XP y FDD.* Disponible en: http://www.javahispano.org/articles.article.action?id=76 (05/01/2007).

20 OPENOFFICE.ORG. *Home*. Disponible en: http://www.openoffice.org/ (11/01/2007)

21 OPENREPORT. *Open Source and profesional reporting solution*. Disponible en: http://openreport.org/ (07/012007)

22 REKALLREVEALED.ORG. *Rekall: The database front-end for KDE and the Web*. Disponible en: http://www.rekallrevealed.org/kbExec.py

23 SCADA, E. P. *Especificación de requisitos de software.*: *Versión 3.0*. Venezuela, 2006.

24 ---. *Plataforma SCADA PDVSA.* Venezuela, PDVSA, 2005.

25 SCADA, S. *Red Nacional de Gasoductos;* . *Sistema SCADA*. Colombia, Bucaramanga, 2002.

26 WIKIPEDIA, L. E. L. *MySQL*. Disponible en: http://es.wikipedia.org/wiki/MySQL (05/01/2007)

27 ---. *Oracle*. Disponible en: Oracle. http://es.wikipedia.org/wiki/Oracle (05/01/2007)

28 ---. *Sistemas Gestores de Bases de Datos*. Disponible en: http://es.wikipedia.org/wiki/DBMS (05/01/2007)

29 YUNIER SOTO LÓPEZ, N. M. Y. S. R. *Propuesta para un sistema de Catalogación y Recuperación de Recursos de Información.* Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría" junio 2004. p.

Bibliografía.

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

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

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

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

Anexos

Anexo 1: Evaluación de generadores de informes.

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

29.1 Generadores de informes propietarios:

29.1.1 Crystal Reports Profesional

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

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

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

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

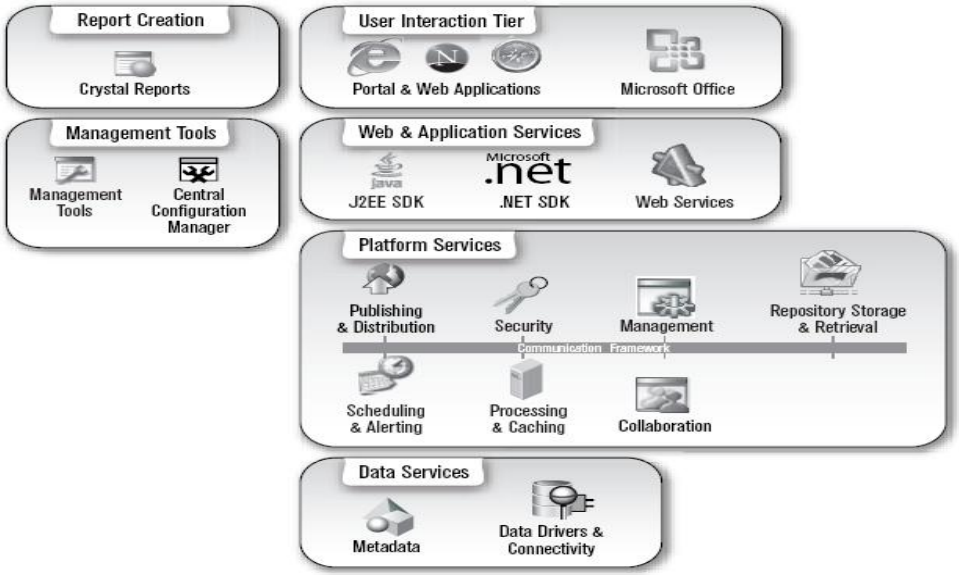

Fig. 37. Arquitectura funcional de Crystal Reports Server.

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

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

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

29.1.2 Access de Microsoft Office.

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

- Crear un informe con el “Asistente para informes”.
- Previsualizar e imprimir un informe.
- Mover y cambiar el tamaño de un control.
- Modificar las propiedades de formato (fuente, estilo, tamaño de fuente, color, título, etc.).
- Utilizar el Cuadro de controles para agregar controles.
- Utilizar secciones del informe (encabezados, notas a pie, detalles, etc.).
- Esta aplicación sólo es soportada por Microsoft Windows. Puede generar distintos formatos de salida (HTML, XML y XSL).

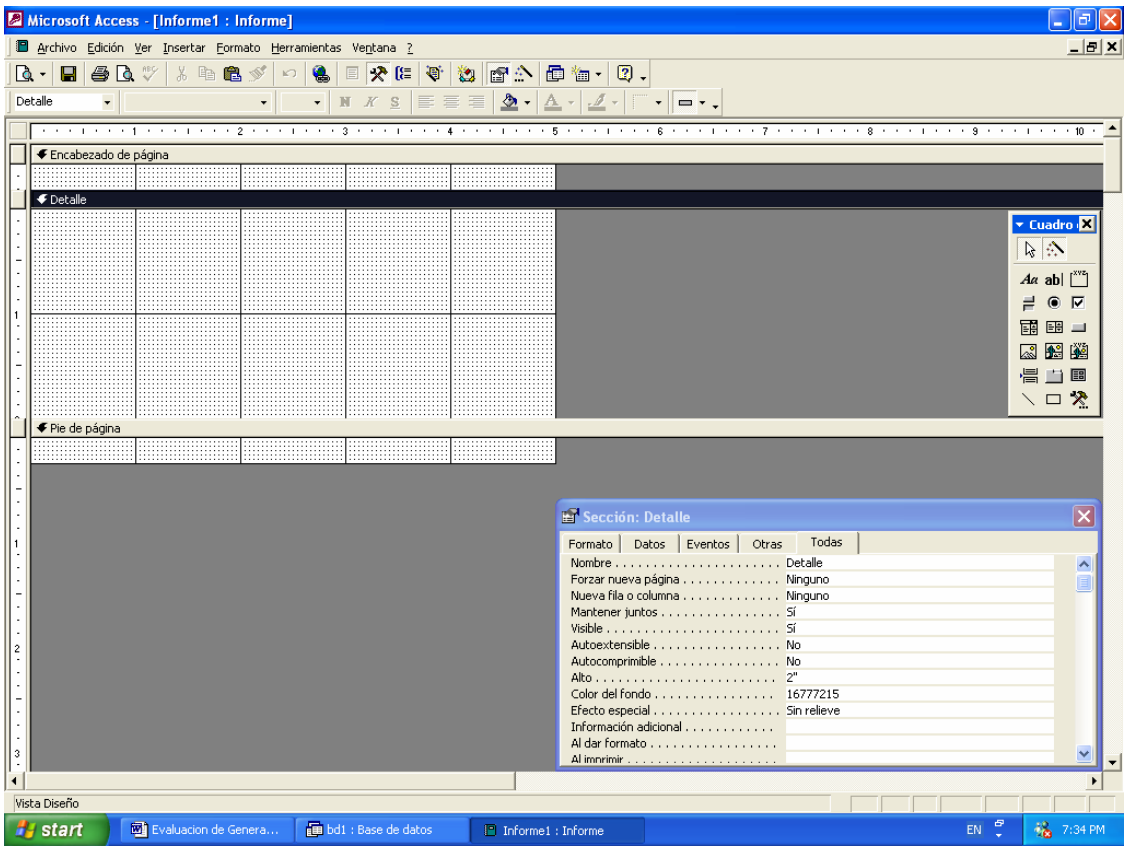

Fig. 38. Vista diseño de Access.

29.2 Generadores de informes de software libre

29.2.1 Report Manager

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

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

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

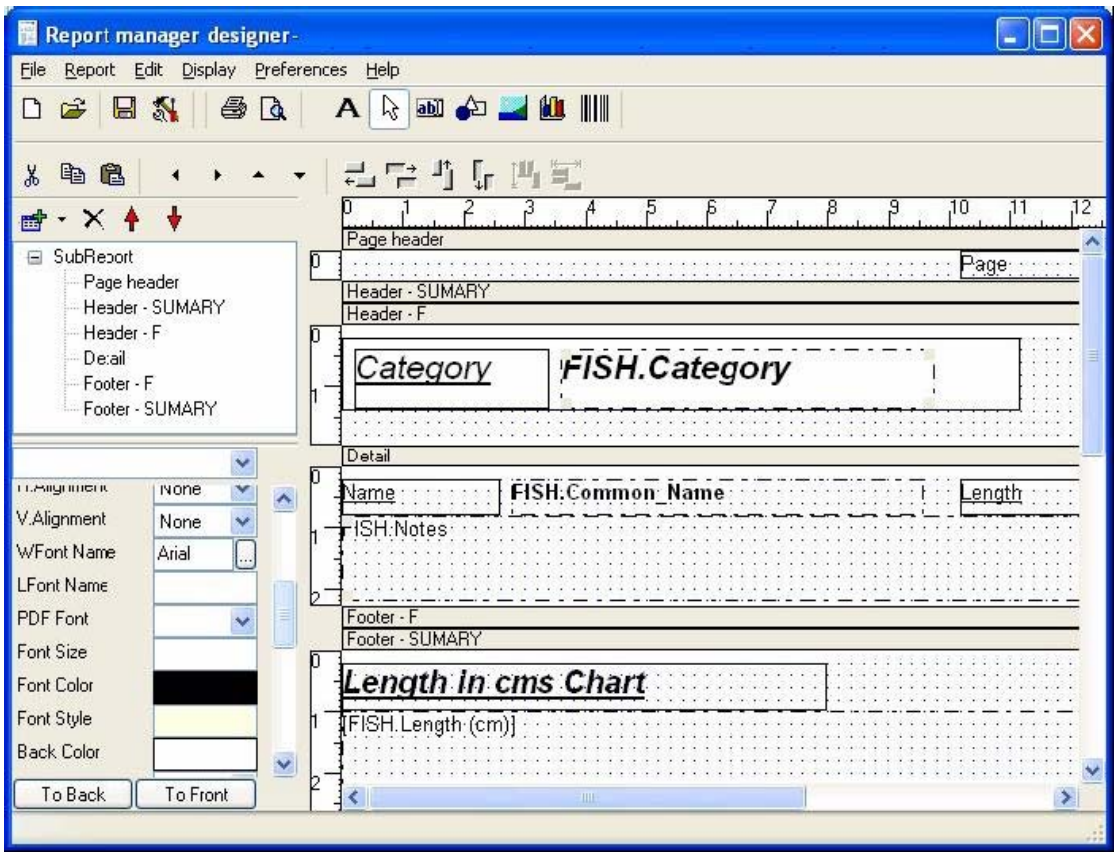

Fig. 39. Pantalla de l a aplicación Report Manager Designer.

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

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

29.2.2 NCReport

NCReport es una herramienta de diseño y generación de informes que permite diseñar de forma visual los informes además de posibilitar su impresión. El ambiente de trabajo para el diseño de informes es bastante amigable al facilitar el acceso a los componentes necesarios para el diseño mediante una barra de componentes y a través del menú. Los componentes que

se pueden utilizar son: fecha y hora, campo de expresión, tabla de referencia cruzada, subinforme, número de página, gráfico, texto fijo o etiquetas, imágenes estáticas, hipervínculo y autoforma (línea, cuadro).

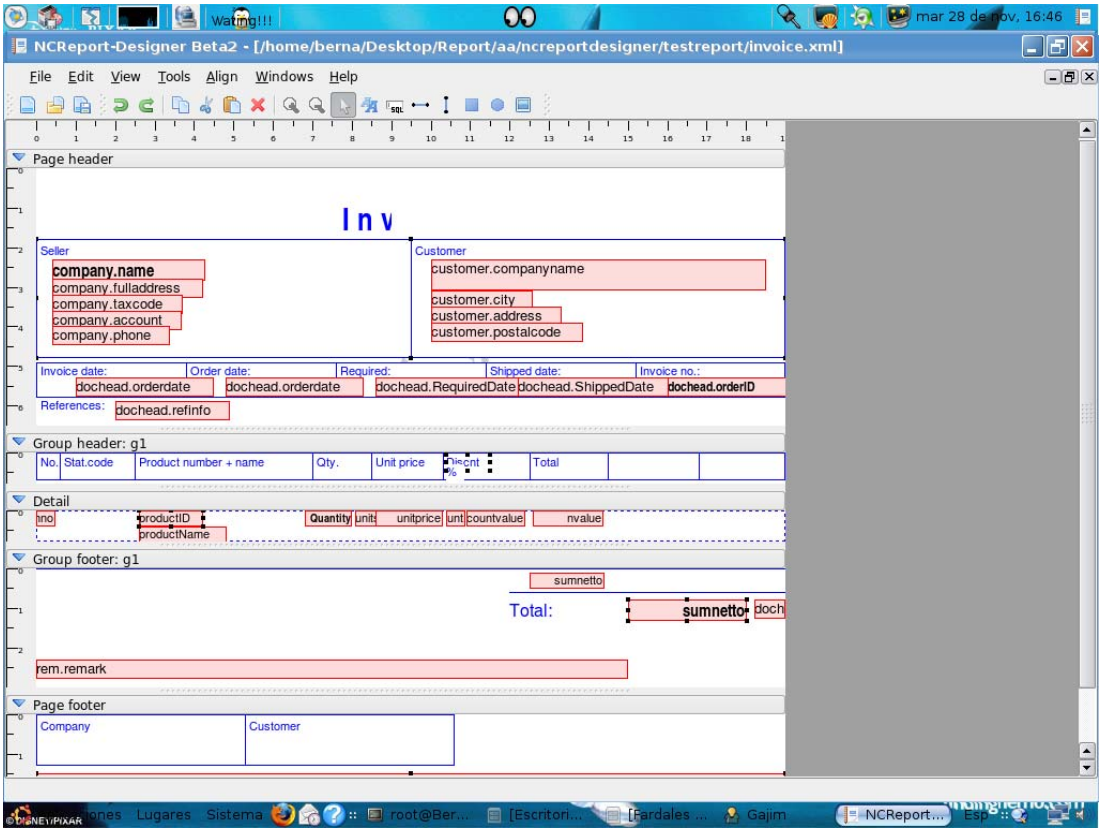

Fig. 40. Ambiente de trabajo en el diseñador de NCReport.

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

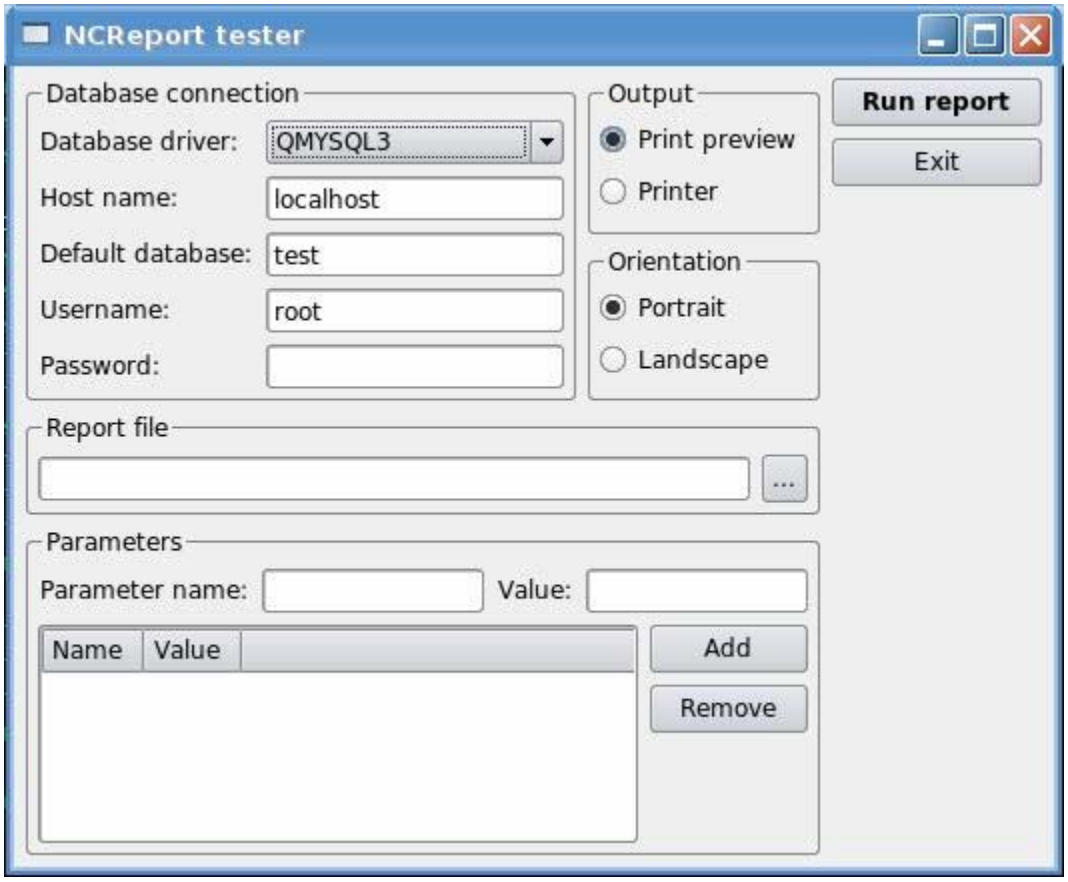

Fig. 41. Pantalla de configuración en NCReport.

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

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

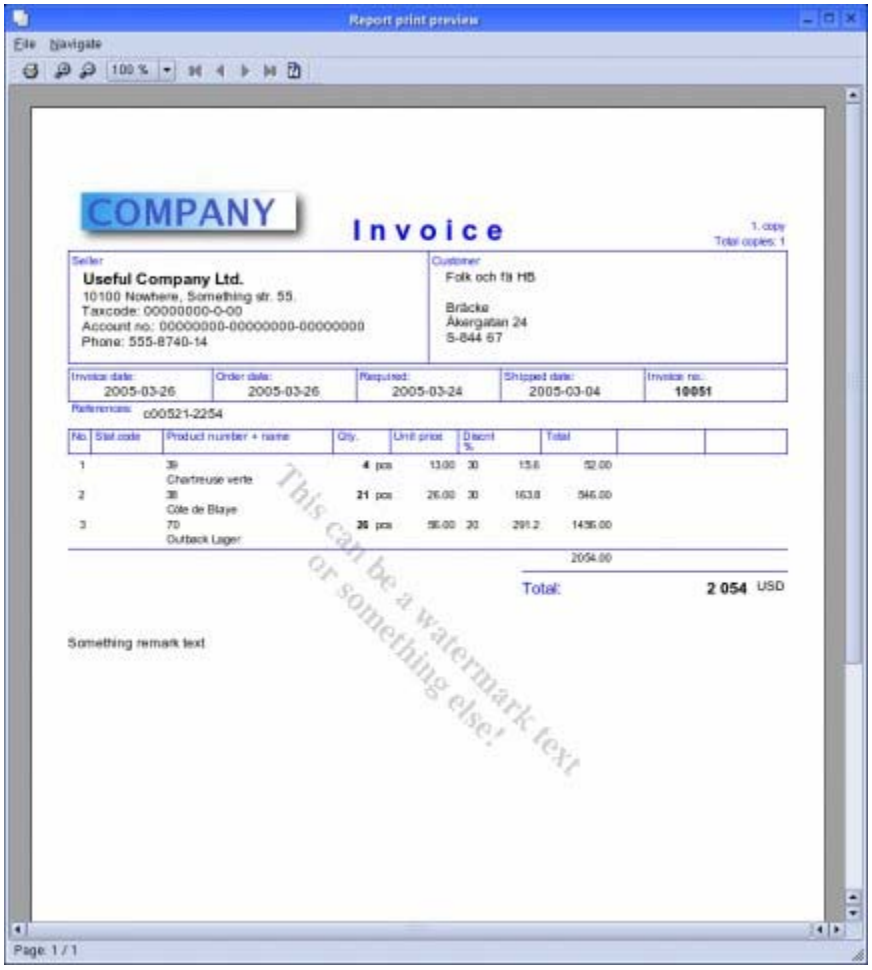

Fig. 42. Ejemplo de informe obtenido con NCReport.

29.2.3 Kugar:

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

El motor de informe de Kugar trabaja para unir los datos generados con una plantilla o diseño de informe, y producir después el informe final. Tanto los datos como la plantilla son especificados usando XML. Este enfoque implica que las aplicaciones solo necesitan preocuparse por la exportación de los datos en el formato determinado para ser entendidos por el motor.

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

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

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

- *Encabezado del informe*, la cual define las secciones de informe que se imprimen generalmente al principio de este.
- *Encabezado de la pagina*, la misma define las secciones de informe que se imprimen generalmente en la parte superior de cada página del informe.
- *Encabezado de detalle*, esta define las secciones del informe que se imprimen ante del detalle en un nivel dado del informe.
- *Detalle*, define las secciones del informe que contienen los datos del informe. El informe puede tener un número ilimitado de detalles.
- *Pie de detalle*, este elemento define las secciones del informe que se imprimen inmediatamente después de detalles de un nivel dado.
- *Pie de la pagina*, esta define las secciones del informe que se imprimen generalmente en el extremo inferior de cada página en el informe
- *Pie del informe* que define la sección del informe que es impresa generalmente al final del informe.

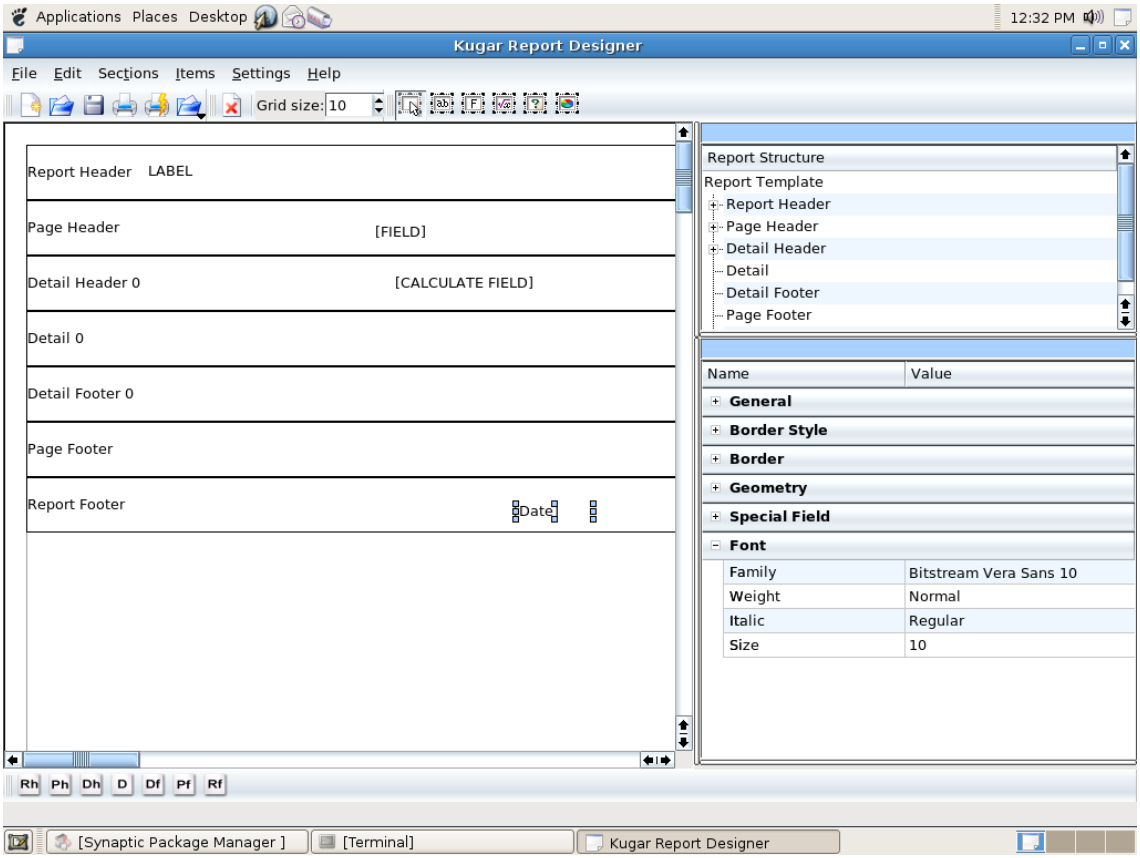

Fig. 43. Vista diseño de Kugar.

Están disponibles para la realización de diseños los siguientes elementos:

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

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

Las principales características que presenta la aplicación son:

- Diseñador de informes.
- Impresión de informes en formato Postscript.
- Base de datos neutral, los datos son suministrados al motor de informes en XML.
- Soporte para el acceso directo a base de datos.

- Los diseños de los informes son almacenados en XML.
- Cabeceras y pies de informe.
- Cabeceras y pies de página incluyendo números de página, fecha y hora actual.
- Cálculos de contadores, sumas, medias, varianzas y desviación típica.
- Formato de número basado en valores.
- Formato de fecha y moneda.
- Control de fuentes, colores, alineamiento de texto, etc.
- Dibujo de línea en diferentes estilos.

29.2.4 Agata Report.

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

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

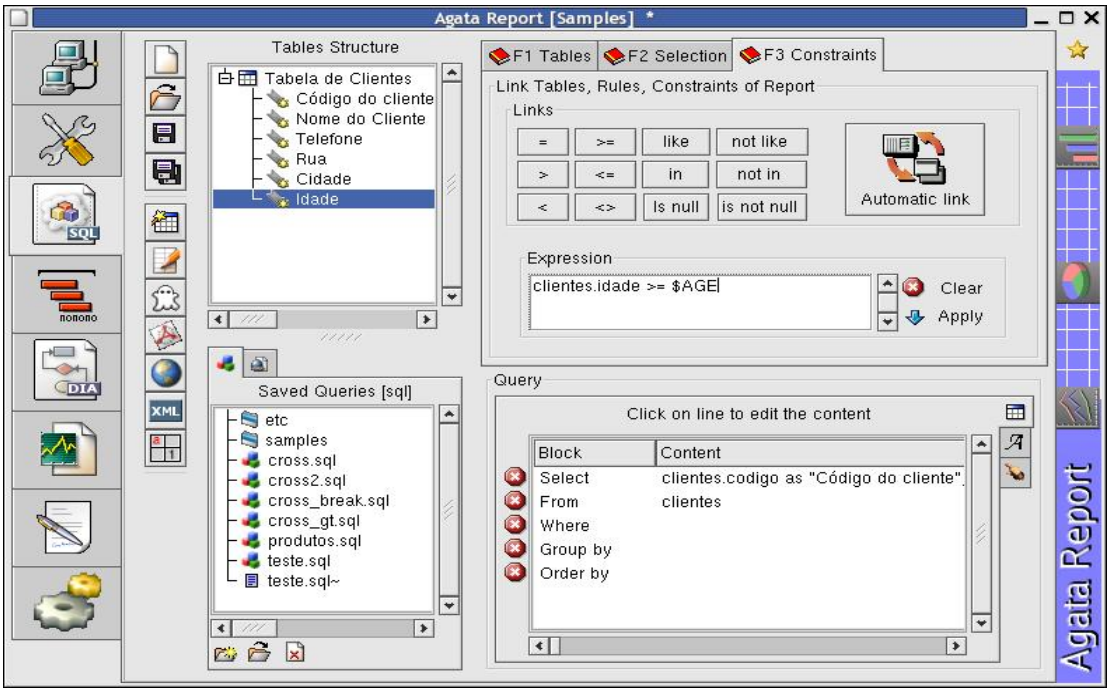

Fig. 44. Conexión a base de datos con Agata Report

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

Agata basa su programación en PHP-GTK que es la fusión del lenguaje de script PHP y la librería de objetos GTK+. Con esta librería y junto a la programación sencilla que proporciona el

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

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

29.2.5 Open Report.

Es una solución para la generación de documentos desde diferentes fuentes de datos. La aplicación tiene funciones para usar y presentar gráficos vectoriales, imágenes de mapas de bits, elementos formateados, plantillas de negocios, datos XML, y une todas estas primitivas para producir un fichero PDF.

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

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

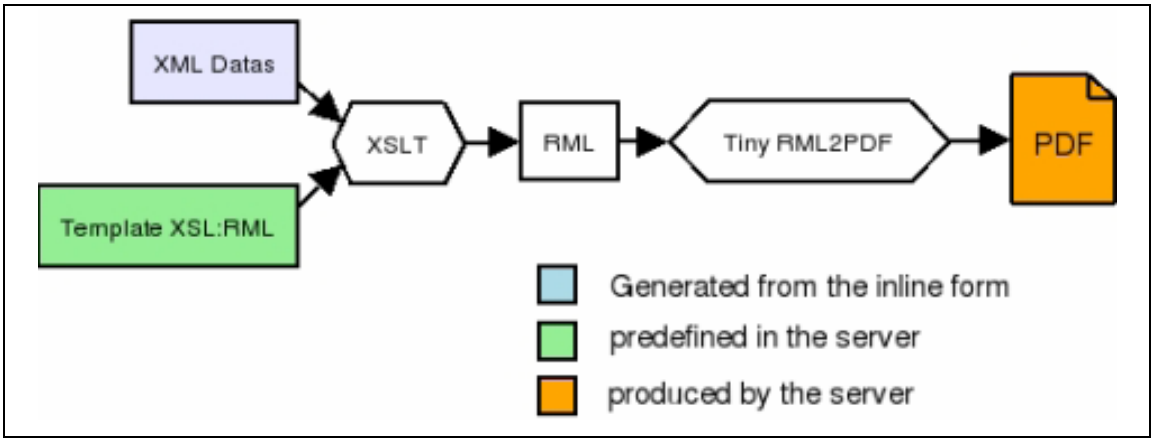

Fig. 45. Proceso de transformación a PDF con OpenReport**.**

29.2.6 Rekall.

Rekall(REKALLREVEALED.ORG) es una interfaz de usuario para bases de datos (Front-End) en entornos KDE similar a Microsoft Access. Sin embargo, no es y no incluye una base de datos. Debido a esto, los datos están almacenados en otra parte como en un servidor SQL. De esta forma Rekall permite gestionar de una forma totalmente gráfica bases de datos (por ahora MySQL, PostgreSQL, xBase con XBSQL, IBM DB2 y ODBC), permitiendo el diseño de formularios e informes, creación de peticiones a bases de datos, importación y exportación de tablas en diversos formatos para extraer, mostrar, y editar las informaciones contenidas en estas bases de datos, de una forma muy similar al Access de Microsoft, en otras palabras, podemos decir que es una especie de Access para GNU/Linux. La diferencia más importante de Rekall con respecto a Access es que de manera predeterminada no está ligada a ningún motor de base de datos en particular sino que se conecta a varios de los motores de bases de datos existentes.

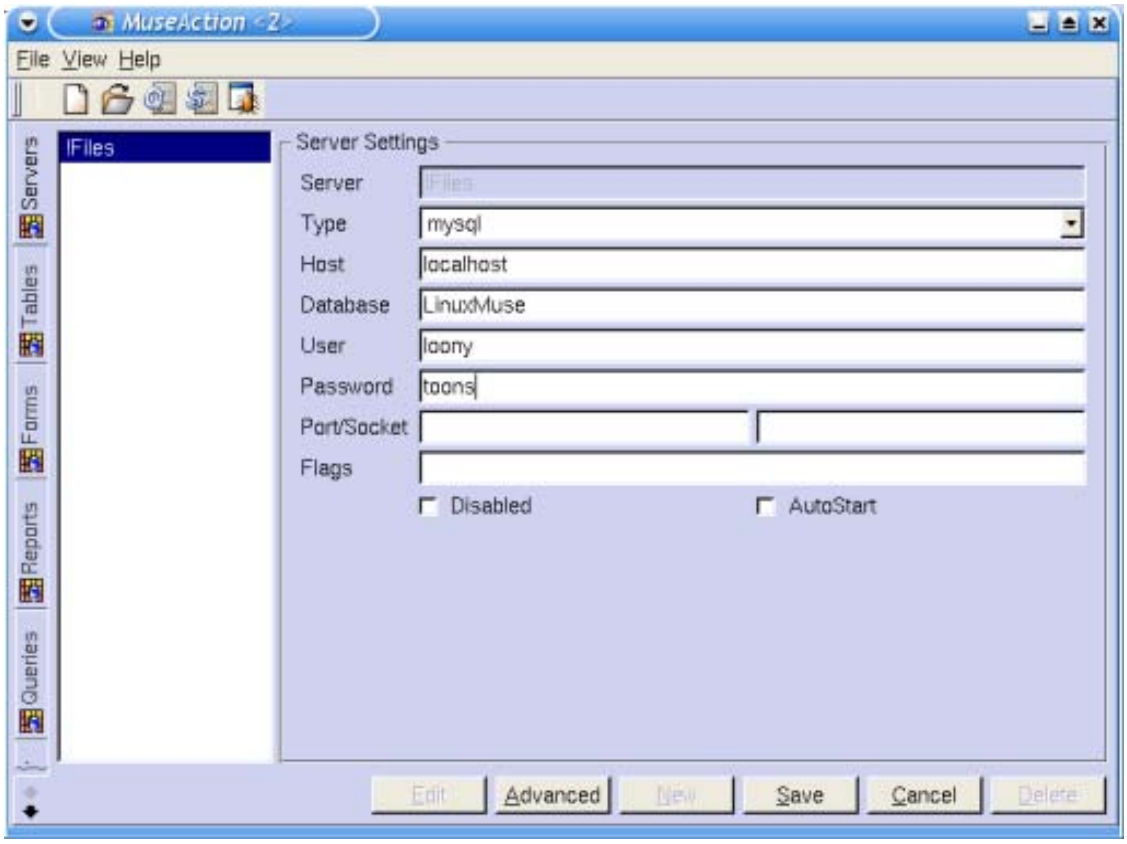

Fig. 46. Creación de una conexión a Base de datos en Rekall.

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

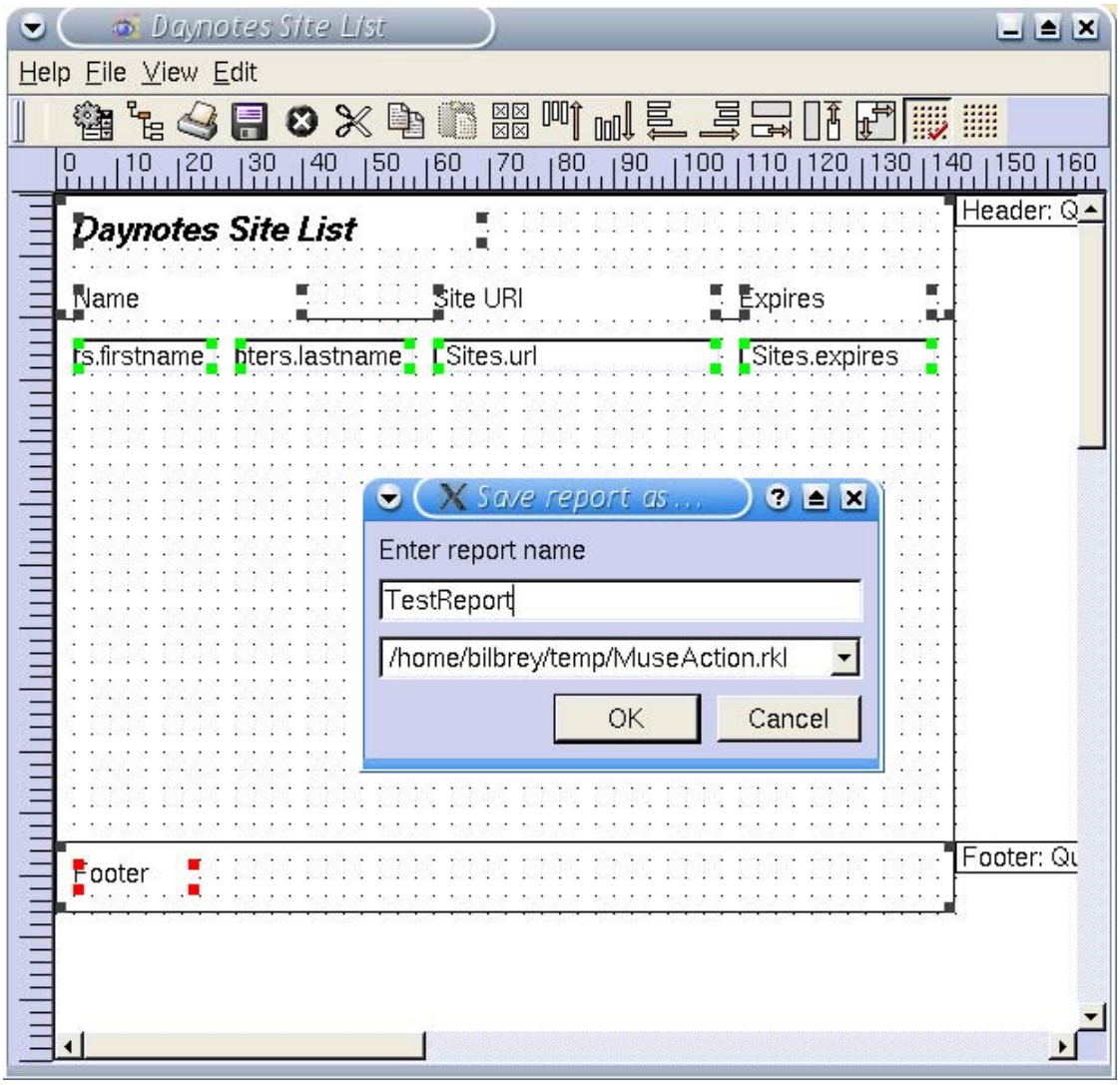

Fig. 47. Ambiente de diseño Rekall.

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

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

En estos momentos Rekall posee una licencia dual, manteniéndose por la comunidad la versión GPL y por sus propietarios la versión comercial, en este último caso se utiliza una "licencia de uso" que impide la distribución libre de modificaciones del código fuente. Cualquier

modificación del código fuente debe ser autorizada por theKompany e incorporada por ésta en el producto.

Fig. 48. Ejemplo de informe generado con Rekall.

29.2.7 JasperReports.

JasperReports(JASPERREPORTS) es una herramienta de fuente abierta para la generación de informes escrita en Java, que puede mostrar contenido de calidad en ficheros PDF, HTML, XLS, CSV (formatos de hoja de calculo) y XML. Puede usarse en aplicaciones embebidas en

Java, incluyendo J2EE (plataforma JAVA2 edición empresarial) o aplicaciones Web, para generar contenidos dinámicos.

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

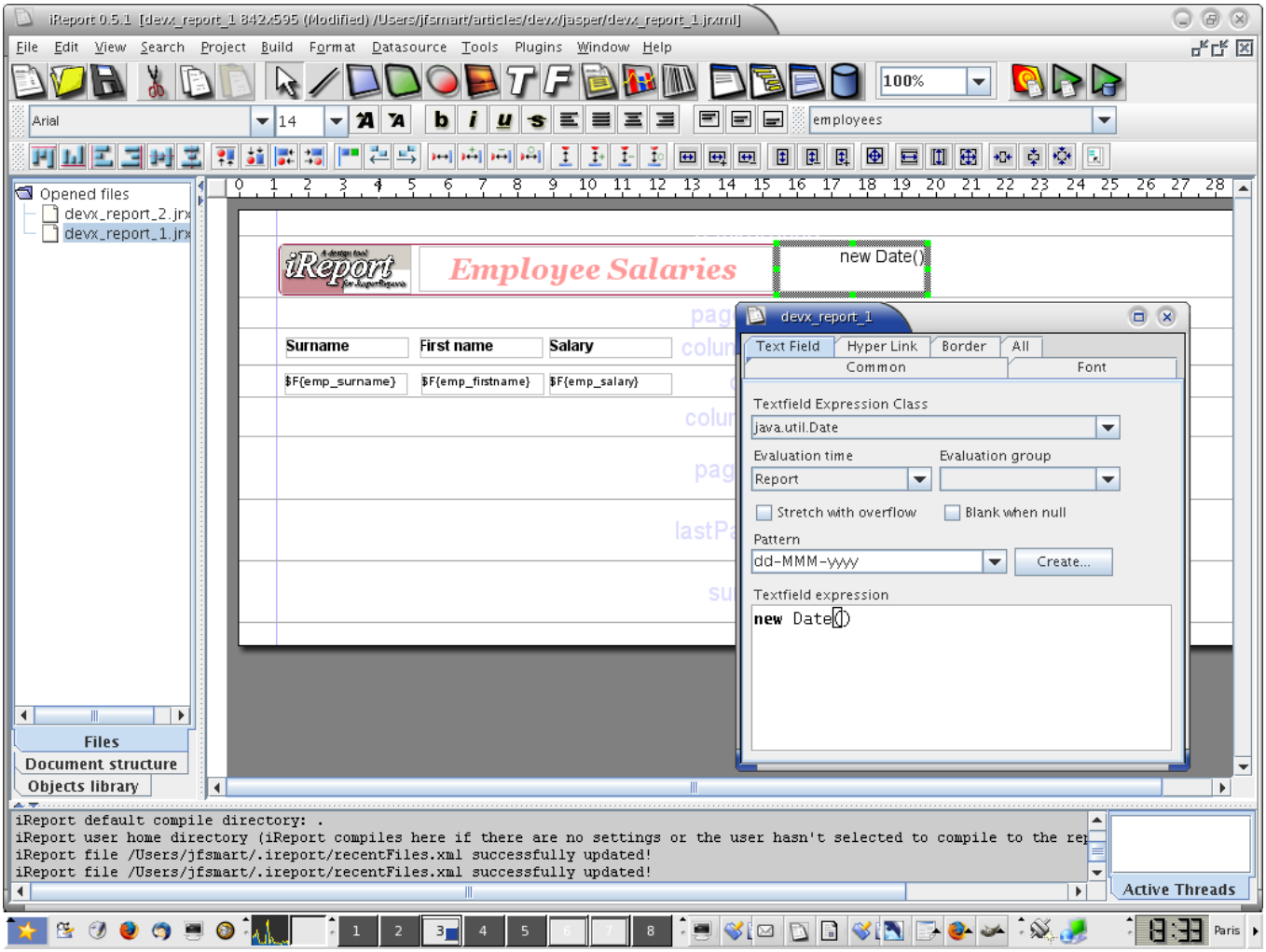

Fig. 49. Ventana de diseño de informe.

29.2.8 JfreeReport

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

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

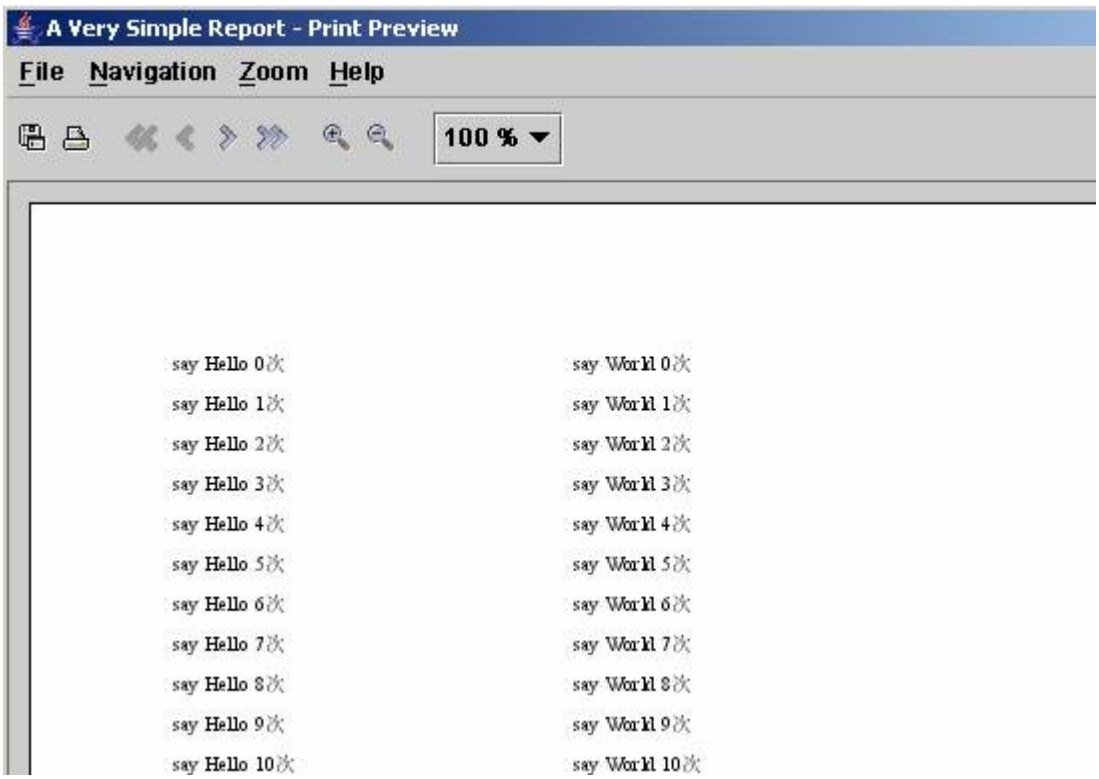

Fig. 50. Ejemplo de informe obtenido con JfreeReport.

29.2.9 DataVision

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

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

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

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

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

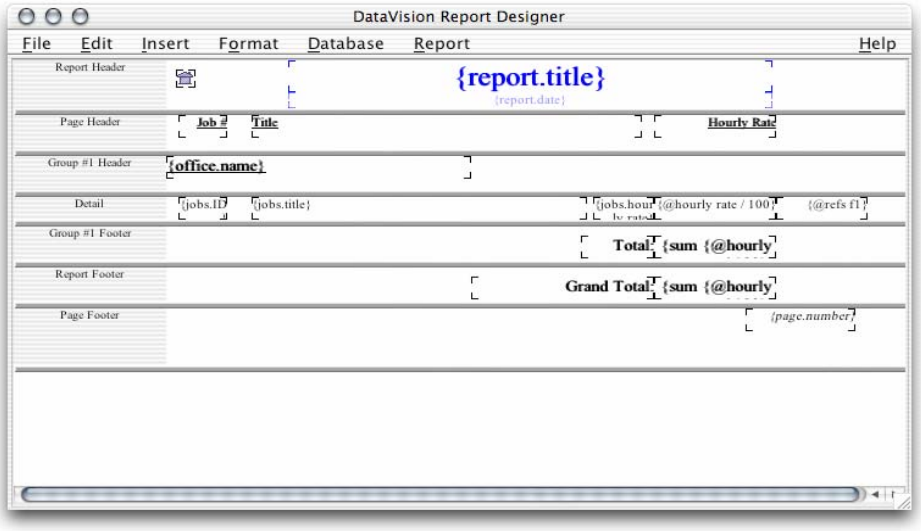

Fig. 51. Ventana de diseño de informe**.**

Como se puede apreciar en la figura, en la filosofía de DataVision, un informe tiene múltiples secciones. En la parte izquierda están los nombres de las secciones. Las largas áreas blancas son secciones que contienen campos del informe: columnas de la base de datos, campos de texto, fórmulas, agregaciones, y campos especiales como el título del informe o el número de la página.

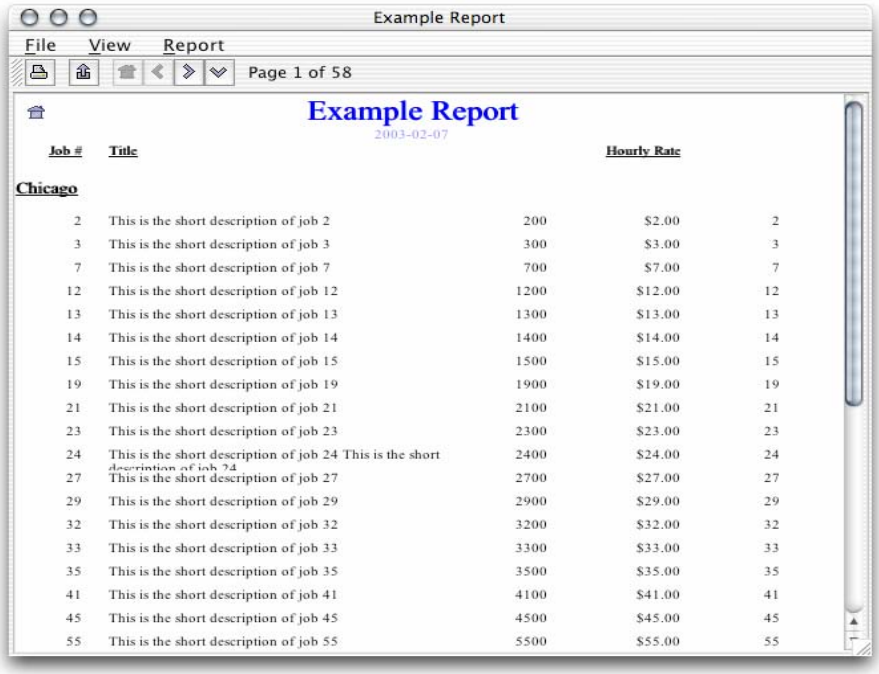

Fig. 52. Ejemplo de informe obtenido con DataVision**.**

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

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

- Corre donde quiera que Java corra: Microsoft Windows, GNU/Linux, Mac OS X, BSD, Solaris y otros.
- Trabaja con cualquier base de datos que tenga un manejador JDBC: MySQL, Oracle, PostgreSQL, Informix, hsqldb, Microsoft Acces, DB/2 y otras.
- Genera en el informe cabeceras y pies de páginas.
- Los informes tienen formatos: fuentes, colores, estilos de letras (negrita, itálica, subrayada), bordes de líneas, etc.
- Genera subinformes.
- Agregaciones (sum, min, max, count, average, stddev) por grupos y al finalizar el informe.
- Parámetros en tiempo de ejecución, le pregunta al usuario por los valores cuando el informe está corriendo; si está corriendo desde una línea de comando lee los valores de un fichero XML.
- Utiliza la cláusula SELECT de SQL.
- Oculta columnas y secciones enteras.
- Los informes pueden ser leídos de fuentes de datos. Actualmente, las fuentes de datos están definidas por base de datos y ficheros de textos.
- Permite imprimir informes desde la aplicación.
- Exporta los informes a HTML, XML, PDF, DocBook, Latex, etc.
- Configuración de la interfaz gráfica para diferentes idiomas.
- Informes almacenados en XML que pueden ser leídos fácilmente por las personas.
- DataVision es software libre.
- DataVision puede ser empotrado en otras aplicaciones.

29.2.10 OpenOffice 2 Writer

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

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

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

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

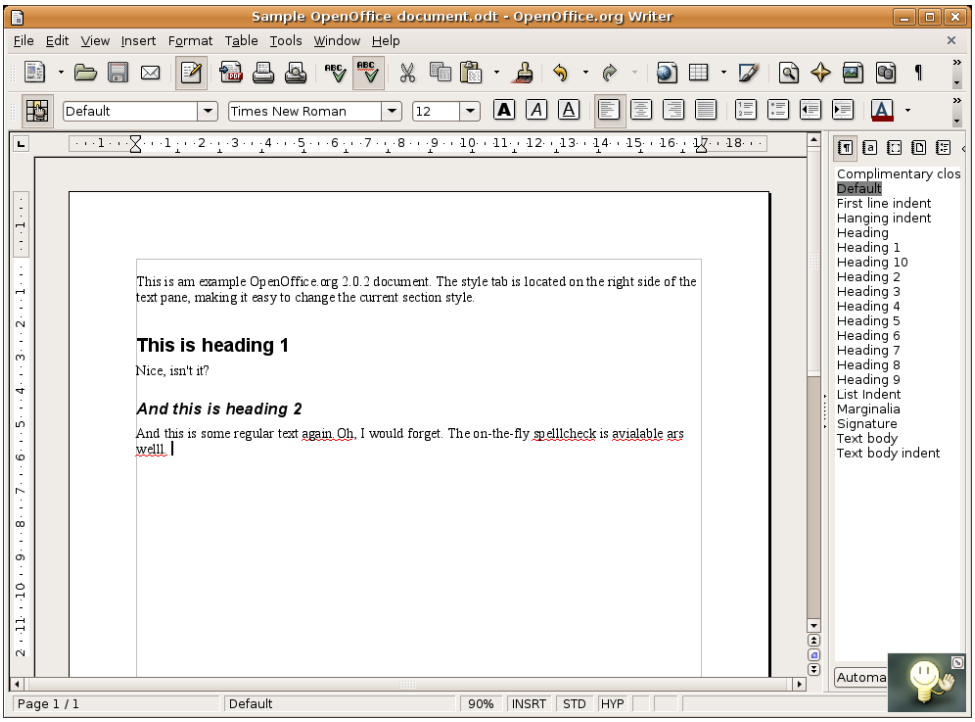

Fig. 53. Ventana de diseño de informe.

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

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

Glosario de términos.

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

Actor*:* Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.

API (Application Programming Interface - Interfaz para Programación de Aplicaciones): Conjunto de especificaciones de comunicación entre componentes software. Se trata del conjunto de llamadas que ofrecen acceso a los servicios del sistema desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general.

Artefactos*:* Una parte de la información que (1) es producida, modificada, o usada por un proceso, (2) define un área de responsabilidad, y (3) está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos. Una parte de la información que es usada o producida por un proceso de desarrollo.

Asistente (Wizard): Herramienta software que guía o asiste a un usuario en el uso de alguna funcionalidad de un sistema.

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

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

Casos de uso: Los casos de uso constituyen una forma de representación visual de las funcionalidades del sistema.

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

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

DocBook: Es un aplicación del estándar SGML/XML e incluye una DTD propia y que se utiliza de manera más destacada en el área de la documentación técnica, especialmente para documentar todo tipo de material y programas informáticos.

ERP: Sistema de Información General que integra y manejan muchas de las prácticas de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

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

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

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

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

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

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

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

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

JPEG (*J**oint*** ***P****hotographic* ***E****xperts* ***G****roup)*: Algoritmo diseñado para comprimir imágenes con 24 bits de profundidad o en escala de grises. JPEG es también el formato de fichero que

utiliza este algoritmo para comprimir imágenes. JPEG sólo trata imágenes fijas, pero existe un estándar relacionado llamado MPEG para videos. El formato de archivos JPEG se abrevia frecuentemente JPG debido a que algunos sistemas operativos sólo aceptan tres letras de extensión. JPEG es un algoritmo de compresión con pérdida. Esto significa que al descomprimir la imagen no obtenemos exactamente la misma imagen que teníamos antes de la compresión.

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

Kpart: Es el término con el que se designa a cualquier componente perteneciente a KDE.

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

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

MPL (Mozilla Public License): Es una licencia de código abierto y software libre. Fue desarrollado originalmente por Netscape Communications Corporation una división de la compañía 'America Online', y más tarde su control fue traspasado a la 'Fundación Mozilla'.

MPL (**Licencia Pública de Mozilla**): Es una licencia de código abierto y software libre. Fue desarrollada originalmente por Netscape Communications Corporation –una división de la compañía 'América Online'–, y más tarde su control fue traspasado a la 'Fundación Mozilla'.

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.

PDA (***P****ersonal* ***D****igital* ***A****ssistant*, Ayudante personal digital): Es una computadora de mano originalmente diseñada como agenda electrónica, en la actualidad se puede usar como una computadora doméstica.

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

Plataforma: Base, elemento de apoyo

PLC: Controlador Lógico Programable.

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

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

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

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

SISSL: Son las siglas de Sun Industry Standard Source License. Bajo esta licencia los desarrolladores pueden modificar y distribuir código fuente libremente.

SMS (***S**hort **M**essage **S**ervice*): Servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano.

SRS: Especificación de Requisitos Software

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

TIFF (***T**agged **I**mage **F**ile **F**ormat,* formato de archivo de imágenes con etiquetas): Formato de archivo de imágenes que contienen además de los datos de la imagen propiamente dicha, "etiquetas" en las que se archiva información sobre las características de la imagen, que sirve para su tratamiento posterior.

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

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

XSL (siglas de Extensible Stylesheet Language, expresión inglesa traducible como "lenguaje extensible de hojas de estilo"): Es una familia de lenguajes basados en el estándar XML

que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.