Universidad de las Ciencias Informáticas

Facultad 5

XStormServer, servidor para el cliente de visualización web del SCADA Guardián del ALBA

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

Autor: Julio César Moreno Cámbar.
Tutor: Ing. Ernesto Leyva Barrero.
Co-Tutor: Ing. Adisleidys Mirabal Pérez.

La Habana, junio 2012.

DECLARACIÓN DE AUTORÍA

Por este medio se declara que soy el único autor de este trabajo y se autoriza a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo. Para que así conste se firma la presente a los __ días del mes de _____ del 2012.

_____	_____
Firma del Autor	Firma del Tutor
(Julio César Moreno Cámbar)	(Ernesto Leyva Barrero)

DATOS DE CONTACTO

Ing. Ernesto Leyva Barrero.

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

e-mail: ebarrero@uci.cu

Ing. Adisleidys Mirabal Pérez.

Graduado de la especialidad de Ingeniero en Ciencias Informáticas desde hace un año, recién graduada en adiestramiento y con tres años vinculada al desarrollo de software.

e-mail: amirabal@uci.cu

Agradecimientos

A toda mi familia, en especial a mamá y abuelos que son mis otros padres, por todo su apoyo y amor infinito hacia mí.

A mi tutor y amigo, muy atento siempre, impulsando siempre mi formación como profesional.

A mi co-tutor por su gran ayuda, consejos y dedicación.

A todos mis compañeros del proyecto HMI, por brindarme su apoyo y contribuir de una forma u otra al desarrollo de esta investigación.

A todos mis amigos, que de cierta manera hemos compartido numerables momentos y recorrido el mismo camino en estos últimos años, por ofrecerme una de las cosas más hermosa que puede tener un ser humano.

Dedicatoria

A toda mi familia, a todos dedico mi trabajo de diploma y en especial a mi madre y mis abuelos queridos, por brindarme todo su apoyo, dedicación y amor, por ser una guía inmejorable durante toda mi vida, por quererlos tanto, para ustedes este trabajo.

Resumen

En los últimos años el crecimiento de las industrias y el avance adquirido en las Tecnologías de la Información y las Comunicaciones (TIC) han impulsado la extensión de las fábricas, maquinarias y redes lo que ha traído como consecuencia un aumento en la complejidad de los procesos industriales. Para lograr un control eficiente de estos procesos surgieron los sistemas de supervisión, control y adquisición de datos, conocidos como SCADA, que es un término tomado del Inglés, acrónimo de Supervisory Control and Data Adquisition. En la actualidad una de la tendencia de desarrollo de estos sistemas, se basa en el envío o publicación de cierta información que manejan en la web, debido a que la misma posee una gran extensibilidad en casi todo el mundo, comunicando una gran cantidad de personas.

Este trabajo surge a partir de la necesidad de enviar la información referente a los puntos y alarmas del proyecto SCADA Guardián del ALBA hacia un cliente de visualización web. Donde se refleja una investigación acerca de algunas de las tecnologías de comunicación actuales, orientándola hacia la automática. El objetivo de esta investigación es obtener un mecanismo de comunicación capaz de enviar la información referente a los puntos y alarmas del SCADA, así como autenticar los usuarios con sus respectivos privilegios, en dependencia de la petición que realice un cliente web. Para cumplir esta meta se describe una solución partiendo del diseño de software y se detalla la implementación del sistema.

Palabras Claves: Adquisición de datos, alarmas, autenticar, redes, mecanismo de comunicación, petición, puntos, SCADA, tecnologías de comunicación, web.

Índice

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han adquirido un gran avance y con ellas surgen nuevas soluciones eficientes para los problemas existentes en la sociedad, ejemplo de ello es la automatización de los procesos industriales que tienen entre sus objetivos ahorrar capital, tiempo y recursos; siendo esto sumamente útil en cualquier proyecto. Muchas empresas en el mundo han optado por esta solución y sin duda alguna, han alcanzado resultados que superan a los de épocas pasadas, donde no existía un gran desarrollo de esta técnica. La automática, ha revolucionado las producciones continuas a gran escala y con ella los ordenadores y programas para controlar todos estos procesos.

En Cuba desde hace algunos años se trabaja en la automatización de los procesos de diferentes industrias, con el objetivo de mejorar la productividad y la calidad de los productos. La Universidad de las Ciencias Informáticas cuenta hoy con varios centros de desarrollo entre los que se encuentra el Centro de informática Industrial que tiene algunos proyectos de desarrollo de software, que persiguen como objetivo fundamental la realización de sistemas para la automatización de los servicios de numerosas empresas que necesitan de la supervisión y control de sus procesos. Tal es el caso del proyecto SCADA Guardián del ALBA (SCADA-GALBA) que se desarrolla para la empresa Petróleos de Venezuela.

Este proyecto cuenta con mecanismos para visualizar toda una serie de pantallas, que se configuran de acuerdo con las necesidades del ambiente de producción que se desea automatizar. Estas pantallas contienen información importante, pues manejan valores e históricos de numerosas variables, entiéndase presión, temperatura, nivel de líquido, flujo, estado de bombas y válvulas, entre otras, además el sistema es capaz de visualizar las distintas alarmas asociadas al cambio de estado o de valor de las variables de interés del proceso, como puede ser un alto nivel de crudo en un tanque.

La aplicación encargada de visualizar todo este comportamiento en el SCADA-GALBA, es el ambiente de ejecución perteneciente al subsistema de Interfaz Hombre Máquina. La misma depende de su previa instalación en las consolas o computadoras clientes, las cuales deben tener instalado el Sistema Operativo Debian en su versión 6.0. Esta situación dificulta el trabajo de los directivos de la empresa así como los mantenedores del sistema, los cuales necesitan tener un tipo de acceso especial a algunas de las pantallas o despliegues de interés sin necesidad de instalar la solución íntegramente. La forma más

eficaz y más utilizada en la actualidad para soluciones de este tipo, que necesiten visualización mediante estaciones remotas, es el acceso web mediante los distintos navegadores. De esta forma las aplicaciones que permiten el acceso web a la información que manejan, pueden estar en un servidor y desde cualquier estación, con un navegador web y conexión a la red del servidor, con los debidos niveles de privilegios y autenticación, se podrá acceder a la información deseada, sin importar la plataforma que se use.

Considerando la situación antes planteada se define como **problema a resolver** en esta investigación**:** ¿Cómo proveer a un cliente de visualización web los datos necesarios para mostrar los despliegues del SCADA-GALBA?

Por lo que el **objeto de estudio** es: la comunicación entre un software de aplicación y un cliente web, teniendo como **campo de acción** la comunicación entre el SCADA-GALBA y un cliente web.

Definiendo como **objetivo general** de la investigación: Desarrollar un mecanismo que permita la adquisición de información desde la capa de comunicaciones del SCADA-GALBA, para satisfacer las peticiones de un cliente web.

La **idea a defender** con este trabajo es que, el proyecto SCADA-GALBA, al contar con un mecanismo de comunicación con un cliente web, brindará una solución que permitirá enviar la información necesaria a estaciones remotas, sin importar la plataforma de las mismas.

El desarrollo de esta investigación estará guiado por las siguientes tareas:

- Estudio del estado del arte de las tecnologías para la comunicación de aplicaciones web con los sistemas SCADA y caracterización de sus antecedentes.
- Selección de una o varias tecnologías de comunicación robusta y eficiente para su uso.
- Implementación de un mecanismo de comunicación con un cliente web de visualización de despliegues para el SCADA.
- Selección y aplicación de pruebas al software para comprobar sus funcionalidades.

Durante el desarrollo de esta investigación se usarán varios **Métodos Teóricos** y **Empíricos** para profundizar en algunos temas y dar cumplimiento a las tareas antes mencionadas:

Se usarán **Métodos Teóricos** para poseer los conocimientos necesarios sobre el estado del arte de los procesos que se quieren automatizar y su relación con otros procesos.

- **Método histórico-lógico:** Para la comprensión de los antecedentes y las tendencias actuales de los sistemas SCADA.

- **Método analítico-sintético:** Para el estudio de los conceptos empleados en los sistemas SCADA, analizando los documentos elaborados por desarrolladores, para la extracción de los elementos más importantes.

Los **Métodos Empíricos** por otra parte permiten extraer de los procesos analizados las informaciones que se necesitan sobre ellos a través de observaciones y del uso de otras técnicas de recopilación de la información.

- **Experimento:** Para la elaboración de un mecanismo de comunicación con un cliente web para el SCADA.

- **Criterio de expertos:** Para la obtención del conocimiento necesario referente al tema y continuar con la investigación, mediante encuestas o entrevistas, cuyas experiencias y opiniones pueden ser de una valiosa contribución para el desarrollo del trabajo.

Al terminar esta investigación, se espera tener una aplicación capaz de satisfacer las peticiones que realice un cliente de visualización web, enviando la información correspondiente, luego de realizarse la adquisición desde la capa de comunicación del SCADA-GALBA y que pueda ser utilizada en las instalaciones de PDVSA, donde se requiera la supervisión de los procesos mediante el uso de un cliente de visualización web.

Este trabajo de diploma contiene los siguientes capítulos:

- **Capítulo 1 Fundamentación Teórica:** En este capítulo se hace referencia a los sistemas SCADA, así como el estado del arte de las tecnologías actuales de comunicación.

- **Capítulo 2 Construcción de la solución propuesta:** En este capítulo se definen las funcionalidades y se construye la solución propuesta.

- **Capítulo 3 Pruebas al software:** En este capítulo se realizan las pruebas al software con el objetivo de comprobar sus funcionalidades.

Capítulo 1: Fundamentación teórica

Introducción

El siguiente capítulo hace referencia a las características generales de los sistemas SCADA, reflejándose sus principales funciones y los módulos que los componen. Además se realiza un estudio de las tecnologías, herramientas y tendencias más utilizadas actualmente en el desarrollo de software.

1.1 Fundamento de los sistemas SCADA

Normalmente al hablar de SCADA se hace referencia a un sistema central utilizado para controlar un sitio completo o cuya extensión abarca una gran distancia. Para instalar un sistema SCADA se necesita un hardware de señal de entrada y salida, sensores y actuadores, controladores, HMI (Human-Machine Interface o Interfaz Hombre-Máquina), redes, comunicaciones, bases de datos, entre otros elementos. Estos sistemas contribuyen a aumentar la eficiencia durante el proceso de monitoreo y control proporcionando la información necesaria en el momento oportuno para poder tomar decisiones apropiadas respecto a cada operación [1]. Así mismo al disponer de la 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 (Planificación de Recursos Empresariales o Enterprise Resource Planning), entre otras.

1.2 Flujo de la información en los sistemas SCADA

Los sistemas SCADA maneja un gran número de variables por lo que la naturaleza de las mismas, dependiendo del proceso y el modelo seleccionado para su supervisión, puede ser muy diversa: presión, temperatura, flujo de potencia, intensidad de corriente, voltaje, entre otros. Todos esos parámetros deben transformarse en variables que un sistema SCADA pueda reconocer, es decir, en una variable eléctrica. Es necesario convertir esta señal en un valor digital equivalente en el bloque de conversión de datos. Generalmente, esta función es llevada a cabo por un circuito de conversión analógico/digital. La computadora almacena esta información, la cual es utilizada para su análisis y para la toma de decisiones. Simultáneamente, se muestra la información al usuario del sistema en tiempo real. Basado en la información recibida, el operador puede tomar la decisión de realizar una acción de control sobre el

proceso. El operador ordena al computador a ejecutarla y nuevamente debe convertirse la información digital a una señal eléctrica [2].

1.3 Módulos de los sistemas SCADA

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

- **Base de Datos Históricos**
  Implementa los mecanismos para almacenar la información recibida desde los dispositivos de campo, así como las alarmas y eventos generados. La información almacenada es utilizada por las diferentes aplicaciones del sistema.

- **Middleware**
  Constituye la capa de software encargada de la comunicación entre los distintos procesos distribuidos, de mediano y alto nivel.

- **Seguridad**
  Brinda las interfaces necesarias para que los usuarios se autentiquen en el sistema, y accedan únicamente a los recursos que tengan asignados según su rol. Incluye herramientas para la protección ante ataques, fallos eléctricos, problemas de red, entre otros.

- **Reportes**
  Brinda al usuario, la información referente al comportamiento de las variables a través de reportes especializados, los cuales se diseñan, teniendo en cuenta la necesidad de los usuarios, gracias a un editor que posee este módulo.

- **HMI**
  Permite representar gráficamente, de forma segura los procesos operacionales con los que los usuarios interactúan. A través del mismo se pueden visualizar condiciones operacionales, valores de variables, visualizar alarmas, navegar entre despliegues, enviar comandos, entre otros.

- **Núcleo de procesamiento de datos (Recolector)**

El núcleo de procesamiento de datos es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.

- **Manejadores**

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

- **Configuración**

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

1.4 El módulo Interfaz Hombre-Máquina

Esta interfaz, muestra los datos de uno o varios procesos a un operador (humano) y además permite al operario, ejercer el control sobre los mismos. Los HMI surgen esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, PLC (Controlador lógico programable en inglés *Programmable Logic Controller*) y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA lo hacen de manera automática. Históricamente los PLC no tienen un modo estándar de presentar la información al operador. La obtención de los datos por el sistema SCADA parte desde el PLC o desde otros controladores y se realiza por medio de algún tipo de red, a continuación esta información es combinada y formateada. Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de problemas.

Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLC, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfaces por sí mismos, sin la necesidad de un programa hecho a medida escrito por un desarrollador de software [3].

Aplicaciones que componen un HMI

- **Ambiente de configuración**

Permite configurar varios procesos o parte de ellos, posibilitando al usuario definir el entorno de trabajo de su aplicación, según la disposición de pantallas requeridas y los niveles de acceso para los distintos usuarios. Dentro del módulo de configuración el usuario define las pantallas gráficas o de texto que va a utilizar, se seleccionan los drivers de comunicación que permitirán el enlace con los elementos de campo y la conexión o no en red de estos últimos. Estas configuraciones son las que más tarde utilizará el visualizador para poner en ejecución la representación gráfica de la planta, fábrica o lugar donde ha sido instalado.

- **Ambiente de ejecución**

Proporciona al operador las funciones de control y supervisión de la planta. El proceso a supervisar se representa mediante sinópticos gráficos que cambian dinámicamente a diferentes formas y colores, según los valores leídos en la planta o en respuesta a las acciones del operador.

Las funcionalidades principales de este ambiente son:

- **Visualización de despliegue:** Es el encargado de mostrar de forma gráfica los procesos que se siguen en una planta. Esto se realiza a través de objetos gráficos que suelen dividirse en dos grupos, simples y complejos, los primeros se caracterizan por ser objetos estáticos, como circunferencias, líneas, imágenes, y los complejos se caracterizan por variar su estado en el transcurso del tiempo según los datos leídos en la planta, entre los principales se encuentran: las gráficas de tendencia, los relojes y las reglas.
- **Visualización de alarmas:** Es una interfaz gráfica que concentra las condiciones de procesos críticas, medias y bajas presentes en el sistema, cuyo objetivo primordial es guiar al operador a

la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual.

- **Visualización de puntos:** Es la interfaz gráfica encargada de visualizar los detalles de un punto, tales como: nombre, calidad, valor, fecha, entre otros valores.
- **Visualización de eventos:** Muestra las acciones que se han realizado en el sistema.

1.5 Tendencias y tecnologías actuales a considerar

En la actualidad existen y continúan surgiendo nuevas tendencias y tecnologías que, en la medida de lo posible, nos proporcionan una mejor forma para el desarrollo de aplicaciones. Una tendencia muy notable en Cuba es la migración al uso del software libre, la nación hace grandes esfuerzos por obtener la soberanía tecnológica; desarrollando nuevos sistemas sobre licencias no propietarias. En cuanto a las tecnologías de comunicación se observa un avance constante, con un incremento en la cantidad de personas con acceso a ellas y una evolución exponencial desde el punto de vista técnico. Hoy en día las vías de comunicación son más rápidas y variadas, interconectando al mundo entero, sin importar la distancia entre el emisor y el receptor. En este epígrafe se hace un estudio de algunas de estas tendencias y tecnologías y se analizan sus características más relevantes.

1.5.1 Software Libre

El software libre es la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Precisamente, significa que los usuarios de programas tienen las siguientes libertades esenciales [4].

- Ejecutar el programa, para cualquier propósito.
- Estudiar cómo trabaja el programa, y cambiarlo para que haga lo que se quiera. El acceso al código fuente es una condición necesaria para ello.
- Redistribuir copias para que pueda ayudar al prójimo.
- Distribuir copias de versiones modificadas a terceros, dándole a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

Sin embargo, ciertos tipos de reglas sobre la manera de distribuir software libre son aceptables, cuando no entran en conflicto con las libertades principales. Por ejemplo, el *copyleft* (definido muy resumidamente) es la regla en base a la cual, cuando redistribuye el programa, no puede agregar restricciones para denegar a las demás personas las libertades principales. Esta regla no entra en conflicto con las libertades principales; más bien las protege. Software libre no significa que no sea comercial. Un programa libre debe estar disponible para el uso comercial, la programación comercial y la distribución comercial. La programación comercial de software libre ya no es inusual; tal software libre comercial es muy importante. Puede haber pagado dinero para obtener copias de software libre, o puede haber obtenido copias sin costo. Pero sin tener en cuenta cómo obtuvo sus copias, siempre tiene la libertad de copiar y modificar el software, incluso de vender copias. Cuando se habla de software libre, es mejor evitar usar términos como regalar o gratuito, porque dichos términos implican que el asunto pasa por el precio, no la libertad [4].

1.5.2 Servidor Web

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

- Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
- Recibe una petición.
- Busca el recurso.
- Envía el recurso utilizando la misma conexión por la que recibió petición.
- Vuelve al primer punto.

Un servidor web que siga el esquema anterior cumplirá todos los requisitos básicos de los servidores HTTP, aunque sólo podrá servir ficheros estáticos. A partir del esquema anterior se han diseñado y desarrollado todos los servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGIs, Servlets, etc.) que pueden atender, en función de que sean o no multi-proceso o multi-hilados [2].

Características básicas de los servidores web:

- Servicio de ficheros estáticos.
- Seguridad y autenticación.
- Contenido dinámico.
- Servidores virtuales.
- Prestaciones extra.
- Actuación como representantes.
- Protocolos adicionales.

1.5.3 Evaluación de tecnologías de comunicación

En este epígrafe se analizan algunas de las tecnologías de comunicación, valorando algunas de sus características más importantes, con el objetivo de seleccionar la más conforme para la construcción de la solución.

XMPP

El protocolo extensible de mensajería y comunicación de presencia (Extensible Messaging and Presence Protocol, más conocido como XMPP), es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea, con el cual queda establecida una plataforma para el intercambio de datos XML que puede ser usada en aplicaciones de mensajería instantánea. Las características en cuanto a adaptabilidad y sencillez del XML son heredadas de este modo por el protocolo XMPP, se encuentra documentado y se insta a utilizarlo en cualquier proyecto. Existen servidores y clientes libres que pueden ser usados sin coste alguno [5].

Características del XMPP:

- **Descentralizado:** Arquitectura de las redes XMPP similar a la del correo electrónico; cualquiera puede poner en marcha su propio servidor XMPP, sin que haya ningún servidor central.
- **Abierto:** La IETF[1] ha formalizado el protocolo XMPP como una tecnología de mensajería instantánea estándar. El desarrollo de esta tecnología no está ligado a ninguna empresa en concreto y no requiere el pago de regalías.

[1] Internet Engineering Task Force (IETF) (en español Grupo Especial sobre Ingeniería de Internet), organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet.

- **Seguro:** Los servidores XMPP pueden estar aislados de la red pública XMPP, y poseen robustos sistemas de seguridad como SASL (Capa de Seguridad y Autenticación Simple, en inglés Simple Authentication and Security Layer) y TLS (Seguridad en la Capa de Transporte, en inglés Transport Layer Security). Para apoyar la utilización de los sistemas de cifrado, la XMPP Standards Foundation pone a disposición de los administradores de servidores XMPP autoridad de certificación en xmpp.net ofreciendo certificados digitales gratis.
- **Flexible:** Se pueden hacer funcionalidades a medida sobre XMPP; para mantener la interoperabilidad, las extensiones más comunes son gestionadas por la XMPP Software Foundation.

Server-Sent Events

Eventos enviados del servidor (SSE) es un estándar que describe cómo los servidores pueden iniciar la transmisión de datos hacia el cliente una vez que la conexión inicial del cliente ha sido establecida. Se usa comúnmente para enviar actualizaciones de mensajes o flujos de datos continuos a un cliente navegador y diseñado para mejorar, *cross-browser* (capacidad de un sitio web o aplicaciones web, de construir HTML o secuencia de comandos del lado del cliente para apoyar a todos los navegadores web) de *streaming* (distribución de multimedia a través de una red de computadoras de manera que el usuario consume el producto al mismo tiempo que se descarga) a través de una API de JavaScript llamada EventSource, a través del cual un cliente solicita una dirección URL concreta para recibir un flujo de eventos [6].

WebSocket

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. La API de WebSocket está siendo normalizada por el W3C[2], y el protocolo WebSocket, a su vez, está siendo normalizado por el IETF. Como las conexiones TCP ordinarias sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones brindando una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexando diferentes servicios WebSocket sobre un único puerto TCP (a costa de una pequeña sobrecarga del protocolo) [7].

[2] World Wide Web Consortium, abreviado W3C, consorcio internacional que produce recomendaciones para la World Wide Web.

Ventajas:

- Se abandonan las técnicas de consulta constante.
- Reducción en el consumo de red de una aplicación tanto del lado del cliente como del servidor.
- Se tienen un marco estandarizado para el desarrollo de bases de datos del lado del cliente, comunicación bidireccional con websocket, hilos del lado del cliente.
- Estándar abierto y respaldado por la W3C y la IETF.

Desventajas:

- Aún está en desarrollo y no todos los navegadores lo soportan.
- Todavía se están analizando las posibles vulnerabilidades y problemas de seguridad que podría llegar a tener.

A estas tecnologías se le aplicaron algunas pruebas, teniendo en cuenta las características más importantes que debe de tener la aplicación, las mismas son: tiempo real, seguridad, sentido de la comunicación y concurrencia, subrayando el tiempo real, cabe destacar que se hace alusión al tiempo real de un software, es decir la rapidez con que se realice una determinada acción.

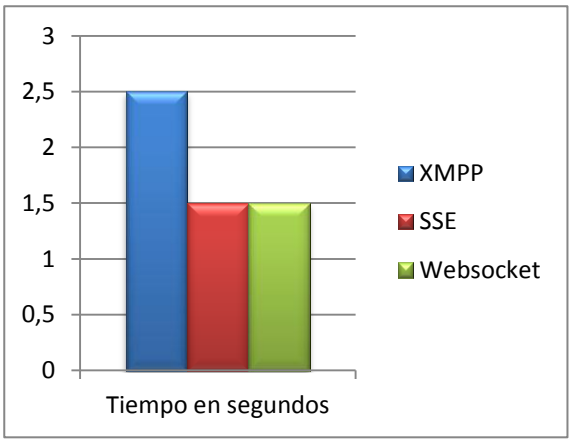
El experimento consistió en el envío de 150 puntos; cada X segundo y analizar la capacidad de enviar la cantidad de datos requeridos en el menor tiempo, basándose en el segundo donde desaparecía la latencia (suma de retardos temporales dentro de una red, un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.). A continuación se muestra una tabla con los resultados obtenidos para el protocolo XMPP.

Tabla 1. Resultados del experimento realizado al protocolo XMPP.

Protocolo	Cantidad de Puntos	Tiempo en segundos	Presencia de latencia
XMPP	150	1	SI
XMPP	150	1.5	SI
XMPP	150	2	SI
XMPP	150	2.5	NO

Como se muestra en la tabla anterior la latencia para el protocolo XMPP, desaparece al enviar los 150 puntos cada 2.5 segundos, lo que significa que es el menor tiempo en el que el protocolo XMPP puede enviar esa secuencia de datos sin que exista pérdida o retardo en el arribo de los datos enviados. Este experimento fue realizado para los restantes protocolos. A continuación se muestra una figura donde se puede apreciar el tiempo en el cual desaparece la latencia para cada protocolo.

Figura 1. Comparaciones de capacidad de enviar la cantidad de datos requeridos en el menor tiempo.



Con el resultado de este experimento se puede observar que en la rapidez con que se envían los datos, los protocolos que más se destacan son SSE y WebSocket, XMPP es débil en ese aspecto. En cuanto a la seguridad podemos decir que el protocolo más robusto es XMPP el mismo posee sistemas de seguridad como SASL y TLS. En cuanto al sentido de las comunicaciones se descarta SSE dado que el mismo solo permite conexiones en un solo sentido, del servidor al cliente. Todas estas técnicas y protocolos permiten gran concurrencia.

Se llegó a la conclusión de que el protocolo WebSocket es el que más se ajusta para el desarrollo de la solución, porque soporta el envío de grandes cantidades de datos en el menor tiempo posible, múltiples conexiones, presenta una comunicación bidireccional y posee una seguridad aceptable.

1.6 Bibliotecas

Existen varias bibliotecas que implementan tecnologías de comunicación, por lo que fue necesario usar o analizar algunas de ellas para la realización de los experimentos del epígrafe anterior, con el fin de

escoger la más idónea para la construcción de la solución.

1.6.1 Qxmpp

QXmpp es una biblioteca multi-plataforma en C++ del cliente XMPP. Se basa en Qt y C++. QXmpp es bastante intuitivo y fácil de usar. Se utiliza Qt ampliamente y es la única biblioteca de terceros de la que depende. Los usuarios necesitan tener un conocimiento acerca del trabajo con C + + y Qt básico (Signals, Slots y tipos de datos de Qt).

1.6.2 Libwebsocket

Libwebsocket es una biblioteca que implementa la tecnología Websocket, escrita en C y bajo la licencia LGPL, implementa una interfaz (API) teniendo en cuenta las diferencias de algunos de los lenguajes de programación, con el fin de mantener la compatibilidad con otras implementaciones y simplificar el desarrollo. Esta biblioteca permite instanciar objetos del tipo cliente y servidor websocket y además permite manejar socket TCP y establecer una comunicación bidireccional, de una forma fácil.

Finalmente se decidió usar la biblioteca Libwebsocket, pues implementa la tecnología Websocket, la cual quedó demostrada, en el experimento realizado, que es la que más se ajusta, en cuanto a las características más críticas que debe tener la solución.

1.7 Ambiente de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o Integrated Development Environment (IDE, por sus siglas en inglés), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. 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. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Los IDEs proveen un marco de trabajo para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic. Otra característica importante para un IDE es que puede funcionar como un sistema en tiempo de ejecución en algunos lenguajes de programación. Actualmente existen varios IDEs que ofrecen un gran número de funcionalidades, facilitando en gran medida el trabajo del programador, aportando algunas ventajas como la facilidad de sugerir y luego autocompletar el código, se integran con sistema de control de versiones (SVN) por lo cual no hay que usarlo aparte, entre otras [8].

1.7.1 Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma desarrollado inicialmente por la IBM y en la actualidad por la Fundación Eclipse creado para el desarrollo de lo que el proyecto llama" Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. La definición que da el proyecto Eclipse acerca de su software es: "*una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular*". Este programa 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. El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad. Este mecanismo de módulos es una plataforma ligera para componentes de *software*. Por otra parte, permite trabajar con lenguajes para procesamiento 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 [9].

Ventajas:

- Eclipse es adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, PHP, entre otros.
- La característica clave de Eclipse es la extensibilidad.
- Compilación en tiempo real.
- La depuración e implementación de aplicaciones resultan mucho más sencillas.
- Es multiplataforma.

1.8 Lenguajes de programación

Actualmente existe una gran variedad de lenguajes de programación, y debido a esto existen muchas soluciones, aplicaciones o programas, implementados en distintos lenguajes, por lo que es necesario estudiarlos o dominarlos para comprender su funcionamiento y poder construir una solución.

1.8.1 C

Creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos [10].

Las principales características del lenguaje C son:

- Tiene un conjunto completo de instrucciones de control.
- Permite la agrupación de instrucciones.
- Incluye el concepto de puntero (variable que contiene la dirección de otra variable).
- Los argumentos de las funciones se transfieren por su valor. Por ello, cualquier cambio en el valor de un parámetro dentro de una función no afecta al valor de la variable fuera de ella.
- La E/S no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones.
- Permite la separación de un programa en módulos que admiten compilación independiente.

1.8.2 C++

C++ es un lenguaje imperativo orientado a objetos derivado del C, en realidad un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía, diseñado a mediados de los años 1980, por Bjarne Stroustrup. 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. 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 trae menos automatismos (obliga a hacerlo casi todo manualmente al igual que C) lo que dificulta mucho su aprendizaje.

Algunas de sus características distintivas clave incluyen:

- Muy claro, sintaxis legible.
- La orientación a objetos intuitiva.
- Expresión natural del código de procedimiento.
- La modularidad completa, compatible con paquetes jerárquicos.
- Excepción basada en el manejo de errores.
- Muy altas a nivel de los tipos de datos dinámicos.
- Extensas librerías estándar y módulos de terceros para prácticamente todas las tareas.
- Extensiones y módulos fácilmente escritos en C, C + + (o Java para Jython, o. NET para IronPython).
- Integrable dentro de las aplicaciones como una interfaz de scripting.
- Poderoso y rápido.
- Multiplataforma (Linux / Unix, Windows, OS / 2, Mac, Amiga, entre otros).
- Fácil aprendizaje.
- Libre uso y distribuibles, incluso para uso comercial.

Es necesario usar el lenguaje C pues la biblioteca Libwebsocket, una de las implementaciones de la tecnología Websocket está escrita en este lenguaje, por otra parte C++, para el trabajo con el Framework Qt. Finalmente se llegó a la conclusión que se usarán los lenguajes C/C++.

1.9 Framework Qt

Qt es una biblioteca multiplataforma para el desarrollo de interfaces gráficas, producida por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008. Es utilizado en aplicaciones como KDE, Google Earth, Skype, Adobe PhotoshopAlbum, VirtualBox y Opie. Qt está desarrollado de forma nativa en C++, pero ha sido portado para ser utilizado desde varios lenguajes de programación como Java, Python, C#, entre otros y está disponible para varios sistemas operativos entre ellos: Windows, Linux y Mac OSx, teniendo un amplio respaldo en el sector empresarial y en las comunidades de desarrollo. Incluye un amplio conjunto de widgets que proporcionan las funcionalidades estándar de interfaz gráfica de usuario, introduce una innovadora alternativa para la comunicación entre objetos, llamados "signals y slots".

Puede soportar toda la funcionalidad de interfaz de usuario que requieren las aplicaciones modernas, tales como menús, menús contextuales, arrastrar y soltar, y barras de herramientas acoplables. También propone el patrón de diseño model/view (modelo/vista) para la representación gráficas de los datos con la separación de las funcionalidades introducidas por esta arquitectura, ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de elementos y proporciona una interfaz de modelo estándar que permitir una amplia gama de fuentes de datos. Hay que destacar que comenzó como un framework para el desarrollo de interfaces gráficas, pero ya el API de la biblioteca cuenta con tecnologías que facilitan el trabajo a los desarrolladores como son:

- Máquina de estados.
- Soporte para hilos.
- Sistema de pintado.
- Animaciones y multimedia.
- Vista para gráficos 2D.
- Vista para gráficos 3D.
- Integración de documentación en aplicaciones.
- Modelos dinámicos de objetos.
- Pruebas de unidad.
- Serialización y trabajo con DOM.

- Trabajo con extensiones.
- Contenedores genéricos.
- Estilos.
- Eventos y filtrado de eventos.

Además permite crear aplicaciones de bases de datos independientes de la plataforma que se utiliza. Incluye drivers nativos para Oracle, Microsoft SQL Server, SybaseAdaptive Server, IBM DB2, PostgreSQL TM, MySQL, BorlandInterbase, SQLite, y bases de datos compatibles con ODBC. Incluye widget personalizado para la representación de los datos de las bases de datos. Otras características son: la disponibilidad de código fuente, la excelente documentación organizada en un asistente (QtAssitant), y un editor para el diseño de formularios visualmente (QtDesigner). Qt se distribuye bajo un sistema de licenciamiento dual, que incluye una licencia comercial y una de código abierto bajo los términos de GNU Lesser General Public License (LGPL) [11].

1.10 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. Constituyen un soporte automatizado para el desarrollo y mantenimiento del software. Se pueden ver como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Beneficios de estas herramientas en todas las etapas del proceso de desarrollo de software:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código.

1.10.1 Visual Paradigm

Visual Paradigm para UML es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Los

sistemas de modelado UML ayudan a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código a partir de diagramas, generación de objetos a partir de bases de datos, generación de bases de datos a partir de diagramas de entidad relación y generar documentación, posee licencia gratuita y comercial , permite interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse). Posee una plataforma, llamada SDE, capaz de integrarse con Eclipse, NetBeans, Oracle JDeveloper, JBuilder, IntelliJ IDEA, WebLogicWorkshop, Microsoft Visual Studio, entre otras [12].

1.11 Metodología de Desarrollo de Software

Una Metodologías de Desarrollo de Software es una filosofía, herramientas, modelos y métodos para asistir al proceso de desarrollo de software. Hasta hace poco este proceso de desarrollo llevaba asociada un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema tradicional, ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos). Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del “buen hacer” de la ingeniería del software, asumiendo el riesgo que ello conlleva. En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, con plazos reducidos, requisitos volátiles, y/o basados en nuevas tecnologías [13].

1.11.1 Programación extrema (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Las características fundamentales de esta metodología ágil son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas.
- Programación en parejas.
- Frecuente integración del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código.
- Propiedad del código compartida.
- Simplicidad en el código.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hace. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores [14].

Las Historias de Usuario

Las historias de usuario son la técnica utilizada en XP para especificar las necesidades del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

Respecto a la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración [13].

Roles de XP

Para una mejor organización de la ejecución de las actividades para el desarrollo de proyectos guiados por esta metodología se establecen los siguientes roles: Programador, Cliente, Encargado de pruebas, Encargado de seguimiento, Entrenador, Consultor, Gestor [13].

Proceso XP

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la

habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al primer paso [13].

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. Se debe presionar al programador a realizar más trabajo que el estimado, ya que, de no ser así, perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de 6 fases: Exploración, Planificación de entrega, Iteraciones, Producción, Mantenimiento, Muerte del Proyecto [13].

Consideraciones parciales

En este capítulo se analizaron algunas de las tecnologías y tendencias actuales para el desarrollo de aplicaciones y en particular aquellas que permitan el tráfico de información en la red, con el objetivo de desarrollar un servidor que permita el envío de información (puntos, alarmas) desde el SCADA-GALBA hacia un cliente web, para que el mismo pueda mostrar esta información en navegadores y estaciones remotas.

A partir del estudio realizado queda demostrado que la tecnología Websocket es la ideal para la construcción de la solución, por lo que se hará uso de la biblioteca libwebsocket, una de las múltiples implementaciones de esta tecnología, la cual está escrita en el lenguaje C, que junto al C++, serán los lenguajes que se usarán para construir la solución, además se usará el framework Qt, por las facilidades que brinda para integrarse con los lenguajes antes mencionados y para integrar la solución al HMI que

actualmente se desarrolla con este framework. Visual Paradigm es la herramienta de modelado a usar, ya que se caracteriza por ser muy intuitiva y como metodología de desarrollo, XP por ser una metodología ágil, que propicia varias ventajas para proyectos pequeños y cambiantes.

Capítulo 2: Construcción de la solución

Introducción

En el siguiente capítulo se hace una descripción general de la propuesta del sistema, se define una arquitectura y se hace una descripción del diseño. Además se exponen las historias de usuarios que regirán el desarrollo de la solución propuesta y se muestran tablas y diagramas que describen el funcionamiento del sistema.

2.1. Descripción de las acciones vinculadas al campo de acción

Actualmente el SCADA-GALBA no posee una aplicación que acceda a su capa de comunicación, adquiera la información referente a las pantallas o despliegues del HMI y los envíe hacia la web, lo antes mencionado representa un inconveniente en cuanto a la portabilidad y accesibilidad de la información, pues no existe la posibilidad que un cliente web adquiera estos datos y los muestre en un navegador o estación remota que no necesariamente esté ubicada en la empresa. Hoy en día con el gran advenimiento tecnológico existen más procesos automatizados y controlados a distancia haciendo uso de la gran red de redes, pues la misma posee un gran alcance y extensibilidad en casi todo el mundo.

2.2. Propuesta del sistema

La realización de este trabajo está orientada al desarrollo de un servidor capaz de conectarse con la capa de comunicación del SCADA-GALBA, autenticar los usuarios con sus respectivos privilegios, adquirir la información (puntos y alarmas) y enviarla hacia un cliente web con el cual se establecerá un canal de comunicación bidireccional usando la tecnología Websocket, para atender las peticiones que el mismo realice, también será capaz de cargar la configuración general del sistema SCADA de forma dinámica desde su respectivo archivo(.XML). Por último, el servidor debe ser capaz de brindar un servicio continuo a no ser que el usuario lo detenga.

2.3. Exploración

La metodología XP plantea, que en esta fase los clientes definen a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto [13].

Esta fase se sugiere que sea de 2 ó 3 semanas hasta pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Para esta fase se decidió utilizar 3 semanas, debido a que el sistema que se desea desarrollar no es un programa de una gran extensión y el equipo de desarrollo tiene conocimiento acerca del trabajo con la mayoría de las herramientas que se utilizarán.

2.3.1. Actores del sistema

Tabla 2. Actores del sistema

Actores	**Descripción**
Administrador	Usuario capaz de configurar, iniciar y detener
Cliente de Visualización	Agente externo que se comunica y realiza peticiones al servidor.

2.3.2. Historias de usuario

En la metodología XP las historias de usuario (HU) son utilizadas para especificar los requisitos del software, garantizando que esta tarea sea considerablemente simple. Consisten en descripciones cortas y escritas sin terminología técnica que el cliente hace acerca del software, cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas, con ellas el cliente describe y prioriza sus necesidades. Por otra parte los programadores estiman el esfuerzo asociado y las dependencias entre ellas y planifican el trabajo desde el punto de vista técnico, las HU son divididas en tareas para las cuales también se realiza una estimación. Teniendo en cuenta el esfuerzo asociado a las HU y las prioridades del cliente se define una versión que sea de valor y que tenga una duración de unos pocos meses.

Tabla 3. Iniciar servidor

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Iniciar servidor

Actor: Administrador	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.5
Nivel de Complejidad: Media	**Puntos Reales:** 0.5
Descripción: El sistema debe permitir al administrador iniciar el servicio	
Observaciones: El sistema se inicia, esperando por las peticiones del cliente	

Tabla 4. Detener servidor

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Detener servidor
Actor: Administrador	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.5
Nivel de Complejidad: Media	**Puntos Reales:** 0.5
Descripción: El sistema debe permitir al administrador detener el servicio	
Observaciones: El sistema detiene todos los servicios.	

Tabla 5. Enviar la configuración del proyecto

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Enviar la configuración del proyecto
Actor: Cliente de Visualización	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 1
Nivel de Complejidad: Alta	**Puntos Reales:** 1
Descripción: El Cliente de Visualización hace la petición de la configuración del proyecto y el servidor debe ser capaz de atenderla y enviar la información necesaria.	
Observaciones:	

Tabla 6. Enviar los puntos de un despliegue

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Enviar los puntos de un despliegue
Actor: Cliente de Visualización	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 1.5
Nivel de Complejidad: Alta	**Puntos Reales:** 1.5
Descripción: El Cliente de Visualización hace la petición de los puntos de un despliegue y el servidor debe ser capaz de atenderla y enviar la información necesaria.	
Observaciones:	

Tabla 7. Enviar las últimas cinco alarmas

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Enviar las últimas cinco alarmas
Actor: Cliente de Visualización	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 1.5
Nivel de Complejidad: Alta	**Puntos Reales:** 1.5
Descripción: Luego de establecerse la conexión con el Cliente de Visualización, el servidor debe ser capaz de enviar en todo momento, los detalles de las últimas cinco alarmas del SCADA-GALBA.	
Observaciones:	

Tabla 8. Autenticar usuario

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Autenticar usuario
Actor: Cliente de Visualización	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 1
Nivel de Complejidad: Alta	**Puntos Reales:** 1
Descripción: El Cliente de Visualización hace la petición de autenticar un usuario enviando su usuario y contraseña, y el sistema debe ser capaz de verificar si los parámetros son válidos y enviar una respuesta al cliente de visualización.	

Observaciones:

Tabla 9. Enviar sumario de puntos

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Enviar sumario de puntos
Actor: Cliente de Visualización	**Iteración Asignada:** 3
Prioridad de Negocio: Alta	**Puntos Estimados:** 1
Nivel de Complejidad: Alta	**Puntos Reales:** 1
Descripción El Cliente de Visualización hace la petición del sumario de puntos y el servidor debe ser capaz de atenderla y enviar todos los detalles de todos los puntos(nombre, descripción, valor, grupo operacional, calidad)	
Observaciones:	

Tabla 10. Enviar sumario de alarmas

Historia de Usuario	
Número: 8	**Nombre Historia de Usuario:** Enviar sumario de alarmas
Actor: Cliente de Visualización	**Iteración Asignada:** 3
Prioridad de Negocio: Alta	**Puntos Estimados:** 1
Nivel de Complejidad: Alta	**Puntos Reales:** 1
Descripción El Cliente de Visualización hace la petición del sumario de alarmas y el servidor debe ser capaz de atenderla y enviar todos los detalles de todas las alarmas.	
Observaciones:	

Tabla 11. Enviar detalles de un punto

Historia de Usuario	
Número: 9	**Nombre Historia de Usuario:** Enviar detalles de un punto
Actor: Cliente de Visualización	**Iteración Asignada:** 3
Prioridad de Negocio: Alta	**Puntos Estimados:** 1
Nivel de Complejidad: Alta	**Puntos Reales:** 1

Descripción El Cliente de Visualización hace la petición del detalle de un punto y el servidor debe ser capaz de atenderla y enviar todos los detalles referentes al punto.
Observaciones:

2.3.3. Diseño de Casos de Pruebas

Uno de los pilares de la metodología XP es el proceso de pruebas que anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones [15].

Pruebas de aceptación

Las pruebas de aceptación están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra" (*"Black box system tests"*). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación [14].

Pruebas de rendimiento

Debido a las características que presenta la solución de ser un servidor Websocket que atiende peticiones de múltiples usuarios a través de la web, se decidió realizar algunas pruebas de rendimiento que, desde una perspectiva, determinan lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Además pueden comparar dos sistemas para encontrar cuál de ellos funciona mejor. O pueden medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal [16].

El diseño de las pruebas antes mencionadas, así como los resultados obtenidos luego de su aplicación al sistema se encuentran detallados en el capítulo 3.

2.4. Planificación y entrega

La planificación es la fase donde el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas [14].

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

2.4.1. Plan de Entrega

Luego de que el cliente tenga definida las HU, es posible realizar el Plan de Entregas con la intención de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle, o sea, para fijar el período de tiempo que se puede tardar en la implementación de cada una.

Tabla 12. Plan de entrega

Historias de usuario	Iteración 1 (5 /3/2012)	Iteración 2 (26/3/2012)	Iteración 3 (23/4/2012)
Iniciar servidor Detener servidor Enviar la configuración del proyecto	3	Terminado	Terminado
Enviar los puntos de un despliegue Enviar las alarmas Autenticar usuario	No empezado	4	Terminado
Enviar sumario de puntos Enviar sumario de alarmas Enviar detalles de un punto	No empezado	No empezado	3

Tabla 13.Estimación del esfuerzo por historia de usuario

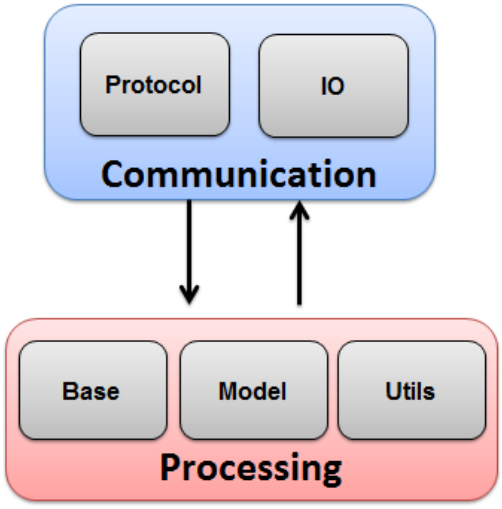
Historias de usuario	Tiempo estimado	Iteración asignada	Tiempo real

	(semana ideal de trabajo)		
Iniciar servidor Detener servidor Enviar la configuración del proyecto	3	1	3
Enviar los puntos de un despliegue Enviar las alarmas Autenticar usuario	4	2	4
Enviar sumario de puntos Enviar sumario de alarmas Enviar detalles de un punto	3	3	3

2.5. Arquitectura del sistema

La aplicación presenta una arquitectura de N Capas pues permite organizar el modelo de diseño de forma que puedan estar físicamente distribuidas, lo que permite simplificar la comprensión y la organización de sistemas complejos [17], para la aplicación se definieron dos capas, la capa Communication encargada de establecer y manejar las comunicaciones con el cliente websocket y con la capa de comunicación del SCADA-GALBA y la capa Processing encargada de realizar todo el procesamiento de los datos.

Figura 2. Arquitectura del sistema.



2.6. Diseño del sistema

Para el diseño de aplicaciones informáticas la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML. En su lugar se usan otras técnicas como las

tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

2.6.1. Patrones de diseño

Los patrones son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Es una descripción de un problema y la solución a la que le da el nombre y que se puede aplicar en nuevos contextos. Muchos patrones ayudan a asignar responsabilidades a los objetos. Un patrón es un par/problema solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos [18].

Para el desarrollo de la solucion se hará uso del patrón Singleton para lograr una única instancia global de la clase System, pues el mismo está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto, además se usa el patrón Iterator para garantizar una gran comodidad a la hora de trabajar con los puntos y alarmas, pues este patrón de diseño permite recorrer una estructura de datos sin que sea necesario conocer la estructura interna de la misma.

2.6.2. Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Para poder diseñar el sistema como un equipo se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente, tal y como muestra la figura.

Tabla 14. Modelo de Tarjeta CRC.

Clase	
Responsabilidades	Colaboradores

Clase: es cualquier persona, evento, concepto, pantalla o reporte.

Responsabilidades: las responsabilidades de una clase son las funciones que conoce y las que realizan, sus atributos y métodos.
Colaboradores: los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

El sistema está compuesto por una serie de paquetes que conforman su arquitectura, a continuación se presentan sus descripciones y las respectivas tarjetas CRC pertenecientes a las clases de los mismos.

Paquete Base

En este paquete se encuentran las clases sobre las cuales se soporta parte del sistema, pues contiene las clases que manejan los puntos, las alarmas, y la configuración.

Tabla 14. Tarjeta CRC para la clase ConfigManager.

ConfigManager	
Descripción: Encargada de cargar las configuraciones contenidas en archivos XML.	
Atributos:	
Nombre	**Descripción**
fileConf	atributo de tipo QFile, contiene el archivo XML con la configuración del proyecto
screenCollection	contiene una lista con los despliegues o pantallas
ip	contiene la dirección ip para la conexión con el módulo de adquisición
port	contiene el puerto para la conexión con el módulo de adquisición
channelId	canal dedicado a la adquisición de puntos de la capa de comunicación
channelAId	canal dedicado a la adquisición de alarmas de la capa de comunicación
timeToRequestBDTR	contiene el tiempo en el cual se consultará la BDTR
timeToSendHTTP	contiene el tiempo en el cual se enviará la información a la web
Responsabilidades:	
Nombre	**Colaborador**
Cargar configuración de un *.xml	
Obtener los id de los puntos de un despliegue	
Leer un proyecto	
Leer la colección de nodos	
Leer un nodo	

Leer la colección de módulos	
Leer un despliegue	Screen
Leer la colección de gráficos	Screen
Leer un gráfico	
Añadir un despliegue	Screen

Tabla 15. Tarjeta CRC para la clase AlarmManager.

AlarmManager	
Descripción: Manejador de las alarmas	
Atributos:	
Nombre	**Descripción**
alarmsCollection	Contiene una lista de alarmas
alarmFiveList	Contiene una lista de 5 alarmas
Responsabilidades:	
Nombre	**Colaborador**
Añadir una alarma	
Comprobar si existen alarmas	
Obtener colección de alarmas	
Obtener las últimas cinco alarmas	Serializer
Obtener alarma por id de recurso	

Tabla 16. Tarjeta CRC para la clase PointManager

PointManager	
Descripción: Esta clase es la encargada del manejo de los puntos.	
Atributos:	
Nombre	**Descripción**
pointCollection	Contiene una lista de puntos.
Responsabilidades:	
Nombre	**Colaborador**
Añadir un punto	
Obtener la lista de puntos de un despliegue	System, Serializer
Obtener la lista de puntos del sumario de puntos	Serializer

Tabla 17. Tarjeta CRC para la clase System

System

Descripción: Esta es la clase principal del sistema, que contiene objetos de las clases encargadas de realizar todas las funcionalidades del sistema, por lo cual está basada en el patrón Singleton para garantizar una única instancia global.	
Atributos:	
Nombre	**Descripción**
ioManager	Atributo de tipo IOManager
pointManager	Atributo de tipo PointManager
configManager	Atributo de tipo ConfigManager
AlarmManager	Atributo de tipo AlarmManager
startSystem	Atributo de tipo boleano para comprobar el estado del sistema
Responsabilidades:	
Nombre	**Colaborador**
Iniciar el sistema	
Obtener manejador de configuración	
Obtener manejador de seguridad	
Obtener manejador de alarmas	
Obtener manejador de puntos	

Paquete IO

Este paquete contiene la clase IOManager la cual posee un mecanismo de adquisición de puntos y alarmas de la capa de comunicaciones del SCADA-GALBA, información que luego será enviada como respuesta a la petición que realice el cliente websocket.

Tabla 18. Tarjeta CRC para la clase IOManager.

IOManager	
Descripción: Encargada de la adquisición de puntos y alarmas desde el SCADA-GALBA	
Atributos:	
Nombre	**Descripción**
factory	Instancia de la fábrica abstracta para la comunicación con el SCADA-GALBA
connectionV	Instancia para crear una conexión para la adquisición de variables
variablesIO	Instancia para recibir el lote de variables que provee la conexión
connectionA	Instancia para crear una conexión para la adquisición de alarmas
alarmsIO	Instancia para recibir el lote de alarmas que provee la conexión
timeToRequestBDTR	Tiempo para solicitar los lotes de puntos y alarmas provenientes del SCADA-GALBA
Responsabilidades:	

Nombre	Colaborador
Conectarse con la base de datos en tiempo real	
Adquisición de puntos y alarmas	
Obtener el siguiente lote de variables	
Obtener el siguiente lote de alarmas	

Paquete Model

En este paquete se encuentran las entidades pertenecientes al modelo de la aplicación.

Tabla 19. Tarjeta CRC para la clase PointData

PointData	
Descripción: Contiene los datos de un punto, y los métodos necesarios para acceder sus atributos.	
Atributos:	
Nombre	**Descripción**
name	Contiene el nombre del punto
description	Contiene la descripción del punto
value	Contiene el valor del punto
group	Contiene el grupo del punto
quality	Contiene la calidad del punto
Responsabilidades:	
Nombre	**Colaborador**
Asignar y obtener las propiedades de un punto	

Tabla 20. Tarjeta CRC para la clase AlarmData.

AlarmData	
Descripción: Contiene los datos de una alarma y los métodos necesarios para acceder y cambiar sus atributos.	
Atributos:	
Nombre	**Descripción**
associateResourceType	tipo de recurso asociado a la alarma
associateResourceId	id del recurso asociado a la alarma
tagName	nombre de la alarma

group	grupo de la alarma
type	tipo de la alarma
description	descripción de la alarma
cause	causa de la alarma
ocurrenceNumber	número de ocurrencia de la alarma
Responsabilidades:	
Nombre	**Colaborador**
Asignar y obtener las propiedades de una alarma	

Tabla 21. Tarjeta CRC para la clase Screen.

Screen	
Descripción: Contiene los datos de una pantalla o despliegue.	
Atributos:	
Nombre	**Descripción**
name	contiene el nombre del despliegue
graphicComponentCollection	contiene una lista con todos los componentes gráficos del despliegue
Responsabilidades:	
Nombre	**Colaborador**
Asignar y obtener las propiedades de un despliegue	
Leer las propiedades de un archivo *.xml	
Añadir los objetos gráficos	

Tabla 22. Tarjeta CRC para la clase GraphicComponent.

GraphicComponent	
Descripción: Contiene los datos de un componente gráfico.	
Atributos:	
Nombre	**Descripción**
varId	contiene el id de un componente gráfico
Responsabilidades:	
Nombre	**Colaborador**
Asignar y obtener las propiedades de un componente gráfico	
Leer las propiedades de un *.xml	
Añadir un objeto gráfico	

Paquete Utils

En este paquete se encuentran las clases que complementan o realizan funciones secundarias para el sistema.

Tabla 23. Tarjeta CRC para la clase Serializer.

Serializer	
Descripción: Esta clase contiene métodos capaces de construir cadenas con la información de los puntos o alarmas, siguiendo el formato ligero para el intercambio de datos JSON, con el objetivo de mantener la rapidez del protocolo de comunicación websocket y establecer un lenguaje de comunicación con el cliente websocket.	
Responsabilidades:	
Nombre	**Colaborador**
Serializar la información de los puntos	
Serializar la información de las alarmas	

Tabla 24. Tarjeta CRC para la clase Singleton.

Singleton	
Descripción: Esta clase es usada para garantizar una única instancia global de la clase System	
Atributos:	
Nombre	**Descripción**
instance	Contiene la clase que será única y global
Responsabilidades:	
Nombre	**Colaborador**
Instanciar una clase única	

2.6.3. Funcionalidades del Paquete Protocol

En este paquete se desarrollan las distintas funciones o llamadas de retorno (como se define en la biblioteca LibWebsocket) destinadas a la comunicación con el protocolo WebSocket. Su implementación es basada en las bondades del lenguaje C y utilizando la biblioteca LibWebsocket. Cada llamada de retorno define la forma de actuar del servidor para cada petición del cliente el cual implementa las mismas funciones para lograr establecer la comunicación bidireccional con el servidor. En la biblioteca se define una estructura a seguir a la hora de implementar estas funciones. A continuación se describe la misma:

Tabla 15. Estructura de las llamadas de retorno

Estructura de las llamadas de retorno	
static int nombre_de_funcion(…**parámetros**…)	
Parámetros	**Descripción**
struct libwebsocket_context * **b**	Contexto Websocket
struct libwebsocket ***wsi**	Instancia de puntero abstracto Websocket
enum libwebsocket_callback_reasons **reason**	La razón de la llamada
void ***user**	Puntero a los datos del usuario a nivel de sesión asignado por la biblioteca
void ***in**	Puntero utilizado para algunas de las razones de devolución de llamada
size_t **len**	Longitud de conjunto de algunas de las razones de devolución de llamada

A partir de esta estructura y de las historias de usuarios definidas por el cliente, se definen las llamadas de retorno que tendrá el servidor para controlar y satisfacer las peticiones del cliente websocket las mismas son:

1. ***callback_http***

   Esta función permite conocer el nombre del ordenador cliente así como su dirección ip, para tener un control sobre el acceso al servidor.

2. ***callback_security***

   Función encargada de manejar la petición de autenticar usuario, enviando la respuesta que ofrezca el módulo de seguridad del SCADA-GALBA, hacia el cliente que previamente envió un nombre de usuario y una contraseña.

3. ***callback_runtime***

   Esta función es la encargada de manejar todas las peticiones que realice el cliente de visualización acerca de la información de los puntos (valor de los puntos de un despliegue, sumario de puntos analógicos y digitales, detalles de un punto).

4. ***callback_alarms***

   Función encargada de manejar las peticiones de las alarmas que realice el cliente.

2.6.4. Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física, que permite que todos los participantes del

proyecto lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible, para facilitar la lectura, comprensión y mantenimiento del código. Por otra parte la metodología XP enfatiza la comunicación de los programadores a través del código, con lo cual es de gran importancia que se sigan ciertos estándares. Para lograr todo lo antes mencionado se usarán los estándares de codificación que se exponen en el documento Estándares de codificación para C++ versión 1.0 del autor Ariel Chávez Lorenzo, por el cual se rige toda la implementación del código del proyecto SCADA Guardián del ALBA.

2.7. Desarrollo de Iteraciones

En esta fase son desarrolladas las funcionalidades del sistema, generando al final de cada iteración un entregable funcional que implementa las historias de usuario pertenecientes a esa iteración. Al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor del negocio), las iteraciones son también utilizadas para medir el progreso del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción [13].

2.7.1. Implementación

Las HU se van desarrollando durante el transcurso de la iteración a la cual pertenecen. Al principio de cada iteración se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan se descomponen las HU en tareas de desarrollo, asignando a un grupo o una persona como responsable de su implementación. Estas tareas son para el uso estricto de los programadores por lo que pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente.

A continuación quedan detalladas las tareas de desarrollo realizadas en cada una de las iteraciones.

Iteración 1

La primera iteración tendrá como objetivo el desarrollo de las HU 1, 2, 3.

Tabla 16. Historias de usuarios planificadas para la primera iteración

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Iniciar servidor	0.5	0.5
Detener servidor	0.5	0.5
Enviar la configuración del servidor	1	1

Tareas de las historias de usuarios desarrolladas en la primera iteración

Luego de relacionar las HU pertenecientes a esta iteración, se procede a la especificación de las principales tareas de desarrollo que se realizaron para cumplir el propósito de la misma. (Anexo 1)

Iteración 2

Se decidió dedicar una iteración a las HU 4, 5, 6 pues las mismas representan una gran importancia, ya que se dedican al envío de puntos y alarmas, funciones rectoras del sistema, además su desarrollo es útil para el desarrollo de otras HU.

Tabla 17. Historias de usuarios planificadas para la primera iteración

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Enviar los puntos de un despliegue	1.5	1.5
Enviar las alarmas	1.5	1.5
Autenticar usuario	1	1

Tareas de las historias de usuarios desarrolladas en la segunda iteración

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

Iteración 3

Esta última iteración tendrá como objetivo el desarrollo de las HU 7, 8, 9 y al finalizar la iteración se tendrá un entregable, con todas las funcionalidades especificadas por el cliente.

Tabla 18. Historias de usuarios planificadas para la primera iteración

Historia de usuario	Tiempo estimado (Semanas)	Tiempo real (Semanas)
Enviar sumario de puntos	1	1
Enviar sumario de alarmas	1	1
Enviar detalles de un punto	1	1

Tareas de las historias de usuarios desarrolladas en la tercera iteración

Luego de relacionar las HU pertenecientes a esta iteración, se procede a la especificación de las principales tareas de desarrollo que se realizaron para cumplir el propósito de la misma. (Anexo 3)

Consideraciones parciales

A partir de las acciones vinculadas al campo de acción, se definió una propuesta del sistema. Se especificaron los usuarios que estarán relacionados con su utilización y funcionamiento. Además se realizó una descripción de las HU precisando por el cliente la prioridad de cada una, definiendo así el orden en el que serán implementadas las mismas. Se cuenta con 9 HU que serán implementadas en tres iteraciones. Quedó elaborado el modelo necesario para llevar a cabo la implementación del sistema mediante la descripción de los estándares de codificación y la arquitectura utilizada. Se plantearon y describieron las tareas de desarrollo para dar cumplimiento a las funcionalidades abordadas por las HU. Por último se diseñaron las pruebas de aceptación y de rendimiento que comprueban que el sistema cumpla con las funcionalidades que el cliente definió, la descripción de estas prueba en conjunto con los resultados se encuentran en el siguiente capítulo.

Capítulo 3: Pruebas al software

Introducción

Todo sistema de software debe ser probado antes de su entrega o puesta en funcionamiento, a este proceso o fase se le denomina prueba o testeo de la aplicación. Es una de las etapas más importante en la vida del software, y que en ocasiones menos recursos se le asignan, y esto es una mala práctica que conlleva en muchos casos al fracaso del producto o a la no satisfacción del cliente. No siempre las pruebas garantizan que el software cumpla con todos los aspectos funcionales, definidos por el cliente, pero si detectan numerosas fallas que el desarrollador no encontró y que ahora pueden ser corregidas y vueltas a probar. En el presente capítulo están detallados los resultados de la ejecución de las pruebas previamente diseñadas para probar las funcionalidades descritas.

3.1. Pruebas de aceptación

La aplicación de estas pruebas permitió comprobar las funcionalidades del sistema definidas por el cliente. A continuación se encuentran los resultados de las mismas:

Tabla 19. Enviar la configuración del proyecto

Caso de prueba de Aceptación	
Número: 1	**Historia de Usuario:** 3
Nombre: Enviar la configuración del proyecto	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena con toda la información que contiene el archivo .XML que genera el SCADA-GALBA acerca de un proyecto existente.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket	
Entradas/Pasos de ejecución: El cliente Websocket envía la cadena ***config***. -El sistema ejecuta la función ***callback_configuration***. -El sistema lee la información de un archivo .xml -El sistema envía una cadena con la información del .xml	
Resultado esperado: El sistema lee la información de archivo xml y la envía correctamente	
Evaluación de la prueba: Satisfactorio	

Tabla 20. Enviar los puntos de un despliegue

Caso de prueba de Aceptación	
Número: 2	**Historia de Usuario:** 4
Nombre: Enviar los puntos de un despliegue	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena en formato JSON, con la información de los puntos del despliegue que se quiere visualizar.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***runtime_protocol***	
Entradas/Pasos de ejecución: El cliente Websocket envía un cadena que contiene el id del despliegue a visualizar ejemplo 1. -El sistema ejecuta la función ***callback_runtime***. -El sistema adquiere los valores de los puntos de la capa de comunicación del SCADA-GALBA. -El sistema construye una cadena en formato JSON, con la información adquirida. -El sistema envía la cadena al cliente Websocket. -El sistema vuelve al primer paso para garantizar la actualización de los valores de los puntos.	
Resultado esperado: El sistema envía una cadena en formato JSON con la información de los puntos de un despliegue.	
Evaluación de la prueba: Satisfactorio	

Tabla 21. Enviar las últimas cinco alarmas

Caso de prueba de Aceptación	
Número: 3	**Historia de Usuario:** 5
Nombre: Enviar las últimas cinco alarmas	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena en formato JSON, con la información de los últimas cinco alarmas que genera el SCADA-GALBA	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***alarms_protocol***	
Entradas/Pasos de ejecución: -El sistema ejecuta la función ***callback_alarms***. -El sistema adquiere los valores de las alarmas de la capa de comunicación del SCADA-GALBA. -El sistema construye una cadena en formato JSON, con la información de las últimas cinco alarmas. -El sistema envía la cadena al cliente Websocket y vuelve al primer paso para garantizar la actualización de las alarmas en todo momento.	
Resultado esperado: El sistema envía una cadena en formato JSON con la información de las últimas cinco alarmas generadas por el SCADA-GALBA.	

Evaluación de la prueba: Satisfactorio

Tabla 22. Autenticar usuario

Caso de prueba de Aceptación	
Número: 4	**Historia de Usuario:** 6
Nombre: Autenticar usuario	
Descripción: El caso de prueba permite comprobar que el sistema es capaz de autenticar un usuario o enviar una respuesta al cliente en caso de existir un error de autenticación.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***security_protocol***	
Entradas/Pasos de ejecución: El cliente Websocket envía un cadena que contiene el nombre y la contraseña del usuario que se desea autenticar (***ejemplo: u:user;p:123456*;)**. -El sistema ejecuta la función ***callback_security***. -El sistema intenta autenticar el usuario en el SCADA-GALBA. -El sistema envía la respuesta que ofrece el módulo de seguridad del SCADA-GALBA.	
Resultado esperado: El sistema envía una cadena con la respuesta del módulo de seguridad del SCADA-GALBA	
Evaluación de la prueba: Satisfactorio	

Tabla 23. Enviar sumario de puntos

Caso de prueba de Aceptación	
Número: 5	**Historia de Usuario:** 7
Nombre: Enviar sumario de puntos	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena en formato JSON, con la información del sumario de puntos analógicos o digitales que previamente fueron adquiridos de la capa de comunicación del SCADA-GALBA.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***runtime_protocol***	
Entradas/Pasos de ejecución: El cliente Websocket envía la cadena SummaryAnalogicPoints o SummaryDigitalPoints. -El sistema ejecuta la función ***callback_runtime***. -El sistema adquiere los valores de los puntos analógicos o digitales de la capa de comunicación del SCADA-GALBA, en correspondencia con la cadena recibida. -El sistema construye una cadena en formato JSON, con la información de los puntos. -El sistema envía la cadena al cliente Websocket y vuelve al primer paso para garantizar la actualización de los puntos.	
Resultado esperado: El sistema envía una cadena en formato JSON con la información de los puntos adquiridos desde la capa de comunicación del SCADA-GALBA.	

Evaluación de la prueba: Satisfactorio

Tabla 24. Enviar sumario de alarmas

Caso de prueba de Aceptación	
Número: 6	**Historia de Usuario:** 8
Nombre: Enviar sumario de alarmas	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena en formato JSON, con la información del sumario alarmas que previamente fueron adquiridas de la capa de comunicación del SCADA-GALBA.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***runtime_protocol***	
Entradas/Pasos de ejecución: El cliente Websocket envía la cadena SummaryAlarms. -El sistema ejecuta la función ***callback_runtime***. -El sistema adquiere los valores de las alarmas de la capa de comunicación del SCADA-GALBA. -El sistema construye una cadena en formato JSON, con la información de las alarmas. -El sistema envía la cadena al cliente Websocket y vuelve al primer paso para garantizar la actualización de las alarmas.	
Resultado esperado: El sistema envía una cadena en formato JSON con la información de las alarmas adquiridas desde la capa de comunicación del SCADA-GALBA.	
Evaluación de la prueba: Satisfactorio	

Tabla 25. Enviar detalles de un punto

Caso de prueba de Aceptación	
Número: 7	**Historia de Usuario:** 9
Nombre: Enviar detalles de un punto	
Descripción: El caso de prueba permite comprobar que el sistema devuelve una cadena en formato JSON, con los detalles de un punto en específico que previamente fueron adquiridas de la capa de comunicación del SCADA-GALBA.	
Condiciones de ejecución: Conexión establecida con el cliente Websocket a través del protocolo ***runtime_protocol***	
Entradas/Pasos de ejecución: El cliente Websocket envía la cadena que contiene el identificador del punto (**ejemplo: id:43552345;**). -El sistema ejecuta la función ***callback_runtime***. -El sistema adquiere los detalles del punto de la capa de comunicación del SCADA-GALBA. -El sistema construye una cadena en formato JSON, con los detalles del punto. -El sistema envía la cadena al cliente Websocket y vuelve al primer paso para garantizar la actualización de los detalles del punto.	

Resultado esperado: El sistema envía una cadena en formato JSON con los detalles del punto que se necesita visualizar.
Evaluación de la prueba: Satisfactorio

Luego de la aplicación de las pruebas al sistema se llegó a la conclusión de que el sistema cumple con las funcionalidades que el cliente definió.

3.2. Pruebas de rendimiento

El diseño de los casos de prueba es una de las tareas más importantes antes de la realización de las pruebas de carga y estrés. Un diseño erróneo puede llevar a tomar conclusiones equivocadas, positiva o negativamente, respecto a los tiempos de respuestas del sistema.

Los casos de pruebas identificados son:

- ✓ CP1_ Enviar los puntos de un despliegue.
- ✓ CP2_ Enviar sumario de puntos
- ✓ CP3_ Enviar sumario de alarmas.
- ✓ CP4_ Enviar detalles de un punto.
- ✓ CP5_Enviar las últimas cinco alarmas.

Los casos de pruebas mencionados anteriormente tienen como objetivo comprobar el rendimiento de las funcionalidades que se corresponden con su nombre, para ello se toman dos variables, cantidad de usuarios concurrentes (U) y tiempo promedio de respuesta en segundos (TRS), y se le asignan dos valores a la variable U, 1 y 20, que representa la cantidad de usuario concurrentes realizando peticiones respectivamente, luego se toman los tiempos en los cual el sistema satisface las peticiones del cliente web, para un posterior análisis de los resultados obtenidos. Se debe tener en cuenta que el sistema lee su configuración desde un archivo *.xml, donde se encuentra el parámetro TimeToRequestBDTR que representa el tiempo entre las adquisiciones de datos de la capa de comunicaciones del SCADA-GALBA y TimeToSendHTTP tiempo entre los envíos de los datos a la web, por lo que si se varía el valor de estos parámetros, los resultados de las pruebas también lo harán. Para realizar estas pruebas los valores asignados a estos parámetros fueron 0.8s y 1s respectivamente.

Para la ejecución de las pruebas de rendimiento se dispone de los siguientes recursos:
Físicos: ordenador con núcleo Intel(R) Core(TM)2 Duo CPU E4500 a 2.20GHz, memoria RAM de 1 Gb y

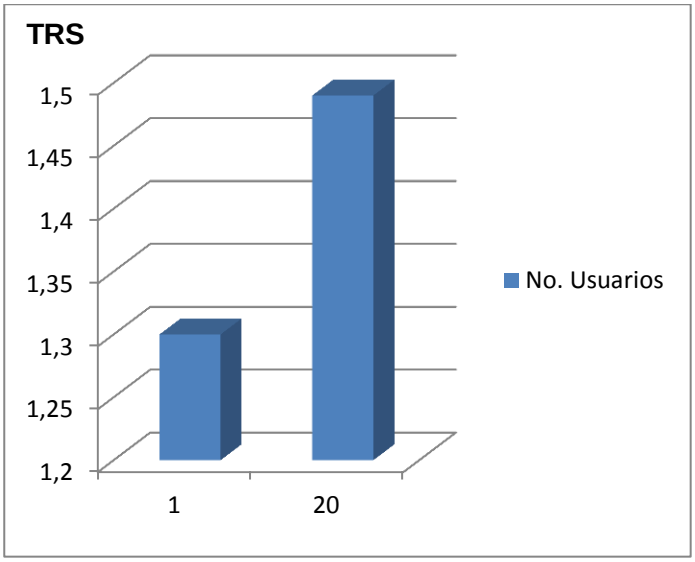
un disco duro con capacidad de 160 Gbytes.

Lógicos: sistema operativo Debian Squeeze con kernel 2.6.32, Monitor de recursos del sistema.

3.2.1. Análisis de los resultados

La ejecución del CP1_ Enviar los puntos de un despliegue arrojó como resultado que el sistema soporta los 20 usuarios conectados concurrentemente.

Figura 3. CP1_Enviar los puntos de un despliegue.



Durante la ejecución de los casos de prueba CP2_ Enviar sumario de puntos (Figura 4) y CP3_ Enviar sumario de alarmas. (Figura 5) el TRS mantuvo un comportamiento estable para los valores de 1 y 20 usuarios haciendo peticiones al sistema.

Figura 4. Enviar sumario de puntos.

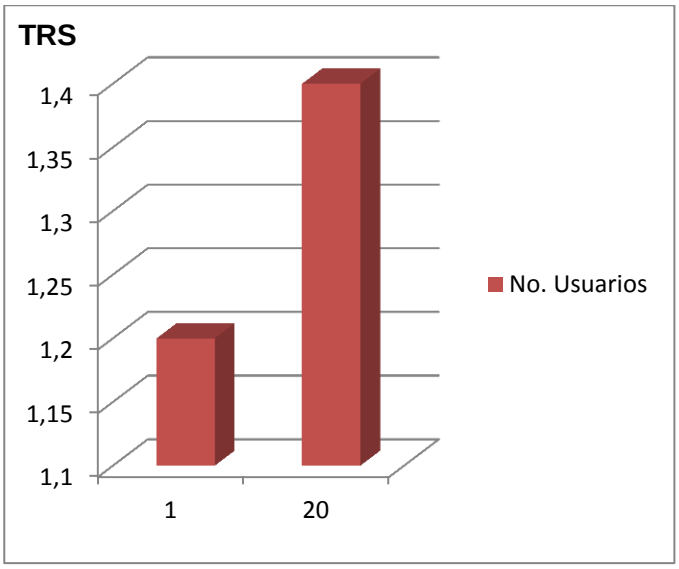
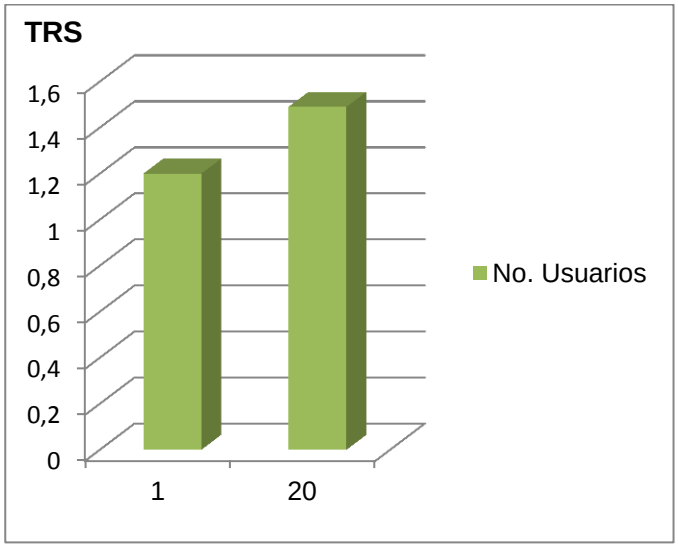
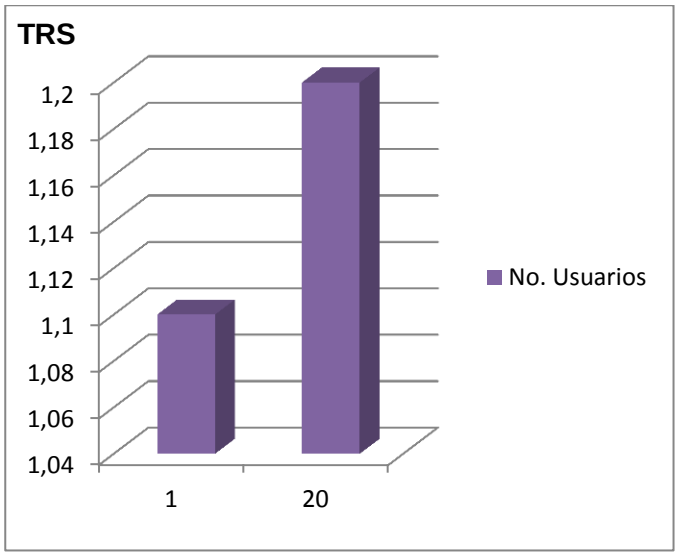


Figura 5. Enviar sumario de alarmas.



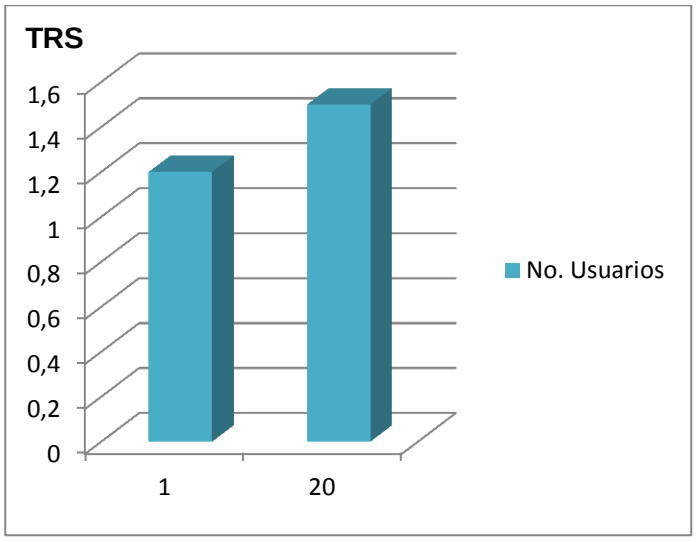
En el caso de prueba CP4_ Enviar detalles de un punto (Figura 6) el rendimiento del sistema tuvo un mejor resultado pues esta funcionalidad maneja una menor cantidad de datos.

Figura 6. Enviar detalles de un punto.



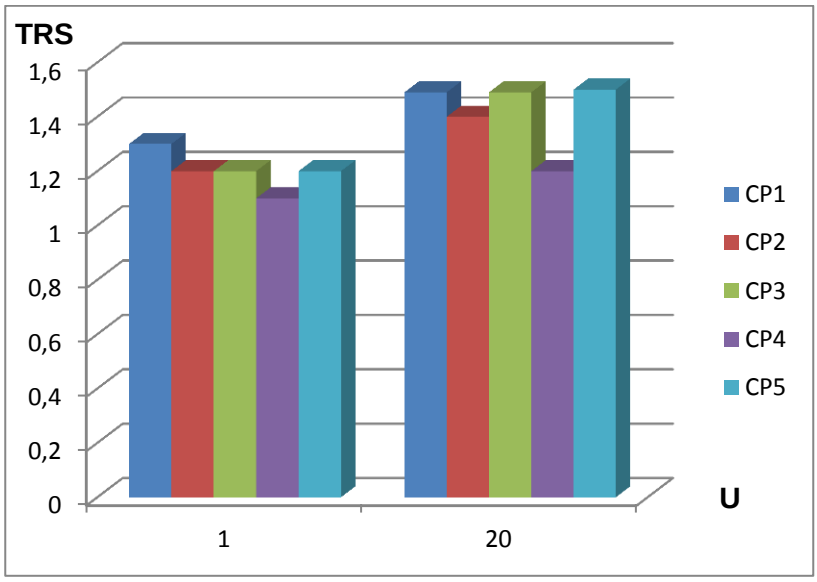
El caso de prueba CP5_Enviar las últimas cinco alarmas (Figura 7) es el que mayor cantidad de peticiones atiende, pues esta funcionalidad envía los datos de las últimas cinco alarmas adquiridas de la capa de comunicación del SCADA-GALBA en todo momento, al realizar esta prueba el TRS se comportó de una manera estable.

Figura 7. Enviar las últimas cinco alarmas



Como resumen de las pruebas de carga, se puede concluir que la aplicación soporta los 20 usuarios conectados simultáneamente, al realizarse las pruebas se monitoreo el uso de la memoria del ordenador y los resultados estuvieron en el rango de 1mb y 3mb, por lo que se considera que el sistema consume una cantidad de memoria aceptable.

Figura 8. Resumen de las pruebas



Consideraciones parciales

Se diseñaron y ejecutaron las pruebas de aceptación y de rendimiento que permitieron demostrar que el sistema cumple con las funcionalidades que el cliente definió, concluyendo que la propuesta de solución cumple con los resultados esperados que se trazaron al inicio de la investigación.

Conclusiones

Finalizada la investigación se puede concluir que:

- Se logró el desarrollo de un mecanismo de comunicación entre el SCADA y un cliente web que servirá como base para futuras labores de implementación.
- Se obtuvo una arquitectura flexible, que permita la modificación o incorporación de nuevos requerimientos.
- Se generaron los artefactos correspondientes a la metodología seleccionada para el desarrollo del software, con la cual quedaron definidos los requerimientos exigidos por el cliente.
- Fue probado el sistema, lo que permitió demostrar el cumplimiento de las exigencias del cliente.
- Los objetivos propuestos al comienzo de este trabajo se cumplieron satisfactoriamente, además se han incluido una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

Recomendaciones

- Continuar el desarrollo de este sistema y agregarle las funcionalidades relacionadas con el envío de la información referente a los reportes y gráficos de tendencia que genera el SCADA.

- Implementar un mecanismo de generación de log para el sistema, con el objetivo de tener un mayor control acerca del acceso y eventos que el mismo realice.

- Empaquetar la solución para que puede ser instalada como un servicio más del SCADA.

Referencias Bibliográficas

1. **Sehara, Yossel Luis.** *Implementación de un modelo para la configuración de un sistema SCADA.* Ciudad de la Habana : s.n., julio,2008.
2. **Luisa Mora Pérez, Roberto Rodríguez Reyes.** *Módulo de visualización web para el generador de informes del Proyecto SCADA-GALBA.* Ciudad de La Habana : s.n., junio,2010.
3. **Londoño, Johnnatan Montoya.** Monitoreo y visualización de temperaturas en el área de generación de frio de la planta de derivados lacteos Colanta San Pedro de los Milagros. *Biblioteca Digital Repositorio Institucional.* [En línea] 2009. http://www.bdigital.unal.edu.co/917/.
4. Foundation, Free Software. [En línea] 2010. http://www.gnu.org/philosophy/free-sw.es.html.
5. **Saint-Andre, Peter, Smith, Kevin y Troncon, Remko.** *XMPP: The Definitive Guide.* San Peterburgo : O`Reilly Media, 2009.
6. **Ian Hickson, Google, Inc.** Server-Sent Events. [En línea] Octubre de 2011. http://www.w3.org/TR/eventsource/.
7. **The WebSocket API.** [En línea] 8 de Diciembre de 2011. http://www.w3.org/TR/websockets.
8. **Ramírez, Oilede Pérez y López, Yonisley García.** *Desarrollo del Componente Trabajador del Subsistema Capital Humano del sistema Cedrux.* La Habana : s.n., 2009.
9. **Bernardo Zaragoza Hijuelos, Raudi Agdel Bacallao.** *Implementación de un módulo de generación de reportes para sistemas de supervisión, control y adquisición de datos.* Ciudad de La Habana : s.n., julio, 2008. .
10. **Parlanti, Gustavo.** Laboratorio de procesamiento de senal. [En línea] [Citado el: 20 de enero de 2012.] www.dsp.efn.unc.edu.ar/documentos/RepasoC.pdf.
11. Qt- Cross Platform Application and UI Framework. [En línea] Nokia, 2006-20012. http://qt.nokia.com/.
12. **Hernandis, J. A.** Why Visual Paradigm for UML? [En línea] 2005. http://www.visual-paradigm.com/product/vpuml/>..
13. **Penadés, Patricio Letelier y Mª Carmen.** Métodologías ágiles para el desarrollo de software. [En línea] http://www.willydev.net/descargas/masyxp.pdf.
14. **José Joskowicz.** Reglas y Prácticas en eXtreme Programming. [En línea] 2008. http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.
15. **Kent Beck.** Extreme Programming Explained. [En línea] 1999. http://dl.acm.org/citation.cfm?id=1076267.
16. *Corporación Sybven Integración Tecnológica.* [En línea] [Citado el: 15 de mayo de 2012.] http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246.
17. **Torre, César de la, Unai Zorrilla, Ramos, Miguel Ángel y Calvarro, Javier.** *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0. 1ra edición Marzo 2010.*
18. **Rojas, Mc Juan Carlos Olivares.** Patrones de Diseño. [En línea] 10 de Mayo de 2007. [Citado el: 25 de Abril de 2012.] http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07b/patrones.pdf.

Glosario

HTTP: Protocolo para la conexión segura.

API: Interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

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

SCADA: Supervisión, Control y Adquisición de Datos (del inglés Supervisory Control And Data Acquisition).

Protocolo: Conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.

Interfaz Hombre Máquina (HMI): Consiste en herramientas para la supervisión y control del proceso (consolas de operación y generador de despliegue).

Adquisición de datos: Consiste en recolectar un conjunto de variables medidas en forma física a través de diferentes elementos (sensores, transductores) para ser convertidas en digital y que puedan ser utilizadas por el computador.

Arquitectura del sistema: Describe cómo las funcionalidades del mismo se distribuyen sobre un número de componentes lógicos y cómo estos componentes se interrelacionan.

Autenticar: Verificación de la identidad de una persona o de un proceso para acceder a un recurso o poder realizar determinada actividad, así como también para detectar la veracidad del origen de un mensaje electrónico.

Evento: Conjunto de acciones que proceden de la ejecución o activación de otra acción y que ha sido registrada, la combinación de estas acciones también pueden dar como resultado otro evento en particular o una serie de eventos.
Multiplataforma: Sistema que puede ejecutarse en diversos sistemas operativos y tipos de hardware.

Control: Se refiere al conjunto de acciones que permiten modificar el estado actual de los actuadores en campo a través del envío de comandos.

Punto: Los puntos se refieren a las direcciones de memoria que se configuran en el SCADA y que referencian a una variable capturada por un dispositivo de campo, o una variable del proceso. Pueden ser de tipo analógico, digital, texto y vector.

Widget (Objeto gráfico): Objeto gráfico que puede contener gráficos vectoriales o imágenes raster, (imágenes de mapas de bits) y presenta un conjunto de propiedades que pueden ser editadas y asociadas a puntos (variables) del SCADA o a expresiones que contengan variables del sistema. Por medio de los objetos gráficos se pueden crear animaciones en función de los valores de los puntos asociados. Los widgets pueden agruparse para formar nuevos objetos gráficos más complejos. Un ejemplo de widget puede ser un contenedor de textos.

Controlador Lógico Programable (PLC): Dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanas al proceso, permitiendo ejecutar las tareas básicas del control, aun cuando no tenga conexión a las capas superiores del control.

XML: siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes.

ANEXOS

Anexo 1

Tareas de desarrollo de la primera iteración

Definir e implementar como iniciar y detener el servidor

Tarea	
Número de la tarea:1	**Número de HU:** 1,2
Nombre de la tarea: Definir e implementar como iniciar y detener el servidor	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea debe permitir iniciar y detener el servidor	

Implementar el callback configuración

Tarea	
Número de la tarea:2	**Número de HU:** 3
Nombre de la tarea: Implementar el callback de configuración	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición de los parámetros de configuración que realice el cliente de visualización web.	

Anexo 2

Tareas de desarrollo de la segunda iteración

Implementar mecanismo de adquisición de variables

Tarea	
Número de la tarea:3	**Número de HU:** 4
Nombre de la tarea: Implementar mecanismo de adquisición de variables	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a la adquisición de variables de la capa de comunicaciones del SCADA.	

Implementar mecanismo de adquisición de alarmas

Tarea	
Número de la tarea:4	**Número de HU:** 5
Nombre de la tarea: Implementar mecanismo de adquisición de alarmas	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a la adquisición de variables de la capa de comunicaciones del SCADA.	

Implementar el callback puntos

Tarea	
Número de la tarea:5	**Número de HU:** 4
Nombre de la tarea: Implementar el callback de envío de puntos	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición de los puntos de un despliegue realice el cliente de visualización web.	

Implementar el callback alarmas

Tarea	
Número de la tarea:6	**Número de HU:** 5
Nombre de la tarea: Implementar el callback de res de alarmas	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición de las alarmas que realice el cliente de visualización web.	

Implementar el callback de seguridad

Tarea	
Número de la tarea:7	**Número de HU:** 6
Nombre de la tarea: Implementar el callback de seguridad	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición de autenticación que realice el cliente de visualización web.	

Anexo 3

Tareas de desarrollo de la tercera iteración

Implementar el callback sumario de puntos

Tarea	
Número de la tarea:8	**Número de HU:** 7
Nombre de la tarea: Implementar el callback de envío del sumario de puntos	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición del sumario de puntos	

que realice el cliente de visualización web.

Implementar el callback sumario de alarmas

Tarea	
Número de la tarea:9	**Número de HU:** 8
Nombre de la tarea: Implementar el callback de envío del sumario de alarmas	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición del sumario de alarmas que realice el cliente de visualización web.	

Implementar el callback detalles de un punto

Tarea	
Número de la tarea:10	**Número de HU:** 9
Nombre de la tarea: Implementar el callback detalles de un punto	
Tipo de tarea: Desarrollo	**Puntos estimados**:
Programador(es) responsable(s): Julio César Moreno Cámbar	
Descripción: Esta tarea está dirigida a satisfacer la petición de mostrar los detalles de un punto en específico, que realice el cliente de visualización web.	