Universidad de las Ciencias Informáticas
"Facultad 5"

Título: **XStormAlmiquiServer, servidor para el cliente de visualización web del SAINUX Almiquí.**

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor:"Adrian Díaz Díaz"

Tutor: "Ing. Ernesto Leyva Barrero"

Co-tutor: "Ing. Roberto Cárdenas Isla"

Co-tutor: "Ing. Evian Suárez Rodríguez"

"Año 55 de la Revolución"

"Junio 2013"

DECLARACIÓN DE AUTORÍA

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

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

_____
Firma del Autor
(Adrian Díaz Díaz)

_____ _____
Firma del Tutor Firma del Co-tutor
(Ing. Ernesto Leyva Barrero) (Ing. Roberto Cárdenas Isla)

_____
Firma del Co-tutor
(Ing. Evian Suárez Rodríguez)

DATOS DE CONTACTO

Tutor: Ing. Ernesto Leyva Barrero
Ing. Ciencias Informáticas
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: ebarrero@uci.cu

Co-Tutor: Ing. Roberto Cárdenas Isla
Ing. Ciencias Informáticas
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: risla@uci.cu

Co-Tutor: Ing. Evian Suárez Rodríguez
Ing. Ciencias Informáticas
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: esrodriguez@uci.cu

AGRADECIMINETOS

A mis padres por ser tan dedicados y preocupados por mis estudios y apoyarme en mis decisiones. Por su ejemplo de superación incasable, su comprensión, confianza y su amor incondicional.

A mi hermana por estar siempre presente, por la ayuda y apoyo que me dio.

A mi novia Sindy Mariam por siempre estar a mi lado, por entenderme en todo momento, por su dedicación, cariño, ternura y amor que siempre me ha dado. Te amo.

Gracias a toda mi familia: abuelos, tíos, y primos por sus consejos, la educación que me han dado.

A mi tutor Ernesto por atenderme siempre que lo necesité, impulsando mi formación como profesional.

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

A Frank Enrique que siempre estuvo ahí cuando lo necesitaba para sacarme de apuros y nunca dijo que no.

A mis suegros y en especial a mi suegra Vivian por ser tan buena conmigo y por su ayuda brindada.

Al aguerrido colectivo del apartamento 72101 por vivir juntos durante 4 años y en especial a Raúl, Ariel, Fidel, Reinier y Eduardo por compartir todos esos momentos divertidos que hemos pasado en la universidad.

A todos los que han ayudado a mi preparación integral para poder aspirar a graduarme como INGENIERO.

A los profesores y trabajadores de esta prestigiosa institución que me han acogido durante cinco años de estudio consagrado.

DEDICATORIA

Dedico mi trabajo de diploma a toda mi familia y en especial:

A mis maravillosos padres por enseñarme, guiarme en mi vida y por amarlos tanto.

A mis abuelos Aldo y Nélida por apoyarme en todo momento y contribuir de una forma u otra a lo que soy hoy en día.

A mi hermana por todo su cariño, preocupación, apoyo y ayuda.

RESUMEN

La investigación tiene como punto de partida lograr un mecanismo de comunicación entre el SAINUX Almiquí y cliente de visualización web XStormClient. El SAINUX Almiquí es una solución que recolecta los datos a partir del estándar OPC (Object linking and envidig for Process Control) y el visualizador web XStormClient brinda el mecanismo necesario para representación de información sumarios de puntos y despliegues. La unión de estas tecnologías provee las potencialidades de OPC con las bondades que brinda la web de accesibilidad, integridad y multiplataforma. La representación en la web de los datos recolectados mediante el estándar OPC mejoraría el acceso a la información desde diferentes plataformas, con una mejor accesibilidad y mayor integración con otros sistemas así como con otros proveedores.

Las herramientas y tecnologías para el desarrollo de la solución son: la metodología XP, el marco de trabajo Qt, el entorno de desarrollo integrado Visual Studio 2010, los lenguajes de programación C y C++, así como las especificaciones del protocolo de comunicación WebSocket y la utilización de la libwebsocket que implementa dicho protocolo.

El resultado de la investigación es una aplicación que permitió elevar los niveles de accesibilidad e integración en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient.

PALABRAS CLAVES: Cliente/Servidor, Comunicación OPC, Tecnologías de Comunicación, Websocket.

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad existe un gran desarrollo en las Tecnologías de la Información y las Comunicaciones (TIC). Unos de estos avances para el sector empresarial es la automatización de los procesos industriales con el que se permite ahorrar tiempo, dinero y recursos.

En Cuba hace algún tiempo se trabaja en la automatización de los procesos industriales, con el propósito de elevar los niveles de calidad y productividad. La Universidad de las Ciencias Informáticas (UCI) cuenta con varios centros que están desarrollando sistemas para la automatización de los servicios de empresas que necesitan la supervisión y control de procesos. En la Facultad 5 se encuentra el Centro de Informática Industrial (CEDIN) en el cual existen varios proyectos que manejan este negocio. La misión del CEDIN va enfocada al desarrollo de líneas de productos para la construcción de componentes reutilizables que optimicen el tiempo y costo de desarrollo, así como la buena planificación de los escasos recursos humanos.

Basado en esta estrategia se ha planteado lograr la integración de dos soluciones que faciliten el proceso de supervisión y control de procesos. La primera es el SAINUX Almiquí, que es una solución OPC[1] a problemas de intercambio de datos entre los componentes de un sistema de control que se encuentren sobre la plataforma Windows. Esta solución facilita el acceso a los datos desde dispositivos de campo, provee información estructurada, permite generación de variables y la comunicación con diferentes proveedores, pero presenta las limitantes de: solamente integrado con aplicaciones y proveedores que incorporen el estándar OPC; la accesibilidad se limita a dispositivos con un alto rendimiento del hardware y se complejiza al tener que instalar un cliente OPC para cada terminal donde se desee supervisar y controlar los procesos.

La otra solución es el visualizador web XStormClient, es capaz de supervisar los procesos industriales mediante tablas, sumarios de puntos y alarmas, así como despliegues; todo esto a partir de un fichero

[1] **OPC (Object linking and envidig for Process Control):** estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft.

XML[2] donde se refleja la configuración del SCADA Guardián del ALBA. Este sistema tiene como limitante que solamente visualiza la información referente a dicho SCADA, desechando la posibilidad de integración y acceso a otros sistemas o proveedores como es el caso del estándar OPC.

A partir de la situación antes planteada se define como **problema a resolver:** ¿Cómo contribuir a elevar la accesibilidad e integración en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient?

De modo que se define el **objeto de estudio**: El proceso de comunicación en sistemas de supervisión y control, identificando el **campo de acción:** La accesibilidad e integración en el proceso de comunicación para sistemas de supervisión y control.

Definiendo como **objetivo general:** Desarrollar una aplicación que permita elevar los niveles de accesibilidad e integración en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient.

Para guiar el desarrollo de esta investigación se proponen las siguientes tareas:

- Definición de los fundamentos teórico-metodológicos para el desarrollo de los mecanismos de comunicación web y las aplicaciones de escritorio.
- Caracterización del proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient basado en los niveles de accesibilidad e integración.
- Establecimiento de los fundamentos para la comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient.
- Desarrollo de una aplicación que comunique el sistema SAINUX Almiquí y el visualizador web XStormClient.

[2] **XML (Extensible Markup Language):** es un lenguaje de marcado sencillo. Su objetivo es facilitar la representación, almacenamiento y trasmisión de información por parte de aplicaciones informáticas, computadoras y medios de comunicación digital en general.

- Valoración de los resultados de la investigación en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient.

Métodos Teóricos:

- Método histórico-lógico: Permitió realizar un estudio y evolución de los procesos de comunicación así como las tecnologías de comunicación existentes para sistemas de supervisión y control y hacer una selección lógica para definir las tecnologías a utilizar.

- Método analítico-sintético: Se realizó un análisis de las tecnologías existentes resumiendo las características más importantes y sintetizando entonces aquellas que podrían proveer una solución a los mecanismos de comunicación de supervisión y control.

Métodos Empíricos

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

El presente documento estará estructurado de la siguiente forma:

- **Capítulo 1. Fundamentación Teórica.** Contiene los principales conceptos relacionados con el tema, así como los elementos teóricos que forman parte de esta investigación. Además se describen las herramientas y tecnologías a utilizar a partir del estudio del arte realizado.

- **Capítulo 2. Análisis y Diseño del sistema.** Se tratan las historias de usuario, la planificación de las iteraciones, la definición de la arquitectura y de las tarjetas CRC (Contenido, Responsabilidad y Colaboración) y el uso de los patrones de diseño.

- **Capítulo 3. Implementación y Pruebas.** Se describen las fases de la implementación del sistema, el estándar de codificación y las pruebas para verificar el funcionamiento del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se abordan conceptos fundamentales que ayudarán al entendimiento de la investigación. Se realiza un análisis de las tecnologías de comunicación existentes en la actualidad seleccionando la más factible para la solución del presente trabajo, así como la descripción de las tecnologías, herramientas y metodología a utilizar en el mismo.

1.2. Sistemas SCADA.

Los sistemas de Supervisión, Control y Adquisición de Datos (SCADA por sus siglas en inglés) son el software que permite el acceso a datos remotos de un proceso y el control del mismo, utilizando las herramientas de comunicación necesarias. Además se deduce que no se trata de un sistema de control, sino de una unidad de software de monitorización o supervisión que realiza la tarea de interfaces entre los niveles de control, donde se sitúan los controladores PLC[3] y los niveles superiores de gestión de los procesos. (1)

Además los sistemas SCADA utilizan las computadoras y tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales partiendo de 3 funcionalidades principales (2):

- **Adquisición de datos:** Para la recolección, el procesamiento y almacenamiento de la información recibida.
- **Supervisión:** Para observar desde un monitor la evolución de las variables de control.
- **Control:** Para modificar la evolución del proceso, actuando tanto sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc.), como directamente sobre el proceso, mediante las salidas conectadas.

[3] PLC **(Programmable Logic Controller):** dispositivo electrónico de control de procesos y se basa en una lógica, definida a través de un programa de computación.

1.2.1. Módulos del SCADA Guardián del ALBA (GALBA).

El SCADA GALBA es un convenio con la empresa de Petróleos de Venezuela SA (PDVSA). Este por su complejidad se desglosa en varios módulos y estos delegan cada una de sus funcionalidades. Los módulos son descritos a continuación:

- **Base de Datos Históricos:** Posee toda la información recibida desde el campo, así como la sucesión de alarmas y eventos generados.
- **Middleware:** Módulo encargado de la comunicación.
- **Seguridad:** Provee las interfaces necesarias para que los usuarios puedan autenticarse en el sistema, y de esta forma cada uno pueda acceder sólo a los recursos que tiene asignado su rol.
- **Ambiente de Configuración:** Configura varios procesos o partes de ellos, se definen y gestionan las variables, la configuración de los manejadores, los comandos, las alarmas y variadas opciones adicionales.
- **Visualización (HMI):** Visualiza los procesos operacionales con los que los usuarios interactúan.
- **Adquisición:** Es el encargado de leer y escribir sobre los dispositivos que conforman la red de campo, así como enviar el estado de las diferentes variables (puntos) que han sido configuradas a los módulos que lo requieren.

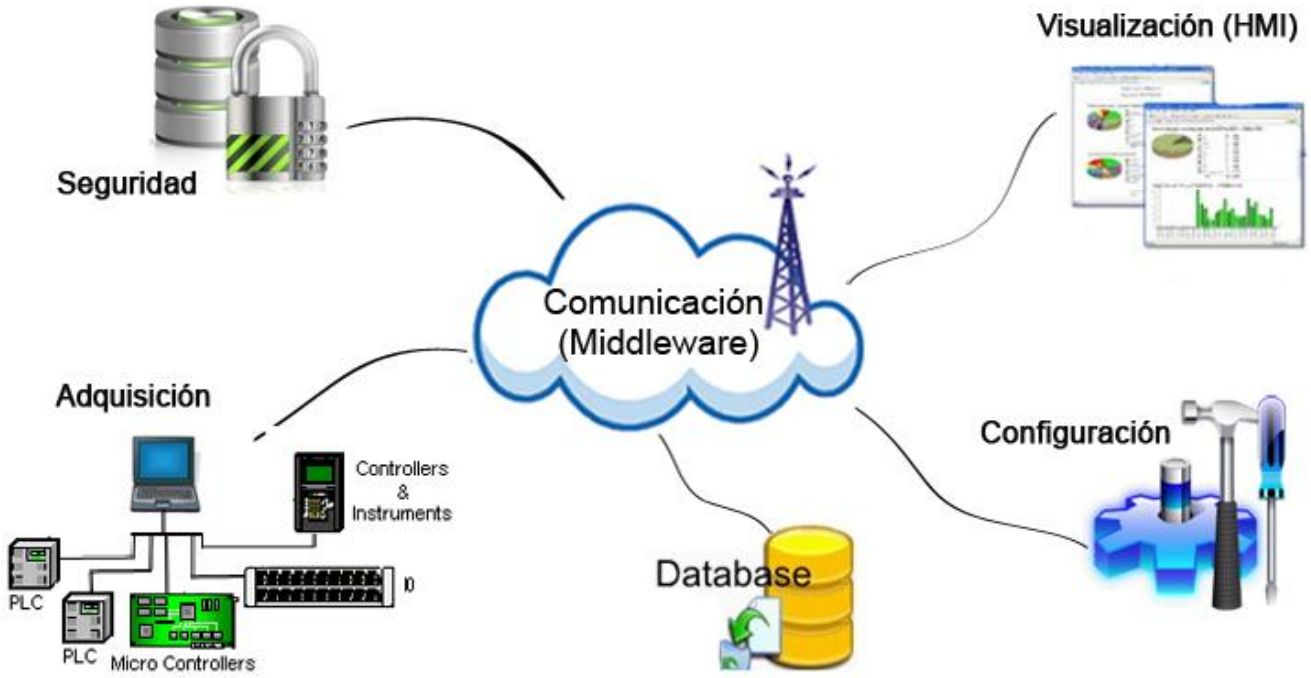

Ilustración 1. Módulos del SCADA GALBA. (Elaboración propia)

1.2.2. Módulo Interfaz Hombre-Máquina (HMI).

Un HMI proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de procesos y generados desde el editor incorporado al sistema SCADA. Tradicionalmente estos sistemas consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y análogos, registradores, pulsadores, selectores y otros que se interconectaban con la máquina o proceso. En la actualidad, dado que las máquinas y procesos en general están implementadas con controladores y otros dispositivos electrónicos que dejan disponibles puertas de comunicación, es posible contar con sistemas de HMI mucho más poderosos y eficaces, además de permitir una conexión más sencilla y económica con el proceso o máquinas. Las señales de los procesos son conducidas por medio de dispositivos: tarjetas de entrada/salida en la computadora, PLC o Controladores.

A partir de un estudio realizado las funciones de un software HMI son (3):

- **Monitoreo:** Permite obtener y mostrar los datos en tiempo real. Los mismos se pueden mostrar como números, textos, gráficos, permitiendo una lectura más fácil de interpretar.

- **Supervisión:** Permite junto con el monitoreo la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.

- **Alarmas:** Capacidad de reconocer eventos excepcionales dentro del proceso y reportar estos eventos. Las alarmas son reportadas basadas en límites de control pre-establecidos.

- **Control:** Es la capacidad de aplicar algoritmo que ajustan los valores del proceso y así mantener estos valores dentro de ciertos límites. Este va más allá del control de supervisión, removiendo la necesidad de la interacción humana. Sin embargo, la aplicación de esta función desde un software corriendo en una PC puede quedar limitada por la confiabilidad que quiera obtenerse del sistema.

- **Históricos:** Almacena en archivos, datos del proceso a una determinada frecuencia. Este almacenamiento de datos es una poderosa herramienta para la optimización y corrección de procesos.

El HMI del GALBA cuenta con dos ambientes: uno es el Ambiente de Configuración (AC), donde el usuario especifica las pantallas gráficas y los manejadores de comunicación los cuales permiten el enlace con los elementos de campo y la conexión en red. Todas estas configuraciones son utilizadas por el Ambiente de Ejecución (AE), que brinda al operador la supervisión y control de todos los procesos a través de sinópticos gráficos que cambian dinámicamente a múltiples formas y colores. El AE se divide en dos entornos: Entorno de Escritorio y el Entorno Web, en este último se enmarca el XStormClient como una solución para la visualización de los datos en la web.

1.3. XStormClient.

El visualizador web XStormClient permite supervisar y controlar los procesos industriales mediante tablas, despliegues, sumarios de puntos y alarmas; a partir de una configuración generada por el GALBA en un

archivo .XML. Este visualizador se comunica con XStormServer, el cual permite adquirir la información referente a los puntos y alarmas del SCADA GALBA. Su implementación se basa en el protocolo WebSocket para establecer la comunicación bidireccional entre el cliente y el servidor.

Cuenta con varios módulos que se utilizan para la autenticación de los clientes y la supervisión de los procesos a partir de las alarmas, puntos y despliegues. El módulo de autenticación permite a los clientes acceder al sistema en correspondencia con los diferentes niveles de privilegios. (Ver Ilustración 2)

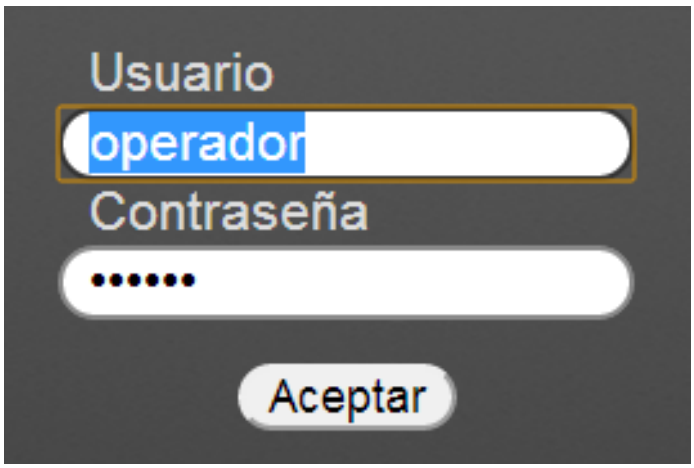

Ilustración 2. Módulo de autenticación

El módulo de los sumarios visualiza los distintos sumarios dígase: puntos digitales, puntos analógicos y alarmas. En el caso de los puntos analógicos y digitales muestra una tabla con los detalles: nombre, descripción, valor y calidad; en el sumario de alarmas muestra características como: id, fecha, recurso, causa, tipo, descripción, número de ocurrencia, prioridad y grupo. En la ilustración 3 se puede observar el sumario de los puntos analógicos.

Nombre	Descripcion	Valor	Grupo	Calidad
analogico21	analog21	80	GOP1	192
analogico22	analog22	42	GOP1	192
analogico23	analog23	61	GOP1	192
analogico24	analog24	76	GOP1	192
analogico25	analog25	12	GOP1	192
analogico26	analog26	60	GOP1	192
analogico27	analog27	53	GOP1	192
analogico28	analog28	2	GOP1	192
analogico29	analog29	96	GOP1	192
analogico30	analog30	84	GOP1	192
analogico31	analog31	80	GOP1	192
analogico32	analog32	42	GOP1	192
analogico33	analog33	61	GOP1	192
analogico34	analog34	76	GOP1	192
analogico35	analog35	12	GOP1	192
analogico36	analog36	60	GOP1	192
analogico37	analog37	53	GOP1	192
analogico38	analog38	2	GOP1	192
analogico39	analog39	96	GOP1	192
analogico40	analog40	84	GOP1	192
analogico41	analog41	80	GOP1	192
analogico42	analog42	42	GOP1	192
analogico43	analog43	61	GOP1	192
analogico44	analog44	76	GOP1	192

Ilustración 3. Sumario de puntos analógicos.

Existe un módulo que procesa las alarmas visualizando las últimas 5 que llegan al sistema, mostrando detalles como: fecha, recurso, causa, tipo, descripción, ocurrencia, prioridad y grupo (Ver Ilustración 4).

Fecha	Recurso	Causa	Tipo	Descripcion	Ocurrencia	Prioridad	Grupo
2013-5-27 =>23:00:37	Analogico-0	Alto-Alto	Valor	Valor Alto-Alto = 35	0	0	Admin
2013-5-27 =>23:00:37	Analogico-1	Alto-Alto	Valor	Valor Alto-Alto = 78	1	1	Admin
2013-5-27 =>23:00:37	Analogico-2	Alto-Alto	Valor	Valor Alto-Alto = 91	2	2	Admin
2013-5-27 =>23:00:37	Analogico-3	Alto-Alto	Valor	Valor Alto-Alto = 3	3	3	Admin
2013-5-27 =>23:00:37	Analogico-4	Alto-Alto	Valor	Valor Alto-Alto = 70	4	4	Admin

Ilustración 4. Últimas 5 alarmas

En el módulo de despliegues muestra los diferentes escenarios configurados por el SCADA GALBA, contenidos por gráficos que simulan los procesos. Los gráficos pueden ser relojes, cuadros de textos, medidores horizontales y verticales. (Ver Ilustración 5)

Ilustración 5. Ejemplo de un despliegue.

Este sistema cuenta con la limitante que solo se puede comunicar con el SCADA GALBA, desechando la posibilidad de integración y acceso a otros sistemas o proveedores como es el caso del estándar OPC.

1.4. SAINUX Almiquí.

El estándar OPC permite acceder a los datos desde dispositivos de campo. Se basa en la tecnología COM[4] de Microsoft, que permite definir cualquier elemento de campo mediante sus propiedades, convirtiéndolo en una interfaz. De esta manera es posible conectar fácilmente cualquier elemento de campo con un servidor de datos local COM o remoto DCOM[5] (4). Emplea la arquitectura Cliente/Servidor, lo que hace posible la comunicación entre elementos que cumplan con el estándar. Su arquitectura y

[4] **COM (Component Object Model)**: filosofía de programación de objetos orientados a interfaces para compartir entre aplicaciones.

[5] **DCOM (Distributed Component Object Model):** mecanismo que logra la evolución de la tecnología COM, para ser portada en redes de computadoras.

diseño nos brinda la posibilidad de construir un servidor OPC que permita a una aplicación cliente, acceder a los datos de muchos servidores OPC, proporcionados por diferentes proveedores. (Ver Ilustración 6)

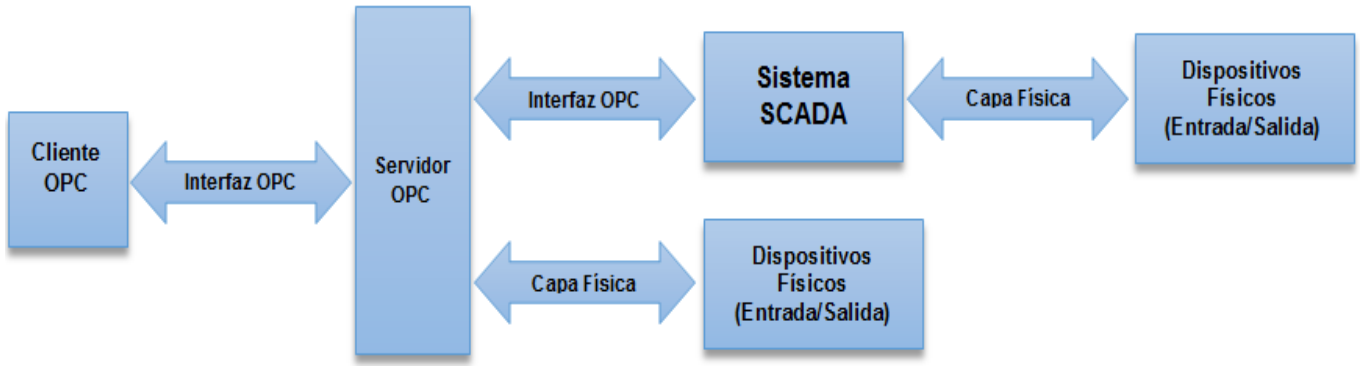

Ilustración 6. Arquitectura Cliente/Servidor OPC. (5)

OPC DA (Data Access) proporciona el acceso en tiempo real a los datos de manera consistente. Los servidores para cualquier elemento de campo proporcionan los datos de un formato único, de manera que el software y el hardware de control podrán intercambiar datos libremente.

SAINUX Almiquí implementa la solución OPC DA para mostrar los datos recolectados por el servidor Almiquí. Estos datos se visualizan en el cliente a través de un sumario donde el operador del sistema puede gestionar variables físicas a monitorear. A esta solución se le incorporó el HMI Almiquí, estructurado en dos subsistemas: el Explorador Almiquí, permite la gestión de recursos necesarios para la configuración del sistema, administra los usuarios que tendrán acceso, configura la comunicación con los diferentes proveedores y crea las pantallas con imágenes y texto que simulan el proceso a controlar, brindando a los usuarios una mejor comprensión de la aplicación creada. Toda esta configuración es guardada en un archivo .XML y utilizada por el subsistema Visualizador Almiquí, para la visualización de los procesos que se deseen controlar.

El sistema SAINUX Almiquí posee la limitante de que los datos solo pueden ser mostrados en el cliente OPC instalado en un ordenador, por lo que una actualización de la solución conllevaría a dar soporte a cada ordenador que tenga instalado el cliente y la accesibilidad se limita a dispositivos con un alto rendimiento del hardware.

1.5. Modelo Cliente/Servidor.

El modelo cliente/Servidor presenta dos funciones que pueden ser asumidas por los procesos de: el papel del usuario del servicio (cliente) y el papel de proveedor de servicios (servidor). La distribución de las funciones implica una asimetría en la ejecución distribuida de una aplicación, el servidor ofrece un servicio a la que uno o más clientes tienen acceso. (6)

Definiéndose así al cliente como el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, y el servidor como el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. (Ver Ilustración 7)

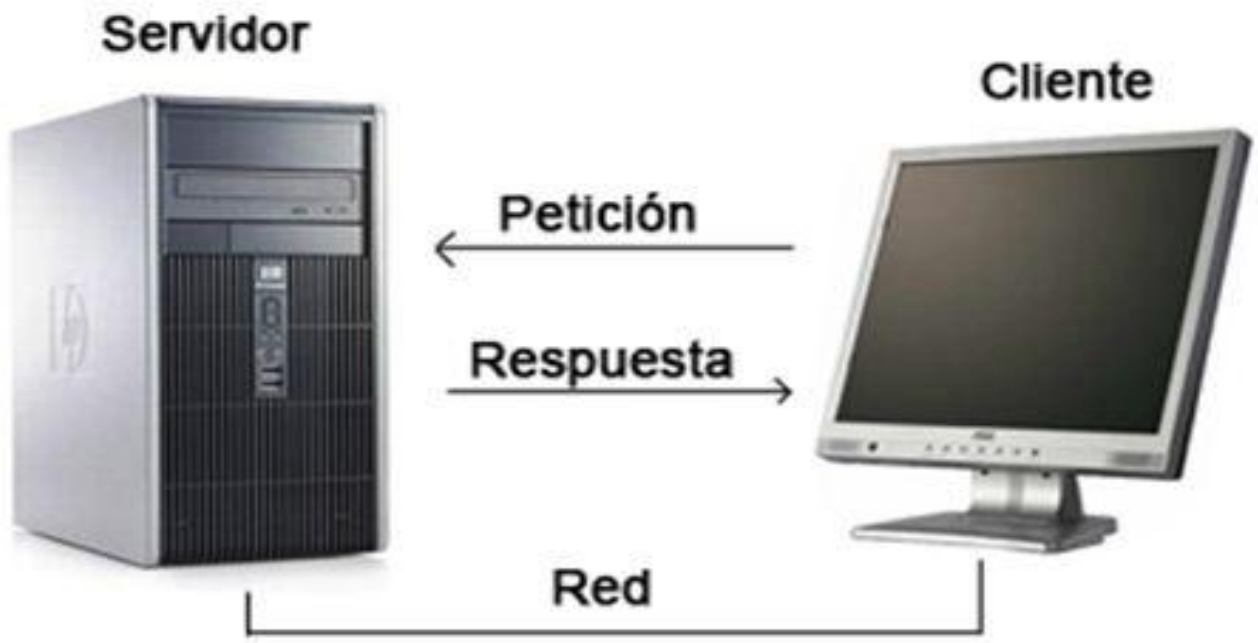

Ilustración 7. Modelo Cliente/Servidor. (Elaboración propia).

Una ventaja de este modelo es la división intuitiva de aplicaciones en una parte cliente y una parte servidor, basado en abstracciones convencionales, tales como la programación de procedimiento que simplifica el diseño y el desarrollo de aplicaciones distribuidas. También hace un uso eficaz de los recursos, cuando un gran número de clientes tienen acceso a un servidor de alto rendimiento. Otra de las ventajas es el potencial para la concurrencia. Todas estas ventajas también se podrían considerar

desventajas, por ejemplo, la restricción a los paradigmas de programación de procedimiento excluye otros enfoques como la programación funcional o declarativa. Por otra parte la concurrencia ya mencionada anteriormente como una ventaja, también puede conducir a problemas debido a su requisito de que se pueden sincronizar. (6)

1.6. Servidores Web.

Un servidor web es un simple servidor de archivos donde los clientes se dirigen a este mediante el protocolo HTTP[6] para obtener recursos. Cuando el servidor web recibe una petición HTTP, extrae simplemente de la petición el nombre del recurso solicitado, lo busca en el disco y lo envuelve dentro de una respuesta HTTP para transmitirlo al cliente. Un servidor no realiza ningún tratamiento en el recurso antes de transmitirlo al cliente. Por lo tanto puede transmitir de manera indiferente a un cliente una página HTML[7], una imagen, un archivo de sonido o incluso un archivo ejecutable. (7)

Se puede decir además que un servidor web se mantiene a la espera de peticiones HTTP por parte de un cliente HTTP conocido como navegador. El cliente realiza una petición al servidor y este le responde con el contenido que el cliente solicita. (Ver Ilustración 8)

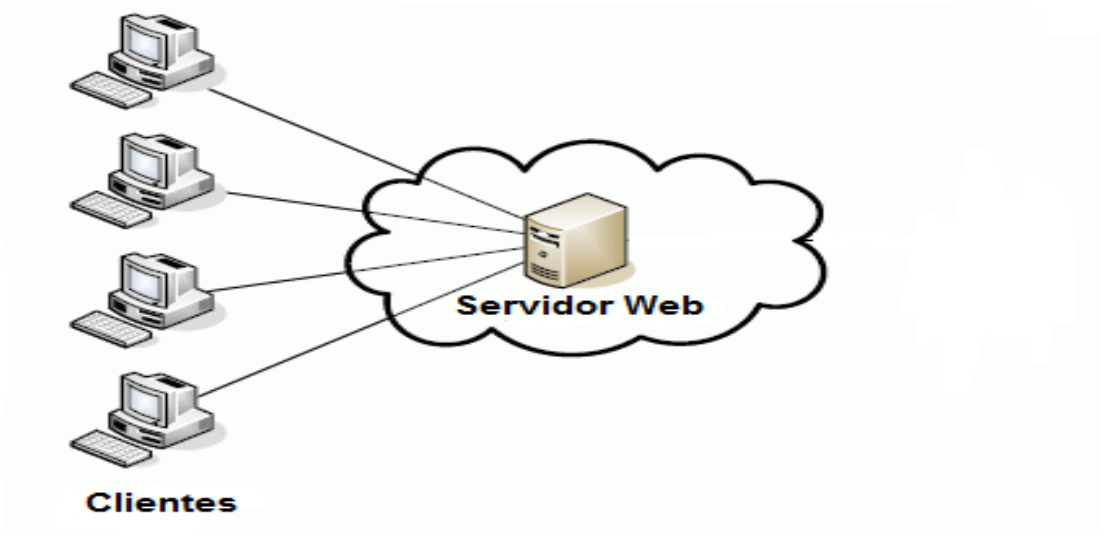

Ilustración 8. Servidor web. (Elaboración propia).

[6] **HTTP (Hypertext Transfer Protocol):** protocolo usado en cada transacción de la World Wide Web.
[7] **HTML (Lenguaje de Marcado de Hipertexto):** lenguaje utilizado para crear páginas web.

Se puede disponer de aplicaciones sobre el servicio web, estas son porciones de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Su funcionamiento consiste en ejecutar aplicaciones en el lado del cliente y del lado del servidor:

- **Aplicaciones en el lado del cliente:** Son las aplicaciones tipo Java o JavaScript, las cuales son ejecutadas por el cliente web mediante un navegador. Comúnmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje JavaScript y Java, aunque pueden añadirse más lenguajes como HTML y CCS.

- **Aplicaciones en el lado del servidor:** Es cualquier programa o conjunto de instrucciones diseñadas con la finalidad de que un servidor web las procese para realizar alguna acción. Las aplicaciones del lado del servidor están escritas mediante un lenguaje de programación, entre los que más se utilizan están PHP, Perl, Python y Ruby.

Son varios los servidores web utilizados en la actualidad, algunos se muestran en diferentes plataformas, otros tienen una mayor respuesta de rendimiento y contienen tecnologías de comunicación los cuales a partir de un estudio se destacan:

- **Apache:** Es una herramienta de código abierto, creado sobre los principales sistemas operativos (Unix y Windows) haciendo frente a los servidores web propietarios de mayor uso en el mercado. Proveen requerimientos de seguridad, eficiencia, extensibilidad y estandarización. Entre sus potencialidades se encuentra que está escrito en C por lo que le da un mayor rendimiento, velocidad de respuesta. Se encuentran actualmente entre los servidores web líderes del mundo. (8)

- **Lighttp:** Diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Consume pocos recursos del sistema como son velocidad procesamiento, capacidad de almacenamiento en la memoria RAM. Apropiado para cualquier servidor que tenga problemas de carga. Funciona en GNU/Linux y Unix de forma oficial y en Microsoft Windows actualmente hay una distribución conocida como Lighttpd For Windows. (9)

Aunque **Node.js** no entra dentro del ámbito de los servidores web es bueno destacarlo porque es una fuerte librería escrita en JavaScript que nos brinda una API[8] para crear servidores web. Es un entorno del lado de servidor que utiliza un modelo asíncrono y dirigido por eventos. Posee bondades como: el soporte de protocolos TCP[9], DNS[10] y HTTP, las capacidades para I/O (Entrada/Salida) son realmente ligeras y potentes. Además crea solo un hilo de procesos para todos los clientes lo que hace que el servidor soporte muchas más conexiones. (10)

1.7. Evaluación de las tecnologías de comunicación.

Las Tecnologías de comunicación son el conjunto de tecnologías desarrolladas para gestionar información. Abarcan un abanico de soluciones muy amplio, incluyen las tecnologías para almacenar información y recuperarla después, enviar y recibir información de un sitio a otro, o procesar información para poder calcular resultados y elaborar informes. En este epígrafe abordaremos sobre algunas tecnologías de comunicación con el objetivo de seleccionar la que se utilizará en la construcción de la solución.

Server-Sent Events (SSE)

Este protocolo de comunicación es un modelo petición-respuesta, lo cual significa que el cliente envía la solicitud HTTP y espera hasta que la respuesta HTTP se reciba. Esto trae consigo que solo el servidor puede comunicarse con el cliente si este lo solicita. HTML5[11] proporciona una forma nativa para controlar los eventos enviados del servidor, define la API SSE para la apertura de una conexión HTTP para recibir notificaciones enviadas desde un servidor. Su funcionamiento comienza cuando el cliente abre una secuencia de eventos que tiene una dirección URL[12] de origen de evento como parámetro. El controlador de eventos se llamará cada vez que el origen genere un nuevo dato. (11)

[8] **API (Application Programming Interface):** 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.
[9] **TCP (Transmission Control Protocol):** Protocolo de comunicación orientado a la conexión fiable del nivel de transporte.
[10] **DNS (DomainName System):** Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.
[11] **HTTP5 (HyperText Markup Language, versión 5):** Es la quinta revisión importante del lenguaje básico de la World Wide Web.
[12] **URL (Uniform Resource Locator):** Es una cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en la Internet.

En resumen SSE está orientada a crear un protocolo de comunicación unilateral, mediante el cual se envían datos desde el servidor al cliente.

XMPP

El Protocolo Extensible Messaging and Presence (XMPP) es una tecnología abierta para interacción en tiempo real. Es seguro ya que ofrece soporte incorporado para el cifrado de canal y fuerte autenticación, la resistencia inherente a muchas formas de malware y un ecosistema diverso de las implementaciones. A diferencia de los hilos de comunicación independientes, las tecnologías XMPP se despliegan en un entorno descentralizado arquitectura Cliente/Servidor con un número ilimitado de número de servidores. Es extensible, porque XMPP es en su esencia una tecnología para la rápida entrega XML a partir de un lugar a otro, se ha utilizado para una amplia gama de aplicaciones más allá de la mensajería instantánea, incluyendo juegos y redes sociales. (12)

WebSocket

WebSocket es una especificación del estándar HTML5, orientada a crear un protocolo de comunicación bilateral, mediante el cual se envían datos desde el servidor al cliente web y viceversa. HTML5 Web Sockets pone a disposición del programador una conexión de socket, a través de Internet, con una sobrecarga mínima, proporcionando una enorme reducción del tráfico innecesario en la red. Para conectarse desde cliente web a un servidor utilizando HTML5 Web Sockets, se crea una instancia nueva WebSocket y se le pasa la dirección URL del servidor. Una conexión WebSocket se establece mediante la actualización del protocolo HTTP, al protocolo Secure Socket Web, durante la conexión inicial entre el cliente y el servidor, sobre la misma conexión subyacente TCP/IP. (13)

Dentro de las ventajas de WebSocket se encuentra una reducción en el consumo de la red de las aplicaciones tanto del lado del cliente como del lado del servidor, se tiene un marco estandarizado para el desarrollo de bases de datos del cliente, posee una comunicación bidireccional con WebSocket e hilos del lado del cliente. Una de las desventajas es que todavía está en desarrollo y no todos los navegadores la soportan.

Para la selección de las tecnologías de comunicación, se enfatizó en las características que poseía cada una: comunicación bidireccional, latencia y seguridad, valorando en todo momento cual se ajustaba más al problema a resolver. En estudios realizados el protocolo XMPP arrojaba que el tiempo de envío de los datos del mismo era superior al de SSE y WebSocket, aunque soportaba mayor seguridad, sin embargo SSE tenía tiempos similares a los de WebSocket pero su comunicación no era bidireccional. (14)

Partiendo del análisis anterior, incluyendo que el cliente web esta implementado sobre WebSocket y que la incorporación de una nueva tecnología sería una dependencia, se realizará una solución basada en la tecnología WebSocket para la comunicación con el estándar OPC. Dicha tecnología soporta el envío de grandes cantidades de datos en corto tiempo, múltiples conexiones, su comunicación es bidireccional y posee una seguridad aceptable.

1.8. Herramientas y Tecnologías.

1.8.1. Libwebsocket.

Esta biblioteca presenta características fundamentales como: escrita en lenguaje C y bajo la licencia LGPL[13], posee alto rendimiento para la comunicación bidireccional, es ligera en tamaño y está optimizada para el manejo de concurrencia.

Partiendo de la experiencia en la implementación de servidores utilizando libwebsocket (14), se decidió implementar el servidor XStormAlmiquiServer usando esta biblioteca, de forma tal que el cliente web XStormClient se reutilizara completamente y solo se modificara la lógica de negocio del servidor que brinda la información, en este caso OPC.

1.8.2. Entorno de Desarrollo Integrado (IDE).

Visual Studio 2010 soporta el uso de marcos de trabajo para el desarrollo de aplicaciones, se puede utilizar como un sistema de desarrollo integrado o como un conjunto de herramientas individuales, entre las que se encuentra Microsoft Visual C++. Este módulo proporciona un gran número de beneficios para el desarrollo de software, así como los mecanismos para interactuar con objetos COM y DCOM de

[13] **LGPL (Lesser General Publical License):** Licencia que permite el desarrollo de software privativo o comerciable.

automatización industrial y la utilización de la biblioteca ATL, que provee un conjunto de funciones para los desarrolladores de .NET.

1.8.3. Lenguajes de Programación.

El propósito general del lenguaje de programación C es que ofrece ventajas como: la agrupación de instrucciones, incluye el concepto de puntero (variable que contiene la dirección de otra variable), no es un lenguaje de muy alto nivel, no está ligado a ningún hardware o sistema en particular y es fácil escribir programas que correrán sin cambios en cualquier máquina que maneje dicho lenguaje (15). Además la biblioteca a utilizar es libwebsocket la cual está basada en este lenguaje.

Se utilizó el lenguaje C++ por el uso del Framework Qt que está basado sobre C/C++, para el trabajo con los punteros, polimorfismo y herencia, además es uno de los más empleados en la actualidad. El mismo es un lenguaje híbrido, ya que permite programar tanto en estilo procedimental o en estilo orientado a objeto. Provee ventajas como la existencia de una gran cantidad de información acerca de este, de propósito general por lo que se puede emplear para solucionar cualquier tipo de problemas, está estandarizado y un mismo código fuente se puede compilar en diversas plataformas y es uno de los lenguajes más rápidos en cuanto a ejecución (16).

1.8.4. Marco de Trabajo.

Qt es un marco de trabajo multiplataforma, que se utiliza para el desarrollo de aplicaciones. Cuenta con un conjunto de bibliotecas que facilitan el trabajo con base de datos, ficheros, gráficos, tratamiento de imágenes, etc. Está desarrollado en C++ y se puede integrar con distintos IDE existentes, como es el caso de Eclipse, Microsoft Visual Studio, entre otros. Está disponible para sistemas operativos como Microsoft Windows, Linux, Mac OS X, bajo licencias GPL[14] v1, GPL v2 y GPL v3. Además proporciona el módulo QXml el cual contiene las clases QXmlStreamReader para leer y QXmlStreamWriter para escribir en un XML, por lo que facilita el trabajo de lectura y escritura con XML. Qt nos ofrece un fácil manejo a la hora de convertir los datos en Json, de ahí el uso de QByteArray para el tratamiento eficiente de cadenas de

[14] **GPL (General Public License):** Licencia orientada principalmente a proteger la libre distribución, modificación y uso de Software.

texto con dinamismo. Además la línea HMI del SCADA GALBA guía el desarrollo de soluciones basado en este framework.

1.8.5. Metodología XP.

Se utiliza XP por ser 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. 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, donde existe un alto riesgo técnico. (17)

Historias de Usuario: Son utilizadas por el cliente para describir en ellas las características que el sistema debe poseer. El tratamiento con las mismas es muy dinámico porque en cualquier momento se pueden reemplazar por otras más específicas y generales. Cada una de ellas es lo suficientemente comprensible para que los programadores puedan implementarla sin que existan problemas. (18)

Los roles de acuerdo con la propuesta original de Beck son: (17)

- **Programador:** Escribe las pruebas unitarias y produce el código del sistema.
- **Cliente:** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (Tester):** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker):** Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

- **Entrenador (Coach):** Responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor:** Miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (Big boss):** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

1.8.6. Herramienta de Modelado.

Visual Paradigm para UML[15] es una herramienta UML 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 costo. 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. Además es una política definida por el CEDIN utilizar esta herramienta para la generación de artefactos y modelados de procesos de desarrollo.

1.9. Conclusiones parciales:

En este capítulo se hizo alusión a los conceptos fundamentales que ayudó al desarrollo de la solución. Se analizaron las tecnologías de comunicación que permiten el tráfico de información en la red con el objetivo de desarrollar un servidor que permita el envío de información desde el SAINUX Almiquí al XStormClient, llegando a la conclusión que la tecnología a emplear es WebSocket, utilizando la biblioteca libwebsocket. Se definieron los lenguajes de programación C y C++, XP como metodología de desarrollo, el marco de trabajo QT y el entorno de desarrollo integrado Visual Studio 2010.

[15] **UML (Unified Modeling Language):** Lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.1. Introducción.

EL siguiente capítulo tiene como objetivo hacer una descripción del análisis y diseño de la solución siguiendo como guía la metodología de desarrollo XP. Se realizan las historias de usuario las cuales fueron escritas por el cliente, la planificación de las iteraciones, la definición de la arquitectura y de las tarjetas CRC (Contenido, Responsabilidad y Colaboración) y el uso de los patrones de diseño.

2.2. Análisis de la solución propuesta.

A partir de la problemática planteada con anterioridad y el análisis de las herramientas a utilizar para la creación de una aplicación que permita elevar los niveles de accesibilidad e integración en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient, se establecieron los siguientes requisitos.

Requisitos Funcionales:

- RF1. Iniciar el Servidor.
- RF2. Detener Servidor.
- RF3. Enviar la Configuración del Proyecto.
- RF4. Enviar los Puntos de un Despliegue.
- RF5. Autenticar Usuario.
- RF6. Enviar Sumarios de Punto.
- RF7. Enviar Detalles de un Punto.

2.3. Exploración.

XP propone que en esta fase se defina el alcance general del proyecto. El cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado. (19)

2.3.1. Actores del sistema.

Actores	Descripción
Administrador	Persona capaz de configurar, iniciar y detener el servidor.
Cliente de Visualización	Agente externo que se comunica y a su vez puede realizar peticiones al servidor.

Tabla 1. Actores del sistema.

2.3.2. Historia de usuario.

Las historias de usuarios (HU) sustituyen a los documentos de especificación funcional, y a los casos de uso las mismas son usadas para la especificación de los requisitos. Son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. Pueden ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia. (19)

Historia de usuario	
Número: 1	**Nombre:** Iniciar el servidor.
Usuario: Administrador	
Modificación de Historia Número:	**Iteración asignada:** 1
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:**1
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos reales:**1

Descripción: El sistema debe permitir al administrador poder iniciar el servicio.
Observaciones: Una vez inicializado el sistema este espera por las peticiones del cliente.

Tabla 2. Iniciar el servidor.

Historia de usuario	
Número: 2	**Nombre:** Detener el servidor.
Usuario: Administrador	
Modificación de Historia Número:	**Iteración asignada:** 1
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:** 1
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos reales:**1
Descripción: El sistema debe permitir al administrador poder detener el servicio.	
Observaciones: El sistema detiene cada uno de los servicios que este brinda.	

Tabla 3. Detener el servidor.

Historia de usuario	
Número: 3	**Nombre:** Enviar la configuración del proyecto.
Usuario: Cliente de Visualización	
Modificación de Historia Número:	**Iteración asignada:** 1
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:** 1
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos reales:** 1
Descripción: El cliente de visualización pide la configuración del proyecto y a su vez el servidor debe ser capaz de poder atenderla y enviar la información necesaria.	
Observaciones:	

Tabla 4. Enviar la configuración del proyecto.

Historia de usuario	
Número: 4	**Nombre:** Enviar los puntos de un despliegue.
Usuario: Cliente de Visualización	
Modificación de Historia Número:	**Iteración asignada:** 2
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:**2
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos reales:**2
Descripción: El cliente de visualización realiza una petición de los puntos de un despliegue y a su vez el servidor debe atenderla y enviar la información.	
Observaciones:	

Tabla 5. Enviar los puntos de un despliegue.

Historia de usuario	
Número: 5	**Nombre:** Autenticar Usuario.
Usuario: Cliente de Visualización	
Modificación de Historia Número:	**Iteración asignada:** 2
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:** 1
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos reales:** 1
Descripción: El Cliente de Visualización realiza la petición para la autenticación del usuario y el sistema debe ser capaz de verificar si los datos son válidos y enviar una respuesta al cliente.	
Observaciones:	

Tabla 6. Autenticar usuario.

Historia de usuario	
Número: 6	**Nombre:** Enviar sumarios de puntos.
Usuario: Cliente de Visualización	
Modificación de Historia Número:	**Iteración asignada:** 3

Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:**2
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos reales:**2
Descripción: El Cliente de Visualización pide el sumario de puntos y el servidor debe poder atenderla y enviar cada uno de los detalles de todos los puntos (nombre, descripción, valor, grupo operacional, calidad).	
Observaciones:	

Tabla 7. Enviar sumarios de puntos.

Historia de usuario	
Número: 7	**Nombre:** Enviar detalles de un punto.
Usuario: Cliente de Visualización	
Modificación de Historia Número:	**Iteración asignada:** 3
Prioridad del Negocio: Alta (Alta/Media/Baja)	**Puntos estimados:** 1
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos reales:** 1
Descripción: El Cliente de Visualización realiza el pedido del detalle de un punto y el servidor debe atenderla y enviar todos los detalles referentes al punto.	
Observaciones:	

Tabla 8. Enviar detalles de un punto.

2.4. Planificación y entrega.

La fase de planificación y entrega es de poca duración, en la misma el cliente decide la prioridad de cada historia de usuario a su vez los programadores realizan un análisis del esfuerzo de cada una de ellas. Se toman acuerdos donde se tienen en cuenta el contenido de la primera entrega y se determina un cronograma conjuntamente con el cliente, cada una de estas entregas debe obtenerse en menos de tres meses.

2.4.1. Plan de entrega.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	El objetivo de esta iteración es darle cumplimiento a estas 3 historias de usuarios las cuales son de gran importancia para la aplicación.	1 ,2 y 3	3
2	El objetivo de esta iteración es darle cumplimiento a estas 2 historias de usuarios las cuales son de gran importancia. Las mismas permiten la autenticación del usuario y la visualización de los puntos de un despliegue en tiempo real.	4 y 5	3
3	El objetivo de esta iteración es darle cumplimiento a estas 2 historias de usuarios las cuales son de gran importancia para la aplicación y permite el trabajo con los puntos.	6 y 7	3

Tabla 9. Plan de entrega.

2.4.2. Estimación del esfuerzo por historia de usuario.

Historias de Usuario	Tiempo estimado (semana ideal de trabajo)	Iteración asignada	Tiempo real
Iniciar servidor. Detener servidor. Enviar la configuración del proyecto.	3	1	3
Enviar los puntos de un despliegue. Autenticar usuario.	3	2	3
Enviar sumario de puntos. Enviar detalles de un punto.	3	3	3

Tabla 10. Estimación del esfuerzo por historia de usuario.

2.5. Diseño del sistema.

XP no requiere la utilización de diagramas de clases utilizando notación UML a la hora de presentar un sistema, sino se guía por el uso de técnicas como las tarjetas CRC, pero esto no quiere decir que no puedan aplicarse estos diagramas siempre que sea para el mejoramiento de la comunicación, no sean extensos y su complejidad no sea muy elevada.

2.5.1. Estructura del sistema.

En este epígrafe se explicará de una forma más detallada la estructura del sistema para un mejor entendimiento del mismo. Para esto se realizó un diagrama de paquetes el cual está conformado por 5 paquetes los mismos son: (Ver Ilustración 9).

- **OPC:** Encargado de la adquisición de los puntos del servidor OPC.
- **Protocolo:** Contiene las distintas funciones de llamada y retorno destinadas a la comunicación con el protocolo WebSocket.
- **Sistema Base:** Se encuentran todas las clases manejadoras del sistema.
- **Entidades:** Contiene las entidades del sistema.
- **Útiles:** Contiene las clases que realizan acciones secundarias para el sistema.

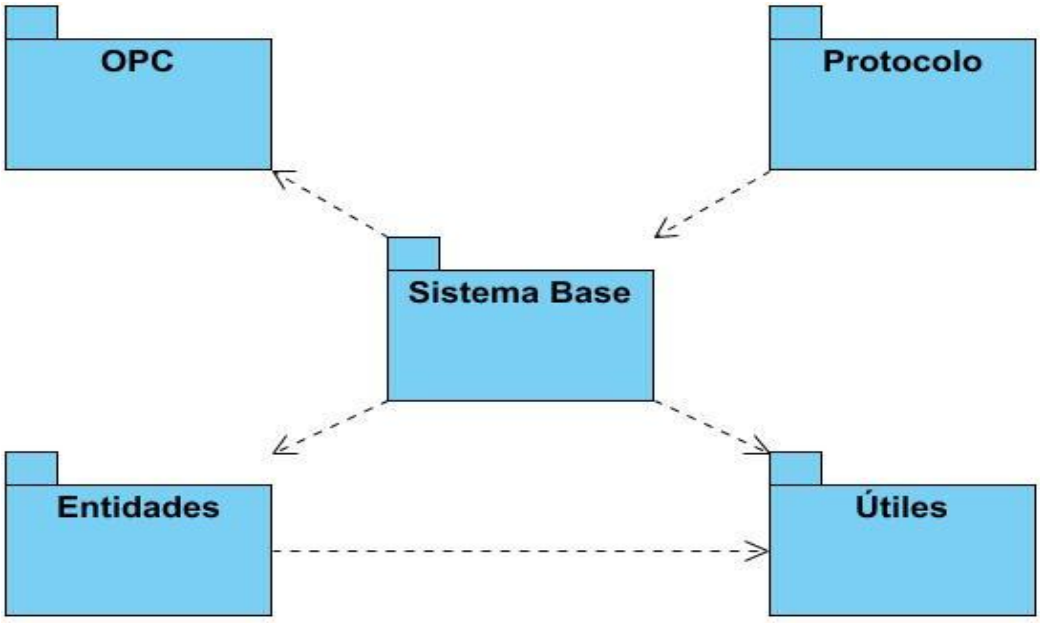

Ilustración 9. Diagrama de paquetes. (Elaboración propia).

Se realizó además un diagrama de despliegues donde se podrá observar el funcionamiento físico del sistema. (Ver Ilustración 10).

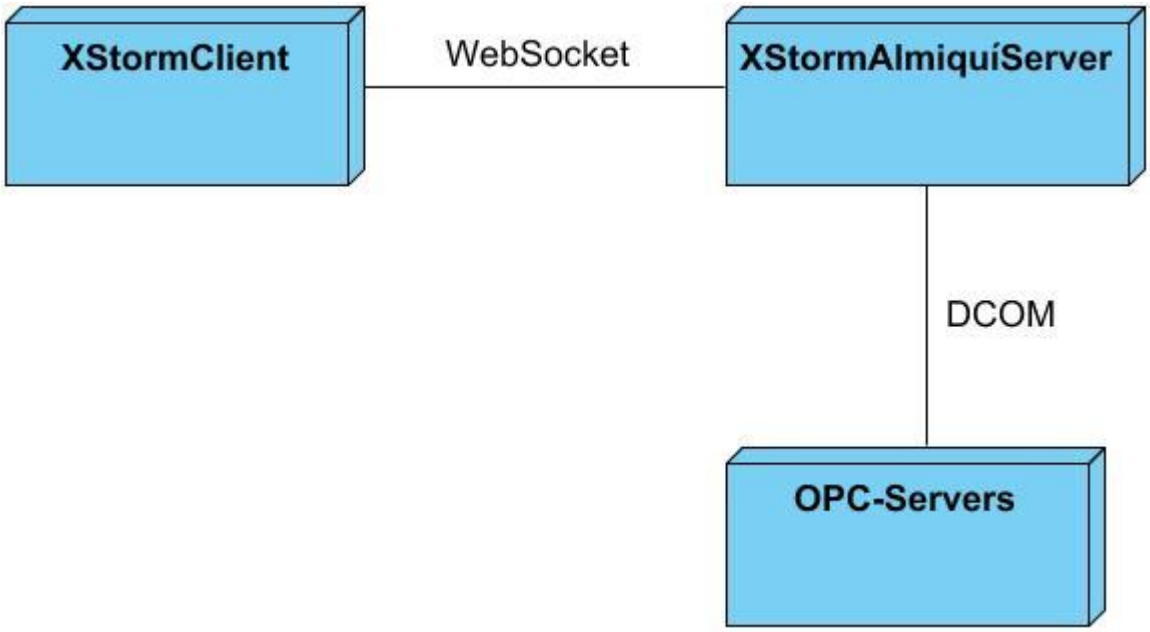

Ilustración 10. Diagrama de despliegue. (Elaboración propia).

2.5.2. Tarjeta CRC (Cargo o clase, Responsabilidad y Colaboración).

Las tarjetas CRC nos permiten trabajar con una metodología la cual está basada en objetos y que el equipo completo contribuya en la tarea del diseño, su principal objetivo es saber cuáles son las clases que necesitarán para implementar el sistema.

Tarjeta CRC	
Clase: Nombre de la clase.	
Responsabilidades	**Colaboraciones**
Responsabilidades de la clase.	Clases colaboradoras.

Tabla 11. Modelo de tarjeta CRC.

El sistema está compuesto por una serie de paquetes que conforman su arquitectura, a continuación se describirán los mismos y las tarjetas CRC de las clases que los integran.

OPC: Este paquete contiene la clase OPCManager la cual posee un mecanismo de adquisición de puntos del SAINUX Almiquí.

Tarjeta CRC	
Clase: OPCManager.	
Responsabilidades	**Colaboraciones**
Se encarga de conectarse con la biblioteca OPCDAClient para ir obteniendo el lote de puntos.	Clase QList Clase QWaitCondition Clase Singleton Clase System

Tabla 12. Tarjeta CRC para la clase OPCManager.

Sistema Base: En este paquete se encuentran las clases mediante las cuales se controla el sistema, contiene las clases que manejan los puntos y la configuración.

Tarjeta CRC	
Clase: ConfigManager	
Responsabilidades	**Colaboraciones**
Es la que se encarga de cargar las configuraciones contenidas en los archivos (.XML).	Clase QByteArray Clase QFile Clase iostream Clase Screen Clase PointData Clase Grupo Clase OPCLayer Clase QDomNodeList Clase ServerData

Tabla 13. Tarjeta CRC para la clase ConfigManager.

Tarjeta CRC	
Clase: PointManager	
Responsabilidades	**Colaboraciones**
Es la clase que se encarga del manejo de puntos.	Clase QList Clase PointData Clase System Clase Serializer

Tabla 14. Tarjeta CRC para la clase PointManager.

Tarjeta CRC	
Clase: System	
Responsabilidades	**Colaboraciones**
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.	Clase PointManager Clase ConfigManager Clase Singleton Clase OPCManager Clase QMutexLocker

Tabla 15. Tarjeta CRC para la clase System.

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

Tarjeta CRC	
Clase: GraphicComponent	
Responsabilidades	**Colaboraciones**
Contiene los datos de un componente gráfico.	Clase QMap Clase QList Clase QPixmap Clase QXmlStreamReader

Tabla 16. Tarjeta CRC para la clase GraphicComponent.

Tarjeta CRC	
Clase: PointData	
Responsabilidades	**Colaboraciones**
Contiene los datos de un punto y los métodos necesarios para acceder a sus atributos.	Clase QString Clase SummaryDataRow Clase QXmlStreamReader

Tabla 17. Tarjeta CRC para la clase PointData.

Tarjeta CRC	
Clase: Screen	
Responsabilidades	**Colaboraciones**
Contiene los datos de una pantalla o despliegue.	Clase QList Clase GraphicComponent

Tabla 18. Tarjeta CRC para la clase Screen.

Útiles: En este paquete se encuentran las clases que realizan acciones secundarias para el sistema.

Tarjeta CRC	
Clase: Serializer	
Responsabilidades	**Colaboraciones**
Esta clase contiene métodos capaces de construir cadenas con la información de los puntos, 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.	Clase Qjson_export Clase QIODevice Clase QString Clase QVariant Clase QDataStream Clase QStringList Clase QVariant

Tabla 19. Tarjeta CRC para la clase Serializer.

Tarjeta CRC	
Clase: Singleton	
Responsabilidades	**Colaboraciones**
Esta clase es usada para garantizar una única instancia global de la clase System.	

Tabla 20. Tarjeta CRC para la clase Singleton.

2.5.3. Funcionalidades del paquete Protocolo.

El paquete contiene las diferentes funciones de llamas de retorno, destinadas a la comunicación WebSocket entre el XStormClient y el XStormAlmiquiServer mediante el uso de la libwebsocket. Cada una de las llamadas especifica la forma de actuar del XStormAlmiquiServer para cada una de las peticiones hechas por el XStormClient, el cual implementa las mismas funciones para poder establecer una comunicación bidireccional. A continuación se muestra la estructura que define la biblioteca cuando se van a implementar estas funciones.

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.

Tabla 21. Estructura de las llamadas de retorno.

A partir de esta estructura y las historias de usuario se definieron las llamadas de retorno siguientes:

- **callback_http:** Permite conocer el nombre del ordenador cliente así como su dirección ip, para tener un control sobre el acceso al servidor.
- **callback_security:** Es la encargada de manejar la petición de autenticar usuario mediante usuario y una contraseña.
- **callback_runtime:** Maneja todas las peticiones que realice el cliente de visualización acerca de la información de los puntos (valor de los puntos de un despliegue, detalles de un punto y sumario de puntos).

2.5.4. Patrones de diseño.

Para la construcción de la solución se emplea el patrón Singleton para lograr una única instancia global de la clase System. Dentro de las características que nos brinda este patrón se encuentran: acceso a la instancia única controlado, reduce el espacio de nombre frente al uso de variables globales, es más flexible que la alternativa de las “operaciones de clase”, además de que éstas en C++, al ser función miembro estáticas, no son virtuales, luego no pueden ser especializadas mediante herencia ni redefinidas polimórficamente. (20)

Además se utiliza el patrón Iterator para garantizar una gran comodidad a la hora de trabajar con los puntos. Algunas de sus ventajas son: incremento de la flexibilidad, dado que para permitir nuevas formas de recorrer una estructura basta con modificar el Iterator en uso, cambiarlo por otro o definir uno nuevo. Se enfatiza la distribución de responsabilidades (y, por ello, la simplicidad de los elementos del sistema), por lo que la interfaz del agregado se ve libre de una serie de operaciones, más propias de la interfaz del Iterator. (20)

2.6. Conclusiones parciales.

A partir de la determinación de los requisitos se definieron las 7 historias de usuarios, las cuales guiaron al proceso del diseño de las tarjetas CRC, realizándose a su vez el plan de entrega para determinar un cronograma conjuntamente con el cliente. Se diseñó el diagrama de paquetes y el diagrama de despliegue para un mejor entendimiento de la solución planteada y se identificaron un conjunto de patrones de diseño que proveen una mejor comunicación entre el equipo de desarrollo y posibles soportes futuros.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción.

En el presente capítulo se detalla la fase de implementación y se ejecutan las pruebas elemento importante para la culminación del ciclo de desarrollo de la solución. La fase de implementación expone la culminación cabal de las iteraciones y las pruebas se ejecutan con el objetivo de saber si los resultados obtenidos son los esperados.

3.2. Estándares de Codificación.

Los estándares de codificación son pautas o convenios que no se enfocan en la lógica del programa, sino que se centran en la escritura del código. Estos estándares sirven como punto de referencia a los programadores para que se les facilite entender su código. En este trabajo se emplean los estándares de codificación para C++ del autor Ariel Chávez Lorenzo, por el que se rige el proyecto SCADA Guardián del Alba. (21)

3.3. Desarrollo de Iteraciones.

Las distintas funcionalidades del sistema son desarrolladas en la presente fase. Una vez terminada cada iteración se genera un entregable funcional, el cual implementa las historias de usuario que hayan sido definidas para esa iteración. Al comienzo de cada iteración se plantean las tareas de ingeniería, siempre teniendo en cuenta todos los detalles necesarios con el cliente. En la primera iteración se puede llegar a plantear una arquitectura del sistema la cual puede ser utilizada en el transcurso del proyecto. Para poder lograr esta arquitectura se seleccionan las historias de usuario las cuales puedan forzar su creación, aunque la creación de esta arquitectura no siempre es posible ya que el cliente es el que decide cuáles serán las historias de usuario que se implementarán en cada una de las iteraciones, debido a que así se puede maximizar el valor del negocio. Mediante las iteraciones se puede medir el avance del proyecto, una vez terminadas todas las iteraciones el sistema está en condiciones para entrar en la producción. (22)

3.3.1. Implementación.

Una vez asignadas las HU a las iteraciones correspondientes estas se van desarrollando, al comenzar una iteración se lleva a cabo una revisión del plan de iteraciones y en caso de ser necesario se le realiza

algún cambio. Como parte del plan las HU se descomponen en tareas de ingeniería y se asignan a un grupo o persona responsable de la implementación. Estas tareas pueden ser escritas en lenguaje técnico y no necesariamente entendible ya que solamente serán usadas por el programador.

A continuación se describen las tareas de ingeniería realizadas en cada una de las iteraciones:

Iteración 1.

La primera iteración tiene como objetivo desarrollar las HU 1, 2, 3.

Historia de Usuario	**Tiempo de implementación (semanas)**	
	Estimación	**Real**
Iniciar servidor.	1	1
Detener servidor.	1	1
Enviar la configuración del servidor.	1	1

Tabla 22. HU abordadas para la primera iteración.

Tareas de las HU desarrolladas para la primera iteración.

Una vez relacionadas las HU con las iteraciones correspondientes se procede a la especificación de las principales tareas que se realizaron para el cumplimiento de la iteración. (Ver Tabla 24 y 25)

Tarea de ingeniería	
No. de la Tarea: 1	**No. de la HU:** 1 ,2
Nombre de la Tarea: Definir e implementar como iniciar y detener el servidor.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**: 2
Fecha de inicio:25/2/2013	**Fecha fin:** 8/3/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Permitir iniciar y detener el servidor.	

Tabla 23. Tarea de las historias de usuarios #1 y #2.

Tarea de ingeniería	
No. de la Tarea: 2	**No. de la HU:** 3
Nombre de la Tarea: Implementar el callback de configuración.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**:1
Fecha de inicio:11/3/2013	**Fecha fin:**15/3/2013
Programador responsable: Adrian Díaz Díaz	

Descripción: Está dirigida a satisfacer la petición de los parámetros de configuración que realice el cliente de visualización web.

Tabla 24. Tarea de la historia de usuario #3.

Iteración 2.

La segunda iteración tiene como objetivo desarrollar las HU 4, 5 las cuales son de gran importancia ya que se dedican al envío de puntos y a funciones rectoras del sistema.

Historia de Usuario	**Tiempo de implementación (semanas)**	
	Estimación	**Real**
Enviar los puntos de un despliegue.	2	2
Autenticar usuario.	1	1

Tabla 25. HU abordadas para la segunda iteración.

Tareas de las HU desarrolladas para la segunda iteración.

Después de relacionadas las HU con las iteraciones correspondientes se procede a la especificación de las principales tareas que se realizaron para el cumplimiento de la iteración. (Ver Tabla 27,28 y 29)

Tarea de ingeniería	
No. de la Tarea: 3	**No. de la HU:**4
Nombre de la Tarea: Implementar mecanismo de adquisición de variables.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**:1
Fecha de inicio:18/3/2013	**Fecha fin:**22/3/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Está dirigida a la adquisición de variables de la capa de comunicaciones del SAINUX Almiquí.	

Tabla 26. Tarea #1 de la historia de usuario #4.

Tarea de ingeniería	
No. de la Tarea: 4	**No. de la HU:** 4
Nombre de la Tarea: Implementar el callback de puntos.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**:1
Fecha de inicio:25/3/2013	**Fecha fin:**29/3/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Está dirigida a satisfacer la petición de los puntos de un despliegue, que realiza el cliente de visualización web.	

Tabla 27. Tarea #2 de la historia de usuario #4.

Tarea de ingeniería	
No. de la Tarea: 5	**No. de la HU:**5
Nombre de la Tarea: Implementar el callback de seguridad.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**:1
Fecha de inicio:1/4/2013	**Fecha fin:**5/4/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Está dirigida a satisfacer la petición de autenticación que realice el cliente de visualización web.	

Tabla 28. Tarea de la historia de usuario #5.

Iteración 3.

La tercera iteración tiene como objetivo desarrollar las HU 6, 7.

Historia de Usuario	**Tiempo de implementación (semanas)**	
	Estimación	**Real**
Enviar sumario de puntos.	2	2
Enviar detalles de un punto.	1	1

Tabla 29. HU abordadas para la tercera iteración.

Tareas de las HU desarrolladas para la tercera iteración.

Al relacionar las HU con las iteraciones correspondientes se procede a la especificación de las principales tareas que se realizaron para el cumplimiento de la iteración. (Ver Tabla 31 y 32)

Tarea de ingeniería	
No. de la Tarea: 6	**No. de la HU:**6
Nombre de la Tarea: Implementar el callback sumario de puntos.	
Tipo de Tarea: Desarrollo.	**Puntos estimados**:2
Fecha de inicio:8/4/2013	**Fecha fin:**19/4/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Está dirigida a satisfacer la petición del sumario de puntos que realice el cliente de visualización web.	

Tabla 30. Tarea de la historia de usuario #6.

Tarea de ingeniería	
No. de la Tarea: 7	**No. de la HU:**7
Nombre de la Tarea: Implementar el callback detalles de un punto.	

Tipo de Tarea: Desarrollo.	**Puntos estimados**:1
Fecha de inicio:22/4/2013	**Fecha fin:**26/4/2013
Programador responsable: Adrian Díaz Díaz	
Descripción: Está dirigida a satisfacer la petición de mostrar los detalles de un punto en específico, que realice el cliente de visualización web.	

Tabla 31. Tarea de la historia de usuario #7.

3.4. Pruebas de Aceptación.

La puesta en práctica de estas pruebas permitió que se comprobaran las funcionalidades definidas por el cliente una vez analizados los resultados, estos se muestran a continuación:

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU3_P1	**Historia de Usuario:** 3
Nombre: Enviar la configuración del proyecto.	
Descripción de la Prueba: Comprueba que el servidor XStormAlmiquiServer devuelve una cadena con toda la información que contiene el archivo .XML	
Condiciones de Ejecución: Conexión establecida con el cliente WebSocket a través del callback_security	
Entrada / Pasos de ejecución: **-**El cliente XStormClient envía la cadena **"config"**. -El servidor XStormAlmiquiServer ejecuta la función callback_security. -El servidor XStormAlmiquiServer lee la información de un archivo .xml. -El servidor XStormAlmiquiServer envía una cadena con la información del .xml.	
Resultado Esperado: El servidor XStormAlmiquiServer lee la información del archivo xml y la envía correctamente.	
Evaluación de la Prueba: Satisfactorio.	

Tabla 32. CP Enviar la configuración del proyecto.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU4_P2	**Historia de Usuario:** 4
Nombre: Enviar los puntos de un despliegue.	
Descripción de la Prueba: Comprueba que el servidor XStormAlmiquiServer devuelve una cadena en formato JSON, con la información referente a los puntos un despliegue que se desea visualizar.	
Condiciones de Ejecución: Conexión establecida con el cliente WebSocket a través del callback_runtime.	
Entrada / Pasos de ejecución:	

El cliente XStormClient envía una cadena que contiene el id del despliegue a visualizar ejemplo: **"1"**. -El servidor XStormAlmiquiServer ejecuta la función callback_runtime. -El servidor XStormAlmiquiServer adquiere los valores de los puntos que contiene ese despliegue. -El servidor XStormAlmiquiServer construye una cadena en formato JSON, con la información adquirida. -El servidor XStormAlmiquiServer envía la cadena al cliente XStormClient. -El servidor XStormAlmiquiServer regresa al primer paso para seguir enviando la actualización de los valores de los puntos enviados.
Resultado Esperado: El servidor XStormAlmiquiServer envía una cadena en formato JSON con la información de los puntos de un despliegue.
Evaluación de la Prueba: Satisfactorio.

Tabla 33. CP Enviar los puntos de un despliegue.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU5_P3	**Historia de Usuario:** 5
Nombre: Autenticar usuario.	
Descripción de la Prueba: Comprueba si el sistema autentica un usuario o envía una respuesta al cliente XStormClient en caso de existir un error de autenticación.	
Condiciones de Ejecución: Conexión establecida con el cliente WebSocket a través del callback_security.	
Entrada / Pasos de ejecución: El cliente XStormClient envía un cadena que contiene el nombre y la contraseña del usuario que se desea autenticar: **"u: operador; p: 123456"**. -El servidor XStormAlmiquiServer ejecuta la función callback_security. -El servidor XStormAlmiquiServer intenta autenticar el usuario, en caso de estar correcto su usuario y contraseña lo autentica, en caso contrario envía un mensaje.	
Resultado Esperado: El sistema envía una cadena en formato JSON con la respuesta del usuario autenticado.	
Evaluación de la Prueba: Satisfactorio.	

Tabla 34. CP Autenticar usuario.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU6_P4	**Historia de Usuario:** 6
Nombre: Enviar sumario de puntos.	
Descripción de la Prueba: Comprueba que el servidor XStormAlmiquiServer devuelve una cadena en formato JSON, con la información del sumario de puntos.	
Condiciones de Ejecución: Conexión establecida con el cliente WebSocket a través del protocolo callback_runtime.	

Entrada / Pasos de ejecución: El cliente XStormClient envía la cadena: **"SummaryPoints"**. -El servidor XStormAlmiquiServer ejecuta la función callback_runtime. -El servidor XStormAlmiquiServer adquiere los valores de todos los puntos. -El servidor XStormAlmiquiServer construye una cadena en formato JSON, con la información de los puntos. -El servidor XStormAlmiquiServer envía la cadena al cliente XStormClient y vuelve al primer paso para garantizar la actualización de los puntos.
Resultado Esperado: El servidor XStormAlmiquiServer envía una cadena en formato JSON con la información de los puntos.
Evaluación de la Prueba: Satisfactorio.

Tabla 35. CP Enviar sumario de puntos.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU7_P5	**Historia de Usuario:** 7
Nombre: Enviar detalles de un punto.	
Descripción de la Prueba: 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 SAINUX Almiquí.	
Condiciones de Ejecución: Conexión establecida con el cliente WebSocket a través del protocolo runtime_protocol.	
Entrada / Pasos de ejecución: El cliente XStormClient envía la cadena que contiene el identificador del punto: **"Id: Matrikon.OPC.Simulation.1-Grupo-Random.UInt1;"**. -El servidor XStormAlmiquiServer ejecuta la función callback_runtime. -El servidor XStormAlmiquiServer adquiere los detalles del punto que es especificado. -El servidor XStormAlmiquiServer construye una cadena en formato JSON, con los detalles del punto. -El servidor XStormAlmiquiServer envía la cadena al cliente XStormClient y vuelve al primer paso para garantizar la actualización de los detalles del punto.	
Resultado Esperado: El servidor XStormAlmiquiServer envía una cadena en formato JSON con los detalles del punto que se necesita visualizar.	
Evaluación de la Prueba: Satisfactorio.	

Tabla 36. CP Enviar detalles de un punto.

Una vez realizadas las pruebas al sistema se puede concluir que el sistema cumple con las funcionalidades definidas por el cliente.

3.5. Pruebas de Rendimiento.

Las pruebas de rendimiento se realizan con el objetivo de determinar si los tipos de respuesta del sistema, tanto en estados normales, como especiales se encuentran dentro de los limites predefinidos, por lo que se hace necesario la correcta realización de los diseños de casos de pruebas, debido a que la existencia de errores en estos diseños provocarían la toma de conclusiones erróneas.

Los casos de pruebas identificados son:

- CP1_Enviar los puntos de un despliegue.
- CP2_Enviar sumario de puntos.
- CP3_Enviar detalles de un punto.

Estos casos de pruebas tienen como objetivo comprobar el rendimiento de las funcionalidades que se corresponden con su nombre, así como la latencia del envío de los puntos desde el servidor hacia el cliente. Se debe tener en cuenta que el sistema lee una configuración que se encuentra en un archivo *.xml el cual pose la variable TimeToSendHTTP que es el tiempo entre los envíos de los datos a la web, por lo que si este valor se cambia, varía el resultado de las pruebas.

Se disponen de los siguientes recursos:

- **Físicos:** Laptop con un microprocesador Intel(R) Core(TM) 2 Duo CPU T5250 a 1.50 GHz, con 2GB de memoria RAM.
- **Lógicos:** Sistema Operativo Windows 7.

3.5.1. Análisis de los resultados.

La ilustración 11 muestra los resultados de las pruebas de rendimiento realizadas. En el eje vertical los milisegundos (ms) y en el eje horizontal la cantidad de usuarios (u).

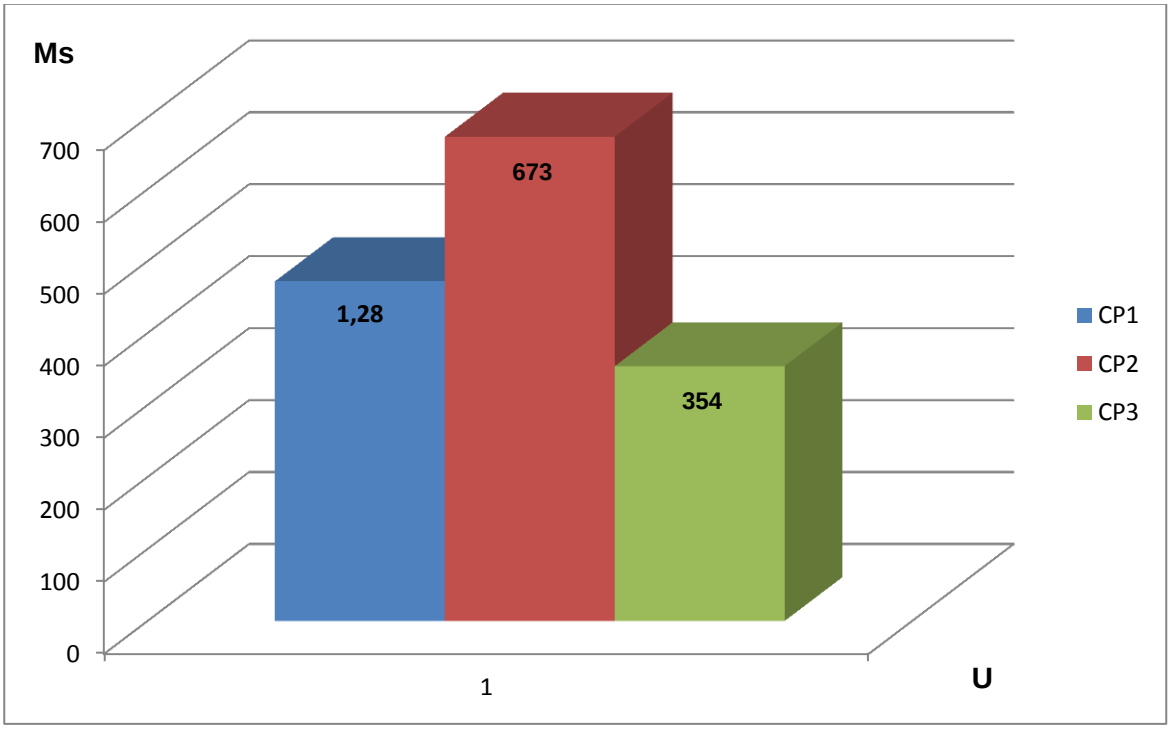

Ilustración 11. Pruebas de rendimiento.

Además se comprobó la memoria del ordenador y los valores oscilan entre 2MB y 3MB por lo que se considera que el sistema consume una cantidad de memoria aceptable.

3.6. Valoración y discusión de los resultados.

Con el resultado de este trabajo se obtuvo un servidor (XStormAlmiquiServer) que a partir de un archivo .XML generado por el SAINUX Almiquí, que contiene la información de los diferentes proveedores, puntos asociados y los despliegues, es capaz de comunicarse con los servidores OPC y enviar los puntos y despliegues a los distintos clientes (XStormClient) que se lo solicitan mediante la comunicación WebSocket. Lográndose que las modificaciones en la configuración del proyecto, sean notificadas de manera inmediata a cada uno de los clientes, elevando el nivel de integridad entre soluciones así como el nivel de accesibilidad a la información. Esto permite que no solamente se acote la solución a clientes OPC sino que pueda visualizarse en dispositivos que manejen o comprendan un navegador web. La ilustración 12 muestra el funcionamiento de la solución.

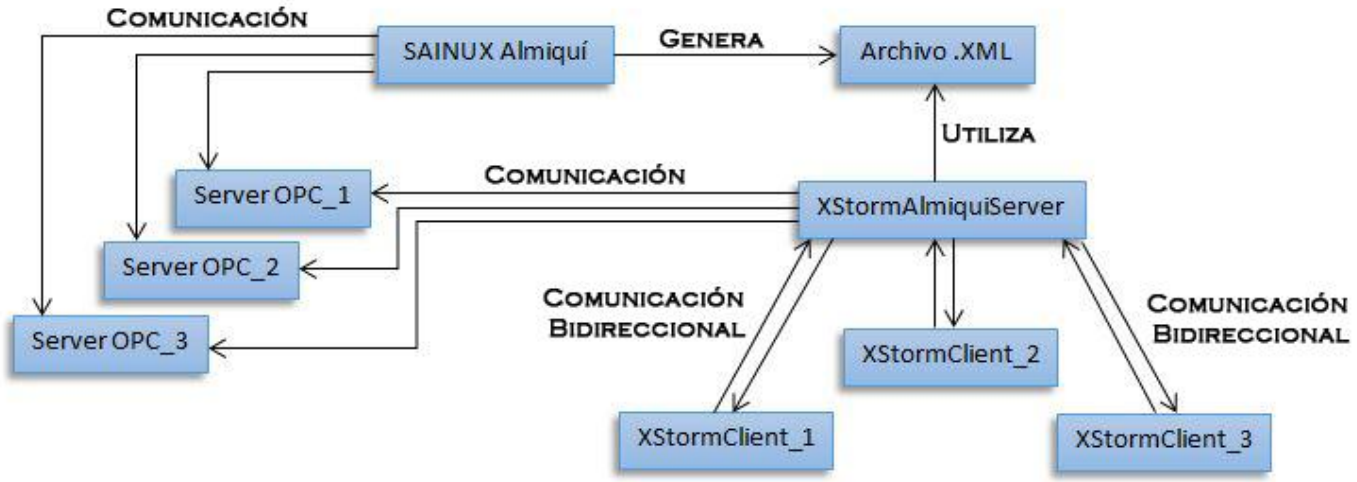

Ilustración 12. Valoración de los resultados.

3.7. Conclusiones Parciales.

Con la realización de este capítulo se desarrolló cada una de las iteraciones de la fase de implementación con sus correspondientes tareas de ingeniería. Además se diseñaron y ejecutaron las pruebas de aceptación y rendimiento que ayudaron a validar que el sistema cumple con las funcionalidades que el cliente definió, concluyendo que la aplicación cumple con las metas trazadas al inicio de la investigación.

CONCLUSIONES

Con la realización del presente trabajo se arribaron a las siguientes conclusiones:

- Se desarrolló una aplicación que elevó los niveles de accesibilidad e integración en el proceso de comunicación entre el sistema SAINUX Almiquí y el visualizador web XStormClient.
- Se logró una integración del cliente de visualización web con el estándar OPC.
- Se realizaron pruebas de aceptación y rendimiento que ayudaron a validar que el sistema cumple con las funcionalidades que el cliente definió.

RECOMENDACIONES

- Implementar mecanismos en el SAINUX Almiquí para las alarmas y la seguridad.
- Incorporar los mecanismos de Request Response para una mayor robustez del sistema a la hora del envío de puntos a la web.
- Robustecer la comunicación en tiempo real entre el servidor y los clientes web.

REFERENCIA BIBLIOGRAFÍA

1. **DEWIRE.D.** Cliente/server Computing. *McGraw-Hill.* Nueva York : s.n., 1993.
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. **Mandado, Enrique.** Autómatas programables y sistemas de automatización. Barcelona : s.n., 2009.
4. **José Miguel Molina Martínez, Manuel Jiménez Buendía.** Programación Gráfica para Ingenieros. Jun 1, 2010.
5. **Foundation, Opc.** OPC Data Access Custom Interface Specification. 2002.
6. **Arno Puder, Kay Romer, Frank Pilhofer.** Distributed Systems Architecture. A Middleware Approach.
7. **Groussard, Thierry.** Recursos Informáticos Java Enterprise Edition - Desarrollo de aplicaciones web con JEE 6. s.l. : Ediciones ENI, 2010.
8. **F. Maciá Pérez, D Marcos Jorquera.** Administración de servicios de Internet: De la teoría a la práctica. Universidad de Alicante : s.n., 2008.
9. www.slideshare.net/josegregoriob/servidor-web-8451426. [En línea] [Citado el: 20 de 2 de 2013.]
10. http://www.rmunoz.net/introduccion-a-node-js.html. [En línea] [Citado el: 12 de 2 de 2013.]
11. **W3C.** http://www.w3.org/TR/eventsource/. [En línea]
12. **Peter Saint-Andre, Kevin Smith, y Remko Tronçon.** XMPP: The Definitive Guide. Abril 2009.
13. **Pfeiffer., Silvia.** The Definitive Guide to HTML5 Video. . Enero de 2011.
14. **Cámbar, Julio César Moreno.** XStormServer, servidor para el cliente de visualización web del SCADA Guardián del ALBA. La Habana : s.n., Junio 2012.
15. **Brian W. Kernighan, Dennis M. Ritchie.** El lenguaje de programación C. s.l. : Pearson Educación, 01/04/1991.
16. **Mora, Sergio Luján.** C++ Paso a Paso. 1998.
17. **Beck, K.** "Extreme Programming Explained. Embrace Change". Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio. s.l. : Pearson Education, 1999.
18. **Jeffries, R., Anderson, A., Hendricks ,C., Addison-Wesley.** "Extreme Programming Installed". 2001.
19. **Joskowicz, Ing. José.** Reglas y Prácticas eneXtreme Programming. . 2008.
20. **Informática, Unidad Docente de Ingeniería del Software- Facultad de.** Patrones del "Gang of Four". Universidad Politécnica de Madrid : s.n.
21. **Lorenzo, Ariel Chávez.** Estándares de codificación para C++. . 2011.
22. **Penadés, Patricio Letelier y Ma Carmen.** Métodologías ágiles para el desarrollo de software. *Métodologías ágiles para el desarrollo de software.* [En línea] [Citado el: 2013 de enero de 23.] Métodologías ágiles para el desarrollo de software..

GLOSARIO

PLC (Controlador lógico programable): dispositivo electrónico de control de procesos y se basa en una lógica, definida a través de un programa de computación.

OPC (Object linking and envidig for Process Control): estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft.

COM (Component Object Model): filosofía de programación de objetos orientados a Interfaces para compartir entre aplicaciones.

DCOM (Distributed Component Object Model): mecanismo que logra la evolución de la Tecnología COM, para ser portada en redes de computadoras.

HTTP (Hypertext Transfer Protocol): protocolo usado en cada transacción de la World Wide Web.

HTTP5 (HyperText Markup Language, versión 5): es la quinta revisión importante del lenguaje básico de la World Wide Web.

HTML (Lenguaje de Marcado de Hipertexto): lenguaje utilizado para crear páginas web.

CPU (Unidad central de procesamiento): controla y procesa todas las operaciones realizadas dentro del PLC.

TCP (Transmission Control Protocol): es uno de los protocolos fundamentales en Internet.

DNS (DomainName System): es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

API (Interfaz de programación de aplicaciones): es el conjunto de funciones y procedimientos 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.

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

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.

Autenticar: verificación de la identidad de una persona o de un proceso para acceder a un recurso o poder realizar determinada actividad.

Punto: 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.

XML (Extensible Markup Language): es un lenguaje de marcado sencillo. Su objetivo es facilitar la representación, almacenamiento y trasmisión de información por parte de aplicaciones informáticas, computadoras y medios de comunicación digital en general.