UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS
INGENIERÍA EN COMPUTACIÓN

“DESARROLLO DE UNA INTERFAZ TCP/IP - HART PARA LA INTEGRACIÓN DE UN MANEJADOR DE ACTIVOS DE INSTRUMENTACIÓN EN LÍNEA, UTILIZANDO HERRAMIENTAS WEB, BAJO SOFTWARE LIBRE”

PRESENTADO POR:

ARTURO JOSÉ VILA GONZÁLEZ
C.I.: V- 17.900.489

Trabajo de grado presentado como requisito parcial para optar al título de
INGENIERO EN COMPUTACIÓN

BARCELONA, OCTUBRE DE 2009.

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS
INGENIERÍA EN COMPUTACIÓN

“DESARROLLO DE UNA INTERFAZ TCP/IP - HART PARA LA INTEGRACIÓN DE UN MANEJADOR DE ACTIVOS DE INSTRUMENTACIÓN EN LÍNEA, UTILIZANDO HERRAMIENTAS WEB, BAJO SOFTWARE LIBRE”

ASESORES:

ING. CLAUDIO CORTÍNEZ	ING. LUIS RODRIGUEZ
ASESOR ACADEMICO	ASESOR INDUSTRIAL

BARCELONA, OCTUBRE DE 2009.

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS
INGENIERÍA EN COMPUTACIÓN

“DESARROLLO DE UNA INTERFAZ TCP/IP - HART PARA LA INTEGRACIÓN DE UN MANEJADOR DE ACTIVOS DE INSTRUMENTACIÓN EN LÍNEA, UTILIZANDO HERRAMIENTAS WEB, BAJO SOFTWARE LIBRE”

JURADO CALIFICADOR:

ING. CLAUDIO CORTÍNEZ
ASESOR ACADEMICO

ING. MARÍA GERARDINO
JURADO PRINCIPAL

ING. PEDRO DORTA
JURADO PRINCIPAL

BARCELONA, OCTUBRE DE 2009.

RESOLUCIÓN

De acuerdo con el artículo 44 del reglamento de Trabajo de Grado:

"Los Trabajos de Grado son de exclusiva propiedad de la Universidad de Oriente y solo podrán ser utilizados a otros fines con el consentimiento del Consejo de Núcleo respectivo, quien lo participará al Consejo Universitario."

DEDICATORIA

Quiero dedicar este trabajo a mi madre. Gracias por todo el amor, la dedicación y apoyo incondicional. Todo mi esfuerzo fue inspirado en ti. Este logro también te pertenece. Te Amo!

AGRADECIMIENTOS

Ante todo agradezco a Dios, por todas las bendiciones que me ha dado y la protección que me brinda.

A mi mamá por todo lo que soy hoy día, por todos sus cuidados y por su amor infinito e incondicional.

A todos mis amigos que me han ayudado en todo momento.

A Marianne, Donato, David, Carlos, Antonella, Elimar y Laura el combo del PIO XII que durante las fases del colegio y la universidad siempre han estado allí. ¡Los quiero mucho y Gracias por Todo!

También quiero a gradecer a todos los buenos amigos que hice durante estos años de vida en la universidad de oriente, especialmente a Fernando y Yubraska que estuvieron conmigo durante todo este camino. Junto a ustedes compartí los más rigurosos y gratos momentos. Gracias por hacer de este viaje uno de los más especiales. ¡Gracias!

Al Prof. Claudio por siempre tenderme la mano y ofrecerme su ayuda. Por todos los conocimientos que me transmitió y su esmero y dedicación para culminar este trabajo.

Al Sr. Luis Rodríguez, Gracias por la oportunidad que me brindó, por su valioso apoyo y siempre respondiendo más allá a mis preguntas.

A todas las personas con las que compartí durante mi pasantía en la empresa, les agradezco por su enorme amabilidad y simpatía, por los conocimientos que me transmitieron tanto académicos como a nivel personal.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

En la presente investigación se desarrolló un sistema que permite monitorear, calibrar y diagnosticar de manera remota la instrumentación de campo instalada en la Refinería de Puerto La Cruz. Esto se logró con la utilización del conjunto de protocolos TCP/IP para extender el alcance al sistema a través de la intranet de la empresa, así también, se utilizó el protocolo de comunicación industrial HART que permite por medio de su conjunto de comandos realizar operaciones sombre los dispositivos de campo. Este sistema reducirá la necesidad del instrumentista que labora en la empresa salir al campo en casos de diagnóstico, calibración y recolección de data, disminuirá de forma notoria el tiempo empleado para la corrección de fallas en los instrumentos, permitiendo una planificación óptima de mantenimiento o reemplazo de un instrumento sin afectar el funcionamiento normal de la planta. El diseño y construcción de la aplicación, se llevó a cabo siguiendo los lineamientos del Proceso Unificado de Desarrollo de Software, cuya metodología implementa el Lenguaje Unificado de Modelado (UML).

CAPITULO 1
PLANTEAMIENTO DEL PROBLEMA

1.1.- Descripción de la empresa [7]

La refinería Puerto La Cruz es uno de los centros de procesamientos de crudo más importantes de PDVSA e integra un circuito de manufactura del petróleo extraído en los campos de los estados Monagas y Anzoátegui.

Geográficamente, esta planta abarca tres áreas operacionales: Puerto La Cruz, El Chaure y San Roque, ubicadas en el norte y centro del estado Anzoátegui, con una capacidad total de procesamiento de crudos de 200 mil barriles por día, de los cuales se obtienen 73 mil barriles de gasolina y nafta, 12 mil barriles de kerosene-jet, 43 mil barriles de gasoil y 73 mil barriles de residual, insumos y requeridos para la mezcla de combustibles comercializados en los mercados interno y de exportación.

El manejo de estos enormes volúmenes de producción requiere de 129 tanques de almacenamiento con capacidad para 13,5 millones de barriles de crudo y productos, que son despachados a otras partes del país y al extranjero por la Terminal Marino de Guaraguao, el cual admite en sus siete muelles un promedio de 55 buques mensuales, que pueden transportar 20,2 millones de barriles mensuales.

Para la distribución de combustibles al circuito de estaciones de servicio de los estados de Nueva Esparta, Sucre, Monagas, Delta Amacuro, Bolívar, Guárico y Anzoátegui, la refinería porteña cuenta con el Sistema de Suministro de Oriente (SISOR).

La Refinería Puerto La Cruz inicia su construcción en el año 1948, por medio de la empresa Vengref, cuando aún la escasa población de la época convivía en una aldea de pescadores. En 1950, Se inicia el funcionamiento de la planta con la unidad de destilación atmosférica número uno (DA-1) para procesar 44 mil barriles diarios (MBD).

Para los años 1955-1962, son instaladas en la refinería nuevas unidades orientadas a aumentar la capacidad de procesamiento de crudo pesado y así mismo aumentar el volumen de sus derivados. En noviembre de 2004 se finaliza el proyecto Valcor, y pasa a manos de PDVSA como responsable, este proyecto incorpora a la refinería mejoramiento de sus productos.

La Refinería Puerto La Cruz cuenta con una capacidad nominal de 200 MBD de crudo en sus tres unidades de destilación, de los cuales 45% (90 MBD) corresponde a crudo pesado. Como insumos de procesos se tiene: Isobutano, Gasóleo de vacío y Residuo Desparafinado (SRQ) y los insumos de mezcla: Gas Natural, Gasolina Natural, Naftas, Alquilato, Gasolinas de motor y Destilados.

1.1.1.- Organización de la empresa [7]

PDVSA cumple con todas las actividades propias del negocio petrolero, constituyéndose en una corporación verticalmente integrada, que abarca todos los procesos, desde la explotación hasta la comercialización de los hidrocarburos gaseosos y no gaseosos, y sus derivados. Los procesos que se realizan en Petróleos de Venezuela S. A., son los siguientes:

- ***Exploración y Producción:*** es el primer eslabón de la cadena. De esta fase depende el hallazgo de hidrocarburos (gaseosos y no gaseosos) en el subsuelo.
- ***Refinación:*** proceso que se encarga de la transformación de los hidrocarburos

en productos derivados.

- ***Comercialización:*** último eslabón de la cadena productiva. En esta etapa se establecen las fórmulas de precios que reflejan las variaciones del mercado para garantizar precios e ingresos justos para el pueblo venezolano.
- ***Gas:*** Venezuela es una de las potencias mundiales del sector de hidrocarburos gaseosos.

1.1.1.1.- Estructura organizativa

La estructura organizativa que rige la empresa se puede visualizar en la figura 1.1. Geográficamente PDVSA se encuentra organizada en tres divisiones: Occidente, Centro-Sur y Oriente. La División Occidente se encarga de todas las operaciones de la empresa en el occidente del país; tiene su sede en Maracaibo, la División Centro-Sur tiene su sede en Barinas; por su parte la División Oriente tiene su sede en Puerto La Cruz, en sus instalaciones albergan entre otras a las Gerencias de Apoyo y Gestión. [7]

1.1.2.- Funciones

Se encarga de regir, proveer y mantener los servicios y soluciones integrales de tecnologías de automatización, información y comunicaciones del área de Refinación Oriente y contribuye a mantener la continuidad operativa y a ejecutar los planes; innovar y actuar como agente de transformación en PDVSA y en la sociedad venezolana con corresponsabilidad social, económica y ambiental; potenciando un ecosistema tecnológico que impulse los poderes creadores del pueblo, el conocimiento libre, el desarrollo endógeno sustentable y la economía social productiva para lograr la soberanía tecnológica. Todo ello alineado con la Constitución de la República Bolivariana de Venezuela y en coordinación con sus organismos rectores. En la figura 1.2, se muestra el organigrama de la Gerencia de

AIT, la cual está integrada por Servicios Comunes, Adiestramiento AIT y AIT Refinación, ésta última está integrada por seis (6) superintendencias, las cuales son Gestión de Nuevas Oportunidades (GNO), Gestión del Servicio (GS), Mantenimiento a la Plataforma (MAP), Control de Plataforma (CP), Planificación de tecnología de información y comunicaciones (P/TIC) y Desarrollo, Implantación y Soluciones (DIS), la cual está compuesta a su vez por tres (3) secciones respectivamente, las cuales son: Proyectos de Automatización, Proyectos de Informática y Proyectos de Telecomunicaciones y por una sección de gestión, Planificación y Control de Proyectos. [7]

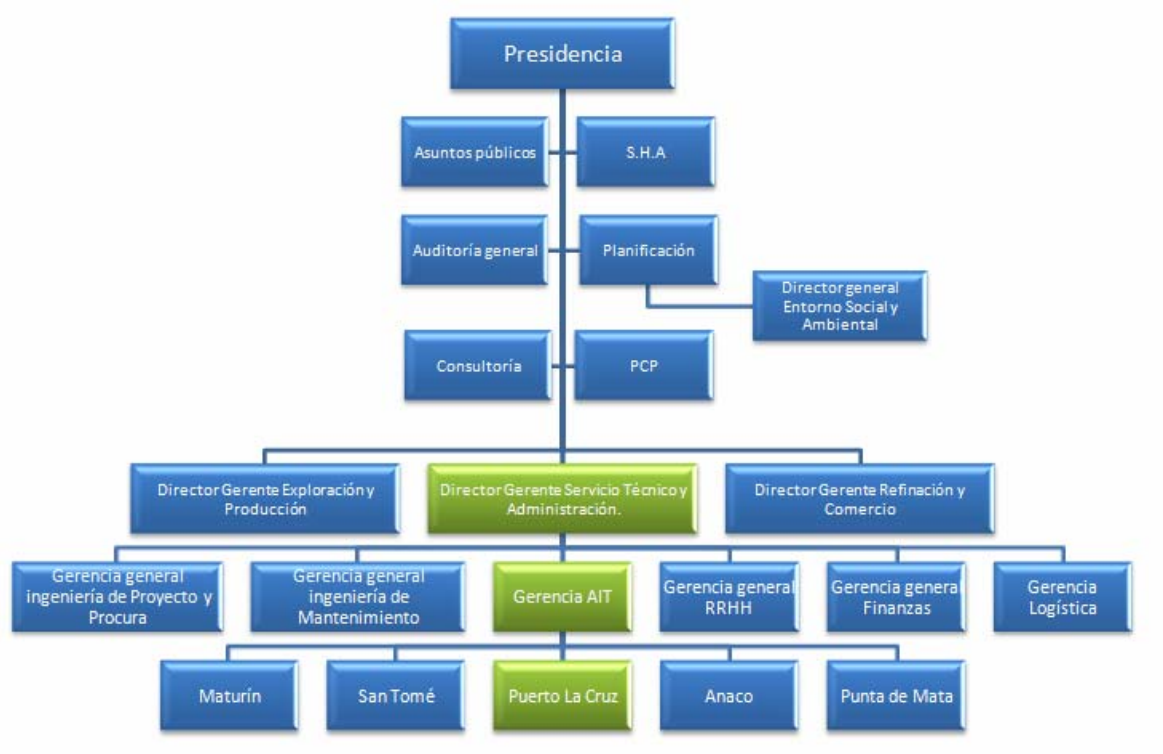

Figura 1.1. Estructura organizativa de PDVSA

Fuente: Descripción general refinería Puerto La Cruz (2005)

1.2.- Planteamiento del problema

En la actualidad, la digitalización de la información influye en el mundo, por tanto los

procedimientos y medios de medición no se ven excluidos de la misma. Con el tiempo se han implementado en el campo industrial otros métodos para poder realizar las mediciones de modo remoto, sin la necesidad de ir al campo y que a su vez, permita la transmisión de una información detallada del equipo y sus funciones.

PDVSA Refinación Oriente, cuenta con una gran cantidad de equipos de campo que permiten que las plantas se encuentren operativas, por tal motivo es fundamental la lectura permanente de las diversas variables de sus tanques, tuberías o máquinas para evitar pérdidas del producto, accidentes o simplemente una falta de eficiencia en la producción y así, poder tomar las medidas de mantenimiento que correspondan de manera oportuna, ya sea mantenimiento preventivo o correctivo. Por esta razón se han instalado diversos dispositivos de medición (instrumentos) a lo largo de la refinería.

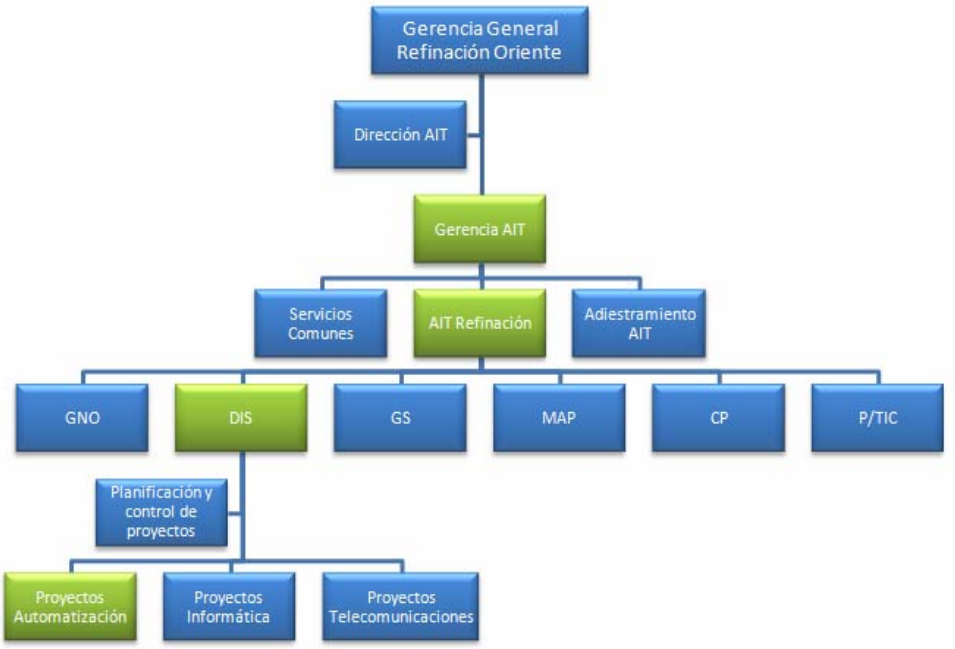

Figura 1.2. Organigrama de AIT

Fuente: Descripción general refinería Puerto La Cruz (2005)

En los últimos años en la Refinería Puerto La Cruz se ha venido realizando una modernización en la instrumentación de cada una de las plantas que la componen (Destiladora Atmosférica1, Destiladora Atmosférica2, Destiladora Atmosférica3, Fraccionadora por Craqueo Catalítico, Alquilación, entre otras), sustituyendo los instrumentos analógicos por transmisores y sensores inteligentes. Los instrumentos analógicos utilizan para la comunicación el lazo de 4-20 mA, un lazo de corriente que dependiendo del valor medido por el sensor, se le asigna un nivel de corriente en mA, es decir, 4 mA equivaldría al menor valor posible de la variable medida y 20 mA al máximo.

Los transmisores y sensores inteligentes operan utilizando comunicación digital bajo el protocolo industrial HART, por sus siglas en ingles (*Highway Addressable Remote Transducer Protocol*), este protocolo es llamado híbrido por conservar la señal analógica de 4-20 mA, insertando en esta, por medio de modulación de frecuencia, datos digitales, sin la necesidad de modificar el conexionado analógico.

El protocolo HART funciona a través del sistema Maestro -Esclavo. Como su nombre lo dice este sistema consta de dos tipos de dispositivos, uno al que llamamos Maestro, el cual está encargado de iniciar las comunicaciones y es el que pide la información. Los dispositivos Esclavos, en tanto, solo envían información cuando se les solicita.

La tarea de monitoreo y control de los procesos en la refinería está bajo la responsabilidad de la Gerencia de Procesos, la emplean los operadores de consola a través del DCS por sus siglas en ingles (*Distributed Control System*), estos tienen acceso automatizado a todo el sistema de la refinería para realizar cualquier maniobra en las plantas, sin embargo, las entradas del DCS son totalmente analógicas, por lo que es desaprovechada casi en su totalidad la información digital que ofrece el

instrumento inteligente utilizando el protocolo HART. Por otra parte, los instrumentistas que laboran bajo la competencia de la Gerencia de Mantenimiento que tienen como funciones el mantenimiento de sistemas automáticos de control, no cuentan con un sistema de monitoreo remoto de los instrumentos, esta labor requiere de cuadrillas que se dirijan directamente a los instrumentos para realizar las supervisiones, diagnósticos y calibraciones.

Para el aprovechamiento de las ventajas que ofrece la instrumentación digital la empresa ha iniciado con una primera fase de este proyecto, en la que se diseñó un manejador de activos, el cual permite la supervisión, diagnóstico y calibración de la instrumentación HART a través de la interfaz RS-232, este último, es un estándar para la conexión serial de señales de datos binarios con conector tipo DB-25 (25 pines), aunque también es común el DB-9 (de 9 pines). Ahora, se plantea rediseñar la aplicación para establecer la comunicación utilizando los protocolos TCP/IP, con esto se consigue extender su alcance a través de toda la intranet PDVSA, ya que la interfaz RS-232 está limitada para distancias cortas de hasta 15 metros según la norma y, para velocidades de comunicación bajas.

Como respuesta a las necesidades de la Gerencia de Mantenimiento, se plantea diseñar una interfaz TCP/IP - HART para la integración de un manejador de activos de instrumentación bajo software libre, que permita acceder vía web a los instrumentos que están ubicados a lo largo de las instalaciones de la refinería. La aplicación web debe ser capaz de enlazarse con el manejador de activos para ejecutar operaciones de configuración, monitoreo y diagnóstico, construyendo la trama en protocolo HART según la operación que se desea ejecutar, encapsulándola como mensaje en un paquete de Protocolo TCP/IP para ser enviada a través de la red. Este proyecto es importante para la Gerencia de Mantenimiento en los siguientes aspectos:

- Reducción de la necesidad de salir al campo en casos de diagnóstico, calibración y recolección de data.
- Interacción con los dispositivos a través de la web en tiempo real.
- Disminución notoria en cuanto al tiempo empleado para la corrección de fallas en los instrumentos.
- Planificación óptima de mantenimiento o reemplazo del instrumento sin afectar el funcionamiento normal de la planta.

Se ha considerado implantar este proyecto abarcando toda la instrumentación que presta servicio en las plantas que componen la Refinería Puerto La Cruz. El proceso de implantación se estima aplicar en un pequeño sector de la refinaría como plan piloto y, paulatinamente, incorporar los sectores restantes.

Es importante destacar que la empresa no cuenta con un sistema de acceso remoto que brinde operaciones de monitoreo, calibración y diagnóstico a la instrumentación instalada en sus plantas. Un sistema similar, lo ofrece Emerson Process Management, una compañía de Emerson, que en su división Asset Management Solutions (AMS), ofrecen soluciones avanzadas de control de procesos y administración de plantas. Este sistema de Emerson tiene la desventaja que es un sistema cerrado, con licencia comercial de costo elevado.

Para el desarrollo de este proyecto, se utilizará la metodología de Proceso Unificado Racional (RUP) conjuntamente con el Lenguaje Unificado de Modelado (UML). Todo el sistema estará basado en herramientas computacionales de software libre, el cual tiene como ventajas principales:

- Código fuente abierto, permitiendo de esta manera modificarlo y adecuarlo a las necesidades del usuario.
- Licencia Pública General (GPL), al ser software libre permite un ahorro

económico al no necesitar el pago de licencias para su uso.

- Compatibilidad, cualquier instrumento dotado con la capacidad de comunicación HART podrá ser usado sin importar el fabricante del mismo.

1.3.- Objetivos

Objetivo General

Desarrollar una interfaz TCP/IP - HART para la integración de un manejador de activos de instrumentación en línea, utilizando herramientas web, bajo software libre.

Objetivos Específicos

- Revisar la documentación sobre el estándar HART (Transductor Remoto Direccionable de Alta velocidad), de la HFC (Fundación de Comunicaciones HART).
- Analizar la estructura y documentación de la aplicación manejador de activos.
- Diseñar una interfaz de usuario para el sistema, enfocadas a las necesidades y metas del usuario.
- Diseñar la arquitectura y la base de datos del sistema.
- Codificar los módulos del sistema.
- Realizar las pruebas de comunicación e integración.

CAPÍTULO 2
FUNDAMENTOS TEÓRICOS

2.1.- Antecedentes

Los antecedentes de este trabajo son proyectos que manejan el conjunto de protocolos TCP/IP y/o HART, así también, algunos de ellos presentan visualización de gráficas de variables y utilización de tecnología Web.

- Hernández, V. (2007), desarrolló el proyecto titulado **"Diseño de un manejador de activos para instrumentación digital, con protocolo HART, basado en herramientas computacionales de software libre"**, para optar al título de Ingeniero Electricista. Este proyecto se realizó en la empresa PDVSA en sus instalaciones de la refinería de Puerto La Cruz, con el propósito de permitir al usuario visualizar la información de los equipos y tener acceso a la documentación de los mismos. Este comunicador está completamente codificado en lenguaje C (GNUC), hace uso de la comunicación serial, empleando como salida el puerto conocido en el entorno Linux como el ttyS0 (COM 1 en ambientes Windows). [1]

- Tenías, J. (2007), desarrolló el proyecto titulado **"Desarrollo de un software basado en aplicaciones web para el monitoreo de dispositivos de la plataforma de telecomunicaciones de PDVSA-GAS"**, para optar al título de Ingeniero en Computación. Este proyecto se realizó en la empresa PDVSA Gas en el distrito de producción Anaco con la finalidad de concentrar las tareas de monitoreo que se ejecutan de manera separada en los departamentos de servidores y redes, en un único sistema denominado Centro de Monitoreo (CMCRTA). [2]

- Núñez, L. (2007), desarrolló el proyecto titulado **"Desarrollo de una aplicación web para la visualización en tiempo real de los parámetros operacionales de producción de la empresa PDVSA"**, este consistía en la visualización en tiempo real de los parámetros operacionales de producción de la empresa PDVSA, tales como temperatura, presión, voltaje, etc. Este sistema estaba constituido por 3 módulos: uno para visualización de despliegues, otro para creación y edición de despliegues y un modulo para la administración de usuarios y otros elementos. El desarrollo de la aplicación se basó en el análisis del proceso de producción, así como las necesidades y la arquitectura disponible en la empresa. [3]

- Ramos, Z. (2000), desarrolló el proyecto titulado **"Desarrollo de un software para la determinación del impacto sobre la plataforma de sistemas, ocasionado por alteraciones en el diseño de la instrumentación de una planta de mejoramiento de crudo"**, para optar al título de Ingeniero en Computación. Este proyecto plantea determinar el impacto que ocasionan los cambios ocurridos en la instrumentación del complejo mejorador, mediante la implementación de un software, y así, poder prever las repercusiones directas que conlleva la optimización en el diseño del proceso de mejoramiento e instrumentación sobre la plataforma de control de procesos, y a su vez, en la plataforma de sistemas de información. [15]

- Querales, O. (1999), desarrolló el proyecto titulado **"Sistema de control distribuido"**, para optar al título de Ingeniero electricista. En este trabajo se evaluó el comportamiento de un Sistema de Control Distribuido típico sobre una red de área local de sistema abierto, utilizando un modelo *Cliente/Servidor* y empleando la interfaz de Programación de Redes *Windows Sockets* con protocolo *TCP/IP* para verificar el funcionamiento adecuado de una planta industrial. [16]

2.2.- Software libre [21]

Software libre (en inglés ***free software***) es la denominación del software que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la *Free Software Foundation*, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- Libertad 0: la libertad de usar el programa, con cualquier propósito.
- Libertad 1: la libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades.
- Libertad 2: la libertad de distribuir copias, con lo que puedes ayudar a tu vecino.
- Libertad 3: la libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie.

La libertad de usar el programa con cualquier propósito, tales como: estudiar su funcionamiento, adaptarlo a las necesidades, distribuir copias, producir mejoras y hacerlas públicas, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo).

El software libre suele estar disponible gratuitamente o al precio de coste de la distribución a través de otros medios, sin embargo, no es obligatorio que sea así, por ende no hay que asociar software libre a "software gratuito" (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente ("software comercial"). Análogamente, el "software gratis" o "gratuito" incluye en algunas ocasiones el código fuente, no obstante, este tipo de

software *no es libre* en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Tampoco debe confundirse software libre con "software de dominio público", este último es aquél que no requiere de licencia, pues sus derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquél cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado, tras un plazo contado desde la muerte de éste, habitualmente 70 años. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es dominio público.

2.3.- Protocolos

Toda transferencia ordenada de información entre puntos que conforman un enlace, se realiza por medio de un protocolo. Este se puede definir como un conjunto de reglas que permiten identificar tanto el emisor como al receptor, supervisar todas las funciones de control implicadas en una transferencia de datos, además de incluir procedimientos para la detección y corrección de errores.

2.4.- Protocolo HART

El Protocolo HART fue desarrollado por Rosemount a finales de los años 80. HART es la sigla de “Highway Addressable Remote Transducer”. El protocolo fue abierto en 1990 para que otras compañías pudieran usarlo, ese mismo año fue creado un Grupo de Usuarios. [5]

En Marzo de 1993, el grupo votó para la creación de una organización independiente y sin fines de lucro, para un mejor soporte del Protocolo HART. En Julio de ese mismo año, la “HART Communication Foundation” (HCF) fue establecida para proveer soporte a escala mundial para la aplicación de esta tecnología. [5]

Diseñado para complementar la tradicional señal análoga de 4-20 mA, el Protocolo HART soporta las comunicaciones digitales por dos cables para medición de procesos y dispositivos de control. Sus aplicaciones incluyen interrogación remota y acceso cíclico a variables de proceso, definición de parámetros y diagnósticos. Además es soportado por los proveedores más grandes de instrumentación y dispone de cobertura para productos de todo el rango de medición de procesos y aplicaciones de control. [5]

2.4.1.- Concepto del Protocolo HART

El método tradicional de transmisión de datos con 4-20 mA, solo se limita a transmitir la magnitud de la medición. Con la evolución en los procesos y la aparición de la instrumentación de campo inteligente, se hizo necesario encontrar nuevas formas de transmisión, en este marco se desarrolla el Protocolo HART, es un protocolo de comunicación diseñado para aplicaciones de medición y control de procesos industriales, es un protocolo híbrido, que mezcla la señal análoga de corriente con la transmisión de datos digitales por los mismos dos cables sin que se distorsionen ninguna de las dos señales. Este tipo de comunicación trae dos grandes ventajas, primero el cableado existente y las estrategias de control actualmente utilizadas, no deberán ser totalmente reemplazados al momento de implementar HART, y segundo toda la información adicional que se puede transmitir (tags, datos de rango y span, información del producto y diagnósticos etc.), la cual puede permitir ahorrar mucho tiempo y dinero a la hora de la mantención, y además mejora el

manejo y la utilización de las redes de instrumentos inteligentes. HART es un protocolo que puede funcionar como Maestro-Esclavo (un dispositivo de campo solo responde cuando se le ha pedido algo previamente). Así como también puede funcionar en modo Ráfaga. Puede haber hasta dos maestros y hasta 15 dispositivos esclavos se pueden conectar en configuración multipunto. Cabe destacar que HART posee una arquitectura abierta, disponible para cualquier proveedor y para cualquier usuario. [5]

2.4.2.- Comunicación bidireccional

Al usar una señal analógica, la información se envía en una sola dirección, ya sea del dispositivo al host (entradas) o del host al dispositivo (salidas). La información digital, por otro lado, puede viajar en ambas direcciones usando la señal de comunicación digital HART. Un transmisor que tradicionalmente sólo envía una variable de proceso al host, ahora también puede recibir información tal como ajustes de configuración. [22]

2.4.3.- Método de Operación

El protocolo HART opera usando el principio de modulación por desplazamiento en frecuencia (FSK), por sus siglas en ingles (Frequency-shift keying), el cual está basado en el estándar de comunicación Bell 202. La señal digital se construye a través de un ciclo 1200 Hz, para representar el bit 1 y aproximadamente dos ciclos de 2200 Hz que representan el bit 0. La tasa de transmisión de datos es de 1200 baudios. Como se observa en la figura 2.1 las señales sinusoidales son sobrepuestas a las señales de corriente, a un bajo nivel, logrando así que las dos señales se transmitan por los mismos dos cables, gracias a que el valor promedio de la señal FSK es siempre cero, la señal de 4-20 mA nunca se verá distorsionada. Esto produce una comunicación simultánea con un tiempo de respuesta aproximado de 500 ms para

cada dispositivo de campo, sin que ninguna de las señales análogas sea interrumpida. [22]

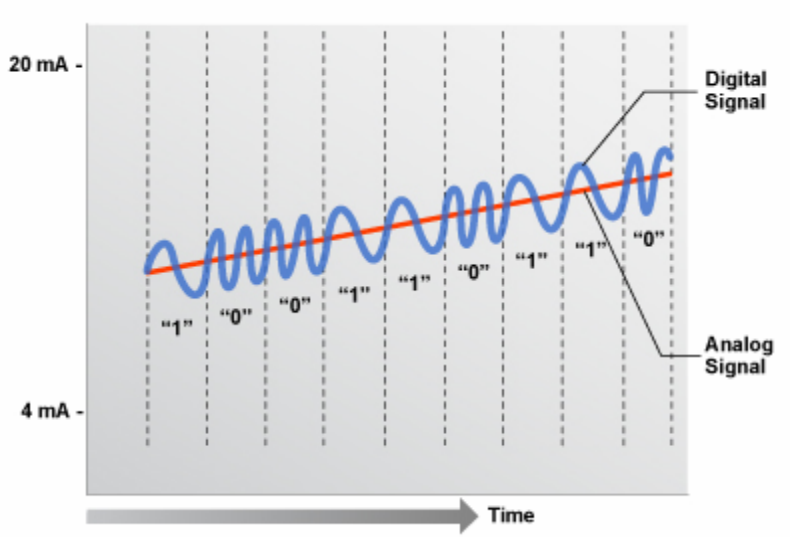

Figura 2.1. Modulación por desplazamiento en frecuencia (FSK)

Fuente: Fundación de comunicación HART

2.4.4.- Esquema Maestro – Esclavo

Como se dijo anteriormente HART funciona a través del sistema Maestro-Esclavo. Como su nombre lo dice este sistema consta de dos tipos de dispositivos, uno al que llamamos Maestro, el cual está encargado de iniciar las comunicaciones y es el que pide la información. Los dispositivos Esclavos, en tanto, solo envían información cuando se les solicita. El proceso por el cual el Maestro envía un mensaje y recibe una respuesta (en caso de haberla) se denomina Transacción.[19]

2.4.5.- Comunicación Punto a Punto

Como se puede apreciar en la figura 2.2, que es el lazo más simple, en el cual

tenemos un maestro primario (PC), y uno secundario (Handheld terminal), así como un esclavo (Dispositivo de campo). [22]

En las operaciones punto a punto el dispositivo de campo tiene la dirección cero (0) y su salida de corriente es de 4-20 mA.

En el caso de los instrumentos pasivos, es decir que obtienen alimentación del lazo se dispone una fuente que alimentará al lazo. Esta se conectará en serie al instrumento y a una resistencia de carga, los estándares HART permiten resistencias de 230 a 1100 Ω. [22]

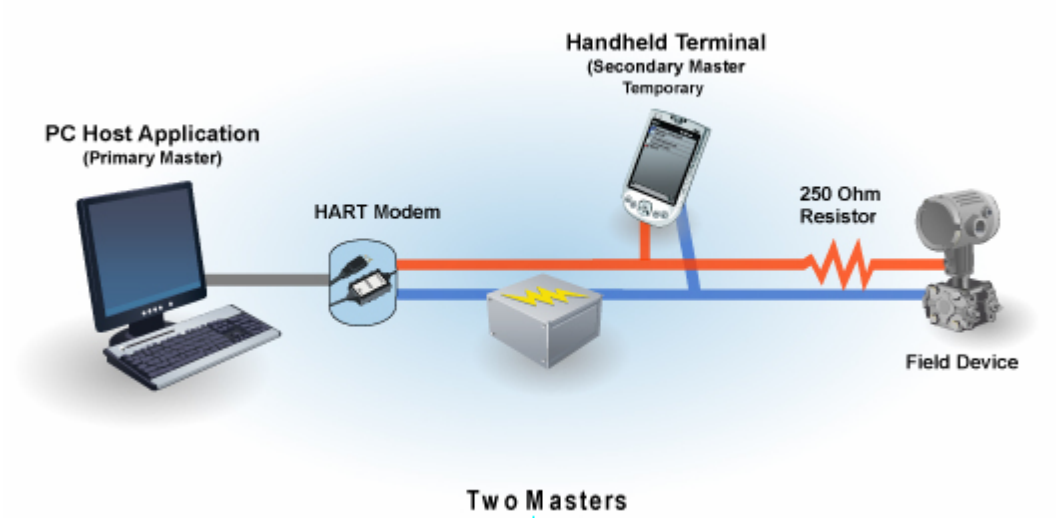

Figura 2.2. Comunicación punto a punto
Fuente: Fundación de comunicación HART

2.4.6.- Comunicación Multipunto

En este modo se pueden conectar hasta 15 instrumentos en paralelo, usando un par de cables, y una fuente en caso que se requiriera, como se ve en la figura 2.3. A

diferencia del modo punto a punto las direcciones de los dispositivos van del 1 al 15 y las salidas de corriente de cada uno se fijan en 4 mA. Para este modo de operación los controladores e indicadores deben contar con un HART modem. Las consecuencias más destacables de este modo de transmisión son dos, retardo en la comunicación Maestro - Esclavo, y perdida de la señal análoga, ya que, como se dijo anteriormente, la corriente de salida de cada instrumento se fija en 4 mA. [22]

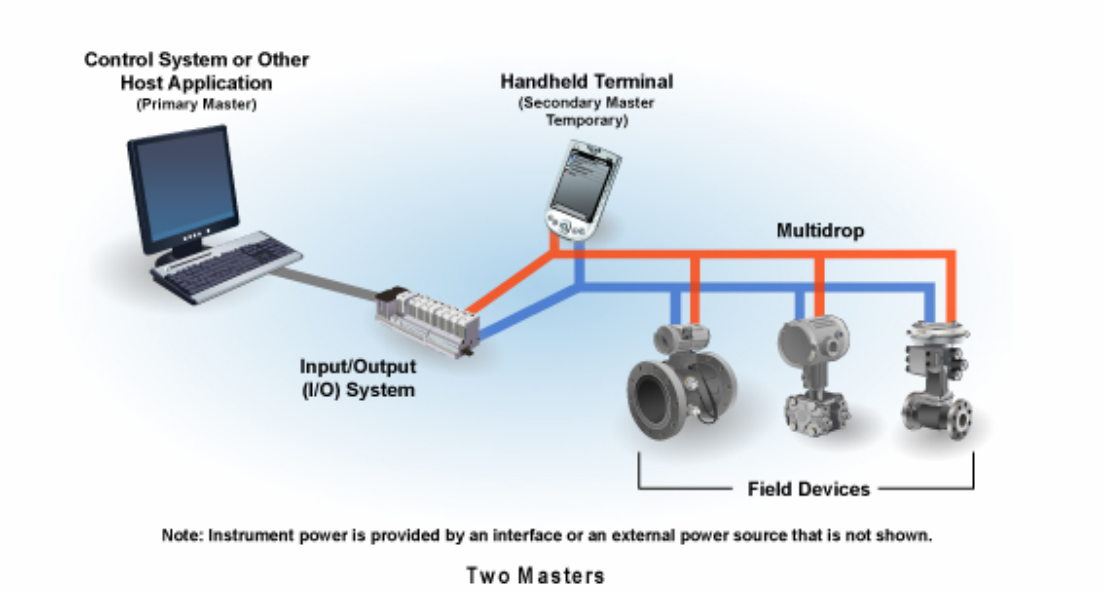

Figura 2.3. Configuración multidrop
Fuente: Fundación de comunicación HART

2.4.7.- Procedimiento de transacciones, código y estructura del mensaje

En esta sección se describe de modo más detallado los comandos del protocolo, los comandos universales, los comandos de práctica común y los comandos de

dispositivo específico. Incluyendo los tipos de datos que corresponden a cada uno de estos. Los bytes de estado son aquellos que indican errores en la comunicación, en esta sección se dan a conocer los valores de estos y su significado.

Se profundiza sobre la transacción de datos entre dispositivos Hart y la estructura de los mensajes, esto corresponde a la capa 2 o nivel 2 del protocolo de referencia o modelo OSI.

HART, como se ha mencionado anteriormente, es un protocolo de maestro-esclavo. Esto significa que cada transacción es originada por el maestro, el dispositivo de campo o esclavo solo responde cuando recibe un comando con su dirección. En la respuesta del esclavo se incluye un comando recibido, y puede que contenga los datos requeridos por el maestro. En el caso de que exista un maestro secundario, estos tienen direcciones diferentes, por lo cual pueden distinguir si la respuesta es para el principal o secundario.

2.4.7.1.- Procedimiento de transacción

HART es un protocolo Half-Duplex, con lo cual se quiere decir que al terminar cada mensaje, la portadora debe ser desactivada para permitir que la otra estación transmita. Las reglas de tiempo de la portadora establecen que la portadora debe ser activada no más del tiempo de 5 bits antes del inicio del mensaje (preámbulo) y ser desactivada no más del mismo tiempo después de la transmisión del último byte del mensaje (la suma de verificación). [18]

El maestro es el responsable de las transacciones de mensajes. Si no hay respuesta a un comando dentro de cierto tiempo, el maestro debe retransmitir el mensaje. Después de unos cuantos intentos debe abandonar la transacción y notificar el problema.

2.4.7.2.- El modo ráfaga

Para lograr una mayor tasa de transmisión de datos, algunos dispositivos utilizan el modo ráfaga. Cuando un dispositivo se encuentra en este modo envía un mensaje repetidas veces. Este modo se activa y desactiva mediante los comandos especiales #107, #108 y #109 (si se implementa el modo ráfaga, los comandos básicos son #1 y #3, los demás son opcionales). Existe una pequeña pausa entre mensaje y mensaje, para permitir que el maestro envíe la señal de desactivación, o para iniciar cualquier otra transacción simple. [19]

Este modo solo funciona para la configuración punto a punto, y se pueden enviar más de tres mensajes por segundo.

2.4.7.3.- Estructura del mensaje [10]

Cada mensaje incluye las direcciones de su fuente y destino, para asegurarse de que es recibido por el dispositivo correcto, y tiene una suma de verificación ("checksum") para poder detectar cualquier corrupción del mensaje, como se puede observar en la figura 2.4. El estado del dispositivo de campo está incluido en cada mensaje de respuesta, indicando su estado de operación correcto. Puede o no haber información o datos incluidos en el mensaje, dependiendo del comando en particular.

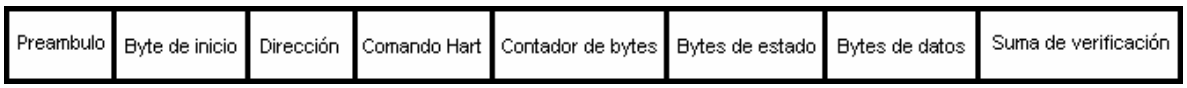

Figura 2.4 Estructura del mensaje HART

Fuente: Febres E., 2001

Existen el formato largo y el formato corto. Los primeros instrumentos Hart (inclusive la revisión 4) siempre utilizaron el formato corto. En este formato, la

dirección del esclavo un byte, de valor cero, para configuración punto-punto o del 1 al 15 para configuración multipunto. Esta corta dirección se denomina dirección multipunto. La revisión 5 introduce el formato largo. En este, la dirección del esclavo es un número de identificación único, un número de 38 bits derivado del código del fabricante, el código del tipo de dispositivo y el número de identificación del dispositivo. Este formato impide que los esclavos tomen mensajes que no le corresponden. De un modo estricto, el identificador único, no es único, puede haber hasta cuatro veces el mismo número, ya que del código del fabricante solo se toman 6 bits, cuando el número en realidad consta de 8 bits.

La mayoría de los dispositivos maestros deben incluir ambos formatos en su totalidad, de modo que puedan trabajar correctamente con los dispositivos ya existentes así como con los nuevos. La revisión 5 establece que todos los dispositivos deben implementar el comando #0 (leer identificación única) en ambos formatos del mensaje. Un maestro normalmente utilizará el comando # 0 para la primera conexión con el dispositivo, ya que en ese momento el número único de identificación no se conoce, sin embargo como el mensaje también incluye el nivel de revisión de HART, el maestro sabrá que formato deberá utilizar.

A continuación se describirán cada uno de los elementos que conforman a la estructura del mensaje HART:

El preámbulo

El preámbulo consiste de 5 a 20 bytes con caracteres hexadecimales FF (todos 1). Esto permite que el receptor sincronice la frecuencia de la señal y la cadena de caracteres que recibe, después de la detección inicial del mensaje Hart. Para el primer intento y cualquier intento sucesivo de comunicación, se deberían utilizar 20 bytes de preámbulo, para tener la mayor probabilidad de éxito. La respuesta al comando #0 le dice al maestro cuantos caracteres de preámbulo le gustaría recibir al dispositivo; el

maestro puede utilizar el comando #59 para indicarle cuantos bytes de preámbulo debe incluir en la respuesta.

El carácter de inicio (byte de inicio)

El carácter de inicio en Hart tiene diversos valores posibles, indicando cual formato está siendo utilizado, la fuente del mensaje, y si es o no un mensaje tipo ráfaga. Estos se muestran en la tabla 2.1.

Tabla 2.1 Valores del byte de inicio

Fuente: Febres E., 2001

Tipo de mensaje	Formato corto	Formato largo
Maestro a esclavo	02_{16}	82_{16}
Esclavo a maestro	06_{16}	86_{16}
Mensaje BURST del esclavo	01_{16}	81_{16}

Cuando están en espera de un mensaje, los receptores se encuentran en la búsqueda de estos caracteres, como el primer carácter después de por lo menos dos caracteres FF, para indicar el inicio del mensaje. Estos mensajes se pueden identificar completamente con el contenido de los bits 0, 1, 2 y 7. Se ha propuesto que para mejoras futuras se utilicen los bits 5 y 6 del carácter de inicio para indicar la presencia de bytes extra entre la dirección y el comando.

La dirección (address bytes)

El campo de dirección contiene tanto la dirección del maestro como la del esclavo del mensaje enviado. Está contenida en un byte, para el formato corto y en 5 bytes para el

formato largo. El bit más significativo de la dirección, indica si el maestro es el primario (1) o el secundario (0). Los mensajes de tipo ráfaga son una excepción, en la cual el dispositivo alterna ambas direcciones, lo que le da oportunidad a ambos maestros de interrumpir.

También en ambos formatos, el bit que le sigue al más significativo indica si el mensaje proviene de un dispositivo en modo ráfaga, lo que no implica que el mensaje sea de tipo ráfaga. En el formato corto, los dispositivos esclavos tienen direcciones de la cero a la quince. Este número se incluye de modo binario en los 4 bits menos significativos del byte de dirección. En el formato largo, la dirección de multipunto no es utilizada, en cambio, los 38 bits restantes de los cinco bytes del campo de direcciones contienen el identificador único como una dirección. En las figuras 2.5 y 2.6 se puede observar la estructura de las direcciones.

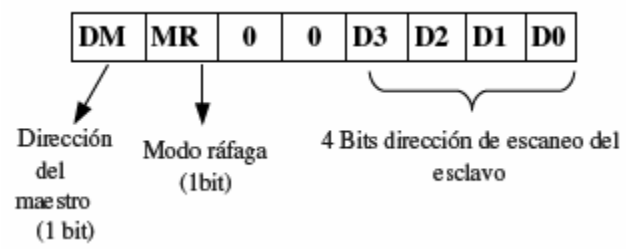

Figura 2.5 Estructura de la dirección en formato corto

Fuente: Febres E., 2001

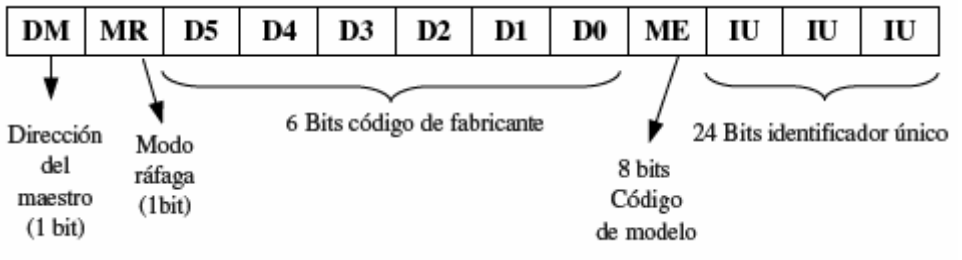

Figura 2.6 Estructura de la dirección en formato largo

Fuente: Febres E., 2001

En la estructura de formato largo, si se asigna cero a todos los bits, se puede utilizar como un mensaje de transmisión sin destinatario específico, un mensaje que sea aceptado por todos los dispositivos; esto es solo posible si los datos en el mensaje determinan cual de los dispositivos debe responder. Por ejemplo, el comando #11 (leer el identificador único asociado a la etiqueta) normalmente utiliza direcciones de transmisión sin destinatario específico con una etiqueta en el campo de datos, de modo que todos los dispositivos conectados reciben el mensaje pero solo uno de ellos responde.

Comando

El campo de comando contiene un entero del 0 al hexadecimal FD o al decimal 253, como su nombre lo indica representa el comando HART. El comando recibido se incluye en la respuesta del esclavo al ser enviada. Ya que para cada comando se define una estructura específica para el campo de datos, y una respuesta en particular, se dedica una sección a éste campo.

Contador de bytes

Este campo contiene un entero, que indica el número de bytes que forman el resto del mensaje (eso es los campos de estado y de datos, la suma de verificación no se incluye). El dispositivo receptor utiliza esto para identificar el byte de suma de verificación y saber cuando el mensaje está completo. Como el campo de datos está

limitado a 25 bytes máximo, esta cuenta puede ser cualquier número entre 0 y 27.

Bytes de estado

El campo de estado también es llamado el "código de respuesta", solo se incluye en el mensaje de respuesta de un esclavo. Consta de dos bytes, que reportan cualquier error de comunicación, el estado del comando recibido (como por ejemplo dispositivo ocupado o que no reconoce dicho comando), y el estado de operación del esclavo.

Bytes de datos

No todas las respuestas contienen datos. Para aquellas que si lo hacen, y de modo que cumplan con las reglas de tiempo, el campo de datos no puede exceder los 25 bytes. Los datos pueden estar en forma de enteros sin signo, números de punto flotante o cadenas de caracteres ASCII. El número de bytes del campo de datos, y el formato de datos utilizado para cada ítem se especifican de acuerdo al comando recibido.

Suma de verificación (checksum)

El byte de suma de verificación contiene el OR exclusivo (paridad longitudinal) de todos los bytes que le preceden en el mensaje, comenzando con el carácter de inicio. Esto provee un segundo chequeo para la integridad de la transmisión después del de paridad por byte. La combinación de estos dos garantiza la detección de hasta tres errores en un mensaje y tiene buenas probabilidades de detectar errores en más bits.

2.4.7.4.- Ejemplos de transacciones en formato corto y largo

En las figuras 2.7 y 2.8 se observan la estructura del formato corto y largo respectivamente. En cada mensaje, los valores de los bytes se muestran en hexadecimal, con los campos de dirección escrito de modo binario para mostrar claramente su composición. Los nombres de cada campo se encuentran indicados con

sus siglas en ingles. *Start* es el byte de inicio, ***Com*** es el byte de comando, *Bcnt* es el byte de cuenta de bytes y el *Chks* es el byte de suma de verificación.

2.4.7.5.- Comandos y datos respectivos

El campo de comandos como ya se mencionó en la sección anterior, contiene un entero entre 0 y 253, en decimal, que representa los comandos de Hart. Los números 31_{10}, 127_{10}, 254_{10} y 255_{10} se encuentran reservados. Además, los comandos se dividen en tres grupos específicos, los comandos universales, los comandos de práctica común y los comandos específicos del dispositivo.

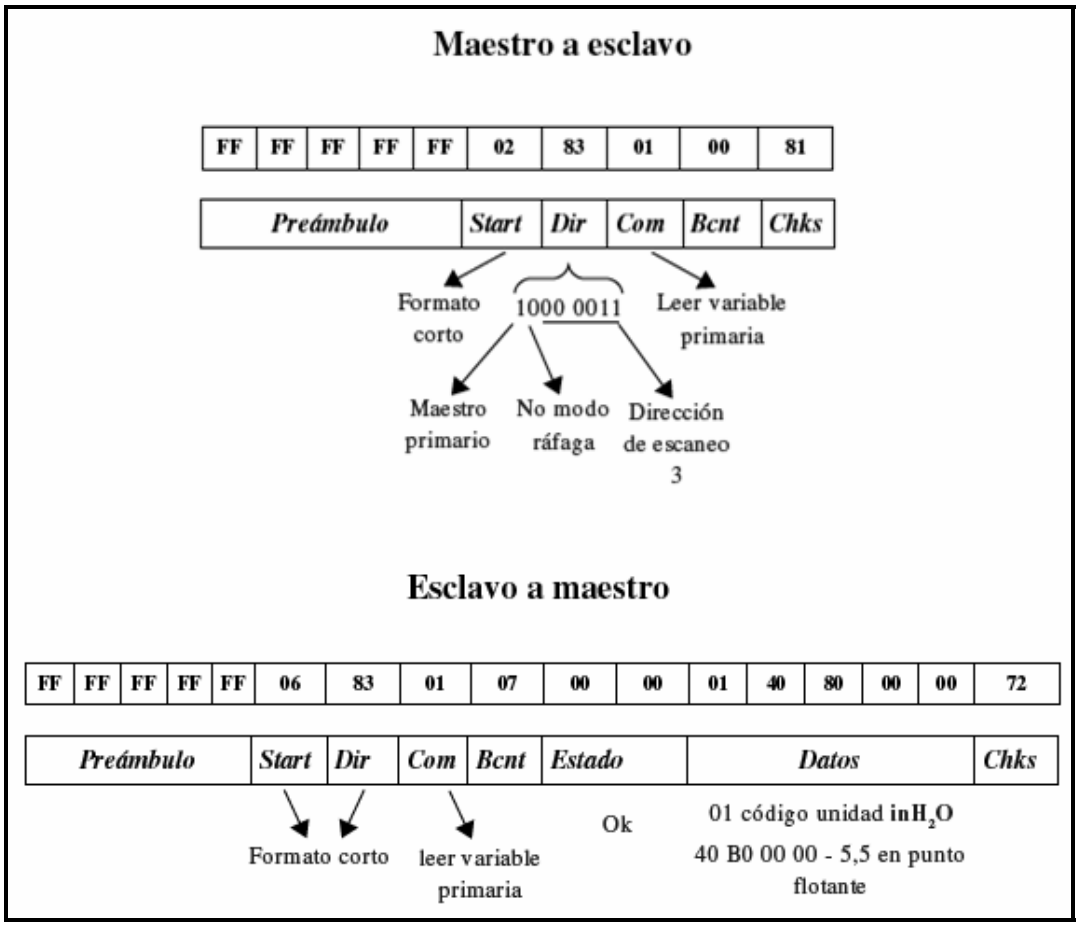

Figura 2.7 Transacciones en formato corto

Fuente: Febres E., 2001

- **Comandos universales**

Los comandos universales se encuentran entre 0 y 30. Estos proveen funciones que son implementadas en todos los dispositivos Hart. La tabla 2.2 contiene un resumen de estas funciones. [10]

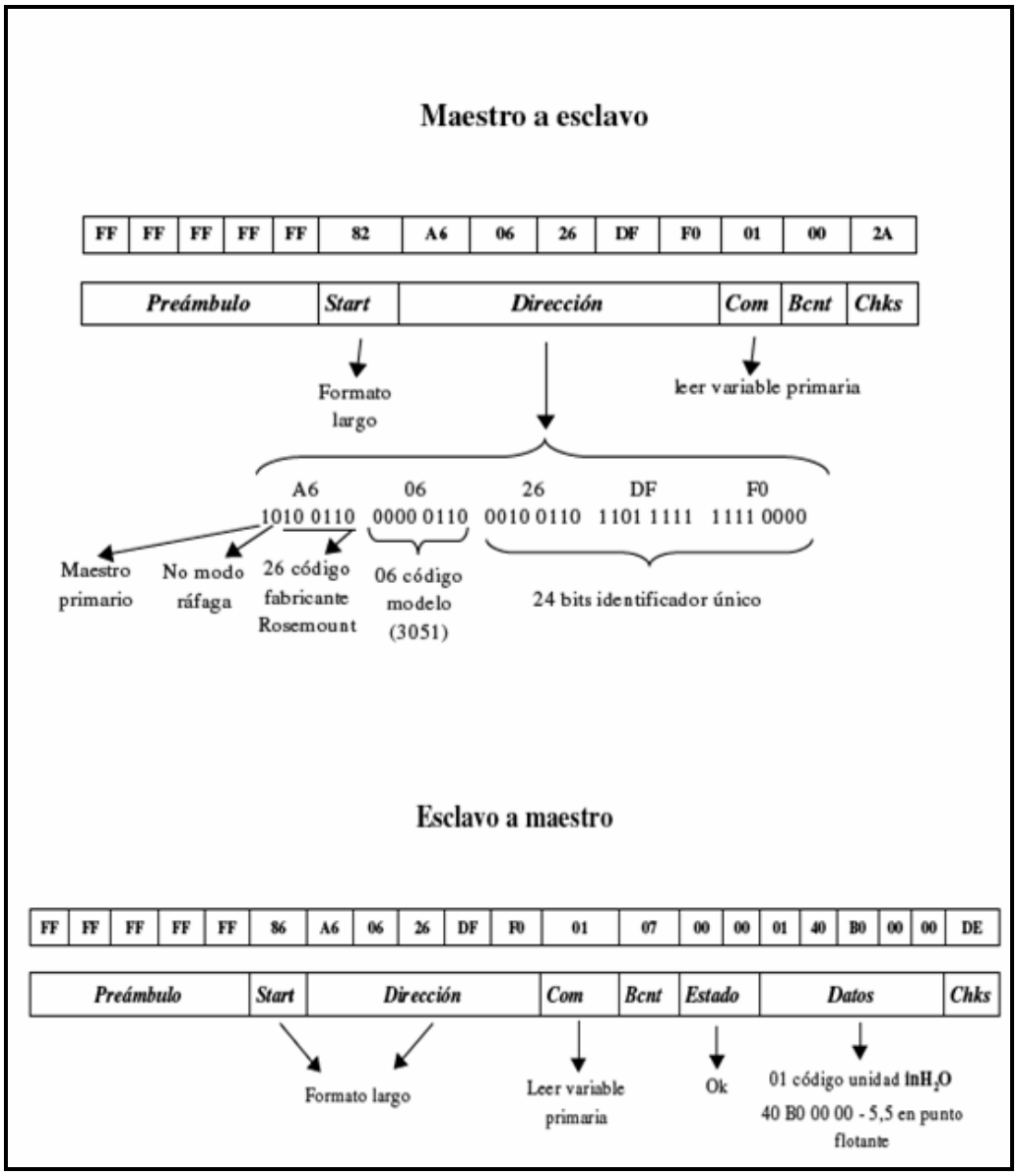

Figura 2.8 Transacciones en formato largo

Fuente: Febres E., 2001

Tabla 2.2 Comandos universales

Fuente: Febres E., 2001

Comandos	Función
0,11	Identificar dispositivo (fabricante, tipo de dispositivo, etiqueta de revisión)
1,2,3	Leer variables medidas
6	Establecer dirección de escaneo. (modo multipunto)
12,13,17,18	Leer y escribir información introducida por el usuario (tag, fecha, mensaje)
14,15	Leer información del dispositivo (numero serial del sensor, límites del sensor, operación de alarma, valores del rango, función de transferencia, constante de tiempo de muestreo)
16,19	Leer y escribir número final de ensamble

Los comandos #0 y #11 (comandos universales)

Los comandos $\#0_{10}$ y $\#11_{10}$, son utilizados para identificar un dispositivo de campo. Desde la revisión 5, todos los equipos utilizan el formato largo, pero el comando $\#0_{10}$ debe ser también aceptado. Esto permite que el maestro Hart identifique un dispositivo nuevo, sin antes saber su número de identificación único. Los datos en la respuesta al comando $\#0_{10}$ incluyen el código de identificación del fabricante, el código del tipo de dispositivo, y el número de identificación del mismo. De estos, el maestro puede construir el número de identificación única para ser utilizado en los mensajes siguientes.

Los comandos #1, #2 y #3 (comandos universales)

Estos se utilizan para leer las variables medidas de diversas formas. Los comandos $\#2_{10}$ y $\#3_{10}$ incluyen la corriente actual de salida en *mA*. Como la verdadera salida analógica, estos valores en mA representan la variable primaria (*VP*) solo cuando está dentro del rango configurado. No cuando el dispositivo se encuentra en operación multipunto ni tampoco cuando la salida tiene un valor fijo, saturado o un valor fuera del rango. Sin embargo, la *VP* y otras variables dinámicas retornadas con sus respectivas unidades a través de estos comandos, no son limitadas por el rango establecido, sino que siguen la medición del sensor. El porcentaje del rango indicado por el comando $\#2_{10}$ también sigue la salida del sensor fuera de los límites, de modo que se puede ir del 0 al 100 % y por encima de esto.

El comando de práctica común $\#61_{10}$ es equivalente al comando $\#3_{10}$, para instrumentos similares con salidas analógicas diferentes a corriente. El comando $\#110_{10}$ también devuelve variables dinámicas (sin el nivel de salida de la señal analógica). El comando $\#33_{10}$ provee una selección de hasta cuatro variables del transmisor. Para múltiples dispositivos de salida, el comando $\#60_{10}$ lee cualquier nivel de señal analógica de salida seleccionada y el porcentaje del rango, y finalmente el comando $\#62_{10}$ provee la selección de hasta 4 niveles de salida.

El comando #6

Este comando establece la dirección de escáner del dispositivo. Cuando se establece en cero, el dispositivo funciona en modo punto a punto, generando una señal analógica de salida. Para cualquier valor entre 1 y 15, el dispositivo se cambia al modo multipunto y su salida analógica se fija en 4 mA.

Los comandos #12 y #19

Estos se utilizan para leer y escribir una selección de la información del dispositivo. Para las revisiones menores e igual a 4, los comandos eran $\#4_{10}$ y $\#5_{10}$, con números de bloques utilizados para seleccionar una particular sección de la información.

- **Comandos de práctica común**

Estos se encuentran en el rango 32 a 126. Proveen funciones comunes a muchos dispositivos de campo. Si estas funciones son implementadas en el dispositivo, estos comandos deben ser utilizados para invocarlas. En la tabla 2.3 se observa un resumen de dichos comandos.

Los comandos de práctica común $\#123_{10}$ y $\#126_{10}$ no son "públicos". Típicamente son utilizados por los fabricantes para insertar información específica del dispositivo durante su instalación, por ejemplo el número de identificación del dispositivo, que nunca será alterado por los usuarios, o para comandos de lectura y escritura directa a la memoria. Frecuentemente se necesita de una clave para acceder estos comandos.

Los comandos de práctica común, del $\#50_{10}$ al $\#56_{10}$ están relacionados a estas variables del transmisor, sus sensores y rangos. En particular, en dispositivos que lo implementan, el comando $\#51_{10}$ permite la selección de las variables del transmisor para las primeras cuatro variables. Estas pueden ser leídas utilizando el comando $\#3_{10}$. De otro modo, el comando $\#33_{10}$ específica cuatro variables para ser enviadas en un mismo mensaje.

Los transmisores multivariables también tienen la posibilidad de más de una salida analógica. Por definición, las salidas enumeradas 1 a 4 representan las variables dinámicas de Hart (VP, VS, VT y VC) respectivamente. Los comandos de práctica común $\#60_{10}$ y $\#62_{10}$ al $\#70_{10}$ tienen que ver con la configuración y control de estas salidas.

- **Comandos específicos de dispositivo**

Los comandos específicos de dispositivo se encuentran en el rango 128 a 253. Sus funciones son más o menos únicas para cada dispositivo. En la revisión 4 y anteriores, los comandos específicos de dispositivo siempre incluían el código del tipo de dispositivo como el primer byte del campo de datos, para asegurarse de que un comando nunca llegará a un dispositivo no compatible. Esto fue abandonado en la revisión 5, cuando se incluyó el número identificador único, que cumple con la misma función.

Tabla 2.3 Comandos de práctica común

Fuente: Febres E., 2001

Comandos	Función
33,61,110	Leer variables medidas
34-37,44,47	Establecer parámetros de operación (rango, unidades, función de transferencia)
38	Reiniciar la bandera de "Configuración modificada"
39	Control de la EEPROM
40-42	Funciones de diagnóstico (modo de corriente fija, auto prueba, reset)
43, 45, 46	Ajuste de la entrada y salida analógica
48	Leer estados adicionales
49	Escribir el número serial del sensor
50-56	Uso de variable del transmisor
57-58	Información de la unidad (tag, descriptor, fecha)
59	Escribir el número de bytes de preámbulo necesitados
60, 62-70	Uso de múltiples salidas analógicas
107 - 109	Control del modo ráfaga

No todas las respuestas a comandos contienen datos. Para esos que si lo hacen, se pueden incluir un máximo de 25 bytes. Los datos pueden ser representados como:

- Enteros 8, 16, 24 o 32 bits sin signo.
- Números de punto flotante – Formato de IEEE 754 de punto flotante de precisión.
- Cadenas de caracteres ASCII usualmente 4 caracteres por cada 3 bytes.
- Ítems enumerados para una lista estándar.

Si un comando no tiene éxito (indicado por error en el campo de estado), las respuestas no deben contener datos. La respuesta a un comando exitoso siempre incluye el mismo set de variables como las contenía el mensaje de comando; sin embargo, los valores en la respuesta son los actualmente utilizados, tomados de la memoria del dispositivo de campo, al igual que cualquier aproximación involucrada. El número de bytes de datos, y el formato de los mismos (de cada elemento) son especificados para cada comando.

- **Elementos**

Los elementos de datos para los cuales se permite seleccionar de una lista de alternativas se codifican como números que corresponden a cada alternativa. La tabla 2.4 muestra algunas de las listas enumeradas estándar. Existen también muchas listas específicas para cada dispositivo.

2.5.- Modelos de Networking [23]

Generalmente, la información que se desplaza por una red recibe el nombre de datos o paquete. Un paquete es una unidad de información, lógicamente agrupada, que se desplaza entre los sistemas de computación. A medida que los datos atraviesan las

capas, cada capa agrega información que posibilita una comunicación eficaz con su correspondiente capa en el otro computador.

Tabla 2.4 Variables enumeradas

Fuente: Febres E., 2001

Variable	Valores
Identificación del fabricante	1-249, establecidos por la fundación HART
Tipo de dispositivo	0-249, establecidos por cada fabricante
Unidades	0-249: 6 = psi, 7 = bar, 32 = C, 33 = F, etc.
Función de transferencia	0 = lineal, 1= raíz cuadrada, etc.
Material	0-249: 2 =316 acero inoxidable, 10 = PTFE, 18 = cerámica, etc
Selección de alarma	0 = bajo, 1 = alto, 239 = mantener el último valor de la salida
Protección a escritura	0 = no protegido contra escritura, 1 = protegido
Control del modo RÁFAGA	0 = salir del modo RÁFAGA, 1 = activar el modo RÁFAGA.
Señalización física	0 = corriente BELL 202, 1 = voltaje BELL 202, 2 = RS485.

Los modelos OSI y TCP/IP se dividen en capas que explican cómo los datos se comunican de un computador a otro. Los modelos difieren en la cantidad y la función de las capas. No obstante, se puede usar cada modelo para ayudar a describir y brindar detalles sobre el flujo de información desde un origen a un destino.

2.5.1.- Modelo OSI [23]

En sus inicios, el desarrollo de redes sucedió con desorden en muchos sentidos. A principios de la década de 1980 se produjo un enorme crecimiento en la cantidad y el tamaño de las redes. A medida que las empresas tomaron conciencia de las ventajas de usar tecnología de 'networking', las redes se agregaban o expandían casi a la misma velocidad a la que se introducían las nuevas tecnologías de red.

Para mediados de la década de 1980, estas empresas comenzaron a sufrir las consecuencias de la rápida expansión. De la misma forma en que las personas que no hablan un mismo idioma tienen dificultades para comunicarse, las redes que utilizaban diferentes especificaciones e implementaciones tenían dificultades para intercambiar información. El mismo problema surgía con las empresas que desarrollaban tecnologías de 'networking' privadas o propietarias. "Propietario" significa que una sola empresa o un pequeño grupo de empresas controlan todo uso de la tecnología. Las tecnologías de 'networking' que respetaban reglas propietarias en forma estricta no podían comunicarse con tecnologías que usaban reglas propietarias diferentes.

Para enfrentar el problema de incompatibilidad de redes, la Organización Internacional de Normalización (ISO) investigó modelos de 'networking' como la red de Digital Equipment Corporation (DECnet), la Arquitectura de Sistemas de Red (SNA) y TCP/IP a fin de encontrar un conjunto de reglas aplicables de forma general a todas las redes. En base a esta investigación, la ISO desarrolló un modelo de red que ayuda a los fabricantes a crear redes que sean compatibles con otras redes.

El modelo de referencia de Interconexión de Sistemas Abiertos (OSI) lanzado en 1984 fue el modelo de red descriptivo creado por ISO por sus siglas en ingles

(*International Organization for Standardization)*. Proporcionó a los fabricantes un conjunto de estándares que aseguraron una mayor compatibilidad e interoperabilidad entre los distintos tipos de tecnología de red producidos por las empresas a nivel mundial.

El modelo de referencia OSI es un marco que se puede utilizar para comprender cómo viaja la información a través de una red. El modelo de referencia OSI explica de qué manera los paquetes de datos viajan a través de varias capas a otro dispositivo de una red, aun cuando el remitente y el destinatario poseen diferentes tipos de medios de red.

En el modelo de referencia OSI, como se puede observar en la figura 2.9, hay siete capas numeradas, cada una de las cuales ilustra una función de red específica.

Ventajas del Modelo OSI:

- Reduce la complejidad.
- Estandariza las interfaces
- Facilita el diseño modular
- Asegura la interoperabilidad de la tecnología
- Acelera la evolución
- Simplifica la enseñanza y el aprendizaje

Figura 2.9. Modelo OSI

Fuente: www.cisco.com

2.5.2.- Modelo TCP/IP [23]

El estándar histórico y técnico de la Internet es el modelo TCP/IP. El Departamento de Defensa de EE.UU. (DoD) creó el modelo de referencia TCP/IP porque necesitaba diseñar una red que pudiera sobrevivir ante cualquier circunstancia, incluso una guerra nuclear. En un mundo conectado por diferentes tipos de medios de comunicación, como alambres de cobre, microondas, fibras ópticas y enlaces satelitales, el DoD quería que la transmisión de paquetes se realizara cada vez que se iniciaba y bajo cualquier circunstancia. Este difícil problema de diseño dio origen a la creación del modelo TCP/IP.

El modelo TCP/IP se desarrolló como un estándar abierto. Esto significaba que cualquier persona podía usar el TCP/IP. Esto contribuyó a acelerar el desarrollo de

TCP/IP como un estándar. Como se observa en la figura 2.10. El modelo TCP/IP tiene cuatro capas.

Figura 2.10. Modelo TCP/IP

Fuente: www.cisco.com

Aunque algunas de las capas del modelo TCP/IP tienen el mismo nombre que las capas del modelo OSI, las capas de ambos modelos no se corresponden de manera exacta. Lo más notable es que la capa de aplicación posee funciones diferentes en cada modelo.

Los diseñadores de TCP/IP sintieron que la capa de aplicación debía incluir los detalles de las capas de sesión y presentación OSI. Crearon una capa de aplicación que maneja aspectos de representación, codificación y control de diálogo.

La capa de transporte se encarga de los aspectos de calidad del servicio con respecto a la confiabilidad, el control de flujo y la corrección de errores. Uno de sus protocolos, el protocolo para el control de la transmisión (TCP), ofrece maneras flexibles y de alta calidad para crear comunicaciones de red confiables, sin problemas de flujo y con un nivel de error bajo.

TCP es un protocolo orientado a conexión. Mantiene un diálogo entre el origen y el destino mientras empaqueta la información de la capa de aplicación en unidades denominadas segmentos. Orientado a conexión no significa que existe un circuito entre los computadores que se comunican. Significa que segmentos de la Capa 4 viajan de un lado a otro entre dos hosts para comprobar que la conexión exista lógicamente para un determinado período.

El propósito de la capa Internet es dividir los segmentos TCP en paquetes y enviarlos desde cualquier red. Los paquetes llegan a la red de destino independientemente de la ruta que utilizaron para llegar allí. El protocolo específico que rige esta capa se denomina Protocolo Internet (IP). En esta capa se produce la determinación de la mejor ruta y la conmutación de paquetes.

La relación entre IP y TCP es importante. Se puede pensar en el IP como el que indica el camino a los paquetes, en tanto que el TCP brinda un transporte seguro.

El nombre de la capa de acceso de red es muy amplio y se presta a confusión. También se conoce como la capa de host a red. Esta capa guarda relación con todos los componentes, tanto físicos como lógicos, necesarios para lograr un enlace físico. Incluye los detalles de tecnología de ‘networking’, y todos los detalles de la capa física y de enlace de datos del modelo OSI.

La figura 2.11 ilustra algunos de los protocolos comunes especificados por las capas del modelo de referencia TCP/IP. Algunos de los protocolos de capa de aplicación más comúnmente usados incluyen los siguientes:

- Protocolo de Transferencia de Archivos (FTP)
- Protocolo de Transferencia de Hipertexto (HTTP)
- Protocolo simple de transferencia de correo (SMTP)
- Sistema de denominación de dominios (DNS)
- Protocolo Trivial de Transferencia de Archivos (TFTP)

Los protocolos de capa de transporte comunes incluyen:

- Protocolo para el Control del Transporte (TCP)
- Protocolo de Datagrama de Usuario (UDP)

El protocolo principal de la capa Internet es:

- Protocolo Internet (IP)

La capa de acceso de red se refiere a cualquier tecnología en particular utilizada en una red específica.

Independientemente de los servicios de aplicación de red que se brinden y del protocolo de transferencia que se utilice, existe un solo protocolo de Internet, IP. Esta es una decisión de diseño deliberada. IP sirve como protocolo universal que permite que cualquier computador en cualquier parte del mundo pueda comunicarse en cualquier momento.

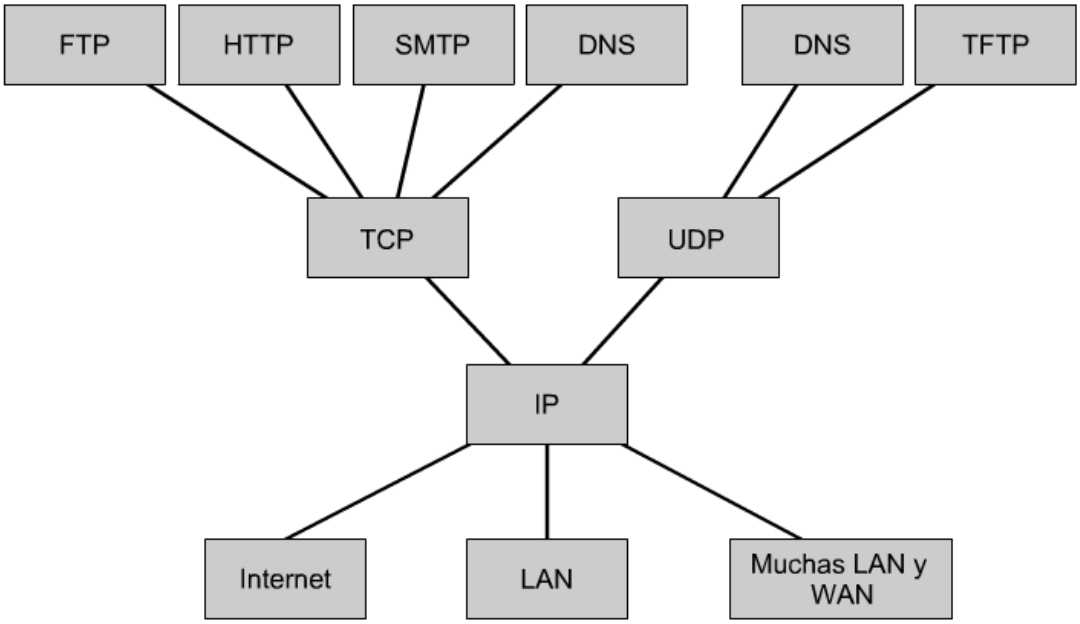

Figura 2.11. Protocolos comunes

Fuente: www.cisco.com

2.5.3.- Comparación entre los modelos TCP/IP y OSI [21]

Comparando el modelo OSI con los modelos TCP/IP, surgen algunas similitudes y diferencias.

Las similitudes incluyen:

- Ambos se dividen en capas.
- Ambos tienen capas de aplicación, aunque incluyen servicios muy distintos.
- Ambos tienen capas de transporte y de red similares.
- Ambos modelos deben ser conocidos por los profesionales de 'networking'.
- Ambos suponen que se conmutan paquetes. Esto significa que los paquetes individuales pueden usar rutas diferentes para llegar al mismo destino. Esto se

contrasta con las redes conmutadas por circuito, en las que todos los paquetes toman la misma ruta.

Las diferencias incluyen:

- TCP/IP combina las funciones de la capa de presentación y de sesión en la capa de aplicación.
- TCP/IP combina la capa de enlace de datos y la capa física del modelo OSI en la capa de acceso de red.
- TCP/IP parece ser más simple porque tiene menos capas.
- Los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló la Internet, de modo que la credibilidad del modelo TCP/IP se debe en gran parte a sus protocolos. En comparación, por lo general las redes no se desarrollan a partir del protocolo OSI, aunque el modelo OSI se usa como guía.

2.6.- Protocolo TCP/IP [23]

TCP/IP es una combinación de dos protocolos individuales. IP por sus siglas en ingles (*Internet Protocol),* opera en la Capa de red (capa 3) y es un servicio no orientado a conexión que proporciona una entrega de máximo esfuerzo a través de una red. TCP por sus siglas en ingles (*Transmission Control Protocol*), opera en la capa de transporte (capa 4), y es un servicio orientado a conexión que suministra control de flujo y confiabilidad. Al unir estos protocolos, se suministra una gama de servicios más amplia. De forma conjunta, constituyen la base para un conjunto completo de protocolos que se denomina conjunto de protocolos TCP/IP. La Internet se basa en este conjunto de protocolos TCP/IP. A continuación se describe detalladamente las capas del modelo TCP/IP donde operan los protocolos TCP e IP.

2.6.1.- Capa de transporte

La capa de transporte proporciona servicios de transporte desde el host origen hacia el host destino. Esta capa forma una conexión lógica entre los puntos finales de la red, el host transmisor y el host receptor. Los protocolos de transporte (Ver figura 2.12) segmentan y re ensamblan los datos mandados por las capas superiores en el mismo flujo de datos, o conexión lógica entre los extremos. La corriente de datos de la capa de transporte brinda transporte de extremo a extremo.

Generalmente, se compara la Internet con una nube. La capa de transporte envía los paquetes de datos desde la fuente transmisora hacia el destino receptor a través de la nube. El control de punta a punta, que se proporciona con las ventanas deslizantes y la confiabilidad de los números de secuencia y acuses de recibo, es el deber básico de la capa de transporte cuando utiliza TCP. La capa de transporte también define la conectividad de extremo a extremo entre las aplicaciones de los hosts. Los servicios de transporte incluyen los siguientes servicios:

TCP y UDP

- Segmentación de los datos de capa superior.
- Envío de los segmentos desde un dispositivo en un extremo a otro dispositivo en otro extremo.

TCP solamente

- Establecimiento de operaciones de punta a punta.
- Control de flujo proporcionado por ventanas deslizantes.

- Confiabilidad proporcionada por los números de secuencia y los acuses de recibo.

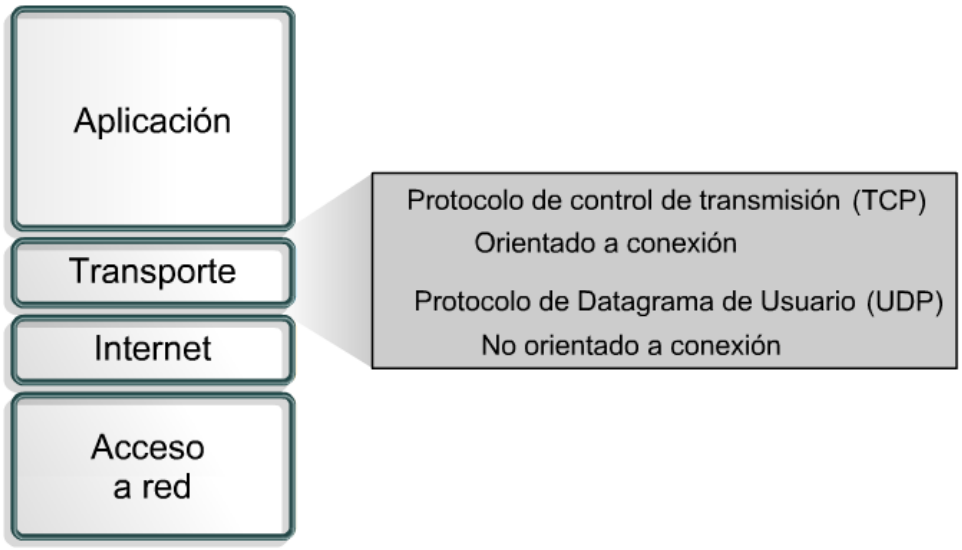

Figura 2.12. Protocolos de la capa de transporte

Fuente: www.cisco.com

2.6.2.- Capa de Internet

El propósito de la capa de Internet es seleccionar la mejor ruta para enviar paquetes por la red. El protocolo principal que funciona en esta capa es el Protocolo de Internet (IP). La determinación de la mejor ruta y la conmutación de los paquetes ocurren en esta capa.

En la figura 2.13 se observan protocolos que operan en la capa de Internet TCP/IP y se describen a continuación:

- IP proporciona un enrutamiento de paquetes no orientado a conexión de máximo esfuerzo. El IP no se ve afectado por el contenido de los paquetes, sino que busca una ruta hacia el destino.

- El Protocolo de mensajes de control en Internet (ICMP) suministra capacidades de control y envío de mensajes.
- El Protocolo de resolución de direcciones (ARP) determina la dirección de la capa de enlace de datos, la dirección MAC, para las direcciones IP conocidas.
- El Protocolo de resolución inversa de direcciones (RARP) determina las direcciones IP cuando se conoce la dirección MAC.

El IP ejecuta las siguientes operaciones:

- Define un paquete y un esquema de direccionamiento.
- Transfiere los datos entre la capa Internet y las capas de acceso de red.
- Enruta los paquetes hacia los hosts remotos.

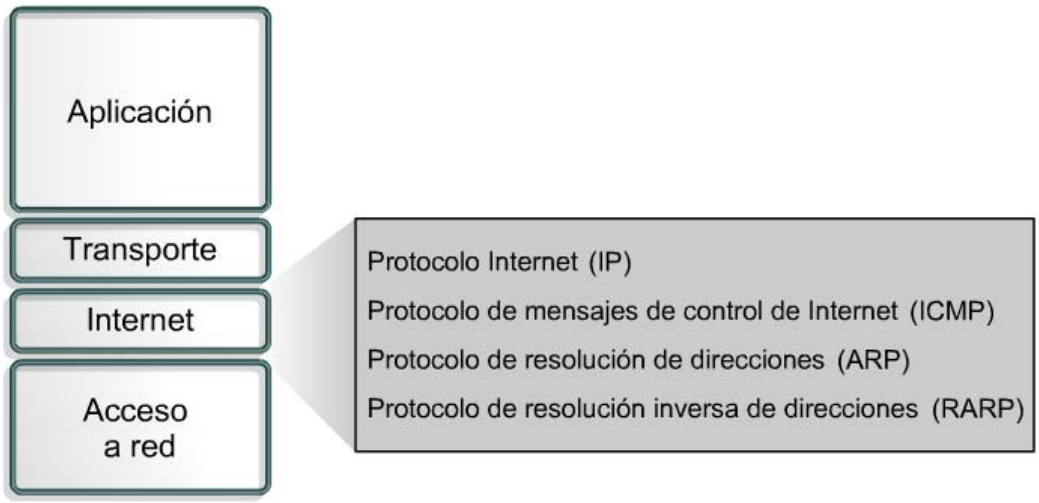

Figura 2.13. Protocolos de la capa de internet

Fuente: www.cisco.com

Por último, a modo de aclaración de la terminología, a veces, se considera a IP como protocolo poco confiable. Esto no significa que IP no enviará correctamente los datos a través de la red. Llamar al IP, protocolo poco confiable simplemente significa que IP no realiza la verificación y la corrección de los errores. Dicha función la realizan los protocolos de la capa superior desde las capas de transporte o aplicación.

2.7.- Sockets [11]

En los orígenes de Internet, las primeras computadoras en implementar sus protocolos fueron aquellas de la universidad de Berkeley. Dicha implementación tuvo lugar en una variante del sistema operativo Unix conocida como BSD Unix. Pronto se hizo evidente que los programadores necesitarían un medio sencillo y eficaz para escribir programas capaces de intercomunicarse entre sí. Esta necesidad dio origen a la primera especificación e implementación de sockets, también en Unix. Hoy día, los sockets están implementados como bibliotecas de programación para multitud de sistemas operativos, simplificando la tarea de los programadores.

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Para ello son necesarios los tres recursos que originan el concepto de socket:

- Un protocolo de comunicaciones, que permite el intercambio de octetos.
- Una dirección del Protocolo de Red (Dirección IP, si se utiliza el Protocolo TCP/IP), que identifica una computadora.

- Un número de puerto, que identifica a un programa dentro de una computadora.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación ha de ser iniciada por uno de los programas que se denomina programa cliente. El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa servidor.

Un socket es un fichero existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Las propiedades de un socket dependen de las características del protocolo en el que se implementan. El protocolo más utilizado es TCP, aunque también es posible utilizar UDP o IPX. Gracias al protocolo TCP, los sockets tienen las siguientes propiedades:

- Orientado a conexión.
- Se garantiza la transmisión de todos los octetos sin errores ni omisiones.
- Se garantiza que todo octeto llegará a su destino en el mismo orden en que se ha transmitido.

Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información.

El protocolo UDP es un protocolo no orientado a la conexión. Sólo se garantiza que si un mensaje llega, llegue bien. En ningún caso se garantiza que llegue o que

lleguen todos los mensajes en el mismo orden que se mandaron. Esto lo hace adecuado para el envío de mensajes frecuentes pero no demasiado importantes.

2.8.- Proceso Unificado Racional

El Proceso Unificado Racional (RUP) es una metodología proporcionada por Rational Software para desarrollar sistemas con gran cantidad de software basado en componentes. El Proceso Unificado usa UML para preparar todas las plantillas y modelos del software en construcción.

En RUP se puede ver la evolución del software en cuatro fases como se muestra en la figura 2.14, al final de las cuales, y tras una serie de iteraciones, establece objetivos a alcanzar bien definidos.

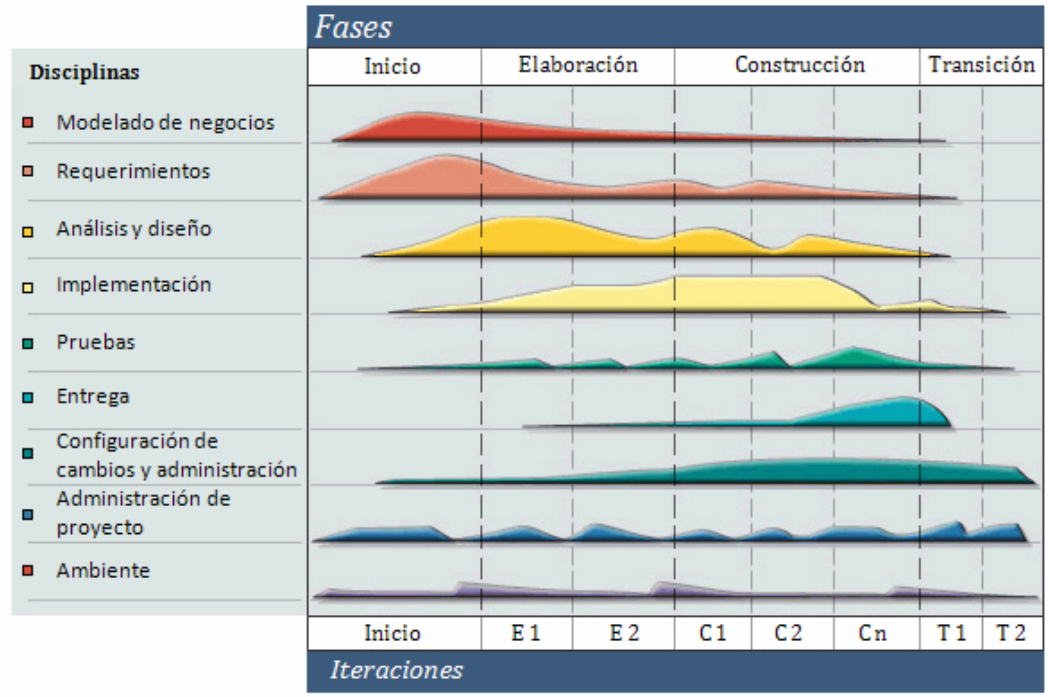

Figura 2.14. Fases del proceso unificado

Fuente: Jacobson, I, Booch, G, Rumbaugh J, 2000

2.8.1.- Fases del RUP

A continuación se describen cada una de las fases del proceso unificado racional:

2.8.1.1.- Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar qué recursos deben ser asignados al proyecto. [17]

Objetivos

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre. [20]

Resultados

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de Casos de Uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.

- Modelo de negocio, si es necesario.
- Prototipos exploratorios para probar conceptos o la arquitectura candidata. [17]

Criterios de evaluación

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados. [17]

2.8.1.2.- Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. [17]

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.[17]

Objetivos

- Definir, validar y cimentar la arquitectura.

- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Resultados

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional). [18]

Criterios de evaluación:

- La visión del producto es estable.

- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación es éstas fases quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

2.8.1.3.- Construcción [17]

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Objetivos

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.

- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Resultados

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación).
- Arquitectura íntegra (mantenida y mínimamente actualizada).
- Riesgos Presentados Mitigados.
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle).
- Prototipo Operacional – beta.
- Caso del Negocio Actualizado

Criterios de evaluación:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

2.8.1.4.- Transición [17]

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Objetivos

Conseguir que el usuario se valga por sí mismo.

Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Resultados

- Prototipo Operacional.
- Documentos Legales.
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema.
- Descripción de la arquitectura completa y corregida.
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

***Criterios de evaluación*:**

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

Cada fase se completa con la realización de varias iteraciones en las que se desarrollan una serie de actividades, que el modelo RUP clasifica en 9 disciplinas que tienen más o menos importancia en función de lo cerca que se esté o no de la finalización del proyecto.

- **Modelado del negocio**. En este conjunto de actividades se persigue el entendimiento de las necesidades de negocio. Documentos de requisitos generales y de alto nivel, reglas del negocio, glosarios, etc. ayudan a definir lo que el producto software deba hacer.
- **Requisitos**. Traduce las necesidades del modelo de negocio a requisitos de sistemas automatizables y que con carácter más técnico (se emplean los casos de uso UML), persiguen obtener un entendimiento más profundo del modelo de negocio por parte de los integrantes del equipo de desarrollo.
- **Análisis y diseño**. Estas actividades determinan, a partir de los requisitos la arquitectura del sistema más adecuada y el diseño detallado necesario previo a las actividades de implementación.
- **Implementación**. Actividades de codificación del software que de acuerdo al diseño, cumplen con los requisitos del sistema.
- **Pruebas**. Comprobaciones hechas a todos los elementos que se producen (documentos, diseños o código) para ver que cumplen con los requisitos y con los estándares de calidad definidos para el proyecto.

- **Despliegue**. Actividades que permiten tener el sistema instalado en los entornos en que finalmente va a ser explotado.
- **Gestión de configuración**. Gestión de los cambios y todos los elementos que intervienen en el proceso de construcción
- **Gestión del proyecto**. Actividades encaminadas a la gestión del desarrollo en cuanto a planes, recursos, seguimiento y control y gestión de riesgos.
- ***Entorno.*** Actividades que van encaminadas a dotar al proyecto de recursos hardware y software para facilitar la puesta en marcha y mantenimiento de los distintos entornos de desarrollo y pruebas o la propia puesta en producción del sistema.

RUP establece lo que denomina **buenas prácticas** como forma de trabajo adecuada para la consecución de objetivos que se pueden ir perfeccionando. Son las siguientes:

- Desarrollo iterativo que permita planificar desarrollos incrementales y entregas priorizando requisitos de modo que se entreguen antes las necesidades del usuario con mayor prioridad.
- Gestión de los requisitos: Documentar los requisitos y los cambios de los requisitos y analizar el impacto de cambios antes de aceptarlos.
- Emplear arquitecturas basadas en componentes para maximizar el aprovechamiento de desarrollos previos o componentes preconstruidos y abaratar los costes.
- Modelar visualmente el software empleando el estándar UML.
- Verificar la calidad de los productos del software asegurando que cumple los estándares de la compañía.

- Controlar los cambios del software.

Finalmente, cabe indicar que el modelo establece que el propio proceso es adaptable a cada caso de desarrollo. Por ejemplo, no todos los proyectos requieren del mismo nivel de documentación. El tamaño, la complejidad y el número de participantes entre otros aconsejan definir para el proyecto cómo adaptar el proceso. En la adaptación se definen qué artefactos hay que producir y en qué detalle, qué roles intervienen y las funciones que desempeñan dentro del proyecto, etc.

2.9.- Lenguaje Unificado de Modelado [12]

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el

Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

2.9.1.- Diagramas UML

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente, como se muestra en la figura 2.15.

2.9.1.1.- Diagrama de casos de uso

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: Actores, casos de uso y relaciones entre casos de uso.

- **Actores**: Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).
- **Casos de Uso:** Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el diagrama de casos de uso mediante una elipse con el nombre del caso de uso en su interior. Dicho nombre debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

- **Relaciones entre Casos de Uso:** Entre dos casos de uso puede haber las siguientes relaciones:

    1. **Extend**: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
    2. **Include**: Cuando un caso de uso utiliza a otro.

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extend>> o <<include>> según sea el tipo de relación. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

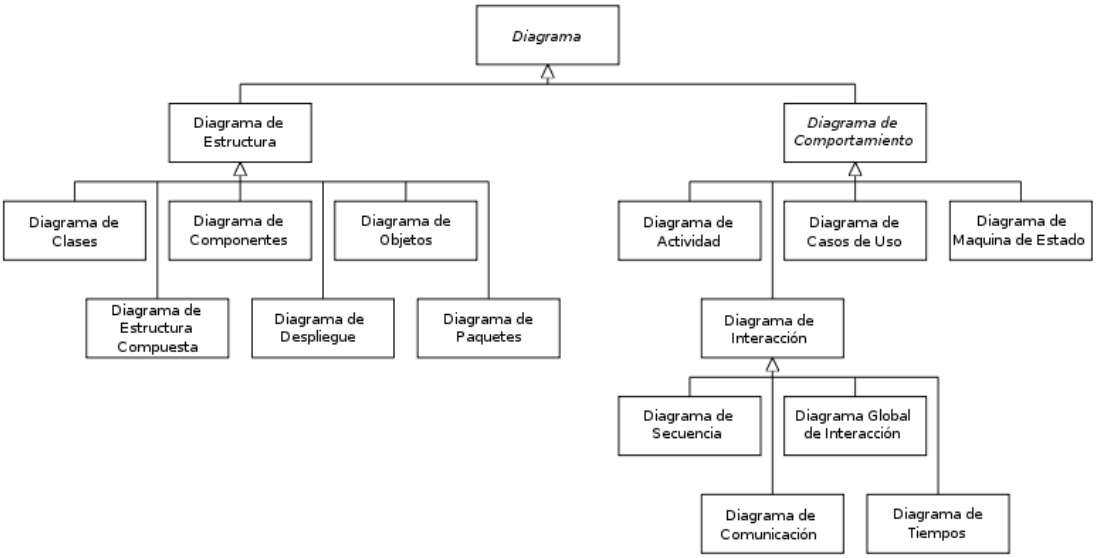

Figura 2.15. Diagrama taxonómico de estructura y comportamiento del UML 2.0

Fuente: Schmuller, J., (2001)

Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x)
- Diagrama de tiempos (UML 2.0)
- Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

2.9.1.2.- Diagrama de clases

Una clase es una descripción de un conjunto de objetos que comparten los mismos

atributos, operaciones, relaciones y semántica. Sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase**:** atributos, métodos y visibilidad.
- Relaciones**:** Herencia, Composición, Agregación, Asociación y Uso.

Elementos

Clase: Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.). En UML, una clase es representada por un rectángulo que posee tres divisiones, ver figura 2.16.

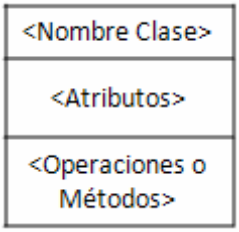

Figura 2.16. Estructura de clase

Fuente: UML y Patrones, Craig Larman

En donde:

- **Superior**: Contiene el nombre de la clase
- **Intermedio**: Contiene los atributos (o variables de instancia) que caracterizan a la clase (pueden ser private, protected o public).

- **Inferior**: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

Atributos: Los atributos o características de una clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

- **public** (+): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private** (-): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).
- **protected** (#): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

Métodos: Los métodos u operaciones de una clase son la forma en cómo ésta interactúa con su entorno, éstos pueden tener las características:

- **public** (+): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private** (-): Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).

- **protected** (#): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

2.9.1.3.- Diagramas de secuencia y de colaboración

Tanto los diagramas de secuencia como los diagramas de colaboración son un tipo de diagramas de interacción. Constan de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar unos objetos a otros. Cubren la vista dinámica del sistema. Los diagramas de secuencia enfatizan el ordenamiento temporal de los mensajes mientras que los diagramas de colaboración muestran la organización estructural de los objetos que envían y reciben mensajes. Los diagramas de secuencia se pueden convertir en diagramas de colaboración sin pérdida de información, lo mismo ocurren en sentido opuesto.
Están compuestas por tres elementos principales:

- **Clase Interfaz:** Modelan la interacción entre el sistema y los actores, representan la interfaz del sistema y describen la información presentada al actor y las peticiones que éste hace al sistema.

- **Clase Control:** Representan la coordinación entre objetos, encapsulan el flujo de control de un determinado caso de uso. No interaccionan con el usuario.

- **Clase Entidad:** Modelan la información de larga vida (persistencia). Pueden ser pasivas o activas y encapsulan información y operaciones asociadas.

Diagrama de secuencia

Un Diagrama de Secuencia muestra una interacción ordenada según la secuencia

temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

Diagrama de colaboración

Un Diagrama de Colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia.

En cuanto a la representación, un Diagrama de Colaboración muestra a una serie de objetos con los enlaces entre los mismos, y con los mensajes que se intercambian dichos objetos. Los mensajes son flechas que van junto al enlace por el que “circulan”, y con el nombre del mensaje y los parámetros (si los tiene) entre paréntesis. Cada mensaje lleva un número de secuencia que denota cuál es el mensaje que le precede, excepto el mensaje que inicia el diagrama, que no lleva número de secuencia.

2.9.1.4.- Diagrama de paquetes

Los paquetes ofrecen un mecanismo general para la organización de los

modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un sub-modelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general en UML. Pueden ser utilizados para el almacenamiento, el control de acceso, la gestión de la configuración y la construcción de bibliotecas que contengan fragmentos reutilizables del modelo.

Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones y colaboraciones; atributos, operaciones, estados, líneas de vida y mensajes están contenidos en otros elementos y no aparecen como contenido directo de los paquetes.

2.9.1.5.- Diagramas de componentes

Un diagrama de componentes muestra la organización y las dependencias entre un conjunto de componentes. Para todo sistema orientado a objetos se han de construir una serie de diagramas que modelan tanto la parte estática (diagrama de clases), como dinámica (diagramas de secuencia, colaboración, estados y de actividades), pero llegado el momento todo esto se debe materializar en un sistema implementado que utilizará partes ya implementadas de otros sistemas, todo esto es lo que se pretende modelar con los diagramas de componentes.

Un diagrama de componentes muestra un conjunto de componentes y sus relaciones de manera gráfica a través del uso de nodos y arcos entre estos. Normalmente los diagramas de componentes contienen:

- Componentes
- Interfaces
- Relaciones de dependencia, generalización, asociaciones y realización.

- Paquetes o subsistemas
- Instancias de algunas clases

2.9.1.6.- Diagramas de despliegue

Representan la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Muestran la vista de despliegue estática de una arquitectura y se relacionan con los componentes ya que, por lo común, los nodos contienen uno o más componentes.

2.9.1.7.- Elementos comunes a todos los diagramas

Asociaciones

Las asociaciones entre dos clases se representan mediante una línea que las une. La línea puede tener una serie de elementos gráficos que expresan características particulares de la asociación. A continuación se verán los más importantes de entre dichos elementos gráficos.

Nombre de la asociación y dirección

El nombre de la asociación es opcional y se muestra como un texto que está próximo a la línea. Se puede añadir un pequeño triángulo negro sólido que indique la dirección en la cual leer el nombre de la asociación. En el ejemplo de la figura 2.17 se puede leer la asociación como “Un jugador participa en un equipo”.

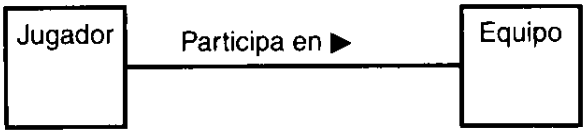

Figura 2.17. Ejemplo de asociación con nombre y dirección

Fuente: Schmuller, J. 1997

Los nombres de las asociaciones normalmente se incluyen en los modelos para aumentar la legibilidad. Sin embargo, en ocasiones pueden hacer demasiado abundante la información que se presenta, con el consiguiente riesgo de saturación. En ese caso se puede suprimir el nombre de las asociaciones consideradas como suficientemente conocidas.

Multiplicidad

La multiplicidad es una restricción que se pone a una asociación, que limita el número de instancias de una clase que pueden tener esa asociación con una instancia de la otra clase. Puede expresarse de las siguientes formas:

- Con un número fijo: 1.
- Con un intervalo de valores: 2..5.
- Con un rango en el cual uno de los extremos es un asterisco. Significa que es un intervalo abierto. Por ejemplo, 2..* significa 2 o más.
- Con una combinación de elementos como los anteriores separados por comas: 1, 3..5, 7,15..*.
- Con un asterisco: * .En este caso indica que puede tomar cualquier valor (0 o más).

Roles

Para indicar el papel que juega una clase en una asociación se puede especificar un nombre de rol. Se representa en el extremo de la asociación junto a la clase que lo desempeña.

Agregación

El símbolo de agregación es un diamante colocado en el extremo en el que está la clase que representa el "todo".

Clases Asociación

Cuando una asociación tiene propiedades propias se representa como una clase unida a la línea de la asociación por medio de una línea a trazos. Tanto la línea como el rectángulo de clase representan el mismo elemento conceptual: la asociación. Por tanto ambos tienen el mismo nombre, el de la asociación. Cuando la clase asociación sólo tiene atributos el nombre suele ponerse sobre la línea (como ocurre en el ejemplo de la figura 2.18). Por el contrario, cuando la clase asociación tiene alguna operación o asociación propia, entonces se pone el nombre en la clase asociación y se puede quitar de la línea.

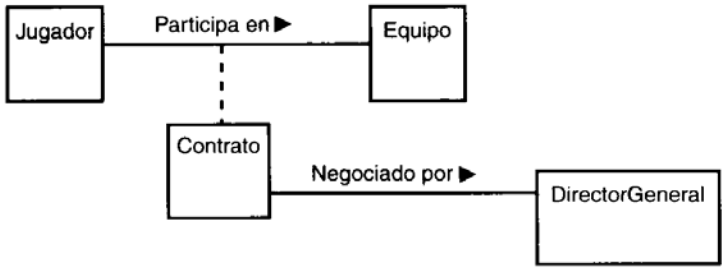

Figura 2.18. Ejemplo de clase asociación

Fuente: Schmuller, J. 1997

Asociaciones N-Arias

En el caso de una asociación en la que participan más de dos clases, las clases se unen con una línea a un diamante central. Si se muestra multiplicidad en un rol, representa el número potencial de tuplas de instancias en la asociación cuando el resto de los N-1 valores están fijos.

Navegabilidad

En un extremo de una asociación se puede indicar la navegabilidad mediante una flecha. Significa que es posible "navegar" desde el objeto de la clase origen hasta el objeto de la clase destino. Se trata de un concepto de diseño, que indica que un objeto de la clase origen conoce al objeto(s) de la clase destino, y por tanto puede llamar a alguna de sus operaciones.

Herencia

La relación de herencia se representa mediante un triángulo en el extremo de la relación que corresponde a la clase más general o clase “padre”.

La jerarquía de la herencia no tiene que finalizar en dos niveles, como se observa en la figura 2.19, una clase secundaria puede ser principal para otra clase secundaria, un Mamífero es una clase secundaria de Animal, y Caballo es una clase secundaria de Mamífero.

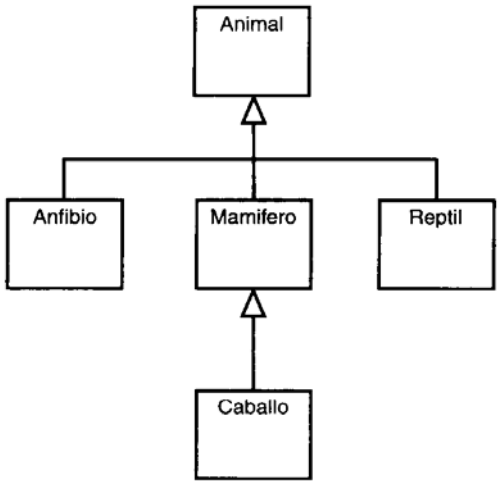

Figura 2.19. Una jerarquía de herencia en el reino animal

Fuente: Schmuller, J. 1997

2.10.- HTML

En 1989 existían dos técnicas que permitían vincular documentos electrónicos, por un lado los hipervínculos (links) y por otro lado un poderoso lenguaje de etiquetas denominado SGML.

Para entonces un usuario conocedor de ambas opciones, Tim Berners Lee físico nuclear del Centro Europeo de Investigaciones Nucleares da a conocer a la prensa que estaba trabajando en un sistema que permitirá acceder a ficheros en línea, funcionando sobre redes de computadoras o máquinas electrónicas basadas en el protocolo TCP/IP.

A principios de 1990, Tim BernersLee define por fin el HTML como un subconjunto del conocido SGML y crea algo más valioso aun, el World Wide Web.

En 1991, Tim Berners Lee crea el primer navegador de HTML que funcionaría en modo texto y para UNIX.

El HTML, acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender.

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de Notas de Windows (o Notepad), o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Wordpad, TextPad etc.

Existen además, otros programas para la realización de sitios Web o edición de código HTML, como por ejemplo Mozilla Composer, Nvu. Estos programas se les conoce como editores WYSIWYG o What You See Is What You Get (en español: “lo que ves es lo que obtienes”). Esto significa que son editores los cuales van mostrando el resultado de lo que se está editando en tiempo real a medida que se va desarrollando el documento. Ahora bien, esto no significa una manera distinta de realizar sitios web, sino que una forma un tanto más simple ya que estos programas, además de tener la opción de trabajar con la vista preliminar, tiene su propia sección HTML la cual va generando todo el código a medida que se va trabajando.

2.11.- PHP [4]

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios web. PHP es un acrónimo recurrente que significa "PHP Hypertext Preprocessor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. Últimamente también para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando la biblioteca GTK+.

El fácil uso y la similitud con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores experimentados crear aplicaciones complejas con una curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

Su interpretación y ejecución se da en el servidor, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de regresarlo al cliente.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de Aplicaciones web muy robustas.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como Linux), Windows y Mac OS X, y

puede interactuar con los servidores de web más populares ya que existe en versión CGI (Common Gateway Interface), módulo para Apache, e ISAPI (Internet Server Application Programming Interface).

2.12.- JavaScript [25]

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien ésta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a

funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Los autores inicialmente lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers' Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen a ambas versiones con frecuencia incompatibles.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento en castellano), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera versión 7, y Mozilla desde su primera versión.

2.13.- AJAX [26]

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el

servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

2.14.- Servidor HTTP Apache [24]

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto

de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

2.15.- PostgreSQL [27]

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

2.15.1.- Características

Algunas de sus principales características son, entre otras:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.

- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (*foreign keys*).
- Disparadores (*triggers*): Un disparador o *trigger* se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:
    - El nombre del disparador o *trigger*
    - El momento en que el disparador debe arrancar
    - El evento del disparador deberá activarse sobre...
    - La tabla donde el disparador se activará
    - La frecuencia de la ejecución
    - La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (*triggers*).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

CAPÍTULO 3
FASE DE INICIO

3.1.- Introducción

En esta fase se realizan los pasos necesarios que permiten establecer la viabilidad del sistema a desarrollar, por lo que la hace fundamental para el desarrollo del nuevo software, ya que se asegura de identificar los riesgos relacionados con el negocio y los requerimientos, así también, definir y acordar el alcance del proyecto y proponer una visión muy general de la arquitectura de software, para luego producir el plan de las fases y el de iteraciones.

La figura 3.1 muestra las fases Proceso Unificado, en donde se muestran las cargas de los flujos de trabajo.

Esta fase comprende la realización de actividades representadas en flujos de trabajo. El mayor flujo de trabajo de esta fase es el de requisitos, seguido por el de análisis donde se tratan los resultados del flujo anterior, encontrando muy poco trabajo que realizar en los flujos restantes que son diseño, implementación y pruebas. Puntualmente esta fase pretende identificar los actores y casos de usos, los requisitos funcionales y no funcionales, siendo seleccionados los casos de uso más relevantes para la arquitectura candidata, y que faciliten la comprensión de los riesgos críticos. Luego, mediante el análisis se lleva a cabo una refinación de los casos de uso, haciendo uso de diagramas de análisis y, de paquetes que permitirán satisfacer la ejecución de esta fase.

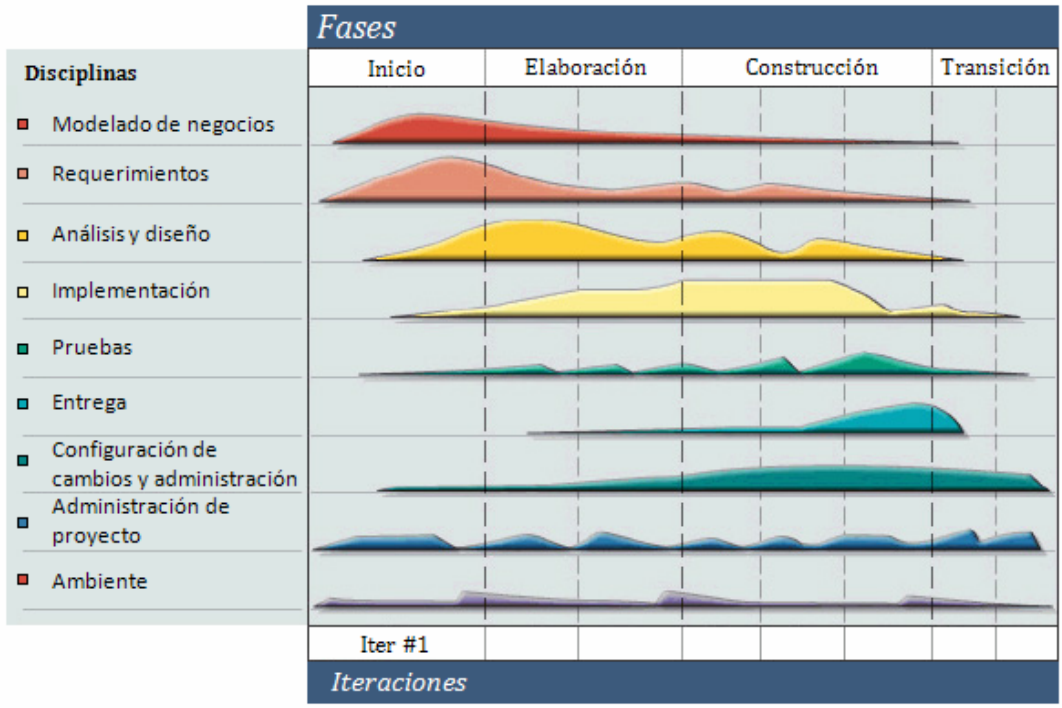

Figura 3.1. Fases del proceso unificado

Fuente: Jacobson, I, Booch, G, Rumbaugh J, 2000

3.2.- Visión general del sistema

El sistema MAI (Manejador de activos de instrumentación) tiene como objetivo ofrecer acceso vía remota a través de los protocolos TCP/IP y HART a la instrumentación de campo instalada en las plantas que operan en la refinería, de esta manera, proveer información sobre variables de proceso, estado y datos de calibración que poseen los transmisores. El sistema va orientado al monitoreo, diagnóstico y calibración sobre la instrumentación en línea.

3.3.- Captura de requisitos

Los requisitos son una descripción de las necesidades o deseos de un producto. La meta primaria de la fase de requisitos es identificar y documentar lo que en realidad se necesita. En una forma que claramente se le comunique al cliente y a los miembros del equipo de desarrollo. El reto es definirlos de manera inequívoca, de modo que se registren los riesgos y no se presenten conflictos al momento de la entrega del producto.

La disciplina de requisitos establece y mantiene un acuerdo entre el desarrollador y los usuarios sobre lo que el sistema debe hacer; asimismo, define los límites y proporciona bases para la estimación de costos y tiempos para el desarrollo del sistema. Esta disciplina también aporta lo necesario para planear el contenido técnico de las iteraciones y permite definir una interfaz de usuario, conduciendo el enfoque hacia las necesidades y metas de los usuarios finales.

Un aspecto importante de este proceso de requisitos es que expone las prácticas recomendadas para instaurar una buena comunicación con los usuarios y el resto de la organización destino del software. Los requisitos son clasificados en: funcionales y no funcionales, siendo los primeros aquellos que representan al sistema en términos de entradas y salidas, mientras que los no funcionales son aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo: facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

3.3.1.- Modelo de dominio

El modelo de dominio es un modelo conceptual, empleado para facilitar la comprensión del contexto del sistema de una manera intuitiva, y para la

representación de conceptos relacionados al dominio de una forma clara.

El UML entre sus muchas herramientas y diagramas de representación cuenta con una notación de estructura estática que explica de forma grafica los modelos conceptuales. Representando en si cosas del mundo real, y no componentes propios del software.

Como se observa en la figura 3.2 los empleados harán uso del sistema a través del computador personal que tengan en cada una de sus oficinas (estaciones de trabajo), estas deben estar conectadas a la intranet de la empresa. Podrán tener acceso a la instrumentación de campo, para monitoreo, calibración y diagnostico, por medio del servidor web que alojará al software a desarrollar. El software que se desarrollará tendrá como base un manejador de activos diseñado para comunicación serial, el cual será adaptado y mejorado para transmitir paquetes por medio de la intranet utilizando los protocolos TCP/IP, permitiéndole a los empleados acceder al sistema de forma remota. Un segundo servidor llamado servidor TCP/IP-HART, prestará servicios al nuevo manejador de activos (software a desarrollar). Este servidor recibirá los paquetes TCP/IP provenientes del manejador de activos que tendrán como mensaje encapsulado una trama en protocolo HART e información del instrumento destino, esto ocurrirá producto de una solicitud de algún empleado. El servidor TCP/IP-HART procesará el mensaje encapsulado para enrutar la trama HART al instrumento adecuado a través de la red HART. Una vez que el instrumento recibe la trama HART realiza la operación que se solicita en dicha trama y genera una respuesta de acuerdo a la solicitud, esta respuesta debe llegar al usuario quien ejecutó dicha operación.

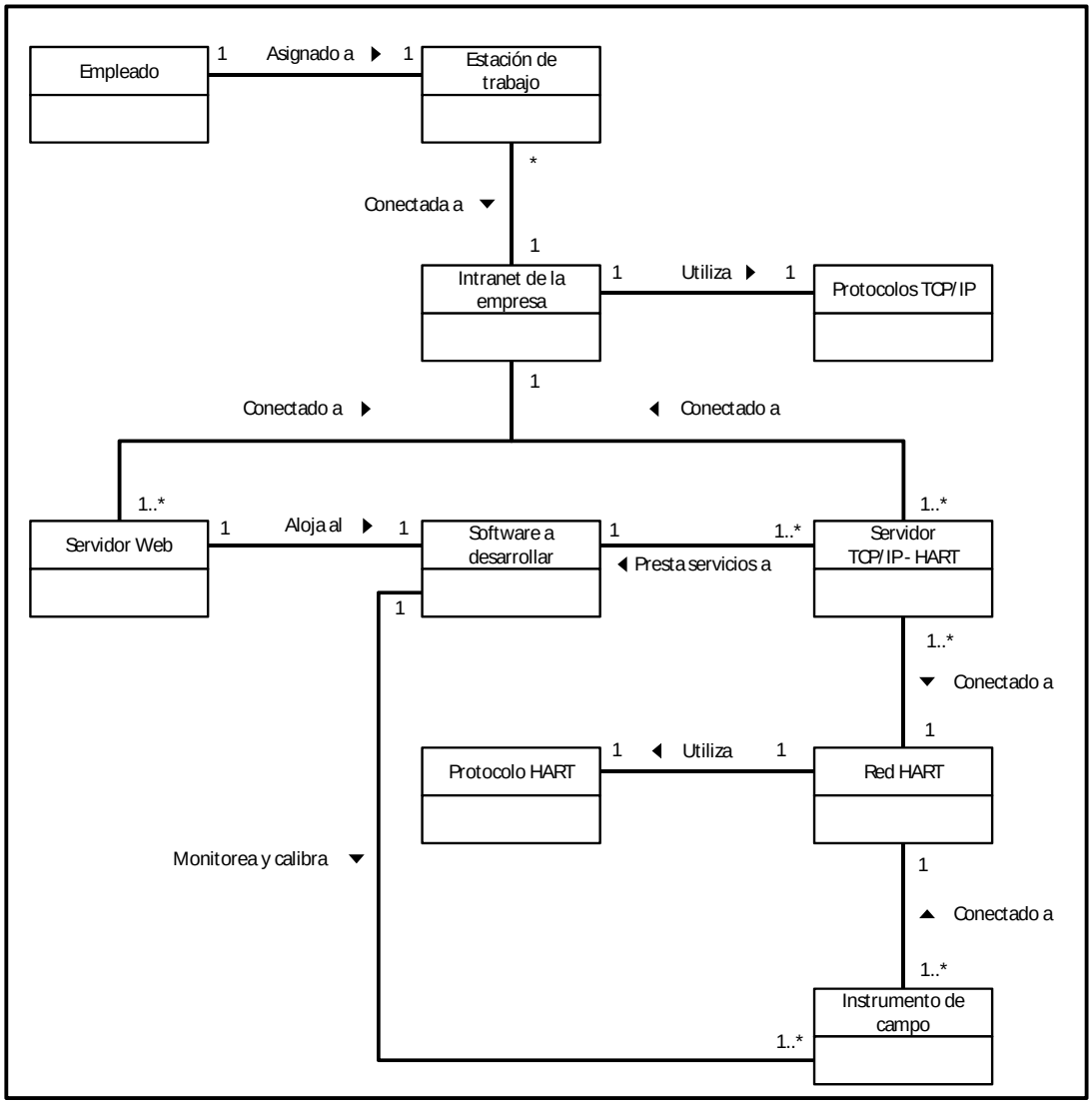

Figura 3.2. Modelo de dominio

Fuente: Elaboración propia

3.3.2.- Glosario de términos

En las tablas 3.1 y 3.2 se describen cada una de las clases que conforman el modelo de dominio.

Tabla 3.1. Glosario de términos 1/2

Fuente: Elaboración propia

Clase de dominio	Descripción
Servidor Web	Es un programa que implementa el protocolo HTTP (*HyperText Transfer Protocol*) y se ejecuta continuamente en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.
Servidor TCP/IP-HART	Es un programa que presta servicios de multiplexión al sistema MAI, que conmuta las tramas HART al puerto donde está conectado el instrumento de campo.
Protocolos TCP/IP	TCP/IP es una combinación de dos protocolos individuales. IP por sus siglas en ingles (*Internet Protocol),* opera en la Capa de red (capa 3) y es un servicio no orientado a conexión que proporciona una entrega de máximo esfuerzo a través de una red. TCP por sus siglas en ingles (*Transmission Control Protocol*), opera en la capa de transporte (capa 4), y es un servicio orientado a conexión que suministra control de flujo y confiabilidad.
Protocolo HART	Protocolo de comunicación industrial utilizado en instrumentos de campo. Este protocolo superpone señales digitales sobre análogas sin distorsionar la señal analógica.

Tabla 3.2. Glosario de términos 2/2

Fuente: Elaboración propia

Clase de dominio	Descripción
Intranet de la Empresa	Conjunto de computadoras y/o dispositivos conectados por enlaces de un medio físico ó inalámbrico que comparten información, recursos y servicios.
Red HART	Conexión de enlaces de medio físico de dos hilos que trasmite tramas en protocolo HART utilizando el estándar de comunicación Bell 202.
Estación de trabajo	Computadoras donde se ejecuta el sistema MAI.
Empleados	Conjunto de personas que prestan servicio en la empresa, y están registrados como usuarios del sistema.
Instrumento de campo	Equipo inteligente utilizado para monitorear variables de procesos.

3.3.3.- Requisitos funcionales

El conjunto de requisitos funcionales especifican las acciones que debe ser capaz de realizar el sistema sin considerar las restricciones físicas, así también, el comportamiento de Entrada/Salida del sistema. A continuación se describen los requisitos funcionales capturados durante la ejecución de esta fase:

- Los usuarios podrán tener acceso al sistema de manera remota a través de los navegadores web en cualquier estación de trabajo dentro de la intranet de la empresa.
- El sistema debe contar con mecanismos de seguridad que permita el acceso solo a los usuarios registrados.

- Para garantizar la integridad en el acceso al sistema las sesiones abiertas de los usuarios deberán tener un tiempo de expiración finito que evite que por olvido de los usuarios personas no autorizadas tengan acceso al sistema. Igualmente se debe contar con una opción que permita cerrar sesión de modo seguro.
- El sistema debe permitir a los usuarios cambiar su contraseña en el momento que así lo deseasen, sin embargo, no podrán realizar cambios en el usuario por ser funciones del administrador.
- Para realizar las operaciones de instrumentación en línea, el sistema debe presentar el listado de todos los instrumentos activos disponibles para realizar operaciones remotas, así también, debe contar con herramientas de búsqueda y de filtrado de los tags o etiquetas de los instrumentos según planta y función para facilitar la ubicación de los instrumentos.
- Las operaciones de calibración en línea solo la podrán llevar a cabo los usuarios que posean el estatus de instrumentistas.
- Cualquier usuario podrá realizar operaciones de consulta a la instrumentación en línea, así también, podrá tener acceso a toda la información técnica de los instrumentos. Todos los accesos de estos usuarios serán solo de consulta sin implicar ningún cambio en la información almacenada.
- El administrador del sistema será el encargado de gestionar todas las operaciones concernientes a los usuarios y privilegios que estos tengan sobre el sistema, también administrará toda la información técnica de la instrumentación referente a datos suministrados por los fabricante e información de operación en planta de los equipos. El administrador también se encargará del mantenimiento y configuración del sistema para hacer posible el tener acceso a los instrumentos de campo conectados a través de los servidores TCP/IP-HART.

- El sistema debe contar con mensajes emergentes que describan de forma clara el posible error que pueda ocurrir en cualquier caso.
- El sistema debe incluir en las operaciones de calibración en línea mensajes indicativos de los procedimientos que se deben seguir durante la calibración para realizar la operación de modo seguro.
- El sistema debe ser capaz de llevar un historial de todas las operaciones de calibración en la instrumentación hechas a través del sistema. Este historial llevará detalles tales como: tag del instrumento, el instrumentista que realizó la operación, la operación que se realizó, fecha, hora y los datos o valores utilizados en la calibración.
- Los usuarios con privilegios de administrador serán los únicos que podrán tener acceso a los historiales de calibración, pero por ser datos muy importantes no podrá realizar cambios en estos ni eliminarlos.

3.3.3.1.- Identificación de riesgos

El desarrollo de un proyecto lleva consigo una serie de variables que ponen en duda la finalización del mismo, conocidos como riesgos. En la fase de inicio se hace importante la identificación de estos, por su influencia en la planificación del desarrollo del sistema.

Existen muchos riesgos que se pueden presentar en el desarrollo de un sistema, estos van desde riesgos de poca importancia hasta los riesgos que podrían no permitir el desarrollo del proyecto, estos últimos son los llamados riesgos críticos, que deben ser tratados en la fase inicial del proyecto. A continuación se presentan la lista de riesgos que se han considerado como significativos en el sistema:

- El proyecto requiere de la utilización de un lenguaje de programación web para desarrollar la interfaz de usuario que permitirá a estos interactuar con el

sistema a través de los navegadores web, se ha considerado usar el leguaje PHP por ser de licencia pública y por demostrar un alto desempeño para la creación de páginas dinámicas. Por otro lado las operaciones de comunicación con la instrumentación requiere de un lenguaje que permita realizar operaciones a bajo nivel, manteniendo el paradigma orientado a objetos, el manejador de activos de instrumentación diseñado en un proyecto anterior utilizó el leguaje C++, por lo tanto, es necesario investigar los modos de integración de los lenguajes PHP y C++, para garantizar la comunicación inequívoca entre ambos lenguajes.

- Existe el riesgo de presentar problemas para la transmisión de datos del sistema a los instrumentos a través de la intranet, por lo que es necesario estudiar el manejo de sockets TCP/IP en C++.
- Un factor de riesgo son los errores que se puedan presentar por registro de datos inválidos, para solucionar estos riegos es necesario validar los datos de entrada, tratar las excepciones con mensajes de error y programar soluciones a las mismas que mantengan la integridad del sistema.
- Cuando una operación que requiere conectarse a un instrumento no logra hacerlo se pueden presentar problemas en el sistema, es importante validar la conexión al servidor de interfaz TCP/IP-HART, mostrar mensajes de error si no se pudo conectar al servidor o el instrumento no se encuentra conectado al puerto indicado.
- La operaciones cliente servidor son muy importantes en el sistema, por esto el servidor siempre debe estar listo para atender las solicitudes, sin embargo, se puede saturar el servidor de solicitudes innecesarias, las transacciones innecesarias entre navegador y servidor web deben ser reducidas al máximo. Una da las causas más comunes son envíos de formularios con datos inválidos. La herramienta que puede solventar esta situación es el leguaje JavaScript que permite validar los datos de envío del lado del cliente y así

solo enviar solicitudes al servidor cuando los datos sean validos.

- Otra de las causas que pueden hacer que la navegación en una página sea lenta es hacer solicitudes al servidor que conlleve a refrescar la pagina completamente. Este tipo de práctica genera transacciones más pesadas e innecesarias al servidor, por esto se considera utilizar la herramienta AJAX *(*Asynchronous JavaScript And XML) para realizar transacciones en segundo plano con el servidor sin afectar la funcionalidad de la página y realizar cambios en pequeños sectores de la pagina sin tener que ser recargada nuevamente.

3.3.3.2.- Identificación de los actores

Los actores se definen como personas, sistemas o hardware externos que interactúan con el sistema. Al interactuar con el sistema estos pueden hacer uso de las funcionalidades provistas él o pueden también proporcionarle funcionalidades. A continuación se muestra la descripción de los actores identificados en esta etapa:

- **Usuario:** Es un actor general, constituye a todos aquellos usuarios que necesitan realizar consultas a la información que maneja el sistema y operaciones de monitoreo a la instrumentación en línea. No tiene ningún privilegio de escritura o modificación de los datos. Sus funciones son básicas, comunes entre los diferentes usuarios, por lo que este actor representa la generalización de los demás usuarios.

- **Administrador:** Su principal función es velar por el buen funcionamiento del sistema. El administrador tiene acceso total al sistema, sobre todo a la parte de seguridad y control. Puede cambiar los permisos de usuario, crear o modificar cuentas de usuario, incluir nueva información técnica sobre instrumentación, realizar respaldos a la base de datos y configurar los parámetros de comunicación con la instrumentación en línea.

- **Instrumentista:** Se encarga de realizar las operaciones de calibración sobre la instrumentación de campo. Este actor es el único que tiene privilegios para realizar operaciones de escritura en la configuración de los instrumentos.

3.3.3.3.- Diagrama general de caso de uso

El sistema en general está representado en el diagrama de casos de uso Figura 3.3, que muestra los diferentes actores que interactúan con el sistema y los casos de uso que lo componen, así también, la manera en cómo estos se relacionan entre sí.

Estos casos de uso que componen el sistema corresponden en sí, a las funcionalidades que este posee, estas funcionalidades son las que en conjunto definen las características de MAI.

Casos de uso detallados

En la tabla 3.3 se describen los casos de uso que componen al sistema y los actores que intervienen en cada uno de ellos.

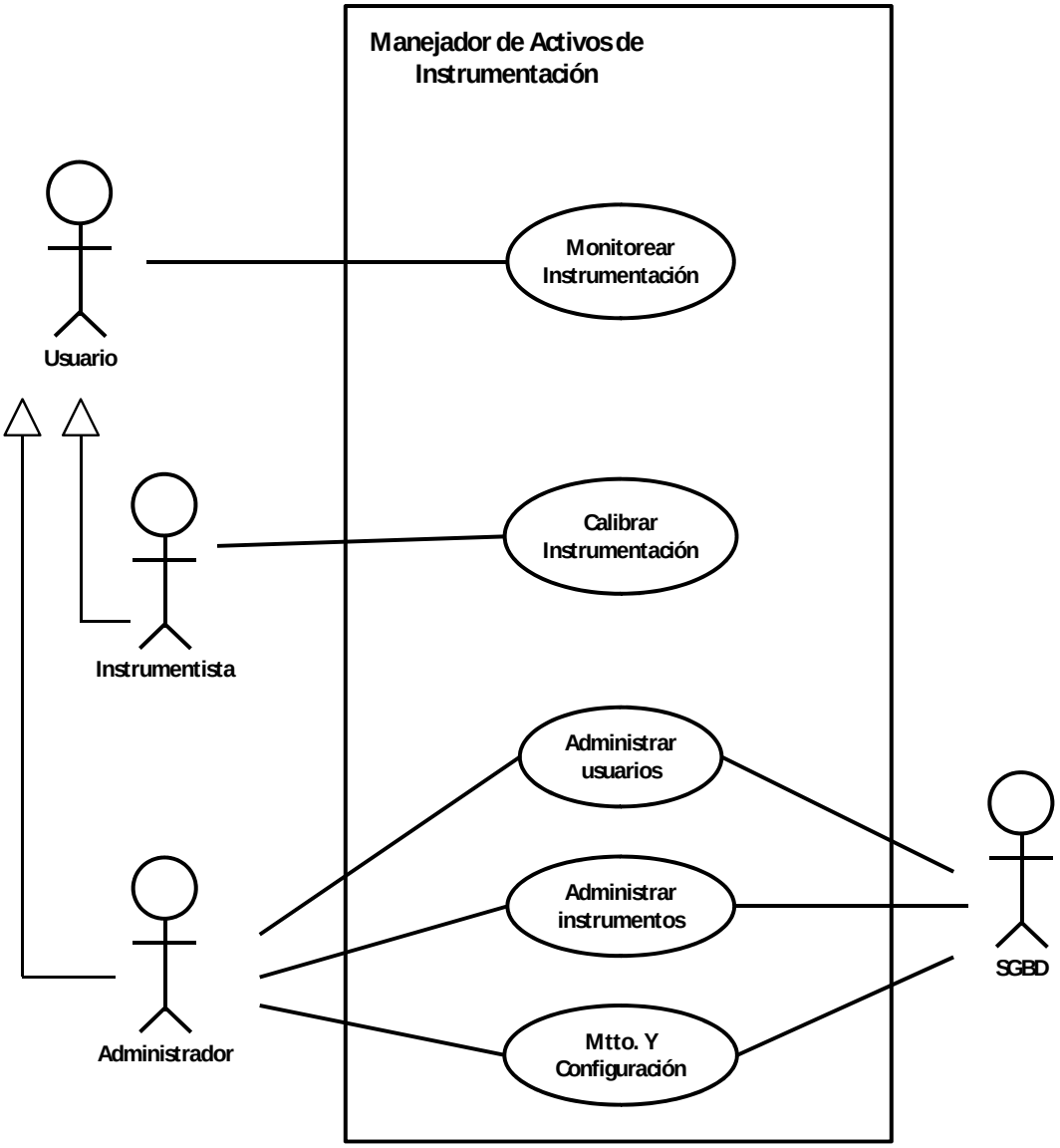

Figura 3.3. Diagrama de casos de uso

Fuente: Elaboración propia

Tabla 3.3. Casos de uso detallados

Fuente: Elaboración propia

Caso de uso	Descripción	Actor
Monitorear instrumentación	Realiza todas las operaciones de monitoreo y consulta sobre la instrumentación de campo.	Usuario
Calibrar instrumentación	Este caso de uso reúne todas las operaciones elementales para la calibración de la instrumentación de campo, para indicar rangos de operación, unidad de ingeniería de la variable de proceso, Etiqueta de identificación del dispositivo, entre otros.	Instrumentista
Administrar usuarios	En él, se realizan las operaciones correspondientes con la administración de usuarios, los privilegios de estos y el historial de las operaciones de calibración que realizan.	Administrador
Administrar instrumentos	Destinado para la gestión de administración de los instrumentos en la base de datos con información detallada de datos de calibración, información de fabricante, servicio que presta en planta, entre otros.	Administrador
Mantenimiento y configuración	En este caso se realiza la administración de las rutas lógicas y servidores TCP/IP-HART que permitirán la comunicación entre los usuarios e instrumentos a través de la red.	Administrador

3.3.4.- Requisitos no funcionales

A continuación se describen los requisitos no funcionales identificados en esta fase:

- El sistema debe contar con una interfaz gráfica amigable y bien estructurada, de fácil comprensión y navegabilidad.
- Cualquier instrumento dotado con la capacidad de comunicación HART podrá ser usado sin importar el fabricante del mismo.
- El sistema deberá operar bajo la plataforma Linux.
- El sistema debe ser capaz de adecuarse a nuevas funcionalidades (acordes al objetivo del software) que se deseen agregar en un futuro.

3.4.- Análisis

El análisis pretende modelar el sistema bajo condiciones ideales, garantizando que la arquitectura de software resulte suficientemente robusta y extensible para servir de base a la estructura lógica de la aplicación. En el análisis se elaboran un conjunto de modelos iníciales del sistema que capturen la semántica y comportamiento.

El objetivo del modelo de análisis es para aclarar aspectos no abordados en la etapa de recopilación de requisitos, sin embargo, no pretende profundizar tanto en detalles, como si se realizará en otras fases.

3.4.1.- Diagrama de clases de análisis

Este diagrama representa una abstracción de lo que serán una o varias clases en diseño, centrándose en los requisitos funcionales. A continuación se muestran los diagramas de clase de análisis del sistema:

3.4.1.1.- Diagrama de clases de análisis del caso de uso Administrar usuarios

Uno de los requisitos funcionales y de gran importancia es restringir el acceso al sistema para evitar que cualquier individuo no autorizado pueda ingresar y obtener información delicada sobre los datos de procesos de la refinería y, en el peor caso que sea capaz de cambiar parámetros de configuración en la instrumentación de campo instalada a lo largo de las plantas que conforman a la Refinería Puerto La Cruz. La figura 3.4 representa el diagrama de clases de análisis para la administración de usuarios. Esto permitirá llevar un control de los usuarios autorizados para tener acceso al sistema. Cada usuario tendrá asociado un estatus que hará referencia a los privilegios que posee sobre el sistema.

También el diagrama describe una clase historial que se plantea para llevar un registro de todos los cambios que se realicen en la configuración de los instrumentos, estos registros contendrán el cambio realizado y el usuario que lo ha ejecutado añadiéndole otros datos de información como fecha, hora, etc.

Todas estas operaciones de administración de usuarios lo llevará a cabo el actor definido como administrador y todos estos registros estarán almacenados en la base de datos del sistema.

3.4.1.2.- Diagrama de clases de análisis del caso de uso Mtto. y configuración

Para que el sistema sea capaz de alcanzar a través de la red a los instrumentos instalados en campo, estos deben estar conectados a servidores que funcionarán como multiplexores TCP/IP-HART, estos servidores a su vez deberán estar conectados a la intranet de la empresa.

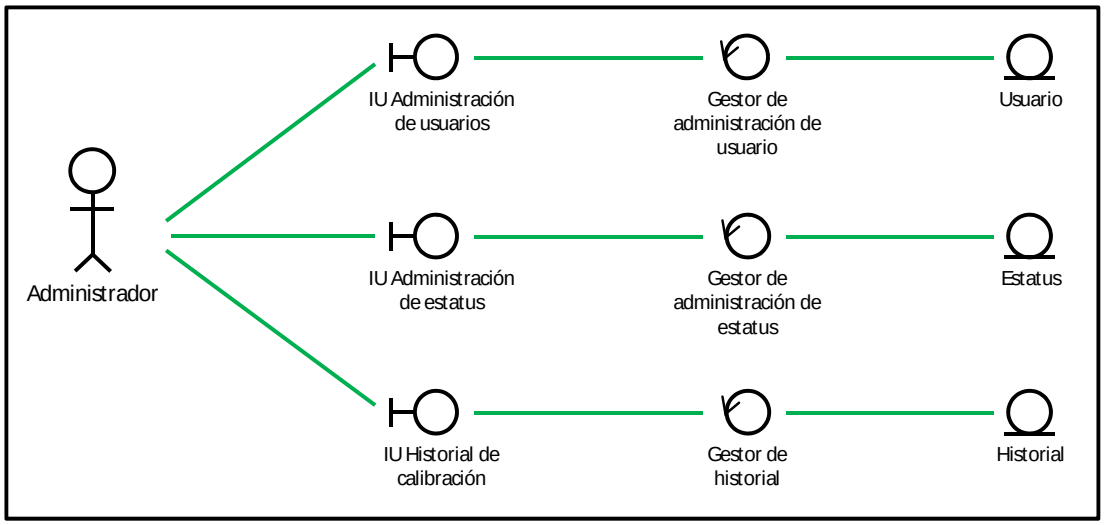

Figura 3.4. Diagrama de clases de análisis del caso de uso Administrar usuarios

Fuente: Elaboración propia

En el sistema se deben registrar los servidores de multiplexión con la dirección IP que le fue asignada a cada uno de estos, la información del servidor con su dirección lógica estará almacenada en la base de datos del sistema. En la figura 3.5 se ilustra el diagrama de clases de análisis para el caso de uso mantenimiento y configuración.

Cuando se ejecute el escaneo de la red el sistema se conectará a cada uno de los servidores de multiplexión registrados previamente y se consultará en cada uno de sus puertos si existe un instrumento conectado. A los instrumentos que respondan se les identificará con su tag y la ruta para lograr el acceso a ellos, la ruta de un instrumento será la dirección IP del servidor TCP/IP-HART y el puerto serial de este último al que está conectado el instrumento. Toda esta información de rutas de los instrumentos dados como respuesta al escaneo de la red estará registrada en la base de datos y se refrescará cada vez que se realice un escaneo.

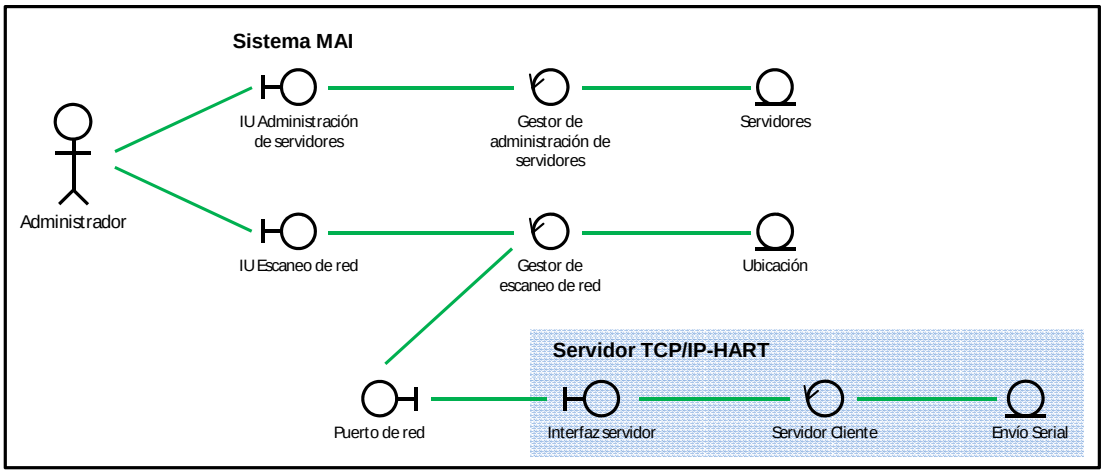

Figura 3.5. Diagrama de clases de análisis del caso de uso Mtto. Y configuración

Fuente: Elaboración propia

3.4.1.3.- Diagrama de clases de análisis del caso de uso Monitorear instrumentación

Una vez que se ha hecho un primer escaneo de la red, se conocen los instrumentos conectados al sistema y sus rutas de acceso. A partir de esto se podrá seleccionar un instrumento y realizar cualquier operación que este dentro de los privilegios de cada usuario.

La figura 3.6 representa el diagrama de clases de análisis para el caso de uso monitorear instrumentación, en este caso de uso se realizan operaciones de consulta en línea sobre los instrumentos. Un usuario podrá realizar una operación de consulta a través de la interfaz de usuario específica para tal operación y así obtener información sobre los instrumentos.

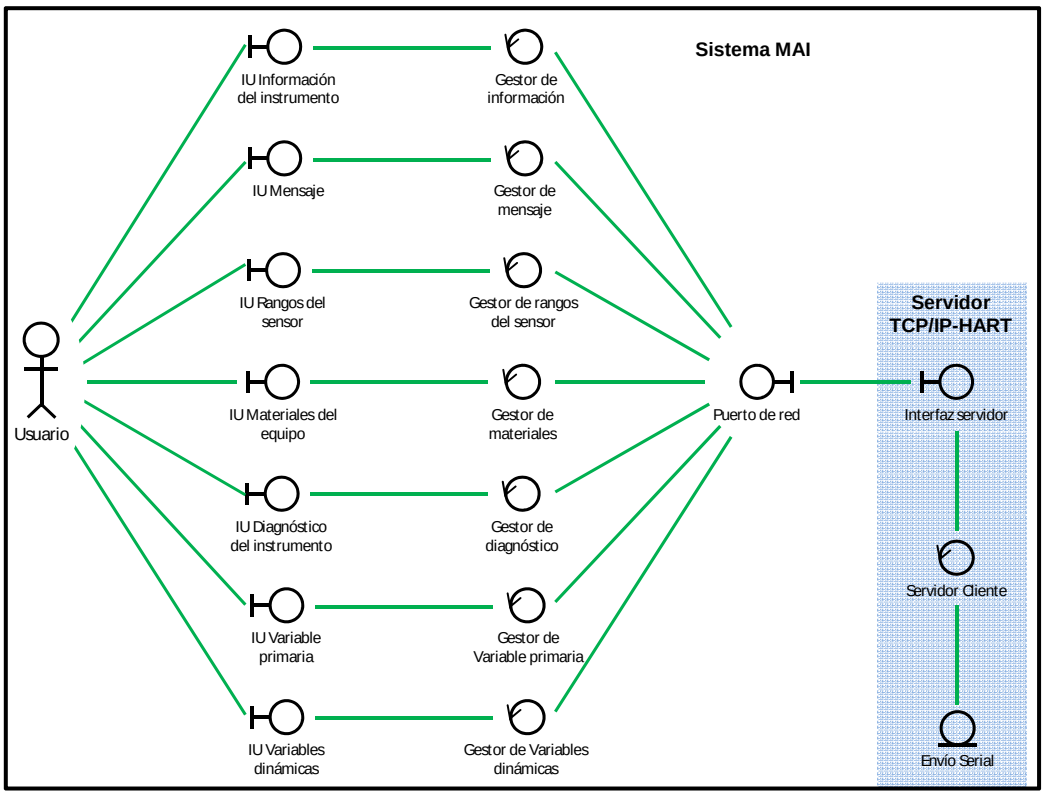

Figura 3.6. Diagrama de clases de análisis del caso de uso Monitorear instrumentación

Fuente: Elaboración propia

3.4.1.4.- Diagrama de clases de análisis del caso de uso Calibrar instrumentación

Las operaciones de calibración de manera remota que se plantean ejecutar a través del Sistema MAI, son una de las funcionalidades más resaltantes de este sistema. La figura 3.7 describe el diagrama de clases de análisis para el caso de uso calibrar instrumentación, representando todas las clases como conjunto de operaciones que se podrán ejecutar para lograr la calibración.

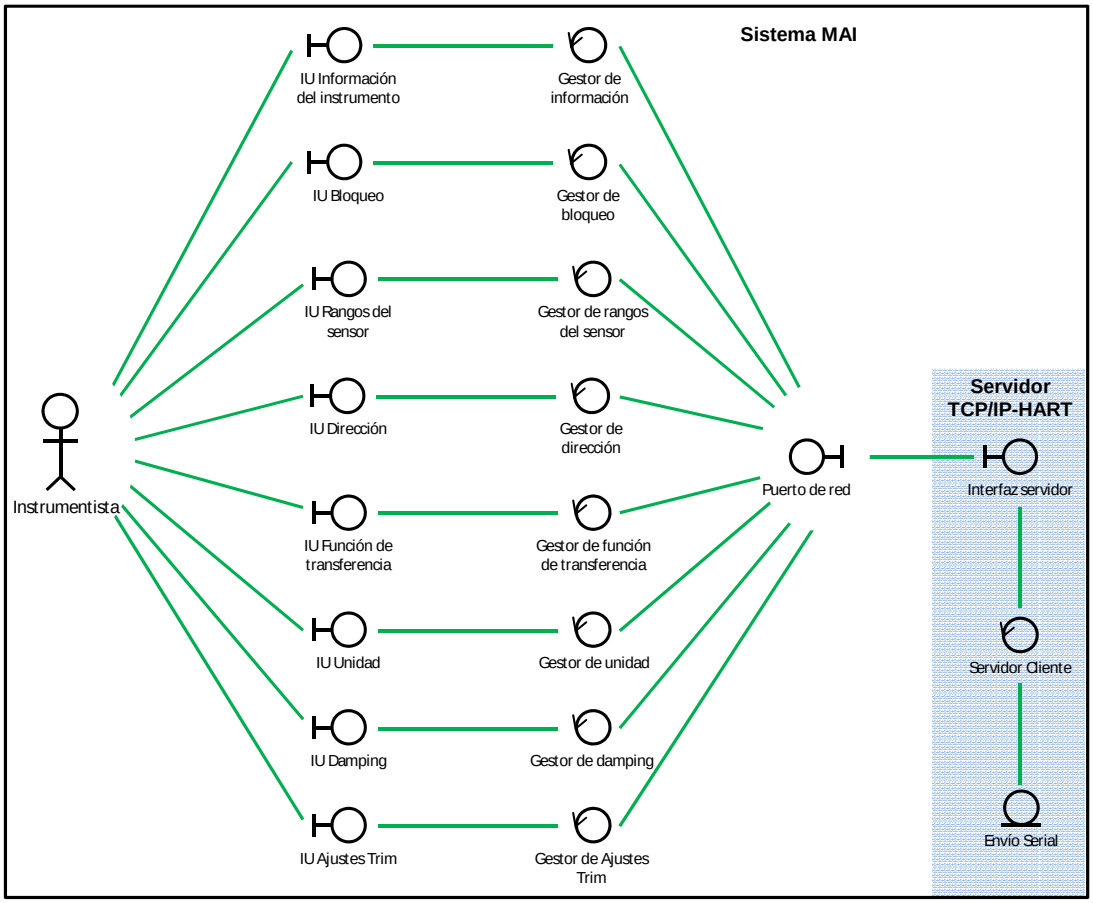

Figura 3.7. Diagrama de clases de análisis del caso de uso calibrar instrumentación

Fuente: Elaboración propia

3.4.1.5.- Diagrama de clases de análisis del caso de uso Administrar instrumentos

Como ya se ha podido observar el Sistema Manejador de Activos de Instrumentación tendrá conexión en línea con la diversidad de instrumentos conectados al sistema que operan en las plantas de la refinería, sin embargo, no es la única manera de obtener información de los instrumentos. La figura 3.8 representa el diagrama de clases de análisis administrar instrumentos, lleva registro de todos los datos inherentes a la instrumentación, información única para cada instrumento, como también, información compartida. Este subsistema de administración lleva registro de las

plantas donde están instalados los instrumentos, los sectores de esta donde se encuentran prestando servicio cada uno de los instrumentos, datos del fabricante como principios de operatividad, alimentación, etc. Toda esta información que es administrada únicamente por el propio administrador del sistema podrá también ser consultada por cualquier usuario.

3.4.2.- Diagrama de colaboraciones

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Un objeto de interfaz recibirá este mensaje del actor y lo enviará a algún otro objeto. De esta forma los objetos implicados interactuarán para llevar a cabo el caso de uso. El análisis de este proceso se representa con diagramas de colaboración ya que el objetivo fundamental es identificar requisitos y responsabilidades sobre los objetos, y no secuencias de interacción detalladas u ordenadas cronológicamente.

A continuación se ilustran los diagramas de colaboración para los casos de uso identificados en esta fase. En los mismos se muestran las interacciones que existen entre los distintos objetos que intervienen en cada caso de uso.

3.4.2.1.- Diagrama de colaboración del caso de uso Administrar usuarios

En la figura 3.9 se ilustra el diagrama de colaboración del caso de uso administrar usuarios, en la tabla 3.4 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

El administrador es el actor quien inicia este caso de uso al activar la interfaz de usuario de acuerdo a la tarea que desee realizar.

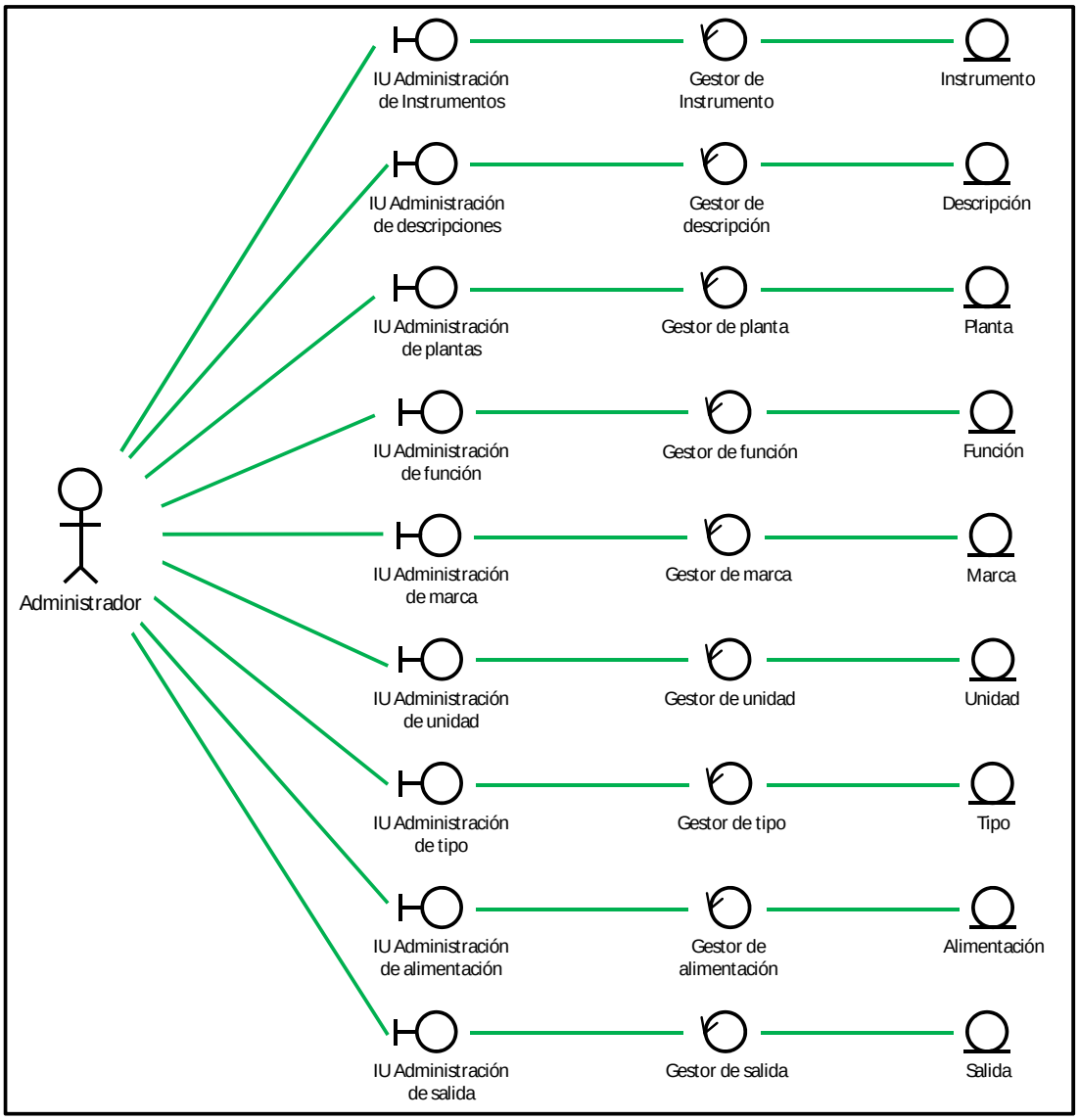

Figura 3.8. Diagrama de clases de análisis del caso de uso Administrar instrumentos

Fuente: Elaboración propia

A través de la IU de administración de usuarios el administrador podrá agregar un nuevo usuario (1), modificar uno ya existente (4) o eliminar un usuario, algunas de estas tres operaciones conllevará a enviar el mensaje (2), (5) u (8) para que el gestor de administración de usuarios valide los datos y así, una vez validados los datos se realizará la operación que corresponda (de acuerdo a la selección del actor al inicio del caso de uso) sobre el registro como indican los mensajes (3, 6 y 9). De manera similar ocurre en la administración de estatus que corresponde a las instancias de las clases IU administración de estatus, gestor de administración de estatus y estatus, con el conjunto de mensajes en el rango 10-18, de esta forma se podrán agregar, modificar o eliminar registros de estatus de usuarios.

Otra de las posibilidades para este caso de uso es la consulta del historial de calibración (19) en la que se podrá ver que usuarios realizaron cuando y cuales operaciones de calibración, al consultar el historial se procede a cargar el historial (20) a partir de los datos obtenidos de la base de datos (21).

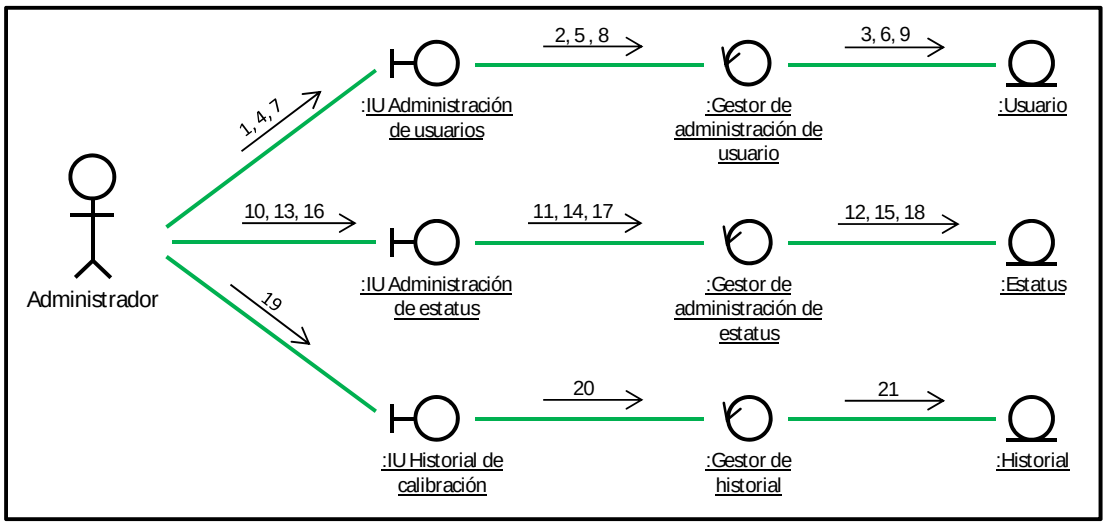

Figura 3.9. Diagrama de colaboración del caso de uso Administrar usuarios

Fuente: Elaboración propia

Tabla 3.4. Lista de mensajes del diagrama de colaboración Administrar usuarios

Fuente: Elaboración propia

Leyenda		
1. Agregar usuario	*8. Validar datos de usuario*	*15. Modificar registro*
2. Validar datos de usuario	*9. Eliminar registro*	*16. Eliminar estatus*
3. Registrar usuario	*10. Agregar estatus*	*17. Validar estatus*
4. Modificar usuario	*11. Validar estatus*	*18. Eliminar registro*
5. Validar datos de usuario	*12. Registrar estatus*	*19. Consultar historial de calibración*
6. Modificar registro	*13. Modificar estatus*	*20. Cargar historial de calibración*
7. Eliminar usuario	*14. Validar estatus*	*21. Obtener datos*

3.4.2.2.- Diagrama de colaboración del caso de uso Mtto. Y configuración

En la figura 3.10 se ilustra el diagrama de colaboración del caso de uso mantenimiento y configuración, en la tabla 3.5 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

El administrador gestiona los servidores TCP/IP-HART que prestan servicios al sistema. Este actor puede realizar a través de la IU de administración de servidores operaciones de agregar, modificar y eliminar servidores descrito en el diagrama con los mensajes (1, 4 y 7) al activar alguna de estas operaciones se envía el mensaje (2, 5

u 8) según sea el caso para que el gestor de administración de servidores valide los datos antes de realizar la operación sobre los registros (3, 6, o 9).

Cuando el administrador a través de la IU escaneo de red solicita escanear la red (10) el gestor de escaneo de red inicia el escaneo (11) para buscar los instrumentos que se encuentran activos y conectados a los servidores, a partir de este punto el escaneo se realiza de manera automática mientras el administrador espera la culminación del escaneo. Primero se procede a limpiar los registros (12) de las rutas a los instrumentos producto de un escaneo anterior, luego el gestor de escaneo genera la trama HART con la dirección física y lógica del instrumento (13) y es enviada al servidor TCP/IP-HART a través del puerto de red (16), el servidor recibe la trama (17) y la retransmite al instrumento por el puerto serial al que se está consultando.

Esta operación se realiza a cada uno de los puertos que componen a los diversos servidores TCP/IP-HART registrados en el sistema, consultando si existe instrumento conectado, de ser así, el instrumento responderá identificándose y así se constituirá la ruta de un instrumento (21) sabiendo al servidor y puerto de este último al que está conectado, de esta manera registrar esta información (22) como datos de ubicación de un instrumento.

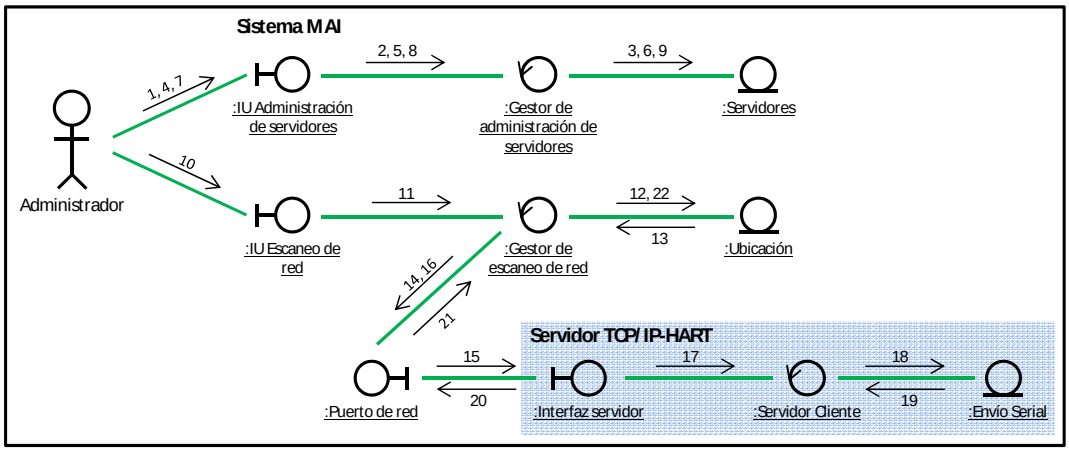

Figura 3.10. Diagrama de colaboración del caso de uso Mtto. Y configuración

Fuente: Elaboración propia

Tabla 3.5. Lista de mensajes del diagrama de colaboración Mtto. Y configuración

Fuente: Elaboración propia

Leyenda		
1. Agregar servidor	*9. Eliminar registro*	*17. Recibir Trama HART*
2. Validar datos de servidor	*10. Escanear red*	*18. Transmitir trama*
3. Registrar servidor	*11. Iniciar escaneo de red*	*19. Enviar trama respuesta*
4. Modificar servidor	*12. Limpiar registros*	*20. Recibir trama de respuesta*
5. Validar datos de servidor	*13. Construir trama de búsqueda*	*21. Interpretar trama*
6. Modificar registro	*14. Conectar a servidor*	*22. Registrar ubicación*
7. Eliminar servidor	*15. Admitir conexión*	
8. Validar datos de servidor	*16. Encapsular trama y enviar*	

3.4.2.3.- Diagrama de colaboración del caso de uso Monitorear instrumentación

En la figura 3.11 se ilustra el diagrama de colaboración del caso de uso monitorear instrumentación, en la tabla 3.6 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

Una vez que se ha hecho un primer escaneo de la red, se conocen los instrumentos conectados al sistema y sus rutas de acceso. A partir de esto se podrá seleccionar un instrumento y realizar cualquier operación que esté dentro de los privilegios de cada usuario.

En este caso de uso se realizan operaciones de consulta en línea sobre los instrumentos. Un usuario podrá realizar una operación de consulta a través de la interfaz de usuario específica para tal operación y así obtener información sobre los instrumentos.

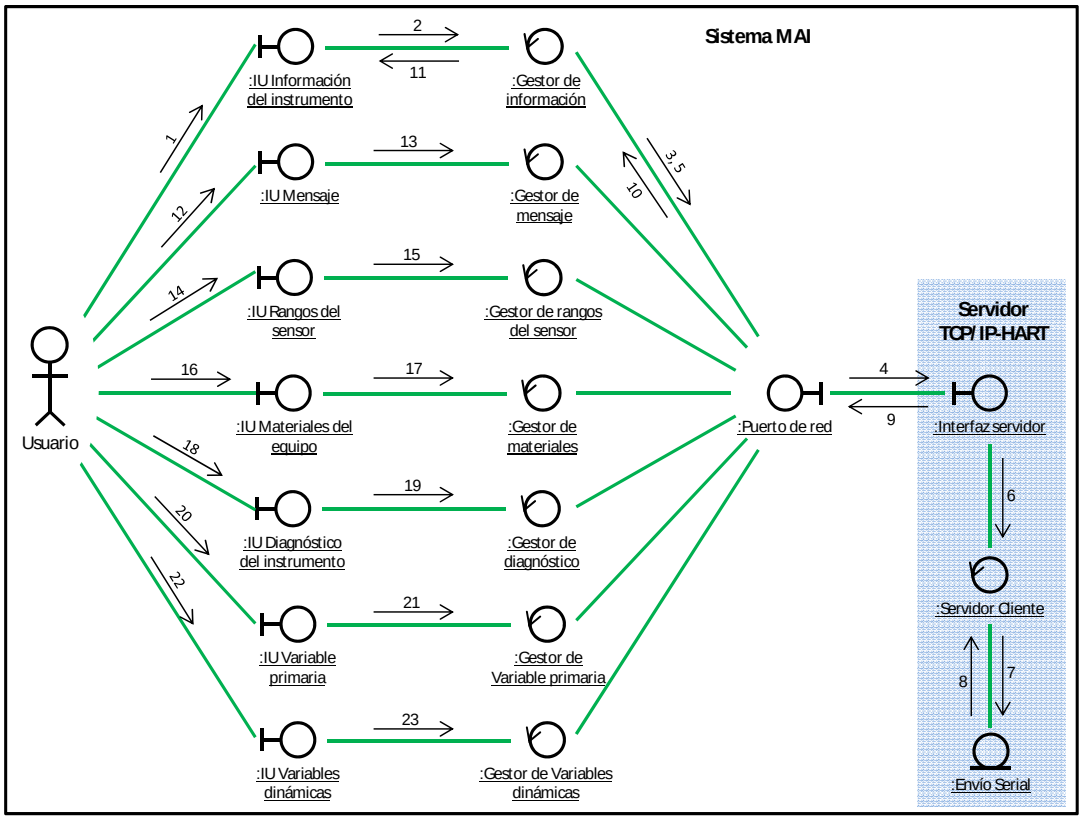

Figura 3.11. Diagrama de colaboración del caso de uso monitorear instrumentación

Fuente: Elaboración propia

3.4.2.4.- Diagrama de colaboración del caso de uso Calibrar instrumentación

En la figura 3.12 se ilustra el diagrama de colaboración del caso de uso calibrar instrumentación, en las tablas 3.7 y 3.8 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

3.4.2.5.- Diagrama de colaboración del caso de uso Administrar instrumentos

En la figura 3.13 se ilustra el diagrama de colaboración del caso de uso administrar instrumentos, en la tabla 3.9 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

Tabla 3.6. Lista de mensajes del diagrama de colaboración Monitorear instrumentación

Fuente: Elaboración propia

Leyenda		
1. Consultar información de instrumento	*9. Recibir trama de respuesta*	*17. Construir trama HART con comando de materiales*
2. Construir trama HART con comando de información	*10. Interpretar trama de respuesta*	*18. Consultar diagnóstico del instrumento*
3. Conectar a servidor	*11. Presentar Información*	*19. Construir trama HART con comando de diagnóstico*
4. Admitir conexión	*12. Consultar mensaje de instrumento*	*20. Consultar variable primaria*
5. Encapsular trama y enviar	*13. Construir trama HART con comando de mensaje*	*21. Construir trama HART con comando de variable primaria*
6. Recibir Trama HART	*14. Consultar rangos del sensor*	*22. Consultar variables dinámicas*
7. Transmitir trama	*15. Construir trama HART con comando de rangos del sensor*	*23. Construir trama HART con comando de variables dinámicas*
8. Enviar trama respuesta	*16. Consultar materiales*	

Tabla 3.7. Lista de mensajes del diagrama de colaboración Calibrar instrumentación 1/2

Fuente: Elaboración propia

Leyenda			
1. Asignar información de instrumento	4. Definir estado de bloqueo	7. Asignar rangos del sensor	10. Asignar dirección
2. Construir trama HART	5. Construir trama HART	8. Construir trama HART	11. Construir trama HART
3. Enviar trama	6. Enviar trama	9. Enviar trama	12. Enviar trama

Tabla 3.8. Lista de mensajes del diagrama de colaboración Calibrar instrumentación 2/2

Fuente: Elaboración propia

Leyenda			
13. Seleccionar función de transferencia	16. Asignar unidad de ingeniería	19. Ajustar damping	22. Realizar ajustes trim
14. Construir trama HART	17. Construir trama HART	20. Construir trama HART	23. Construir trama HART
15. Enviar trama	18. Enviar trama	21. Enviar trama	24. Enviar trama

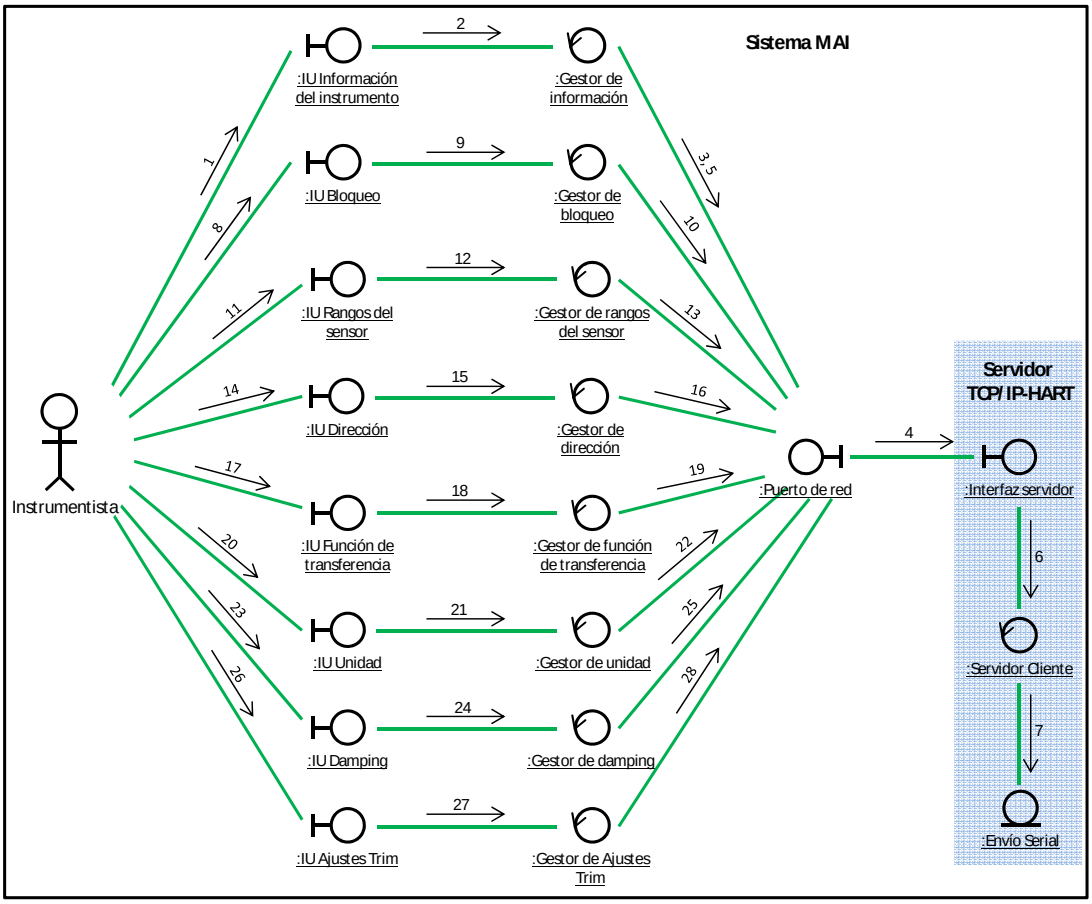

Figura 3.12. Diagrama de colaboración del caso de uso Calibrar instrumentación

Fuente: Elaboración propia

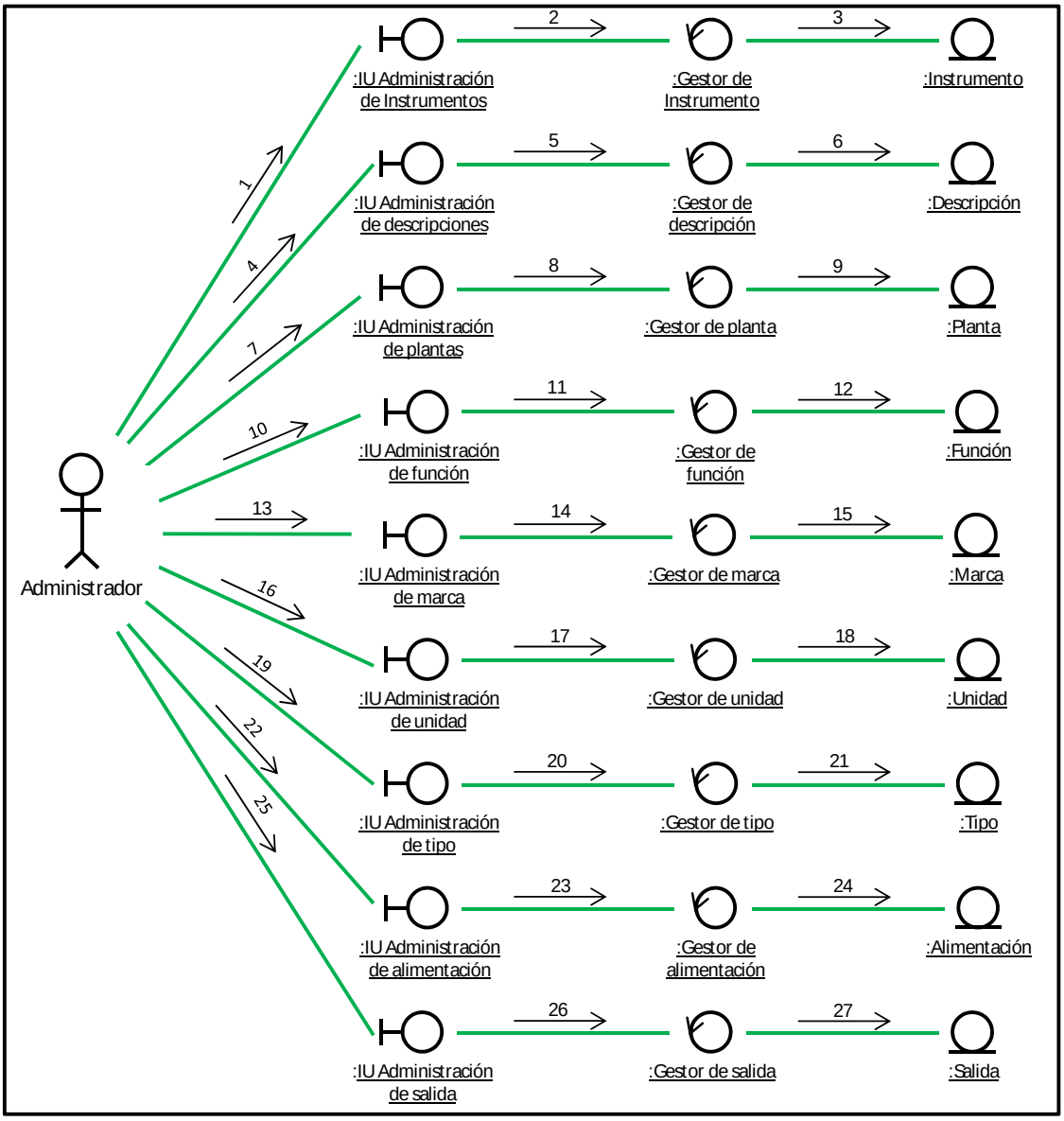

Figura 3.13. Diagrama de colaboración del caso de uso Administrar instrumentos

Fuente: Elaboración propia

Tabla 3.9. Lista de mensajes del diagrama de colaboración Administrar instrumentos

Fuente: Elaboración propia

Leyenda			
1. Agregar instrumento	*8. Validar datos de planta*	*15. Modificar registro*	*22. Modificar alimentación*
2. Validar datos de instrumento	*9. Eliminar registro*	*16. Eliminar unidad*	*23. Validar datos de alimentación*
3. Registrar instrumento	*10. Agregar función*	*17. Validar unidad*	*24. Modificar registro*
4. Modificar descripción	*11. Validar función*	*18. Eliminar registro*	*25. Eliminar salida*
5. Validar datos de descripción	*12. Registrar función*	*19. Agregar tipo*	*26. Validar salida*
6. Modificar registro	*13. Modificar marca*	*20. Validar datos de tipo*	*27. Eliminar registro*
7. Eliminar planta	*14. Validar marca*	*21. Registrar tipo*	

3.5.- Diseño

El proceso de diseño está centrado en desarrollar un modelo del sistema software para implementar los requisitos identificados. La finalidad del flujo de trabajo de diseño en esta primera fase de desarrollo, es identificar los subsistemas que conforman el software con el propósito de modelar la arquitectura base del proyecto, la cual será explotada en las fases subsiguientes.

3.5.1.- Diagrama de Paquetes de análisis

El diagrama de paquetes sirve para encapsular las diferentes funcionalidades con que ha de contar el sistema, relacionando los casos de uso representados por elipsis con el o los paquetes con los cuales se encuentran relacionados.

En la figura 3.14, se ilustran las funcionalidades del sistema encapsulado en paquete de análisis, según determinados requerimientos colectivos que éstas poseen. Se observan en la figura las diferentes elipsis que representan los casos de uso del sistema. Los paquetes se asocian a las funcionalidades mediante líneas punteadas que parten de los paquetes, y en el otro extremo apuntan a los casos de uso.

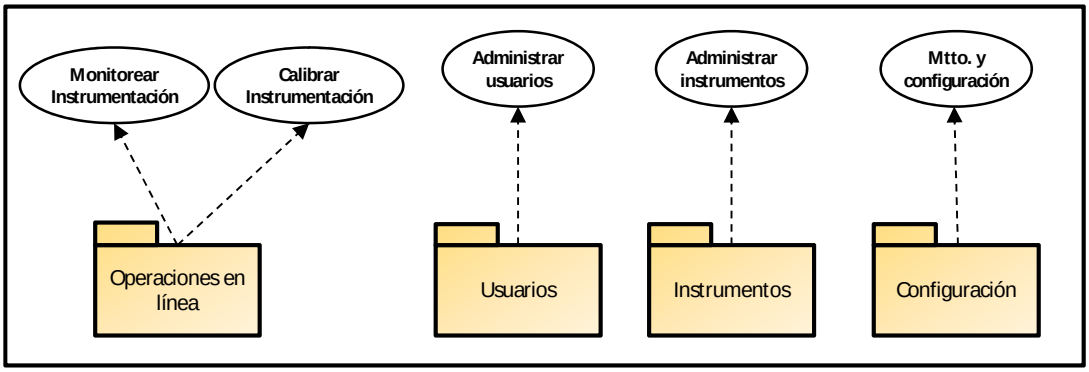

Figura 3.14. Paquetes de análisis a partir de los casos de uso del sistema

Fuente: Elaboración propia

3.6.- Conclusión de la fase de inicio

La fase de inicio sirvió para capturar los requerimientos básicos del sistema, las descripciones tanto generales como del dominio del problema, estableciendo los casos de uso e identificando los posibles riesgos. Los riesgos identificados se mitigaron, haciendo uso de los diagramas de caso de uso, los diagramas de clase de

análisis, de colaboración y la abstracción inicial de las funcionalidades propias del sistema mediante un diagrama de paquetes.

Esta ha sido una etapa fundamental para sentar las bases del proyecto, y así, pasar a la siguiente fase, fase de elaboración, donde se procederá a estudiar con mayor profundidad los requisitos, y establecer diagramas formales de la estructura, comportamiento e interacción del sistema.

3.7.- Planificación de las siguientes fases

Continuando con el proceso de desarrollo, se procederá a ejecutar la siguiente fase. Se tomarán los requerimientos funcionales y técnicos más significativos, recolectados en la fase de inicio, para formular un análisis más detallado y exhaustivo con la cual se pueda diseñar una base solida de la arquitectura del sistema.

CAPÍTULO 4
FASE DE ELABORACIÓN

4.1.- Introducción

El objetivo de esta fase es entender muy bien el problema desde el punto de vista del equipo de desarrollo. Lleva consigo la elaboración de la arquitectura marco del sistema y el diseño de la solución técnica.

En este punto del proceso, la arquitectura, los requerimientos y los planes de desarrollo son estables. Existen menos riesgos y se puede planificar el resto del proyecto con menor incertidumbre. Se construye una arquitectura ejecutable que contemple los casos de uso críticos y los riesgos identificados. Todos los componentes restantes se desarrollan e incorporan al producto, todo es probado en profundidad y el énfasis está en la producción eficiente y no en la creación. Se agregan nuevas clases que proveen de la infraestructura técnica: interfaces de usuario, manejo de bases de datos, comunicaciones con otros sistemas, etc. El diseño resulta en especificaciones detalladas para las tareas de programación.

Al final de la fase se tiene definida la arquitectura, el modelo de requisitos del sistema empleando los diagramas de casos de uso especificados en lenguaje UML, el plan de desarrollo y los estándares de calidad que se han de seguir en el proyecto o las herramientas que se han de emplear durante el transcurso del mismo. A continuación se muestra en un diagrama la realización de las dos iteraciones en la fase de Elaboración.

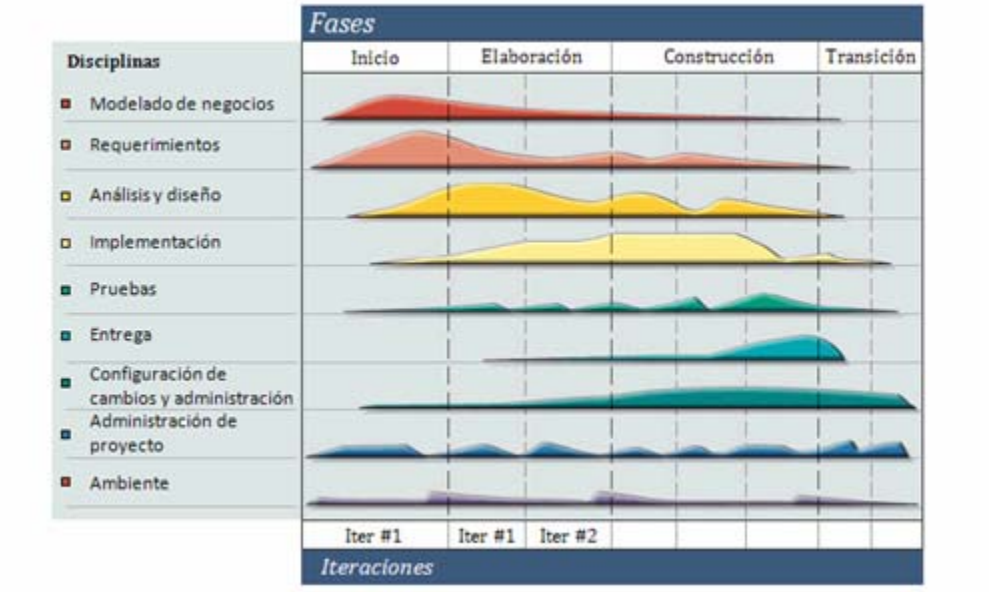

Figura 4.1. Fases del proceso unificado

Fuente: Jacobson, I, Booch, G, Rumbaugh J, 2000

4.2.- Captura de requisitos

En esta disciplina se recolectan los requisitos del sistema, que no fueron identificados en la fase previa. A continuación se describen los nuevos requisitos identificados en esta fase:

- Los usuarios con estatus de instrumentistas serán los únicos con privilegios de calibración en línea de la instrumentación de campo. Sin embargo, para que un instrumentista pueda realizar cualquier operación de calibración requerirá de un permiso de acceso que será otorgado por un conjunto de usuarios involucrados como son: un superintendente de planta, un supervisor de planta y un usuario de soporte de automatización. Serán estrictamente necesarias las autorizaciones de cada uno de los usuarios ya mencionados incluyendo la del propio instrumentista que realizará la operación bajo su sesión. Esto

garantizará que el personal responsable del instrumento este informado de la operación de calibración.

- En las operaciones de consulta en línea de variables primarias y dinámicas sobre la instrumentación, el sistema debe ser capaz de generar gráficos que describan el comportamiento de dichas variables en tiempo real.

4.2.1.- Diagrama general de casos de uso

Esta segunda iteración de captura de requisitos no se ha identificado nuevos casos de uso, sin embargo, si se han identificado nuevos requisitos que amplían la funcionalidad de casos de uso ya existentes. En la figura 4.2 se presenta el diagrama de casos de uso general resaltando los casos de uso que se han adaptado a los nuevos requisitos capturados.

4.3.- Análisis

En esta disciplina se realizará el análisis de los nuevos requisitos identificados en esta fase y así seguir formando el modelo inicial del sistema que captura la semántica y comportamiento.

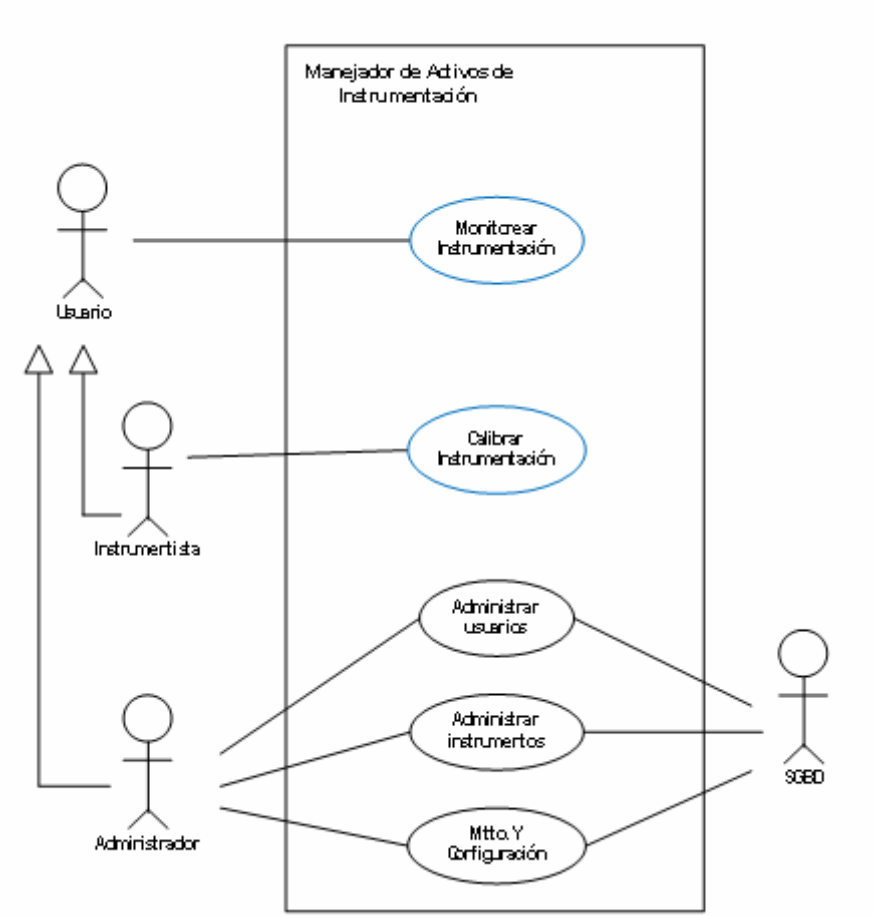

Figura 4.2. Diagrama de casos de uso reestructurado

Fuente: Elaboración propia

4.3.1.- Diagrama de clases de análisis

A continuación se describen los diagramas de clases de análisis a los cuales se les incorporan clases a partir de los nuevos requisitos captados. Estas amplían las funcionalidades de algunos casos de uso identificados en la fase anterior.

4.3.1.1.- Diagrama de clases de análisis del caso de uso Monitorear instrumentación

En la figura 4.3 se ilustra el diagrama de clases de análisis actualizado del caso de uso

monitorear instrumentación. Como se puede observar las clases gestor de variable primaria y gestor de variables dinámicas se conectaron a la clase generador de gráficos la cual se encargará de crear los gráficos para cada una de las variables de manera dinámica permitiendo al usuario ver el comportamiento de las variables a través del tiempo de forma gráfica.

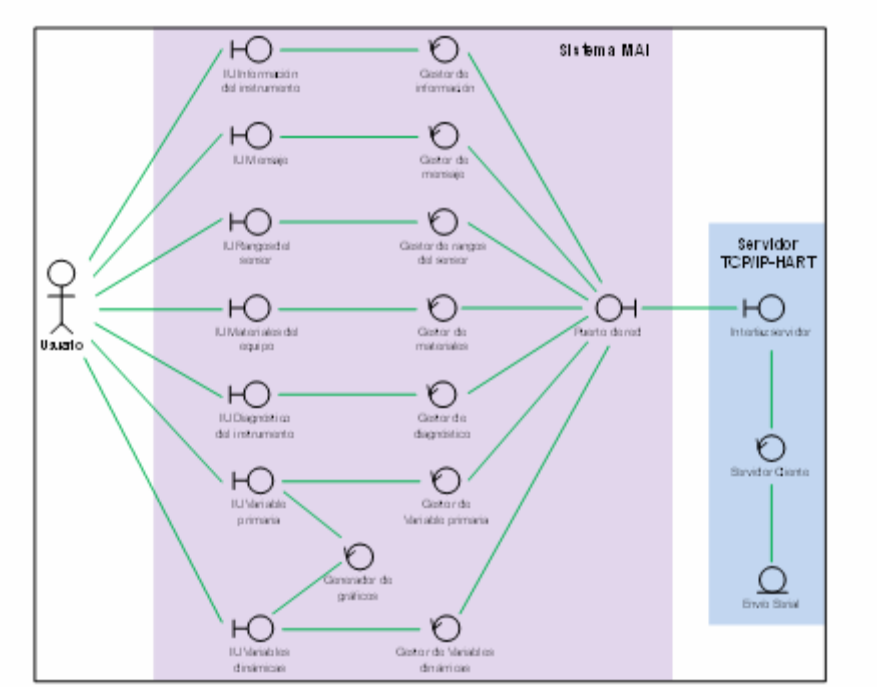

Figura 4.3. Diagrama de clases de análisis actualizado del caso de uso Monitorear instrumentación

Fuente: Elaboración propia

4.3.1.2.- Diagrama de clases de análisis actualizado del caso de uso Calibrar instrumentación

La figura 4.4 ilustra la actualización del diagrama de clases de análisis del caso de uso calibrar instrumentación, donde se incorporaron las clases IU permiso de calibración, Verificación de permiso y usuario que incluirá una verificación de

permiso para realizar cualquier operación de calibración bajo la sesión de instrumentista quien es el único actor con privilegios para realizar este tipo de operaciones en línea.

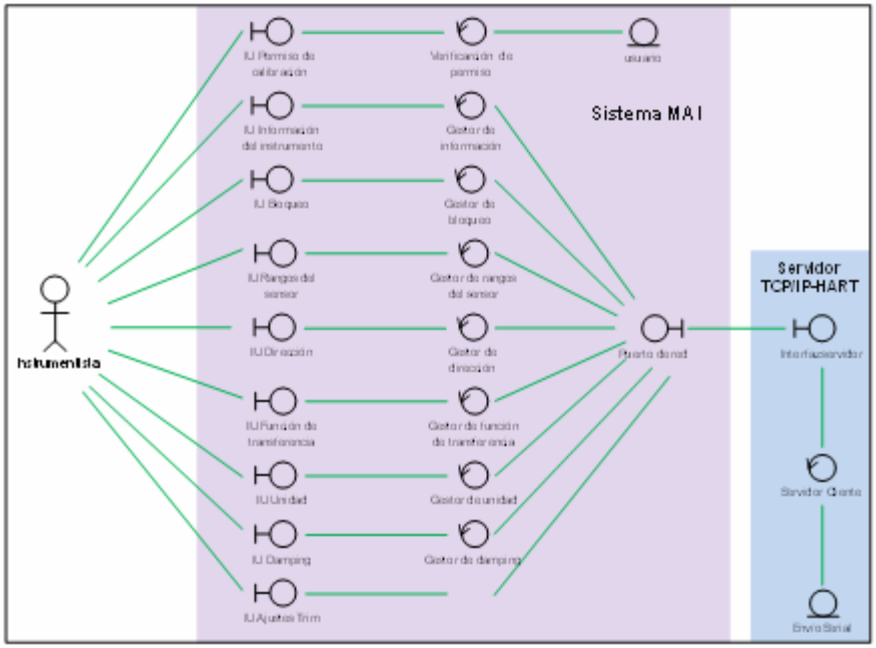

Figura 4.4. Diagrama de clases de análisis actualizado del caso de uso Calibrar instrumentación

Fuente: Elaboración propia

Esta secuencia de verificación se realizará de manera automática cuando se intente realizar una operación de calibración a un instrumento a través del sistema, como se describió en la captura de requisitos en esta fase este mecanismo incrementará la seguridad del sistema en cuanto a las operaciones de calibración y a su vez garantizará que el personal responsable de la planta donde opera un instrumento en particular este informado de cualquier operación que pueda modificar la manera en cómo se comporta el dispositivo que por consiguiente incidirá en el comportamiento de una planta.

4.3.2.- Diagrama de colaboraciones

A continuación se describen los diagramas de colaboración a los cuales se les incorporan las nuevas clases de análisis identificadas a partir de los requisitos captados en esta fase.

4.3.2.1.- Diagrama de colaboración actualizado del caso de uso Monitorear instrumentación

La figura 4.5 ilustra el d*iagrama de colaboración actualizado del caso de uso monitorear instrumentación en donde se ha incorporado la clase generador de gráfica a partir de un nuevo requisito identificado en esta fase.* En la tabla 4.1 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

4.3.2.2.- Diagrama de colaboración actualizado del caso de uso calibrar instrumentación

La figura 4.6 ilustra el diagrama de colaboración actualizado del caso de uso calibrar instrumentación que describe cómo interactúan los objetos de las nuevas clases incluidas al diagrama. En la tabla 4.2 se listan los mensajes que describen como interactúan los objetos entre si durante la ejecución de este caso de uso.

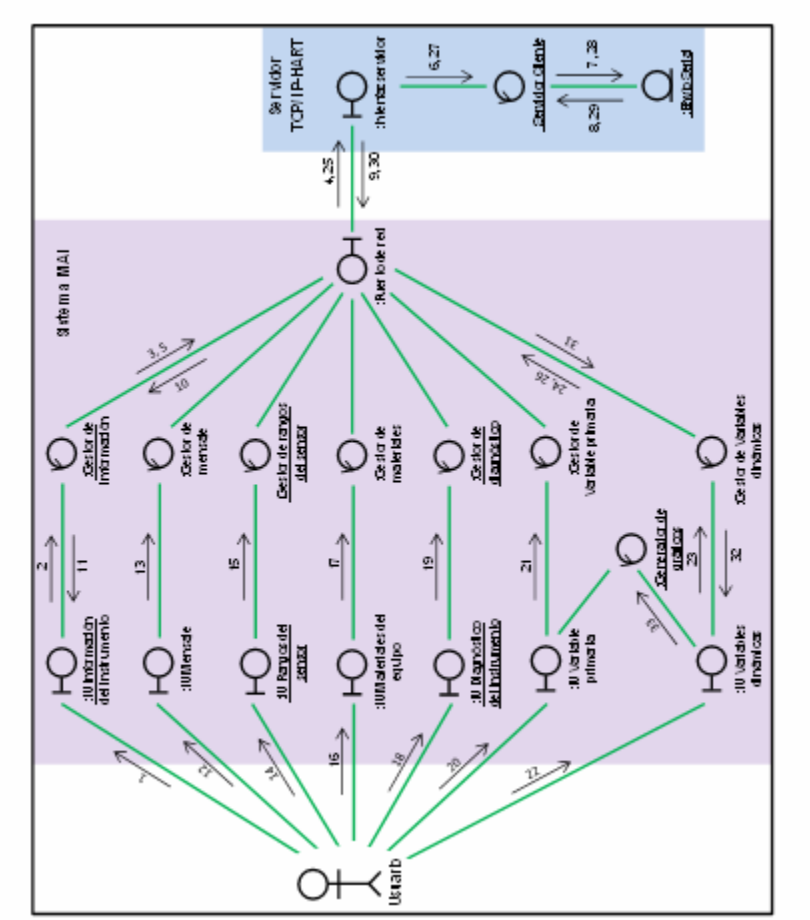

Figura 4.5. Diagrama de colaboración actualizado del caso de uso Monitorear instrumentación

Fuente: Elaboración propia

Tabla 4.1. Lista de mensajes del diagrama de colaboración actualizado del caso de uso Monitorear instrumentación

Fuente: Elaboración propia

Leyenda		
1. Consultar información de instrumento	*12. Consultar mensaje de instrumento*	*23. Construir trama HART con comando de variables dinámicas*

2. Construir trama HART con comando de información	*13. Construir trama HART con comando de mensaje*	*24. Conectar a servidor*
3. Conectar a servidor	*14. Consultar rangos del sensor*	*25. Admitir conexión*
4. Admitir conexión	*15. Construir trama HART con comando de rangos del sensor*	*26. Encapsular trama y enviar*
5. Encapsular trama y enviar	*16. Consultar materiales del instrumento*	*27. Recibir Trama HART*
6. Recibir Trama HART	*17. Construir trama HART con comando de materiales*	*28. Transmitir trama*
7. Transmitir trama	*18. Consultar diagnóstico del instrumento*	*29. Enviar trama respuesta*
8. Enviar trama respuesta	*19. Construir trama HART con comando de diagnóstico*	*30. Recibir trama de respuesta*
9. Recibir trama de respuesta	*20. Consultar variable primaria*	*31. Interpretar trama de respuesta*
10. Interpretar trama de respuesta	*21. Construir trama HART con comando de variable primaria*	*32. Retornar variables dinámicas*
11. Presentar Información	*22. Consultar variables dinámicas*	*33. Generar gráfica*

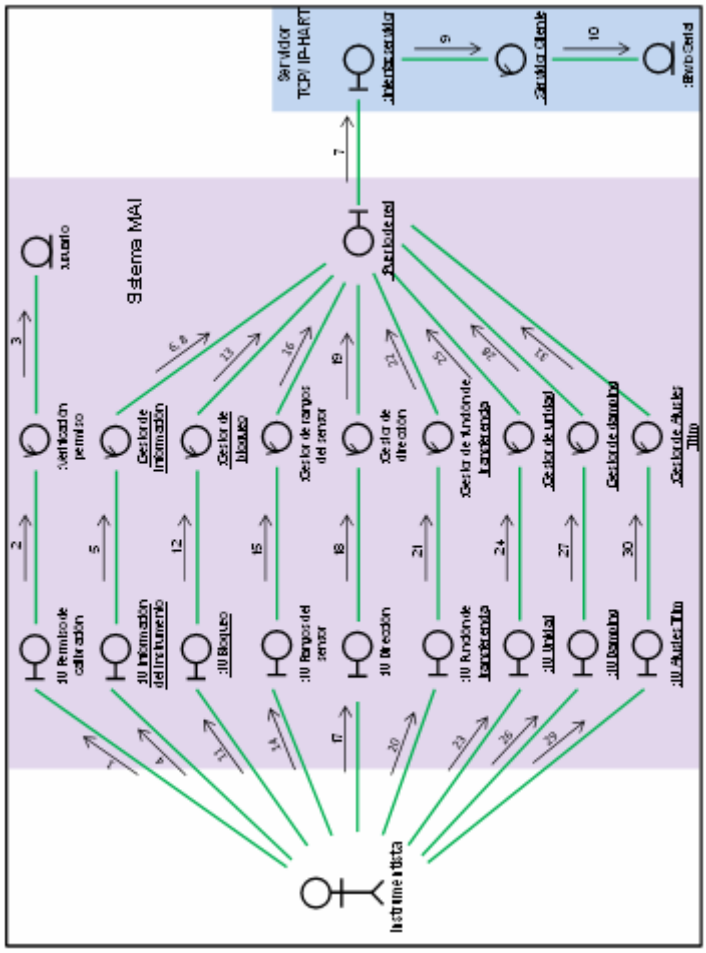

Figura 4.6. Diagrama de colaboración actualizado del caso de uso calibrar instrumentación

Fuente: Elaboración propia

Tabla 4.2. Lista de mensajes del diagrama de colaboración actualizado del caso de uso calibrar instrumentación

Fuente: Elaboración propia

Leyenda		
1.Ingresar	*10. Asignar rangos*	*19. Asignar unidad*

usuarios para permiso de calibración	*del sensor*	*de ingeniería*
2.Validar datos de usuarios	*11. Construir trama HART*	*20. Construir trama HART*
3. Buscar datos	*12. Enviar trama*	*21. Enviar trama*
4. Asignar información de instrumento	*13. Asignar dirección*	*22. Ajustar damping*
5. Construir trama HART	*14. Construir trama HART*	*23. Construir trama HART*
6. Enviar trama	*15. Enviar trama*	*24. Enviar trama*
7. Definir estado de bloqueo	*16. Seleccionar función de transferencia*	*25. Realizar ajustes trim*
8. Construir trama HART	*17. Construir trama HART*	*26. Construir trama HART*
9. Enviar trama	*18. Enviar trama*	*27. Enviar trama*

4.4.- Diseño

Durante el ciclo de desarrollo iterativo, es posible pasar a la fase de diseño, una vez terminados los documentos de análisis. Durante este proceso se logra una solución lógica que se funda en el paradigma orientado a objetos. Su esencia es la elaboración

de diagramas de interacción, que muestran exactamente como los objetos se comunicarán a fin de cumplir con los requisitos.

4.4.1.- Diagrama de clases de diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente estos tipos de diagramas contienen la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real. El UML no define concretamente un elemento denominado “Diagrama de clases de diseño” sino que hace referencia a un término más genérico “Diagrama de clases”. Algunos autores optan por incluir el término de “diseño” en “diagrama de clases”. Para recalcar que se trata de una perspectiva desde el punto de vista del diseño de las entidades de software y no de una concepción analítica sobre los conceptos del domino.

A continuación se presentan los diagramas de clases de diseño por cada caso de uso:

4.4.1.1.- Diagrama de clase de diseño para el caso de uso Administrar usuarios

La figura 4.7 ilustra el diagrama de clases de diseño para el caso de uso Administrar usuarios. En este diagrama se observa la clase IUPresentacion en donde se encuentra el menú para seleccionar cualquier operación, en esta ocasión referido a la administración de usuarios.

La clase IUPresentacion tiene una conexión de dependencia con la clase Sesion esta clase se encarga de validar el estado en la sesión del usuario a fin de verificar que la sesión se encuentre activa. De esta manera si se acciona una operación del menú en la clase IUPresentacion se valida la sesión del usuario y de ser válida se procede a cargar la sub interfaz de acuerdo a la operación seleccionada en el menú. Esta relación entre la clase IUPresentacion y la clase Sesion estará en todos los casos de uso del sistema funcionando de la misma manera.

4.4.1.2.- Diagrama de clase de diseño para el caso de uso Mantenimiento y configuración

En la figura 4.8 se ilustra el diagrama de clases de diseño para el caso de uso mantenimiento y configuración. En este se administran los servidores TCP/IP-HART que permitirán al sistema comunicarse con los instrumentos de campo, la administración se realiza al implementar las clases IU_Adm_Serv, Gestor_Serv y Servidor.

También este caso de uso realiza un escaneo de la red para localizar a los instrumentos conectados a través de los servidores TCP/IP-HART registrados. La clase PuertoRed del sistema MAI se conecta con la clase InterfazServidor perteneciente al sistema Servidor TCP/IP-HART al enviar un paquete por la red a la dirección IP del servidor, esto se represento con el símbolo *interfaz* de UML.

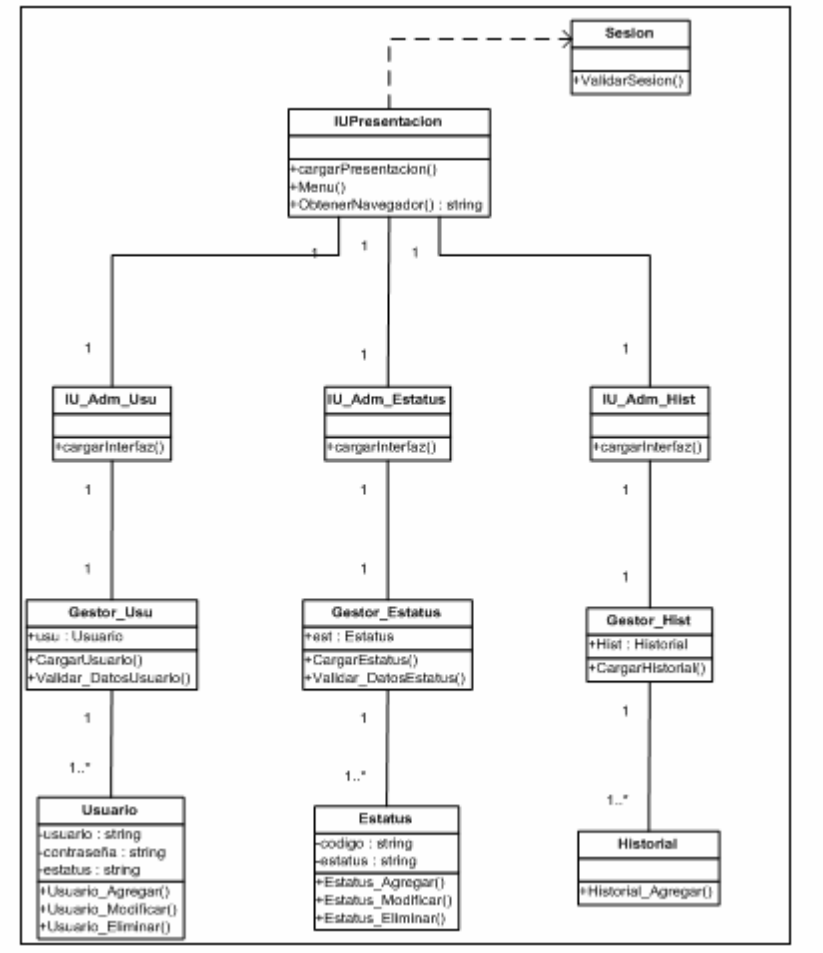

Figura 4.7. Diagrama de clases de diseño del caso de uso Administrar usuarios

Fuente: Elaboración propia

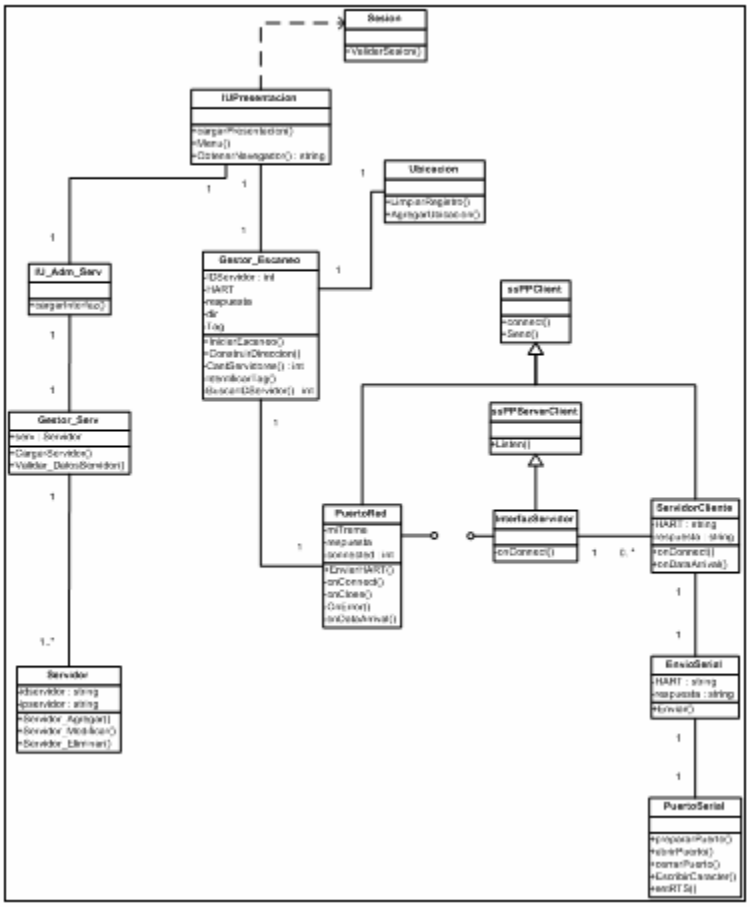

Figura 4.8. Diagrama de clases de diseño del caso de uso Mantenimiento y configuración

Fuente: Elaboración propia

4.4.1.3.- Diagrama de clase de diseño para el caso de uso Monitorear instrumentación

Las figuras 4.9 y 4.10 constituyen el conjunto de clases de diseño para el caso de uso monitorear instrumentación. En este se encuentran las clases que permiten realizar las operaciones de monitoreo sobre la instrumentación de campo.

Las clases con el prefijo IU representan las interfaces de usuario para presentar la información obtenida según la operación de monitoreo a ejecutar. Las clases Gestor se encargan de construir la trama HART que serán enviadas a través de la clase PuertoRed, este último se conectará al servidor TCP/IP-HART donde se encuentre el instrumento a monitorear y enviará la trama HART encapsulada en un paquete TCP/IP a través de la intranet de la empresa. El servidor se encargará de recibir el paquete y transmitir la trama HART por puerto serial al instrumento.

4.4.1.4.- Diagrama de clase de diseño para el caso de uso Calibrar instrumentación

Las figuras 4.11 y 4.12 constituyen el conjunto de clases de diseño para el caso de uso calibrar instrumentación. En este se encuentran las clases que permiten realizar las operaciones de calibración sobre la instrumentación de campo.

Las clases de diseño de este diagrama interactúan de la misma manera al caso de uso anterior, la diferencia a este caso de uso es que realiza operaciones de escritura sobre la instrumentación de campo que modifica los parámetros de operación del instrumento.

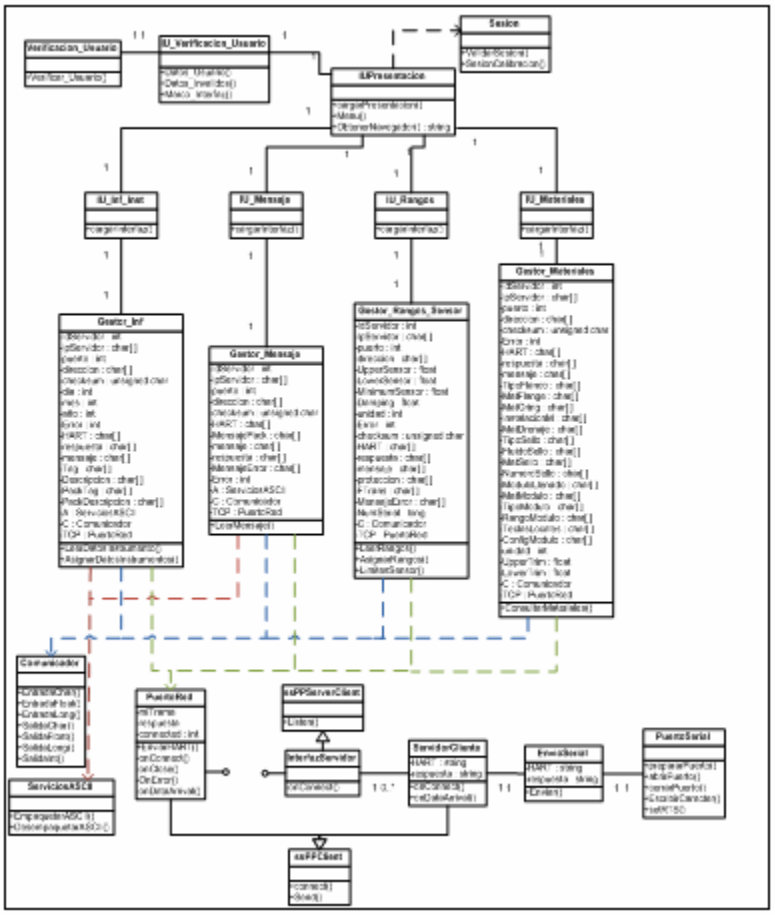

Figura 4.9. Diagrama de clases de diseño del caso de uso Monitorear instrumentación 1/2

Fuente: Elaboración propia

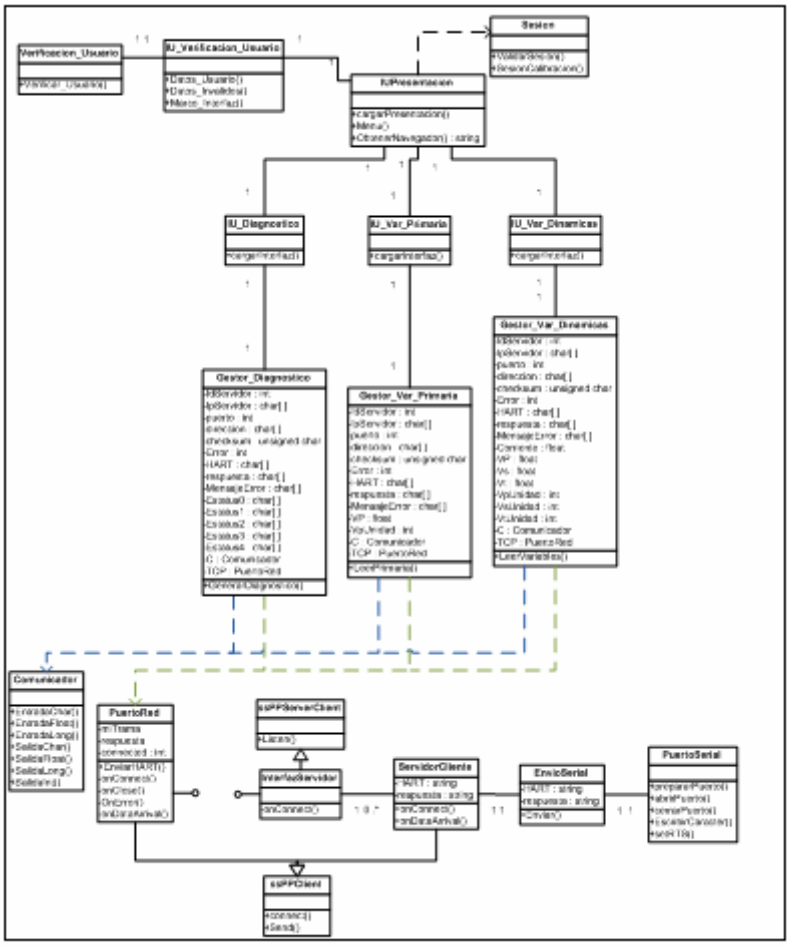

Figura 4.10. Diagrama de clases de diseño del caso de uso Monitorear instrumentación 2/2

Fuente: Elaboración propia

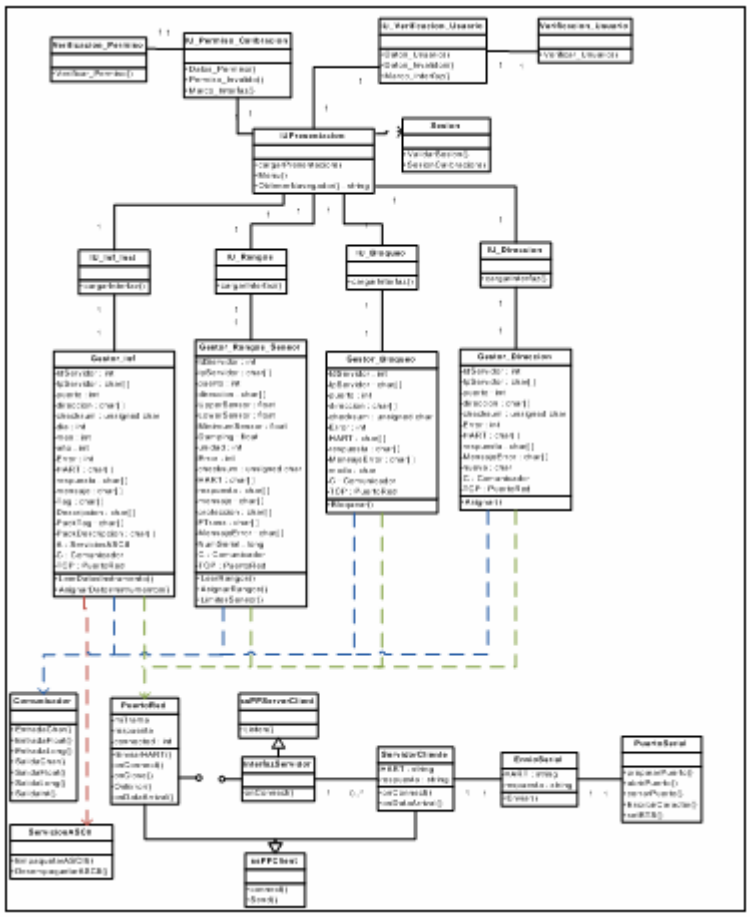

Figura 4.11. Diagrama de clases de diseño del caso de uso Calibrar instrumentación 1/2

Fuente: Elaboración propia

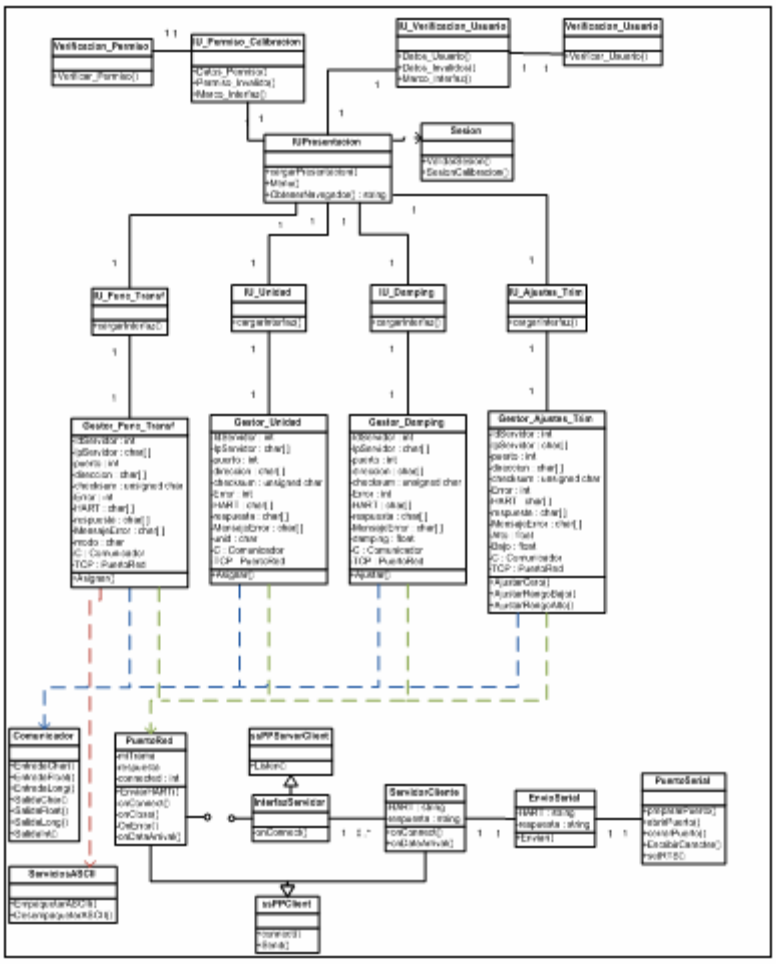

Figura 4.12. Diagrama de clases de diseño del caso de uso Calibrar instrumentación 2/2

Fuente: Elaboración propia

4.4.1.5.- Diagrama de clase de diseño para el caso de uso Administrar instrumentos

En las figuras 4.13 y 4.14 se ilustra el diagrama de clases de diseño para el caso de uso administrar instrumentos. En este se administran en la base de datos del sistema los instrumentos con información detallada de parámetros de operatividad y datos técnicos del dispositivo suministrado por el fabricante. En el diagrama se encuentran todas las clases que permiten la funcionalidad del caso de uso.

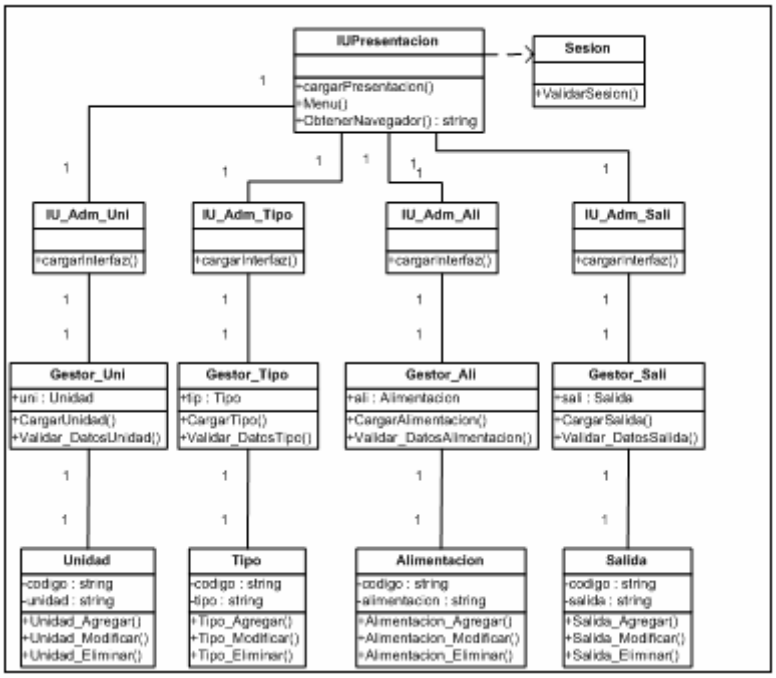

Figura 4.13. Diagrama de clases de diseño del caso de uso Administrar instrumentos 1/2

Fuente: Elaboración propia

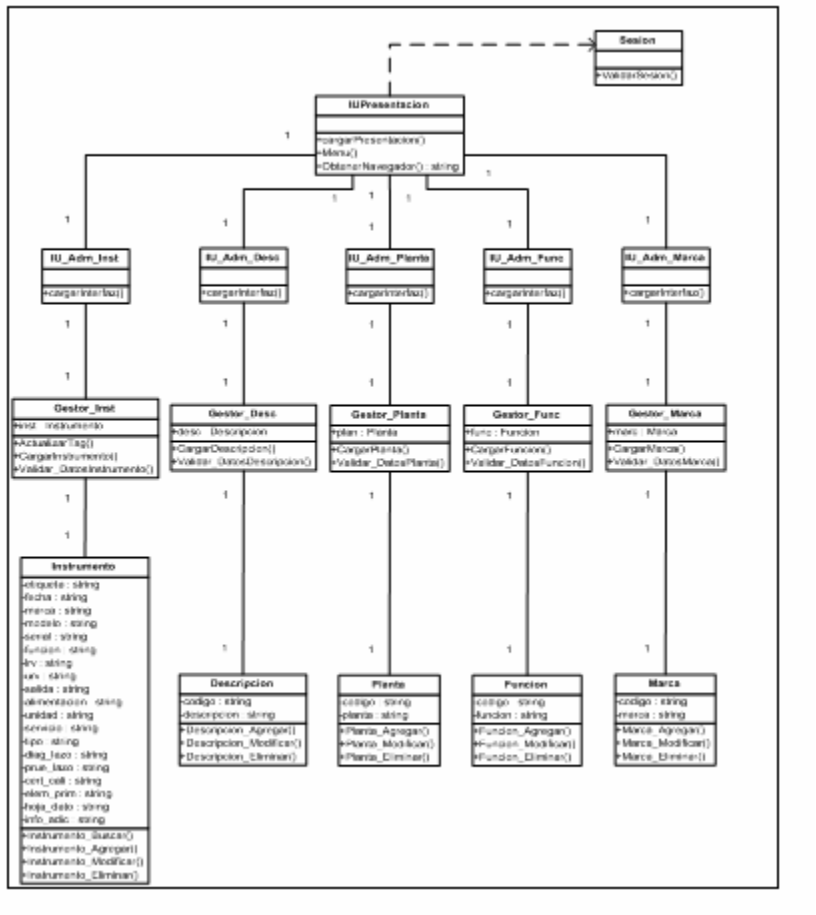

Figura 4.14. Diagrama de clases de diseño del caso de uso Administrar instrumentos 2/2

Fuente: Elaboración propia

4.4.2.- Diagrama de secuencia

El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista de negocio del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para

implementar el escenario, y mensajes pasados entre los objetos.

Con ello, el diagrama de secuencias tiene dos dimensiones. La dimensión horizontal es la disposición de los objetos, y la dimensión vertical muestra el paso del tiempo.

A continuación se presentan los diagramas de secuencia para cada caso de uso describiendo un escenario:

4.4.2.1.- Diagrama de secuencia para el caso de uso Administrar usuarios

El diagrama de secuencia para este caso de uso se ilustra en la figura 4.15 que presenta el escenario en el que el administrador agrega un nuevo estatus y agrega un usuario, por último, consulta el historial de calibración.

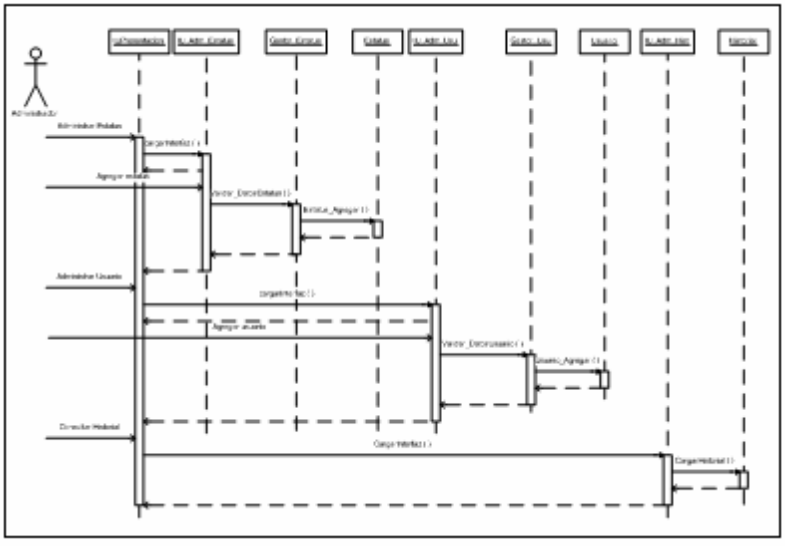

Figura 4.15. Diagrama de secuencia del caso de uso Administrar usuarios

Fuente: Elaboración propia

4.4.2.2.- Diagrama de secuencia para el caso de uso Mantenimiento y configuración

La figura 4.16 ilustra el diagrama de secuencia para el caso de uso mantenimiento y configuración, en un escenario donde el administrador agrega un nuevo servidor y ejecuta posteriormente el escaneo de la red para actualizar las rutas a los instrumentos de campo y así se incluyan los instrumentos conectados al nuevo servidor registrado previamente. En la secuencia del escaneo de red se representa a partir del objeto Gestor_Escaneo una iteración del proceso de escaneo ya que esto se debe realizar por cada servidor TCP/IP-HART registrado y su vez por cada puerto serial que compone a cada servidor.

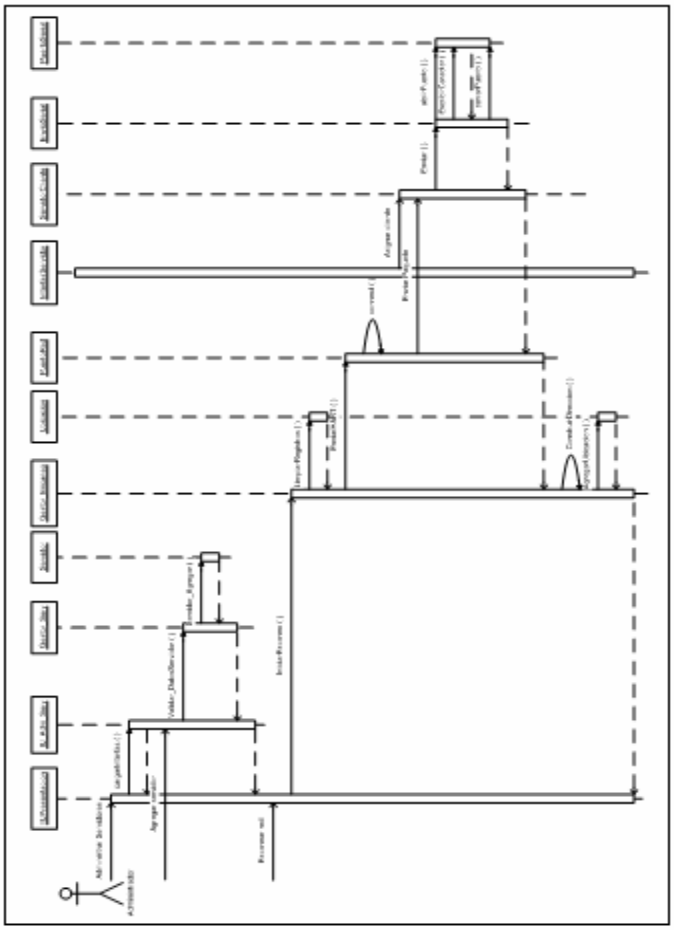

Figura 4.16. Diagrama de secuencia del caso de uso Mantenimiento y configuración

4.4.2.3.- Diagrama de secuencia para el caso de uso Monitorear instrumentación

Las figuras 4.17, 4.18, 4.19 y 4.20 ilustran los diagramas de secuencia para todas las operaciones en el caso de uso monitorear instrumentación.

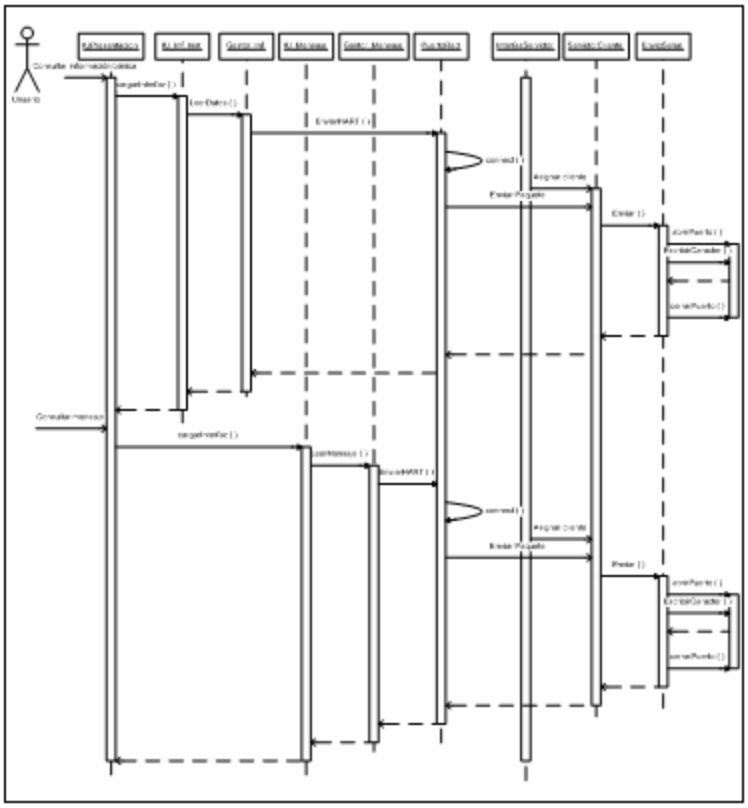

Figura 4.17. Diagrama de secuencia del caso de uso Monitorear instrumentación 1/4

Fuente: Elaboración propia

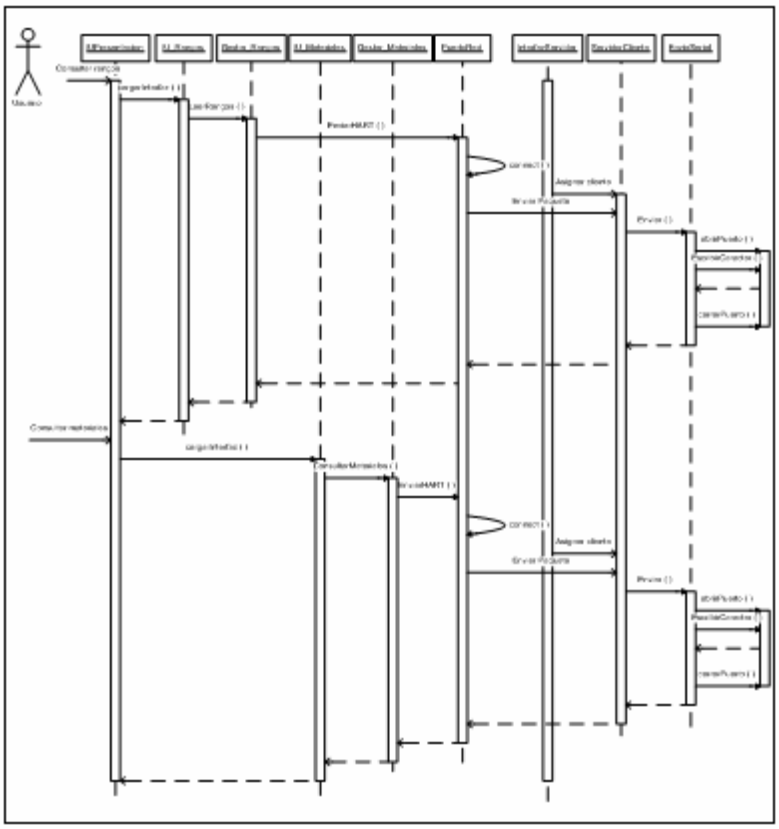

Figura 4.18. Diagrama de secuencia del caso de uso Monitorear instrumentación 2/4

Fuente: Elaboración propia

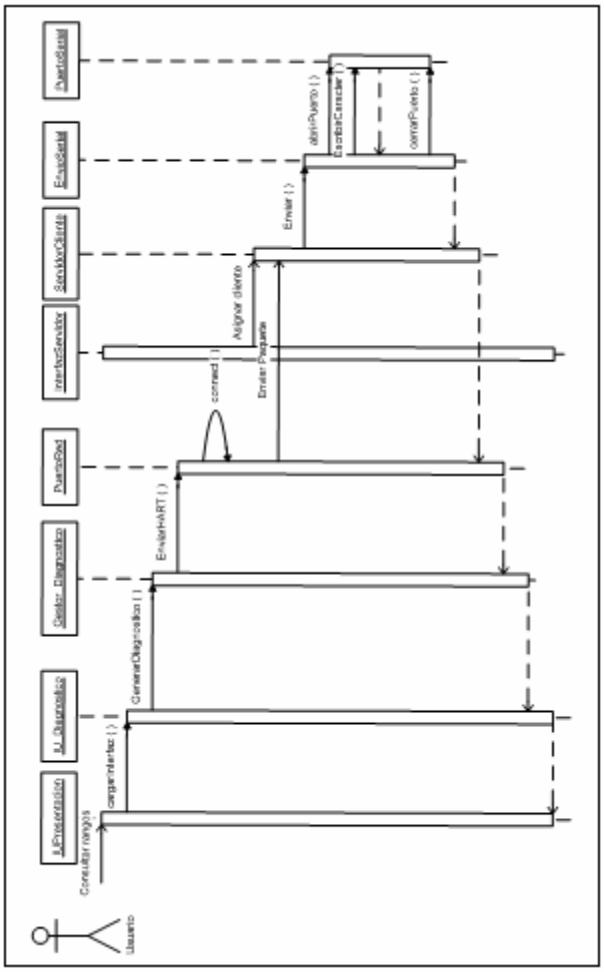

Figura 4.19. Diagrama de secuencia del caso de uso Monitorear instrumentación 3/4

Fuente: Elaboración propia

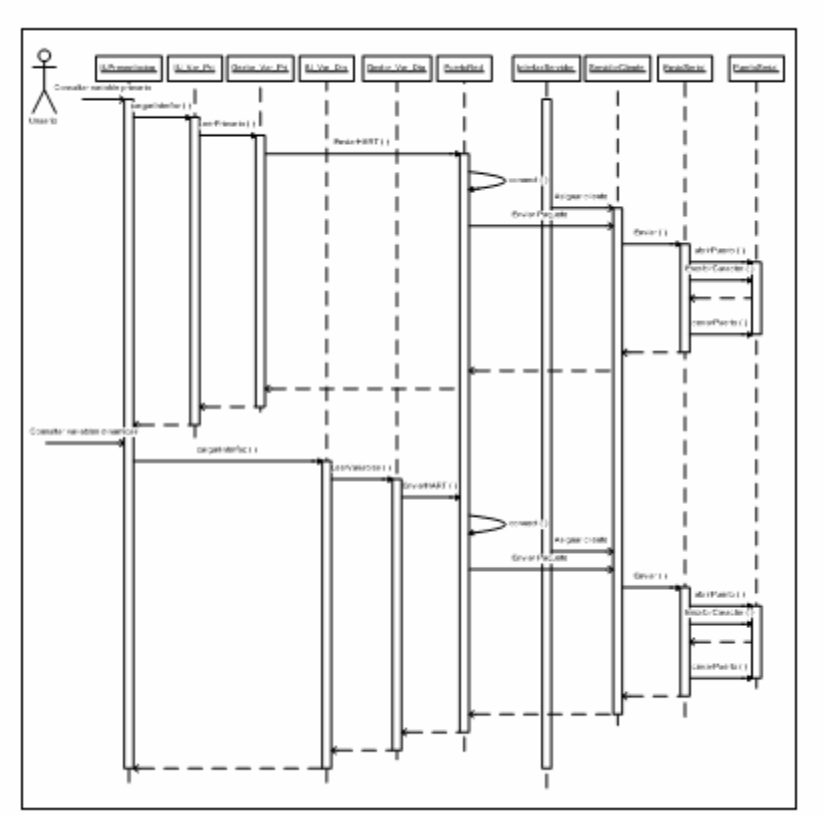

Figura 4.20. Diagrama de secuencia del caso de uso Monitorear instrumentación 4/4
Fuente: Elaboración propia

4.4.2.4.- Diagrama de secuencia para el caso de uso Calibrar instrumentación

Las figuras 4.21, 4.22, 4.23, 4.24 y 4.25 ilustran los diagramas de secuencia para todas las operaciones en el caso de uso calibrar instrumentación.

4.4.2.5.- Diagrama de secuencia para el caso de uso Administrar instrumentos

Las figuras 4.26, 4.27 y 4.28 ilustran los diagramas de secuencia para el caso de uso administrar instrumentos, que presenta un escenario donde el administrador realiza operaciones de agregar, modificar y eliminar.

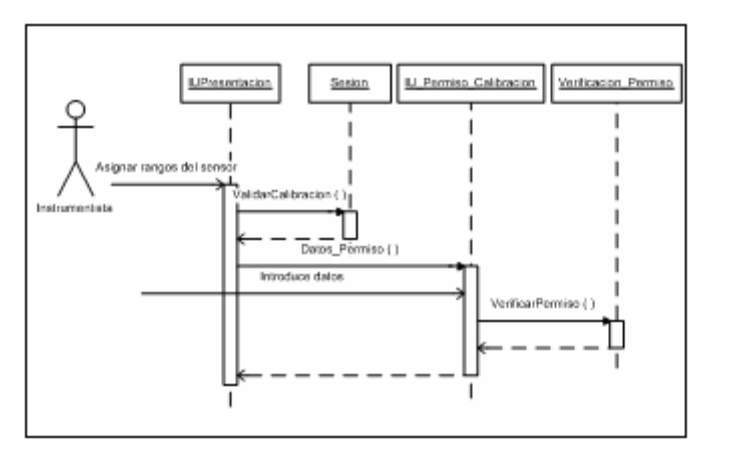

Figura 4.21. Diagrama de secuencia del caso de uso Calibración de instrumentación 1/5

Fuente: Elaboración propia

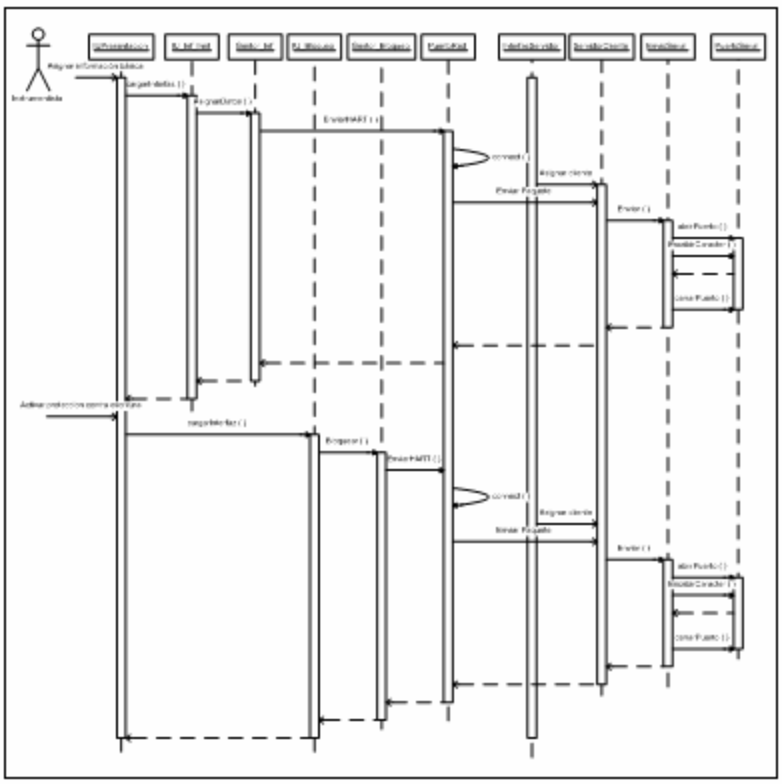

Figura 4.22. Diagrama de secuencia del caso de uso Calibración de instrumentación 2/5

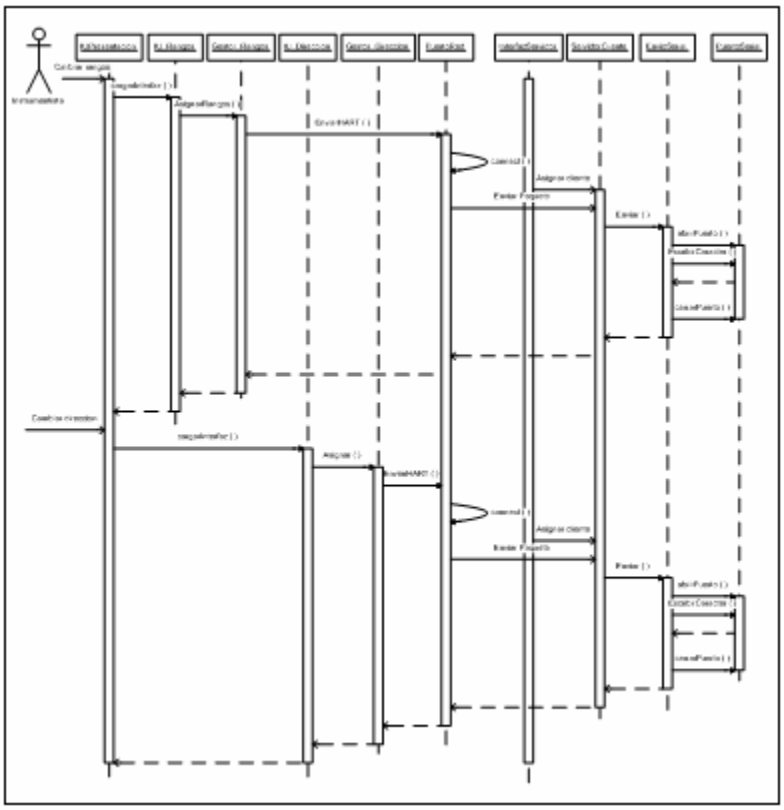

Figura 4.23. Diagrama de secuencia del caso de uso Calibración de instrumentación 3/5

Fuente: Elaboración propia

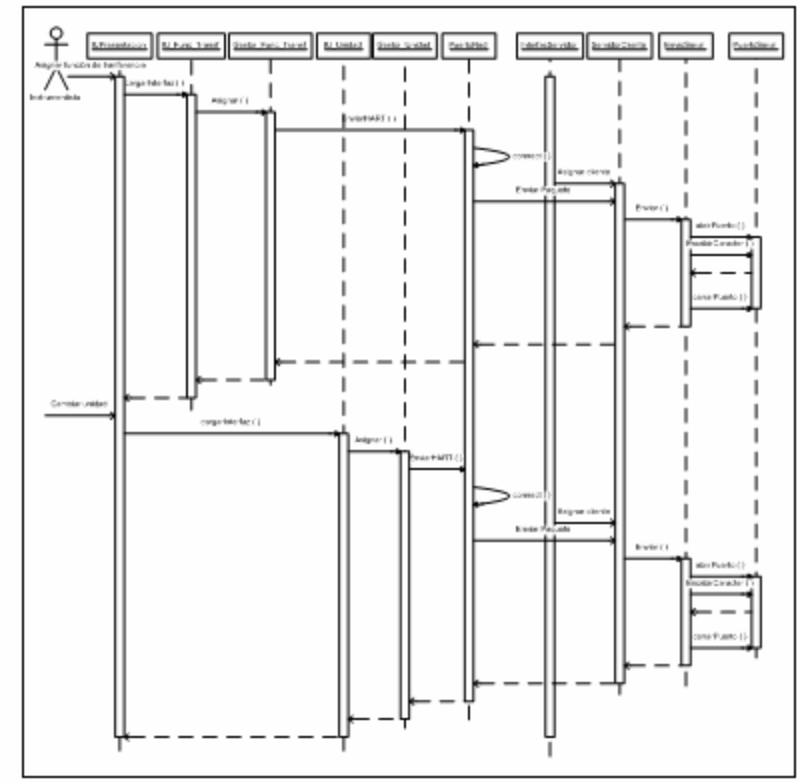

Figura 4.24. Diagrama de secuencia del caso de uso Calibración de instrumentación 4/5

Fuente: Elaboración propia

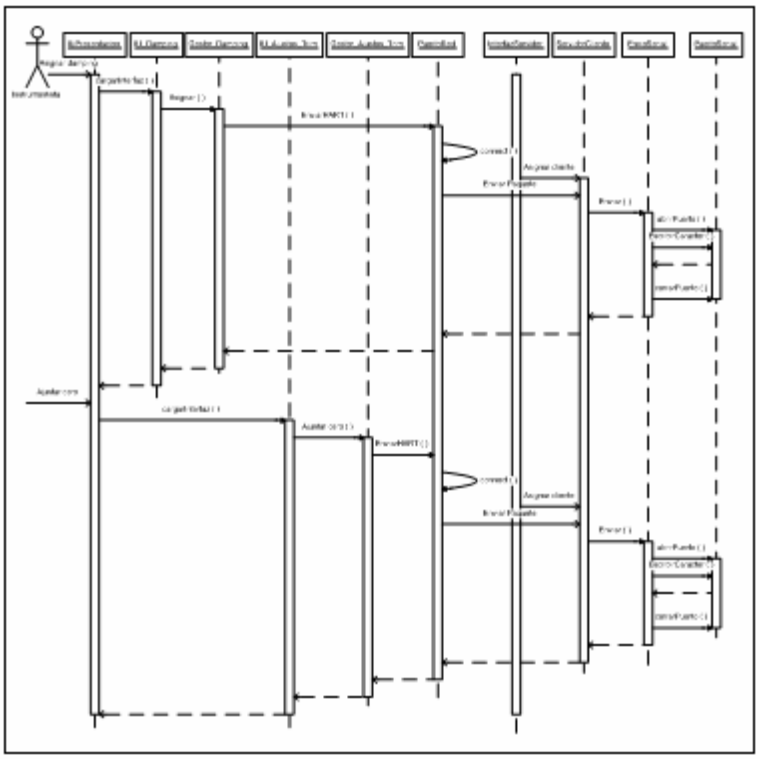

Figura 4.25. Diagrama de secuencia del caso de uso Calibración de instrumentación 5/5

Fuente: Elaboración propia

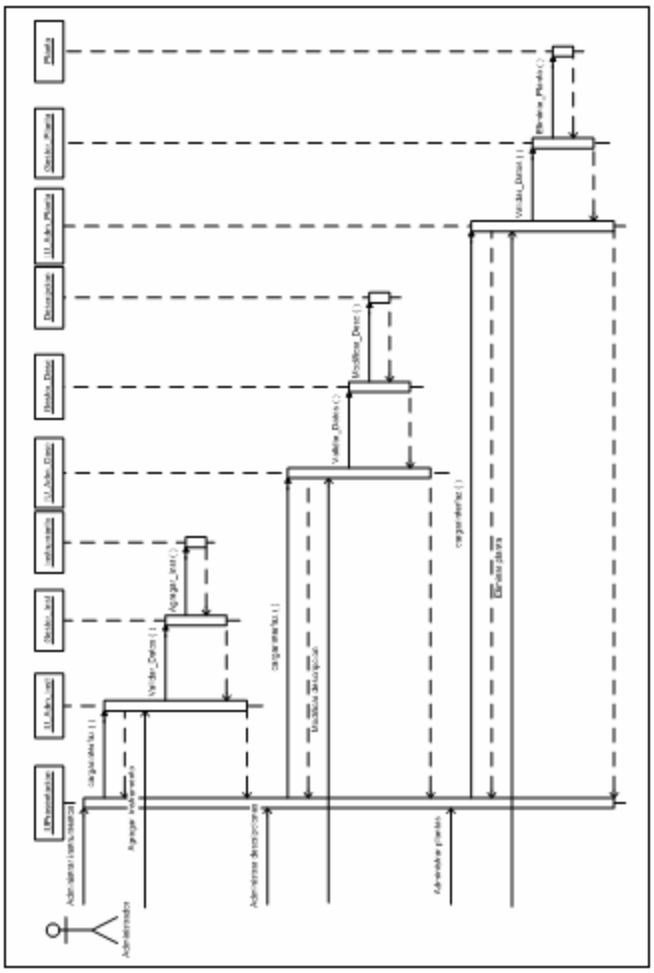

Figura 4.26. Diagrama de secuencia del caso de uso Administrar instrumentos 1/3

Fuente: Elaboración propia

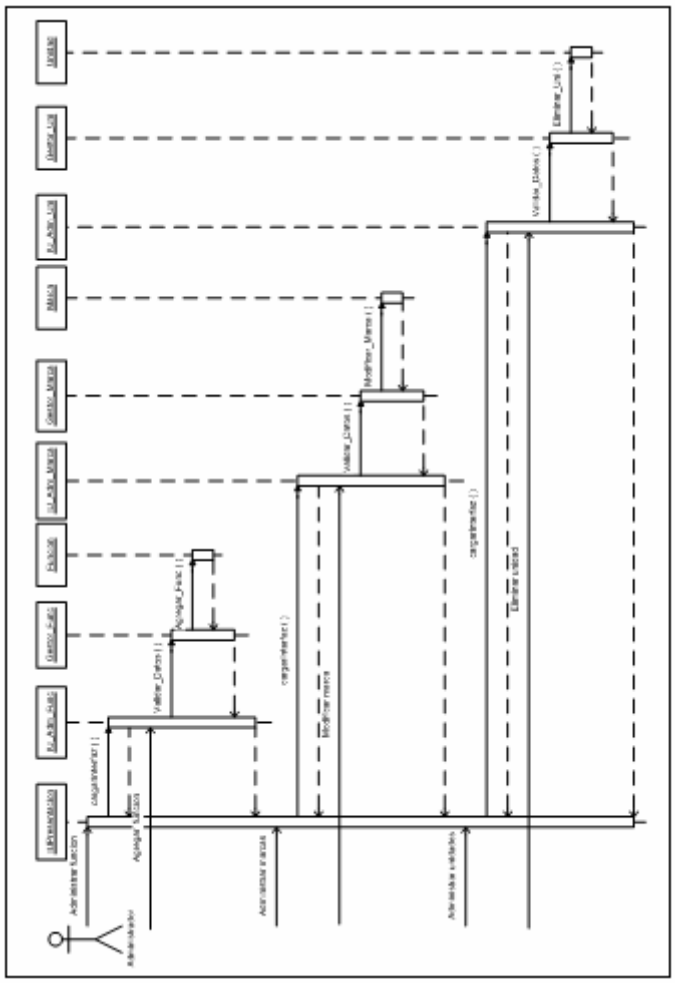

Figura 4.27. Diagrama de secuencia del caso de uso Administrar instrumentos 2/3

Fuente: Elaboración propia

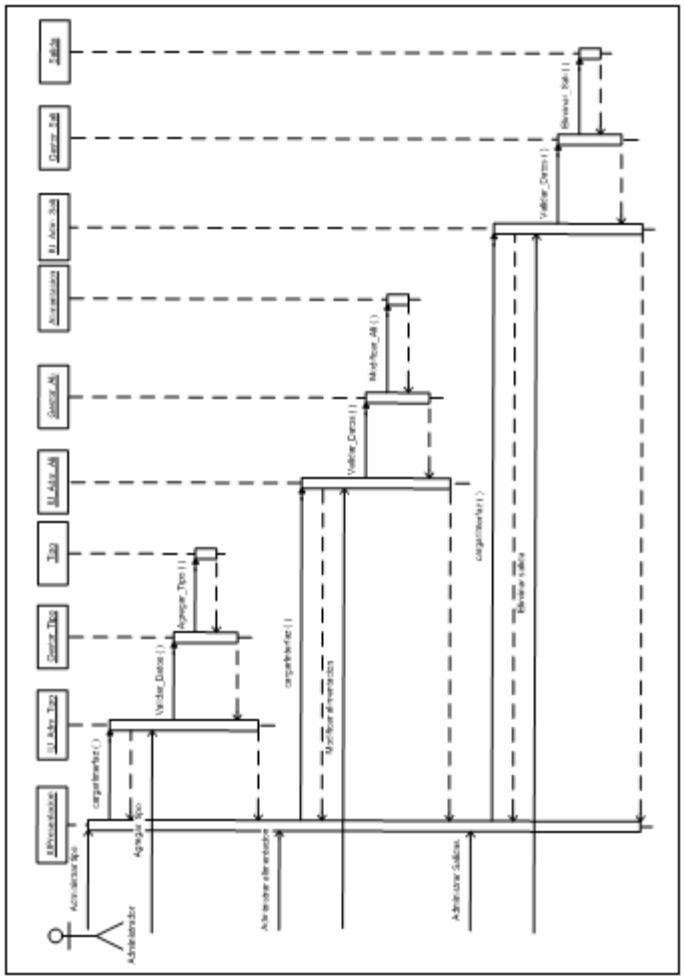

Figura 4.28. Diagrama de secuencia del caso de uso Administrar instrumentos 3/3
Fuente: Elaboración propia

4.4.3.- Diagrama de paquetes

El diagrama de paquetes sirve para encapsular las diferentes funcionalidades con que ha de contar el sistema.

En la figura 4.29 se ilustra el diagrama de paquetes para el subsistema Servidor TCP/IP-HART. Este subsistema presta servicios al Sistema MAI (Manejador de activos de instrumentación).

En la figura 4.30 se observa el diagrama de paquetes del sistema MAI, que está compuesto por los paquetes de seguridad, operaciones en línea y administración. En la figura 4.31 se describe con detalle el contenido del paquete administración, que está conformado por los paquetes usuarios, instrumentos y por último mantenimiento y configuración.

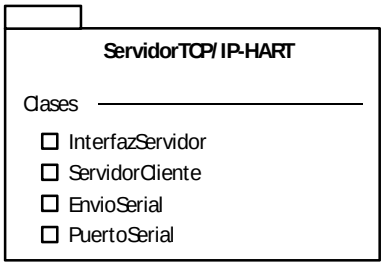

Figura 4.29. Diagrama de paquetes Servidor TCP/IP-HART

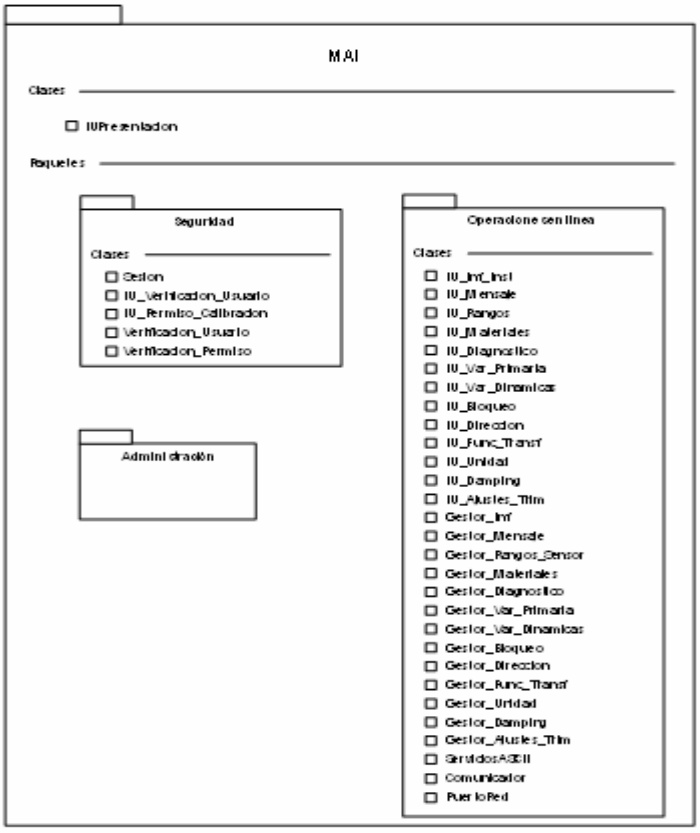

Figura 4.30. Diagrama de paquetes de diseño 1/2

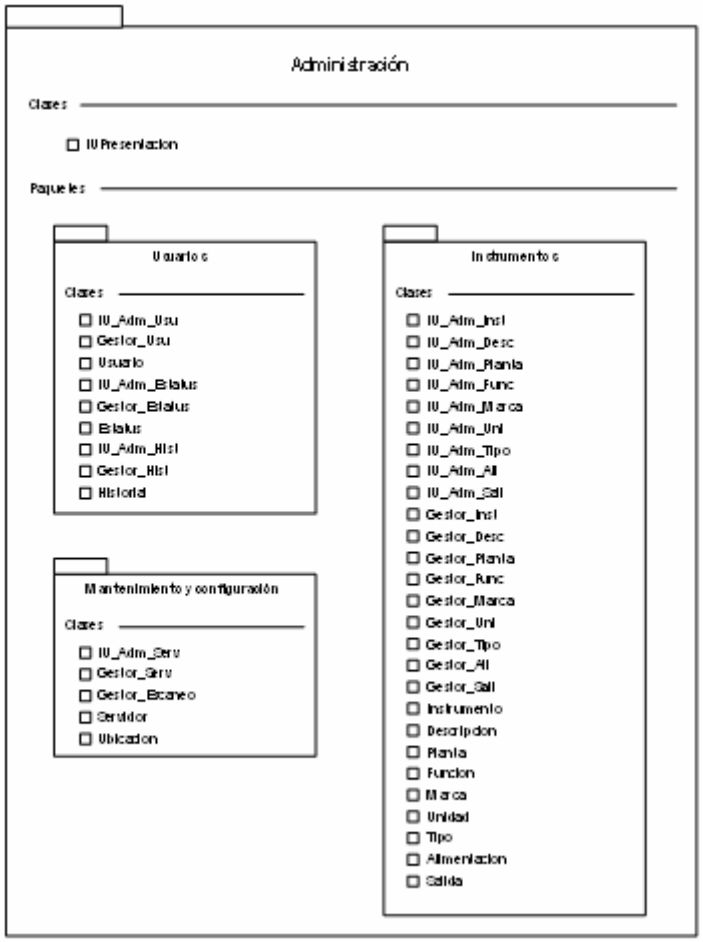

Figura 4.31. Diagrama de paquetes de diseño 2/2

Fuente: Elaboración propia

4.4.4.- Diagrama de capas

En el diagrama de capas de la figura 4.32 se muestra las dependencias y la distribución en capas de los subsistemas del diseño. El diagrama se subdivide en capas de acuerdo al nivel específico en que intervienen los diferentes subsistemas necesarios para MAI.

En la capa específica se encuentran los paquetes Mtto. Y configuración, usuarios e instrumentos, estos se interconectan al paquete de administración del

sistema ubicado en la capa de aplicaciones. También en la capa específica se encuentran los paquetes operaciones en línea y seguridad, ellos en conjunto con el paquete administración del sistema conforman al sistema MAI que se interconecta con el paquete PHP 5.

El paquete del servidor apache depende del servicio de interpretación del paquete PHP 5 como ya se menciono, es este a quien se interconecta todo el sistema MAI. A su vez, PHP 5 dependerá de los paquetes PostgreSQL 8.3, C++ y manejo de gráfica.

Por otra parte, el paquete Servidor TCP/IP-HART ubicado en la capa general se conecta al paquete C++ en la capa intermedia. Este último se conecta con el paquete de comunicación HART.

Finalmente se tiene la capa de software, formada por el protocolo TCP/IP y el sistema operativo del cual dependerán los paquetes C++, navegador web, Apache, PostgreSQL y PHP 5.

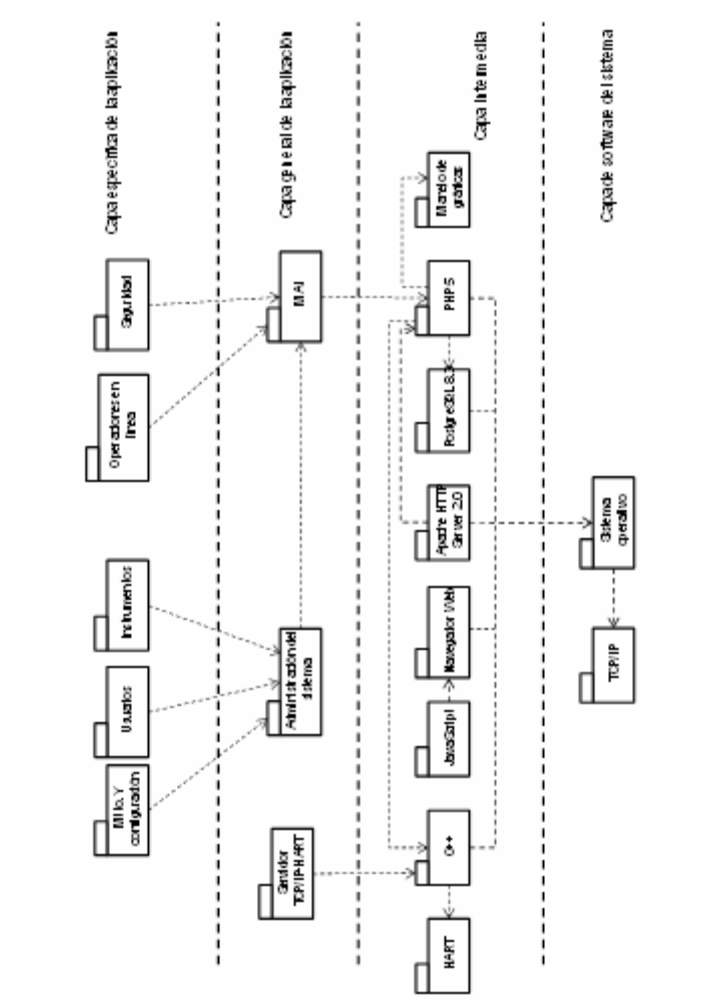

Figura 4.32.

Diagrama de capas del sistema

Fuente: Elaboración propia

4.4.5.- Prototipo de interfaz de usuario

El diseño de la interfaz de usuario crea un medio de comunicación efectiva entre un ser humano y una computadora. Siguiendo un conjunto de principios de diseños de interfaces, el diseñador identifica los objetos y las acciones de la interfaz, luego crea un formato de pantalla que forma la base de un prototipo de interfaz de usuario.

El diseño de la interfaz de usuario empieza con la identificación de los requisitos, la tarea y el ambiente. Una vez identificadas las tareas del usuario, se crean y analizan los escenarios de éste para definir un conjunto de objetos y acciones para la interfaz. Esto constituye la base para la creación de formatos de pantallas que representan el diseño gráfico y la ubicación de los iconos, la definición del texto descriptivo en pantalla, la especificación y asignación de nombres de las ventanas, además de la especificación de los elementos principales y secundarios de los menús. Se recurre a herramientas para crear prototipos y finalmente implementar el modelo de diseño, por último se evalúa la calidad del resultado.

Como primer paso para el diseño de la interfaz de usuario se creó un bosquejo con la distribución de los elementos que conformarán la interfaz, este bosquejo será la plantilla guía que se establecerá para todas las interfaces del sistema. En la figura 4.33 se muestra el bosquejo general del entorno gráfico del sistema. Este entorno gráfico está constituido en su parte superior por el encabezado y logo de la empresa, seguido del nombre del sistema. Se ha asignado un área a la izquierda de la interfaz para indicar el tag (Etiqueta de identificación) del instrumento seleccionado, es decir, el identificador del instrumento sobre el cual se va a realizar una operación. A la derecha una pequeña región para indicar el nombre del usuario que abrió la sesión. En el centro se ubicará el titulo de la interfaz y más abajo el área de trabajo donde se desplegará la información y herramientas que conformaran una interfaz en particular.

Por último a la izquierda se encontrará el menú del sistema para acceder a cualquier funcionalidad de él.

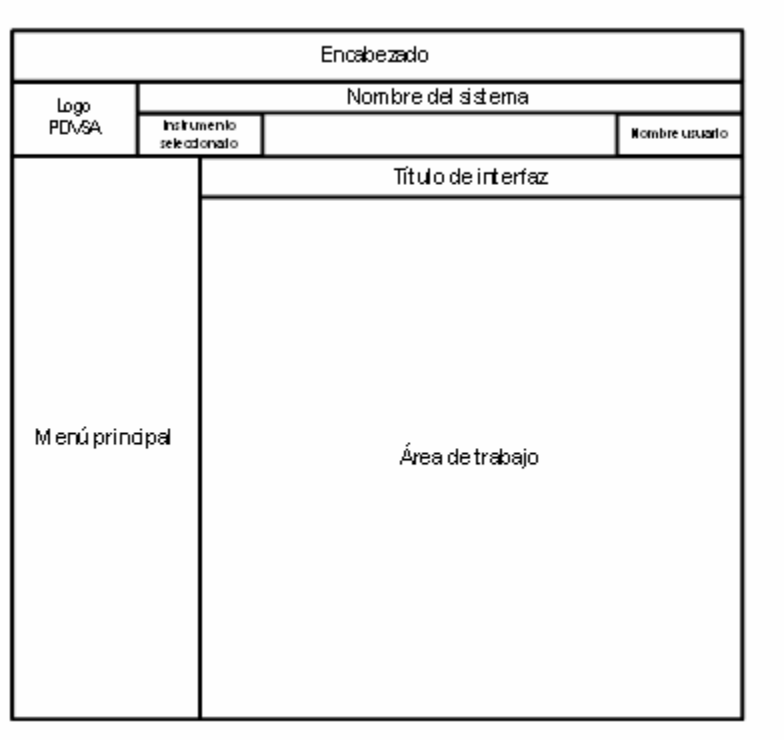

Figura 4.33 Bosquejo general del entorno gráfico del sistema

Fuente: Elaboración propia

4.4.6.- Diseño de la base de datos

Como se ha observado en los diversos diagramas expuestos en esta fase del proceso de desarrollo, es primordial la existencia de una base de datos, por ser necesaria para muchas de las actividades del sistema.

Para el diseño de la base de datos se han tomado los diagramas de análisis y los diagramas de clases como punto de partida para el diseño de las tablas y el manejo de información requerido por los subsistemas, así como la relación existente entre dichas tablas. El diseño de las mismas como indica la figura 4.34 se realizó mediante el modelo relacional.

4.4.6.1.- Tablas de la base de datos

- **Tabla "Instrumento":** contiene información detallada sobre la instrumentación instala en la refinería, esta tabla se relaciona con otras ocho (08) que poseen datos comunes entre los instrumentos.
- **Tabla "Descripción":** contiene información referente al tipo de instrumentación, es decir, la función para la que fue fabricado, por ejemplo: transmisor de temperatura, transmisor de presión, transmisor de flujo, etc. El indicador único (clave primaria) de cada registro en esta tabla es el modo abreviado de la descripción, por tanto, para los tipos de instrumentos mencionados la simbología sería, TT, PT y FT, indicando que es un transmisor con una tarea definida. Este código de descripción es utilizado para formar el tag de cada instrumento, en la figura 4.35 se observa el formato que utiliza la empresa para identificar la instrumentación de campo.

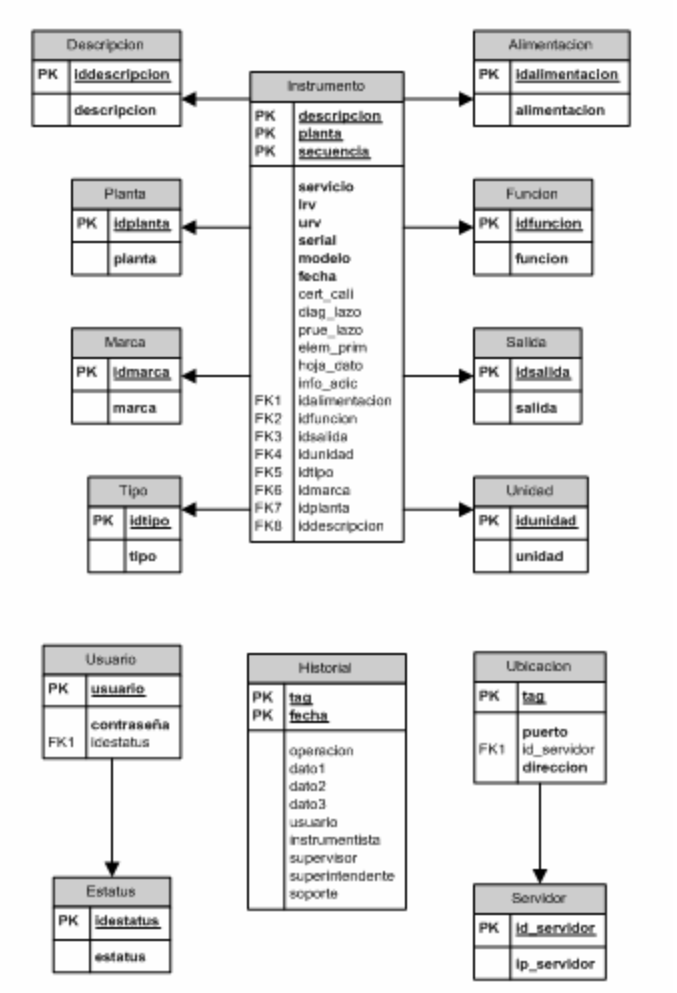

Figura 4.34 Modelo relacional de la base de datos del sistema MAI

Fuente: Elaboración propia

- **Tabla "Planta":** contiene las plantas que conforman la refinería y el código que la identifica, de manera que no existen dos plantas con un mismo código, sin embargo, una misma planta puede poseer diferentes códigos, eso se debe a que en plantas relativamente grandes el código no indica simplemente la planta, sino también, indica el sector de la planta. Estos códigos son utilizados

para formar el tag de la instrumentación de campo y, así indicar en qué planta y sector ha sido instalada (ver figura 4.35).

- **Tabla “Marca”:** almacena los fabricantes de la instrumentación de campo.
- **Tabla “Tipo”:** esta tabla registra información referente al principio utilizado en la fabricación del instrumento para realizar la medición correspondiente al equipo. Esta información es suministrada por el fabricante.
- **Tabla “Función”:** contiene información sobre la función que cumple un instrumento instalado en planta, es decir, un instrumento podría ser utilizado simplemente para monitoreo de un proceso, siendo así un indicador, por otra parte podría operar como un instrumento de control, que según su medición sea configurado para regular algún elemento que incida en el proceso en un momento determinado.
- **Tabla “Alimentación”:** contiene información proporcionada por el fabricante sobre el voltaje de alimentación del instrumento.
- **Tabla “Salida”:** registra información referente al tipo de salida que utilizan los instrumentos para expresar la medición que realizan, pudiendo ser salidas en rangos de corriente o algún protocolo de comunicación determinado.
- **Tabla “Unidad”:** contiene información sobre la unidad de ingeniería en la que opera un instrumento. La unidad de medida va a ser definida en la calibración del instrumento, acorde al tipo de instrumento.
- **Tabla “Usuario”:** utilizada para el acceso al sistema como mecanismo de seguridad y privacidad.
- **Tabla “Estatus”:** maneja los estatus de los usuarios, es utilizado para discriminar el nivel de privilegios del que disponen los usuarios en el sistema.

- **Tabla “Servidor”:** utilizada para registrar las direcciones IP de los servidores TCP/IP-HART que prestan servicio al sistema permitiendo el acceso remoto a los instrumentos.
- **Tabla “Ubicación**”: contiene la información necesaria para conectarse a un instrumento a través de la red.
- **Tabla “Historial”:** contiene registro de todas las operaciones de calibración realizadas a través del sistema sobre los instrumentos de campo.

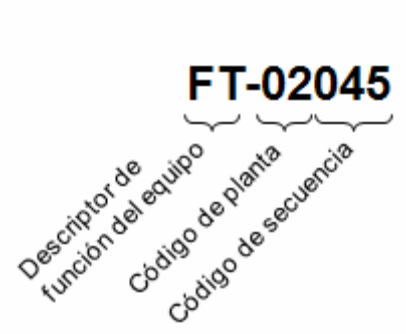

Figura 4.35 Descripción del formato del tag de la instrumentación

Fuente: Elaboración propia

4.5.- Implementación

4.5.1.- Implementación de la arquitectura

Los diagramas de despliegue muestran la configuración en funcionamiento del sistema, incluyendo su hardware y su software. Estos diagramas exponen las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

El diagrama de despliegue del sistema mostrado en la figura 4.36 describe los componentes de hardware y software que conforman el sistema. El nodo estación de trabajo representa cualquier computador conectado a la intranet PDVSA en la que a través del navegador web los usuarios finales podrán conectarse al sistema alojado en el nodo servidor web. El nodo servidor web tiene como componentes la interfaz MAI (manejador de activos de instrumentación) que es la interfaz de usuario del sistema, este se interconecta al manejador de base de datos para consultas y administración del sistema, el componente manejo de gráficas genera las gráficas de comportamiento en línea de las variables de proceso y el ejecutable manejador de activos a través del cual se conecta al nodo servidor TCP/IP-HART, de este último pueden existir numerosos nodos a lo largo de la intranet que permitan el acceso remoto a los instrumentos, de igual manera, al nodo servidor TCP/IP-HART pueden estar conectados varios instrumentos, por esta razón cuenta con un componente que multiplexa dirigiendo el mensaje a un instrumento en particular.

4.6.- Conclusión de la fase de elaboración

En esta fase se ha dado un gran avance en el proceso de desarrollo, en el sentido de que se han diseñado las clases que forman el sistema y se ha establecido el comportamiento que este debe tener para casos tanto generales como puntuales. Por otro lado, se han definido responsabilidades ha componentes particulares que le dan forma a la estructura del sistema y, estos componentes se han organizado según el nivel específico en que intervienen.

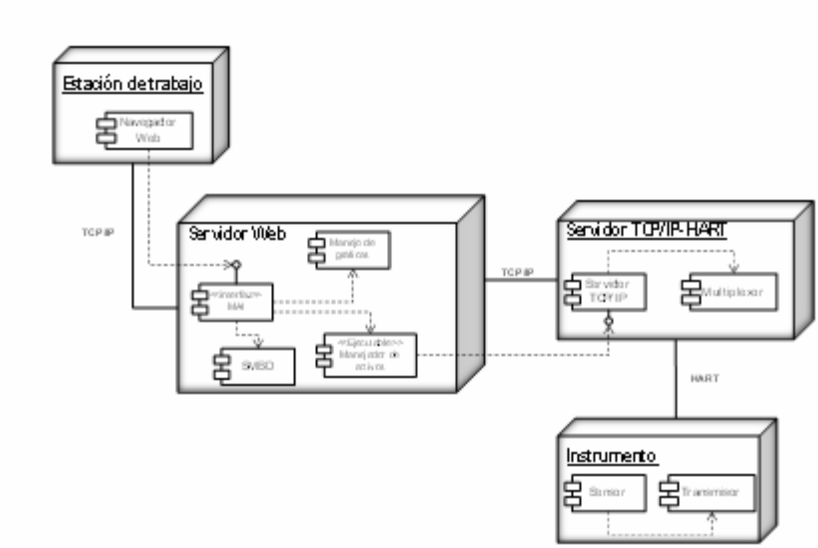

Figura 4.36. Diagrama de despliegue del sistema

Fuente: Elaboración propia

Se ha realizado el diseño de la base de datos. A su vez, se introduce el diseño esquemático de las interfaces.

Por último se realizó el diagrama de despliegues del sistema, el cual facilita la comprensión de cómo están distribuidos los nodos y qué componentes residirán en ellos.

CAPITULO 5
FASE DE CONSTRUCCIÓN

5.1.- Introducción

Una vez obtenidos los diagramas de análisis y diseño de la fase anterior, se procederá a codificar el software en pequeños módulos realizando pruebas en cada uno de ellos, así paulatinamente, integrar los módulos y hacer pruebas de integración. Todo este proceso de codificación, integración y pruebas se hará de manera incremental a través de iteraciones sucesivas con la finalidad de obtener un producto funcional de calidad.

En esta fase de desarrollo es esencial la escogencia de los lenguajes de programación con los que se codificará el sistema, la diversas pruebas que se realizarán y la metodología más eficiente para lograr la integración de todos los módulos desarrollados.

En la figura 5.1 muestran los flujos de trabajo aplicados en la fase de construcción.

5.2.- Herramientas de desarrollo escogidas

El Sistema Manejador de Activos de Instrumentación (MAI) va ha ser implementado por un ente gubernamental como lo es PDVSA (Petróleos de Venezuela) en su filial PDVSA Refinación Oriente, por esto para la selección de las herramientas de desarrollo este sistema requiere de un lenguaje de programación con licencia de software libre debido al decreto presidencial 3390. El sistema será implementado dentro de la intranet de PDVSA, porque de manera rápida les permite a los usuarios el acceso al sistema.

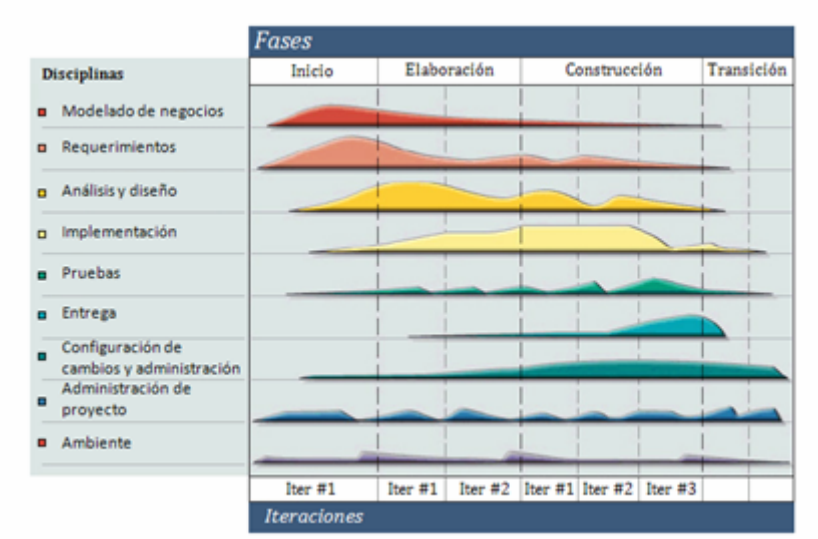

Figura 5.1. Fases del proceso unificado

Fuente: Jacobson, I, Booch, G, Rumbaugh J, 2000

A continuación se describirán las herramientas y lenguajes de programación seleccionados para la ejecución de este proyecto:

5.2.1.- Lenguaje de programación de aplicaciones Web: PHP

Es un lenguaje orientado a objetos, utilizado para la generación de páginas web dinámicas y está basado en herramientas con licencia de software libre. Además, PHP ofrece otras ventajas ya que es un lenguaje interpretado en el lado del servidor, lo que permite acceder a los recursos que tenga el servidor, como por ejemplo las bases de datos; es independiente del navegador, es decir no es necesario que su navegador lo soporte. Otra gran ventaja de PHP es la integración con los sistemas de bases de datos y el soporte nativo a las distintas bases de datos existentes, libres y comerciales.

Ventajas de PHP

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la web.
- Biblioteca nativa de funciones sumamente amplia e incluída.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

A continuación se muestra el logo del Lenguaje de Programación PHP:

Figura 5.2. Logo del leguaje de programación PHP

Fuente: www.php.net

5.2.2.- HTML (Lenguaje De Etiquetas Por Hipertexto)

El lenguaje HTML (*hypertext mark-up language*). Se trata de un lenguaje de marcas (se utiliza insertando marcas en el interior del texto) que nos permite representar de forma rica el contenido y también referenciar otros recursos (imágenes, etc.), enlaces a otros documentos (la característica más destacada del WWW), mostrar formularios para posteriormente procesarlos, etc.

5.2.3.- JavaScript

JavaScript es un lenguaje para la creación de scripts que se usa con páginas HTML o PHP para incrementar la funcionalidad y la interacción con el usuario final. Fue desarrollado por Netscape con el lenguaje Java de Sun. Es importante resaltar que JavaScript en cierta forma es parecido a Java, debido a que este último es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems y que necesita una variedad de compiladores y archivos de soporte para funcionar. Este lenguaje contiene un conjunto de comandos más reducido y más sencillo, que varían ligeramente en su implementación. La sintaxis y estructura de JavaScript son similares a las de Java, pero JavaScript sólo es funcional cuando se incluye como parte de una página HTML o PHP.

JavaScript es un nuevo lenguaje escrito. Los 'scripts' de Javascript pueden ser introducidos dentro de las páginas de HTML. Con Javascript se puede dar respuesta a eventos iniciados por el usuario, eventos tales como la entrada de una forma o algún enlace. Esto sucede sin ningún tipo de transmisión, de tal forma que cuando un usuario escribe algo en una forma, no es necesario que sea transmitido hacia el servidor, verificado y devuelto. Las entradas son verificadas por la aplicación cliente y pueden ser transmitidas después de esto. También se puede pensar de programa que se ejecuta en la versión cliente.

5.2.4.- Sistema manejador de base de datos PostgreSQL

PostgreSQL es un servidor de base de datos relacional libre, liberado bajo la licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle o DB2. Algunas de sus principales características son:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.
- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*).
- Disparadores (*triggers*).
- Vistas.
- Integridad transaccional.

- Herencia de tablas.

Funciones

Bloques de código que se ejecutan en el servidor, pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional. Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos. A continuación se muestra el logo del Sistema Manejador de Bases de Datos PostgreSQL:

Figura 5.3. Logo del Sistema manejador de base de datos PostgreSQL

Fuente: www.postgresql.org

5.2.5.- Librería SolarSockets

SolarSockets es una librería escrita en C++, diseñada para facilitar la programación de aplicaciones que requieran comunicaciones TCP, SolarSockets está orientada a eventos, hace uso de hilos (threads) y es multiplataforma, con lo que se pueden desarrollar aplicaciones para sistemas operativos distintos, sin que para hacerlos funcionar se tenga que escribir doble código, ya que SolarSockets tiene la misma

interface y comportamiento, sin importar el sistema operativo o compilador de C++ que se utilice.

5.3.- Interfaces del sistema

En esta sección se han diseñado todas las interfaces restantes del sistema. Que permitirán a los usuarios tener acceso desde sus navegadores web al manejador de activos, donde podrán monitorear y calibrar de forma remota a los instrumentos que operan a lo largo de las plantas que conforman la refinería de Puerto La Cruz. A través de estas interfaces los usuarios podrán navegar por todo el sistema, y todas las operaciones que este brinda.

La primera interfaz es la de validación de usuario, que solicita la identificación de nombre de usuario y contraseña. Todo esto para garantizar la seguridad del sistema permitiendo solo a personas autorizadas acceder al sistema. También según sea el estatus del usuario (que ha sido asignado a él por el administrador), se le darán atribuciones sobre el sistema.

En la figura 5.4 se observa la interfaz de inicio del sistema, donde se realiza la autentificación de los usuarios. Se solicita ingresar el identificador de usuario y contraseña, estos datos son corroborados en la base de datos para validar la información, de ser acertada se le concede el acceso al sistema de acuerdo a los privilegios del usuario. Si los datos son incorrectos se deniega el acceso (figura 5.5).

La interfaz principal presenta la barra de menú que contiene los principales procesos del sistema. A través de este menú se podrán realizar todas las operaciones de consulta y calibración sobre los instrumentos, así como también, acceder a información de referencia sobre la instrumentación. Las categorías principales del

sistema son: instrumentos en línea, información técnica de instrumentos y administración. La figura 5.6 ilustra la interfaz principal del sistema señalando cada una de las partes que la componen.

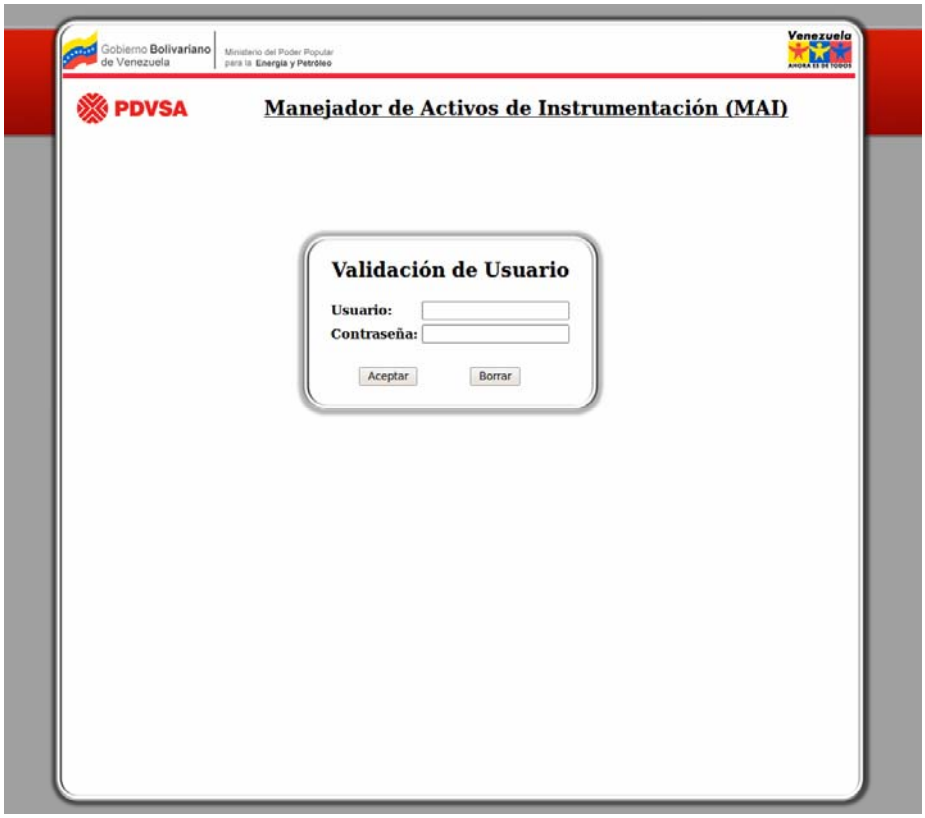

Figura 5.4. Interfaz de inicio

Fuente: Elaboración propia

Figura 5.5. Usuario inválido

Fuente: Elaboración propia

Al seleccionar la opción del menú “Instrumentos en línea”, como se muestra en la figura 5.7, se despliega un submenú con todas las operaciones para monitoreo y calibración de la instrumentación y, se presenta en el área de contenido la interfaz correspondiente a los instrumentos en línea, en dicha área se encuentran los tag de los instrumentos que están conectados al sistema, encima un conjunto de opciones de búsqueda para clasificar o filtrar el listado de los instrumentos mostrados. Para poder ejecutar cualquiera de las operaciones del submenú de Instrumentos en línea, se debe seleccionar previamente el tag del instrumento sobre el cual se desee realizar la operación. Una vez seleccionado un tag este se resaltará en letras rojas debajo del título del sistema, así se sabrá en todo momento sobre que instrumento se están realizando operaciones.

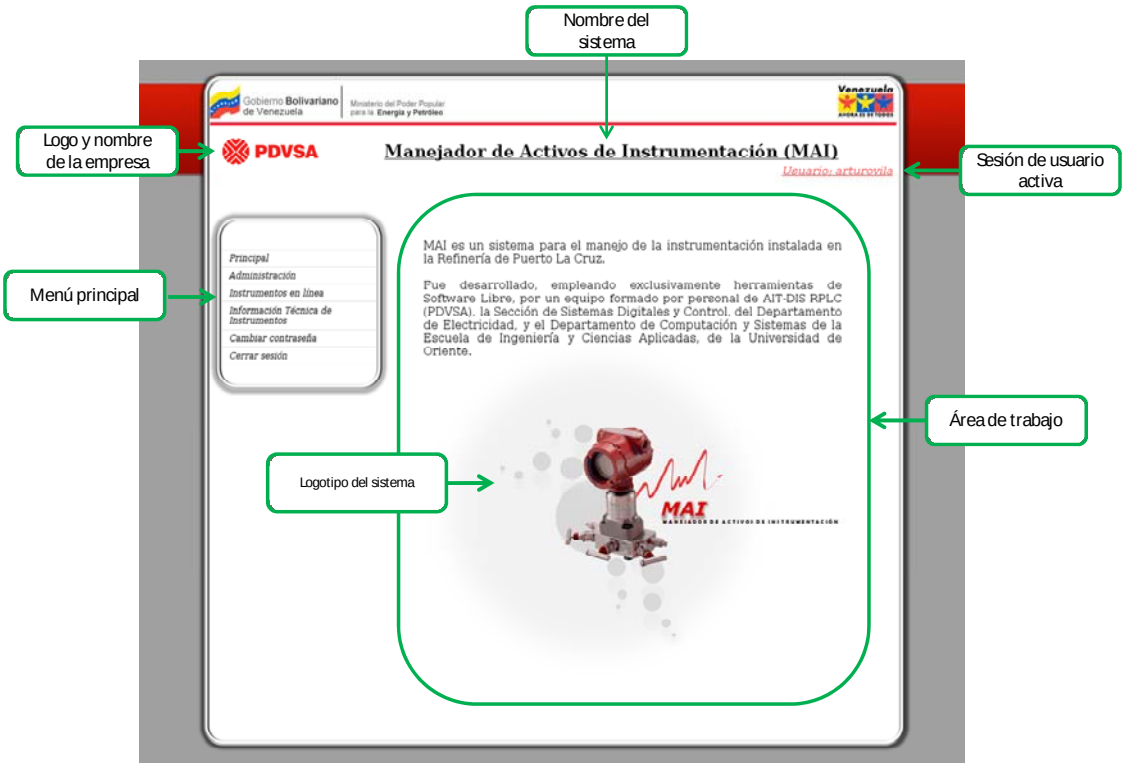

Figura 5.6. Interfaz Principal

Fuente: Elaboración propia

Figura 5.7. Interfaz Instrumentación en línea

Fuente: Elaboración propia

En la figura 5.8 se muestra la interfaz de variables dinámicas donde se monitorea el comportamiento de las variables a través del tiempo, esta interfaz está formada por cuatro gráficos como son: corriente, variable primaria (también llamada variable de proceso), variable secundaria y variable terciaria. Cada gráfico representa el comportamiento de ciertos parámetros del instrumento, en la figura 5.8 se ve que la operación se realiza sobre el instrumento de tag FT-02727, donde FT nos indica que es un transmisor de flujo. El primer gráfico muestra la salida de corriente del instrumento representada en mA, esta señal que varía entre 4-20 mA. es la que va conectada a la entrada del DCS de la refinería. El gráfico verde (variable primaria) representa el comportamiento en el sensor de presión del instrumento, esta salida se muestra en la unidad seleccionada en la configuración, los dos últimos gráficos son variables internas del dispositivo de campo, el gráfico azul representa la temperatura

interna del instrumento y el rojo la presión interna (no está relacionado al proceso aplicado).

En las figuras 5.9, 5.10 y 5.11 se muestran las interfaces de algunas de las operaciones de monitoreo y consulta de información técnica que ofrece el sistema MAI. En la figura 5.12 se presentan algunos de los mensajes de advertencia con sugerencias de procedimientos que se deben cumplir durante el proceso de calibración de un instrumento.

Igualmente se diseñaron interfaces de administración del sistema, como administración de instrumentos (figura 5.13), administración de usuarios y estatus de los usuarios (figura 5.14 y 5.15 respectivamente). Así también, la interfaz de historial (figura 5.16), que muestra cuándo y qué usuarios han realizado cuales operaciones de calibración sobre qué instrumento.

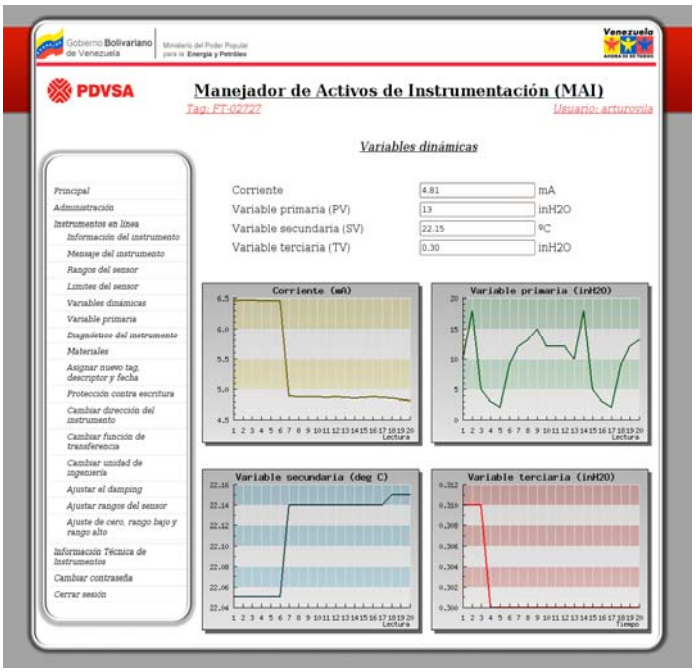

Figura 5.8. Interfaz de monitoreo de variables dinámicas

Fuente: Elaboración propia

Figura 5.9. IU de ejecución de operación "Cambiar unidad de ingeniería"

Fuente: Elaboración propia

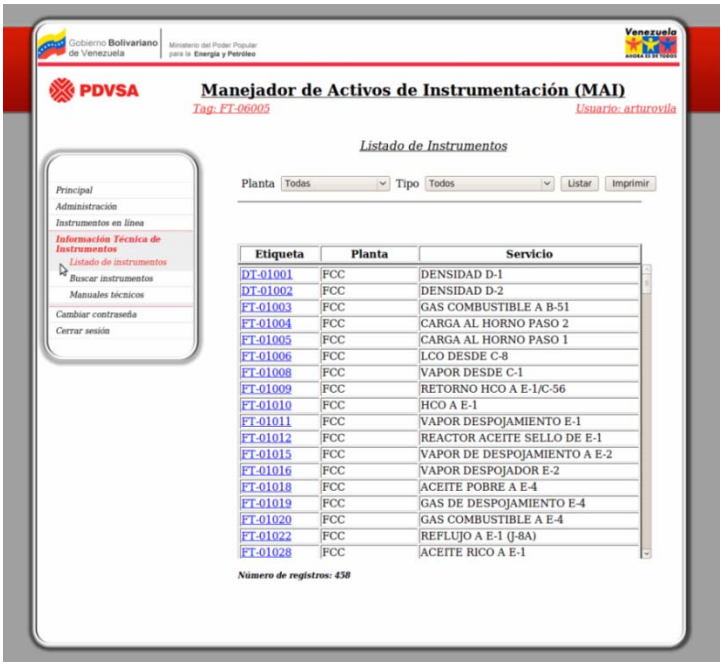

Figura 5.10. Interfaz de listado de instrumentos

Fuente: Elaboración propia

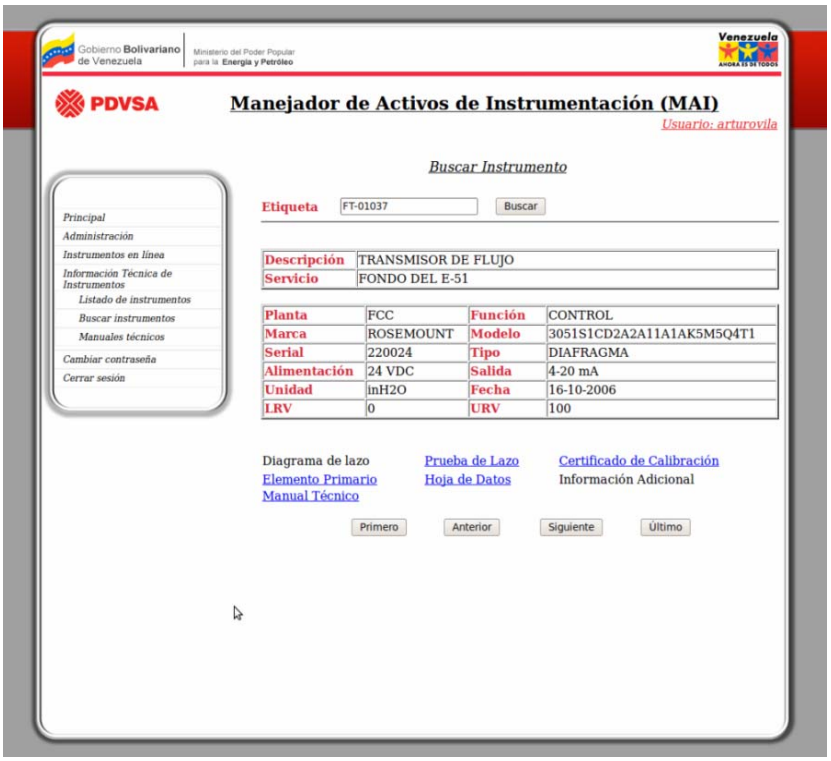

Figura 5.11. Interfaz de información técnica de instrumento

Fuente: Elaboración propia

Figura 5.12. Mensajes de advertencias para operaciones calibración

Fuente: Elaboración propia

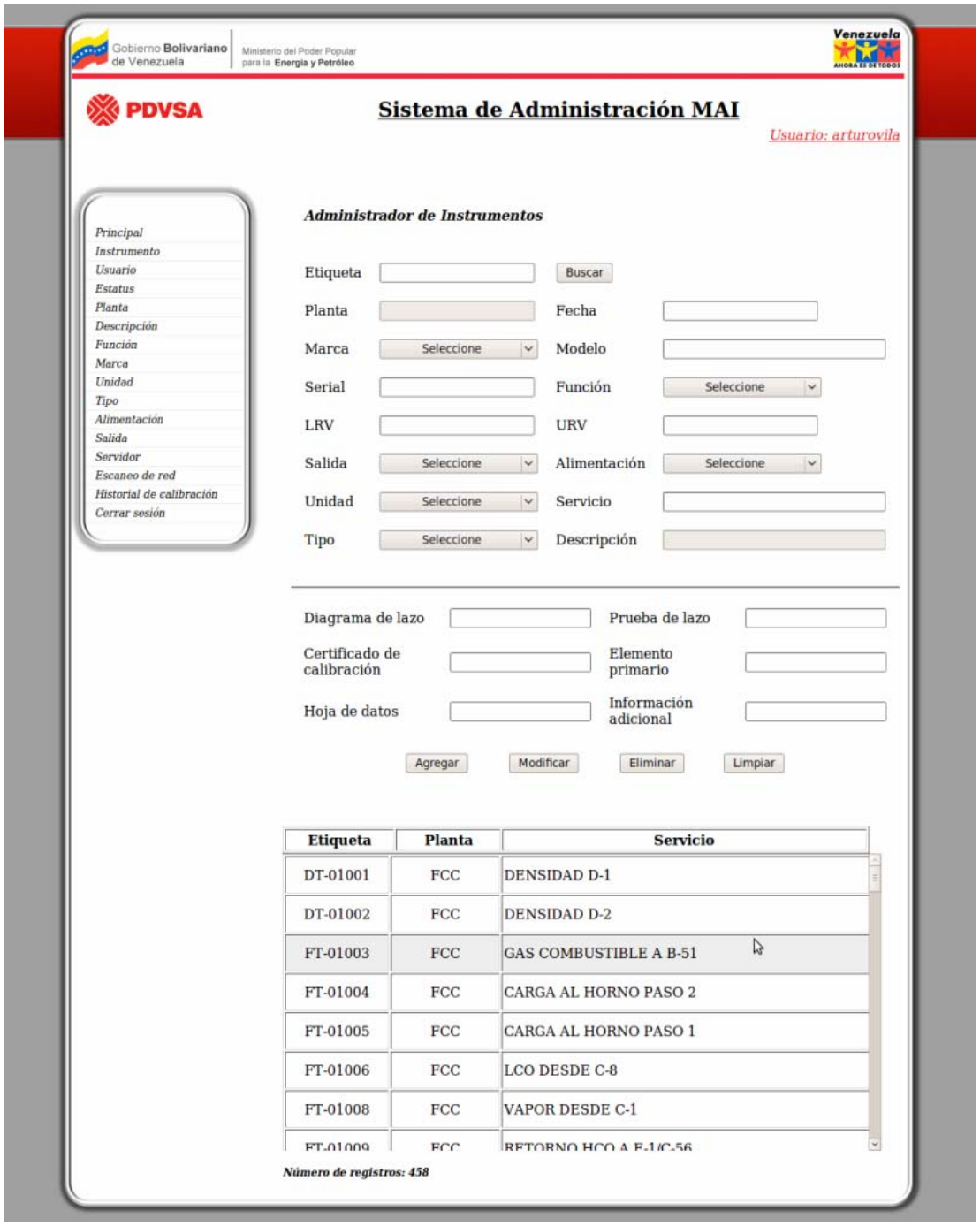

Figura 5.13. Interfaz del subsistema de administración MAI "Administrador de instrumentos"

Fuente: Elaboración propia

Figura 5.14. IU del subsistema de administración "Administrador de Usuarios"

Fuente: Elaboración propia

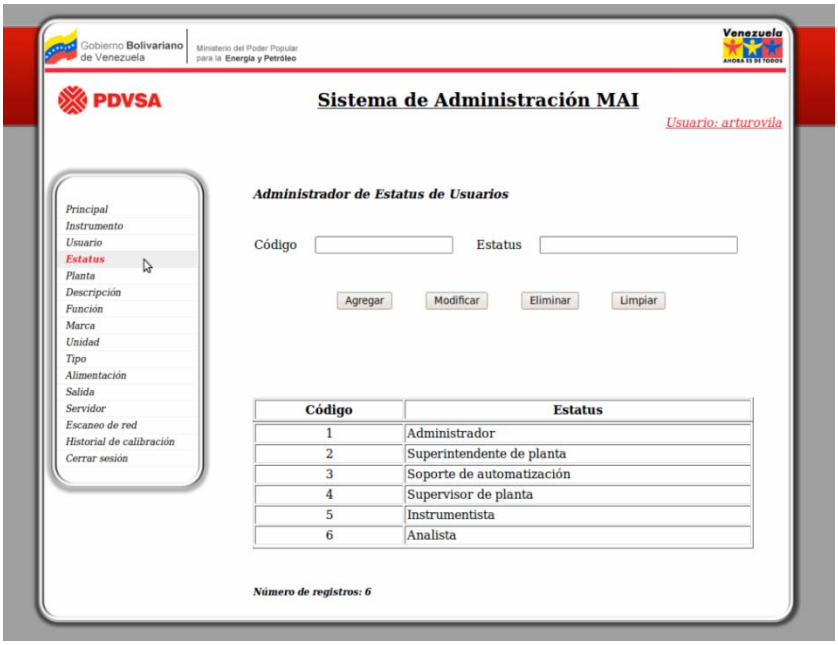

Figura 5.15. IU del subsistema de administración "Administrador de Estatus"

Fuente: Elaboración propia

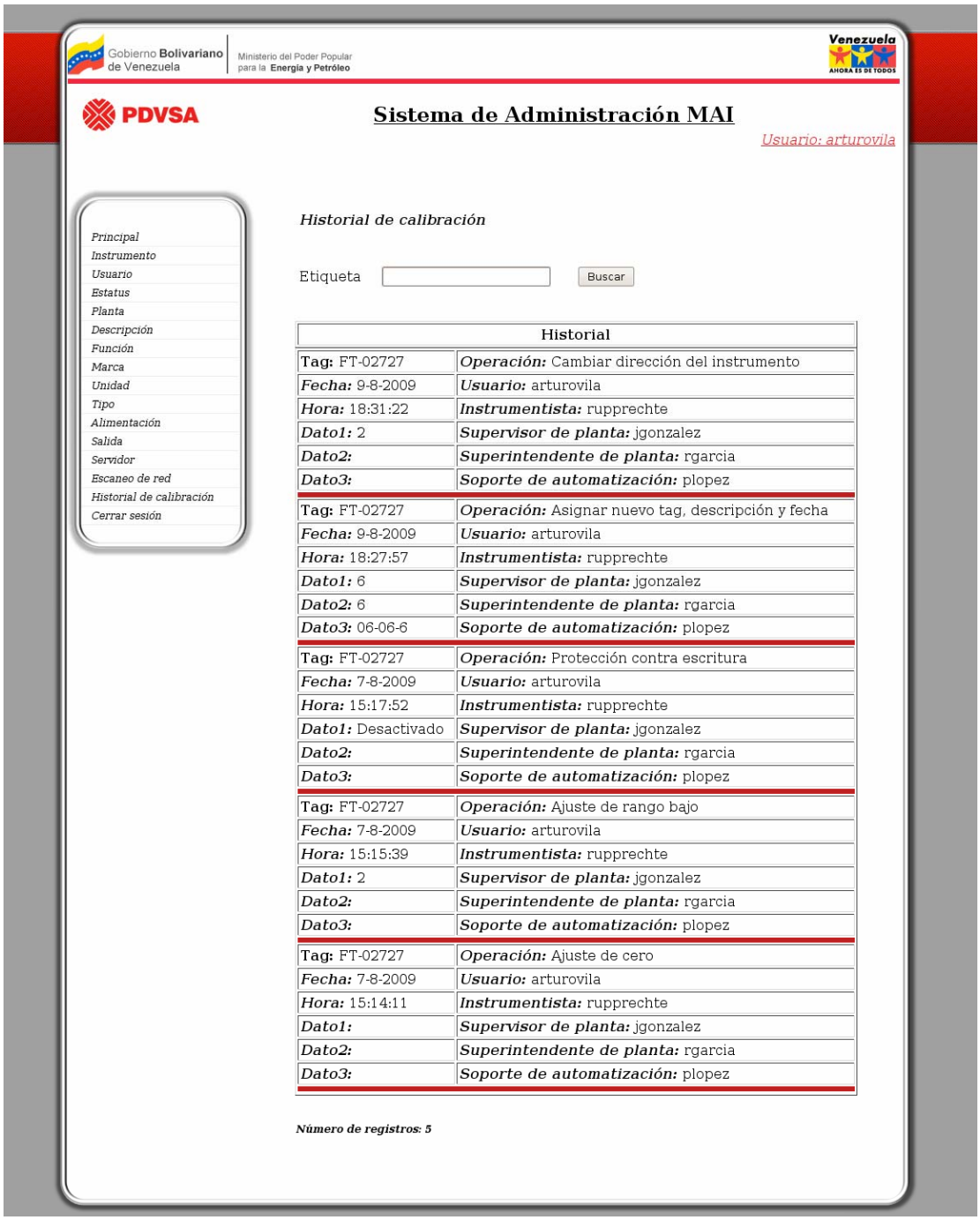

Figura 5.16. Interfaz del subsistema de administración MAI "Historial de Calibración"

Fuente: Elaboración propia

5.4.- Implementación

Durante la implementación se codificó la totalidad de secciones diseñadas para el sistema, sin embargo, por la gran cantidad de secciones y la extensión en páginas que se necesitaría se presentarán sólo algunas de las clases más relevantes.

5.4.1.- Implementación de funcionalidades para monitoreo de instrumentación

En la figura 5.17 se muestra el diagrama de componentes para una de las operaciones perteneciente al subsistema de operaciones en línea. En este diagrama se exhiben los componentes principales que participan en la operación de consulta de la información básica de los instrumentos. A continuación se presenta la implementación de una de las clases más importantes en este diagrama de componentes.

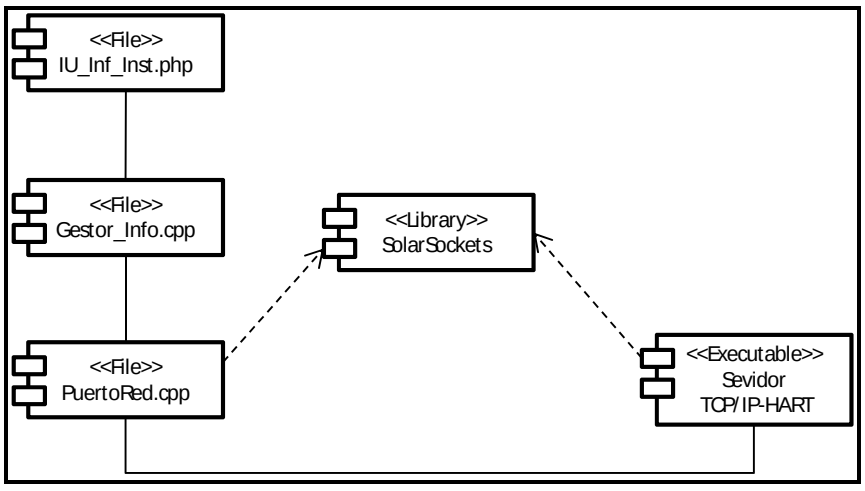

Figura 5.17. Diagrama de componentes para monitoreo de instrumentación

Fuente: Elaboración propia

La clase Gestor_Info tiene una función de clase controladora, en ella se construye las tramas HART para la consulta y asignación de los datos básicos de un instrumento. La clase igualmente implementa la interpretación de la trama de respuesta generada por un instrumento producto de una solicitud previa. A continuación se presenta el código de la clase Gestor_Info.

```
#ifndef GESTOR_INFO_H_
#define GESTOR_INFO_H_
#include "PuertoRed.h"
#include "ServicioASCII.h"
#include "Comunicador.h"

class Gestor_Info
{
public:
	Gestor_Info();
	virtual ~Gestor_Info();
	void LeerDatosInstrumentos(int IdServidor, char IpServidor[], int puerto, char Direccion []);
	void AsignarNuevo(int IdServidor, char IpServidor[], int puerto, char Direccion [], char nTag[], char nDescripcion[], int ndia, int nmes, int nano);

private:
	unsigned char checksum;
	ServicioASCII A;
	Comunicador C;
	PuertoRed TCP;
	int dia, mes, ano, Error;
	char HART[14], respuesta[255], mensaje[60], Tag [9], Descripcion [17], PackTag[6], PackDescripcion[12], MensajeError[100];
};

#endif /*GESTOR_INFO_H_*/



#include "Gestor_Info.h"

Gestor_Info::Gestor_Info()
```

```
{
}

Gestor_Info::~Gestor_Info()
{
}

void Gestor_Info::LeerDatosInstrumentos(int IdServidor, char IpServidor[], int puerto, char Direccion [])
{
	Error=-1;
	int byte=0;
	int datos [7];
	char modelo[15], fabricante[20];

//----------------------------------ENVIO TCP/IP ------------------------------

	checksum = (0x82 ^ Direccion[0] ^ Direccion[1] ^ Direccion[2] ^ Direccion[3] ^ Direccion[4] ^ 0x00);

	HART [0] = IdServidor;
	HART [1] = puerto;
	HART [2] = 13;
	HART [3] = 0xff;		//Preambulo
	HART [4] = 0xff;		//Preambulo
	HART [5] = 0xff;		//Preambulo
	HART [6] = 0xff;		//Preambulo
	HART [7] = 0x82;		//Byte de inicio Maestro (Formato largo)
	HART [8] = Direccion[0];	// Direccion del instrumento (Formato largo)
	HART [9] = Direccion[1];
	HART [10] = Direccion[2];
	HART [11] = Direccion[3];
	HART [12] = Direccion[4];
	HART [13] = 0x00;		// Comando Universal 00
	HART [14] = 0x00;		// Byte count
	HART [15] = checksum;		// Suma de verificacion


	char* PRespuesta1 = TCP.EnviarHART(HART);

	strcpy(respuesta, PRespuesta1);
	respuesta[50]='\0';
```

```
// ------------------------- RECEPCION DE RESPUESTA 1 -------------------------

	for(int i=0; i<25; i++)
		if((unsigned char)respuesta[i] == 0x88)
			respuesta[i] = 0x00;

	//Conexion Fallida, no se pudo conectar con el servidor
	if((unsigned char)respuesta[0]==0xCC && (unsigned char)respuesta[1]==0xCC && (unsigned char)respuesta[2]==0xCC && (unsigned char)respuesta[3]==0xCC && (unsigned char)respuesta[4]==0xCC && (unsigned char)respuesta[5]==0xCC)
	{
		strcpy(MensajeError, "Error, no se pudo conectar al servidor");
		Error=1;
	}

	//Servidor encontrado pero Instrumento no respondio a solicitud
	else if((unsigned char)respuesta[0]==0xAA && (unsigned char)respuesta[1]==0xBB && (unsigned char)respuesta[2]==0xAA && (unsigned char)respuesta[3]==0xBB && (unsigned char)respuesta[4]==0xAA)
	{
		strcpy(MensajeError, "Error, Instrumento no respondiÃ³ a solicitud, verificar si estÃ¡ conectado");
		Error=2;
	}

	else if((unsigned char)respuesta[12]==0x20)
	{
		Error=4;
		sprintf(MensajeError, "Error, en cÃ³digo de respuesta, Intentelo nuevamente, codigo: %X", (unsigned char)respuesta[12] );
	}

	//Solicitud satisfactoria, Identificando Instrumento
	else if( (unsigned char)respuesta[0]==0xff && (unsigned char)respuesta[1]==0xff && (unsigned char)respuesta[2]==0xff && (unsigned char)respuesta[3]==0xff)
		Error=0;


	//Servidor encontrado pero Instrumento dio respuesta no esperada
	else
	{
```

```
        strcpy(MensajeError, "Error desconocido");
        Error=3;
}


if(Error==0)
{
        byte = 0;
        while((unsigned char)respuesta[byte]!= 0x86)
        byte++;

        // Se guarda cada dato obtenido del comando 00 de los bytes (10-17)
en arreglo datos
        for(int i=0; i<8; i++)
                datos[i]= (unsigned char)respuesta[byte+11+i];


                        switch (datos[0])
                        {
                                case 38:
                                        strcpy(fabricante, "ROSEMOUNT");
                                        break;
                                default:
                                        strcpy(fabricante, "GENERICO");
                                        break;
                        }

                        switch (datos[1])
                        {
                                case 6:
                                        strcpy(modelo, "3051");
                                        break;

                                case 14:
                                        strcpy(modelo, "3001");
                                        break;

                                case 35:
                                        strcpy(modelo, "2088");
                                        break;

                                case 39:
                                        strcpy(modelo, "2090");
```

```
                                        break;

                                default:
                                        strcpy(modelo, "DESCONOCIDO");
                                        break;
                        }

//-----------------------------------ENVIO 2 TCP/IP ------------------------------

                checksum = (0x82 ^ Direccion[0] ^ Direccion[1] ^ Direccion[2] ^
Direccion[3] ^ Direccion[4] ^ 0x0d);

                HART [0] = IdServidor;
                HART [1] = puerto;
                HART [2] = 13;
                HART [3] = 0xff;
                HART [4] = 0xff;
                HART [5] = 0xff;
                HART [6] = 0xff;
                HART [7] = 0x82;
                HART [8] = Direccion[0];
                HART [9] = Direccion[1];
                HART [10] = Direccion[2];
                HART [11] = Direccion[3];
                HART [12] = Direccion[4];
                HART [13] = 0x0d;
                HART [14] = 0x00;
                HART [15] = checksum;

                char* PRespuesta2 = TCP.EnviarHART(HART);

                strcpy(respuesta, PRespuesta2);
                respuesta[50]='\0';

//----------------------------------- RESPUESTA 2 ----------------------------------
                for(int i=0; i<35; i++)
                        if((unsigned char)respuesta[i] == 0x88)
                                respuesta[i] = 0x00;

                //Conexion Fallida, no se pudo conectar con el servidor
                if((unsigned char)respuesta[0]==0xCC && (unsigned
char)respuesta[1]==0xCC && (unsigned char)respuesta[2]==0xCC && (unsigned
char)respuesta[3]==0xCC && (unsigned char)respuesta[4]==0xCC && (unsigned
```

```
char)respuesta[5]==0xCC)
		{
			strcpy(MensajeError, "Error, No se pudo conectar al servidor");
			Error=1;
		}

		//Servidor encontrado pero Instrumento no respondio a solicitud
		else if((unsigned char)respuesta[0]==0xAA && (unsigned char)respuesta[1]==0xBB && (unsigned char)respuesta[2]==0xAA && (unsigned char)respuesta[3]==0xBB && (unsigned char)respuesta[4]==0xAA)
		{
			strcpy(MensajeError, "Error, Instrumento no respondiÃ³ a solicitud, verificar si estÃ¡ conectado");
			Error=2;
		}

		else if((unsigned char)respuesta[12]==0x20)
		{
			Error=4;
			sprintf(MensajeError, "Error, en cÃ³digo de respuesta, Intentelo nuevamente,  codigo:  %X", (unsigned char)respuesta[12] );
		}

		//Solicitud satisfactoria, Identificando Instrumento
		else if( (unsigned char)respuesta[0]==0xff && (unsigned char)respuesta[1]==0xff && (unsigned char)respuesta[2]==0xff && (unsigned char)respuesta[3]==0xff)
			Error=0;


		//Servidor encontrado pero Instrumento dio respuesta no esperada
		else
		{
			strcpy(MensajeError, "Error desconocido");
			Error=3;
		}

		if(Error==0)
		{

			byte=0;
			while((unsigned char) respuesta[byte]!= 0x86)
				byte++;
```

```
                        //Obteniendo Tag en paquete ASCII
                        char tagPack[7];

                        for(int i=0; i<6; i++)
                                tagPack[i] = (unsigned char) respuesta[byte+10+i];
                        tagPack[6]='\0';


                        //Desempaquetando Tag
                        char* PTag=A.DesempaquetarASCII(tagPack);
                        strcpy(Tag, PTag);
                        Tag[8]='\0';

                        //Obteniendo Descripcion en paquete ASCII
                        char DescripcionPack[13];
                        for(int i=0; i<12; i++)
                                DescripcionPack[i]      =       (unsigned       char)
respuesta[byte+16+i];
                        DescripcionPack[12]='\0';

                        //Desempaquetando Descripcion
                        char* PDesc=A.DesempaquetarASCII(DescripcionPack);
                        strcpy(Descripcion, PDesc);
                        Descripcion[16]='\0';

                        //Obteniendo Fecha
                        dia= (int) ( (unsigned char) respuesta[byte+28] );
                        mes= (int) ( (unsigned char) respuesta[byte+29] );
                        ano= (int) ( (unsigned char) respuesta[byte+30] )+1900;

// ---------------------------- GRABAR RESPUESTA -----------------------------

                        C.SalidaChar(Tag, true);
                        C.SalidaChar(Descripcion, false);
                        C.SalidaInt(dia, false);
                        C.SalidaInt(mes, false);
                        C.SalidaInt(ano, false);

                        C.SalidaChar(modelo, false);
                        C.SalidaChar(fabricante, false);
                        for(int i=2; i<7; i++)
                                C.SalidaInt((int)datos[i], false);
```

```
                }
                else
                C.SalidaChar(MensajeError, true);

        }//endif primer if(Error==0)
        else
                C.SalidaChar(MensajeError, true);

}

void Gestor_Info::AsignarNuevo(int IdServidor, char IpServidor[], int puerto, char
Direccion [], char nTag[], char nDescripcion[], int ndia, int nmes, int nano)
{
        nano = nano -1900;

        while(strlen(nTag) < 8)
                strcat(nTag, " ");

        char* Ptag = A.EmpaquetarASCII(nTag);
        strcpy(PackTag, Ptag);

        while(strlen(nDescripcion) < 16)
                strcat(nDescripcion, " ");

        char* Pdesc = A.EmpaquetarASCII(nDescripcion);
        strcpy(PackDescripcion, Pdesc);


// ---------------------------- ENVIO DE SOLICITUD ----------------------------
        checksum = (0x82 ^ Direccion[0] ^ Direccion[1] ^ Direccion[2] ^
Direccion[3] ^ Direccion[4] ^ 0x12 ^ 0x15 ^ (char)ndia ^ (char)nmes ^ (char) nano);
        for(int i=0; i<6; i++)
                checksum = checksum ^ PackTag[i];

        for(int i=0; i<12; i++)
                checksum = checksum ^ PackDescripcion[i];

        //Comando 18: Escribir Tag, Descriptor y fecha

        HART [0] = IdServidor;
        HART [1] = puerto;
        HART [2] = 34;
        HART [3] = 0xff;                                //Preambulo
```

```
	HART [4] = 0xff;				//Preambulo
	HART [5] = 0xff;				//Preambulo
	HART [6] = 0xff;				//Preambulo
	HART [7] = 0x82;				// Byte de inicio Maestro (Formato largo)
	HART [8] = Direccion[0];		// Direccion del instrumento (Formato largo)
	HART [9] = Direccion[1];
	HART [10] = Direccion[2];
	HART [11] = Direccion[3];
	HART [12] = Direccion[4];
	HART [13] = 0x12;				// Comando Universal 18(12)
	HART [14] = 0x15;				// Byte count

	for(int i=0; i<6; i++)			//Tag (Paquete ASCII)
		HART [15+i] = PackTag[i];

	for(int i=0; i<12; i++)
		HART [21+i] = PackDescripcion[i]; //Descripcion (Paquete ASCII)

	HART [33] = (char)ndia;		//Dia
	HART [34] = (char)nmes;		//Mes
	HART [35] = (char)nano;		//AÃ±o
	HART [36] = checksum;			// Suma de verificacion

	char* PRespuesta = TCP.EnviarHART(HART);

	strcpy(respuesta, PRespuesta);
	respuesta[50]='\0';


// -------------------------- RECEPCION DE RESPUESTA -------------------------
	for(int i=0; i<40; i++)
		if((unsigned char)respuesta[i] == 0x88)
			respuesta[i] = 0x00;

	int pos = 0;
	while((unsigned char)respuesta[pos]!= 0x86)
	{
		pos++;
		if(pos==20)
			break;
```

```
	}

	Error=-1;
	//Conexion Fallida, no se pudo conectar con el servidor
	if((unsigned char)respuesta[0]==0xCC && (unsigned char)respuesta[1]==0xCC && (unsigned char)respuesta[2]==0xCC && (unsigned char)respuesta[3]==0xCC && (unsigned char)respuesta[4]==0xCC && (unsigned char)respuesta[5]==0xCC)
	{
		strcpy(MensajeError, "Error, No se pudo conectar al servidor");
		Error=1;
	}

	//Servidor encontrado pero Instrumento no respondio a solicitud
	else if((unsigned char)respuesta[0]==0xAA && (unsigned char)respuesta[1]==0xBB && (unsigned char)respuesta[2]==0xAA && (unsigned char)respuesta[3]==0xBB && (unsigned char)respuesta[4]==0xAA)
	{
		strcpy(MensajeError, "Error, Instrumento no respondiÃ³ a solicitud, verificar si estÃ¡ conectado");
		Error=2;
	}

	else if( pos==20 || (unsigned char)respuesta[pos+8]==0x20)
	{
		Error=4;
		sprintf(MensajeError, "Error, en cÃ³digo de respuesta, Intentelo nuevamente, codigo: %X", (unsigned char)respuesta[pos+8] );
	}
	else if( (unsigned char)respuesta[pos+8]==0x07)
	{
		Error=4;
		sprintf(MensajeError, "Error, dispositivo protegido contra escritura" );

	}

	//Solicitud satisfactoria, Identificando Instrumento
	else if((unsigned char)respuesta[pos]== 0x86)
		Error=0;


	//Servidor encontrado pero Instrumento dio respuesta no esperada
	else
```

```
{
	strcpy(MensajeError, "Error desconocido");
	Error=3;
}

if(Error==0)
{
	int byte = 0;
	while((unsigned char)respuesta[byte]!= 0x86)
		byte++;

	//Obteniendo Tag en paquete ASCII
	char tagPack[6];
	for(int i=0; i<6; i++)
		tagPack[i] = (unsigned char) respuesta[byte+10+i];

	//Desempaquetando Tag
	char* PTag=A.DesempaquetarASCII(tagPack);
	strcpy(Tag, PTag);
	Tag[8]='\0';


	//Obteniendo Descripcion en paquete ASCII
	char DescripcionPack[12];
	for(int i=0; i<12; i++)
		DescripcionPack[i] = (unsigned char) respuesta[byte+16+i];

	//Desempaquetando Descripcion
	char* PDesc=A.DesempaquetarASCII(DescripcionPack);
	strcpy(Descripcion, PDesc);
	Descripcion[16]='\0';

	//Obteniendo Fecha
	dia= (int) ( (unsigned char) respuesta[byte+28] );
	mes= (int) ( (unsigned char) respuesta[byte+29] );
	ano= (int) ( (unsigned char) respuesta[byte+30] )+1900;

	//Verificando si se asignÃ³ la informacion correctamente

	int codigo = (int)respuesta[byte+8];

	if(codigo==0)
		strcpy(mensaje, "Tag, descripcion y fecha asignadas
```

```
correctamente");
                else if(codigo==7)
                        strcpy(mensaje, "Error, dispositivo protegido contra escritura");
                else
                        strcpy(mensaje, "Error, no se pudo asignar la informaciÃ³n");

// ---------------------------- GRABAR RESPUESTA -----------------------------

                C.SalidaChar(mensaje, true);
                C.SalidaChar(Tag, false);
                C.SalidaChar(Descripcion, false);
                C.SalidaInt(dia, false);
                C.SalidaInt(mes, false);
                C.SalidaInt(ano, false);

        }//final de if fin > 0
        else
                C.SalidaChar(MensajeError, true);
}
```

5.4.2.- Implementación de funcionalidades para mantenimiento y configuración

En la figura 5.18 se muestra el diagrama de componentes para el subsistema de mantenimiento y configuración del sistema. En este diagrama se encuentran los componentes principales que participan en la operación de escaneo de la red. A continuación se presenta la implementación de la clase más importantes en este diagrama de componentes.

La clase Gestor_Escaneo es la encargada de controlar el proceso de escaneo de la red HART en busca de instrumentos conectados y activos. La clase está compuesta principalmente por dos ciclos anidados que generan cada iteración de la búsqueda a través de todos los servidores registrado en el sistema y, a su vez por cada uno de los puertos seriales que componen a cada servidor TCP/IP-HART esta operación consulta en cada puerto de los servidores si existe un instrumento conectado. Para ello se

construye una trama de formato corto ya que se desconoce la identidad de los dispositivos de campo a los que se están consultando. De esta manera los instrumentos que reciben la solicitud responden con sus datos de identificación. A continuación se presenta el código de la clase Gestor_Escaneo.

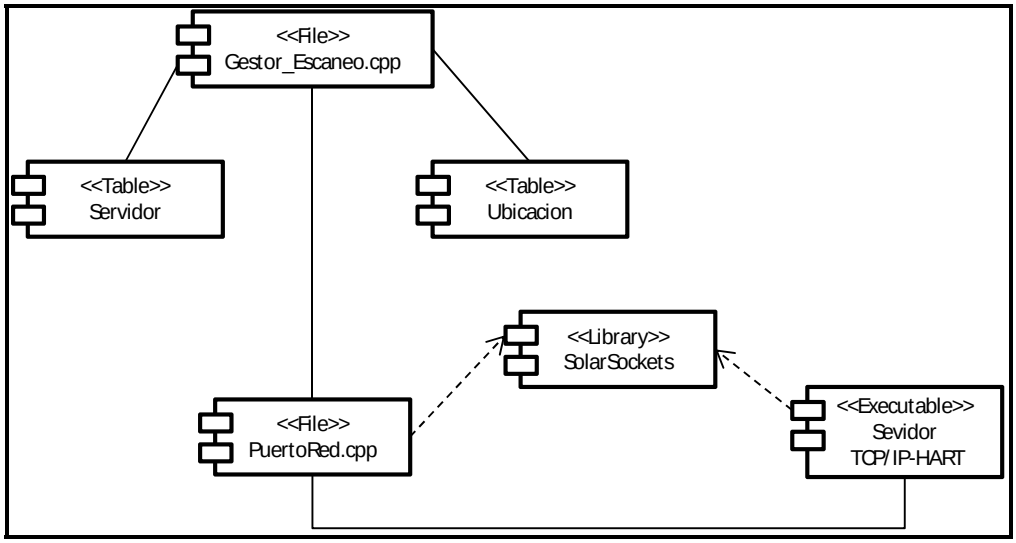

Figura 5.18. Diagrama de componentes para mantenimiento y configuración

Fuente: Elaboración propia

```
#ifndef GESTOR_ESCANEO_H_
#define GESTOR_ESCANEO_H_
#include <fcntl.h>
#include <termios.h>
#include "Comunicador.h"
#include "PuertoRed.h"
#include "ServicioASCII.h"
#include "Ubicacion.h"


#include <iostream>
#include <string>
#include <stdlib.h>
```

```
#include <stdio.h>
#include <postgresql/libpq-fe.h>
#include "postgresql/libpq-fe.h"
#include <pqxx/pqxx>
#include <string.h>

class Gestor_Escaneo
{
public:
	Gestor_Escaneo();
	virtual ~Gestor_Escaneo();
	void IniciarEscaneo(void);
	char* ConstruirDireccion(char respuesta[]);

private:
	int CantServidores();
	void IdentificarTag(int servidor, int puerto, char Direccion []);
	int BuscarIDServidor(int posServidor);
	Comunicador C;
	ServicioASCII A;
	PuertoRed TCP;
	int IDServidor;
	char HART[14], respuesta[255], dir[5], Tag[9];
};

#endif /*GESTOR_ESCANEO_H_*/


#include "Gestor_Escaneo.h"

Gestor_Escaneo::Gestor_Escaneo()
{
}

Gestor_Escaneo::~Gestor_Escaneo()
{
}

void Gestor_Escaneo::IniciarEscaneo()
{
	Ubicacion ubicacion;
	ubicacion.LimpiarRegistro();//Ordena limpiar los registros de rutas previas
	int Cant_Servidores= CantServidores();
```

```
int Cant_Puertos=0;
unsigned char checksum = (0x02 ^ 0x00);

for(int servidor=0; servidor< Cant_Servidores; servidor++)
//Ciclo de escaneo por cada servidor
        for(int puerto=0; puerto<= Cant_Puertos; puerto++)
        //Ciclo de escaneo por cada puerto de servidor
        {
                IDServidor = BuscarIDServidor(servidor);

                //Armando trama HART
                HART [0] = IDServidor ;      //posicion en tabla Servidor donde se realiza la busqueda
                HART [1] = puerto ;          //Puerto donde se realiza la busqueda
                HART [2] = 9;          //Longitud de Trama HART
                HART [3] = 0xff;             //Preambulo
                HART [4] = 0xff;
                HART [5] = 0xff;
                HART [6] = 0xff;
                HART [7] = 0x02;             //Cabecera (Formato corto)
                HART [8] = 0x00;             //Direccion
                HART [9] = 0x00;             //Comando universal 00
                HART [10] = 0x00;            //Byte count
                HART [11] = checksum;        //Suma de verificacion

                //Enviando Trama HART Via TCP/IP
                char* PRespuesta = TCP.EnviarHART(HART);
                strcpy(respuesta, PRespuesta);
                respuesta[50]='\0';

//Conexion Fallida, no se pudo conectar con el servidor, rompe busquedas a puertos en ese servidor
                if((unsigned char)respuesta[0]==0xCC && (unsigned char)respuesta[1]==0xCC && (unsigned char)respuesta[2]==0xCC && (unsigned char)respuesta[3]==0xCC && (unsigned char)respuesta[4]==0xCC && (unsigned char)respuesta[5]==0xCC)
                {
                        //cout << endl << "No se pudo conectar al servidor\nRompiendo busquedas en este servidor" << endl;
                        break;
                }
```

```
                        //Solicitud satisfactoria, Identificando Instrumento
                        else if( (unsigned char)respuesta[0]==0xff && (unsigned
char)respuesta[1]==0xff)
                        {
                                strcpy (dir, ConstruirDireccion(respuesta) );

                                IdentificarTag(IDServidor, puerto, dir);

                                ubicacion.AgregarUbicacion(Tag, IDServidor, puerto,
dir);
                        }

                        //Servidor encontrado pero Instrumento dio respuesta no
esperada
                        else
                                cout << endl << "Error desconocido" << endl;
                }
}

char* Gestor_Escaneo::ConstruirDireccion(char respuesta[])
{
        int byte=0;
        char Direccion[5];

        while( (unsigned char)respuesta[byte]==0xff)
                byte++;

        if((unsigned char)respuesta[byte]==0x06)
        {
                Direccion[0] = respuesta[byte+7];
                Direccion[1] = respuesta[byte+8];
                Direccion[2] = respuesta[byte+15];
                Direccion[3] = respuesta[byte+16];
                Direccion[4] = respuesta[byte+17];
                Direccion[5] = '\0';
        }
        char* Dir = Direccion;
        strcpy (Direccion, Dir);
        return Dir;
}

int Gestor_Escaneo::CantServidores()
```

```
{
	PGconn* conn;
	PGresult* result;

	char Host [] = "localhost";
	char Port [] = "5432";
	char BDName [] = "MAIP";
	char LOGIN [] = "pgmaster";
	char PWD [] = "21102110";

	conn = PQsetdbLogin(Host, Port, NULL, NULL, BDName, LOGIN, PWD);
//Conectando a Base de Datos

	if (PQstatus(conn) == CONNECTION_BAD)
	{
		fprintf(stderr, "La conexion a la base de datos; '%s' fallo.\n",
BDName);
		fprintf(stderr, "%s", PQerrorMessage(conn));
		PQfinish(conn);
		exit(1);
	}

	char operacion[80];
	sprintf( operacion, "SELECT * FROM servidores");
	result = PQexec(conn, operacion);

	if ((!result) || (PGRES_TUPLES_OK != PQresultStatus(result)))
	{
		fprintf(stderr, "Error al enviar la consulta.\nInforme detallado: %s\n",
		PQerrorMessage(conn));
		PQfinish(conn);
		exit(1);
	}
	int CantServidores = PQntuples(result);
	PQfinish(conn);
	return CantServidores;
}



void Gestor_Escaneo::IdentificarTag(int IDServidor, int puerto, char Direccion [])
{
```

```
//  ------------------------  ARMANDO TRAMA HART  ------------------------
        unsigned char checksum = (0x82 ^ Direccion[0] ^ Direccion[1] ^ Direccion[2]
^ Direccion[3] ^ Direccion[4] ^ 0x0d);

        HART [0] = IDServidor;
        HART [1] = puerto;
        HART [2] = 13;
        HART [3] = 0xff;
        HART [4] = 0xff;
        HART [5] = 0xff;
        HART [6] = 0xff;
        HART [7] = 0x82;
        HART [8] = Direccion[0];
        HART [9] = Direccion[1];
        HART [10] = Direccion[2];
        HART [11] = Direccion[3];
        HART [12] = Direccion[4];
        HART [13] = 0x0d;
        HART [14] = 0x00;
        HART [15] = checksum;

//  ------------------------  ENVIANDO TRAMA HART VÃ-a TCP/IP  ----------------
------
        char* PRespuesta = TCP.EnviarHART(HART);

        strcpy(respuesta, PRespuesta);
        respuesta[50]='\0';

        int byte=0;
        while((unsigned char) respuesta[byte]!= 0x86)
                byte++;

        //Obteniendo Tag en paquete ASCII
        char tagPack[7];
        for(int i=0; i<6; i++)
                tagPack[i] = (unsigned char) respuesta[byte+10+i];
        tagPack[6]='\0';

        //Desempaquetando Tag
        char* PTag=A.DesempaquetarASCII(tagPack);
        strcpy(Tag, PTag);

        if(Tag[7]==0x20)
```

```
		Tag[7]='\0';
	Tag[8]='\0';


}


int Gestor_Escaneo::BuscarIDServidor(int posServidor)
{
	PGconn* conn;
	PGresult* result;

	char Host [] = "localhost";
	char Port [] = "5432";
	char BDName [] = "MAIP";
	char LOGIN [] = "pgmaster";
	char PWD [] = "21102110";

	int conexion=0, inicio=conexion;
	do
	{
		inicio = conexion;
		conn = PQsetdbLogin(Host, Port, NULL, NULL, BDName, LOGIN,
PWD); //Conectando a Base de Datos

		if (PQstatus(conn) == CONNECTION_BAD)
		{
			fprintf(stderr, "La conexion a la base de datos; '%s' fallo.\n",
BDName);
			fprintf(stderr, "%s", PQerrorMessage(conn));
			PQfinish(conn);
			conexion++;
		}

	}while (conexion>0 && conexion <3 && inicio!=conexion);

	char operacion[80];
	sprintf( operacion, "SELECT * FROM servidores ORDER BY id_servidor
ASC");
	result = PQexec(conn, operacion);

	if ((!result) || (PGRES_TUPLES_OK != PQresultStatus(result)))
```

```
        {
                fprintf(stderr, "Error al enviar la consulta.\nInforme detallado: %s\n",
                PQerrorMessage(conn));
                PQfinish(conn);
                exit(1);
        }
        IDServidor = atoi (PQgetvalue(result,posServidor,0));
        PQfinish(conn);
        return IDServidor;
 }
```

5.4.3.- Implementación de la clase PuertoRed

La clase PuertoRed es una de las clases principales del sistema y participa en todas las operaciones que implica realizar una transacción en línea con la instrumentación instalada en campo. Esta clase cumple la función de clase interfaz que permite que las tramas HART ensambladas en las clases gestoras para operaciones en línea a la instrumentación de campo, sean empaquetadas implementando la librería de sockets en paquetes TCP/IP que serán enviados a través de la intranet de la empresa dirigido al servidor TCP/IP-HART que aloje al instrumento al cual se desea realizar la transacción. Esta clase también recibe los paquetes de respuesta provenientes de los servidores TCP/IP-HART para desempaquetarla y retornar la trama respuesta a la clase gestora que originalmente genero la operación.

```
#ifndef PUERTORED_H_
#define PUERTORED_H_

#include <SolarSockets/SolarSockets++.h>
#include <iostream>
#include <string.h>

#include <iostream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
```

```
#include <postgresql/libpq-fe.h>
#include "postgresql/libpq-fe.h"
#include <pqxx/pqxx>

#include <string.h>

class PuertoRed : public ssPPClient
{
public:
	PuertoRed();
	virtual ~PuertoRed();
	char* EnviarHART(char Trama []);
	char IPServidor[15];
private:
	void onConnect();
	void onError(int ssError);
	void onDataArrival(string Data);
	char* BuscarServidor(int servidor);
	char miTrama [50], Respuesta[50];
	char* respuesta;
	int Error, connected;
};

#endif /*PUERTORED_H_*/


#include "PuertoRed.h"

PuertoRed::PuertoRed()
{
}

PuertoRed::~PuertoRed()
{
}

void PuertoRed::onConnect()
{
	Error=0;
	connected=1;
	Send(miTrama);
}
```

```
void PuertoRed::onDataArrival(string Data)
{
	for(int i=0; i<50; i++)
	Respuesta[i] = Data[i];
	respuesta = Respuesta;
	strcpy(Respuesta, respuesta);
}


void PuertoRed::onError(int ssError)
{

	Error=ssError;
}


char* PuertoRed:: EnviarHART (char Trama [])
{
	// Trama[0] Id del servidor TCP/IP-HART donde se encuentra conectado el instrumento
	// Trama[1] puerto serial del servidor donde estÃ¡ conectado directamente el instrumento
	// Trama[2] Longitud de trama HART
	// Trama[3] en adelanta trama HART

	connected=0;

	for(int i=0; i<= (int) Trama[2]+2; i++)
	{
		if( (unsigned char) Trama[i]==0x00)
			miTrama[i] = 0x88;
		else
			miTrama[i] = (unsigned char) Trama[i];
	}

	char* Pserver = BuscarServidor( (int)Trama[0]);		//Se busca la direccion Ip del servidor buscandola por su Identificador registrado en la tabla servidores
	strcpy(IPServidor, Pserver);


	Connect(Pserver, 1080);
	//Conexion a servidor seleccionado
```

```
	usleep(3500000);
	//Tiempo de espera de respuesta

			//se recibe en funcion onDataArrival

	if(Error==2004 || connected==0)
	//Error 2004 Conexion Fallida, no se pudo alcanzar el servidor
	{
		Error=0;
		for(int i=0; i<6; i++)
	//Codificando trama de Conexion Fallida
			Respuesta[i]=0xCC;
		respuesta=Respuesta;
	}

	Close();
	return respuesta;

}

char* PuertoRed::BuscarServidor(int IDServidor)
{
	PGconn* conn;
	PGresult* result;

	char Host [] = "localhost";
	char Port [] = "5432";
	char BDName [] = "MAIP";
	char LOGIN [] = "pgmaster";
	char PWD [] = "21102110";

	int conexion=0, inicio=conexion;
	do
	{
		inicio = conexion;
		conn = PQsetdbLogin(Host, Port, NULL, NULL, BDName, LOGIN,
PWD); //Conectando a Base de Datos

		if (PQstatus(conn) == CONNECTION_BAD)
		{
			/*cout << endl << "Error Fallo Conexion Buscar Servidor" <<
endl;
			fprintf(stderr, "La conexion a la base de datos; '%s' fallo.\n",
```

```
BDName);
                        fprintf(stderr, "%s", PQerrorMessage(conn));*/
                        PQfinish(conn);
                        conexion++;
                }

        }while (conexion>0 && conexion <3 && inicio!=conexion);

        char operacion[80];
        sprintf( operacion, "SELECT ip_servidor FROM servidores WHERE
id_servidor='%i'", IDServidor);
        result = PQexec(conn, operacion);

        if ((!result) || (PGRES_TUPLES_OK != PQresultStatus(result)))
        {
                fprintf(stderr, "Error al enviar la consulta.\nInforme detallado: %s\n",
                PQerrorMessage(conn));
                PQfinish(conn);
                exit(1);
        }
        char* IPServer = PQgetvalue(result,0,0);
        PQfinish(conn);
        return IPServer;
 }
```

5.5.- Pruebas del Sistema

El software se prueba para descubrir errores cometidos durante el diseño y construcción. La prueba empieza por lo “pequeño” y avanza hacia lo “grande”. Esto significa que, en las primeras etapas, la prueba se concentra en un solo componente o en un grupo pequeño de componentes relacionados y, se aplica para descubrir errores en la lógica de datos y del procesamiento que se ha encapsulado en el componente. Una vez aprobados los componentes, deben integrarse hasta que todo el sistema se haya construido. En este punto se ejecuta una serie de pruebas de alto nivel para descubrir errores en la satisfacción de los requisitos del cliente. A medida que se

cubren, los errores deben diagnosticarse y corregirse empleando un proceso llamado depuración.

Si se considera el proceso desde un punto de vista procedimental, en realidad la prueba dentro del contexto de la ingeniería del software consiste en una serie de cuatro pasos que se implementan de manera secuencial. Esos pasos se muestran en la figura 5.19. Al principio, la prueba se concentra en cada componente individual, asegurando que funciona de manera apropiada como unidad (por ello se denomina prueba de unidad). La prueba de unidad emplea en forma recurrente las técnicas de prueba que recorren caminos específicos en una estructura de control del componente, lo que asegura una cobertura completa y una detección máxima de errores. Enseguida deben ensamblarse o integrase los componentes para formar el paquete de software completo. La prueba de integración atiende todos los aspectos asociados con el doble problema de verificación y construcción del programa. Las técnicas de diseño de casos de prueba que se concentran en entradas y salidas son más dominantes durante la integración, aunque podrían usarse técnicas que recorren rutas específicas del programa para asegurar la cobertura de los principales caminos de control.

Después de que se ha integrado (construido) el software se aplica un conjunto de pruebas de alto nivel. Se deben evaluar los criterios de validación establecidos durante el análisis de requisitos. La prueba de validación proporciona un aseguramiento final de que el software cumple con todos los requisitos funcionales, de comportamiento y desempeño.

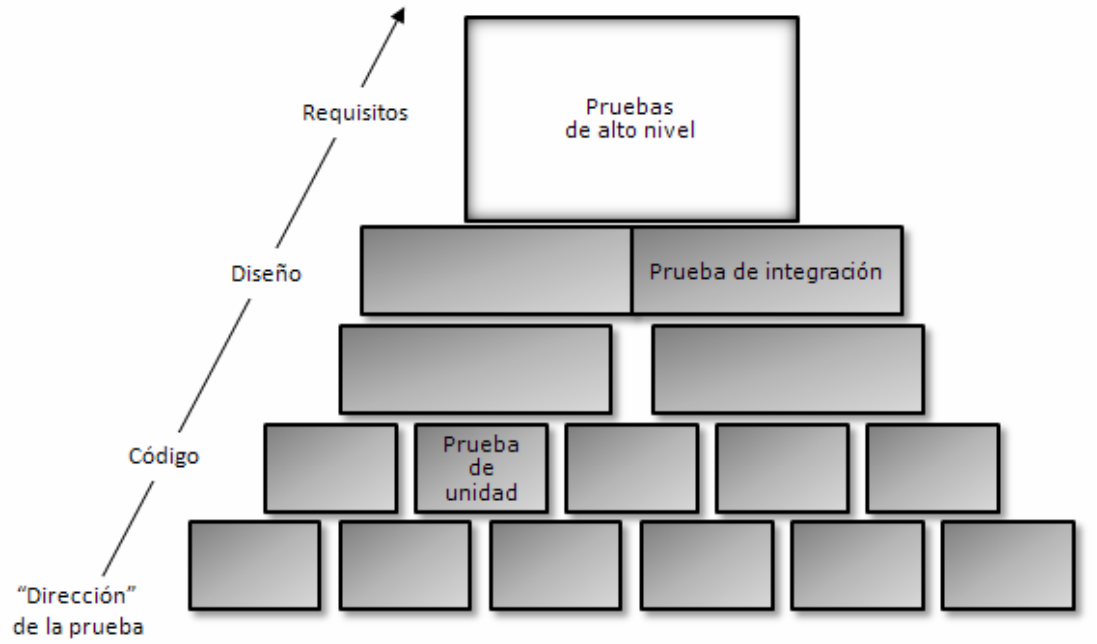

Figura 5.19. Pasos de la prueba del software

Fuente: Pressman, R. 2008

5.5.1.- Prueba de Unidad

La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño de software, entre el componente o módulo de software. Tomando como guía la descripción del diseño al nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. El alcance restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que éstas descubren. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente. Este tipo de pruebas se puede aplicar en paralelo a varios componentes.

Las pruebas de unidad se basan en los requerimientos del sistema teniendo claro lo que debe hacer funcionalmente hablando cada unidad, se realizaron pruebas individuales a cada unidad de la siguiente manera:

Se le proporcionó a la unidad la entrada esperada, esto de acuerdo a la función que cumple o debe cumplir dicha unidad, la unidad se puso en marcha y se estudio la salida, comparando la salida obtenida con la salida esperada. Estas pruebas se realizaron con diferentes patrones de entrada, siendo cada uno de estos patrones, posibles entradas reales cuando se haya integrado con otros componentes. Para las unidades que en sus diversas pruebas no presentaron los resultados deseados se procedió a verificar cada uno de los posibles fallos en la unidad. Las pruebas de unidad sirvieron para descubrir errores debidos a cálculos incorrectos, comparaciones erróneas o flujos de control inapropiados.

A continuación se listan los puntos revisados que han podido ocasionar fallos.

Errores más comunes en los cálculos

- Aplicación incorrecta o mal entendida de la precedencia aritmética.
- Operaciones de modo mezcladas.
- Inicialización incorrecta.
- Falta de precisión.
- Representación simbólica incorrecta de una expresión.

Errores de comparación y de flujo de control

- Comparación entre diferentes tipos de datos.
- Operadores lógicos o precedencia de éstos aplicados incorrectamente.

- Expectativa de igualdad cuando los errores de precisión hacen que sea poco probable.
- Comparación incorrecta de variables.
- Terminación inapropiada o inexistente de bucles.
- Falla en la salida cuando se encuentra una iteración divergente.
- Variables de bucle modificadas de manera inapropiada.

5.5.2.- Pruebas de Integración

La integración de los módulos del sistema y la ejecución de las pruebas, se planean realizar utilizando la integración incremental (antítesis del enfoque del "big bang"). El programa se construye y prueba en pequeños incrementos, en los cuales resulta fácil aislar y corregir errores. Se planea utilizar como estrategias de integración la integración incremental ascendente.

La prueba de integración ascendente, como su nombre lo indica, empieza la construcción y la prueba con módulos atómicos (es decir, componentes de los más bajos de la estructura del programa). Debido a que los componentes se integran de abajo hacia arriba, siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel y, se elimina la necesidad de resguardos.

5.5.2.1.- Prueba de caja negra

Las pruebas de caja negra se concentran en los requisitos funcionales del software, es decir, permiten derivar un conjunto de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. La prueba de caja negra es

un enfoque complementario al de caja blanca, este enfoque tiene probabilidades de descubrir una clase diferente de errores.

Las pruebas de caja negra tratan de encontrar errores derivados por funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos o en acceso a bases de datos, errores de comportamiento o desempeño, errores de inicialización y término.

Para realizar las pruebas de caja negra existe una técnica algebraica llamada "clases de equivalencia", consiste en tratar a todos las posibles entradas y parámetros como un modelo algebraico, y utilizar las clases de este modelo para probar un amplio rango de posibilidades.

Tabla 5.1. Clase de equivalencia para operación "Asignar nuevo Tag, descriptor y fecha" 1/2

Fuente: Elaboración propia

Nº	Campo	Caso de Prueba	Salida
1	Tag	nulo	"Tag tiene valor nulo"
2	Tag	FT- 2 32	"Tag no debe contener espacios en blanco"
3	Tag	FT-A2727	Tag tiene mal formato se esperan dos dígitos seguidos del guión que indican el código de la planta Ejm. FT-01003.

Tabla 5.2. Clase de equivalencia para operación "Asignar nuevo Tag, descriptor y fecha" 2/2

Fuente: Elaboración propia

Nº	Campo	Caso de Prueba	Salida
4	Tag	FT02727	Tag tiene mal formato debe contener guión (-) Ejm. FT-01003
5	descripción	nulo	Descripción tiene valor nulo
6	Fecha	nulo	Fecha tiene valor nulo Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)
7	Fecha	abcd	Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)
8	Fecha	123	Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)
9	Fecha	12-24-09	Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)
10	Fecha	12-02-2009A	Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)
11	Fecha	"21-10 -2007"	Fecha no debe contener espacios en blanco
12	Fecha	43-05-2009	Fecha tiene día invalido
13	Fecha	03-18-2006	Fecha tiene mes invalido

14	Fecha	03-18-3060	Fecha tiene año invalido
15	Fecha	21/10/2007	Fecha tiene formato incorrecto, debe ser (dd-mm-aaaa)

Tabla 5.3. Clase de equivalencia para operación "Cambiar dirección del instrumento"

Fuente: Elaboración propia

Nº	Caso de Prueba	Salida
1	aac	Nueva dirección tiene valor invalido, (0-15)
2	16	Nueva dirección está fuera de rango, (0-15)
3	-5	Nueva dirección está fuera de rango, (0-15)
4	nulo	Nueva dirección tiene valor nulo
5	" 3"	Nueva dirección no debe contener espacios en blanco

Tabla 5.4. Clase de equivalencia para operación "Ajustar Damping"

Fuente: Elaboración propia

Nº	Caso de Prueba	Salida
1	"tres"	Damping debe tener valor numérico
2	-8	Damping fuera de rango, (0-60)Sg
3	61	Damping fuera de rango, (0-60)Sg
4	"6 1"	Damping no debe contener espacios en blanco

Tabla 5.5. Clase de equivalencia para operación "Ajustar rangos" 1/2

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Bajo	Alto	Protección	
1	Nulo	Nulo	"Activada"	La protección contra escritura se encuentra activa, de esta manera no podrá realizar la operación de ajuste

Tabla 5.6. Clase de equivalencia para operación "Ajustar rangos" 2/2

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Bajo	Alto	Protección	
2	Nulo	Nulo	"Desactivada"	- Rango alto tiene valor nulo - Rango bajo tiene valor nulo - Rango alto por debajo del rango bajo
3	Nulo	3	"Desactivada"	- Rango bajo tiene valor nulo
4	3	Nulo	"Desactivada"	- Rango alto tiene valor nulo - Rango alto por debajo del rango bajo
5	5	2	"Desactivada"	- Rango alto por debajo del rango bajo
6	"A"	"253"	"Desactivada"	- Rango alto debe tener valor numérico - Rango alto no debe contener espacios en blanco

				- Rango bajo debe tener valor numérico - Rango alto por debajo del rango bajo

Tabla 5.7. Clase de equivalencia para cambio de contraseña

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Actual	nueva	Repite nueva	
1	Nulo	Nulo	Nulo	- Contraseña tiene valor nulo - Nueva contraseña tiene valor nulo - Repetición nueva contraseña tiene valor nulo
2	“21102110”	“147”	“A147”	Nueva contraseña fue escrita incorrectamente, no coincide con la replica
3	“21102110”	“AV2110”	“AV2110”	Error! - Contraseña actual invalida
4	“21102110”	“av2110”	“AV2110”	Nueva contraseña fue escrita incorrectamente, no coincide con la replica

Tabla 5.8. Clase de equivalencia para Administración de usuarios operación agregar

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Usuario	Contraseña	Estatus	
1	Nulo	Nulo	"Seleccione"	- Usuario tiene valor nulo - Contraseña tiene valor nulo - Estatus no se ha hecho ninguna selección
2	"P Sucre"	"123 ABC"	"Seleccione"	- Usuario no debe contener espacios en blanco - Contraseña no debe contener espacios en blanco - Estatus no se ha hecho ninguna selección
3	"PSucre"	"123ABC"	"Instrumentista"	Usuario Registrado Satisfactoriamente
4	"PSucre"	"Paco123"	"Administrador"	Error! - Usuario Ya Registrado
5	"Morales05"	Nulo	"Supervisor de planta"	Contraseña tiene valor nulo

Tabla 5.9. Clase de equivalencia para Administración de usuarios operación modificar

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Usuario	Contraseña	Estatus	
1	“PSucre”	“123ABC”	“Administrador”	Usuario Modificado Satisfactoriamente
2	“PSucre”	Nulo	“Administrador”	Contraseña tiene valor nulo
3	“PSucreX”	“PLC05”	“Analista”	Error! - Usuario Ha Modificar No Existe
4	Nulo	“123ABC”	“Administrador”	Usuario tiene valor nulo

Tabla 5.10. Clase de equivalencia para Administración de usuarios operación eliminar

Fuente: Elaboración propia

Nº	Caso de Prueba			Salida
	Usuario	Contraseña	Estatus	
1	“PSucre”	“123abc”	“Administrador”	Error! - Usuario Ha Eliminar No Existe
2	“PSucre”	“123ABC”	“Administrador”	Usuario Eliminado Satisfactoriamente
3	“JuanT”	“1159gzT”	“Administrador”	Error! Operación Denegada, Único Administrador Existente

Tabla 5.11. Clase de equivalencia para Administración de Estatus de usuarios operación agregar

Fuente: Elaboración propia

Nº	Caso de Prueba		Salida
	Código	Estatus	
1	1	"Administrador"	Error! - Estatus de Usuario Ya Registrado
2	"AA"	"Director"	Código debe tener valor numérico
3	"1 5"	"Director"	Código no debe contener espacios en blanco

Tabla 5.12. Clase de equivalencia para Administración de Estatus de usuarios operación modificar

Fuente: Elaboración propia

Nº	Caso de Prueba		Salida
	Código	Estatus	
1	3	"Soporte de automatizaciónZZZ"	Operación denegada, usuario critico del sistema
2	99	"Coordinador"	Error! - Estatus a modificar no existe
3	6	"Analista123"	Estatus de Usuario Modificado Satisfactoriamente
4	99	"Coordinador"	Error! - Estatus a eliminar no existe

Tabla 5.13. Clase de equivalencia para Administración de Estatus de usuarios operación eliminar

Fuente: Elaboración propia

Nº	Caso de Prueba		Salida
	Código	Estatus	
1	1	"Administrador"	Operación denegada, usuario critico del sistema
2	5	"Instrumentista"	Operación denegada, usuario critico del sistema
3	6	"Analista"	Estatus Eliminado Satisfactoriamente
4	99	"Coordinador"	Error! - Estatus a eliminar no existe

Tabla 5.14. Clase de equivalencia para Administración de Servidores operación agregar

Fuente: Elaboración propia

Nº	Caso de Prueba		Salida
	Identificador	Dirección IP	
1	"IP"	198.0.0.1	Identificador debe tener valor numérico
2	"5 "	198.0.0.1	Identificador no debe contener espacios en blanco
3	"40"	192.0.0.25	Servidor registrado satisfactoriamente
4	"15"	192.0.0.25	Error! - Servidor ya registrado, puede que el identificador o la dirección IP ya exista
5	"40"	178.25.30.11	Error! - Servidor ya registrado, puede que el identificador o la dirección IP ya exista

6	“12”	“198”	Dirección IP tiene formato invalido, debe ser XXX.XXX.XXX.XXX
7	“12”	Nulo	Dirección IP tiene valor nulo
9	“35”	198.10.	Dirección IP tiene formato invalido, debe ser XXX.XXX.XXX.XXX
9	“2”	192.0.0.20	Servidor registrado satisfactoriamente

Tabla 5.15. Clase de equivalencia para Administración de Servidores operación modificar

Fuente: Elaboración propia

Nº	Caso de Prueba		Salida
	Identificador	Dirección IP	
1	“2”	192.0.0.25	Error! - Dirección IP ya ha sido asignada a otro servidor

5.6.- Conclusión de la fase de construcción

La funcionalidad en ésta fase es de suma importancia, se han presentado algunas de las interfaces del sistema, los distintos componentes que conforman el sistema y la selección de los lenguajes y herramientas utilizadas para llevar a cabo el desarrollo del software. Además, se han realizado las pruebas del sistema, pruebas de unidad e integración que estuvieron enfocadas en la verificación funcional de cada componente y en la incorporación de componentes a la arquitectura del software.

CAPITULO 6
FASE DE TRANSICIÓN

6.1.- Introducción

Los objetivos básicos de esta fase son cumplir los requisitos establecidos en las fases anteriores, hasta la satisfacción de todos los requerimientos y, gestionar todos los aspectos relativos a la operación en el entorno del usuario, incluyendo la corrección de los defectos remitidos por los encargados de la versión beta o por aquellos seleccionados para las pruebas de aceptación.

Esta fase busca obtener un software de calidad final ajustado a los requerimientos y, orientado por la operatividad de los usuarios, para lograr así, un producto de gran desempeño que cubra todas las expectativas.

6.2.- Evaluación de la fase de transición

El software cumple con los requisitos obtenidos en las fases de inicio y elaboración, obteniéndose así el funcionamiento adecuado con respecto a lo requerido por los usuarios que realizan los procesos para la evaluación de las solicitudes.

La ejecución de las actividades asociadas al procesamiento de estas solicitudes obedece al resultado que arroje las pruebas sobre la versión beta del software. La corrección de estos errores y modificaciones sugeridas por los usuarios es lo que determinará la calidad funcional del software.

6.3.- Planificación de la fase de transición

La fase de transición para el desarrollo del sistema manejador de activos de instrumentación no es abarcada en este proyecto. La tarea de llevar a cabo la

ejecución de esta fase se atribuye al equipo del Departamento de Mantenimiento de Sistemas de la empresa, que se encargará de todo lo relacionado a la corrección de fallas o inclusión de nuevas funcionalidades.

CONCLUSIONES

- Considerar y utilizar la metodología RUP para el desarrollo del software sirvió para mantener una línea organizada, estructurada y confiable en la elaboración del proyecto. A su vez, el Lenguaje Unificado de Modelado constituyó una herramienta muy útil para la representación de los distintos módulos del software.

- El uso de tecnologías Web permitió el desarrollo de un sistema centralizado en un servidor, al cual, los usuarios pueden tener acceso desde un navegador web sin requerir instalaciones adicionales y sin importar el sistema operativo que usen. Sin embargo, el servidor requirió una plataforma de Software Libre para alojar al sistema.

- El lenguaje PHP, por su semejanza al lenguaje C, fue de fácil comprensión y aprendizaje, permitiendo que la elaboración de la aplicación Web no presentara muchos inconvenientes. Además, la incorporación de su librería gráfica JpGraph fue de gran ayuda para la generación de gráficos de comportamiento de las variables de los instrumentos durante el monitoreo en línea.

- El uso de AJAX como técnica de desarrollo web permitió realizar solicitudes al servidor en segundo plano, pudiendo hacer cambios en la página sin necesidad de recargarla, lo que implica un aumento en la interactividad, velocidad y usabilidad de la aplicación.

- La implementación de la librería SolarSockets ha permitido la conexión remota a la instrumentación de campo para realizar operaciones de monitoreo y calibración de forma rápida y eficiente.
- Durante la ejecución de las pruebas, se comprobaron todas las funcionalidades del sistema, especialmente, aquellas que permiten realizar operaciones de monitoreo y cambios en la configuración de la instrumentación en línea.

RECOMENDACIONES

- Utilizar un servidor dedicado, que albergue al sistema y la base de datos, para un mayor rendimiento, resguardo de la información y reducción de probabilidades de daños.

- Ampliar las funcionalidades del sistema, incluyendo otras operaciones de comandos HART para diversificar aun más las tareas que se pueden realizar sobre la instrumentación.

- Construir un dispositivo hardware de arquitectura sencilla y producción de bajo costo, que permita conectar una cantidad determinada de instrumentos. El dispositivo debe prestar servicios de multiplexión con una entrada TCP/IP y varias salidas de dos hilos Bell 202.

- Seleccionar correctamente la ubicación lógica del servidor que alberga al sistema, en la red, es muy importante, de manera que no disminuya el desempeño de la red y se minimice la cantidad de dispositivos de red por los cuales deben pasar los paquetes.

- Utilizar la información de la instrumentación, registrada en la base de datos para crear un sistema de inventario y auditoría de los activos.

BIBLIOGRAFÍA

[1] Hernández, V. (2007), **"Diseño de un manejador de activos para instrumentación digital, con protocolo HART, basado en herramientas computacionales de software libre"**, Trabajo de Grado, Departamento de Electricidad, Universidad de Oriente. Anzoátegui, Venezuela.

[2] Tenías, J. (2007), **"Desarrollo de un software basado en aplicaciones web para el monitoreo de dispositivos de la plataforma de telecomunicaciones de PDVSA-GAS"**, Trabajo de Grado, Departamento de Computación y Sistemas, Universidad de Oriente. Anzoátegui, Venezuela.

[3] Núñez, L. (2007), **"Desarrollo de una aplicación web para la visualización en tiempo real de los parámetros operacionales de producción de la empresa PDVSA"**, Trabajo de Grado, Departamento de Computación y Sistemas, Universidad de Oriente. Anzoátegui, Venezuela.

[4] Alonso, S., (2006), **"Creación de sitios web con PHP5"**. Editorial Mc Graw Hill. Segunda edición. España.

[5] Reyes G., (2006), **"Tópicos de control"** Universidad Santiago de Chile. www.geocities.com/khifar/Protocolo_HART.pdf

[6] Software for business solutions, (2006), **"Proceso Unificado Racional".** http://www.kynetia.es/calidad/rup-rational-unified-process.html

[7] "Descripción general Refinería Puerto La Cruz", (2005).

[8] Pressman, R. (2002), **"Ingeniería del Software: un enfoque práctico"**. Editorial McGraw-Hill. Quinta edición. España.

[9] Silberschatz, A., (2002), **"Fundamentos de Bases de Datos"**. Editorial Mc Graw Hill. Cuarta

edición. España.

[10] Febres E., (2001), **"Protocolo de Comunicación HART para los instrumentos de Flotech S.A."** Universidad Simón Bolívar. Caracas. Venezuela.

[11] Walton, S., (2001), **"Programación de Socket Linux".** Editorial SAMS. Primera edición. España.

[12] Schmuller, J., (2001), **"Aprendiendo UML en 24 Horas".** Editorial Prentice-Hall. Primera edición. España.

[13] Larman, C., (2000), **"UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos"**, Editorial Prentice Hall. Segunda edición. España.

[14] Jacaboson, I, Booch, G, Rumbaugh J, (2000), "**El Proceso Unificado de Desarrollo de Software**", Editorial Addison Wesley. España.

[15] Ramos, Z. (2000), **"Desarrollo de un software para la determinación del impacto sobre la plataforma de sistemas, ocasionado por alteraciones en el diseño de la instrumentación de una planta de mejoramiento de crudo"**, Trabajo de Grado, Departamento de Computación y Sistemas, Universidad de Oriente. Anzoátegui, Venezuela.

[16] Querales, O. (1999), **"Sistema de control distribuido"**, Trabajo de Grado, Departamento de Electricidad, Universidad de Oriente. Anzoátegui, Venezuela.

[17] Rational, (1999) **"Rational Unified Process: Best Practices for Software Development Teams"** Rational Software Corporation. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[18] Walker R, (1998), "**Software Project Management-A Unified Framework".** *Editorial* Addison-Wesley.

[19] Bowden R., (1997), **"*HART Field Communications protocol. A Technical Overview*".** Segunda Edición.

[20] Barry W, (1988), "**A Spiral Model of Software Development and Enhancement"**. Computer society.

[21] **Free Software Foundation**
http://www.fsf.org/about/what-is-free-software

[22] **HART Communication Foundation**
http://www.hartcomm.org/

[23] **Programa de la academia de networking de CISCO CCNA1**
http://www.cisco.com/web/learning/netacad/index.html

[24] **Apache Software Foundation**
http://www.apache.org/

[25] **World Wide Web School JavaScript**
http://www.w3schools.com/JS/js_intro.asp

[26] **World Wide Web School Ajax**
http://www.w3schools.com/Ajax/ajax_intro.asp

[27] **Documentación PostgreSQL**
http://www.postgresql-es.org

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	"Desarrollo de una interfaz TCP/IP - HART para la integración de un manejador de activos de instrumentación en línea, utilizando herramientas web, bajo software libre"
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Vila G., Arturo J.	**CVLAC:** 17.900.489 **E MAIL:** arturo.vila.86@gmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

Protocolo HART

Desarrollo web

Acceso remoto

RUP

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

AREA	SUBAREA
Ingeniería y Ciencias Aplicadas	Ingeniería en Computación

RESUMEN (ABSTRACT):

En la presente investigación se desarrolló un sistema que permite monitorear, calibrar y diagnosticar de manera remota la instrumentación de campo instalada en la Refinería de Puerto La Cruz. Esto se logró con la utilización del conjunto de protocolos TCP/IP para extender el alcance al sistema a través de la intranet de la empresa, así también, se utilizó el protocolo de comunicación industrial HART que permite por medio de su conjunto de comandos realizar operaciones sombre los dispositivos de campo. Este sistema reducirá la necesidad del instrumentista que labora en la empresa salir al campo en casos de diagnóstico, calibración y recolección de data, disminuirá de forma notoria el tiempo empleado para la corrección de fallas en los instrumentos, permitiendo una planificación óptima de mantenimiento o reemplazo de un instrumento sin afectar el funcionamiento normal de la planta. El diseño y construcción de la aplicación, se llevó a cabo siguiendo los lineamientos del Proceso Unificado de Desarrollo de Software, cuya metodología implementa el Lenguaje Unificado de Modelado (UML).

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Rodríguez Q., Luis A.	**ROL**	**CA**	**AS X**	**TU**	**JU**
	CVLAC:	V-10.885.794			
	E_MAIL	larodriguezq@cantv.net			
	E_MAIL				
Cortínez N., Claudio A.	**ROL**	**CA**	**AS**	**TU X**	**JU**
	CVLAC:	V-12.155.334			
	E_MAIL	cl_cortinez@hotmail.com			
	E_MAIL				
Gerardino G., María A.	**ROL**	**CA**	**AS**	**TU**	**JU X**
	CVLAC:	V-10.953.467			
	E_MAIL	magerardino@yahoo.com			
	E_MAIL				
Dorta B., Pedro A.	**ROL**	**CA**	**AS**	**TU**	**JU X**
	CVLAC:	V-12.914.617			
	E_MAIL	dortap@hotmail.com			
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2009	10	27
AÑO	**MES**	**DÍA**

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS.Manejador de activos.doc	Aplication/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: ____AIT-DIS (PDVSA)____ (**OPCIONAL**)

TEMPORAL: _____6 semestres_____ (**OPCIONAL**)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Ingeniero en computación

NIVEL ASOCIADO CON EL TRABAJO:

Pre-Grado

ÁREA DE ESTUDIO:

Departamento de Computación y Sistemas

INSTITUCIÓN:

Universidad de Oriente – Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

De acuerdo en el artículo 44 del Reglamento de Trabajos de Grado

"Los Trabajos de Grado son de exclusiva propiedad de la Universidad de Oriente y sólo podrán ser utilizados para otros fines con el consentimiento del Concejo de Núcleo respectivo, quien lo participará al Consejo Universitario".

Arturo Vila

AUTOR **AUTOR** **AUTOR**

Claudio Cortinez María Gerardino Pedro Dorta

TUTOR **JURADO** **JURADO**

POR LA SUBCOMISION DE TESIS