TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE LA LÓGICA DE CONTROL DEL PROCESO DE DESARROLLO DE UN SISTEMA HEAD-END PARA UNA RED DE DISPOSITIVOS INTELIGENTES"

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN SISTEMAS COMPUTACIONALES

PRESENTA

LIC. DORIS CASTRO VELASCO

DIRECTORES

M.C. JOSÉ JUAN HERNÁNDEZ MORA

M. C. MARÍA GUADALUPE MEDINA BARRERA

INSTITUTO TECNOLOGICO DE APIZACO

Apizaco, Tlaxcala, Junio 2018

SEP

TECNOLOGICO NACIONAL DE MEXICO

Instituto Tecnológico de Apizaco

Apizaco, Tlax., 25 de Junio de 2018

No. de Oficio: DEPI/235/18

ASUNTO: **Se Autoriza Impresión de Tesis de Grado.**

LIC. DORIS CASTRO VELASCO
CANDIDATA AL GRADO DE MAESTRA EN
SISTEMAS COMPUTACIONALES
No. de Control: **M00370031**
P R E S E N T E.

Por este medio me permito informar a usted, que por aprobación de la Comisión Revisora asignada para valorar el trabajo, mediante la Opción: **I Tesis de Grado por Proyecto de Investigación**, de la **Maestría en Sistemas Computacionales**, que presenta con el tema: "**DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE LA LÓGICA DE CONTROL DEL PROCESO DE DESARROLLO DE UN SISTEMA HEAD-END PARA UNA RED DE DISPOSITIVOS INTELIGENTES**" y conforme a lo establecido en el Procedimiento para la Obtención del Grado de Maestría en el Instituto Tecnológico, la División de Estudios de Posgrado e Investigación a mi cargo le emite la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar un ejemplar del mismo debidamente encuadernado y seis copias en CD en formato PDF, para presentar su Acto de Recepción Profesional a la brevedad.

Sin otro particular por el momento, le envío un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
PENSAR PARA SERVIR, SERVIR PARA TRIUNFAR®

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN.

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIO
DE POSGRADO E INVESTIGACIÓN

JFCV/M[illegible]mebr.

C.p. Expediente.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco

Apizaco, Tlax., 31 de Mayo de 2018

ASUNTO: **Aprobación del trabajo de Tesis de Maestría.**

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.
P R E S E N T E.

Por este medio se le informa a usted, que los integrantes de la **Comisión Revisora** para el trabajo de tesis de maestría que presenta la **LIC. DORIS CASTRO VELASCO**, con número de control **M00370031** candidata al grado de **Maestra en Sistemas Computacionales** y egresada del **Instituto Tecnológico de Apizaco,** cuyo tema es "**DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE LA LÓGICA DE CONTROL DEL PROCESO DE DESARROLLO DE UN SISTEMA HEAD-END PARA UNA RED DE DISPOSITIVOS INTELIGENTES",** fue:

A P R O B A D O

Lo anterior, al valorar el trabajo profesional presentado por la candidata y constatar que las observaciones que con anterioridad se le marcaron así como correcciones sugeridas para su mejora ya han sido realizadas.

Por lo que se avala se continúe con los trámites pertinentes para su titulación.

Sin otro particular por el momento, le envió un cordial saludo.

LA COMISIÓN REVISORA

M.C. JOSÉ JUAN HERNÁNDEZ MORA

M.C. MARÍA GUADALUPE MEDINA BARRERA

M.D.S. HIGINIO NAVA BAUTISTA

M.C.C. MARÍA JANAÍ SÁNCHEZ HERNÁNDEZ

C. p.- Interesada.

Dedico esta tesis a mis hijas:

Valeria Zoé y Andrea

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología
(CONACYT) por el apoyo para solventar la maestría
y realizar las prácticas profesionales fuera del estado.

Al Instituto Tecnológico de Apizaco
por brindar los medios y herramientas necesarias
que me permitieron realizar este trabajo.

A la empresa Eos Tech S.A. de C.V. por la
capacitación y facilidades proporcionadas para
la realización del proyecto dentro de sus instalaciones.

A mi director de tesis por las observaciones
realizadas y su apoyo en el desarrollo del proyecto.

Especialmente a mis hijas por su comprensión
y paciencia por estar lejos de ellas durante
la realización de mi proyecto.

Gracias a mi compañero de vida por
cuidarme y estar siempre pendiente de mí.

Resumen

El sistema desarrollado está enfocado en el diseño, desarrollo e implementación de la lógica de control del proceso de desarrollo de un Sistema Head-End (HES) para una red de dispositivos inteligentes para la empresa EosTech del estado de Querétaro. Esta empresa cuenta con una red de comunicación entre medidores inteligentes que permite hacer llegar la información desde un dispositivo hasta el software que lo requiera.

El caso de estudio está enfocado en la manipulación de los datos obtenidos de los medidores inteligentes, los cuales generan de forma programada o a petición de usuario, la información para ser almacenada. Esta información se recoge en una base de datos capaz de guardar todos los registros generados.

La problemática es obtener la información de la base de datos y procesarla según la aplicación que lo solicite y mostrar los datos. El sistema se desarrolla bajo la metodología *Building Blocks* que permite la creación de bloques funcionales del proyecto general, dando como resultado un proyecto escalable y con mejor calidad, gracias a las pruebas realizadas por bloques terminados.

El trabajo se complementa con la metodología ágil SCRUM, con iteraciones mensuales en las que se desarrollaron *millestons* propuestos en los componentes de software de una arquitectura Modelo Vista Controlador (MVC). La recuperación, interpretación y actualización en la base de datos se hace a través de Hibernate, empleando patrones de diseño Java para una programación optimizada.

Abstract

The developed system is focused on the design, development and implementation of the control logic process of a Head-End System (HES) for a network of intelligent devices of the EosTech company of Querétaro state. This company has a network communication between smart meters that allows to get the information from a device to the software that requires it.

The study case is focused on the manipulation of the information obtained from the smart meters, which generate in a programmed way or at the request of the user, information to be stored. This information is contained in a database with the capacity to save all the generated registers.

The problem is to obtain the information from the data base and process it according to the application that required it, and be able to show the data. The system developes under the Building Blocks methodology that allows the creation of functional blocks of the general project, resulting in a scalable project with better quality due to the tests carried out by finished blocks.

The work complemements with the agile SCRUM methodology with monthly iterations in which millestons proposed in the software components obtained from a Model View Controller (MVC) architecture were developed. The recovery, interpretation and updating in the database is make through Hibernate using Java design patterns for optimized programmin.

Tabla de contenido

CAPÍTULO 1 INTRODUCCIÓN 1
1.1 Descripción del problema 2
1.2 Justificación 5
1.3 Objetivo General 6
1.4 Objetivos Específicos 6
1.5 Metas 7
1.6 Pregunta de investigación 7
1.8 Análisis del estado del arte 7
1.8.1 Internet de las Cosas 7
1.8.2 Monitoreo de la información obtenida de los medidores electrónicos 8
1.8.3 Sistemas Comerciales de monitoreo de datos de medidores de electricidad 13
1.8.4 Seguridad en la comunicación a través de un Servicio Web 16
1.8.5 Inteligencia de Negocio 17
CAPÍTULO 2 MARCO TEÓRICO 19
2.1 Planeación de proyectos 19
2.2 Patrón de Diseño Arquitectónico Modelo Vista Controlador (MVC) 21
2.3 El componente Controlador 23
2.4 Tecnologías de controladores 25
2.4.1 Java Server Faces (JSF) 25
2.4.2 Spring 27
2.5 Metodología SCRUM 28
2.6 Maven 30
2.7 Hibernate 31
2.8 Servicio Web 34
CAPÍTULO 3 METODOLOGÍA 35
3.1 Servicio Web FTP 38
3.1.2 Levantar Servicio Web 42

3.1.3 Conexión y acceso con servidor remoto para obtener el archivo XML *43*
3.1.4 Validación de archivos XML *45*
3.1.5 Guardar tareas en base de datos Oracle *46*
3.2 Servicio de ejecución de tareas 48
3.2.1 Mantener en ejecución servicio Windows *52*
3.2.2 Enviar tarea a red distribuida para ser ejecutada *52*
3.3 Manejador de tareas 53
3.3.1 Obtener tareas priorizadas *56*
3.3.2 Control de excepciones para el manejo de interrupción de tareas *57*
3.3.3 Generación de archivo XML de respuesta de la red distribuida *57*
CAPÍTULO 4 PRUEBAS Y RESULTADOS **59**
4.1 Extracción y validación de archivos XML 59
4.2 Interpretación del archivo XML 63
4.3 Monitoreo de extracción de tareas en tiempo real 64
4.4 Ejecución de servidor Windows al iniciar computadora 64
4.5 Envío de petición de tarea al socket UDP 65
4.6 Activación de tiempos para la creación de thread pool 66
4.7 Extracción de tareas priorizadas 67
4.8 Distribución de procesos en los hilos configurados 69
4.9 Liberación de Módulos 71
CAPÍTULO 5 CONCLUSIONES Y TRABAJOS FUTUROS **75**
5.1 Conclusiones 75
5.2 Trabajos futuros 78
GLOSARIO DE TÉRMINOS **80**
REFERENCIAS **81**
ANEXOS **84**
Anexo 1. Formato para el registro de pruebas 84
Anexo 2. Publicaciones 85
Anexo 2. Cartas 99

3.1 Carta de satisfacción 99
3.2 Carta de término de estancias 100

Lista de figuras

Figura No. 2.1 El proceso de planeación del proyecto (Sommerville, 2011) 20
Figura No. 2.2 Arquitectura de aplicación Web con el patrón MVC (Sommerville, 2011) 22
Figura No. 3.1 Metodología de desarrollo de la Lógica de Control del Proceso de Desarrollo de un Sistema Head End para una Red de Dispositivos Inteligentes (elaboración propia) ... 36
Figura No. 3.2 Diagrama de flujo del proceso general del web service (elaboración propia) .. 39
Figura No. 3.3 Diagrama de Casos de Uso del servicio web (elaboración propia) 39
Figura No. 3.4 Diagrama de secuencias del servicio web (elaboración propia) 42
Figura No. 3.5 Estructura del servicio web (elaboración propia) 42
Figura No. 3.6 Conexión y acceso al servidor remoto del MDM (elaboración propia) 44
Figura No. 3.7 Diseño general del esquema XSD (elaboración propia) 45
Figura No. 3.8 Diseño de clases para recibir las tareas del archivo XML (elaboración propia) 46
Figura No. 3.9 Diagrama casos de uso del servicio ejecución de tareas (elaboración propia) . 48
Figura No. 3.10 Diagrama de secuencias del servicio windows (elaboración propia) 51
Figura No. 3.11 Servicio windows en ejecución (elaboración propia) 52
Figura No. 3.12 Protocolo para comunicación con UDP (elaboración propia) 53
Figura No. 3.13 Diagrama de clases del administrador de tareas (elaboración propia) 53
Figura No. 3.14 Diagrama de secuencias del administrador de tareas (elaboración propia) 56
Figura No. 4.1 Servicio web en ejecución (imagen tomada de la pantalla en ejecución) 59
Figura No. 4.2 Ejemplo de prueba realizada en un navegador de internet (imagen tomada de la pantalla en ejecución) 60
Figura No. 4.3 Archivo del log de errores del servicio web (imagen tomada de la pantalla con el archivo txt abierto que almacena los errores) 61
Figura No. 4.4 Diagrama mapeo de información extraída del archivo XML (elaboración propia) 63

Figura No. 4.5 Ventana de monitoreo de las tareas a ejecutar (imagen tomada de la pantalla en ejecución) 64
Figura No. 4.6 Servicio windows en ejecución (imagen tomada de la pantalla en ejecución). 65
Figura No. 4.7 Resultado de selección de tareas priorizadas (elaboración propia) 69
Figura No. 4.8 Diagrama del ambiente de pruebas (elaboración propia) 72

Lista de tablas

Tabla 1.1 Descripción general de algunos softwares disponibles en internet..... 14
Tabla 3.1 Descripción del diagrama casos de uso del servicio web 40
Tabla 3.2 Descripción del diagrama casos de uso del servicio windows 49
Tabla 3.3 Descripción del diagrama casos de uso del administrador de tareas 54
Tabla 4.1 Acciones y/o condiciones para pruebas de comunicación del servicio web..... 60
Tabla 4.2 Casos tomados en cuenta para generación de archivos XML de prueba..... 62
Tabla 4.3 Operaciones válidas para ejecutar sobre medidores 62
Tabla 4.4 Acciones y/o condiciones para las pruebas de inicio del servicio windows..... 65
Tabla 4.5 Registros muestra para la extracción de tareas por prioridad 68
Tabla 4.6 Casos de prueba para el Thread pool 70
Tabla 4.7 Casos de prueba en la liberación de módulos 72

Lista de gráficos

Gráfica 4.1 Tiempos utilizados para probar la activación de tiempos para la creación de thread pools (elaboración propia) 66

Capítulo 1 Introducción

El Internet de las Cosas, IoT por sus siglas en inglés (Internet Of Things), representa cosas cotidianas que se conectan al internet para poder ser programados eventos en función a las tareas que le sean asignadas remotamente. Si los objetos de la vida diaria tuvieran incorporadas etiquetas de radio, podrían ser identificados y gestionados por otros equipos a través de una red *mesh* de dispositivos inteligentes.

Una red *mesh* permite unirse a terminales que a pesar de estar fuera del rango de los puntos de acceso están dentro del área de alguna tarjeta de red que directamente o indirectamente es parte de un punto de acceso.

Los objetos cotidianos con sistemas inteligentes generan gran cantidad de información, que requiere de un sistema MDM (Meter Data Management) para realizar el procesamiento, estimación, transformación y entrega de reportes de forma oportuna y confiable.

La gestión de flujo de información entre el MDM y la red de dispositivos está a cargo de un Sistema Head End (HES) que mantiene un seguimiento de las comunicaciones y los componentes implicados, solicitando registros o ejecución de funcionalidades.

El desarrollo del sistema descrito en esta tesis está enfocado en el diseño, desarrollo e implementación de la lógica de control del proceso de desarrollo de un HES para una red de dispositivos inteligentes realizado para una empresa del sector eléctrico ubicada en el estado de Querétaro. Esta empresa cuenta con una red de comunicación entre medidores inteligente que permite hacer llegar la información desde un dispositivo hasta el software que lo requiera.

El caso de estudio a tratar será enfocado en la manipulación de los datos que se obtienen de los medidores inteligentes, los cuales generan de forma programada o a petición de usuario información para ser almacenada. Esta información se almacena en una base de datos con la capacidad para guardar todos los registros que se generan.

El sistema se desarrolló bajo la metodología *Building Blocks* que permite la creación de bloques funcionales del proyecto general, dando como resultado un proyecto escalable y con mejor calidad debido a las pruebas realizadas por bloques terminados.

El trabajo se complementó con la metodología ágil SCRUM con iteraciones mensuales en las que se desarrollaron *millestons* propuestos en los componentes de software obtenidos de una arquitectura Modelo Vista Controlador (MVC). Para la recuperación, interpretación y actualización en la base de datos se hizo a través de *Hibernate* haciendo uso de patrones de diseño Java para una programación optimizada.

1.1 Descripción del problema

Una ciudad inteligente parte de una ciudad digital que permite que la ciudadanía participe a través de sistemas que interactúan con el entorno donde viven mejorando la calidad de vida de los habitantes de forma sustentable. Actualmente la generación de información a través de tecnologías inteligentes está creciendo proporcionalmente al incremento de la población, lo que requiere de sistemas escalables que soporten la administración y flujo de información.

Este documento se centra en las solicitudes de información y ejecución de tareas que se requieran realizar a los medidores inteligentes que obtienen información del consumo eléctrico de la infraestructura de Comisión Federal de Electricidad (CFE).

La manipulación de esta información se llama Centro de gestión AMI que está integrado por todos los servidores, equipos periféricos y programas informáticos que administran la operación del sistema de medición de energía eléctrica de infraestructura avanzada (AMI) y centralizada en gabinete (SMCG). Para efectos de la presente especificación, al conjunto de programas informáticos se le denomina Sistema Informático de Gestión AMI (SIG-AMI).

El SIG-AMI requiere características de seguridad informática de hardware y software para asegurar que solamente los usuarios autorizados pueden acceder al sistema, a sus datos, funciones y tener la menor vulnerabilidad posible a ataques cibernéticos:

- Debe interactuar con las interfaces del SICOM (Sistema Comercial de CFE) y SICOSS, los cuales requieren que los comandos que se reciban y/o envíen sean en archivos con formato XML y de acuerdo al estándar CIM (Common Information Model).
- El Sistema de Gestión AMI debe contar con un procedimiento automatizado de respaldo de la información.
- Soportar el acceso de al menos 30 operadores simultáneamente, así como de asegurar su funcionamiento.
- Operar la totalidad de medidores formas y tipo gabinete los cuales conforman el sistema AMI.
- Acceder remotamente, a través de claves para acceso con encriptación de al menos AES 128 bits, en cualquier momento, a todos los medidores que integran el sistema AMI.
- Recolectar de manera automática programada y a petición del operador, desde un medidor, un grupo de medidores hasta la totalidad de medidores en cualquier combinación de cualquier área geográfica de la red en cualquier momento: las lecturas, las alarmas y la información de eventos de los medidores.
- Enviar de manera programada y/o a petición del operador; sincronía de tiempo, instrucciones de corte y conexión a un medidor, un grupo de medidores o a todos los medidores que conforman el sistema AMI, en cualquier combinación de cualquier área geográfica de la red, en cualquier momento, así como la confirmación de que éstos recibieron y operaron la instrucción.
- La base de datos debe soportar lenguaje de consulta estructurado (*structured query language* "SQL").
- Ante una interrupción del suministro de energía eléctrica, el SIG se debe reiniciar en forma automática y funcionar totalmente en un tiempo máximo de 15 min después de reanudado el suministro eléctrico.

- Tener la capacidad de realizar automáticamente el cambio de horario estacional y de manejar al menos 4 zonas horarias.

Especificación Técnica del Cliente CFE G0100-05 6.2 Centro de Gestión AMI:

- Debe ser capaz de configurar hasta 5000 medidores en Tarea Programada a ejecutar a una hora definida por excepción o de forma repetitiva diaria, semanal, mensual, etc., ruta de facturación o 1000 usuarios a los cuales le suministra energía un transformador.
- Debe de soportar un mínimo de 100 tareas programadas o cien rutas de trabajo de facturación que requieran lectura de registro.
- Para el caso de los medidores en gabinete debe ser capaz de asociar uno, dos o tres medidores a un cliente, y poder reconfigurar cuando se realice el cambio de uno de ellos.
- Expedir reportes de lecturas de registros, eventos, alarmas, perfiles de carga, estatus de la red NAN y módulos concentradores de datos con salida por pantalla, impresora, medio electrónico en formato compatible con hoja de cálculo y PDF.
- Listado de medidores que presentaron alguna clase de evento, el cual debe ser seleccionable por el usuario y en base a un rango de fechas también especificado por el usuario. En el detalle de la información se deben incluir las cédulas a los cuales se encuentran asociados los medidores, así como las fechas en que el sistema y el medidor detectaron el evento.
- Debe cumplir como mínimo con las normas IEC 61968 Part 1: Interface Architecture and general Recommendations, IEC 61968-9: Interfaces for meter reading and control, IEC 61970-301: Common Information Model (CIM) Base, IEC 61970-501: Common Information Model (CIM) XML Codification for Programmable Reference and Model Data Exchange e interactuar con sistemas del tipo MDM, CIS, WFM, OMS.

Los requerimientos que se enlistan son estándares y especificaciones solicitados por la empresa en la que se realizó el proyecto.

1.2 Justificación

Existen sistemas que ayudan a la gente a interactuar con parte de su entorno, la tendencia actual es la colaboración ciudadana para la integración de proyectos orientados a obtener como resultado una ciudad inteligente.

La creación de proyectos que faciliten al usuario la gestión y control de dispositivos inteligentes requieren que se desarrollen con metodologías que permitan su adaptación a las exigencias tecnológicas.

La empresa se especializa en el desarrollo tecnológico en infraestructuras eléctricas, pero no cuenta con un sistema adecuado para la administración de una red de dispositivos inteligentes, lo que le ocasiona que invierta mucho tiempo en las actividades diarias de consulta y ejecución de tareas hacia los dispositivos.

Este proyecto aprovecha los recursos actuales de la empresa para realizar el diseño, desarrollo e implementación de un sistema Head-End que permita la administración adecuada de la información de los dispositivos para que la CFE pueda programar las tareas que requiera se ejecuten en los dispositivos que integran la red *mesh*.

Captar datos precisos y relevantes de manera oportuna es esencial para la medición inteligente, que incluye la recolección, transferencia y almacenamiento. Los medidores inteligentes han dado como resultado un enorme aumento en el volumen de información, así como los tipos de datos generados y recolectados, lo que genera oportunidades potenciales para generar valor a partir de estos.

Estos datos pueden ser obtenidos en intervalos de tiempo, así como valores agregados para propósitos de facturación. Los obtenidos a través de intervalos de tiempo proporcionan valores más granulares que abren las posibilidades de análisis de tendencias, ciclos y diferentes análisis

de consumo de tiempo de día. Los eventos y las alertas no se programan, se producen al azar debido a situaciones inesperadas.

Toda esta información se almacena en una base de datos con la capacidad para almacenar lo que se genera. La problemática es obtener los registros de la base de datos y procesarlos de acuerdo a la aplicación que lo solicite y poderla proporcionar con las características correspondientes, el reto radica en la tendencia en el avance tecnológico que abre las posibilidades de un nuevo entendimiento y toma de decisiones que se utiliza para describir enormes cantidades de datos, sea de forma estructurada, no estructurada o semi estructurada que tomaría mucho costo y tiempo analizarlos normalmente.

1.3 Objetivo General

Diseñar, desarrollar e implementar la Lógica de Control del proceso de desarrollo de un Sistema *Head-End* para una red de dispositivos inteligentes que garantice el flujo y registro de información entre el MDM y la red de medidores.

1.4 Objetivos Específicos

- Desarrollar la lógica de control para el proceso de desarrollo de un sistema *Head-End* utilizando un Servicio Web FTP para interacción de información masiva.
- Aplicar los estándares en protocolo de seguridad para el manejo de información.
- Plantear la metodología de la lógica de control para el desarrollo y codificación de un Servicio Web FTP.
- Realizar los métodos y algoritmos necesarios para el procesamiento de información proveniente de la red distribuida de medidores inteligentes.
- Realizar los métodos correspondientes a la entrega de información que requiere la CFE a través de la empresa a la que se le realizó el proyecto.
- Realizar las pruebas de funcionalidad de la lógica de control para el desarrollo y codificación del sistema Head-End.

1.5 Metas

- Desarrollo y codificación de un la Lógica de Control del proceso de desarrollo de un Sistema *Head-End* para una red de dispositivos inteligentes.
- Diseño de la Lógica de Control del proceso de desarrollo de un Sistema *Head-End* para su implementación en un patrón Modelo Vista Controlador.
- Dos publicaciones.

1.6 Pregunta de investigación

¿Es factible diseñar, desarrollar e implementar la Lógica de Control para el proceso de desarrollo de un sistema *Head-End* para una red de dispositivos inteligentes?

1.8 Análisis del estado del arte

La descripción de las investigaciones descritas en este apartado, analizan las tecnologías que interactúan en Internet de las Cosas, el monitoreo de información obtenida de los medidores electrónicos y la seguridad en la comunicación utilizando un Servicio Web.

1.8.1 Internet de las Cosas

Damminda Alahakoon y Xinghuo Yu (2016) realizaron un análisis sobre los factores clave que determinarán el éxito de los medidores inteligentes que se han instalado en países de todo el mundo. Describen las capacidades que debe tener un medidor para ser considerado inteligente (Damminda Alahakoon, 2016)

En el artículo “Sensor activado en Internet de las Cosas para las ciudades inteligentes”, se describen los conceptos de sensor habilitado en Internet de las cosas y describen su potencial en la construcción de una ciudad inteligente (Lucy Sumi, 2016)

El modelo flexible de arquitectura jerárquica de IOT se presenta en el documento “Una revisión sobre las soluciones de Internet de Cosas para el control inteligente de energía en edificios para

aplicaciones ciudades inteligentes" con una visión general de cada componente clave para el control inteligente de energía en edificios para ciudades inteligentes. (Iman Khajenasiria, 2016)

En el futuro las ciudades inteligentes, el consumidor tendrá un papel determinante, posiblemente convirtiéndose en un usuario con perfiles de consumo/generación programados de tal manera que maximice su utilidad. En el artículo Optimización del perfil de consumo en Smart City Vision (Mihaela Teliceanu, 2017) contribuye a la comprensión de los consumidores y los edificios dentro de las ciudades inteligentes (patrones de consumo optimizado, gestión de la demanda, eficiencia energética, todo ello utilizando información de señales de precios).

Con base a los artículos descritos sobre internet de las cosas, se observa la importancia de poder obtener y administrar la información que nos rodea para lograr un mejor aprovechamiento de nuestros recursos. Tomando lectura cuantificada de nuestros consumos a través de medidores.

1.8.2 Monitoreo de la información obtenida de los medidores electrónicos

En la investigación realizada por Chunchi Gu, Hao Zhang y Qijun Chen (2014) (Chunchi Gu, 2014) acerca del diseño e implementación del sistema de recolección de datos de energía utilizando un módulo de Fidelidad Inalámbrica (WiFi) y un transformador de corriente, refiere que un sistema de visualización y análisis de energía es una componente muy útil para ahorrar energía, pero en este sistema que proponen solo obtiene los datos de consumo del medidor que es el primer paso.

El software se divide en configuración de parámetros, el bucle de trabajo principal y las funciones del *Modbus*.

En el primero, los ajustes de parámetros son donde se define la configuración de E/S de los pines del micro controlado y la velocidad en baudios para que se pueda comunicar correctamente con el sensor de electricidad.

En el segundo, el ciclo del trabajo principal con la configuración establecida de los parámetros, la recopilación de datos empieza a realizarse en este ciclo de trabajo de la siguiente forma:

- Inicializa algunos arreglos en blanco para guardar los datos y monitorearlos, tales como voltaje, corriente, potencia, factor de potencia, etc.
- Utiliza una función de registro de lectura para realizar la recolección y guardado de datos.
- Envía todos los datos a los pines configurados.
- Define el tiempo de retardo para comendar con el punto uno.

Con estos pasos se mantiene la recopilación de datos y monitoreo constante. El servidor de destino o la dirección IP se configuran durante la configuración del módulo WiFi.

Y por último las funciones del *Modbus,* tenemos la unidad de sensor de electricidad que utiliza el protocolo *Modbus* estándar para transmitir datos, este protocolo de comunicación es simple y robusto por lo que se ha convertido en un medio comúnmente disponible para conectar diferentes dispositivos industriales.

Como conclusión de este trabajo de investigación se tiene un diseño fácil de configurar y adecuado para la mayoría de los medidores para funcionar correctamente. Esta tecnología tiene auge principalmente en China, menciona que la recopilación de datos aun no es perfecta, y menciona como trabajos futuros una optimización de diseño.

En otra investigación planteada desde las necesidades actuales de los edificios inteligentes, Xiujie Dong y You Zhou (2010) (Xiujie Dong, 2010) plantean el proceso de diseño de un software con diferentes controladores de interfaz periférica, con el protocolo de comunicación RS232 y la implementación de funciones del medidor de electricidad como la recolección en tiempo real, procesamiento, almacenamiento de datos, transmisión inalámbrica y comunicación por computadora.

El software fue implementado en lenguaje C en entorno Nios II IDE, herramienta de desarrollo de Eclipse C/C++. El diagrama de flujo del software utiliza la programación modular. Primero, el sistema inicia cada módulo, para leer los datos del medidor regularmente y almacenarlos. Si la dirección que recibe es correcta, empaqueta los datos y los envía. Si la dirección no es correcta, se comprueba nuevamente desde el medidor, para completar el envío y decepción de datos se utiliza el modelo *ShockBurst*. En este modelo, cuando el controlador recibe el comando de llamada, envía la dirección y datos válidos a nRF905 a través de la interfaz SPI t activa la transmisión de datos para envío y para alta. El sistema suele estar en modo espera.

Para la programación de software de dispositivos de lectura móviles, la idea: al terminar el sistema la inicialización, que envíe comandos a la terminal de lectura del medidor a través de la comunicación inalámbrica, cuando el sistema anfitrión reciba la orden, estará listo para recibir los datos. Por último, el sistema enviará datos almacenados del medidor a la computadora *host*, donde la comunicación con la computadora *host* usa el modo de interrupción.

Debido a que el sistema puede realizar la expansión rápida y la capacidad de actualización en línea, recude el tiempo y el costo de actualización del producto, elimina la necesidad de programación de software y operaciones de corrección de errores de microcontrolador, con lo que reduce la dificultad del desarrollo de aplicaciones inalámbricas.

Damminda Alahakoon y Xinghuo Yu (2016) realizaron un análisis sobre los factores clave que determinarán el éxito de los medidores inteligentes que se han instalado en países de todo el mundo desde principios del año 2000, a través de su artículo “Inteligencia de Datos para el Medidor Inteligente de Electricidad para Futuros Sistemas de Energía: Una Encuesta”. Mencionan que uno de los factores es el análisis de datos de medidores inteligentes, trata de adquisición, transmisión, procesamiento e interpretación de datos que se encuentra en las redes inteligentes (SG), que existen muchas definiciones para las SG, pero un *framework* conceptual comúnmente utilizado en el Instituto Nacional de Estándares y Tecnología (NIST), define siete dominios importantes: generación, transmisión, distribución, clientes, proveedores de servicios, operaciones y mercados.

Los medidores evolucionaron desde los medidores leídos manualmente a los medidores con Medición Integrada Avanzada (AMI) con interfaces de tablero y con capacidad de comunicación bidireccional, las capacidades que debe tener un medidor para ser considerado inteligente son (Damminda Alahakoon, 2016):

- Captura en tiempo real o casi en tiempo real del uso de electricidad.
- Lectura remota y local del medidor.
- Control remoto del medidor.
- Posibilidad de vincularse a otras fuentes de suministro (gas y agua).
- Capacidad para capturar eventos: estado, calidad (incluido el voltaje).
- Ser interoperables dentro de un entorno SG.

El medidor inteligente es un dispositivo de medición y captura de información conectado al dispositivo de comunicación llamado Gateway que podría recibir y comunicar información en tiempo real del proveedor, ser un punto de control para los aparatos, iniciar y detener el suministro de energía, etc.

Actualmente existen las herramientas necesarias para conseguir la inteligencia de medición, los trabajos relacionados a estas herramientas usan datos de consumo de energía variables en tiempo real que muestran el comportamiento de uso para los consumidores identificando perfiles, detección anómala del consumo, diseño de ofertas, desarrollo de estrategias de mercado o respuestas a políticas de demanda. A través del estudio que realizaron Damminda Alahakoon y Xinghuo Yu (2016) listan algunas actividades de inteligencia de medición más utilizadas:

- Perfiles de consumidores segmentación y análisis de clústeres.
- Pronóstico de carga.
- Inteligencia de precios.
- Captura de irregularidades.

- Inteligencia de medición para apoyar operaciones en tiempo real.

A partir de esta investigación se identificaron algunos desafíos principales en cuanto a la medición inteligente:

- Problemas en el análisis de datos de los Medidores Inteligentes. La monitorización en tiempo real y la analítica centrada en el diagnóstico serían un requisito importante en estos sistemas.
- Medidores Inteligentes y Big Data. "Big Data" es un término utilizado actualmente en el análisis de datos, tiene tres características principales: volumen, velocidad y varianza.
- Medidores Inteligentes y Cloud Computing. Aunque SG permite la generación distribuida y de RE, el control de la generación de energía según la demanda sigue siendo un problema. El principal problema es la seguridad y privacidad con la protección de los datos de medidores inteligentes de uso no autorizado.
- Medidores Inteligentes e Internet de las Cosas. El SG y el medio ambiente que crea se denomina Internet de la Energía. Si hay una adaptación en el comportamiento de los dispositivos producidores en la información que recibe, como el precio de la electricidad, los dispositivos serán capaces de adaptarse, a este ambiente se le llama Internet de las Cosas (IoT) desde una perspectiva SG.

El resultado que obtuvieron Damminda Alahakoon y Xinghuo Yu (2016) de esta encuesta es que el SG y la medición inteligente serán una "forma de vida" en el futuro.

Con el análisis de estos trabajos se pudo identificar las limitaciones actuales en la medición inteligente, en la que los módulos de análisis permitan vincular la amplia gama de herramientas utilizadas para la obtención de información e identificar los principales datos que deben ser administrados en tiempo real, así como su proceso y flujos de trabajo para su visualización en tiempo real.

1.8.3 Sistemas Comerciales de monitoreo de datos de medidores de electricidad

Los softwares se encuentran disponibles para su descarga en la misma página Web del proveedor del medidor electrónico. El software se descarga de manera gratuita con acceso a la aplicación en un periodo de tiempo para que se prueba probar su funcionamiento, al término de este tiempo se debe de pagar una cantidad que varía dependiendo el proveedor.

Existen muchas empresas que venden medidores de energía para realizar una comparativa de los softwares se tomaron algunas como EKM Metering esta empresa ofrece medidores que para datos eléctricos, agua y gas con acceso desde la nube, el software que proporciona es EJM Dash (EKM Metering, 2016). La empresa Schneider Electric cuenta con variedad de medidores para proporcionar el que mejor se adapta a las necesidades de sus clientes, los softwares que proporciona a sus clientes son: PowerLogic ION EEM 4.0, Power Monitoring Expert Data Center Edition y Power Monitoring Expert Healthcare Edition (Schneider Electric, 2016) y EZMeter es una pequeña empresa de gestión familiar, proporciona soporte técnico y equipo con atención personalizada, maneja cualquier aplicación que se necesite y donde sea, ofrece a sus clientes de medidores el EZ Power Suite que consta de tres programas más una base de datos SQLite: EZ Server, EZMeter Access y EZMeter Power Watch (EZ Meter, 2015).

Para realizar una comparativa de los softwares que proporcionan las empresas antes citadas, se realiza una breve descripción en la Tabla 1.1.

Tabla 1.1 Descripción general de algunos softwares disponibles en internet

Software	Descripción	Sistema Operativo	Días prueba	Costo
EKM Dash	Software diseñado para registrar lectura de medidores. Lee medidores que están conectados a la computadora, y si está conectado directamente al EKM Blink USB a RS-485 a través de una red LAN o por internet con el iSerial TCP/IP a Serial Converter o el sistema EKM Push basado en la nuve. Se visualiza la información de forma gráfica, información de los medidores, configuraciones, informes de correo electrónico y lectura de contadores en archivos .csv. Puede generar informes mensuales automáticos a los usuarios de los medidores.	Windows Mac Linux	30	U$30 una exhibición
PowerLogic ION EEM 4.0	Software de gestión de energía empresarial. Incluye sistemas de monitorización y control de potencia sistemas de medición, automatización de subestaciones y sistemas SCADA, sistemas EMS, sistemas de automatización de edificios y procesos, sistemas de facturación de servicios públicos, entre otros. Control personalizado en navegador y herramientas de visualización para monitorear, validar, predecir y controlar los gastos relacionados con la energía. Aseguramiento de calidad de datos, datawarehouse, framework web. Análisis, modelado de energía, asignación de costos, exporta datos a otros negocios empresariales o sistemas de automatización	Windows	n/a	Sin costo
Power Monitoring Expert Data Center Edition	Sistema de gestión de energía que proporciona inteligencia de sistemas de energía para el sistema de distribución de energía de los centros de datos. Monitorea y analiza la infraestructura de distribución de energía eléctrica. Permite agregar aplicaciones adicionales de forma incremental, mientras que un diseño “plug and play” garantiza que las aplicaciones se conectarán sin	Windows	n/a	Sin costo

Continuación Tabla 1.1…

Software	Descripción	Sistema Operativo	Días prueba	Costo
	problemas. Con los estándares abierto funcionará con cualquier software, hardware o sistema que se utilice.			
Power Monitoring Expert Healthcare Edition	Sistema de gestión de energía que proporciona inteligencia de sistemas de energía para sistemas de distribución de energía de asistencia sanitaria. Aumenta la continuidad de los servicios, enfocado a cargas médicas más sensibles. Promueve un mantenimiento preventivo, reduciendo el riesgo de interrupciones. Reduce el tiempo de recuperación en caso de interrupción, con alertas al personal de mantenimiento en tiempo real, indicando donde se encuentra el problema. Cuenta con un motor de alarma que ayuda a evitar escenarios de sobrecarga y sobrecalentamiento en los sistemas de energía. Cuenta con interfaz directa con el sistema de distribución eléctrica en las salas de operaciones para que el personal de mantenimiento pueda solucionar cualquier alarma remotamente. Enfocado a las instalaciones eléctricas de infraestructura hospitalaria.	Windows	n/a	Sin costo
EZ Power Suite	Consta de tres programas más una base de datos SQLite: EZ Server. Se ejecuta en segundo plano con un servicio, lee los medidores y registra los datos en la base de datos EZMeter Access. Aplicación de escritorio para configurar la base de datos y el consumo de energía de informes. Tiene una función con diferentes tipos de facturación, incluido el tiempo de uso EZMeter Power Watch. Servidor web que permite a los usuarios acceder los datos disponibles desde un navegador. Se ejecuta en una red de área local, pero se puede configurar el enrutador para acceso desde internet. Se requiere adaptador USB a RS-485 o adaptador Ethernet a RS-485 o algún tipo de enlace de radio serie.	Windows	90	U$100 anual

Nota. Elaboración propia a partir de las características de los softwares comerciales de monitoreo de datos de medidores.

Se analizaron algunas empresas proveedoras de medidores de energía que cuentan con su correspondiente software de administración de datos que se obtienen de sus medidores.

Estas aplicaciones se enfocan en el control del suministro eléctrico, facilitando al usuario el monitoreo de sus recursos, así como la facturación individual de cada medidor o de forma grupal de la información que es recolectada.

De lo anterior podemos concluir que falta el diseño adecuado de un software escalable que permita la lectura, administración de datos y una interacción con los medidores en general no solo de electricidad.

1.8.4 Seguridad en la comunicación a través de un Servicio Web

Al aprovechar la World Wide Web, el enfoque orientado al servicio reduce la barrera de interoperabilidad de integrar modelos en términos de lenguaje de programación y sistema operativo. Si bien este paradigma se ha aplicado para integrar modelos envueltos con algunas interfaces estándar, este documento considera la interfaz de modelo básico (BMI) como interfaz de modelo. (Peishi Jiang, 2017)

La evolución del paisaje móvil se combina con la naturaleza omnipresente de Internet con su conectividad inalámbrica intermitente y los servicios web. El logro de la fiabilidad del servicio web resulta en una baja sobrecarga de comunicación y en la recuperación de la respuesta adecuada. Los resultados también muestran que el tamaño de la solicitud se encontró que era constante, que el tamaño de la respuesta era idéntico a la arquitectura tradicional y que el aumento en el tiempo de consumo era inferior al 5% del tiempo de transacción con los diferentes tamaños de respuesta. (Amr S. Abdelfattah, 2017)

La arquitectura orientada al servicio desempeña un papel vital que facilita el marco del Servicio Web. Un Servicio Web permite la disponibilidad de cualquier pieza del módulo en Internet. Cualquier aplicación autónoma se puede exponer como un servicio web y es accesible mediante el mecanismo XML estándar. (Tejal Ghadge, 2016)

Gajanan P. Bherde y Dr. M. A. Pund describen el ataque de aplicaciones basadas en XML, incluyendo la inyección de Xpatth, la inyección de Xquery y la inyección de XSS (Gajanan P. Bherde, 2016)

Como conclusión de estos artículos se puede decir que se deben implementar las herramientas seguridad que corresponden a una comunicación a través de Internet debido a los constantes ataques cibernéticos que realizan las personas para obtener la información desprotegida.

1.8.5 Inteligencia de Negocio

La Inteligencia de Negocios BI por sus siglas en inglés, según el Data Warehouse Institute lo define como la combinación de tecnologías, herramientas y procesos que permiten transformar los datos almacenados en información, esta información en conocimiento y este conocimiento dirigirlo a un plan o estrategia comercial.

En el artículo Business Intelligence Revisited (Jianwen Su, 2017) analizan que un *framework* eficaz y de propósito general para apoyar a los analistas de negocios puede construirse sobre la noción clave de un registro de flujo de trabajo. En el modelado de procesos casi no prestan atención a los datos de modelado que se utilizan en sus registros y estos proporcionan una estructura lógica que puede ocultar detalles del sistema dependiente de la aplicación de las tareas de análisis empresarial. Esta separación permite enfocar las estrategias de división para evitar entrar en los detalles de los registros utilizados por el DBMS.

La implementación de un modelo con técnicas DW facilita la agregación de la información del mercado de electricidad en todos los niveles deseados. El documento “A hybrid approach to building a multi-dimensional business intelligence system for electricity grid operators” (Jelena Luki, 2016) habla sobre la relevancia de DW/BI para la electricidad y otros servicios públicos y sus respectivos procesos (Sinnexus , 2016).

El enfoque de adaptar una metodología actual se realizó cuando el desarrollo se enfrentó con los desafíos de implementar un sistema BI capaz de soportar las necesidades específicas de un mercado de la energía.

Derivado del análisis anterior, se puede concluir que la aplicación de Inteligencia de Negocio en el área de dispositivos inteligentes, será de gran ayuda para el analista de información que deberá visualizar gran cantidad datos. Estos datos procesados a través de BI se podrán ver como información tangible para poder desarrollar a partir de esta un plan de acción.

Capítulo 2 Marco Teórico

2.1 Planeación de proyectos

La planeación de proyectos es una de las actividades previas al desarrollo del proyecto, es donde se establecen los recursos disponibles, se analizan riesgos y se realiza la distribución del trabajo calendarizado. Con esta actividad, podemos tener un panorama general de la dimensión del proyecto y tener gran parte del control en el transcurso del desarrollo.

Al realizar la planeación del proyecto se debe involucrar a todos los participantes, tanto del lado del equipo de desarrollo (programadores, analistas, etc.) como del lado del cliente (jefes del departamento involucrado, usuario final, gerente general, etc.), para establecer las condiciones del plan lo más real posibles en el periodo de tiempo que se debe realizar el proyecto, una forma de contemplar los suficientes elementos para la elaboración del plan es a través del principio de W^5HH (Roger S. Pressman, 2010) que consta de una serie de preguntas que ayudan a definir las características clave del proyecto y el plan del proyecto:

- ¿Por qué (why) se desarrollará el sistema? Todos los participantes deben valorar la validez de las razones empresariales para el trabajo de software. ¿El propósito de la empresa justifica el gasto de personal, tiempo y dinero?
- ¿Qué (what) se hará? Define el conjunto de tareas requeridas para el proyecto. ¿Cuándo (when) se hará? El equipo establece un calendario de proyecto al identificar cuándo se realizarán las tareas del proyecto y cuándo se alcanzarán los hitos.
- ¿Quién (who) es responsable de cada función? Define el papel y la responsabilidad de cada miembro del equipo de software.
- ¿Dónde (where) se ubicarán en la organización? No todos los roles y responsabilidades residen dentro de los profesionales del software. Clientes, usuarios y otros participantes también tienen responsabilidades.
- ¿Cómo (how) se hará el trabajo, técnica y organizativamente? Una vez establecido el ámbito del producto, debe definirse una estrategia técnica para el proyecto.

- ¿Cuánto (how much) se necesita de cada recurso? La respuesta a esta pregunta se deriva al desarrollar estimaciones con base en las respuestas a las preguntas anteriores.
- ¿Riesgos? Se entiende por riesgo cualquier modificación en el entorno del desarrollo del proyecto, se deben abordar desde una perspectiva realista que permita su conocimiento y el trazado de planes de actuación en caso de que se presenten.

El proceso de planeación es interactivo, como se puede ver en la Figura No. 2.1 y debe actualizarse conforme se va evaluando el avance del proyecto contra el plan, y termina cuando el proyecto es finalizado.

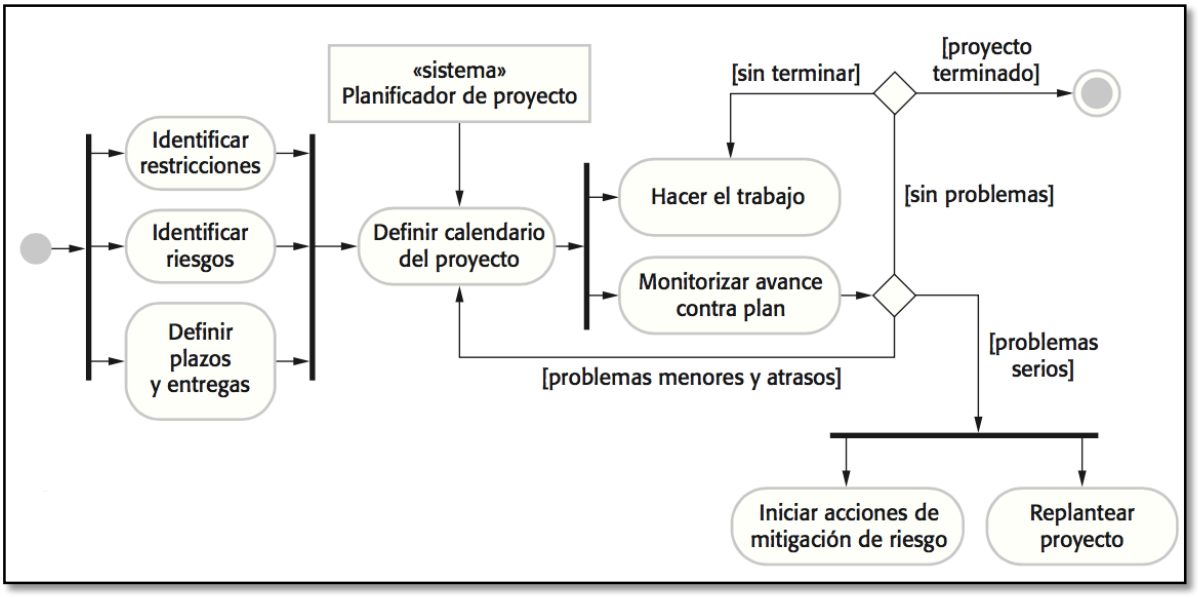


Figura No. 2.1 El proceso de planeación del proyecto (Sommerville, 2011)

Es importante tener una representación gráfica del plan del proyecto a través de: gráficas de barras llamadas gráficas de Gantt, basada en el calendario, las cuales señalan al responsable de cada actividad el tiempo transcurrido previsto y la fecha en que se programó el inicio y el fin de la actividad o por medio de redes de actividad que muestran las dependencias entre las diferentes actividades que constituyen un proyecto.

La estimación del calendario del proyecto no es una tarea fácil debido a que depende de diversos factores que están relacionados con la realización de las tareas, como guía o apoyo para la estimación de tiempos existen dos tipos de técnicas:

- Técnicas basadas en la experiencia: La estimación de los requerimientos de esfuerzo futuro se basan en la experiencia del administrador con proyectos anteriores y el dominio de aplicación.
- Modelado algorítmico de costo: Se usa un enfoque formulista para calcular el esfuerzo del proyecto con base en estimaciones de atributos del producto.

Antes de iniciar un proyecto, se deben identificar las tareas que se realizarán, ¿quiénes las realizará? y ¿en qué tiempo debe concluirse?, esta identificación de tareas y asignaciones garantiza que el proyecto se realice dentro del tiempo estipulado con su respectivo margen de error que también debe de tomarse en cuenta por los posibles contratiempos no previstos. Así mismo debe de revisarse la planeación periódicamente para realizar los ajustes necesarios.

2.2 Patrón de Diseño Arquitectónico Modelo Vista Controlador (MVC)

El patrón de arquitectura MVC (Modelo Vista Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (Interfaz con el usuario u otro sistema) y el Controlador (controlador del *workflow* de la aplicación).

La arquitectura MVC separa elementos de un sistema, permitiéndoles cambiar de forma independientemente, es decir, es una arquitectura en capas. Este patrón se ilustra en la Figura No. 2.2 que muestra una posible arquitectura en tiempo de operación cuando esta arquitectura se usa para el manejo de la interacción en un sistema basado en la Web. Aquí, la funcionalidad del sistema está organizada en capas separadas, y cada una se apoya sólo en las facilidades y los servicios ofrecidos por la capa inmediatamente debajo de ella.

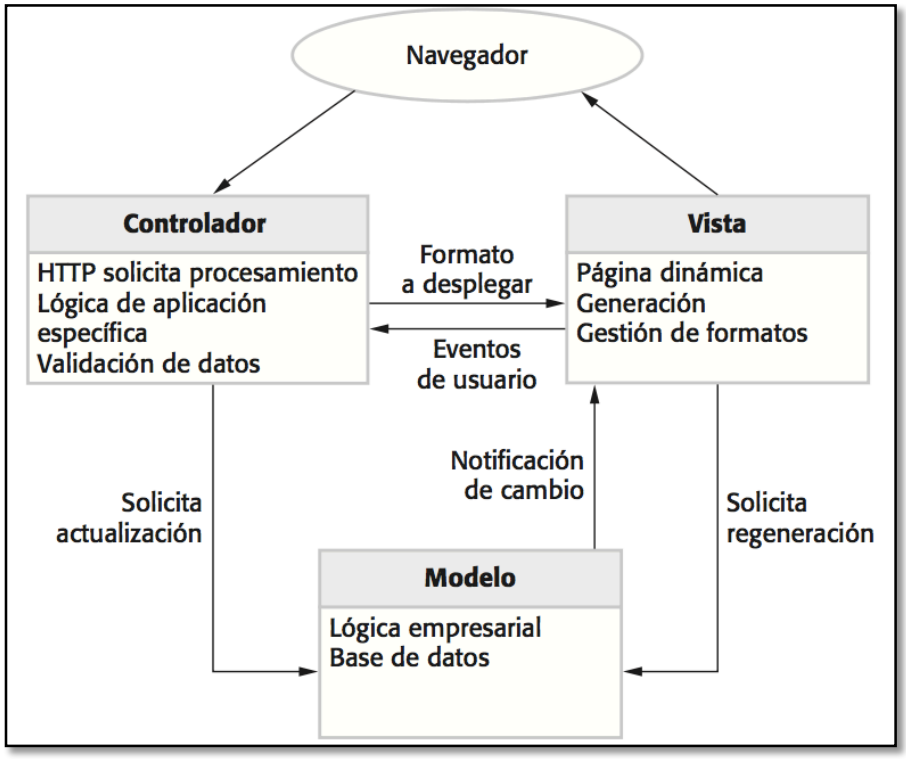


Figura No. 2.2 Arquitectura de aplicación Web con el patrón MVC (Sommerville, 2011)

En MVC separa presentación e interacción de los datos del sistema en una estructura de tres componentes lógicos que interactúan entre sí:

- Modelo. Es la capa donde se trabaja con los datos que habitualmente se encuentran en una base de datos. Los modelos tienen todas las funciones para acceder a las tablas y hacer los correspondientes *selects*, *updates*, *inserts* y *deletes.*
- Vista. Contiene el código de la aplicación que va a producir la visualización de las interfaces de usuario. En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a éstos. La vista solicita los datos a los modelos a través del controlador y estos generan la salida, tal como la aplicación lo requiera.
- Controlador. El controlador contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una búsqueda de información, realizar una compra, etc. Esta capa sirve de enlace entra las vistas y los

modelos, respondiendo a los mecanismos que puedan requerirse para implementar lo requerido por la aplicación.

El acoplamiento entre los tres componentes principales de una aplicación MVC también favorece el desarrollo paralelo. Por ejemplo, un desarrollador de software puede trabajar en la vista, un segundo desarrollador puede ocuparse de la lógica del controlador y un tercero se puede centrar en la lógica de negocios del modelo.

2.3 El componente Controlador

El Controlador es uno de los tres componentes del patrón de diseño Modelo-Vista-Controlador, patrón de diseño para la arquitectura de aplicaciones Web ampliamente adoptado, en muchos lenguajes y *frameworks* de implementación, cuyo propósito es lograr una separación limpia entre sus componentes:

- Modelo: Lógica del negocio y tratamiento.
- Vista: Interfaz de usuario (UI)
- Controlador: Navegación y entrada

La primera responsabilidad de los Controladores es reconocer y manejar la entrada del usuario desde el teclado y el ratón. Cada vista tiene su propio controlador para permitir un manejo correcto de lo que venga desde el teclado y el ratón. Entonces, el controlador decide cómo manejar la entrada y qué parte del modelo debe actualizar. (Kaptein, 2012)

De la definición original no está claro cuál es el alcance exacto del Controlador: se limita a portar la entrada específica una variable específica en el Modelo. La pregunta principal con el patrón MVC es: ¿quién toma las decisiones? En la mayoría de las interpretaciones modernas de MVC, el Controlador es la parte activa, teniendo cuidado de la gestión de esa parte de la aplicación como una cuestión de hablar.

Según el patrón MVC se tiene que:

- El controlador trabaja con elementos y actualizaciones de los elementos GUI. El controlador es la parte del código que, literalmente, toma la entrada del usuario desde cualquier dispositivo y luego actualiza el elemento GUI que corresponda. Puede ser una barra de desplazamiento, un botón o un campo de texto. El controlador es un elemento muy duro, a nivel de máquina, que utiliza la estructura del controlador en la aplicación para ver dónde deben terminar estas acciones de usuario en la interfaz de usuario.
- El controlador actualiza el modelo en la entrada de usuario. El controlador actualiza el modelo con la entrada de usuario: pasando las pulsaciones de teclas y los clics del ratón donde son necesarios.
- El controlador no interviene en procesos internos. Lo que el controlador no hace es el control de procesos en la aplicación. Para eso se tienen otros objetos y partes dentro del Modelo.

El controlador es pare de un patrón separado, como un árbol controlador. Este árbol contiene todos los controladores en la jerarquía de los que aparecen en la aplicación, para cada acción del usuario, el árbol es recorrido hasta que se encuentre el controlador que se necesita.

Los controladores activos de cada proyecto forman un árbol jerárquico. En la raíz de este árbol está la variable global, que es un Administrador del Controlador conectado al proyecto activo.

La ramificación del árbol son los controladores de cada ventana activa, además de un controlador adicional que gestiona el sistema principal. Puesto que cada vista está asociada con un controlador único, el árbol vista/subvista induce un árbol de control paralelo dentro de cada vista activa.

El Administrador del Controlador de nivel superior pregunta a cada uno de los controladores de las vistas activas si lo desea controlar. Solo la persona cuya vista contiene el cursor responde afirmativamente y se le da control. A su vez, consulta a los controladores de las subvistas. Una

vez más el que contiene el cursor acepta el control. Este proceso encuentra la vista anidada más interna que contiene el cursor y, en general, el controlador de esa vista conserva el control mientras el cursor permanezca en su vista.

El controlador se utiliza principalmente como un puente entre el usuario y la vista activa para permitir al usuario introducir y manipular esa vista. En resumen, el controlador se encarga de manejar las acciones del usuario como clics de botón, pulsaciones de teclas y cambios en los valores debido a las acciones del usuario. Todos los procesos son entonces delegados al Modelo.

2.4 Tecnologías de controladores

Existen numerosos *frameworks* de aplicación que se pueden utilizar para proporcionar la capa de control en una aplicación de arquitectura MVC. Estas incluyen: Java Server, Spring

2.4.1 Java Server Faces (JSF)

JSF es un *framework* MVC (Modelo-Vista-Controlador) basado en API de *Servlets* que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el *framework Facelets*. *Facelets* se define en la especificación 2 de JSF como un elemento fundamental de JSF que proporciona características de plantillas y de creación de componentes compuestos.

JSF utiliza las páginas *Facelets* como vista, objetos Javabean como modelos y métodos de esos objetos como controladores. El *servlet* FacesServlet realiza toda la tarea de procesar las peticiones http, obtener los datos de entrada, validarlos y convertirlos, colocarlos en los objetos del modelo, invocar las acciones del controlador y renderizar la respuesta utilizando el árbol de componentes. (Artificial, Introducción a JavaServer Faces, 2014)
JSF proporciona las siguientes características destacables:

- Definición de las interfaces de usuario mediante vistas que agrupan componentes gráficos.

- Conexión de los componentes gráficos con los datos de la aplicación mediante los denominados beans gestionados.
- Conversión de datos y validación automática de la entrada del usuario.
- Navegación entre vistas.
- Internacionalización.
- A partir de la especificación 2.0 un modelo estándar de comunicación Ajax entre la vista y el servidor.

Para poder definir el código del Controlador que se ejecuta cuándo el usuario realiza alguna acción sobre algún componente se debe tener claro la diferencia entre dos tipos de acciones, las acciones del componente y las acciones de la aplicación. En el primer tipo de acciones, es el propio componente el que contiene el código (HTML o JavaScript) que le permite reaccionar a la interacción del usuario.

En el caso, por ejemplo, de un menú que se despliega o un calendario que se abre. En este ejemplo no hay ninguna petición al controlador de la aplicación para obtener datos o modificar algún elemento, sino que toda la interacción la maneja el propio componente. Con RichFaces y JSF 2.0 es posible utilizar eventos JavaScript para configurar este comportamiento.

Las acciones de la aplicación son las que determinan las funcionalidades de negocio de la aplicación. Se trata de código que queremos que se ejecute en el servidor cuando el usuario pulsa un determinado botón o pincha en un determinado enlace. Este código realizará llamadas a la capa de negocio de la aplicación y determinará la siguiente vista a mostrar o modificará la vista actual.

En el controlador se define mediante métodos de los *beans* ligados a acciones de la vista. La acción a ejecutar se define en el código del método y la vista resultante depende de la cadena devuelta y del fichero de configuración faces-config.xml (Artificial, El MVC en JavaServer Faces, 2014)

2.4.2 Spring

Spring es un *framework* de tecnologías estándar en aplicaciones JavaEE. Desde un punto de vista genérico, Spring se puede ver como un soporte que nos proporciona tres elementos básicos:

- Servicios Enterprise: se puede hacer de manera sencilla que un objeto sea transaccional, o que su acceso esté restringido a ciertos roles, o que sea accesible de manera remota y transparente para el desarrollador sin tener que escribir el código de manera manual. En la mayoría de los casos solo es necesario anotar el objeto.
- Estereotipos configurables para los objetos de nuestra aplicación: se puede anotar clases indicando por ejemplo que pertenecen a la capa de negocio o de acceso a datos. Se dice que son configurables porque podemos definir los estereotipos "a medida": por ejemplo, definir un nuevo estereotipo que indicará un objeto de negocio que además sería atrapado automáticamente y con acceso restringido a usuarios con determinado rol.
- Inyección de dependencias: La inyección de dependencias permite solucionar de forma sencilla y elegante cómo proporcionar a un objeto cliente acceso a un objeto que da un servicio que este necesita. Por ejemplo, que un objeto de la capa de presentación se pueda comunicar con un objeto de negocio. En Spring las dependencias se pueden definir con anotaciones o con XML.

Spring puede gestionar el ciclo de vida de los objetos. Los objetos gestionados por el *framework* se denominan genéricamente *beans* de Spring. Esto es lo que sucede por ejemplo con los servlets, normalmente no los instancia el desarrollador, sino que lo hace el contenedor web cuando es necesario. Spring extiende esta idea permitiendo gestionar el ciclo de vida de cualquier objeto. Para ello se tiene que hacer uso de las anotaciones o crear un fichero de configuración XML, aunque esta opción tiende a estar en desuso.

Spring se acopla a la aplicación que se esté desarrollando sin la obligación de modificar el código para utilizar las funcionalidades y beneficios que ofrece. Para utilizar Spring no es imprescindible implementar un interfaz propio o heredar de una clase propia, esta característica

no se puede realizar en otros *frameworks* debido que para poder utilizarlos se tiene que implementar un interfaz propio de ese framework y a la vez implementar los métodos establecidos en dicho interfaz.

Gracias a Spring, el código Java puede ser más limpio, elegante y reutilizable debido a que la filosofía de Srping guía al programador a realizarlo el código orientado a interfaces, de modo que toda la aplicación sea altamente modular ocasionando que la programación se realice orientada a Aspectos, es decir se aplican aspectos al código basado en la funcionalidad extra a las clases y/o métodos sin alterar ni el comportamiento ni el código original del método o clase.

Esta funcionalidad extra que se cruza por varios puntos de nuestra aplicación (*logging*, seguridad, etc.) es lo que se conoce como *Cross-Cutting Concerns* y la programación orientada a aspectos permite mantener dos principios básicos en un buen diseño: SoC y DRY.

SoC (Separation of Concerns) y DRY (Don't repeat yourself) ayudan a crear e implementar clases y métodos que sigan el patrón 1:1, es decir, "Una determinada funcionalidad debe estar implementada en un solo sitio de la aplicación y sólo debe llevar a cada esa funcionalidad". (Walls, 2015)

2.5 Metodología SCRUM

SCRUM es un método de desarrollo ágil de software concebido por Jeff Sutherland y su equipo de desarrollo a principios de la década de 1990. En su proceso se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente y obtener el mejor resultado posible de un proyecto.

En SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto, por ello es indicado para proyectos en entornos complejos, donde se necesita obtener prontos resultados, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Los principales beneficios que proporciona SCRUM son:

- Entrega mensual o quincenal de resultados de los requisitos más prioritarios en ese momento, lo cual proporciona las siguientes ventajas:
- Gestión regular de las expectativas del cliente y basada en resultados tangibles
- Resultados anticipados
- Flexibilidad y adaptación respecto a las necesidades del cliente
- Gestión sistemática del retorno de inversión
- Mitigación sistemática de los riesgos del proyecto
- Productividad y calidad
- Alineamiento entre el cliente y el equipo de desarrollo
- Equipo motivado

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos, cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado en el mínimo esfuerzo al cliente cuando lo solicite.

Las actividades que se llevan a cabo en SCRUM son:

- Planificación de la iteración: El primer día de la iteración se debe presentar lista de requisitos priorizada del producto o proyecto. Elaborar la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido.
- Ejecución de la iteración: Cada día el equipo realiza una reunión de sincronización. Cada miembro del equipo inspecciona el equipo que está realizando el resto, dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que puedan impedir este objetivo, para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido.

- Inspección y adaptación: El último día de la iteración se realiza la reunión de revisión, donde el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para entregar con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva. Con este resultado, el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. (Proyectosagiles.org, s.f.)

2.6 Maven

Maven es una herramienta para la gestión de proyectos de software, que se basa en el concepto de POM (Proyect Object Model), es decir, con Maven se puede compilar, empaquetar, generar documentación, pasar los test, preparar *builds*, entre otros.

Maven se basa en patrones y en estándares que permite a los desarrolladores moverse entre proyectos sin necesitar aprender como compilar o empaquetar. Esto mejora el mantenimiento y la reusabilidad. Dentro del archivo POM.xml se hace una descripción del proyecto, ahí se especifica nombre, versión, librerías de las que depende y Maven se encarga de hacer todas las tareas para la gestión de librerías.

Incluso Maven toma en cuenta las dependencias transitivas, es decir, si A depende de B y B depende de C, es que A depende de C. Esto quiere decir que cuando se empaqueta A, Maven se encargará de incluir tanto B como C en el paquete.

Para la creación de proyectos y arquetipos Maven se basa en plantillas, es decir Maven es capaz de generar una estructura de directorios y ficheros para la creación del tipo de proyecto que se va a iniciar.

Al crear un nuevo proyecto se debe de asignar un *grupoId* que es el identificador único de la organización o grupo que crear el proyecto, se podría decir que es el identificador de la aplicación. Y con el *artifactId* se asigna el identificador único del artefacto principal del proyecto, se podría decir que es el identificador del módulo dentro de la aplicación, es decir, este será el nombre del jar.

Como se dijo anteriormente, en el archivo pol.xml se describe el proyecto, con la etiqueta *packaging* se indica el tipo de empaquetado que hay que hacer con el proyecto (jar, warm ear, pom) Con versión se indica la versión del proyecto con la que se está trabajando. Al indicar SNAPSHOT se quiere decir que es una versión evolutiva, es decir que se está trabajando para obtener la versión 1.0.

También se puede describen las dependencias del proyecto con la etiqueta *dependencies* para agrupar todas las dependencias que utilizará el proyecto y con la etiqueta *dependency* se especifica cada una de ellas indicando el *grupoId*, *artifactId*, *version* y *scope*. (Apache.org, 2017)

2.7 Hibernate

En la actualidad el paradigma utilizado por la mayoría de analistas programadores a la hora de construir software es el orientado a objetos; independientemente de los avances en esta área, la base fundamental es este estilo de programación. Partiendo de esta afirmación, toda aplicación estaría conformada por una gran cantidad de objetos interactuando y compartiendo información, y al momento de que estos objetos tengan que trascender inician una serie de problemas al escoger un modelo para persistir.

La palabra persistencia viene del latín “persistere” que significa durar por largo tiempo. En el mundo de los objetos todo se maneja en memoria volátil, esto quiere decir que los objetos creados en un programa siempre son temporales, porque son almacenados en la memoria RAM.

Lo que busca la persistencia es guardar un objeto permanentemente en un recurso de almacenamiento, base de datos o archivo, para recuperarlo más tarde. El almacenamiento más utilizado, son las bases de datos relacionales, el cual consiste en un sistema formado por un conjunto de datos almacenados en disco que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

La persistencia puede ser trabajada desde dos puntos de vista: el programador debe conseguir que sus datos (objetos) sobrevivan a la ejecución del proceso que los creo, de modo que puedan ser utilizados en otro proceso. Esto quiere decir que la persistencia es responsabilidad del programador. El otro punto de vista depende de que un lenguaje de programación o de entorno de desarrollo para almacenar y recuperar el estado de los datos de modo que sobrevivan a los procesos que los manipulan. Actualmente existen muchos de estos entornos y es en donde se ubica Hibernate (Documentation, 2004)

Con la conceptualización anterior se puede definir a Hibernate como: "un *framework* para persistir objetos en una base de datos relacional". La idea de Hibernate es de Gavin King,ingeniero actual del grupo de JBoss, el cual lidiaba con la ineficiencia y complejidad de los sistemas de persistencia de la época, ideo un sistema base que fue apoyado por un grupo de programadores alrededor del mundo. Esta idea de King se transformó en un proyecto robusto con licencia libre que actualmente es el más utilizado en cuando a *framework* dedicados a persistencia se refiere.

Hibernate es un potente servicio de persistencia Objeto – Relacional de alto rendimiento que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Ofrece persistencia automatizada y transparente de objetos a tablas en una base de datos relacional, utilizando metadata que describe el mapeo entre objetos y la base de datos en si. Entre las características más importantes destacan: Licencia LGPL, ofrece su propio lenguaje de consulta HQL, implementado con XML o anotaciones (JPA), excelente documentación, fácil de aprender y comunidad activa.

Utilizar Hibernate proporciona las siguientes ventajas:

- Productividad: Evita mucho del código confuso de la capa de persistencia, permitiendo centrarse en la lógica de negocio.
- Mantenibilidad: Por tener pocas líneas de código permite que el código sea más claro. Al dividir la capa de persistencia se puede identificar los errores muy fácilmente.
- Rendimiento: Existe la tendencia a pensar que una solución "manual" es más eficiente que una "automática". Hibernate tiene un buen desempeño, pero todo depende realmente de cómo se realicen las consultas y como se configure el *framework*.
- Independencia del proveedor: Una solución ORM (Object Relational Mapping) te abstrae del SGBD (Sistema de Gestión de Base de Datos) permitiendo desarrollar en local con bases de datos ligeras sin complicación en el entorno de producción.

Para utilizar Hibernate es importante saber que este se compone de un conjunto de librerías reunidas en varios Apis, en donde, dependiendo del problema se debe escoger y configurar las librerías necesarias en el proyecto (Christian Bauer, 2007).

Del conjunto de Apis de Hibernate existen varias clases que permiten el trabajo básico con el *framework*. Entre las que se encuentran:

- Session: corresponde con un objeto que representa una unidad de trabajo con la base de datos (transacción). Además, representa el gestor de persistencia, ya que dispone de la API básica para poder cargar y guardar objetos.
- Transaction: La API de Hibernate contiene utilidades para demarcar la transaccionalidad de operaciones de manera programática.
- Query: Este interfaz permite crear consultas y enlazar argumentos a parámetros de la consulta. Permite definir consultas en HQL (Hibernate Query Language) o en SQL.
- SessionFactory: Es una factoría de sesiones. Proporciona objetos Session. Permite concurrencia. (Bauer & King, 2007)

2.8 Servicio Web

Un Servicio Web es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre programas. Distintas aplicaciones de software desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Los Servicios Web representan la evolución informática de la nueva generación de aplicaciones que trabajan colaborativamente en las cuales el software está distribuido en diferentes servidores; de esta manera se logra la creación de grandes aplicaciones que pueden funcionar en una gran multitud de dispositivos, entre los que se encuentran los teléfonos móviles, las tabletas y computadoras personales, todos ellos interactuando con un servidor. (Docs.oracle, s.f.)

Para la descripción de servicios web se utiliza WSDL (Web Services Description Lenguage) que es un formato XML. WSDL describe la interfaz pública a los servicios web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.

El protocolo estándar que se utiliza para enviar la información, es SOAP (Simple Object Access Protocol). Este define el formato del “envelope” que se intercambia entre cliente y servicio, así como las convenciones para representar invocaciones y respuestas, estos mensajes son transmitidos en formato XML, montado sobre http. (Jayasinghe & Azeez, 2011)

Capítulo 3 Metodología

Se ha demostrado que los proyectos exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar el proyecto, dependiendo sobre todo de la comunicación efectiva con los interesados y el manejo de las expectativas con todos los participantes en el proyecto.

Un modelo para el desarrollo de software es una representación abstracta de un proceso. Cada modelo representa un proceso desde una perspectiva particular y así proporcione información parcial sobre el proceso. Puede visualizarse los modelos de desarrollo de software como *frameworks* del proceso que pueden ser adaptados para crear procesos más específicos.

Para el desarrollo del proyecto se utilizó la metodología tradicional incremental adaptada con *building blocks* y la metodología ágil *Scrum*. Por lo que se realizó de forma incremental bloques de código independiente funcional y con su integración se obtuvo la solución final, con la finalidad de que cada bloque de código sea independiente para poder dar un mantenimiento óptimo.

El principal objetivo del proyecto es realizar la comunicación entre la red de medidores (*red mesh*) y los sistemas de CFE, para el desarrollo de esta comunicación se fracciona en varios módulos objeto de esta tesis:

1. Servicio Web FTP: Mantiene la comunicación entre el administrador de dispositivos (MDM), recibe peticiones para ser enviadas a la red distribuida y regresa la respuesta recibida de la red.
    - Levantar Servicio Web
    - Conexión y acceso con servidor remoto para obtener archivo XML
    - Validación de archivos XML
    - Guardar tareas en base de datos Oracle

2. Servicio de ejecución de tareas: Mantiene la comunicación entre la red distribuida y las tareas que se deben ejecutar.
   - Mantener en ejecución servicio Windows
   - Enviar tarea a red distribuida para ser ejecutada
3. Manejador de tareas: Administra las tareas que deben ser ejecutadas, controlando las excepciones que se generen para continuar con las tareas sin perder la constancia y coherencia en la ejecución.
   - Obtener tareas priorizadas
   - Control de excepciones para el manejo de interrupción de tareas
   - Generación de archivo XML de respuesta de la red distribuida

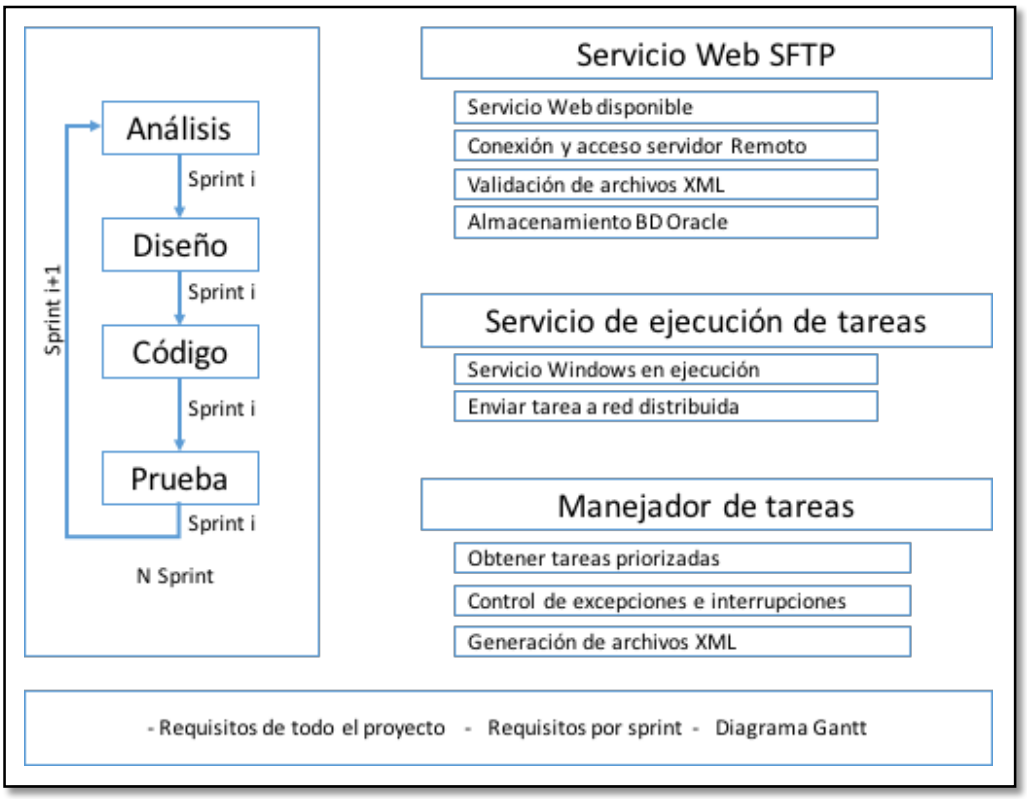


Figura No. 3.1 Metodología de desarrollo de la Lógica de Control del Proceso de Desarrollo de un Sistema Head End para una Red de Dispositivos Inteligentes (elaboración propia)

En la Figura No. 3.1 se puede observar la aplicación de la metodología incremental por cada módulo que integra el proyecto, donde la documentación para el control durante el desarrollo fue:

- Requisitos de todo el proyecto. Contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno de utilidad.
- Requisitos por iteración. Subconjunto de requisitos para ser desarrollados en la iteración (sprint).
- Diagrama de Gantt. Donde se visualizan los pendientes al comienzo de cada sprint.

La implementación de estas metodologías, se realizan en procesos para cumplir con los objetivos de entrega en fechas establecidas, por cada iteración se realiza: la comunicación, planeación, modelado (análisis, diseño), construcción (código, prueba) y despliegue (entrega, retroalimentación) con entregables funcionales.

Todos los días antes de iniciar actividades se realizan reuniones rápidas con una duración máxima de 15 minutos para dar seguimiento a las preguntas como: ¿qué hiciste ayer?, ¿qué vas a hacer hoy? y ¿que necesitas para terminar tu trabajo? Se elabora una lista de actividades con fechas de entrega para llevar el control de objetivos, metas y lo que falta por hacer; indicando el porcentaje de avance de las actividades que se están realizando.

El uso de metodologías ayuda a establecer orden en la forma de trabajar, ver el estado actual de las fases del proyecto, pendientes para finalización y corrección de errores, lo que da como resultado un producto de calidad.

La empresa para la que se desarrolló el proyecto utiliza el IDE de programación Eclipse que emplea *plug-in* para agregar funcionalidades de acuerdo a lo que se requiera desarrollar, se agregan las librerías de java y los *frameworks* de desarrollo. Se realizó el sistema bajo la estructura de proyecto Maven para el control de repositorios de las librerías de java. Se utiliza el *framework* Spring para la organización del proyecto en MVC.

Este proyecto es el Controlador encargado de la comunicación entre la base de datos y las pantallas que visualiza el usuario, también contiene los métodos que interactúan entre procesos detonados a partir de las peticiones realizadas por el MDM el detalle de este funcionamiento se encuentra en el trabajo "Modelado e implementación de un Data Warehouse para un MDMS utilizando algoritmos ETL y VEE" (Carmona, 2018).

La estructura del proyecto requirió de la integración con los sistemas:

- Vista, encargado de la interacción con el usuario. Este trabajo se encuentra en la tesis "Vista Diseño e Implementación de una Interfaz Gráfica de Usuario para un Sistema Head End de una Red Distribuida de Medidores Inteligentes" (Guzmán, 2018).
- Y Modelo que refiere a la estructura del almacenamiento de los datos (base de datos), detallado en el trabajo "Diseño e implementación de una base de datos relacional utilizando ORM programing technique, Spring Security e Hibernate para un Head End System" (Hernández, 2018).

3.1 Servicio Web FTP

El servicio web está a la escucha de la solicitud por parte del MDM, para la extracción de un archivo XML del servidor remoto y validar el archivo XML contra el esquema XSD definido, posteriormente realizar el mapeo de los datos y crear los objetos para almacenar las tareas en la base de datos para que sean procesadas posteriormente, en la Figura No. 3.2 se observa el diagrama de flujo del servicio web.

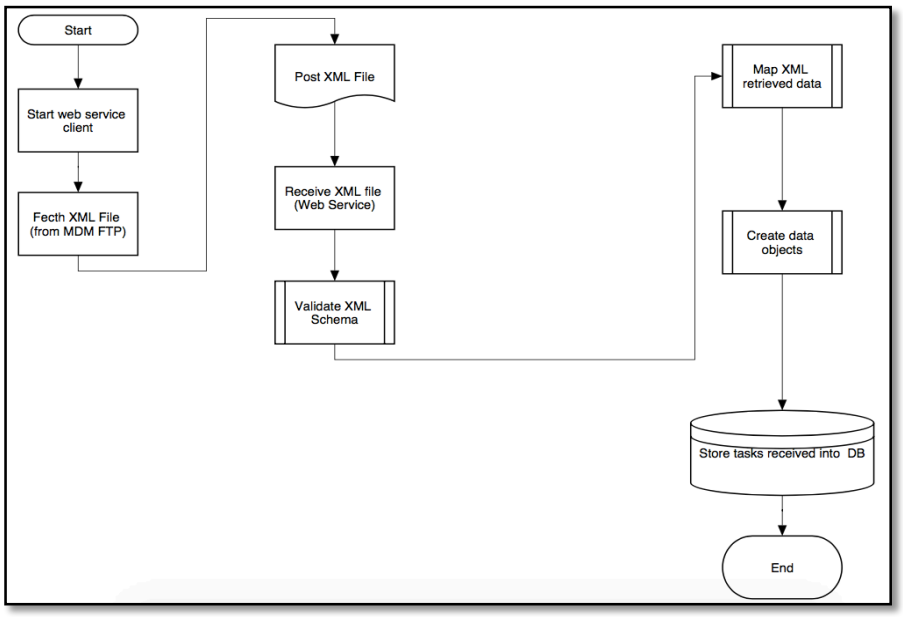


Figura No. 3.2 Diagrama de flujo del proceso general del web service (elaboración propia)

Con base al análisis realizado de los procesos requeridos para el funcionamiento del servicio web, se realizó el diagrama de casos de uso para visualizar de quienes son los actores que interactúan y que actividades les corresponden, en la Figura No. 3.3 se observan el diagrama de casos de uso para el servicio web.

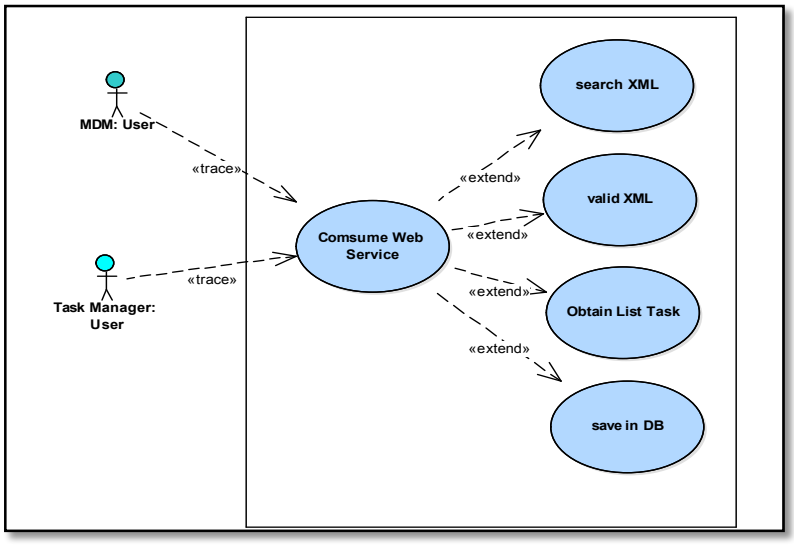


Figura No. 3.3 Diagrama de Casos de Uso del servicio web (elaboración propia)

En la tabla No. 3.1 se describe el diagrama de casos de uso para el servicio web especificando los puntos más importantes como son: los actores principales, objetivos, precondiciones, acción o proceso que dispara el inicio del servicio web, escenarios principales y las excepciones que pudieran ocurrir cuando el proceso este ejecutándose.

Tabla 3.1 Descripción del diagrama casos de uso del servicio web

Caso de uso	**Servicio Web**
Actor principal	MDM, Administrador de Tareas
Objetivo	Administrar la comunicación bidireccional de dispositivos en una red mesh hacia el MDM
Precondiciones	La red mesh de medidores ANSI este arriba, el MDM este escuchando
Disparador	Solicitud de información y tareas programadas
Escenario	
1. Medidor ANSI: Envía registros al MDM	
2. Medidor ANSI: Envía su perfil de carga al MDM	
3. Medidor ANSI: Envía su calendario al MDM	
4. Medidor ANSI: Envía eventos al MDM	
5. Medidor ANSI: Envía alarmas al MDM	
6. MDM: Envía configuración de perfil de carga a Medidor ANSI	
7. MDM: Envía configuración de calendario a Medidor ANSI	
8. MDM: Envía acciones al Medidor ANSI	
Excepciones	
1. La interfaz del headEnd no responde: el Medidor ANSI debe intentar comunicarse nuevamente en determinado tiempo	

Caso de uso	**Servicio Web**
2. El headEnd indica al Medidor ANSI y/o MDM que la información no está completa o no es correcta: El Medidor ANSI y/o MDM deben enviar nuevamente los datos al headEnd	
3. Se queda sin energía eléctrica el headEnd: El headEnd debe de reestablecerse con la última información que estaba procesando	
Prioridad	Esencial, debe implementarse
Frecuencia de uso	Varias veces al día
Canal para el actor	C12.22. Estándar Nacional Americano para Especificación de Protocolo para Interfaz a Redes de Comunicación de Datos CIM. Common Information Module
Actores secundarios	Administrador TI
Canales para los actores secundarios	Interfaz de usuario
Aspectos	
1. Recepción masiva de información.	
2. Envío masivo de información.	

Nota. Elaboración propia

Para mostrar la interacción entre los procesos se realiza el diagrama de secuencias, en la figura No. 3.4 se observa que el cliente MDM inicia el proceso con una petición en un archivo XML a través de escuchador JSON se busca el archivo en el servidor remoto, si lo no lo encuentra el proceso almacena el mensaje de que el archivo no fue encontrado, de lo contrario lo obtiene para realizar la validación de su estructura contra un esquema XSD.

Si el proceso no se encuentra un esquema válido, se genera un mensaje de que el archivo no se admite, de lo contrario se extrae la lista de tareas que contenga para realizar la persistencia en los objetos que corresponden a la base de datos y las peticiones queden disponibles para el administrador de tareas, el proceso devuelve el resultado final de la operación.

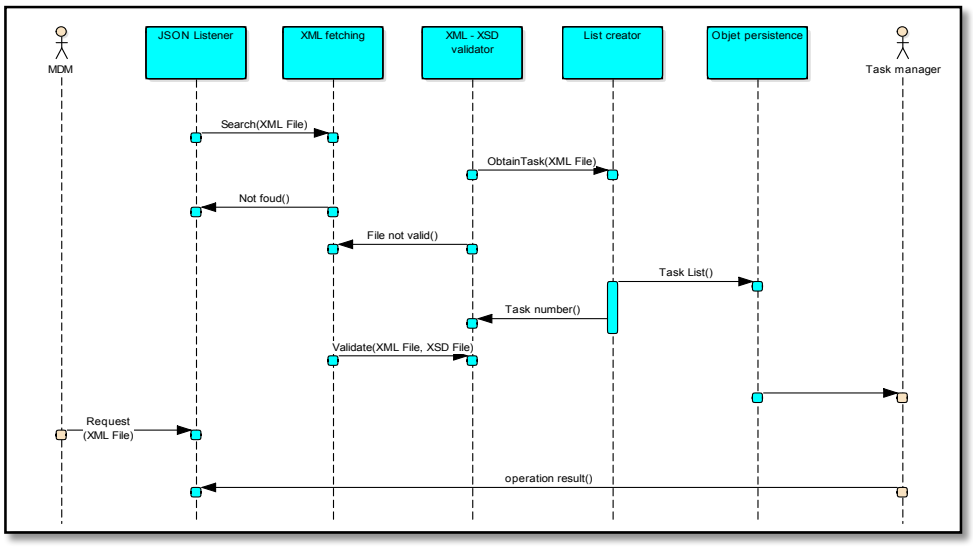


Figura No. 3.4 Diagrama de secuencias del servicio web (elaboración propia)

3.1.2 Levantar Servicio Web

Para realizar el servicio web se utiliza el motor de servicios web Axis y un contenedor de servlets Tomcat, Las librerías de java javax.servlet.jstl y org.apache.axis2 se incluyen en el pom.xml de Maven para poder hacer uso de los métodos en Eclipse (Deepal Jayasinghe).

```
@Path("/ReadValidXML")
    public class borrar {

        @GET
        @Path("/ReadValid") // Name with which the method is identified
        @Consumes({ MediaType.APPLICATION_JSON }) // The service that will consume
                                // the method
        @Produces({ MediaType.APPLICATION_JSON }) // Everything will turn into json

        public String ReadValid(@QueryParam("strFile") String strFile) {
            /*
             * web service code
             */
            return strFile;

        }
    }
```

Figura No. 3.5 Estructura del servicio web (elaboración propia)

Como se puede ver en la Figura No. 3.6 se especifica mediante la anotación @Path la ruta para el servicio web y el método de servicio web, se indica que proporciona un acceso de sólo lectura a un recurso con el método http @GET.

La comunicación se realizará por medio de JSON por lo que se debe configurar a través de las anotaciones @Consumes y @Procedures

El servicio Web FTP desarrollado para el proyecto contiene los siguientes elementos, que son los que debe de tener un servicio WebRESTful (Leonard Richardson):

- URI del recurso. El identificador de recurso uniforme es una cadena de caracteres que identifica los recursos en una red de forma unívoca. Los recursos REST se identifican mediante un URI exclusivo. Por ejemplo: http://webservice/ReadValidXML/ReadValid esto nos da acceso al servicio web ReadValidXML y al recurso ReadValid.
- Tipo de representación de dicho recurso. El tipo de respuesta que regresa puede ser JSON, XML y TXT. Para el objeto de estudio de este documento se utiliza el tipo JSON para comunicar la respuesta del proceso de almacenamiento de la información enviada por el cliente.
- Operaciones soportadas. HTTP soporta varios verbos (tipos de operaciones) que puede ser GET, PUT, POST, DELETE, PURGE entre otros. En el proyecto de *Web Service FTP* se utiliza el verbo POST para que el servicio realice los procedimientos necesarios para almacenar la información extraída de un archivo en base de datos.

3.1.3 Conexión y acceso con servidor remoto para obtener el archivo XML

El MDM utiliza el servicio web para solicitar la ejecución de tarea mediante un archivo XML que se encuentra almacenado en su servidor. El servicio web accede al servidor para extraer el archivo XML con las tareas a ser ejecutadas, por lo que se requiere la integración de un protocolo de transferencia.

Se propuso utilizar el protocolo de transferencia SFTP que permite una serie de operaciones sobre archivos remotos, intenta ser más independiente de la plataforma SCP (Secure Copy Protocol) es más avanzado que FTP, sin embargo, algunos dispositivos pueden no ser

compatibles con SFTP, como los móviles, consolas, etc. por esta situación la empresa solicito se utilizara el protocolo de comunicación FTP.

La problemática del uso de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, debido a que todo el intercambio de información, desde el *login* y *password* del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún cifrado, con lo que un posible atacante puede capturar este tráfico y acceder al servidor o apropiarse de los archivos transferidos.

Para solucionar el problema anterior se planteó que en el Servidor Web FTP se utilice un algoritmo de encriptación de al menos AES 128 bits para proteger la información que viaja a través de archivos XML. Como se muestra en la Figura No. 3.7, el cliente MDM almacena los archivos XML de forma encriptada en su servidor remoto, cuando solicita a través JSON al servidor web que ejecute las tareas que generó, el servidor web obtiene de la base de datos la información necesaria para conectarse al servidor remoto del MDM, obtiene los archivos, los desencripta para procesarlos y nuevamente realiza el proceso de encriptación para almacenar en su servidor los archivos con la respuesta de las tareas que ya fueron procesados.

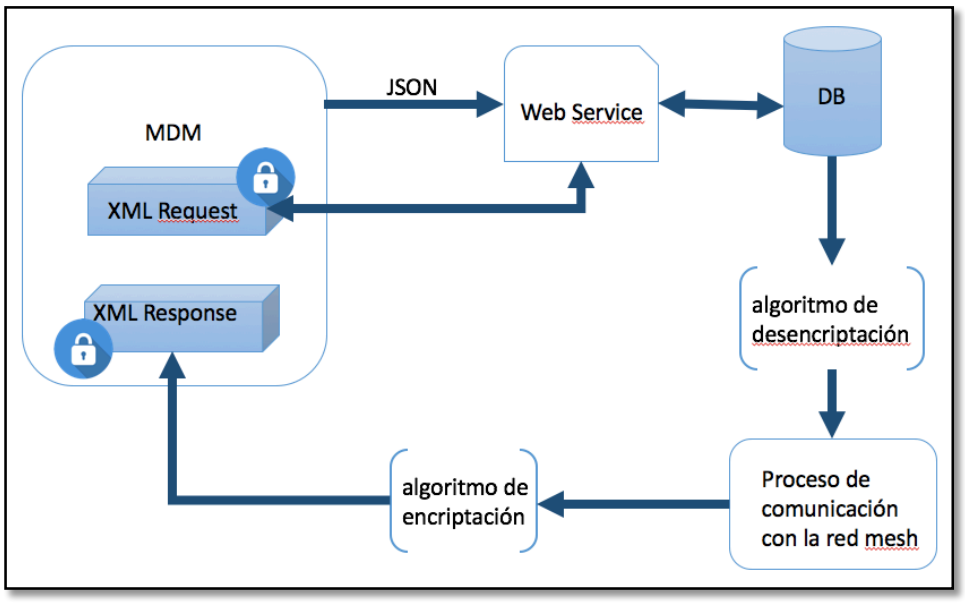


Figura No. 3.6 Conexión y acceso al servidor remoto del MDM (elaboración propia)

Para la realización de los métodos de acceso al servidor remoto se utiliza la librería org.apache.commons.net.ftp.

3.1.4 Validación de archivos XML

El servicio web debe garantizar que los archivos que se extraigan con peticiones de ejecución de tareas, correspondan a un formato XML válido. Para esta validación se realiza una comparación con un esquema XSD que contiene las etiquetas que debe tener el archivo XML.

El esquema se realiza con base al documento G0100-5 (CFE, 2015), que contiene las especificaciones del Sistema de Infraestructura Avanzada de Medición (AMI) donde se encuentran las estructuras de comandos requeridas por la CFE para el proyecto HES en la figura No. 3.8 se muestra el esquema que se realizó.

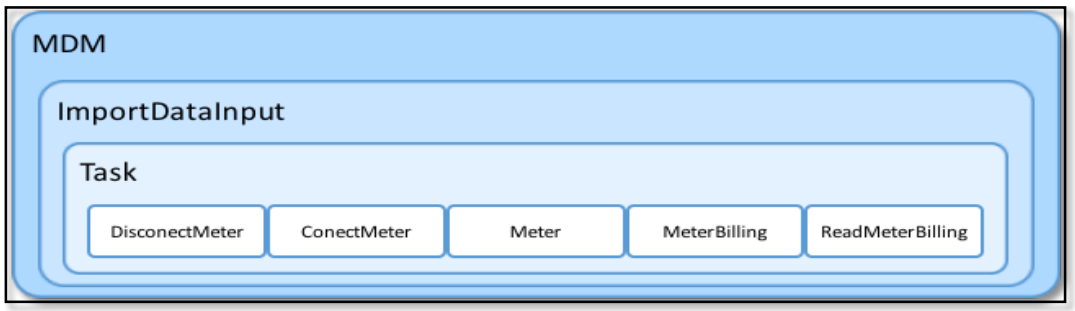


Figura No. 3.7 Diseño general del esquema XSD (elaboración propia)

El proceso inicia cuando el MDM realiza una petición, después de cargar la configuración que corresponda al cliente que solicita, se valida contra el esquema XSD especificado. Una vez que el archivo XML es válido, se procede a extraer las tareas que se ejecutarán en la *red mesh*.

Las aplicaciones Web ofrecen diversos tipos de servicios a los usuarios de Internet. De modo que, la prevención de los servicios web de los ataques se convierte en tarea esencial y desafiante para proporcionar servicios seguros a los usuarios. Existen diversos tipos de ataques como la vulnerabilidad XXE, XSS, inyección de SQL, codificación de solicitudes, ataques DoS, etc.,

estos tipos de ataques en páginas web degradan el rendimiento o el bloqueo completo de servicios a los usuarios. (Amr S. Abdelfattah, 2017)

La vulnerabilidad XXE (Xml eXternal Entity) se produce en aplicaciones que hacen uso de "parsers" XML. Es decir, aplicaciones que reciben como entrada un documento XML y para procesarlo hacen uso de alguna librería de "parseo". En el Servicio Web FTP, objeto de estudio, se utiliza la librería javax.xml.parsers para procesar los archivos XML que recibe el servicio web, para prevenirlo se utilizan las propiedades access_external_dtd y access_external_schema (Oracle.com, s.f.) para evitar que se trate de malformar las peticiones, porque podrían añadirnos funcionalidades al interpretar XML, por ejemplo, inyectando código para leer un archivo de nuestro servidor local y enviar su contenido a un servidor remoto.

3.1.5 Guardar tareas en base de datos Oracle

La extracción de tareas de los archivos XML se almacena en una lista de objetos para que pueda ser manipulada de una manera eficiente. En la figura No. 3.8 se pueden ver el diseño de las clases clsElemento, clsTask, clsAttribute, clsDoTask para poder obtener cada tarea y medidor con sus atributos.

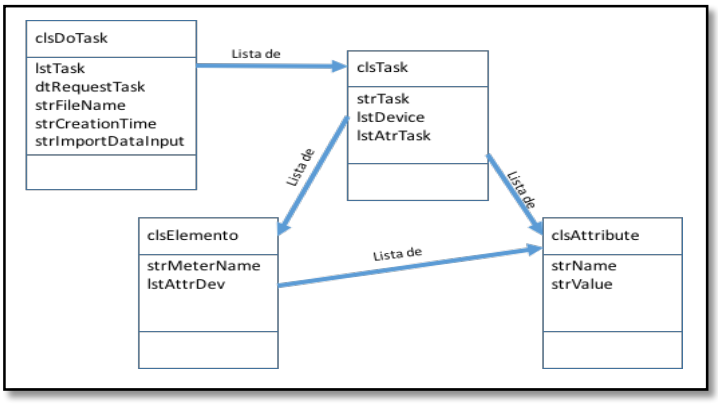


Figura No. 3.8 Diseño de clases para recibir las tareas del archivo XML (elaboración propia)

La estructura con la que se reciben las tareas a través del archivo XML especifica cómo se ejecutarán y a qué dispositivos se debe enviar, cada una contiene atributos y los dispositivos a su vez sus propios atributos. El archivo de petición de ejecución tiene por sí mismo también atributos.

Con base a lo anterior, se define las siguientes clases:

- clsDoTask: contiene sus atributos específicos y una lista de tareas que serán ejecutadas.
- clsTask: contiene el nombre de la ejecución que se realizará, una lista de dispositivos a los que se ejecutará y una lista de atributos que la caracterizan.
- clsElemento: contiene el nombre del medidor y una lista de atributos que lo caracterizan.
- clsAttribute: especifica el nombre del atributo y su valor.

Una vez que se tienen los datos en la lista de tareas, se realiza el mapeo en las entidades donde se almacena la información de su ejecución y a los medidores que se realizará. Las entidades que se utilizan son: BusBar, CCG, Cabinet, DeviceStatusCatalogue, ElectricalDevice, Job, JobDevRelationship, JobStatusCatalogue, MDM, TaskType, UtilityType.

Un gabinete contiene una CCG y tres BusBar; cada BusBar contiene cuatro medidores, a estos elementos se les puede solicitar la ejecución de alguna tarea por lo que en el archivo XML puede estar especificado cualquiera de estos elementos.

El mapeo de los datos se va realizando dependiendo si el dispositivo es un BusBar, CCG o un Gabinete, para almacenarlo en la tabla que le corresponda, su programación se almacena en el Job y en el JobDevRelationship se guarda los dispositivos que afectará y se hace la relación con el MDM que le corresponde cada una.

Para la estandarización de estatus de los medidores y tareas se hace uso de catálogos que corresponden a las tablas DeviceStatusCatalogue y JobStatusCatalogue. En la tabla TaskType se almacenan los tipos de operaciones que pueden ser ejecutados. Cada MDM corresponde a un tipo de servicio que es almacenado en la tabla UtilityType.

3.2 Servicio de ejecución de tareas

El servicio automatiza la ejecución de tareas para la red de medidores basado en los registros que se encuentran almacenados en la base de datos. Por lo cual monitorea constantemente la base de datos en busca de tareas que cumplan ciertos criterios, éstas serán agregadas a una lista priorizada para ser traducidas y enviadas por la red de radiofrecuencia hacia el dispositivo correspondiente.

Un servicio de windows, por definición es una aplicación que no cuenta con una interfaz gráfica de usuario, se ejecuta en segundo plano de manera independiente iniciándose casi al mismo tiempo que el sistema operativo. Para monitorear constantemente la ejecución de este, es necesario de una segunda aplicación que está basada en el patrón de diseño "observador".

Este patrón esta creado para vigilar cualquier cambio en el estado de un objeto, en este caso el objeto es la última línea de un archivo (log) de sucesos de servicio, haciendo uso de la clase *tailer listener*, la aplicación monitorea la última línea agregada a este archivo para presentarla en tiempo real en su interfaz. La GUI, el usuario ejecuta *batch files* diseñados para el inicio detención y reinicio del servicio a petición del usuario.

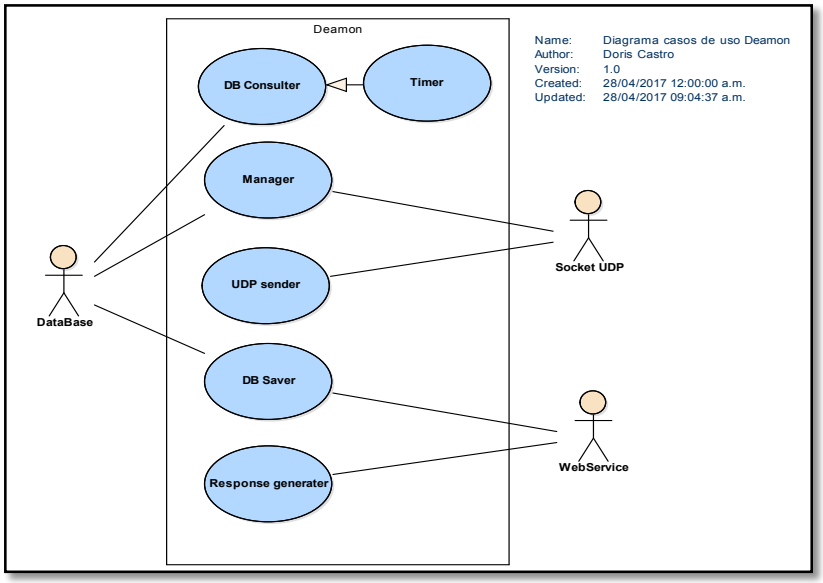


Figura No. 3.9 Diagrama casos de uso del servicio ejecución de tareas (elaboración propia)

Para representar la secuencia de interacciones que se desarrollan entre el sistema y los actores en respuesta a los eventos que inicia el actor principal, se diseñó el diagrama de casos de uso como se puede ver en la figura No. 3.9 el cual representa el proceso desencadenado desde el evento *tick* del objeto *timer* que consulta la base de datos en busca de registros que cumplan con parámetros como: no ejecutado, fecha próxima a ejecutar, entre otros.

El administrador de tareas prioriza de acuerdo a los criterios establecidos. Se hace uso de una clase de tipo cliente UDP (socket cliente UDP) que está relacionado con el *router*, se implementa un tipo de comunicación unicast para enviar el mensaje directamente al dispositivo y este se encarga de descomponer el mensaje para obtener el identificador del dispositivo dentro de la red al cual va dirigido el datagrama.

En la tabla No. 3.2 se describe el diagrama de casos de uso para el servicio windows especificando los puntos más importantes como son: los actores principales, objetivos, precondiciones, acción o proceso que dispara el inicio del servicio web, escenarios principales y las excepciones que pudieran ocurrir cuando el proceso este ejecutándose.

Tabla 3.2 Descripción del diagrama casos de uso del servicio windows

Caso de uso	**Windows Service**
Actor principal	Base de Datos, Socket UDP, Servicio Web
Objetivo	A través de un timer consultar a la base de datos para obtener las tareas que se deben ejecutar, mandarlas al socket para su ejecución y obtener una respuesta de la red mesh
Precondiciones	La red mesh de medidores ANSI este arriba, el servicio web este escuchando, el

Caso de uso	**Windows Service**
	socket UDP este funcionando, la Base de Datos este arriba.
Disparador	Timer programado
Escenario	
1. Base de datos: Obtener tareas programadas para ejecución	
2. Base de datos: Administrar tareas programadas para ejecución	
3. Base de datos: Guardar tareas programadas para ejecución	
4. Socket UDP: Prioriza tareas programadas para ejecución	
5. Socket UDP: Envía tarea a ejecución al UDP	
6. Servicio Web: Guarda tareas programadas en la BD	
7. Servicio Web: Genera archivo XML con la respuesta obtenida	
Excepciones	
1. El Timer no se encuentra funcionando, el servidor Windows deberá enviar error para que pueda ser reiniciado	
2. El socket UDP no responde, el Servicio web debe de intentar nuevamente mandar la tarea a ejecutar.	
3. Se queda sin energía eléctrica el headEnd: El headEnd debe de reestablecerse con la última información que estaba procesando	
Prioridad	Esencial, debe implementarse
Frecuencia de uso	Varias veces al día
Canal para el actor	C12.22. Estándar Nacional Americano para Especificación de Protocolo para Interfaz a Redes de Comunicación de Datos CIM. Common Information Module
Actores secundarios	Administrador TI
Canales para los actores secundarios	Interfaz de usuario
Aspectos	

Caso de uso	**Windows Service**
1. Recepción masiva de información.	
2. Envío masivo de información.	

Nota. Elaboración propia

Para visualizar las interacciones dentro del sistema, se diseñó el diagrama que se muestra en la figura No. 3.10. El servidor web inicia con una petición para crear los registros de la operación que son administradas posteriormente por el servicio. A través del *timer* cada *tick* obtiene los registros y después se obtiene la lista priorizada para que el manejador de tareas envío la operación que ejecutará primero.

El módulo de excepciones se encarga de controlar los errores generados por la aplicación para que en ningún momento detenga su ejecución creando una entrada en el archivo log, informando que error ocurrió y continuando con la ejecución de la lista priorizada de tareas.

Una vez que se obtiene la ejecución para el medidor indicado, se envía la petición a través del socket UDP y este regresa el resultado de la operación.

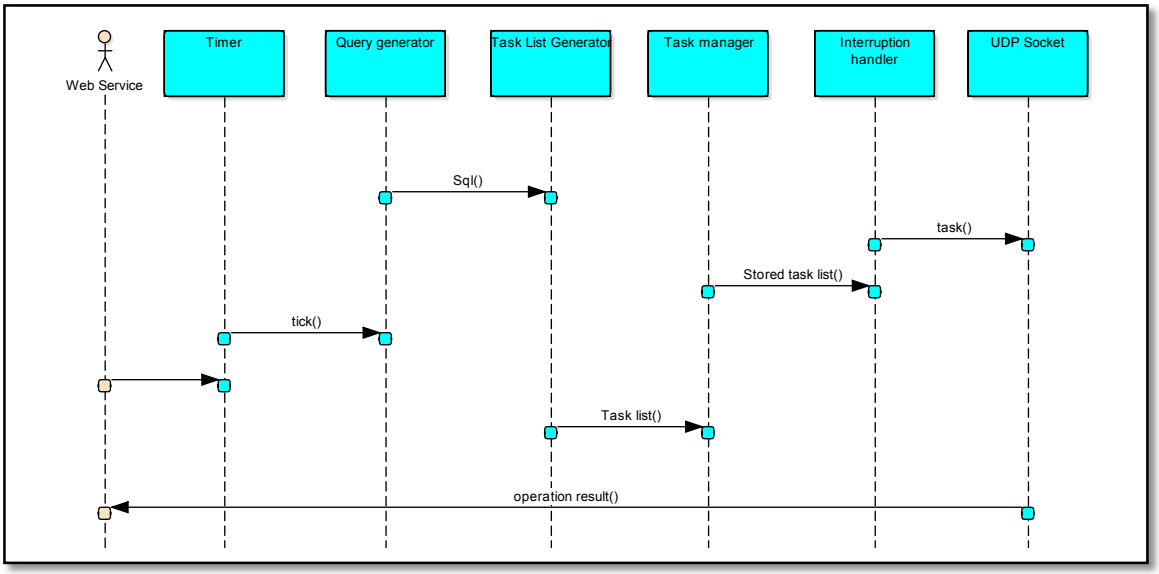


Figura No. 3.10 Diagrama de secuencias del servicio windows (elaboración propia)

3.2.1 Mantener en ejecución servicio Windows

El servicio Windows debe estar siempre en ejecución con el objetivo de estar monitoreando las tareas que se van enviando a ejecución al socket UDP. El proceso que corre en segundo plano es el sql que obtiene los registros de la tabla job.

En la figura No. 3. 11 se visualiza de manera gráfica el servicio Windows en ejecución con un monitor de actividad, el que podrá detener, reiniciar e iniciar.

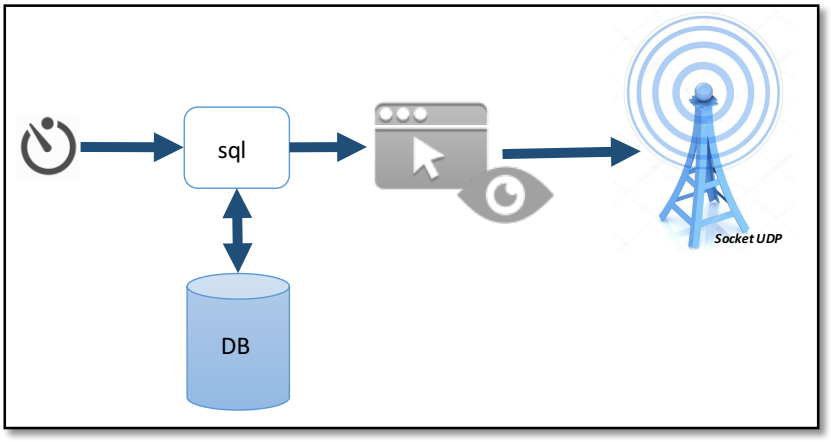


Figura No. 3.11 Servicio windows en ejecución (elaboración propia)

3.2.2 Enviar tarea a red distribuida para ser ejecutada

Cuando el servicio windows recibe el tipo de tarea y sus parámetros desde la lista priorizada, la lógica se encarga de generar el *frame* para montarlo en un JSON y posteriormente serializarlo. Es enviado a través del socket al gateway para que este interprete el datagrama en su capa de aplicación y lo baje a la capa de red para que este sea reenviado.

Una vez que el medidor ejecuta la operación, regresa el resultado para que le llegue al servicio windows y se realice la conversión de la información que llega al protocolo que pueda procesar el sistema. En la figura No. 3.12 se observa gráficamente el proceso.

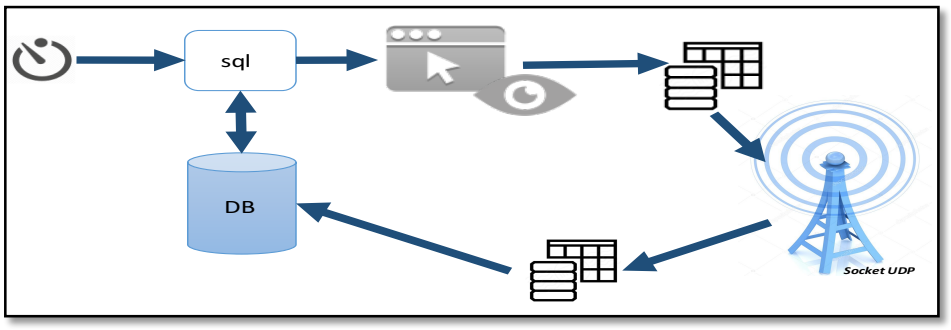


Figura No. 3.12 Protocolo para comunicación con UDP (elaboración propia)

3.3 Manejador de tareas

El administrador de tareas se encarga de la selección parametrizada de cada petición que debe ser enviada a la red de dispositivos para que el dispositivo con el que se quiere interactuar reciba las acciones que debe realizar.

El servidor Windows solicita la selección de las tareas que se deben realizar las procesa y las envía a la red de dispositivos para que sean ejecutadas. Las peticiones enviadas en tiempo real requieren que sean administradas para que no genere conflicto entre ellas por lo que se emplea un *Thread Pool* configurable en el número de hilos que trabaja, esto permite la optimización de recursos al momento de distribuir la asignación de procesos en los hilos programados, en la que realiza una selección de registros de solicitudes desde la base de datos que deben ser ejecutadas tomando en cuenta su priorización. Cada entrada de la base de datos que es recuperada, se interpreta y actualiza a través de Hibernate.

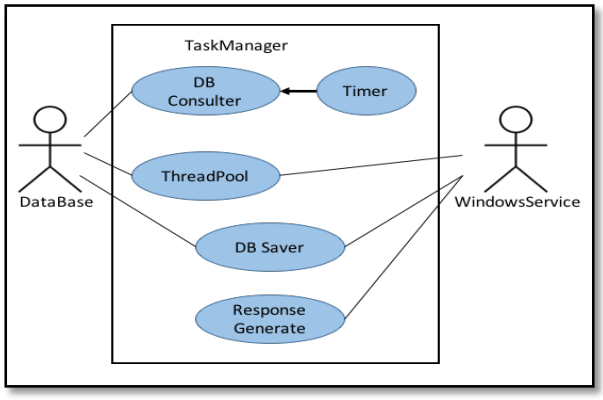


Figura No. 3.13 Diagrama de clases del administrador de tareas (elaboración propia)

Para la utilización de Hiberntate se hizo uso del patrón de diseño java Interface, para separar las funciones (firma de métodos) de las implementaciones (cuerpo del método) así cualquier solicitud que coincida con la firma de la interfaz de un objeto también pueda ser enviada a ese objeto, independientemente de su aplicación. A través de la interfaz desarrollada se realiza la recuperación, interpretación y actualización de las tareas.

En la figura No. 3.13 se visualiza la secuencia de interacciones que se desarrollan entre el sistema y los actores en respuesta a los eventos que inicia el actor principal.

En la tabla No. 3.3 se describe el diagrama de casos de uso para el administrador de tareas especificando los puntos más importantes como son: los actores principales, objetivos, precondiciones, acción o proceso que dispara el inicio del servicio web, escenarios principales y las excepciones que pudieran ocurrir cuando el proceso este ejecutándose.

Tabla 3.3 Descripción del diagrama casos de uso del administrador de tareas

Caso de uso	**Task Manager**
Actor principal	Base de Datos, Servicio Web
Objetivo	A través de un timer consultar a la base de datos para obtener las tareas que se deben ejecutar y obtener una respuesta de la red mesh
Precondiciones	La red mesh de medidores ANSI este arriba, el servicio web este escuchando, el socket UDP este funcionando, la Base de Datos este arriba.
Disparador	Timer programado
Escenario	
1. Base de datos: Obtener tareas priorizadas programadas para ejecución	
2. Base de datos: Guardar tareas programadas para ejecución	

Caso de uso	**Task Manager**
3. Base de datos: Distribuye tareas a ejecutar en cada hilo de ejecución del Thread Pool	
4. Servicio Windows: Envía tareas al socket UDP para que se envie a la red de medidores	
5. Servicio Windows: Guarda tareas programadas en la BD	
6. Servicio Windows: Genera archivo XML con la respuesta obtenida	
Excepciones	
1. El Timer no se encuentra funcionando, el servidor Windows deberá enviar error para que pueda ser reiniciado	
2. El socket UDP no responde, el Servicio web debe de intentar nuevamente mandar la tarea a ejecutar.	
3. Se queda sin energía eléctrica el headEnd: El headEnd debe de reestablecerse con la última información que estaba procesando	
Prioridad	Esencial, debe implementarse
Frecuencia de uso	Varias veces al día
Canal para el actor	C12.22. Estándar Nacional Americano para Especificación de Protocolo para Interfaz a Redes de Comunicación de Datos CIM. Common Information Module
Actores secundarios	Administrador TI
Canales para los actores secundarios	Interfaz de usuario
Aspectos	
• Recepción masiva de información.	
• Envío masivo de información.	

Nota. Elaboración propia.

Las interacciones dentro del sistema, se visualizan en el diagrama que se muestra en la figura No. 3.14. El administrador de tareas inicia cuando el servicio *windows* a través de un *tick* del *timer* obtiene los registros con sql, posteriormente se obtiene la lista priorizada para que el manejador de tareas envío la operación que ejecutará primero. La lista de tareas se distribuye en los *thread pool*. Las tareas se van enviando al *socket* UDP para ser ejecutadas en los medidores al finalizar la ejecución el UDP regresa el resultado de la operación.

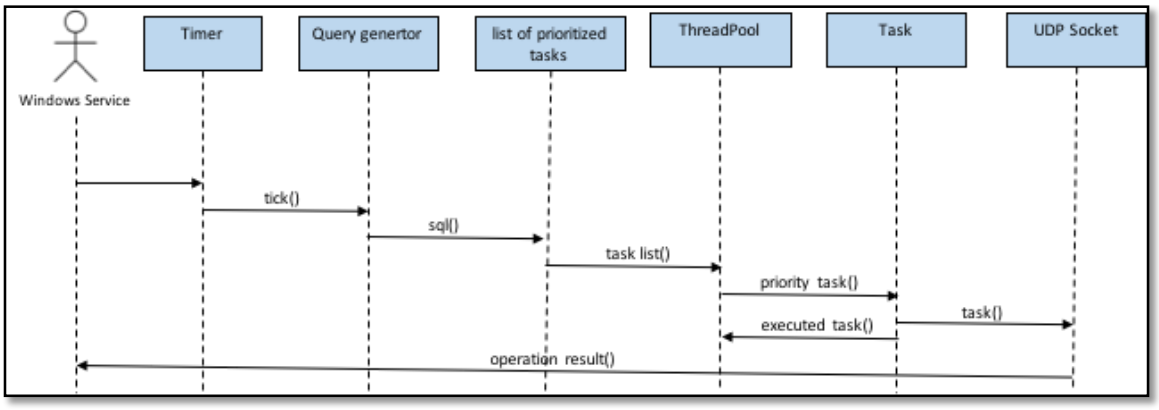


Figura No. 3.14 Diagrama de secuencias del administrador de tareas (elaboración propia)

3.3.1 Obtener tareas priorizadas

La alta concurrencia que tiene el sistema tiene como característica recibir varias peticiones del MDM que se deben atender lo más pronto posible. Por lo que se integró un *thread pool* donde se puede definir un número determinado de *threads*, en los que se van procesando los registros de la base de datos que se toman de una cola. De esta forma, se procesan varias peticiones simultáneamente en un determinado número de *threads*.

El módulo de administración de tareas realiza la selección de trabajos identificando la prioridad que tienen, estas pueden ser de prioridad: alta, media y baja. El número de tareas puede variar dependiendo de la cantidad de peticiones que haya realizado el MDM y el número de dispositivos que afecte cada una de ellas.

A cada uno de los procesos se le asigna un hilo de ejecución en un *thread* pool que se crea cada determinado tiempo. El intervalo de tiempo para ir creando un *thread pool* y los hilos de ejecución que lo integran son configurables. Cuando llegan las siguientes operaciones, se busca primero los *threads* que ya hayan terminado de procesar, para procesar las peticiones en los *threads* que se encuentran libres.

La forma como se van buscando hilos libres se realiza tomando en cuenta la prioridad de la tarea debido a que se asignaron tres *threads* a prioridad alta, dos a prioridad media y uno a prioridad baja, como se muestra en la figura No. 2. El que se asignen un determinado número de *threads* a ciertas operaciones por su prioridad solo es para asignar el recurso, pero si llega una tarea y encuentra su *thread* asignado ocupado utilizará el que se encuentre libre empezando a buscar en la prioridad baja inmediata.

Para aplicaciones con procesos que se deben de ejecutar simultáneamente, los *thread* pools son una solución muy útil que se sirven de los núcleos de los procesadores de la computadora para trabajar de forma concurrente. Lo ideal es crear el mismo número de *threads* que equivalgan al número de núcleos que tenga el procesador donde se ejecute la aplicación.

3.3.2 Control de excepciones para el manejo de interrupción de tareas

El manejador de tarea es parte del servicio *windows* por lo que es importante que no se detenga por lo que es necesario tener controladas todo tipo de excepciones. Para este se utilizan try-catch para cada proceso indicando que error ocurrió.

Para el almacenamiento de errores se utiliza la clase BAL_FileCreator, cada que se detecta una excepción recibe qué tipo de error se originó y lo almacena en un archivo de texto con el día y hora que se generó.

3.3.3 Generación de archivo XML de respuesta de la red distribuida

El servidor Windows envía la tarea que se debe ejecutar en la red distribuida al medidor que indica se indica. Al generar una respuesta la red distribuida, envía la respuesta al servidor

Windows para que sea procesada y se genera un archivo XML con la respuesta generada, este archivo se almacena en el servidor remoto del MDM que la solicito y el servidor web envía el aviso de que la tarea fue ejecutada y ya se depositó la respuesta en el servidor remoto.

Para la generación del archivo XML respuesta se realiza con base al documento CFE G0100-5, que contiene las especificaciones del Sistema de Infraestructura Avanzada de Medición (AMI) donde se encuentran las estructuras de comandos requeridas por la CFE para el proyecto HES.

Capítulo 4 Pruebas y Resultados

En este capítulo se comentan las pruebas realizadas a los bloques de código desarrolladas para la lógica de control del proceso de desarrollo de un Sistema Head-End. Se realizaron pruebas de caja blanca por bloques de código para comprobar su correcta funcionalidad. Al finalizar cada módulo se utilizaron pruebas de caja negra para verificar el funcionamiento integral bajo los estándares de la "Metodología de aseguramiento de la calidad de un sistema Head End (HES) de una red distribuida de medidores inteligentes" (Nava, 2018).

El proyecto descrito aquí es una parte de un proyecto mayor, por lo que se crearon ambientes de prueba antes de realizar su liberación para hacer las pruebas integrales en el sistema. Se hizo un módulo que pudiera simular ser el cliente MDM; el módulo de simulación MDM fue codificado para que generara archivos XML con las posibles tareas aceptables y realizara varias peticiones de forma automática en un determinado *tick* de un *timer*.

4.1 Extracción y validación de archivos XML

Para realizar la prueba de comunicación, se ejecutó el servicio web desde el servidor apache Tomcat a través de la aplicación Eclipse, esto fue hecho para dejarlo disponible en su uso, como se puede observar en la figura No.4.1

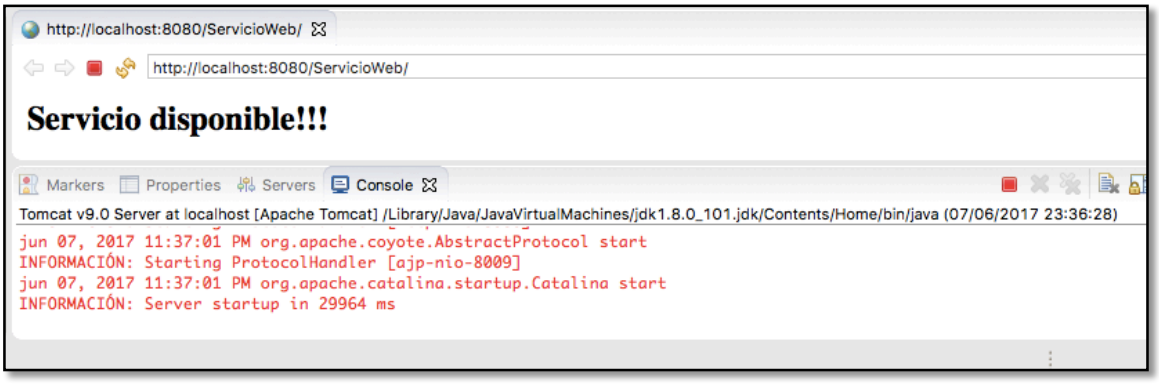


Figura No. 4.1 Servicio web en ejecución (imagen tomada de la pantalla en ejecución)

Las pruebas se realizaron de forma directa desde la url de un *browse* de navegación de internet, como se muestra en la figura No. 4.2. Desde la url se especifica el nombre del archivo XML que contiene las tareas que se deben extraer.

Figura No. 4.2 Ejemplo de prueba realizada en un navegador de internet (imagen tomada de la pantalla en ejecución)

Las acciones descritas en la tabla No. 4.1 tienen como objetivo validar la comunicación con el servicio web, provocando los posibles errores, extracción y validación de los archivos XML.

Tabla 4.1 Acciones y/o condiciones para pruebas de comunicación del servicio web

	Acciones	***Salida esperada***	***Resultado***
1	Recepción de petición desde MDM con una configuración errónea del servidor remoto	Error mostrado en el navegador por no poder realizar la comunicación	ok
2	Recepción de petición desde MDM con una configuración correcta del servidor remoto y un archivo existente	Respuesta del servidor web hacia el MDM del número de tareas procesadas	ok
3	Recepción de petición desde MDM con una configuración correcta del servidor remoto y un archivo que no existente	Mostrar el aviso que existe un error para consultar detalles en el archivo de errores.	ok

	Acciones	***Salida esperada***	***Resultado***
4	Validación de archivo XML incorrecto contra esquema XSD	Muestra aviso de error y almacena los detalles en el archivo de errores	ok
5	Validación de archivo XML correcto contra esquema XSD	El proceso avanza al próximo proceso de interpretación de tareas	ok

Nota. Elaboración propia.

Al realizar las pruebas de la comunicación del servicio web, se consultó el archivo de errores que se generaban durante el proceso de extracción y validación de archivos XML para comprobar que los tipos de errores que se provocaban se almacenaran correctamente en el archivo. En la figura No. 4.3 se muestra un ejemplo de los errores obtenidos.

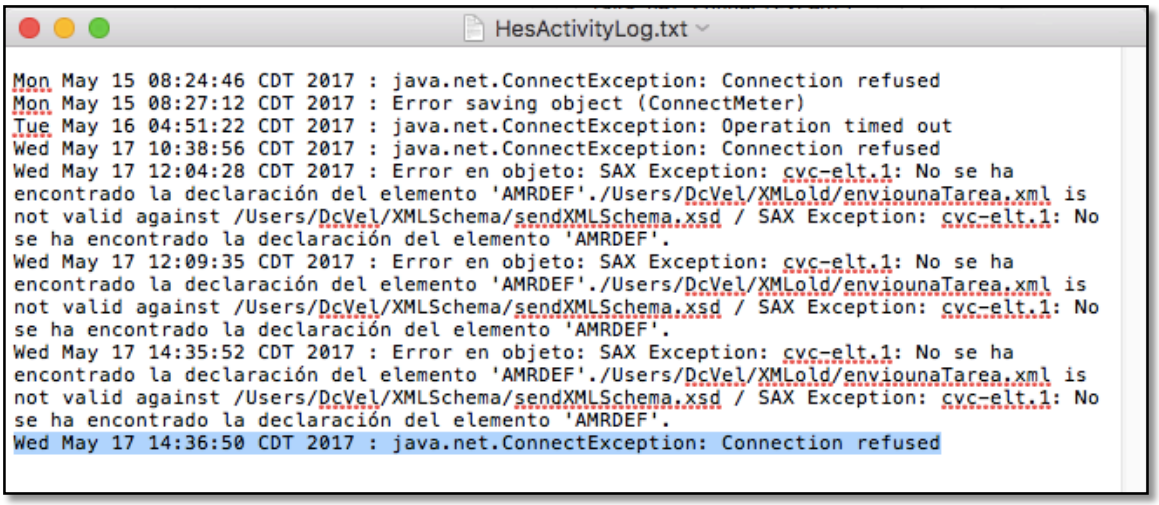


Figura No. 4.3 Archivo del log de errores del servicio web (imagen tomada de la pantalla con el archivo txt abierto que almacena los errores)

En el caso de la validación del archivo XML contra el esquema XSD, se realizaron las pruebas correspondientes para la validación del diseño correcto del esquema en que se deben aceptar *n*

cantidad de tareas, cada una con sus atributos y *n* operaciones aceptables, cada una con las características específicas para ejecutar en los medidores.

Tabla 4.2 Casos tomados en cuenta para generación de archivos XML de prueba

Tareas	***Operaciones por tarea***				
1	1	2	3	…	n
2	1	2	3	…	n
3	1	2	3	…	n
…					
n	1	2	3	…	n

Nota. Elaboración propia.

Para hacer las pruebas, se desarrolló un código que los armara dichos casos de manera automática, como se muestra en la tabla No. 4.2. En el caso de la variable *n* se obtiene un valor aleatorio para generar un caso al azar de la cantidad de tareas u operaciones. Las operaciones válidas que debe aceptar el servidor web se muestran en la tabla No. 4.3.

Se crearon veinte pruebas hasta para comprobar que la comunicación con el servicio web manejara todos los errores posibles y aceptara solamente archivos XML válidos de acuerdo al MDM que solicitará la ejecución de tareas.

Tabla 4.3 Operaciones válidas para ejecutar sobre medidores

Operación valida	Atributos obligatorios
DisconnectMeter	Si
ConnectMeter	Si
Meter	Si
MeterBilling	Si
ReadMeterBilling	Si

Nota. Elaboración propia.

4.2 Interpretación del archivo XML

Una vez que se comprobó que el proceso de validación aceptara solamente archivos XML que cumplieran con el esquema XSD, se realizaron las pruebas al proceso de extracción y mapeo de tareas para poder hacer el almacenamiento de los datos en la base de datos.

Con base a los casos para crear los archivos XML, se realizó la validación del almacenamiento en la estructura de la base de datos a través de un mapeo. Se obtiene la información que pertenece a registros de catálogo o información almacenada en tablas de configuración que caractericen a la tarea, estos datos se almacenan en objetos para posteriormente realizar la persistencia en la base de datos.

En la figura No. 4.4 se muestra de forma gráfica el proceso de mapeo para la realización de la persistencia en la base de datos. Con base a este diagrama general se validó que cada caso de prueba obtuviera los identificadores (ids) de los datos característicos de la operación para generar los registros de las tareas que se almacenan en la tabla *job*.

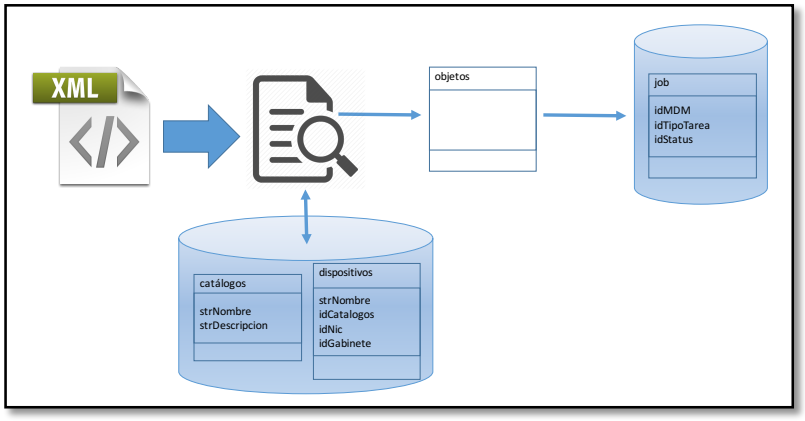


Figura No. 4.4 Diagrama mapeo de información extraída del archivo XML (elaboración propia)

Las pruebas realizadas de extracción y persistencia en base de datos se realizaron en tres fases debido a que se detectaron cambios significativos al momento de almacenar la información, esto

produjo que se realizaran ajustes en el diagrama entidad-relación para tener una adecuada consistencia en el almacenamiento de los datos. Durante la ejecución de pruebas se comprobó que se obtuvieron los ids de la información y que los registros generados de las tareas fueron almacenados correctamente.

4.3 Monitoreo de extracción de tareas en tiempo real

La validación del monitoreo se realizó al momento que se cumplía el *tick* del *timer* que iniciaba la selección de la tarea que se debe ejecutar; esta se mostró correctamente en la ventana de visualización que se diseñó para el servicio *windows.* La ventana de monitoreo se puede ver en la figura No. 4.5

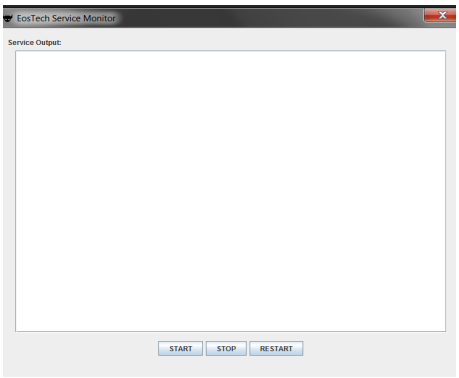


Figura No. 4.5 Ventana de monitoreo de las tareas a ejecutar (imagen tomada de la pantalla en ejecución)

Durante esta prueba se probaron los botones de inicio, parar y reiniciar el servicio *windows.* Los eventos de los botones se ejecutaron de la manera esperada: iniciando, deteniendo y reiniciando respectivamente el servicio.

4.4 Ejecución de servidor Windows al iniciar computadora

Para que el servicio *windows* siempre inicie, se realizaron las pruebas mostradas en la tabla No. 4.4, apagando la computadora en las posibles formas y que el servicio inicie al momento en que el sistema operativo termine de cargar.

Tabla 4.4 Acciones y/o condiciones para las pruebas de inicio del servicio windows

	Acción	Salida esperada	Resultado
1	Apagar la computadora correctamente a través del menú.	Al encender la computadora el servicio *windows* se inicia	ok
2	Forzar el apagado de la computadora.	Al encender la computadora el servicio *windows* se inicia	ok
3	Al estar funcionando la computadora, desconectar la computadora	Al encender la computadora el servicio *windows* se inicia	ok

Nota. Elaboración propia.

En cada caso de prueba se comprobó el inicio de la aplicación a través de los servicios de *windows* en la figura No. 4.6 se muestra la ventana donde se visualizan los servicios de *windows* que se encuentran en ejecución y también los que no.

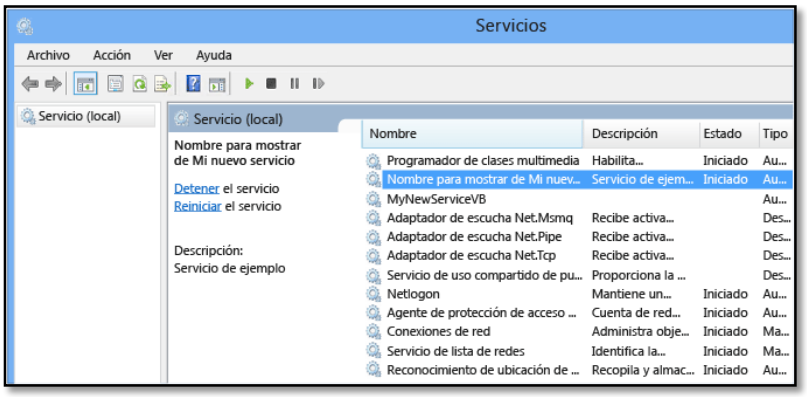


Figura No. 4.6 Servicio windows en ejecución (imagen tomada de la pantalla en ejecución)

4.5 Envío de petición de tarea al socket UDP

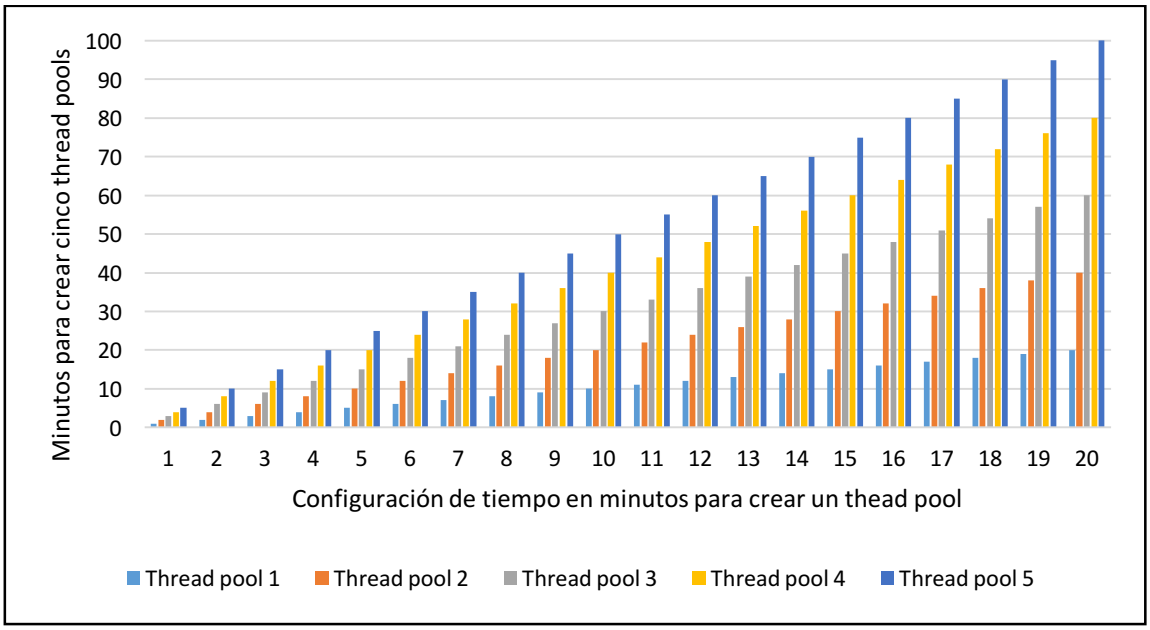
Una vez que se ha recibido el tipo de tarea y sus parámetros desde la lista priorizada, la lógica se encarga de generar un *frame* C1218 y lo monta en un JSON y tras haberlo serializado, es enviado a través del *socket* al *gateway* para que este interprete el datagrama en su capa de aplicación y lo baje a la capa de red para que este sea reenviado.

Estas pruebas se realizaron con base al apartado 4.8 donde se realiza la validación de la distribución de procesos en los hilos configurados a partir de estos casos se validó que el socket UDP regrese el datagrama para interpretarlo y continuar con el proceso de respuesta al MDM.

Las pruebas comprobaron que se recibía la respuesta a través del UDP que da paso al proceso para generar la respuesta en XML.

4.6 Activación de tiempos para la creación de *thread pool*

En el método para la activación de tiempos se utiliza la librería de java java.util.Timer.schedule(TimerTask task, long delay, long period) en combinación con el método java.util.TimerTask.TimerTask() para obtener la funcionalidad de la ejecución de consulta a la base de datos en el tiempo especificado.



Gráfica 4.1 Tiempos utilizados para probar la activación de tiempos para la creación de thread pools (elaboración propia)

En la realización de las pruebas correspondientes a este bloque, se imprimió en la consola el resultado de la información obtenida de la base de datos comprobando que se realice en el

tiempo establecido, obteniendo los tiempos esperados que se configuraron para cada prueba. En la gráfica No. 4.1. se puede visualizar fácilmente el comportamiento de las configuraciones probadas. El eje "x" representa configuraciones de 1 a 20 minutos que son los intervalos para crear los *thread pools*, en el eje "y" se representa el tiempo invertido en la creación de cinco conjuntos de hilos.

En el primer grupo de barras representa que en cada minuto se crea un *thread pool*, el segundo es cada dos minutos y así sucesivamente para cada bloque. En el eje principal se puede observar el tiempo de duración de cada configuración donde la programación de cada veinte minutos duro cien minutos para finalizar la prueba.

Los resultados de esta prueba arrojaron los números de acuerdo a cada configuración por lo que se terminó cada prueba en el tiempo esperado.

4.7 Extracción de tareas priorizadas

Para poder validar la correcta extracción de tareas de la base de datos según su prioridad, se diseñaron registros de prueba con la asignación de priorización de los tipos aceptados que son: alta, media y baja. En la tabla No. 4.5 se muestran los registros con los campos que se toman en cuenta para la selección, el estatus del registro contiene el estado inicial de la operación y en la última columna es la cantidad de registros con las mismas características.

La prueba de validación en la extracción de tareas priorizadas consiste en comprobar que el registro obtenido con la prioridad especificada es el que debe ejecutarse primero tomando en cuenta que la fecha de aplicación de la tarea sea menor o igual a la fecha en que se realiza la extracción y que el estatus sea "no ejecutado" una vez que se obtiene el registro, el proceso debe cambiar el estatus a "en proceso" para que la siguiente selección no tome en cuenta el registro porque ya fue seleccionado.

Tabla 4.5 Registros muestra para la extracción de tareas por prioridad

	Fecha de aplicación	Prioridad	Estatus	Registros en base de datos
1	13/06/17	alta	no ejecutado	2
2	12/06/17	media	no ejecutado	3
3	15/06/17	baja	no ejecutado	1
4	05/06/17	alta	no ejecutado	3
5	08/06/17	media	no ejecutado	8
6	02/06/17	baja	no ejecutado	1
7	11/06/17	alta	no ejecutado	5
8	11/06/17	media	no ejecutado	10
9	11/06/17	baja	no ejecutado	3

Nota. Elaboración propia.

En la Figura No. 4.7 se muestran las pruebas con la simulación de tres fechas en que fueron realizadas las selecciones de los registros, en el eje “x” refiere al tipo de priorización de los registros, en el eje “y” corresponde a la cantidad de registros obtenidos de acuerdo a la fecha de extracción.

El gráfico (a) muestra la prueba realizada con fecha de selección del once de junio por lo tanto se obtuvo los registros que tienen fecha hasta el once de junio distribuidos de la siguiente manera: de prioridad alta, con fecha menor tres y con fecha igual cinco; de prioridad media, con fecha menor ocho y con fecha igual diez; de prioridad baja, con fecha menor uno y con fecha igual tres.

En el gráfico (b) se hizo con fecha del ocho de junio por lo que el resultado muestra los registros con fecha hasta el ocho de junio distribuidos de la siguiente manera:

La última prueba se muestra en el gráfico (c) donde se realizó con fecha del cinco de junio la cual no arrojó ningún registro porque no se tenía en base de datos información con fechas que

cumplieran la condición en esta prueba. En las tres gráficas se puede observar que no arrojo ningún registro con fecha mayor a la indicada en cada prueba.

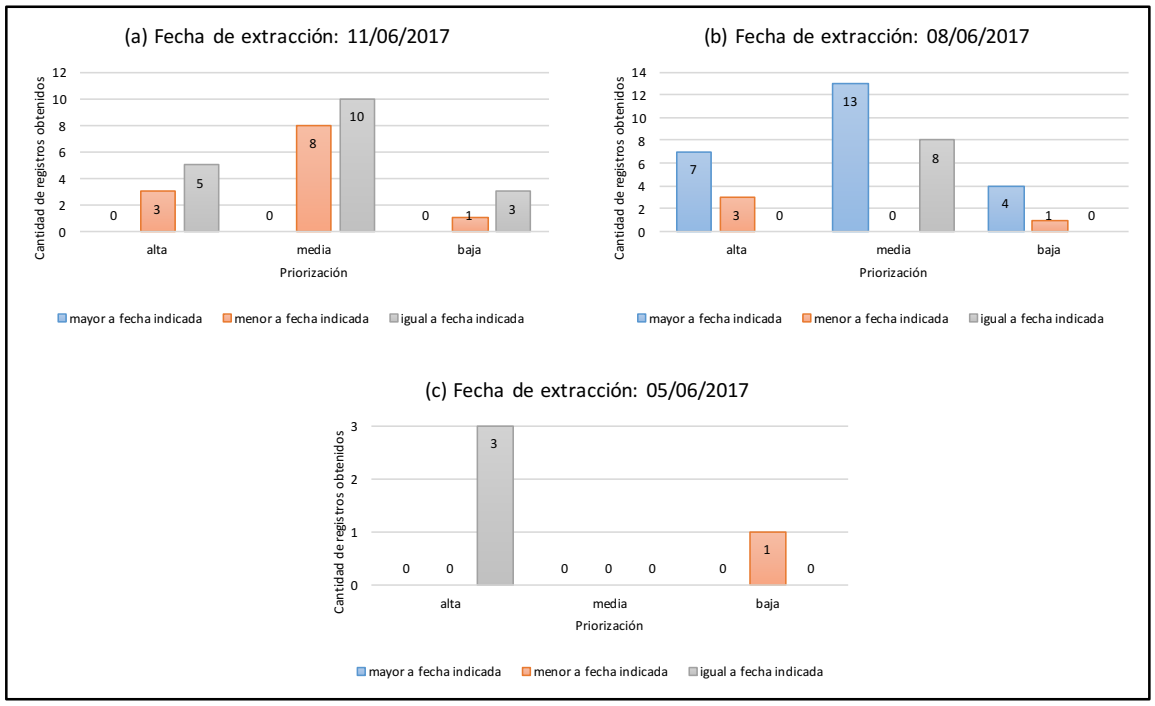


Figura No. 4.7 Resultado de selección de tareas priorizadas (elaboración propia)

Para validar la consulta a la base de datos de acuerdo a las tareas correspondientes de priorización, se desplegó en terminal el resultado del proceso para comprobar que los registros obtenidos fueran los correctos y se comparó directamente con los registros en base de datos.

El resultado general de las pruebas realizadas con respecto a fechas específicas corresponde a las cifras esperadas de acuerdo a los registros generados para la realización de las pruebas.

4.8 Distribución de procesos en los hilos configurados

La configuración que fue seleccionada por la empresa para el proceso, fue de un *thread pool* con seis hilos de ejecución dando prioridad a las tareas de la siguiente manera: tres hilos alta, dos hilos para media y un hilo para baja.

Para validar la configuración se realizaron varias combinaciones de consulta según su priorización para comprobar que no afecta el proceso de administración de procesos. Se crearon datos para generar selección de registros con las combinaciones de prioridad que se muestran en la tabla No. 4.6.

Tabla 4.6 Casos de prueba para el Thread pool

	Tareas			**Cantidad de hilos para priorizar**		
	alta	**media**	**baja**	**3**	**2**	**1**
Caso 1	5	1	0	3 altas	2 altas	1 media
Caso 2	4	2	0	3 altas	1 alta, 1 media	1 media
Caso 3	3	3	0	3 altas	2 medias	2 medias
Caso 4	2	4	0	2 altas, 1 media	2 medias	1 media
Caso 5	1	5	0	1 alta, 2 medias	2 medias	1 media
Caso 6	5	0	1	3 altas	2 altas	1 baja
Caso 7	4	0	2	3 altas	1 alta, 1 baja	1 baja
Caso 8	3	0	3	3 altas	2 bajas	1 baja
Caso 9	2	0	4	2 altas, 1 baja	2 bajas	1 baja
Caso 10	1	0	5	1 alta, 2 baja	2 bajas	1 baja
Caso 11	0	5	1	3 medias	2 medias	1 baja
Caso 12	0	4	2	3 medias	1 media, 1baja	1 baja
Caso 13	0	3	3	3 medias	2 bajas	1 baja
Caso 14	0	2	4	2 medias, 1 baja	2 bajas	1 baja
Caso 15	0	1	5	1 media, 2bajas	2 bajas	1 baja

Nota: Elaboración propia

Los resultados de las pruebas reflejaron que no afecta la administración de procesos, los hilos se van ejecutando y los *thread pools* se van creando en los tiempos configurados, con lo que podemos observar que trabaja según lo planeado.

También se generaron excepciones provocadas que no interrumpieron los procesos debido a que se comprobó que se encuentran controladas. Se generaron diferentes tipos de fallos que pudieran ocurrir al estar ejecutando la aplicación, validando que los errores se registraran en la excepción correspondiente a través de un log de errores. Esto validó que tuviera un catch para excepciones no controladas y que se registrara el error que se produce.

4.9 Liberación de Módulos

Para ejecutar las pruebas de los módulos se integró el sistema en un ambiente simulado con la configuración de tiempos cada diez minutos para validar la ejecución de las tareas encendido y apagado de un medidor especifico en un gabinete prueba con cuatro medidores. Desde la computadora, donde se encuentra montado el sistema MDM (Carmona, 2018), se realizó la solicitud de ejecución de tareas a través del servidor web.

Las pruebas de liberación de módulos se realizaron partiendo de los requisitos funcionales donde se especifica que se debe enviar a ejecución las tareas de acuerdo a su prioridad sin generar duplicidad y controlando las excepciones que se generen para no detener la ejecución de todos los procesos. El resultado debe ser el encendido y apagado de un medidor o medidores en el caso de que éste se trate de un medidor asociado.

Con la solicitud realizada el sistema HES inicia el proceso de extracción del archivo XML del servidor remoto, obtiene las tareas a ejecutar y las almacena en la base de datos (Hernández, 2018) para que el servidor *windows* de acuerdo a la programación de tiempos, obtenga las tareas que se deben ejecutar y las envía al gabinete de prueba. En la figura 4.8 se muestra el diagrama general del ambiente de pruebas para el proceso realizado.

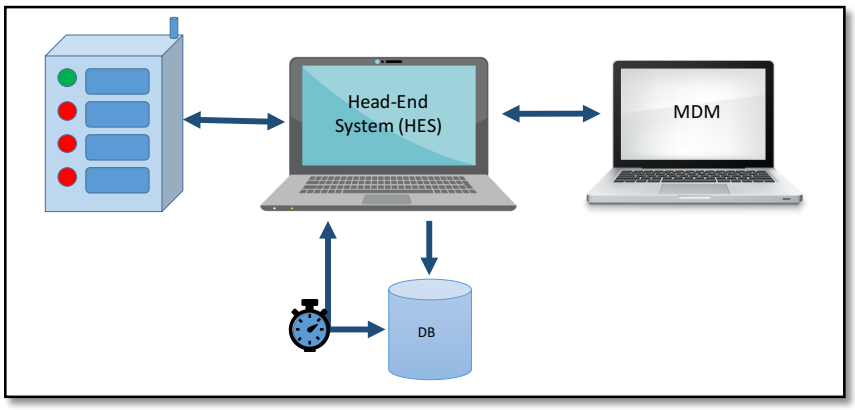


Figura No. 4.8 Diagrama del ambiente de pruebas (elaboración propia)

Tabla 4.7 Casos de prueba en la liberación de módulos

	tarea	**medidor**	**asociación**	**medidores afectados**	**hora de ejecución**	**resultado**
1	encendido	1	No	1	10:10	ok
2	apagado	1	No	1	10:20	ok
3	encendido	2	No	2	10:30	ok
4	apagado	2	No	2	10:40	ok
5	encendido	3	No	3	10:50	ok
6	apagado	3	No	3	11:00	ok
7	encendido	4	No	4	11:10	ok
8	apagado	4	No	4	11:20	ok
9	encendido	1	Si	1,2	11:30	ok
10	apagado	2	Si	1,2	11:40	ok
11	encendido	1	Si	1,3	11:50	ok
12	apagado	3	Si	1,3	12:00	ok
13	encendido	1	Si	1,4	12:10	ok
14	apagado	4	Si	1,4	12:20	ok
15	encendido	2	Si	2,3	12:30	ok
16	apagado	3	Si	2,3	12:40	ok

	tarea	medidor	asociación	medidores afectados	hora de ejecución	resultado
17	encendido	2	Si	2,4	12:50	ok
18	apagado	4	Si	2,4	13:00	ok
19	encendido	3	Si	3,4	13:10	ok
20	apagado	4	Si	3,4	13:20	ok
21	encendido	1	Si	1,2,3	13:30	ok
22	pagado	2	Si	1,2,3	13:40	ok
23	encendido	3	Si	1,2,3	13:50	ok
24	apagado	1	Si	1,3,4	14:00	ok
25	encendido	3	Si	1,3,4	14:10	ok
26	apagado	4	Si	1,3,4	14:20	ok
27	encendido	1	Si	1,2,4	14:30	ok
28	pagado	2	Si	1,2,4	14:40	ok
29	encendido	4	Si	1,2,4	14:50	ok
30	apagado	2	Si	2,3,4	15:00	ok
31	encendido	3	Si	2,3,4	15:10	ok
32	pagado	4	Si	2,3,4	15:20	ok

Nota. Elaboración propia

Los casos que se validaron en las pruebas se muestran en la tabla No. 4.7 con las combinaciones entre cuatro medidores instalados en un gabinete, que tienen como objetivo comprobar el encendido y apagado del medidor que se especifique. Se probaron casos en los que los medidores se encuentran asociados y se envía el encendido o apagado de uno de ellos para que se vea como afectó de la misma forma a los que están asociados.

Para realizar estas pruebas se preparó cada caso en un archivo xml de solicitud, que el MDM envió al HES a través del servicio web para que el servicio *windows* extrajera el trabajo cada diez minutos de la base de datos.

El resultado de los casos probados se visualizó en el gabinete cuando los medidores se encendieron y apagaron según lo especificó la tarea de acuerdo a la tabla No. 4.7 que indica la acción que se solicita, a que medidor se le debe realizar, si el medidor está asociado a otros y la acción a que medidores debe afectar, la hora en la que se ejecutó y si la prueba afecto a los medidores que se indica.

Se realizaron dos etapas de prueba para asegurar que las tareas se ejecutaron de acuerdo a su prioridad y en el tiempo configurado. Los dispositivos se apagaron y encendieron según lo que indicó cada petición en las tareas y afectó a los medidores que tenían asociación.

La información referente a los medidores se visualizó a través de la interfaz gráfica del HES (Guzmán, 2018), donde se puede consultar los trabajos que se encuentran almacenados en la base de datos, el estatus del medidor, si está asociado, a que medidores se encuentra asociado si es el caso y las demás características de un medidor.

Siguiendo la metodología de aseguramiento de la calidad de un sistema Head End (HES) de una red distribuida de medidores inteligentes (Nava, 2018) todas las pruebas realizadas a cada módulo y las pruebas de integración, se registraron en el formato de pruebas que se muestra en el Anexo 1.

Las pruebas internas de cada módulo las realizó el programador para comprobar que cada módulo desarrollado cumpla con el funcionamiento solicitado, las pruebas de integración se llevaron a cabo bajo la supervisión de un *tester* externo que validó el buen funcionamiento de acuerdo a los resultados obtenidos en cada prueba acorde a los estándares de la metodología aplicada.

Capítulo 5 Conclusiones y Trabajos Futuros

En la actualidad se requiere de sistemas que ayuden al manejo de la información que generamos de forma exponencial a través de todos los dispositivos electrónicos con los que interactuamos a través de una red. Por esta razón, los proyectos deben ser modelados y desarrollados para ser capaces de recibir los datos que se van transfiriendo en la red ya sea para almacenar, procesar o presentar la información al usuario final.

Al término del desarrollo del proyecto presentado en este documento se llegaron a conclusiones y se detectaron trabajos que se pueden realizar a partir de este desarrollo.

5.1 Conclusiones

En esta tesis se diseñó, desarrolló e implementó la lógica de control del proceso de desarrollo de un Sistema *Head-End* para una red de dispositivos inteligentes que garantiza el flujo y registro de información entre el MDM y la red de medidores para resolver la problemática descrita en el Capítulo 1 sección 1.

Las tecnologías (Capitulo 2) empleadas para el desarrollo de este trabajo, se lograron implementar adecuadamente para interactuar con los demás sistemas, logrando un sistema robusto y escalable.

Durante el desarrollo del proyecto se utilizó la metodología tradicional incremental adaptada con el método Building Blocks y la metodología ágil *Scrum*, que facilitó la elaboración de bloques de código independiente funcional logrando un software sustentable y de fácil mantenimiento (Capitulo 3).

Como se describe en el Capítulo 3 sección 1, para el manejo de información constante entre los sistemas, se utilizó un servicio Web FTP que permitió la comunicación segura a través de archivos encriptados donde se especifican las acciones que deben ser ejecutadas en los

medidores inteligentes. Los archivos al ser almacenados en los servidores de cada participante, permitió tener el historial de las peticiones enviadas a través del servicio.

Debido a que el protocolo de transferencia FTP fue el solicitado por la empresa (que se menciona en el Capítulo 3 sección 2.3), se utilizó la encriptación de datos y archivos para poder realizar la comunicación de forma segura cumpliendo con los estándares en protocolo de seguridad para el manejo de información.

Al realizar el módulo de extracción y validación de archivos XML se utilizó esquemas XSD (explicado en el Capítulo 3 sección 1.4), que permiten la escalabilidad para la implementación de módulos para que el sistema pueda aceptar otros dispositivos inteligentes con sus propias estructuras para el envío de peticiones de ejecución de tareas.

La interpretación del archivo XML se realizó a través de una lista de objetos para facilitar el mapeo con las entidades correspondientes en la base de datos (Capitulo 3 sección 1.5). Para realizar la persistencia a través de Hibernate sólo se requirió llenar los objetos anidados con la información correcta y al ejecutar la instrucción del ORM para almacenar el objeto en la base de datos, este guarda la información de los objetos anidados en las tablas que corresponden de acuerdo al mapeo.

Se explica en el Capítulo 3 sección 2.1 la ejecución del servicio *windows* que utilizó el patrón de diseño "*observer*" lo que facilitó la visualización en tiempo real de las tareas que deben ser ejecutadas. Este método cada que encuentra la existencia de tareas a realizar en el momento las muestra en el servicio.

Para realizar la comunicación con los dispositivos inteligentes se hizo uso de un *socket* UDP que se encarga de realizar la comunicación con el dispositivo al cual va dirigido el mensaje. En este módulo sólo se realizó el envío de tarea a través del *servicio windows* por que los métodos para realizar el datagrama ya los tenía realizados la empresa.

El envío de tareas al socket UDP se realizó la implementación de *thread pools* con lo que se logró aprovechar los recursos con la comunicación eficiente entre los hilos en los tiempos programados como se demuestra en el Capítulo 4 sección 6. Se utilizó una bandera almacenada en base de datos para identificar los trabajos que ya fueron tomados para ser ejecutados en el dispositivo especificado.

El proceso encargado de obtener los registros que contienen la acción que debe ser enviada a ejecución, obtiene satisfactoriamente una a una la tarea con la prioridad que se le va solicitando, cumpliendo con la configuración que se haya realizado para prioridades alta, media y baja. El realizar el proceso de extracción con parámetros de priorización, facilitó su uso en el módulo de configuración con llamadas personalizadas (Capítulo 4 sección 7).

Al realizar las pruebas de funcionalidad de la lógica de control por módulo independientes y al final pruebas de integración para validar el desarrollo y codificación del sistema Head-End bajo los estándares de una metodología de calidad, ayudó para que se obtuviera el menor número posible de errores en la realización de pruebas finales (Capitulo 4 sección 9).

Durante el desarrollo del proyecto fue fundamental aplicar una metodología ágil porque se administró de una forma adecuada el trabajo en equipo que se vio reflejado en cada etapa del desarrollo sobre todo en la integración del sistema. El tener objetivos a corto, mediano y largo plazo favoreció los tiempos de entrega final.

Cuando se realizó la integración entre los módulos con las vistas fue necesario una comunicación y trabajo directo entre los desarrolladores involucrados, haciendo uso de la persistencia de datos de Hibernate y selección de información a través de objetos que agilizó la integración y la organización adecuada del código.

Es importante tener en cuenta al momento de participar en un proyecto de desarrollo considerar que, si se está invirtiendo mucho tiempo en investigar o entender cómo funciona un proceso, es mejor preguntar al líder del proyecto para resolverlo lo antes posible y no retrasar entregas.

Con base a lo anterior podemos concluir que, si es posible diseñar, desarrollar e implementar la lógica de control para el proceso de desarrollo de un sistema Head-End para una red de dispositivos inteligentes.

La afirmación anterior está respaldada por el proyecto integrado con métodos independientes que se pueden adaptar a las características cambiantes que identifican a una red de dispositivos inteligentes. Al realizar la entrega final del sistema se demostró la ejecución exitosa de tareas sobre los medidores.

5.2 Trabajos futuros

El sistema Head-End desarrollado en este trabajo proporciona las bases para un sistema robusto, se diseñó la estructura principal del sistema con los módulos que lo comunican con dispositivos inteligentes medidores de electricidad, a partir del cual se pueden realizar los siguientes trabajos futuros:

- Desarrollar los módulos que corresponden a la comunicación del HES con dispositivos inteligentes medidores de agua y de gas, para este trabajo se dejaron diseñadas las tablas generales para los dos tipos de medidores.
- Debido a que el sistema se encuentra en su fase inicial, se realizaron pruebas de funcionamiento en un ambiente interno simulado, queda como trabajo futuro realizar pruebas de campo en un ambiente real conectando al sistema con la *red mesh* que se encuentre en funcionamiento continuo.
- Desarrollar métodos dinámicos para la extracción de información que se requiera analizar proveniente de la base de datos para su visualización en las pantallas gráficas que tiene acceso el usuario, en el sistema se dejaron métodos genéricos para la visualización estándar de la información.
- Desarrollar un módulo de alarmas que genere y muestre los datos representativos que informen al usuario de tareas no realizadas por parte de los dispositivos inteligentes.

- Otro tema que queda como abierto para desarrollo, es la implementación de módulos que permitan la administración de otros esquemas xsd de acuerdo a los requerimientos de otros MDMs.

Glosario de términos

Esquema XSD.- XML Schema Definition de sus siglas en inglés, es un mecanismo para comprobar la validez de un documento XML, es decir, definir su estructura: qué elementos, qué tipos de datos, que atributos, en qué orden, cuántas veces se repiten, etc.

HES.- Head End System por sus siglas en inglés, es un sistema de gestión de flujo de información. En esta tesis el HES gestiona la información entre un MDM y una red mesh.

IoT.- Internet of the Things por sus siglas en inglés, se refiere a la representación de objetos cotidianos conectados al internet, con los que podemos interactuar programando remotamente eventos en función de las tareas que puedan realizar.

MDM.- Meter Data Management por sus siglas en inglés, se encarga del procesamiento de gran volumen de datos, genera estimaciones y transformaciones de datos para entregar información confiable y oportuna.

Red Mesh.- Se compone de nodos de la malla que forman la columna vertebral de la red. Los nodos son capaces de configurarse automáticamente y volver a configurarse de forma dinámica para mantener la conectividad de la malla, lo que da a la malla las características de autoformación y auto reestructuración.

Socket.- Mecanismo que permite la conexión entre distintos procesos, habitualmente se utilizan para establecer comunicaciones entre distintas máquinas que estén conectadas a través de la red.

UDP.- Protocolo de comunicación que está orientado a la conexión, garantiza la correcta transmisión de los ficheros, mantiene el orden de los ficheros en la transmisión y cuando llegan los paquetes el receptor emite un mensaje de recepción.

Referencias

Carmona, M. A. (2018). Modelado e implementación de un Data Warehouse para un MDMS utilizando algoritmos ETL y VEE.

Leonard Richardson, M. A. (s.f.). RESTful Web APIs. United States of America.: O'Reilly.

CFE. (abril de 2015). Sistema de Infraestructura Avanzada de Medición (AMI). ESPECIFICACIÓN CFE G0100-05. México.

Christian Bauer, G. K. (2007). Java Persistence with Hibernate. MANNING Greenwich.

Lucy Sumi, V. R. (2016). Sensor enabled Internet of Things for Smart Cities. Fourth International Conference on Parallel, Distributed and Grid Computing.

Chunchi Gu, H. Z. (2014). Design and Implementation of Energy Data Collection System Using Wireless Fidelity (WiFi) Module and Current Transformer. International Conference on System Science ang Engineering (JCSSE) (pág. 5). Shanghai,China: IEEE.

Amr S. Abdelfattah, T. A.-S.-H. (2017). RSAM: An enhanced architecture for achieving web services reliability in mobile cloud computing. Journal of King Saud University – Computer and Information Sciences.

Apache.org. (2017). Apache Maven Project. Obtenido de Maven: https://maven.apache.org/

Artificial, D. d. (26 de 06 de 2014). El MVC en JavaServer Faces. Obtenido de Título de Experto Universitario en Desarrollo de Aplicaciones y Servicios con Java EE: http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion02-apuntes.html

Artificial, D. d. (26 de 06 de 2014). Introducción a JavaServer Faces. Obtenido de Titulo de Experto Universitario de Desarrollo de Aplicaciones y Servicios con Java EE: http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html

Bauer, C., & King, G. (2007). Java Persistence with Hibernate. United States of America: Manning.

Damminda Alahakoon, X. Y. (02 de 2016). Smart Electricity Meter Data Intelligence for Future Energy Systems: A Survey. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, 12(1).

Deepal Jayasinghe, A. A. (s.f.). Apache Axis2 Web Services 2nd Edition. Birmingham: PACKT. Open source.

Docs.oracle. (s.f.). The Java EE 6 Tutorial. Obtenido de Introduction to Web Services : http://docs.oracle.com/javaee/6/tutorial/doc/gijti.html

Documentation, C. (2004). Hibernate. Obtenido de Hibernate - Relational Persistence for Idiomatic Java: https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/index.html

EKM Metering. (2016). Meter Software. Obtenido de EKM Metering: http://www.ekmmetering.com/meter-software

EZ Meter. (2015). EZ Power Suite. Obtenido de EZ Meter: http://www.ezmeter.com/ez-power-suite/

Gajanan P. Bherde, D. M. (2016). Gajanan P. BherdeRecent Attack Prevention Techniques in Web Service Applications. International Conference on Automatic Control and Dynamic Optimization Techniques.

Guzmán, M. M. (2018). Interfaz Gráfica de Usuario para un Sistema Head End de una Red Distribuida de Medidores Inteligentes.

Hernández, L. A. (2018). Diseño e implementación de una base de datos relacional utilizando ORM programing technique, Spring Security e Hibernate para un Head End System.

Iman Khajenasiria, A. E. (2016). A review on Internet of Things solutions for intelligent energy control in buildings for smart city applications. 8th International Conference on Sustainability in Energy and Buildings.

Jayasinghe, D., & Azeez, A. (2011). Apache Axis2 Web Services. Lincoln Road: Packt Publishing Ltd.

Jelena Luki, M. R.-Z. (2016). A hybrid approach to building a multi-dimensional business intelligence system for electricity grid operators. Utilities Policy.

Jianwen Su, Y. T. (2017). Business Intelligence Revisited. Fifth International Conference on Advanced Cloud and Big Data.

Kaptein, P. (17 de diciembre de 2012). The MVC Pattern. Obtenido de Refactorización y patrones de diseño: http://patterns.instantinterfaces.nl/current/Refactoring-and-Design-Patterns-MVC-PATT.html#MVC-PATT-BAS-001

Mihaela Teliceanu, G. C. (2017). Consumption Profile Optimization in Smart City Vision. THE 10th INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING.

Miiller, J. K. (s.f.). Aspect Design with the Building Block Method. Springer Science+Business Media.

Nava, M. F. (2018). Metodología de aseguramiento de la calidad de un sistema head end (HES) de una red distribuida de medidores inteligentes.

Oracle.com. (s.f.). The Java Tutorials. New Properties. Obtenido de Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jaxp/properties/properties.html

Peishi Jiang, M. E. (2017). A service-oriented architecture for coupling web service models using the Basic Model Interface (BMI). Environmental Modelling & Software.

Proyectosagiles.org. (s.f.). Qué es SCRUM. Obtenido de proyectosagiles.org: https://proyectosagiles.org/que-es-scrum/

Roger S. Pressman, P. (2010). El principio W5HH. En P. Roger S. Pressman, Ingeniería del software. Un enfoque práctico. México, DF: McGrawHill.

Schneider Electric. (2016). Schneider Electric. Obtenido de Energy Management Software: http://www.schneider-electric.com/en/product-subcategory/4175-energy-management-software/?parent-category-id=4100

Sinnexus . (2016). Obtenido de Business Intelligence Informática estratégica: ¿Qué es Business Intelligence?

Sommerville, I. (2011). Ingeniería de Software. México: Pearson Educación.

Tejal Ghadge, N. B. (2016). Framework For Web Service Composition And Invocation. International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT).

Walls, C. (2015). Spring in Action. United States of America: Manning.

Xiujie Dong, Y. Z. (2010). The Design of Wireless Automatic Meter Reading System Based on SOPC. 2010 WASE International Conference on Information Engineering (pág. 4). IEEE.

Anexos

Anexo 1. Formato para el registro de pruebas

Nombre del Documento: BENCH TEST PROCEDURE REPORT	
Código: F_P06-DP_03	Nivel de Revisión: 02
Fecha de Emisión: 23-08-16	Página: 1 de 3

Part Number Information / Descripción N/P

Product P/N Description	WebService (First Iteration)
Product P/N	P001339
Product P/N Version	1
Test Procedure Number	1
Module or Functionality	Procesamiento de archivos XML
Test Date	27/03/2017
Test Engineer	Doris Castro Velasco
Result	
Reviewed by	Agustín Sánchez Atonal.

List of Equipment / Lista de Equipo

#	Equipment Name	Description	Calibration Due Date
1	MacBook Air	1.4 GHz Intel Core i5 4 GB 1600 MHz DDR3 Intel HD Graphics 5000 1536 MB 13.3 pulgadas (1440 x 900)	
2			
3			

Test Procedure Set-Up / Configuración de Procedimiento de Prueba

Test Case TC001001 / Caso de Prueba TC001001

Test Case Overview:

Pass/Fail Criteria:

Test Method:

Remarks:
N/A
Result: Pass

Test Case TC001002 / Case de Prueba TC001002

(Nava, 2018)

Anexo 2. Publicaciones

Memorias del Congreso Internacional
de Investigación Academia Journals
CICS Tuxpan 2017

Tuxpan, Veracruz, México
Septiembre 27 al 29, 2017

Interoperabilidad en sistemas informáticos de ciudades inteligentes a través de servicios web

Lic. Doris Castro Velasco[1], M.C. Jose Juan Hernandez Mora[2],M.C. Maria Guadalupe Medina Barrera[3], MSC. Agustín Sánchez Atonal[4]

***Resumen*—El concepto de internet de las cosas ha tenido gran auge en los desarrollos más innovadores como resultado de la continua evolución de las tecnologías de comunicación inalámbricas; uno de sus campos de implementación con mayor importancia son las ciudades inteligentes. Dentro de la arquitectura de las ciudades inteligentes, existen sistemas para la gestión del flujo de datos (HES) y para la generación de información (MDM) a partir de los datos recabados de la red de dispositivos inteligentes. El servicio web que se presenta en este trabajo está desarrollado bajo una arquitectura *RestFul* y combina tecnologías como servidores SFTP, validadores XSD, DOM Parsers, entre otras para la óptima comunicación entre los sistemas informáticos y la red de dispositivos inteligentes de cualquier prestador de servicios públicos. El proceso de ejecución comienza con una petición del MDM cliente al servicio web que extrae archivos XML comparándolos contra esquemas XSD para validar que la estructura XML sea la admitida por el HES. Si el esquema es correcto, convierte la información extraída en objetos java para posteriormente persistirlos a una base de datos (ORM).**

***Palabras clave*—IoT, Ciudades inteligentes, servicio web, MDM, HES.**

1. Introducción

Una ciudad inteligente es aquella que aprovecha las Tecnologías de la Información y de la Comunicación (TIC) en la creacion y mejoramiento de los sistemas que componen la infraestructura de la ciudad para que facilite el desarrollo sostenible, el incremento de la calidad de vida de los ciudadanos, administración adecuada de los recursos disponibles y la participación ciudadana activa(Iman Khajenasiria, 2016). Para lograr una ciudad inteligente se requiere de una ciudad digital en la que intervienen ciertos objetos inteligentes que generan datos que pueden ser recopilados a través de una red inalámbrica, es decir, se necesita del uso del internet de las cosas, aspectos de seguridad, ahorro de recursos a empresas que suministran servicios (gas, electricidad, teléfonos, etc.), automatización, respuestas en tiempo real y sustentabilidad, por mencionar algunos.

Internet de las Cosas (IoT) representa objetos cotidianos conectados al internet, con los que podemos interactuar programando remotamente eventos en función de las tareas que puedan realizar, si a cada objeto se le pudiera implementar una antena de radiofrecuencia, los datos generados se podrían transferir a través de una red para posteriormente ser procesados y aprovechados por el administrador de servicios en su toma de decisiones o la optimización de sus procesos.

Los objetos cotidianos con sistemas inteligentes generan gran volumen de datos por lo cual, para su procesamiento se requiere de un sistema MDM (Meter Data Management), el cual puede hacer estimaciones, y transformaciones entregando información confiable y oportuna.

MDM es un sistema de software que realiza el almacenamiento y la gestión de datos a largo plazo con lo que se puede generar información de facturación y estadísticas. La gestión de flujo de información entre el MDM y la red de dispositivos está a cargo de un Sistema Head-End (HES) que mantendrá un seguimiento de las comunicaciones y los dispositivos implicados. Para la interacción entre estos sistemas es obligatoria la interoperabilidad debido a que cada sistema reside en servidores que pueden estar en locaciones geográficas distintas, la arquitectura de las aplicaciones debe ser escalable e incremental para poder garantizar la adición de nuevas funcionalidades, además la optimización de las ya existentes e inclusive la integración de diversos protocolos de comunicación sin representar complicaciones. En este trabajo el canal de comunicación entre las dos entidades (MDM-HES) está configurado para trabajar bajo formatos XML consumiendo servicios web.

La generación continua de información requiere que sea almacenada por cada solicitud al servicio web, esto permite que se administren los registros almacenados en lugar de las solicitudes realizadas al servicio web. Para el almacenamiento de información se utiliza la persistencia de datos con Hibernate para realizar el mapeo objeto

[1]Lic. Doris Castro Velasco estudiante de maestría en Sistemas Computacionales en el Instituto Tecnológico de Apizaco, Tlaxcala. doris.castro.velasco@gmail.com
[2]M.C. José Juan Hernández Mora Catedrático de Maestría en Sistemas Computacionales del instituto Tecnológico de Apizaco, Tlaxcala. jjhmora@itamail.itapizaco.edu.mx
[3]M.C. Ma. Guadalupe Medina Barrea Catedrática de Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco, Tlaxcala. lupita_medina@hotmail.com
[4]Líder de proyecto en EosTech. Maestría en sistemas computacionales por el Instituto Tecnológico de Apizaco, Tlaxcala. agustin.sanchez@eostech.mx

ISSN 1946-5351
Vol. 9, No. 4, 2017

ACADEMIA JOURNALS

Memorias del Congreso Internacional
de Investigación Academia Journals
CICS Tuxpan 2017

Tuxpan, Veracruz, México
Septiembre 27 al 29, 2017

relacional (ORM). ORM permite convertir los datos de los objetos en un formato que puede guardar información en una base de datos (mapeo) creando una base de datos virtual donde los datos que se encuentran en la aplicación quedan vinculados a la base de datos (persistencia).

Este documento se encuentra organizado como sigue: en la sección 2 se presenta la arquitectura para el Servicio Web SFTP; en la sección 3 se describe el ORM para la asignación de objetos a través de anotaciones java; en la sección 4 se explican los componentes que integran el software HES y se finaliza con la sección 5 con algunas conclusiones y trabajos futuros.

2. Arquitectura del Servicio Web SFTP

El conectar muchos dispositivos heterogéneos a través de internet implica la necesidad de una arquitectura en capas que permita la flexibilidad de interacción. En la figura No. 1 se muestra la arquitectura empleada en el proyecto que se esta desarrollando. Esta arquitectura crea el flujo de trabajo entre los modelos que integran el Servicio Web SFTP, administran la ejecución de los modelos que se ejecutan en diferentes tiempos y coordina la información que fluye entre las capas para almacenar los datos en la Base de Datos (Peishi Jiang, 2017).

Para poder integrar modelos en el manejo de información, se introduce la conexión al servidor remoto y tener acceso a los archivos con la información. El acceso al Servicio Web se realiza a través de un proceso que hace la función de cliente para realizar la solicitud de ejecución de tareas, el Servicio Web accede al servidor remoto como parte de la comunicación entre el MDM y el HES.

A continuación se explican las capas que integran el Servidor Web SFTP mostrado en la figura No. 1:

Capa Aplicación Cliente. El MDM es un sistema independiente que administra la información almacenada proveniente de la red de dispositivos con la que puede generar diversos reportes para dejarlos diponibles a sistemas terceros. El MDM a través de solicitudes HTTP se comunica con la Capa de Servidor Web.

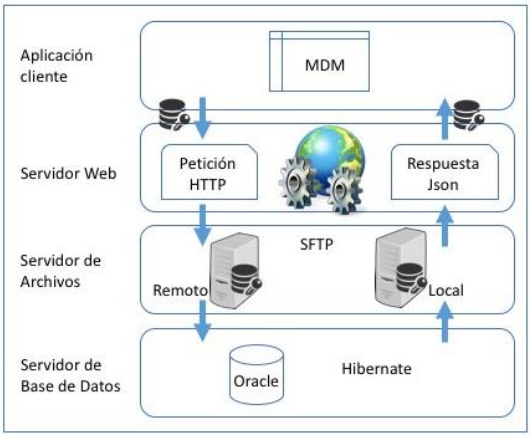


Figura No. 1. Arquitectura en capas de Servidor Web SFTP

Capa Servidor Web. El Servidor Web recibe las solicitudes realizadas por el MDM a través de la URI (Uniform Resource Locator) de acceso con el nombre de arhivo encriptado. El Servicio Web copia del servidor remoto al servidor local para extraer las tareas que se deben ejecutar. La respuesta de que la información se extrajo correctamente se realiza a traves de JSON (JavaScript Object Notation)

Capa Servidor de Archivos. Se trabaja con dos servidores para almacenar los archivos, uno para las peticiones de ejecución de tareas y otro para almacenar el resultado de la ejecución de tareas. Para seguridad de la información los archivos se almacenan encriptados.

Capa Servidor de Base de Datos. La información extraida de los archivos se almacena en la base de datos a través del *framework* Hibernate. Esta información se interpreta como tareas a ser ejecutadas en los dispositivos identificando la prioridad, intervalos de ejecución entre otras características.

ISSN 1946-5351
Vol. 9, No. 4, 2017

ACADEMIA JOURNALS

Memorias del Congreso Internacional
de Investigación Academia Journals
CICS Tuxpan 2017

Tuxpan, Veracruz, México
Septiembre 27 al 29, 2017

2.1. Verbos http, json y códigos de respuesta

REST es un estilo de arquitetura que abstrae los elementos dentro de un sistema hypermedia distribuido, a través de un conjunto de principios que definen la interacción entre distintos componentes. REST esta basado en HTTP (HyperText Transfer Protocol) que opera por petición y respuesta entre cliente y servidor utilizado cada vez que se transfiere un documento o se realiza una solicitud a un servicio web (Deepal Jayasinghe). El servicio Web SFTP desarrollado para el proyecto contiene los siguientes elementos, que son los que debe de tener un servicio WebRESTful:

- **URI del recurso.** El identificador de recurso uniforme es una cadena de caracteres que identifica los recursos en una red de forma unívoca. Los recursos REST se identifican mediante un URI exclusivo. Por ejemplo: http://webservice/ReadValidXML/ReadValid esto nos da acceso al servicio web ReadValidXML y al recurso ReadValid.
- **Tipo de representación de dicho recurso.** El tipo de respuesta que regresa puede ser JSON, XML y TXT. Para el objeto de estudio de este documento se utiliza el tipo JSON para comunicar la respuesta del proceso de almacenamiento de la información enviada por el cliente.
- **Operaciones soportadas.** HTTP soporta varios verbos (tipos de operaciones) que puede ser GET, PUT, POST, DELETE, PURGE entre otros. En el proyecto de Web Service SFTP se utiliza el verbo POST para que el servicio realice los procedimientos necesarios para almacenar la información extraida de un archivo en base de datos.
- **Hipervínculos.** En la respuesta se pueden incluir hipervínculos hacia otras acciones para poder realizar sobre otros recursos, el o los hipervínculos se incluyen en el contenido de respuesta, por ejemplo, en un objeto JSON se puede añadir una propiedad más con los hipervínculos a las acciones que admite el objeto.

HTTP maneja códigos de respuesta que estandarizan una forma de informar al cliente sobre el resultado de la solicitud. Para que el servidor devuelva el código de respuesta más apropiado, se debe de especificar el verbo apropiado al tipo de operación que realiza el servicio web. El significado de un código de respuesta HTTP no es muy preciso debido a que son genéricos (Leonard Richardson). Los códigos de respuesta HTTP más utilizados con REST son:

- **200 OK.** Indica que la solicitud se ha realizado correctamente.
- **201 Created**. Indica que la solicitud tuvo éxito y se creó un recurso, utilizado para confirmar el éxito de una solicitud PUT o POST.
- **400 Bad Request.**Sucede especialmente con las solicitudes PUT y POST cuando no pasan la validación o están en el formato incorrecto.
- **401 UnauthorizeAllowed.** El método http no es compatible con el de este recurso-
- **409 Conflict.** Indica un conflicto en la solicitud, por ejemplo si se esta utilizando una solicitud PUT para crear el mismo recurso dos veces.
- **500 Internal Server Error.** Indica que el procesamiento falla debido a circunstancia imprevistas en el lado del servidor.

2.2. Seguridad en un servicio web

En la arquitectura REST, los clientes y servidores intercambian representaciones de recursos utilizando una interfaz y un protocolo estandarizados. Estas características hacen que las aplicaciones REST sean simples y ligeras, por lo tanto, en cuanto al alcance de la confiabilidad, los servicios RESTFUL superan las limitaciones de los servicios SOAP (Simple Object Access Protocol) y logran mejores resultados. La combinación de diseño REST con otras tecnologías proporciona una buena escalabilidad del sistema.

Las aplicaciones Web ofrecen diversos tipos de servicios a los usuarios de Internet. De modo que, la prevención de los servicios web de los ataques se convierte en tarea esencial y desafiante para proporcionar servicios seguros a los usuarios. Existen diversos tipos de ataques como la vulnerabilidad XXE, XSS, inyección de SQL, codificación de solicitudes, ataques DoS, etc., estos tipos de ataques en páginas web degradan el rendimiento o el bloqueo completo de servicios a los usuarios. (Amr S. Abdelfattah, 2017)

La vulneravilidad XXE (Xml eXternal Entity) se produce en aplicaciones que hacen uso de “parsers” XML. Es decir aplicaciones que reciben como entrada un documento XML y para procesarlo hacen uso de alguna librería de parseo. En el Servicio Web SFTP, objeto de estudio, se utiliza la librería javax.xml.parsers para procesar los archivos XML que recibe el servicio web, para prevenirlo se utilizan las propiedadesaccess_external_dtd y access_external_schema(Oracle.com, s.f.) para evitar que se trate de malformar las peticiones, porque podrían añadirnos funcionalidades al interpretar XML, por ejemplo, inyectando código para ler un archivo de nuestro servidor local y enviar su contenio a un seridor remoto.

ISSN 1946-5351
Vol. 9, No. 4, 2017

Academia Journals

Memorias del Congreso Internacional
de Investigación Academia Journals
CICS Tuxpan 2017

Tuxpan, Veracruz, México
Septiembre 27 al 29, 2017

Otra medida de seguridad que se esta tomando es almacenar los archivos XML encriptados, tanto en el servidor remoto como en el servidor local, asi como la encriptacion de información vulnerable que se encuentre almacenada en base de datos.

3. Modelado para asignación de objetos con ORM

La infomación que se almacena en la base de datos proviene de los archivos que proporciona el MDM para indicar las tareas que se ejecutan sobre la red de dispositivos. Para realizar el mapeo de información, se diseñaron clases con caracteristicas generales (clases genéricas) para recibir tareas, dispositivos y atributos de cada uno, este diseño general permite recibir la información que se extrae de los archivos. Con la información en las clases genéricas se realiza el mapeo a las clases que corresponden a las entidades de la base de datos ver Figura No. 2.

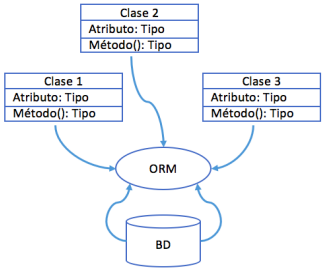


Figura No. 2. Diagrama ORM

Para relizar el mapeo objeto-relacional (ORM), se utiliza hibernate en plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y las anotaciones en los *beans* de las entidades que permiten establecer las relaciones. Para poder realizar la persistencia de objetos a base de datos se desarrollaron clases por cada entidad utilizando las anotaciones @Entity, @Table, @Id, @Column, entre otras, para el mapeo de entidad-tabla. (Christian Bauer, 2007)

4. Componentes de software del HES

El sistemas para la gestión del flujo de datos (HES) se encarga de recibir las peticiones que realice el MDM a través de archivos XML y realizar las operaciones que se requieren para la comunicación entre la red de dispositivos y empresas que suministran servicios, este proyecto se desarrolla para la Comisión Federal de Electricidad (CFE). En la Figura No. 3 se muestra el diagrama general del sistema completo en el que interactua el HES.

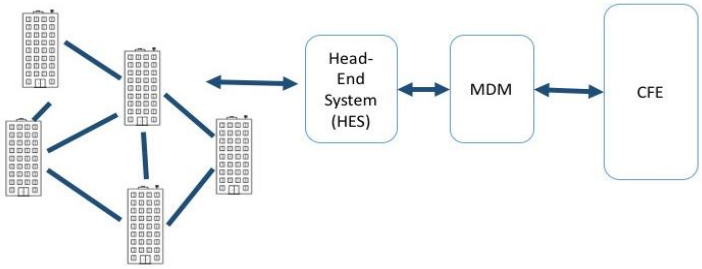


Figura No. 3. Diagrama General del Sistema de interacción con HES

Para realizar un sistema complejo y de gran extensión se requiere que se desarrolle bajo una organización que permita realizar el sistema por partes, por lo que se implementa “building blocks” para un desarrollo incremental de bloques de código funcionales independientes para que con la integración de cada componente se obtenga el sistema final, esto permite que cada bloque de código sea independiente y de fácil mantenimiento.

ISSN 1946-5351
Vol. 9, No. 4, 2017

Memorias del Congreso Internacional
de Investigación Academia Journals
CICS Tuxpan 2017

Tuxpan, Veracruz, México
Septiembre 27 al 29, 2017

La estructura del proyecto se adminstra a través del marco de trabajo Maven, que facilita la definición del proyecto y la integración de las librerias de java que se configuran en el archivo pom.xml para que sean descargadas desde los repositorios de la web.

El HES para realizar sus funciones se integra de los siguientes componentes: Servicio Web SFTP, Servicio de ejecución de tareas y Manejador de tareas.

El Servicio Web SFTP se encarga de la comunicación HTTP con el MDM mediante archivos XML, administrando la petición de ejecución de tareas que se realizan en la red de dispositivos. El Servicio Web SFTP se integra de los siguientes componentes: Validador de archivos XML, Extracción de tareas del archivo XML, Persistencia en Base de Datos.

- Validador de archivos XML. En este componente el sistema se encarga de validar si los archivos de petición XML son los que espera de acuerdo a esquemas XSD. En los esquemas se especifica la estructura que debe de tener el archivo XML de acuerdo a lo que espera el sistema encontrar para poder realizar el proceso de extracción de tareas. Para identificar que MDM esta realizando la petición, se cuenta con un estandar en el nombre de los archivos XML donde el HES puede identificar el MDM que se está comunicando a través del Servicio Web. El sistema, una vez identificado a su cliente, utiliza el esquema XSD correspondiente para la validación de los archivos XML del MDM que solicita.
- Extracción de tareas de archivo XML. Componente que se utiliza para realizar la extracción de tareas que se almacenan en espera de su ejecución. Cuando se identifica que el archivo XML es válido respecto al esquema XSD, se inicia el proceso de extracción de tareas para obtener un arreglo de objetos identificando los atributos que corresponden a cada tarea.
- Persistencia en Base de Datos. Conjunto de metodos que se encargan de realizar la interacción con la base de datos. Una vez que se tienen las tareas en una lista de objetos, se inicia el proceso de mapeo a las entidades de la base de datos para guardar los registros en la base de datos.

Servicio de ejecución de tareas. Este componente se encarga de mantener la comunicación entre la red de dispositivos y las tareas que deben ser ejecutadas. Aabre puertos de comunicación con los dispositivos para que ejecuten las tareas, tiene el control de las tareas que se estan ejecutando y si existen interrupciones manejar las tareas para que se de seguimiento.

Manejador de tareas. Administra las tareas que deben ser ejecutadas, controlando las excepciones que se generen para continuar con las tareas sin perder la constancia y coherencia en la ejecución. Administra las priorización de tareas y almacena el resultado de la ejecución de las tareas.

En la Figura 4 se muestra el diagrama de flujo del Servicio Web SFTP donde se observa el inicio del servicio con una petición al servicio a través de un archivo XML que se obtiene a través de SFTP para acceder al servidor remoto del MDM solicitante, se almacena el archivo en el servidor local del Servidio Web para poder trabajarlo en los siguientes procesos. Una vez almacenado se valida la estructura del archivo XML comparandolo con el esquema XSD válido para el proceso. Posteriormente se realiza el mapeo de la información obtenida del archivo a las entidades de la base de datos a través de objetos para poder realizar la persistencia la base de datos.

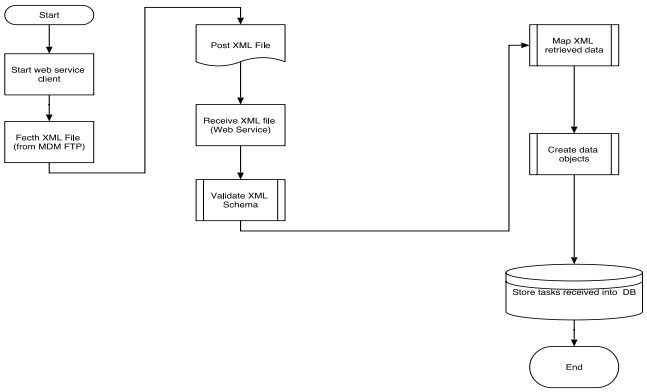


Figura 4. Diagrama de Flujo del Servicio Web SFTP

ISSN 1946-5351
Vol. 9, No. 4, 2017

ACADEMIA JOURNALS
OPUS PRO SCIENTIA ET STUDIUM

5. Conclusiones y trabajos futuros

La interoperabilidad es una característica crítica en el diseño y la construcción de servicios de IoT para satisfacer los requisitos de los usuarios. Uno de los desafíos en la realización de la interoperabilidad es que diferentes proveedores interpretan el mismo estándar de forma diferente. Exponer modelos a través de servicios web reduce significativamente la barrera de interoperabilidad de la comunicación entre modelos, y un cliente solo necesita configurar el protocolo de comunicación utilizado por el servicio modelado.

El desarrollo de un Servicio Web SFTP, que implementa todas las tecnologías mencionadas en este artículo, a través de su arquitectura de cuatro capas, facilitará la comunicación con diferentes MDMs (clientes) definiendo el protocolo de comunicación de acuerdo al tipo de cliente y la interoperabilidad con dispositivos heterogéneos utilizando las clases genéricas en los procesos de mapeo de información entre la capa de Servidor de Archivos y la capa de base de datos.

Este trabajo corresponde a la recepción de ejecución de tareas que solicita como cliente el MDM. Se extraen las tareas a ejecutar para almacenarlas en la base de datos y dejarlas disponibles en trabajos futuros se debe desarrollar un proceso independiente que obtenga las tareas priorizadas y administre la lista de ejecución de las tareas controlando las interrupciones y excepciones que se generen para continuar con las tareas sin perder constancia y coherencia en la ejecución, es decir, un manejador de tareas. Las pruebas y resultados del presente trabajo se estan realizando en la empresa para la que se desarrolló el proyecto.

Otro tema que queda como abierto para desarrollo, es la implementación de módulos que permitan la administración de otros esquemas xsd de acuerdo a los requerimientos de otros MDMs.

Referencias

Leonard Richardson, M. A. (s.f.). *RESTful Web APIs*. United States of America.: O'Reilly.

Christian Bauer, G. K. (2007). *Java Persistence with Hibernate.* MANNING Greenwich.

Amr S. Abdelfattah, T. A.-S.-H. (2017). RSAM: An enhanced architecture for achieving web services reliability in mobile cloud computing. *Journal of King Saud University – Computer and Information Sciences*.

Deepal Jayasinghe, A. A. (s.f.). *Apache Axis2 Web Services 2nd Edition.* Birmingham: PACKT. Open source.

Iman Khajenasiria, A. E. (2016). A review on Internet of Things solutions for intelligent energy control in buildings for smart city applications. *8th International Conference on Sustainability in Energy and Buildings*.

Oracle.com. (s.f.). *The Java Tutorials. New Properties*. Obtenido de Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jaxp/properties/properties.html

Peishi Jiang, M. E. (2017). A service-oriented architecture for coupling web service models using the Basic Model Interface (BMI). *Environmental Modelling & Software*.

AcademiaJournals.com

Universidad Veracruzana

Congreso Internacional de Investigación Academia Journals Tuxpan 2017
Ciencias y Sustentabilidad

Certificado

otorgado a

Lic. Doris Castro Velasco
M.C. Jose Juan Hernandez Mora
M.C. Maria Guadalupe Medina Barrera
MSC. Agustín Sánchez Atonal

por su artículo intitulado

Interoperabilidad en sistemas informáticos
de ciudades inteligentes a través de servicios web

Artículo No. TX17-509

El artículo fue presentado en el congreso llevado a cabo los días 27 al 29 de septiembre del año 2017 en Tuxpan, Veracruz, México y se publicó (1) en el portal de internet AcademiaJournals.com con ISSN 1946-5351 online e indizado por Fuente Académica Plus de EBSCO y (2) en el e-libro intitulado *Investigación en la Educación Superior: Eje de Competencias,* mismo que cuenta con versiones ISBN 978-1-939982-30-8 (CDROM) e ISBN 978-1-939982-29-2 (online).

FACULTAD DE CONTADURIA
ZONA POZA RICA-TUXPAN
DIRECCION

Dr. Edalid Álvarez Velázquez
Presidente de la Comisión Organizadora
Directora de la Facultad de Contaduría
Universidad Veracruzana Región Poza Rica-Tuxpan

Dr. Rafael Moras
Editor, Academia Journals
Profesor de Ing. Industrial
St. Mary´s University, San Antonio, TX, EEUU

Memorias del Congreso Internacional de Investigación Academia Journals Celaya 2017 Celaya, Guanajuato, México Noviembre 8, 9, y 10, 2017

Desarrollo del módulo de administración de tareas para un sistema Head End

Lic. Doris Castro Velasco[1], M.C. José Juan Hernández Mora[2], M.C. María Guadalupe Medina Barrera[3], M.S.C. Agustín Sánchez Atonal[4]

***Resumen*— El proyecto descrito en el presente artículo es sobre la implementación de un sistema informático (Head End) que ayuda en la gestión de flujos de información para dar soporte a una solución en tecnológica para ciudades inteligentes, mejorando la eficiencia en la prestación de servicios públicos y en consecuencia la calidad de vida de los usuarios. La arquitectura del sistema está basada en *Building Blocks* que garantiza un incremento escalable, modular y de fácil mantenimiento. Para el desarrollo se utilizó la metodología ágil SCRUM, de tal forma que las iteraciones son mensuales creando los *milestones* propuestos en los componentes de software de una arquitectura Modelo Vista Controlador (MVC). Este artículo se enfoca en la creación de un módulo de administración de tareas, el cual emplea un *Thread Pool* para la rutina periódica de selección y ejecución de estas de manera priorizada. Cada tarea a su vez es recuperada, interpretada y actualizada utilizando Hibernate y JSP.**

***Palabras clave*— ThreadPool, ORM, Hibernate, HES, ciudad inteligente**

1. Introducción

Un sistema para la gestión de flujo de datos (HES – Head End System) es un intermediario entre los solicitantes de información o ejecución de funcionalidades hacia dispositivos y una red de dispositivos autónomos. Para poder decir que una ciudad es inteligente se requiere que la ciudadanía participe a través de sistemas que les permitan mantener una interacción con el entorno donde viven, a través de esta información que genera la ciudadanía y la información generada por los mismos dispositivos electrónicos, se podrá disponer de los datos actuales sobre, por ejemplo, el flujo vehicular, espacios cercanos de estacionamiento, puntos de la ciudad donde se concentran más deshechos y poder realizar una administración adecuada de recolección de basura, entre otros, lo que ayuda a tener una mejor vida al ciudadano.

Este artículo está enfocado en el desarrollo de un administrador de tareas para un sistema Head End que permite la interacción entre medidores de electricidad que se encuentran comunicados mediante una red, recibe las solicitudes de ejecución de las respectivas funcionalidades, las procesa y las envía a la red de dispositivos para que sean ejecutadas. Las peticiones enviadas en tiempo real requieren que sean administradas para que no genere conflicto entre ellas.

El manejador de solicitudes se desarrolló bajo la metodología de *Building Blocks* que permite la creación de bloques funcionales del proyecto general, dando como resultado un proyecto escalable y con mejor calidad debido a las pruebas realizadas por bloques terminados. El desarrollo se complementó con la metodología ágil SCRUM con iteraciones mensuales en las que se desarrollaron *millestons* propuestos en los componentes de software obtenidos de una arquitectura Modelo Vista Controlador (MVC).

El módulo emplea un *Thread Pool* configurable en el número de hilos que trabaja, esto permite la optimización de recursos al momento de distribuir la asignación de procesos en los hilos programados, que ejecuta una rutina configurable periódica, en la que realiza una selección de registros de solicitudes desde la base de datos que deben ser ejecutadas tomando en cuenta su priorización. Cada entrada de la base de datos que es recuperada, se interpreta y actualiza a través de Hibernate.

Para la utilización de Hiberntate se hizo uso del patrón de diseño java Interface, para separar las funciones (firma de métodos) de las implementaciones (cuerpo del método) así cualquier solicitud que coincida con la firma de la interfaz de un objeto también pueda ser enviada a ese objeto, independientemente de su aplicación. A través de la interfaz desarrollada se realiza la recuperación, interpretación y actualización de las tareas.

[1] Lic. Doris Castro Velasco estudiante de maestría en Sistemas Computacionales en el Instituto Tecnológico de Apizaco, Tlaxcala. doris.castro.velasco@gmail.com
[2] M.C. José Juan Hernández Mora Catedrático de Maestría en Sistemas Computacionales del instituto Tecnológico de Apizaco, Tlaxcala. jjhmora@itamail.itapizaco.edu.mx
[3] M.C. Ma. Guadalupe Medina Barrea Catedrática de Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco, Tlaxcala. lupita_medina@hotmail.com
[4] Líder de proyecto en EosTech. Maestría en sistemas computacionales por el Instituto Tecnológico de Apizaco, Tlaxcala. agustin.sanchez@eostech.mx

Para la visualización de la información que puede ser consultada por un usuario, se implementó la tecnología de JSP para crear páginas web dinámicas basadas en HTML.

2. Metodologías aplicadas en el desarrollo del sistema Head End

El proceso genérico de ingeniería requiere la aplicación de metodologías de trabajo, que no se limita a que se lleve a cabo el proyecto con una, se presta a la integración de diferentes técnicas dependiendo el tipo y tamaño de proyecto que se va a realizar. Es importante tomar en cuenta la exigencia actual del cliente que requiere soluciones en el menor tiempo posible pero no por eso se deben aplicar metodologías que acorten el tiempo y descuiden la calidad, se debe encontrar un balance al momento de elegir la forma de trabajo para que se pueda ir mostrando al cliente avances del proyecto y él pueda ir detectando aspectos no contemplados por el cliente.

En el método de *building blocks*, un método de objeto se define como un bloque mínimo de funcionalidad. Esto corresponde a una porción del diseño. La funcionalidad del sistema completo está dividida por el conjunto de los módulos diseñados, facilitando el desarrollo de sistemas a gran escala debido a que la integración de módulos puede ser infinita gracias a su funcionalidad individual.

El implementar metodologías ayudan a establecer orden en la forma de trabajo, estar conscientes en qué fase se encuentra el proyecto, qué se requiere para finalizarlo y poder corregir los errores en fases tempranas antes de que se entregue el proyecto final, lo que da como resultado un producto de calidad, por lo que se utilizó la metodología ágil Scrum que otorga adaptabilidad a los cambios de requerimientos constantes.

Scrum trabaja a través de *sprints* que son unidades básicas para una iteración llevada a cabo por los miembros del equipo de desarrollo. El equipo puede completar varios *sprints* durante el desarrollo del proyecto, esto da la oportunidad de ir mejorando cada módulo conforme se van analizando en cada *milestone* definido.

Cada iteración consta de cinco etapas: reunión de planificación de sprint, *scrums* diarios, trabajo de desarrollo, revisión de sprint y retrospectiva del sprint. Cada sprint se inició con una reunión de planeación, especificando la duración y los objetivos para cada integrante del equipo.

A lo largo del desarrollo del módulo se realizaron reuniones diarias, antes de iniciar las actividades del día, para llevar el control del avance de cada miembro del equipo de desarrollo en los diferentes módulos que integran el sistema Head End, en la que se analizaba el avance hasta ese momento y que se necesitaba para continuar el desarrollo con el menor contratiempo posible. Las reuniones diarias ayudaron a detectar y mitigar posibles retrasos al detectar a tiempo contratiempos que pudieran retrasar la entrega. La revisión del *sprint* se realizó con base a retrospectivas de los módulos.

Para la correcta aplicación de *building blocks* y *Scrum* se partió de la arquitectura general del HES, Modelo Vista Controlador (MVC), que permite ver al sistema en tres bloques principales, el módulo de administración de tareas se encuentra en la lógica de control del sistema que corresponde al Controlador del MVC. El Controlador del HES se compone a su vez de varios componentes de software que permiten un mejor funcionamiento y aprovechamiento de los recursos.

3. Integración del módulo administrador de tareas en el sistema Head End

Un sistema Head End que ayuda a la gestión de flujo de información, se compone de varios submódulos que interactúan entre sí para lograr el propósito. El sistema Head End para el cual se trabajó el módulo de administración de manejo de operaciones está integrado por un *Web Service* que está a la espera de que un sistema MDM (Meter Data Management) solicite comunicación los dispositivos para enviar estas solicitudes al *Task Manager* (administrador de tareas), y al obtener la respuesta de la red de dispositivos, regresa al MDM la información obtenida.

El *Windows Service* cada determinado *Tick* de un *Timer* solicita al módulo de selección y priorización las tareas que deben ser ejecutadas en los dispositivos para enviar las instrucciones al *UDP Comunication* que se encarga de realizar la comunicación con los dispositivos que integran la red; el Decoding Device Responses decodifica la información obtenida de los dispositivos para que sea interpretada y procesada.

El módulo administrador de tareas, objeto de estudio de este artículo, se encarga de la selección parametrizada de cada petición que debe ser enviada a la red de dispositivos y que el dispositivo con el que se quiere interactuar reciba las acciones que debe realizar. Una vez que el dispositivo realiza las acciones, emite datos del resultado de las acciones que realizó, esta información la recibe el módulo administrador a través del *Decoding Device Responces* para poder generar la respuesta que envía el *Web Service* al MDM. En la Figura No. 1 se muestra la arquitectura general del sistema Head End.

Memorias del Congreso Internacional de Investigación Academia Journals Celaya 2017 Celaya, Guanajuato, México Noviembre 8, 9, y 10, 2017

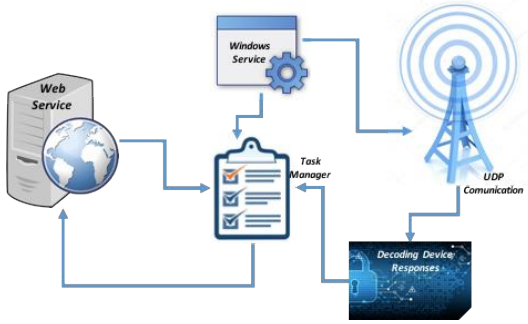


Figura No. 1. Arquitectura general del sistema Head End

El administrador de tareas trabaja en la priorización de la ejecución de procedimientos que se van solicitando en tiempo real de tal forma que no se realice duplicidad en la ejecución, controla las excepciones que se generen para continuar con los demás registros recuperados sin perder la constancia y coherencia en la ejecución, controla el manejo de excepciones para prevenir la interrupción abrupta de la aplicación y genera archivos XML de respuesta de la red de dispositivos.

4. Arquitectura a través de Thread Pools, entidades y objetos

El sistema Head End interactúa con una red de dispositivos tipo mesh que permite que todos los elementos inalámbricos se comuniquen entre sí, independientemente del punto de acceso. Es decir, los dispositivos que actúan como tarjeta de red pueden no mandar directamente su paquete al punto de acceso si no que pueden pasárselos a otro módulo de radiofrecuencia para que llegue a su destino.

Una red mesh involucra cientos de dispositivos, por lo que el HES requiere de un sistema que soporte la multitarea a través de hilos de ejecución, que permita ejecutar varios procesos a la vez realizándolo en mejor tiempo y haciendo más eficiente el sistema.

4.1 Criterios de priorización de tareas en *Thread Pools*

La alta concurrencia que tiene el sistema tiene como característica recibir varias peticiones del MDM que se deben atender lo más pronto posible. Por lo que se integró un *thread* pool donde se puede definir un número determinado de *threads*, en los que se van procesando los registros de la base de datos que se toman de una cola. De esta forma, se procesan varias peticiones simultáneamente en un determinado número de *trheads*.

El módulo de administración de tareas realiza la selección de trabajos identificando la prioridad que tienen, estas pueden ser de prioridad: alta, media y baja. El número de tareas puede variar dependiendo de la cantidad de peticiones que haya realizado el MDM y el número de dispositivos que afecte cada una de ellas.

A cada uno de los procesos se le asigna un hilo de ejecución en un *thread pool* que se crea cada determinado tiempo. El intervalo de tiempo para ir creando un *thread pool* y los hilos de ejecución que lo integran son configurables. Cuando llegan las siguientes operaciones, se busca primero los *threads* que ya hayan terminado de procesar, para procesar las peticiones en los *threads* que se encuentran libres.

La forma como se van buscando hilos libres se realiza tomando en cuenta la prioridad de la tarea debido a que se asignaron tres *threads* a prioridad alta, dos a prioridad media y uno a prioridad baja, como se muestra en la figura No. 2. El que se asignen un determinado número de *threads* a ciertas operaciones por su prioridad solo es para asignar el recurso, pero si llega una tarea y encuentra su *thread* asignado ocupado utilizará el que se encuentre libre empezando a buscar en la prioridad baja inmediata.

Para aplicaciones con procesos que se deben de ejecutar simultáneamente, los *thread pools* son una solución muy útil que se sirven de los núcleos de los procesadores de la computadora para trabajar de forma concurrente. Lo ideal es crear el mismo número de *threads* que equivalgan al número de núcleos que tenga el procesador donde se ejecute la aplicación.

ISSN 1946-5351 online
Vol. 9, No. 6, 2017

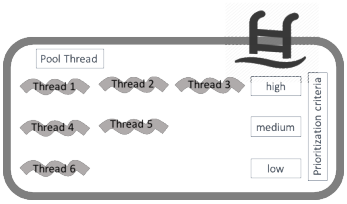


Figura No. 2. Thread Pools para priorización de tareas

4.2 ORM, creación de entidades, consulta de entidades, anidamiento de objetos

El ORM (Object Relational Mapping) es el modelo de programación que permite transformar las tablas de una base de datos en entidades con la finalidad de simplificar las tareas básicas de acceso a los datos. El ORM tiene una capa intermedia que abstrae las funciones de los distintos Sistemas de Gestión de Base de Datos (SGBD) lo que permite que la aplicación pueda ser migrada a cualquiera de las principales bases de datos existentes.

En el módulo de administración de tareas se utiliza el ORM Hibernate de Java, con el que se diseñaron métodos genéricos para interactuar con la base de datos y métodos para extracción específica de información. Se utilizó el patrón de diseño Interface que separa las funciones (firma de métodos) de las implementaciones (cuerpo del método) así cualquier solicitud que coincida con la firma de la interfaz de un objeto también pueda ser enviada a ese objeto, independientemente de su aplicación.

Una vez que se desarrolló la Interface con los métodos básicos de interacción con una tabla en una base de datos. Se crearon los DAO (Data Access Object) por cada entidad correspondiente a las tablas existente para la aplicación y se implementó a cada DAO la interface para poder utilizar de forma genérica los métodos básicos.

Para hacer uso de los métodos basta con instanciar la clase DAO que corresponda a la entidad que se va a trabajar. Al acceder a cada método, Hibernate en conjunto con los métodos desarrollados, se encarga de acceder al SGBD que se encuentre configurado para la aplicación.

La estructura de las entidades se encuentra diseñada con anidamiento de objetos, es decir, que sus atributos pueden llegar a ser objetos que tienen sus propios atributos, por ejemplo, algunos atributos de la entidad ElectricalDevice son: Id (long), nombre (String), address (Address), strHWModel (String), strHWVersion (String) y strStatus (String)

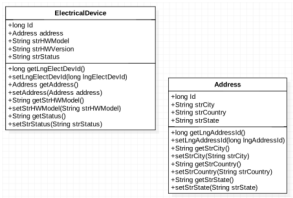


Figura No. 3 Ejemplo de anidamiento de objetos

Como se puede visualizar en la Figura No. 3, el atributo address es de tipo Address que tiene su propia entidad con sus atributos y métodos. Para acceder a al método getStrCity a través de la entidad Electrical Device, se instancia la clase: objeto Electrical Device y se accede de la siguiente manera ojbeto.address.getStrCity(), con el que se tiene acceso a la información que obtenga el método.

Los métodos de las entidades se utilizan para interactuar con la información almacenada en la base de datos y que se pueda visualizar en las vistas realizadas en jsp.

5. Pruebas y resultados de la implementación del módulo

Con base a la metodología propuesta se realizaron pruebas por cada iteración establecida lo que garantiza la calidad del software dejando al mínimo las fallas que pueda tener. Por lo que se fueron realizando de análisis con caja blanca por bloques de código para comprobar la correcta funcionalidad. Y al finalizar el módulo se utilizó caja negra para validar la funcionalidad integral.

Las pruebas de caja blanca que se realizaron fueron las siguientes:

- Activación de tiempos para la creación de *thread pool* en el periodo establecido: Se utilizó el método *java.util.Timer.schedule(TimerTask task, long delay, long period)* en combinación con el método *java.util.TimerTask.TimerTask()* para obtener la funcionalidad de la ejecución de consulta a la base de datos en el tiempo especificado.
    - Se imprimió en la consola el resultado de la información obtenida de la base de datos, validando que se realice en el tiempo establecido, obteniendo los tiempos esperados que se configuración para cada prueba.
- Extracción de tareas priorizadas
    - Para la validación de que la consulta a la base de datos fueran las tareas correspondientes a la priorización, se desplegó en terminal el resultado para validar que los registros obtenidos fueran los correctos se comparó con los registros en base de datos.
- Distribución de procesos en los hilos configurados
    - La configuración seleccionada fue de un *thread pool* con 6 hilos de ejecución dando prioridad a las tareas de la siguiente manera: tres hilos alta, dos hilos para media y un hilo para baja.
    - Para validar la configuración se realizaron varias combinaciones de consulta según su priorización para comprobar que no afecta el proceso de administración de procesos. Se crearon registros para generar selección de registros con las siguientes combinaciones de prioridad, ver tabla No. 1

	Tareas			Cantidad de hilos para priorizar		
	alta	media	baja	3	2	1
Caso 1	5	1	0	3 altas	2 altas	1 media
Caso 2	4	2	0	3 altas	1 alta, 1 media	1 media
Caso 3	3	3	0	3 altas	2 medias	2 medias
Caso 4	2	4	0	2 altas, 1 media	2 medias	1 media
Caso 5	1	5	0	1 alta, 2 medias	2 medias	1 media
Caso 6	5	0	1	3 altas	2 altas	1 baja
Caso 7	4	0	2	3 altas	1 alta, 1 baja	1 baja
Caso 8	3	0	3	3 altas	2 bajas	1 baja
Caso 9	2	0	4	2 altas, 1 baja	2 bajas	1 baja
Caso 10	1	0	5	1 alta, 2 baja	2 bajas	1 baja
Caso 11	0	5	1	3 medias	2 medias	1 baja
Caso 12	0	4	2	3 medias	1 media, 1baja	1 baja
Caso 13	0	3	3	3 medias	2 bajas	1 baja
Caso 14	0	2	4	2 medias, 1 baja	2 bajas	1 baja
Caso 15	0	1	5	1 media, 2bajas	2 bajas	1 baja

Tabla No. 1 Casos de prueba para el *Thread pool*

- Generación de excepciones sin que se interrumpan los procesos
    - Se generaron diferentes tipos de fallos que pudieran ocurrir al estar ejecutando la aplicación, validando que los errores se registraran en la excepción correspondiente a través de un log de errores.
    - Se validó que tuviera un *catch* para excepciones no controladas y que se registrara el error que se produce.

Las pruebas de caja negra se realizaron partiendo de los requisitos funcionales donde se especifica que se debe enviar a ejecución las tareas de acuerdo a su priorización sin generar duplicidad y cachando las excepciones que se generen para no detener la ejecución de todos los procesos. Se integró el sistema en un ambiente simulado para validar la ejecución de las tareas encendido y apagado en un gabinete prueba con tres medidores. Las tareas se fueron ejecutando de acuerdo a su priorización y en el tiempo configurado. Los dispositivos se apagaban y encendían según lo que indicó cada petición en las tareas.

6. Conclusiones

El desarrollo de sistemas sobre una arquitectura modular garantiza la escalabilidad y mantenimiento que debe ser enfocado para que se pueda implementar en proyectos para ciudades inteligentes, obteniendo ciudades sostenibles económica, social y medioambientalmente. Actualmente en México el enfoque para sistemas inteligentes apenas inicia por lo que se deben aplicar metodologías de desarrollo que faciliten la entrega de un software con funcionalidad independiente y mayor calidad.

El desarrollo del módulo de administración de tareas para un *Head End* bajo la metodología ágil SCRUM facilitó la descomposición del proyecto en pequeños procesos funcionales y permitió la incorporación del administrador de peticiones con menores fallas dejando al módulo a disposición de uso para su utilización en otros proyectos si fuera necesario. Al integrarlo se garantizó su funcionamiento integral por el resultado obtenido en las pruebas.

Al utilizar un ORM para la interacción con la base de datos permitió almacenar y obtener información de forma automática sin necesidad de escribir de forma directa consultas SQL. Hace al módulo independiente de la base de datos, haciéndolo de fácil mantenimiento si en un futuro se requiere trabajar con otra base de datos. Además, garantiza que en memoria solo se tiene cargada una única instancia de cada entidad.

La correcta administración en la ejecución de tareas de acuerdo a las prioridades que se asignan es de suma importancia para que el sistema *Head End* refleje un funcionamiento óptimo al momento de solicitar la ejecución de peticiones y que los dispositivos interactúen como si fuera en tiempo real. Si los registros son obtenidos de forma errónea o no se manejan adecuadamente las excepciones generadas, provocaría que el HES no realice las peticiones con la prioridad requerida.

Actualmente el *Head End,* de comunicación con medidores inteligentes, trabaja con el administrador de tareas descrito en este artículo y se comunica con la red de dispositivos a través de un *UDP (User Datagram Protocol)* que permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, intercambiando información en forma de bloques de bytes siendo su gran ventaja que provoca poca carga adicional en la red ya que es sencillo y emplea cabeceras simples. Como trabajo futuro queda la integración de un módulo de comunicación de Radio Frecuencia (RF) que permita una transferencia serial.

Referencias

Christian Bauer, G. K. (2007). *Java Persistence with Hibernate*. MANNING Greenwich.

Lucy Sumi, V. R. (2016). Sensor enabled Internet of Things for Smart Cities. *Fourth International Conference on Parallel, Distributed and Grid Computing*.

Chunchi Gu, H. Z. (2014). Design and Implementation of Energy Data Collection System Using Wireless Fidelity (WiFi) Module and Current Transformer. *International Conference on System Science ang Engineering (JCSSE)* (pág. 5). Shanghai,China: IEEE.

Damminda Alahakoon, X. Y. (02 de 2016). Smart Electricity Meter Data Intelligence for Future Energy Systems: A Survey. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, 12*(1).

Deepal Jayasinghe, A. A. (s.f.). *Apache Axis2 Web Services 2nd Edition*. Birmingham: PACKT. Open source.

Iman Khajenasiria, A. E. (2016). A review on Internet of Things solutions for intelligent energy control in buildings for smart city applications. *8th International Conference on Sustainability in Energy and Buildings*.

Mihaela Teliceanu, G. C. (2017). Consumption Profile Optimization in Smart City Vision. *THE 10th INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING*.

Miiller, J. K. (s.f.). Aspect Design with the Building Block Method. *Springer Science+Business Media*.

Oracle.com. (s.f.). *The Java Tutorials. New Properties*. Obtenido de Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jaxp/properties/properties.html

Peishi Jiang, M. E. (2017). A service-oriented architecture for coupling web service models using the Basic Model Interface (BMI). *Environmental Modelling & Software*.

Proyectosagiles.org. (s.f.). *Qué es SCRUM*. Obtenido de proyectosagiles.org: https://proyectosagiles.org/que-es-scrum/

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MÉXICO

ACADEMIA JOURNALS

CONGRESO INTERNACIONAL DE INVESTIGACIÓN DE ACADEMIA JOURNALS.COM, CELAYA 2017

OTORGAN EL PRESENTE

CERTIFICADO

A

LIC. DORIS CASTRO VELASCO
MC JOSÉ JUAN HERNÁNDEZ MORA
MC MARÍA GUADALUPE MEDINA BARRERA
MSC AGUSTÍN SÁNCHEZ ATONAL

POR SU PARTICIPACIÓN CON LA PONENCIA TITULADA

DESARROLLO DEL MÓDULO DE ADMINISTRACIÓN DE TAREAS PARA UN SISTEMA HEAD END

PUBLICADA EN EL PORTAL DE INTERNET
CELAYA.ACADEMIAJOURNALS.COM
VOLUMEN ONLINE CON ISSN 1946-5351 VOL. 9, NO. 6, 2017
E INDIZACIÓN EN FUENTE ACADÉMICA PLUS (EBSCO) Y
LIBRO DIGITAL EBOOK CON ISBN 978-1-939982-32-2, ONLINE.
LA CUAL FUE PRESENTADA EN EL
TECNOLÓGICO NACIONAL DE MÉXICO EN CELAYA
LOS DÍAS 8, 9 Y 10 DE NOVIEMBRE DE 2017, CELAYA, GUANAJUATO, MÉXICO.

DR. RAFAEL MORAS
EDITOR, ACADEMIAJOURNALS.COM
PROFESOR DE INGENIERÍA INDUSTRIAL Y ADMINISTRATIVA
ST. MARY´S UNIVERSITY, SAN ANTONIO, TX. EEUU

ING. ALEJANDRO ÁLVAREZ BÁRCENAS
COORDINADOR GENERAL DEL
CONGRESO INTERNACIONAL DE INVESTIGACIÓN
ACADEMIA JOURNALS, CELAYA 2017

N° 1296

ANAB

Cel0511

Anexo 2. Cartas

3.1 Carta de satisfacción

EOS TECH

Querétaro, Qro., a 02 de Agosto del 2017

Asunto: Constancia de satisfacción

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR DEL INSTITUTO TECNOLÓGICO DE APIZACO
PRESENTE:

AT'N: DR. JOSÉ FEDERICO CASCO VÁZQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio del presente le envío un cordial saludo y aprovecho para hacerle constar que posterior a la recepción del sub-proyecto: **"Diseño e implementación de servicio Web FTP y módulo para manejo de tareas y excepciones de un sistema *Head-End* de una red distribuida de medidores inteligentes"** realizado por la Lic. Doris Castro Velasco con número de control M00370031, y dirigido por el M.C. José Juan Hernández Mora, alumna y profesor respectivos de la Maestría en Sistemas Computacionales, de la Institución que usted destacadamente dirige, **cubre y satisface las expectativas planteadas al inicio de la fase de este proyecto.**

Tenemos bien a dar constancia de que dicho trabajo realizado por la Lic. Doris Castro Velasco. Agradeciendo sus atenciones quedo de ustedes.

ATENTAMENTE

Ing. Juan González Salomón
Director General
EOS TECH S.A. DE C.V.

EOS TECH S.A de C. V.
Luis G. Malvaez No. 523 Col. Reforma Agraria 2da Sección.
Querétaro, Qro., México. C.P. 76086
Tel.: (442) 243 1331

www.eostech.mx
"Mejoramos tu mundo con innovación"

3.2 Carta de término de estancias

EOS TECH

Querétaro, Qro., a 02 de Agosto del 2017

Asunto: Constancia

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR DEL INSTITUTO TECNOLÓGICO DE APIZACO
PRESENTE:

AT'N: DR. JOSÉ FEDERICO CASCO VÁZQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio del presente le envío un cordial saludo y aprovecho para hacerle constar que la Lic. Doris Castro Velasco, alumna de la Maestría en Sistemas Computacionales con número de control M00370031, de la Institución que usted destacadamente dirige, participó en el sub-proyecto:

"Diseño e implementación de servicio Web FTP y módulo para manejo de tareas y excepciones de un sistema *Head-End* de una red distribuida de medidores inteligentes"

La etapa del proyecto realizada tuvo una duración de 6 meses y se desarrolló en las oficinas centrales de EOS TECH S.A. DE C.V. en la ciudad de Querétaro, Querétaro.

En virtud de que se han cubierto satisfactoriamente la fase del proyecto. Tenemos bien a dar constancia de que dicho trabajo realizado por la Lic. Doris Castro Velasco, cubre y satisface las expectativas planteadas al inicio de la fase de este proyecto.

Agradeciendo sus atenciones a la presente quedo de ustedes.

ATENTAMENTE

Ing. Juan González Salomón
Director General
EOS TECH S.A. DE C.V.

EOS TECH S.A de C. V.
Luis G. Malvaez No. 523 Col. Reforma Agraria 2da Sección.
Querétaro, Qro., México. C.P. 76086
Tel.: (442) 243 1331

www.eostech.mx
"Mejoramos tu mundo con innovación"