Universidad de las Ciencias Informáticas

Facultad 1

Componente para generar alertas en el Sistema *Xilema Smart Keeper 3.0*

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Henry Ramírez Bullaín

Tutores:

MSc. Graciela González Pérez

Ing. Yojahny Sánchez Marrero

La Habana, 2018

¨RECUERDA: NO ERES TORPE, NO IMPORTA LO QUE DIGAN ESOS LIBROS. LOS TORPES DE VERDAD SON GENTE QUE, CREYÉNDOSE EXPERTOS TÉCNICOS, NO PODRÍAN DISEÑAR HARDWARE Y SOFTWARE MANEJABLE POR USUARIOS NORMALES, AUNQUE LA VIDA LES FUERA EN ELLO¨.

WALTER MOSSBERG

Declaración de Autoría:

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI), los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los __ días del mes _____ del año _____.

_____

Firma del autor

_____ _____

Firma del tutor Firma del tutor

MCs. Graciela González Pérez Ing. Yojahny Chávez Marrero

Dedicatoria

A mi hermanita Geinis Ramírez Bullaín, mi mamá Iraí Bullaín Núñez y mi papá Orlando Antonio Ramírez Rodríguez por educarme, quererme, apoyarme y sobre todo por confiar en mi desde el primer día.

Agradecimientos

A MI MAMÁ MI HERMANITA Y MI PADRE POR SIEMPRE CONTAR CON SU APOYO, POR ESTAR PENDIENTES DE MÍ EN TODO MOMENTO, INCLUSO EN LOS PEORES.

A TODOS LOS AMIGOS Y PERSONAS QUE A LO LARGO DE ESTOS 5 AÑOS HE CONOCIDO Y QUE DE UNA FORMA U OTRA ME HAN AYUDADO.

A MIS TUTORES POR ESTAR PENDIENTES DE MÍ DURANTE TODO EL DESARROLLO DE LA TESIS, ESPECIALMENTE A YOJAHNY SÁNCHEZ MARRERO.

Resumen

En la Universidad de las Ciencias Informáticas se desarrolla, desde el año 2005, el filtro de contenido *web Smart Keeper*. Su principal objetivo es controlar (permitir o denegar) el acceso a contenidos inadecuados teniendo en cuenta las políticas y reglas establecidas en las instituciones donde sea instalado. Actualmente dicho sistema no cuenta con una herramienta capaz de generar alertas de carácter administrativo referente al uso o consumo de los recursos de *hardware* que lo componen, así como del estado de los servicios (activo o detenido).

El objetivo de este trabajo es elaborar un componente para generar alertas en el Sistema *Xilema Smart Keeper* que haga uso de las herramientas ya implementadas en *Smart Keeper*. Para lograr dicho objetivo se ha realizado un estudio acerca de las herramientas que generan alertas tanto en filtros de contenido como fuera de estos.

Para la elaboración de la aplicación se utilizaron los lenguajes de java y php, y se utilizó el *framework* de trabajo de *Symfony3.4* teniendo en cuenta todos los pasos y métodos que ofrece la metodología variación de AUP[1] para la UCI en el escenario número 4. La ejecución de las pruebas permitió comprobar el correcto flujo de información entre las herramientas que conforman la solución, así como la correspondencia de esta con los requerimientos del cliente. Al terminar la investigación se obtuvo como resultado un componente para generar alertas en el sistema *Xilema Smart Keeper*.

Palabras claves: Alertas, componente, filtro, hardware, Smart Keeper

[1] Agile Unified Process, en español Proceso Unificado Ágil de *Scott Ambler*

Índice

Índice de tablas

Índice de figuras

Introducción

En el mundo actual las tecnologías de la información forman parte de la vida humana y con el desarrollo alcanzado por el Internet se ha posibilitado que la gama de contenidos que se generan se encuentre en ascenso considerable. Los usuarios que componen la red de redes consumen y comparten estos contenidos algunos de carácter informativo, recreativo, científico o educativo. Sin embargo, hay otros de carácter nocivo e incluso ilegal según las leyes en algunos países.

El uso de Internet entre el ámbito laboral revela que los empleados leen noticias, hacen compras en línea y se comunican con sus amigos y familiares durante las horas de trabajo. Un estudio estadístico revela que el 35% de los empleados usa el Internet en horas de trabajo. El 20% usa el Internet por al menos 2 horas diarias durante su periodo laboral. El 15% Usa el Internet para motivos personales como lo son: compras en línea, leer y contestar correos personales, visitar páginas de entretenimiento, deportes o con contenido sexual. Es aquí cuando las alertas entran en función informando al equipo administrativo de lo que hacen sus subordinados (salixnetworks, 2015).

Es por ello que las instituciones toman medidas que permiten regular el acceso a los materiales disponibles, entre ellas se encuentra el uso de los filtros de contenidos. Estos son los encargados en gran medida de generar alertas tanto de carácter administrativo como del lado del cliente las cuales son de vital importancia porque permiten mantener informados a ambas partes y debido a la alta variedad de contenidos que circulan en la red se hace necesario estar al tanto de lo que ocurre en todo momento y las alertas se encargan de ello fundamentalmente desde el ámbito administrativo.

Otro aspecto importante a tener en cuenta es el uso de recursos de hardware donde se encuentran los servicios debido a que si falla alguno puede ocasionar el colapso total de sistemas es por ello que mantener informado al personal administrativo mediante un conjunto de alertas es de vital importancia, para estar al tanto de las principales fallas de *hardware* que puedan surgir es decir fallas de memoria, calentamiento del procesador y fallas del disco duro que son las más comunes.

La Universidad de Ciencias Informáticas (UCI) ocupa un lugar prominente en la producción de *software*, en ella se desarrolla una alta gama de proyectos que tributan al desarrollo económico y tecnológico del país, y ayudan a la adquisición de nuevos conocimientos. Dicha universidad se encuentra estructurada en varias

facultades con diversas líneas de investigación y producción, una de ellas es la facultad 1 “José Martí” en la cual se encuentra el Centro de Ideo-Informática (CIDI).El cual posee dentro de sus líneas directrices el filtro de contenido *Xilema Smart Keeper* que tiene como fin apoyar a las instituciones en el control de los materiales a los que acceden sus miembros contribuyendo así a la toma de decisiones en aras de optimizar el uso de Internet.

El sistema Xilema Smart Keeper al estar compuesto por varios componentes es complejo desde el punto de vista arquitectónico por lo que se hace difícil detectar el fallo de alguno de los mismos en tiempo real. Además, no posee componentes capaces de generar alertas de carácter administrativo, imposibilitando la toma de decisiones tempranas en la institución.

Esto provoca que en muchas ocasiones se vean afectados los servicios de red debido al uso desmedido de los recursos, lo cual podría dejar incomunicada la institución. Por otra parte, con la caída de algún servicio, dígase por ejemplo el servidor proxy Squid sería fatal debido a que es el encargado de generar los logs de las trazas de los usuarios y la perdida de los logs imposibilita chequear la actividad de los usuarios. También es vital en todo sistema contar con un mecanismo que posibilite conocer uso del CPU y la RAM, así como el espacio en disco HDD para poder prevenir catástrofes referentes a la perdida de datos debido a la falta de espacio en disco o una respuesta tardía de alguna funcionalidad critica del sistema provocada por la falta RAM para el procesamiento, entre otros. Es por ello que se hace necesaria la elaboración de un mecanismo de carácter administrativo que informe en tiempo real la ocurrencia de algún fallo referente al uso y consumo de los recursos de hardware (uso de memoria RAM, uso del microprocesador CPU y espacio en disco HDD) o problema ocurrido en el sistema (detención del servidor proxy Squid, del servidor de BD, etc.), para contribuir a la toma de decisiones de la institución.

Teniendo en cuenta esta situación se ha definido como **problema de investigación** ¿Cómo generar alertas en el Sistema Xilema Smart Keeper 3,0 para contribuir a la toma de decisiones?

El **objeto de estudio** del presente trabajo de diploma está definido en el proceso de generación de alertas con filtros de contenido.

El **campo de acción** está enmarcado en la generación de alertas mediante el filtro de contenidos *Xilema Smart Keeper.*

Estableciendo como **objetivo general** el desarrollo de un componente que permita generar alertas en el Sistema Xilema Smart Keeper 3.0 para contribuir a la toma de decisiones.

A partir del objetivo general se derivan los siguientes **Objetivos Específicos:**

1. Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con el desarrollo de herramientas para la generación de alertas en los softwares de filtrado de contenidos de Internet.
2. Diagnosticar el estado de la generación de alertas en *software* de filtrado de contenidos de Internet.
3. Diseñar las funcionalidades del componente generador de alertas para el sistema *Xilema Smart Keeper*.
4. Implementar las funcionalidades del componente generador de alertas para el sistema *Xilema Smart Keeper.*
5. Validar las funcionalidades del componente generador de alertas para el Sistema *Xilema Smart Keeper.*

Después de haber tratado los elementos fundamentales del área de la ciencia a incidir y los objetivos primordiales, se plantea como **idea a defender** que el desarrollo de un componente para generar alertas en el Sistema *Xilema Smart Keeper 3.0* contribuye a la toma de decisiones.

Para estudiar los sistemas de filtrado de contenidos y guiar la investigación se utilizaron los siguientes **métodos científicos:**

Métodos Teóricos:

El método **Histórico-Lógico** posibilitó hacer un estudio de los fundamentos teóricos de los sistemas de filtrado de contenidos fundamentalmente en Sistema *Xilema Smart Keeper 3.0* desde sus inicios hasta la actualidad en el ámbito Internacional y Nacional, así como las tendencias actuales de este tipo de sistemas. El **Analítico-Sintético** permitió extraer y analizar los conceptos y definiciones más importantes relacionadas con el objeto de estudio a partir de los cuales se obtuvo una propuesta que diera cumplimiento a la situación problemática.

La **modelación** se emplea a través de los diagramas del lenguaje de modelado UML[2], se utilizará en la representación de las características y las relaciones entre los objetos de la solución.

Métodos Empíricos:

Mediante la **entrevista** al cliente se pudo identificar las necesidades reales y se obtuvo la información relacionada con el funcionamiento del *Sistema Xilema Smart Keeper*. Este método se utilizó durante el transcurso de la investigación para obtener información acerca del problema en cuestión para poder eliminar las no conformidades del cliente sobre el sistema desarrollado anteriormente.

A través de la **observación** se pudo conocer la realidad mediante percepción del objeto de estudio, es decir, cuando se comenzó la investigación la observación permitió saber cómo son los problemas existentes, teniendo como parte además de la misma las entrevistas.

Se define la siguiente **estructura del documento**, quedando conformado en tres capítulos:

Capítulo 1: Fundamentación Teórica. Se realiza un estudio del estado del arte sobre las reglas de negocio y algunas aplicaciones Web que notifican sus alertas mediante estas. Se describen tanto las tecnologías como metodologías y herramientas definidas por el modelo de desarrollo del Centro de Ideo Informática (CIDI) para el desarrollo de sistemas informáticos.

Capítulo 2: Propuesta de solución. Se modela el negocio, se definen los requisitos funcionales y no funcionales con que contará el componente de alertas para el Sistema *Xilema Smart Keeper*. Además del análisis y diseño donde se define la arquitectura de datos y del sistema para generar los artefactos definidos por el modelo de desarrollo, y por último se describen los patrones de diseño y arquitectónicos definidos para el desarrollo de la solución.

Capítulo 3: Implementación y validación del componente para generar alertas en el Sistema Xilema Smart Keeper 3.0. Se especifican aspectos de interés para el proceso de implementación tales como los estándares de codificación para el código fuente del sistema. Se especifican además los resultados de la validación del componente mediante pruebas de *software* realizadas y finalmente se valora que beneficios trae el resultado obtenido en dichas pruebas.

---

[2] El lenguaje unificado de modelado (*UML*, por sus siglas en inglés, *Unified Modeling Language*)

Capítulo 1: Fundamentación teórica

En este capítulo se explican los principales problemas que fundamentan la propuesta de solución, los objetivos generales que se persiguen, así como los conceptos relacionados con el tema en cuestión. También permite conocer las tecnologías, lenguajes, metodologías y herramientas utilizadas para el análisis, diseño e implementación del sistema sobre las cuales se apoya la propuesta. Además, se abundará sobre el estado del arte, específicamente en temas relacionados con el desarrollo del presente trabajo.

1.1. Marco conceptual

Para lograr una mejor comprensión de la investigación, se abordan un conjunto de conceptos necesarios que están estrechamente relacionados con el dominio de problema:

Alerta: El término alerta**,** del italiano *all'erta*, hace referencia a una situación de vigilancia o atención. Un estado o una señal de alerta es un aviso para que se extremen las precauciones o se incremente la vigilancia(Julián y Ana 2014).

Componente: Un componente es una parte encapsulada de un sistema; idealmente, una parte sustituible, casi independiente y segura de un sistema no trivial que desempeñe una función clara en el contexto de una arquitectura bien definida. Es un elemento el cual, al estar en contigüidad, es decir, unidos con otros permite la formación de un conjunto completo totalmente uniforme. Dicho de otra forma un componente es una unidad que compone o forma parte de una estructura completa, y a su vez cumple una función importante en la misma.

En la industria del software y la literatura al respecto se utiliza el término "componente" para referirse a muchas cosas diferentes. A menudo se utiliza en un sentido amplio para hacer referencia a una "parte constitutiva". También suele utilizarse en un sentido más restringido para denotar características específicas que permiten la sustitución y el ensamblaje en sistemas más grandes(Definición, 2016).

Sistema: Del latín *systema*, un sistema es módulo ordenado de elementos que se encuentran interrelacionados y que interactúan entre sí. El concepto se utiliza tanto para definir a un conjunto de conceptos como a objetos reales dotados de organización(Pérez Porto, 2008).

Sistema informático: Un sistema informático es un conjunto de partes o recursos formados por el hardware y software y las personas que lo emplean, que se relacionan entre sí para almacenar y procesar información con un objetivo en común. El recurso físico (o de hardware) está compuesto por computadoras, impresoras, escáneres, memorias, lectores de código de barras, estructura física de una red de computadoras, etc. El lógico (o de software) por manuales de uso, sistema operativo, archivos, documentos, aplicaciones, firmware, base de datos, información de una red de computadoras, etc. Y el recurso humano son todos las personas que forman parte del sistema, como son los operadores del sistema, los técnicos que lo mantienen y los usuarios finales (Alegsa, 2015).

Alertas tempranas: Instancia primaria, que implica la vigilancia permanente de las distintas áreas y escenarios de riesgos. Provisión de información oportuna y eficaz a través de instituciones identificadas, que permiten a individuos expuestos a una amenaza, la toma de acciones para evitar o reducir su riesgo y su preparación para una respuesta efectiva.

Registros *logs*: Un log es un registro oficial de eventos durante un rango de tiempo en particular. En informática, principalmente en seguridad informática, es utilizado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación (*Schneier y Kelsey* 2015).

Propósito de los *logs:* Si un log tiene la capacidad de almacenar los eventos que ocurren en cierto dispositivo o aplicación, entonces el objetivo de un log es proveer a los profesionales de la seguridad informática la habilidad de monitorear las actividades realizadas por quien haya generado el fichero. Revisando los log, se puede prevenir ataques informáticos y tomar las decisiones necesarias para corregir problemas de funcionamiento (*Schneier y Kelsey* ,2015).

1.2. Estudio de sistemas homólogos

En la actualidad existen diferentes filtros de contenido que se utilizan para controlar el acceso a la red y generar alertas para el apoyo a la toma de decisiones en las empresas e instituciones. Estos en el mercado se categorizan, a partir de si se publican como código abiertos o son aplicaciones privativas. En busca de elementos y funcionalidades que pudieran ayudar a resolver el problema planteado se realizó un análisis de los filtros de contenido más conocidos y utilizados:

EventLog Analyzer:

Permite centralizar los *logs* (visor de sucesos) de sus servidores, aplicaciones y dispositivos de red, otorgando visibilidad al histórico de sucesos a través de una sencilla e intuitiva consola web. La centralización de *logs* hace que la monitorización de sistemas críticos sea mucho más sencilla. Se pueden configurar notificaciones y alertas para informar sobre sucesos concretos y los informes se pueden programar para enviar por correo electrónico(IREO ,2018).

EventLog Analyzer optimiza sus recursos y facilita las tareas de monitorización, análisis y diagnóstico de sus sistemas críticos. Es una solución SIEM (*Security Information and Event Management*) que centraliza la información sobre los eventos que ocurren en sus dispositivos, sistemas y aplicaciones. Ayuda a obtener visibilidad sobre los eventos críticos y aporta herramientas para poder analizar y entender lo que sucede en su red. Posee varias funcionalidades entre las que destacan las siguientes:

- Recoge y almacena los datos del Visor de Sucesos de Windows, así como *SysLog* y otros *logs* de sus sistemas y aplicaciones.
- Los *logs* históricos también se pueden importar fácilmente, para mantener un repositorio centralizado de información sobre los eventos.
- Alertas y notificaciones personalizadas.
- Alertas por mensajes cortos (SMS).
- Informes estándar sobre la monitorización de eventos. Los informes pueden ser personalizados.
- Análisis del histórico y detección automática de tendencias.
- Se pueden recolectar los datos localmente en cada red y enviar la información al servidor central.
- Muy escalable (permite monitorizar hasta 12.000 sistemas).
- Idóneo para proveedores de servicio (MSP[3]).

[3] Proveedor de Servicios Gestionados (MSP, por sus siglas en inglés "*Manage Service Provider*)

Una vez vistas las características más destacadas del sistema en cuestión se decide no utilizarlo en la propuesta de solución debido a que se centra en sistemas privados, dígase Windows, no obstante, se tendrá en cuenta para la elaboración de las alertas.

Firewall Analyzer

Genera alertas automáticamente y notifica a los administradores de red, cuando hay un intento de brecha de seguridad en la red o tráfico anormal. Detecta el comportamiento anómalo del tráfico. Las alertas se pueden configurar para notificar a los administradores/operadores por correo electrónico. Las alertas se pueden notificar mediante SMS y se puede ejecutar un programa para iniciar otras acciones (por ejemplo: generar una captura SNMP[4] para ser dirigida a la aplicación de administración de activos/red). *Firewall Analyzer* también le facilita la gestión de los dispositivos proporcionando informes predefinidos y alertas sobre cambios en la configuración compatible con prácticamente todos los fabricantes, incluyendo los *firewalls* de código abierto, además de los dispositivos comerciales (*Squid Project, Check Point, Cisco, Juniper, Fortinet, Snort*), IDS/IPS, las VPN[5], *Proxies* y dispositivos de seguridad relacionados.

Según el sitio web manageengine.com en el cual 120,000 organizaciones y 190 países confían para administrar y garantizar una alta disponibilidad en sus negocios. El análisis de registro de proxy de *Squid* proporciona una visión más profunda de los patrones de acceso de los sitios web de los usuarios de la red interna. *Firewall Analyzer* procesa los registros del servidor proxy de *Squid* para generar informes exclusivos de *Squid y Firewall Analyzer* es totalmente compatible con (*Squid*) proxy, combinación de *firewall* y como dos entidades separadas. La aplicación proporciona informes de proxy exhaustivos (es decir, informes de *Squid*)(*ManageEngine*, 2017). Una vez analizado este sistema se decide implementar algunas de las funcionalidades que posee en la propuesta de solución del presente trabajo de diploma por ejemplo (la notificación a administradores/operadores por correo electrónico), descartando las alertas mediante SMS debido al costo monetario que implican.

---

[4] Protocolo simple de administración de red o *SNMP* (del inglés *Simple Network Management Protocol*)

[5] Red Privada Virtual (VPN, por sus siglas en inglés Virtual Private Network)

Event detected

Es la plataforma remota que le permite la monitorización y gestión avanzada de alertas, vía web, de todos los recursos de su estación de servicio o centro de distribución. Permite registrar eventos, mediciones y pruebas de vida de dispositivos físicos, procesos y aplicaciones externas, garantizando el funcionamiento correcto de los mismos y la continuidad de su uso. *Event Detect* posibilita una toma de decisiones inmediata ante cualquier incidencia que se produzca, registrándola e informando de ella en la forma en que haya sido definida por el usuario.

Event Detect permite crear configuraciones de alertas totalmente personalizadas**,** asociándolas a mediciones, contadores y pruebas de vida o, sencillamente**,** aplicar políticas estándar que podrá cargar rápidamente en la aplicación. Registra toda la actividad recibida desde los dispositivos. Permite acceder al historial de alertas y consultar las acciones realizadas. Dispone de múltiples opciones de configuración para el envío de alertas y notificaciones a destinatarios o grupos, pudiendo definir, para cada tipo de alerta, a qué persona se le enviará el mensaje: al encargado de la estación, al proveedor de consumibles, al servicio técnico, a varios. La plataforma puede enviar notificaciones vía SMS, correo electrónico, FTP[6], *web-service* o *Twitter*, entre otros y cuenta con una gran cantidad de funcionalidades entre las que destacan(avaloninformatica ,2017):

- ✓ Reglas de activación de alertas, basadas en mediciones, contadores y pruebas de vida.
- ✓ Políticas de notificación, según tipo de alerta
- ✓ Posibilidad de especificar causa al crear alertas.
- ✓ Estadísticas de alertas a nivel de dispositivo y área.
- ✓ Opción de elegir política de detección de alarmas por defecto.
- ✓ "Muro" con visualización personalizada de las alertas.

[6] File Transfer Protocol en inglés o Protocolo de Transferencia de Ficheros en español

Este sistema de forma general al igual que los anteriormente analizados ofrece funcionalidades que se tendrán en cuenta en la propuesta de solución dígase especificar la causa y el tipo al crear la alerta. Y se decide no utilizarlo debido a que no permite generar alertas referentes al hardware del sitema.

Log Analytics

Las alertas de *Log Analytics* identifican información importante en el repositorio de *Log Analytics*. Se crean mediante reglas de alerta que ejecutan automáticamente búsquedas de registro en intervalos regulares, y si los resultados de la búsqueda de registro coinciden con criterios determinados, se crea un registro de alerta y se puede configurar para realizar una respuesta automatizada.

Las alertas se crean mediante reglas de alerta que ejecutan automáticamente búsquedas de registros a intervalos regulares. Puede crear alertas basadas en métricas de rendimiento específicas o cuando se creen determinados eventos, haya un evento ausente o se cree una serie de eventos dentro de una ventana de tiempo determinada. Por ejemplo, se pueden usar alertas para avisar cuando el uso medio de la CPU supere un determinado umbral o generar un evento cuando un servicio de Windows específico o Linux *Daemon* no funcionen. Si los resultados de la búsqueda de registros coinciden con determinados criterios, se crea un registro de alertas. Luego, la regla puede ejecutar automáticamente una o varias acciones para avisarle proactivamente de la alerta o invocar otro proceso. Puede realizar acciones avanzadas con alertas, como crear una notificación por correo electrónico, Las acciones de correo electrónico envían un correo electrónico con los detalles de la alerta a uno o varios destinatarios. Puede especificar el asunto del mensaje de correo, pero su contenido es un formato estándar construido por *Log Analytics (MGoedtel, 2016)*.

Una vez vistas las principales características y funcionalidades del sistema analizado, se decide tomar en cuenta una de ellas y a partir de la misma, la elaboración de otras, el autor se refiere a las alertas cuando el uso del CPU supere determinado umbral y las derivadas son las referentes a al uso de la RAM y el espacio en disco duro(HDD). A pesar de que este sistema posee gran parte de las funcionalidades que se proponen para la propuesta de solución, se descarta debido a la incompatibilidad con el servidor Proxy Squid.

IBM Tealeaf

Tiene como objetivo, a través de la grabación de la sesión del usuario (capturando el DOM[7] de la página), y con el posterior análisis, encontrar errores, limitaciones y, en definitiva, puntos de acción para mejorar tanto la experiencia de usuario como los KPIs[8] básicos de negocio. *IBM Tealeaf* se basa en la captura de todos los *hits* que se producen a lo largo de la sesión del usuario, por tanto, en inicio, se dispone de una gran cantidad de información sobre la cual es necesario actuar de algún modo para poder sacarle partido. Esa palanca, son los eventos, base de cualquier informe que se quiera consultar en *IBM Tealeaf*.

Los eventos monitorizan cada uno de los activos en cuestión mediante la búsqueda de determinadas condiciones, y se graban cuando esa condición se cumple pudiendo, además, almacenar información.

Son la base para:

- Identificar comportamientos: una vez establecidos, se podrá ver cuántas veces o bajo qué circunstancias se produce determinada acción.
- Configurar alertas: se podrán programar alertas en función de si determinado evento ocurre un número de veces por debajo o por encima de determinado umbral.
- Reproducción de sesiones mediante la herramienta de búsqueda**:** a través de la consola de búsqueda es posible filtrar aquellas sesiones que se ajustan a determinados patrones, fijados mediante segmentos, y visualizar las grabaciones específicas que sigan ese patrón. Un claro ejemplo, visualizar qué pasa en las sesiones en las que el evento “clic en pagar con tarjeta de crédito” se ha grabado más de 3 veces.

[7] *Document Object Model* o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos')

[8] *Key Performance Indicator* (indicador clave de desempeño o indicadores de gestion)

- Configuración de informes: es posible ver la información asociada a cualquier evento mediante la interfaz de *reporting*, donde se pueden construir *funnels* y *dashboards* combinando eventos y dimensiones(Ferreira, 2017).

Este sistema al igual que el anterior posee funcionalidades interesantes que se pueden implementar en la propuesta de solución para darle la libertad al administrador de configurar las alertas, así como la configuración de los informes. Y no se utiliza en la solución por ser de carácter privativo.

1.3. Selección del entorno de trabajo para la construcción de la solución.

Para el desarrollo de la propuesta de solución se hace necesario la definición de las herramientas, tecnologías y metodología que se emplearán en correspondencia con las usadas en el desarrollo del componente de alertas para el sistema *Xilema Smart Keeper*.

1.3.1. Metodología de desarrollo de software

El sitio web *OkHosting.com* con sede en Guadalajara Jalisco, México que cuenta con más de 5000 clientes satisfechos plantea que una Metodología de desarrollo de *software*, permite principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo**.** Regularmente este tipo de metodología, tienen la necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto, comprendan perfectamente la metodología y en algunos casos el ciclo de vida del software que se pretende seguir.

Existen numerosas metodologías a la hora de desarrollar un proyecto y para la implementación del componente de alertas para el sistema *Xilema Smart Keeper 3.0* se decidió emplear la siguiente metodología(*Marketing Online*, 2016):

Variación de AUP para la UCI:

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés es una versión simplificada del Proceso Unificado de Racional por sus siglas en inglés (RUP). Este describe de una manera

simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva
1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

De las 4 fases que propone AUP se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que se llama ejecución y se agrega la fase de Cierre quedando de la siguiente forma

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Como se puede apreciar en la descripción anterior la variación de la metodología AUP para la UCI logra estandarizar la actividad productiva del desarrollo de software en la universidad dando cumplimiento

además a las buenas practicas que define CMMI-DEV v1.3, se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y producto de trabajo. Teniendo en cuenta estos aspectos se considera que la metodología Variación AUP-UCI es idónea para el desarrollo del componente de alertas para el sistema *Xilema Smart Keeper 3.0* (Rodríguez Sánchez ,2014).

1.3.2. Lenguajes

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Estos se dividen en dos grupos principales en base al procesamiento de sus comandos: lenguajes imperativos y lenguajes funcionales(*IsraelCCM* ,2017).

PHP 7.0

PHP es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Código abierto significa que es de uso libre y gratuito para todos los programadores que quieran usarlo. Incrustado en HTML significa que en un mismo archivo se va a poder combinar código PHP con código HTML, siguiendo unas reglas. PHP se utiliza para generar páginas web dinámicas. Recordar que se llama página estática a aquella cuyos contenidos permanecen siempre igual y dinámicas a aquellas cuyo contenido no es el mismo siempre. PHP también puede utilizar y presentar resultados en otros estándares de datos o lenguajes propios de los desarrollos web, como XHTML y cualquier otro tipo de ficheros XML. Puede autogenerar éstos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla, utilizando estos ficheros para generar contenido dinámico. Es decir, el contenido dinámico puede surgir de otros sitios además de desde bases de datos. Además, se puede interactuar con otros servidores usando cualquier protocolo.

Este lenguaje interpretado del lado del servidor que surge dentro de la corriente denominada código abierto (*open source*). El mismo se caracteriza por su potencia, versatilidad, robustez y modularidad. Al igual que ocurre con tecnologías similares, los programas son integrados directamente dentro del código HTML. Comparado con ASP, la principal ventaja de PHP es su carácter multiplataforma. Por otro lado, los programas en ASP resultan más lentos y pesados, y también menos estables. Comparando el lenguaje PHP con el lenguaje Perl, utilizado habitualmente en la programación CGI, puede decirse que PHP fue diseñado

para desarrollo de scripts orientados a web, mientras que Perl fue diseñado para hacer muchas más cosas y debido a esto, se hace muy complicado. La sintaxis de PHP es menos confusa y más estricta, pero sin perder la flexibilidad.

En definitiva, PHP es uno de los lenguajes más utilizados actualmente en el desarrollo de aplicaciones web y viene experimentado un constante crecimiento en su nivel de utilización en Internet., PHP puede enlazarse con otros lenguajes muy potentes como Java demostrando potencia y viabilidad. (GOMEZ ,2005).

HTML5

HTML5 (*HyperText Markup Language*) es la quinta revisión del lenguaje de marcado estándar que se emplea para la web. Es uno de los lenguajes de marcado más usados en todo el mundo y la razón es bastante obvia: gracias a HTML5 se puede crear la estructura de una página web. Texto, imágenes y material multimedia pueden mostrarse correctamente gracias a HTML5, este ofrece a los programadores de sitios web nuevas oportunidades. Así, navegadores web tan populares como Internet Explorer, Mozilla Firefox o Google Chrome ya son compatibles con HTML5. Además, este estándar también funciona correctamente con Smartphone y *tablets* De este modo, también es posible mejorar la velocidad y visualización de las páginas webs en dispositivos móviles. Por medio de las etiquetas video y audio de HTML5, ahora se puede añadir videos o audio sin necesidad de usar otro *plugin* de tercero. Toda la acción sucede desde el propio navegador, lo que puede ayudar a disminuir al tamaño del archivo final de tu página. En HTML5 se puede crear animaciones en 2D gracias a la etiqueta *canvas*. La API para esta etiqueta te permite dibujar elementos en 2D y animarlos. Pero no es todo, la API te permite añadir eventos de teclado, ratón y cualquier otro mando que desees incluir. Además de estas características HTML5 también es gratuito no se necesita ningún tipo de programa especial para empezar a programar, incluso se puede hacer en un bloc de notas, guardar el documento como HTML y se podrá visualizar desde cualquier navegador(Aula Formativa, 2016).

JavaScript

JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con *JavaScript* se puede crear diferentes efectos e interactuar con nuestros usuarios.

Este lenguaje posee varias características, entre ellas se puede mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, es un lenguaje que utiliza *Windows*, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros. *JavaScript* es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros(Perez Valdez, 2007).

jQuery

Es una librería *JavaScript open-source*, que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación "scripting" mucho más fácil y rápida del lado del cliente. Con *jQuery* se pueden producir páginas dinámicas, así como animaciones parecidas a Flash en relativamente corto tiempo. Fue publicado por primera vez en enero del 2006 en "*BarCamp* NYC" por John Resign. Soporte para AJAX fue agregado un mes después, y el modelo de licenciamiento *open source* del MIT fue adoptado en mayo de ese mismo año.

La ventaja principal de *jQuery* es que es mucho más fácil que sus competidores. Se puede agregar *plugins* fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo. De hecho, una de las principales razones por la cual Resig y su equipo crearon *jQuery* fue para ganar tiempo (en el mundo de desarrollo web, tiempo importa mucho). La licencia *open source* de *jQuery* permite que la librería siempre cuente con soporte constante y rápido, publicándose actualizaciones de manera constante. La comunidad *jQuery* es activa y sumamente trabajadora. Otra ventaja de *jQuery* sobre sus competidores como Flash y puro CSS es su excelente integración con AJAX.(jQuery, 2013).

Java

Es una tecnología que se usa para el desarrollo de aplicaciones que convierten a la Web en un elemento más interesante y útil. Java no es lo mismo que *javascript*, que se trata de una tecnología sencilla que se usa para crear páginas web y solamente se ejecuta en el explorador. Java le permite jugar, cargar fotografías, chatear en línea, realizar visitas virtuales y utilizar servicios como, por ejemplo, cursos en línea, servicios bancarios en línea y mapas interactivos. Si no dispone de Java, muchas aplicaciones y sitios web no funcionarán(Java ,2015).

1.3.3. Herramientas

A continuación, se describen las herramientas necesarias para la implementación de la propuesta de solución

Herramienta *CASE Visual Paradigm 8.0*

La herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. En el presente trabajo se decidió utilizar *Visual Paradigm* que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML, presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones(Hernández ,2016)**.**

Gestor de Base de Datos

***MySQL 5.5.59*:** Es un sistema de gestión de base de datos relacional o SGBD. Este gestor de base de datos en multihilo y multiusuario, lo que le permite ser utilizado por varias personas al mismo tiempo, e incluso, realizar varias consultas a la vez, lo que lo hace sumamente versátil.

Nació como una iniciativa de *Software* Libre y aún sigue ofreciéndose como tal, para usuarios particulares. Pero si se desea utilizarlo para promover datos en una empresa, se puede comprar una licencia, como un *software* propietario, que es autoría de la empresa patrocinante (Actualmente Oracle *Corporation*).

La mayor parte del código se encuentra escrito en lenguaje C/C++ y la sintaxis de su uso es bastante simple, lo que permite crear bases de datos simples o complejas con mucha facilidad. Además, es compatible con

múltiples plataformas informáticas y ofrece una infinidad de aplicaciones que permiten acceder rápidamente a las sentencias del gestor de base de datos.

Como se plantea anteriormente este gestor de base de datos es muy utilizado en desarrollo web, ya que permite a los desarrolladores y diseñadores, realizar cambios en sus sitios de manera simple, con tan sólo cambiar un archivo, evitando tener que modificar todo el código web. Esto se debe a que *MySQL*, trabaja con un sistema centralizado de gestión de datos, que permite realizar cambios en un solo archivo y que se ejecuta en toda la estructura de datos que se comparte en la red. Además, permite incluir noticias e información rápidamente en un sitio web, utilizando un simple formulario, sin tener que tocar el código del sitio web.

Cuando se combina con PHP, se convierte en una mezcla poderosa, que siempre es tomada en cuenta para realizar aplicaciones cliente/servidor, que requieran el uso de una base de datos rápida, segura y potente. *MySQL*, también ofrece la posibilidad de realizar programas o aplicaciones que requieran acceso a plataformas de base de datos rápidas (culturacion ,2014).

***Framework* de desarrollo:**

Symfony 3.4

Es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. *Symfony* está desarrollado completamente con PHP. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. *Symfony* es compatible con la mayoría de gestores de bases de datos, como *MySQL*, *PostgreSQL*, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación, se muestran algunas de sus características:

1. Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
2. Independiente del sistema gestor de bases de datos.
3. Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
4. Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
5. Sigue la mayoría de *mejores prácticas* y patrones de diseño para la web.
6. Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
7. Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
8. Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como, por ejemplo:

1. La capa de internacionalización que incluye *Symfony* permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
2. La capa de presentación utiliza plantillas y *layouts* que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los *helpers* incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
3. Los formularios incluyen validación automatizada y relleno automático de datos(*"repopulation"*), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
4. Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
5. La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
6. La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.

7. El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
8. El soporte de e-mail incluido y la gestión de APIs permiten a las aplicaciones web interactuar más allá de los navegadores.
9. Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.

Se decide utilizar este *framework* de trabajo para la realización de la aplicación por las características y ventajas que ofrece las cuales quedaron explícitas anteriormente. También para lograr compatibilidad con la aplicación a desarrollar puesto que la misma fue desarrolla en versiones de *Symfony* poco utilizadas en la actualidad y se hace necesario actualizarlo debido al constante cambio y desarrollo de las tecnologías de la información.

IDE de desarrollo

Netbeans 8.2

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. *Sun MicroSystems* fundó el proyecto de código abierto *NetBeans* en junio 2000 y continúa siendo el patrocinador principal de los proyectos. *NetBeans IDE* es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el *NetBeans IDE*, es un producto libre y gratuito sin restricciones de uso.(*netbeans*, 2017).

Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: **Java**, **PHP**, **Groovy**, **C/C++**, **HTML5**, **Maven**... Además, puede instalarse en varios sistemas operativos: *Window*s, Linux, Mac OS, ...y tiene como características principales las siguientes:

Buen **editor de código**, **multilenguaje**, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones

sintácticas y semánticas, plantillas de código, *coding tips*, herramientas de refactorización, y un largo etcétera.

Simplifica la **gestión de grandes proyectos** con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario.

Herramientas para **depurado de errores**: el *debugger* que incluye el IDE es bastante útil para encontrar dónde fallan las cosas. Se puede definir puntos de ruptura en la línea de código que sea de interés, monitorizar en tiempo real los valores de propiedades y variables, se permite ir paso a paso, ejecutar un método de un tirón, o entrar dentro.

Optimización de código: por su parte el ***Profiler*** ayuda a optimizar las aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria. Lo importante es que se puede ver el comportamiento de la aplicación y obtener indicadores e información de cómo y cuantos recursos consume, cuantos objetos se crean, también se pueden obtener capturas del estado del sistema en diferentes momentos (*Snapshots*) y compararlos entre sí.

Acceso a base de datos: desde el propio *Netbeans* se puede conectar a distintos sistemas gestores de bases de datos, como pueden ser *Oracle*, *MySql* y demás, y ver las tablas, realizar consultas y modificaciones, y todo ello integrado en el propio IDE(calendamaia, 2014).

Herramientas de prueba

JMeter 2.12

Es una herramienta de *testing* creada por Apache. Está completamente escrita en *JAVA*. *Jmeter* se suele usar para hacer pruebas de carga, aunque también soporta aserciones para asegurar que los datos recibidos son correctos y muchas posibilidades a la hora de generar reportes. Mediante un fichero (. jmx) previamente generado que realiza peticiones contra la aplicación objetivo a testear (dispone de la posibilidad de generar muchos tipos de peticiones: HTTP, FTP, LDAP, …).

Además, en el fichero (. jmx) se puede especificar qué número de usuarios de cada tipo ejecuta las peticiones contra la aplicación simulando uno o más grupos de usuarios trabajando contra la aplicación objetivo. Los datos generados por la herramienta para cada petición se pueden canalizar a través de un tipo de componente que proporciona la interfaz GUI[9] llamados *listeners (Toledo ,2016)***.**

Dicha herramienta cuenta con numerosas ventajas y algunas inconvenientes entre las que destacan las siguientes:

Ventajas:

1. Su uso es bastante sencillo y asequible.
2. Aparte tiene gran cantidad de documentación y tutoriales disponibles.
3. Es una herramienta que sirve para realizar pruebas funcionales, pero también sirve para realizar pruebas de regresión en aplicaciones web.
4. *JMeter* también permite la ejecución de pruebas distribuidas entre distintos ordenadores, que actuarán como clientes.

Inconvenientes:

1. *JMeter* NO se comporta como un navegador.
2. Esto implica por ejemplo que por defecto no guarde ni envíe *cookies*, no interpreta código *JavaScript*.
3. La interfaz GUI y el reporte de resultados no es muy agradable.
4. Es poco práctico a la hora de realizar acciones como copiar componentes.

1.4. Conclusiones del capítulo

Partiendo de los resultados de la investigación antes expuesta se llega a las siguientes conclusiones:

- Mediante las herramientas, tecnología y metodología se obtiene una base tecnológica adecuada que permitirá el desarrollo de la solución cumpliendo con las características de la misma.

[9] *Interfaz* gráfica de usuario, conocida también como *GUI* (del inglés graphical user *interface*)

- El análisis de varios conceptos relacionados con el campo de acción permitió y abrió el camino para una mejor comprensión y un mayor entendimiento de la investigación.
- Mediante el análisis de los sistemas homólogos se pudo obtener determinar que funcionalidades se pueden implementar en la propuesta de solución desechando las menos relevantes.
- Se utiliza como guía para el proceso de desarrollo del software la metodología AUP-UCI. Para la implementación del módulo se emplea el marco de trabajo *Symfony* bajo el lenguaje de programación *PHP*. El modelado de diagramas se realizó con la herramienta *Visual Paradigm* empleando el lenguaje *UML*.

Capítulo 2: Propuesta de solución

En el siguiente capítulo se aborda la descripción de los pasos a seguir durante el análisis y el diseño de la solución. Contiene la especificación de los requisitos funcionales y los no funcionales que debe cumplir el sistema, con sus pertinentes descripciones, se realiza la modelación del sistema, así como el diseño que poseerá el mismo. Constituye un plano bastante cercano a la implementación, y contribuirá a una arquitectura estable y sólida.

2.1. Descripción de la propuesta de solución

Para dar solución a los objetivos expuestos se propone el desarrollo de un componente para generar alertas en el Sistema *Xilema Smart Keeper*, utilizando el marco de trabajo *Symfony3.4*, como gestor de base de datos *MySQL* y el servidor web apache2, esta aplicación será capaz de integrarse al Sistema Xilema Smart Keeper. El usuario(administrador) podrá acceder al componente solamente desde el servidor donde esté desplegada la aplicación, en esta él podrá configurar los umbrales para la generación de las alertas (estas son referentes al uso del CPU, RAM y el espacio en disco (HDD)) y tendrá acceso al listado de las alertas generadas.

2.2. Requerimientos del sistema

La determinación de requerimientos es la etapa más importante en el desarrollo de un sistema de información. Comienza después de que el Cliente ha detectado una ausencia, falla o falta de oportunidad de la información o simplemente, luego de que la organización ha determinado un cambio en sus políticas, reglas o tecnologías a aplicar. En esta etapa, debe responder a la pregunta fundamental: ¿Qué es lo que quiere el Cliente? y para ello, se debe diagnosticar la situación actual, recopilar los requerimientos del cliente, tanto en relación al sistema, como generales respecto del área Informática, es decir la situación ideal, para así poder definir alternativas de solución, según las cuales se podrá avanzar desde lo que hoy se posee, hacia el objetivo que se quiere alcanzar(«análisis y diseño», 2013).

2.2.1. Requisitos funcionales

La guía del *Business Analysis Body of Knowledge(BABOK)* en su versión 3, proporciona la siguiente definición para los requerimientos funcionales de una solución:

“Los requerimientos funcionales son las descripciones explicitas del comportamiento que debe tener una solución de software y que información debe manejar.” Por lo tanto, los requerimientos funcionales:

Expresan las capacidades o cualidades que debe tener la solución para satisfacer los requerimientos de los interesados de proyecto.

Se expresan en términos de cuál debe ser el comportamiento de la solución y que información debe manejar.

Deben proporcionar una descripción lo suficientemente detallada para permitir el desarrollo e implementación de la solución.

Son los que más influyen en si la solución será aceptada o no por los usuarios(pmoinformatica.com ,2018).

A continuación, se listan los requisitos funcionales de la propuesta de solución:

Tabla 1 Requisitos funcionales del módulo

No	Nombre	Descripción	Prioridad	Complejidad
RF1	Configurar alertas	El administrador configura las alertas del sistema de acuerdo a necesidades y características específicas.	Baja	Media
RF2	Enviar correo electrónico	El sistema informa mediante correo electrónico	Alta	Alta

RF3	Generar alertas basadas en umbral relacionadas con CPU	El sistema genera alertas teniendo en cuenta el uso del CPU.	Alta	Alta
RF4	Generar alertas basadas en umbral relacionadas con HDD	El sistema genera alertas teniendo en cuenta el espacio del HDD.	Alta	Alta
RF5	Generar alertas basadas en umbral relacionadas con RAM	El sistema genera alertas teniendo en cuenta el uso de la RAM.	Alta	Alta
RF6	Listar alertas enviadas	Se mostrará un listado con las alertas que genera y envía el sistema	Alta	Alta
RF7	Comprobar servicios	Se emitirán alertas cuando alguno de los servicios(*elasticsearsh*, *logstash*, squid) no se encuentre en ejecución	Alta	Alta

2.2.2. Requisitos no funcionales

Un requisito no funcional (RnF) especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema, en lugar de un comportamiento específico. En general, los requisitos funcionales definen lo que el sistema debería de hacer, mientras que los requisitos no funcionales verifican cómo un sistema debería de ser. Requisitos no funcionales son a menudo llamados las “cualidades de un sistema”. Pueden dividirse en dos categorías:

1. ***Execution qualities***: como por ejemplo la seguridad y facilidad de uso, que son observables en tiempo de ejecución.
2. ***Evolution qualities***, como por ejemplo la mantenibilidad, extensibilidad y escalabilidad, que están más vinculados a la estructura del sistema de software(Tello, 2009).

A continuación, se listan los requisitos no funcionales establecidos para el desarrollo de la propuesta de solución

- **Usabilidad:**

  RnF1: Debe ser intuitivo para permitir a los administradores de red de poca experiencia hacer uso del componente.

- **Hardware:**

  RnF2: El ordenador (servidor) donde se ejecute el componente debe tener una memoria *RAM* de 4GB o superior y un procesador a una velocidad 2.15 GHz o superior.

  RnF3: Los servidores para el almacenamiento de la base de datos deben contar con un disco duro de más de 250 GB.

- **Software**

  RnF4: El servidor donde se instalará el componente debe tener una distribución *GNU/Linux Ubuntu mayor a 14.04.*

  RnF5: El ordenador donde se instale el componente debe contar con un servidor proxy(*squid3*).

  RnF6: Se requiere de un servidor de base de datos para el almacenamiento de los datos introducidos (*MySQL 5.5.59*).

  RnF7: El ordenador donde se instale el componente debe contar con un servidor web(apache2)

  RnF8: El servidor donde se instalará el componente debe tener instalado JRE (Java Runtime Environment) antiguamente conocida como Java Virtual Machine (JVM).

2.3. Historias de usuario

La metodología AUP-UCI, en su escenario 4 para la disciplina Requisitos, genera como artefacto a las Historias de Usuario (HU) (Sánchez, 2015), estas se utilizan para especificar los requisitos de software en las aplicaciones. Las HU se descomponen en tareas de programación y describen las características que el sistema debe cumplir, están escritas en un formato legible por el cliente, sin necesidad de sintaxis técnicas (Ordoñez y otros, 2015).

A continuación, se muestran dos (2) HU de las nueve (9) que fueron generadas, pertenecientes a los RF 1 y RF7 respectivamente, el resto puede ser consultada en el Anexo 1 de la presente investigación.

Tabla 2 HU_1 Configurar alertas

Historia de Usuario	
Número: HU_1	**Nombre:** Configurar alertas
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 2 horas	**Tiempo real:** 2 horas
Descripción: Los administradores pueden configurar a través de un formulario los distintos tipos de alertas .	
Observaciones: El administrador Selecciona los umbrales(valor a tener en cuenta para generar la alerta) según estime ,en caso de no realizar ninguna acción el sistema de forma automática toma como valor de los umbrales 60% por defecto	
Prototipo de interfaz:	

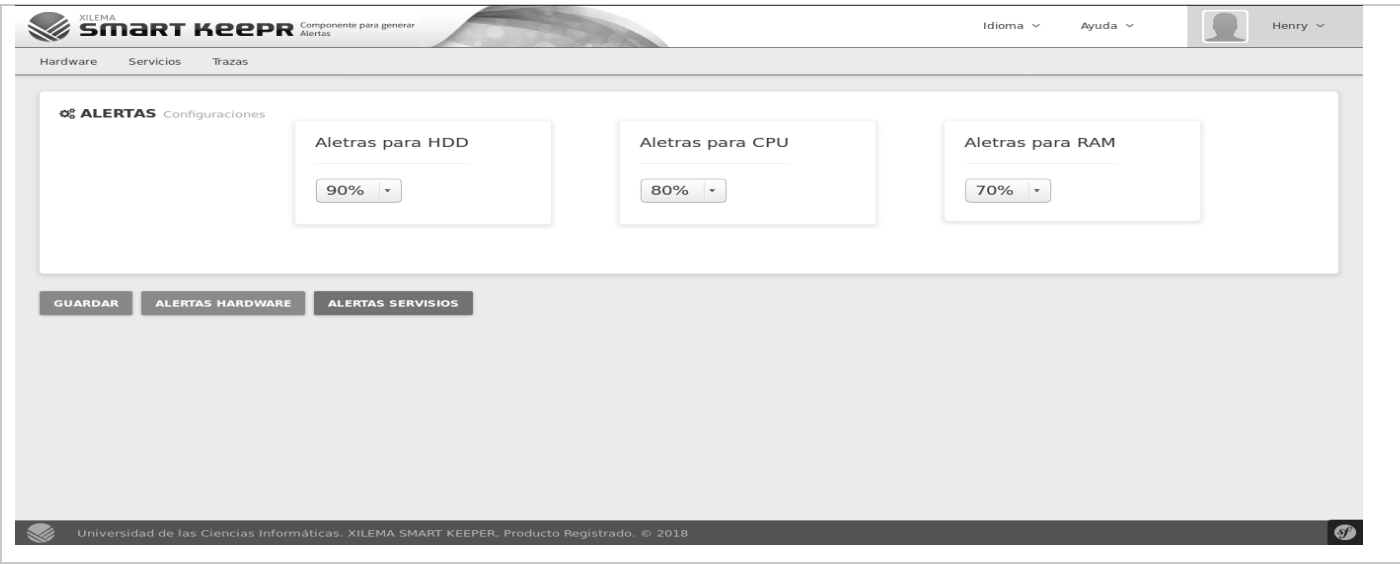

Tabla 3 HU_7 Comprobar servicios

Historia de Usuario	
Número: HU_7	**Nombre:** Comprobar servicios
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 1 horas	**Tiempo real:** 1horas
Descripción: El sistema verifica que todos los servicios se encuentren en ejecución .	
Observaciones: Los servicios a tener en cuenta son: • *Elasticsearsh* • *Logstash* • Squid	
Prototipo de interfaz: **No aplica**	

2.5. Arquitectura de la propuesta de solución

Para el desarrollo de la solución propuesta se decide utilizar el estilo arquitectónico Modelo-Vista-Controlador, en lo adelante MVC, dado que el marco de trabajo *Symfony3.4* basa su funcionamiento y estructura en el mismo. Este estilo arquitectónico se ha hecho famoso en el mundo de la programación web, y consiste en dividir la construcción de un sistema en 3 capas, que reciben los nombres de Modelo, Vista y Controlador**.**

En MVC la aplicación se va a separar en 3 capas, la primera es el modelo, que es la capa que maneja las reglas del negocio, y se encarga de cuidar los datos. Cuando se habla de aplicaciones web el modelo es el que se encarga de conectarse con la base de datos y mantener la integridad de ellos. La segunda capa es la vista, esto es lo que ve el usuario, por ejemplo, las páginas web, el HTML, el *JavaScript*, el css. La tercera capa es la que es difícil de explicar, pero es la que maneja que páginas son accesible y no, y se encarga de rescatar los datos del modelo y luego entregárselo a las vistas(Gonzalo Sánchez ,2014)

Dentro de las ventajas del empleo de este patrón arquitectónico se enuncian las siguientes (Mashita, 2012):

1. Se puede dividir la lógica de negocio del diseño, haciendo tu proyecto más escalable.
2. Te facilitará el uso de URL amigables, importantes para el SEO (Posicionamiento web), la mayoría de *frameworks* MVC lo controlan.
3. La mayoría de *frameworks* controlan el uso de la memoria Caché, hoy en día muy importante para el posicionamiento web, ya que buscadores como google dan prioridad a las webs que tengan menor tiempo de descarga.
4. Un *Framework* MVC te ayuda a controlar los recursos del servidor, evitando *Bugs* que puedan repercutir en el rendimiento, por ejemplo, muchas veces olvidamos cerrar conexiones a la base de datos, sobrecargando el servidor.
5. Utilizar herramientas con tecnología escalable hace más atractivo tu proyecto en caso de buscar inversión externa, muchas veces para hacer crecer un proyecto, es necesario buscar socios o *Bussines Angels* que te ayuden a impulsarlo.

2.6. Patrones de diseño

Brad Appleton define un patrón de diseño de la manera siguiente: “Es una mezcla con nombre propio de puntos de vista que contienen la esencia de una solución demostrada para un problema recurrente dentro de cierto contexto de necesidades en competencia”. Dicho de otra manera, un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre “fuerzas” que afectan la manera en la que se aplica y en la que se utiliza dicho patrón. El objetivo de cada patrón de diseño es proporcionar una descripción que permita a un diseñador determinar si el patrón es aplicable al trabajo en cuestión, si puede volverse a usar (con lo que se ahorra tiempo de diseño) y si sirve como guía para desarrollar un patrón distinto en funciones o estructura(*Largman*, 2011).

Se clasifican en dos categorías: GRASP, es un acrónimo de *General Responsibility Assignment Software Patterns* (en español, patrones generales de software para asignar responsabilidades) y GoF es un acrónimo de *Gang-Of-Four* (en español, La pandilla de los cuatro). Para el diseño del desarrollo del componente para generar alertas en el Sistema *Xilema Smart Keeper 3.0* se tuvieron en cuenta los siguientes patrones GRASP:

Alta cohesión:

Define que la información almacenada en una clase debe ser coherente y estar relacionada con esta. Propone, además, que no se debe saturar una clase de métodos, sino asignar las responsabilidades a cada clase correspondiendo a la información que almacena. Este patrón se utiliza en todas las clases del módulo, permite incrementar la claridad y facilitar la comprensión del diseño. Simplificar el mantenimiento y las mejoras. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Una clase con alta cohesión es relativamente fácil de mantener, entender y reutilizar(Largman, 2011)

El componente para generar alertas en el Sistema *Xilema Smart Keeper 3.0* implementa la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *Controller*, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

Bajo acoplamiento:

El patrón de bajo acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a resultados negativos que puede producir un acoplamiento alto.

El bajo acoplamiento soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio. No se puede considerar de forma aislada a otros patrones como el Experto o el de Alta Cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección al asignar una responsabilidad.

No existe una medida absoluta de cuando el acoplamiento es demasiado alto. Lo que es importante es que un desarrollador pueda medir el grado de acoplamiento actual, y evaluar si aumentarlo le causara problemas. En general las clases que son inherentemente muy genéricas por naturaleza, y con una probabilidad de reutilización alta, deberían tener un acoplamiento especialmente bajo(Largman ,2011).

Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Experto en información (o experto):

El experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. El experto no pretende ser una idea oscura o extravagante; expresa la intuición común de que los objetos hacen las cosas relacionadas con la información que tienen. El cumplimiento de la responsabilidad a menudo requiere información que se encuentra dispersa por diferentes clases de objetos. Esto significa que hay muchos expertos en información parcial que colaboraran en la tarea. Por lo general el experto conduce a diseños donde los objetos del software realizan aquellas operaciones que normalmente se hacen a objetos inanimados del mundo real que representan (*Largman ,* 2011).

El uso del patrón es evidenciado en la utilización del *document* manager encargado de mapear la Base de Datos. Se utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades. Las clases de abstracción de datos, poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla de la base de datos, por ejemplo, el uso de la clase *Category*, que contiene atributos, constructor y métodos para representar la información de la entidad.

Controlador: Un Controlador es un objeto de interfaz no destinado al usuario, se encarga de manejar un evento del sistema a una clase que represente el sistema global. En el componente este patrón se evidencia en A*lertasControler* que la encargada de controlar el proceso de generación de alertas.

2.7. Modelo de diseño

El modelo de diseño es aquel que describe la realización de los casos de uso del sistema, y sirve como una abstracción del modelo de implementación y el código fuente del software. Su objetivo fundamental es transmitir, a través de la representación mediante diagramas, una comprensión en profundidad de los aspectos relacionados con los requerimientos no funcionales y restricciones relacionados con los lenguajes de programación, en resumen, cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades (*Presman*, 2011).

2.7.1. Diagrama de clases del diseño con estereotipos web

Un Diagrama de Clases de Diseño muestra la especificación para las clases software de una aplicación. Incluye clases, asociaciones, atributos. Interfaces, con sus operaciones y constantes, métodos, navegabilidad y dependencias. A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real (*Presman* ,2011).

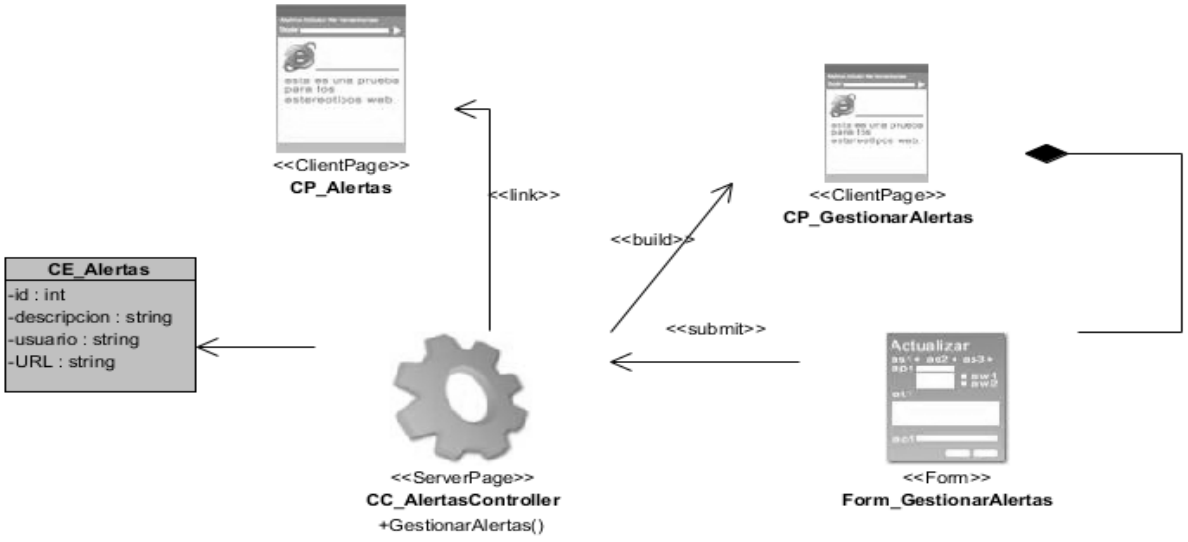

Figura 1 Diagrama de clases del diseño del HU configurar alertas

2.7.2. Diagrama de secuencias

Dentro de los diagramas de comportamiento en UML que permiten enfatizar las interacciones entre los objetos se encuentran los diagramas de secuencias, este describe el comportamiento del sistema y las operaciones que se realizan representando los objetos y los mensajes que se intercambian, ya que en un sistema real y funcional los objetos interactúan entre sí, y tales iteraciones suceden con el tiempo que se asigna, es decir que el diagrama de secuencias de UML es una mecánica de interacción en base a los tiempos.

Un diagrama de secuencias muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo, en el cual se indicarán los módulos o clases que formaran parte del programa y las llamadas que se hacen cada uno de ellos para realizar una tarea determinada, por esta razón permite observar la perspectiva cronológica de las interacciones. Es importante recordar que el diagrama de secuencias se realiza a partir de la descripción de un caso de uso.(Cevallos 2015). A continuación, se muestran los diagramas relacionados con los requisitos funcionales RF1 y RF8 respectivamente.

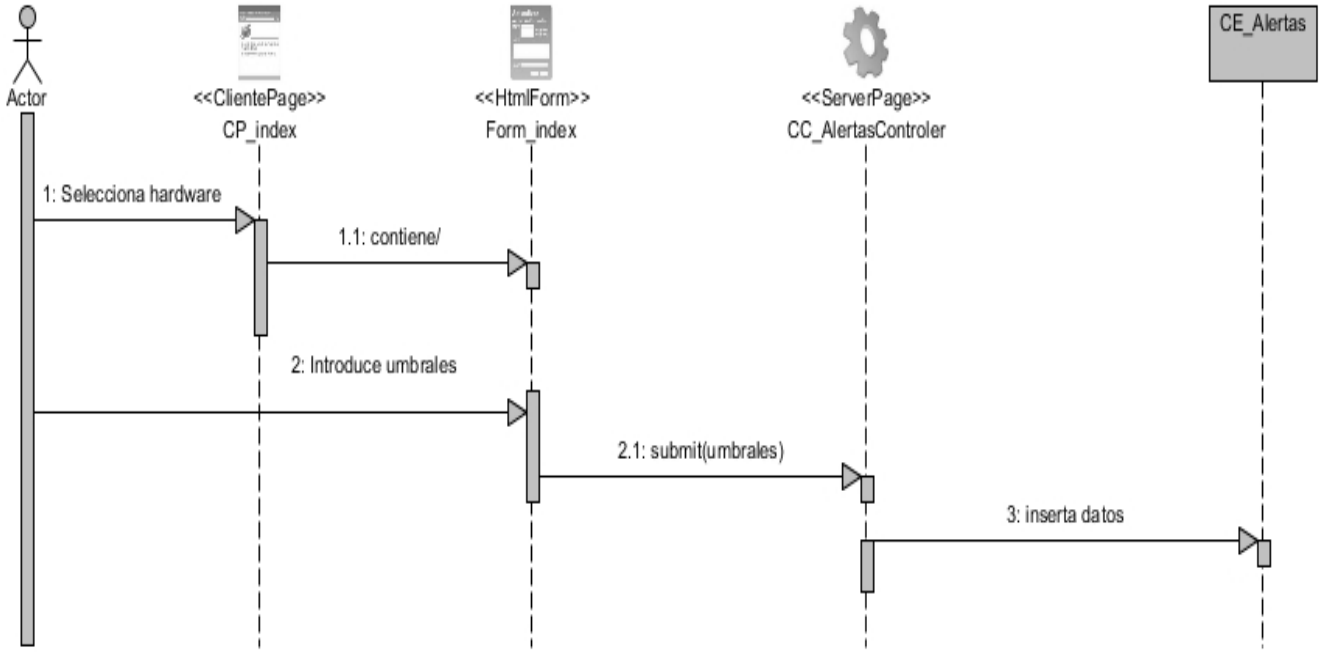

Figura 2 Diagrama de secuencias HU 1 Configurar alertas

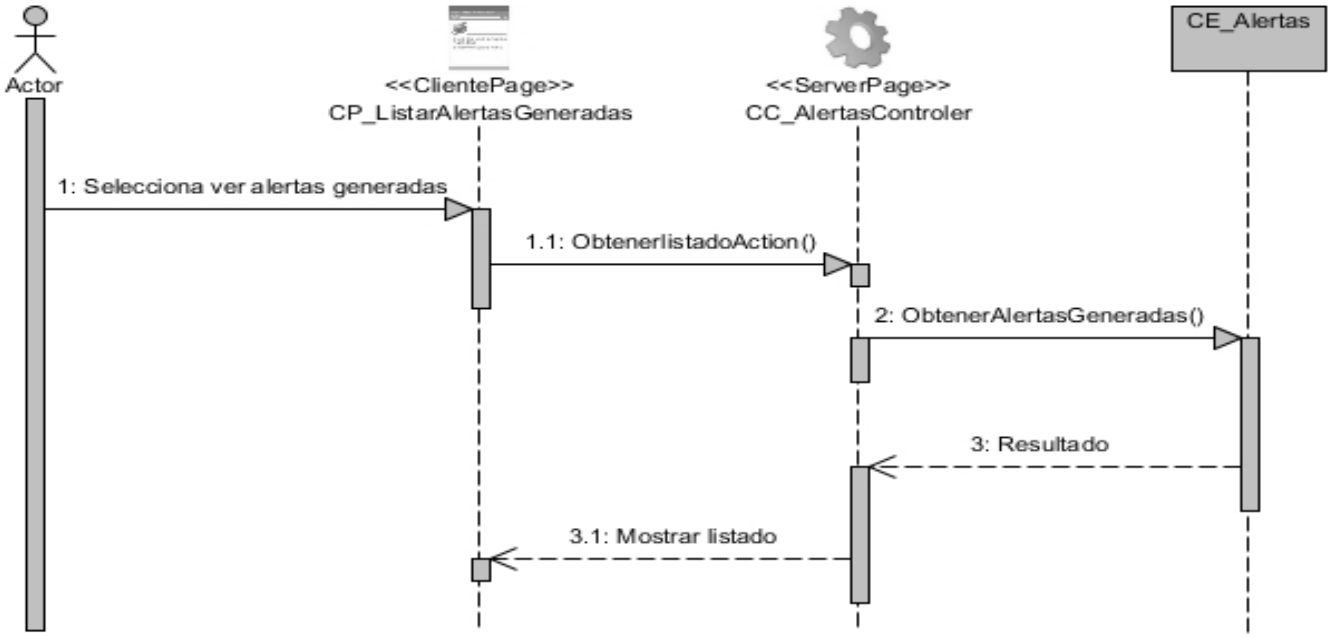

Figura 3 Diagrama de Secuencias HU 8 Listar alertas enviadas

2.8. Diseño de la base de datos

La persistencia de la información es uno de los requisitos claves a tener en cuenta a la hora de diseñar la arquitectura de una aplicación informática. En este sentido, es de vital importancia la realización de un diseño de base de datos coherente con las necesidades puntuales del sistema a desarrollar, permitiendo obtener acceso a información exacta y actualizada. El objetivo del diseño de una base de datos, es la representación de las clases entidades y sus relaciones, que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información.

El componente que se propone hace uso de una base de datos ya que es necesario persistir una cantidad considerable de información relacionada con los logs generados por los usuarios al navegar por la web y los datos introducidos por estos. A continuación, se explican las tablas que se encuentran en la base de datos.

Alertas: En esta tabla se almacenan los datos introducidos por el administrador referentes al uso de la RAM, CPU y el disco duro(hdd) para ser utilizados posteriormente.

Alertas Enviadas: Almacena la información de las alertas generadas por el componente para ser mostradas en la interfaz de la aplicación.

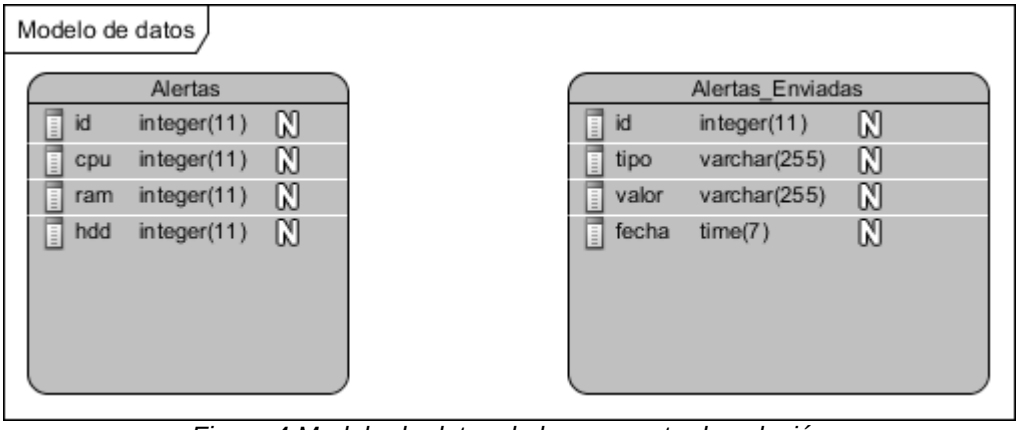

Figura 4 Modelo de datos de la propuesta de solución

2.9. Modelo de despliegue

El diagrama muestra la configuración de nodos de procesamiento en tiempo de ejecución y las instancias de componentes y objetos que se encuentran dentro de esos nodos. Los componentes representan

manifestaciones de ejecución de unidades de código. Los componentes que no existen como entidades en tiempo de ejecución (porque han sido recopilados de distancia) no aparecen en estos diagramas, sino que debe ser mostrado en los diagramas de componentes. Un diagrama de despliegue muestra casos mientras que un diagrama de componentes muestra la definición de tipos de componentes propios.

El diagrama de despliegue contiene instancias de nodos conectados por enlaces de comunicación. La instancia de nodo puede contener instancias de tiempo de ejecución, como instancias de componentes y objetos. Instancias de componentes y objetos también pueden contener otros objetos. El modelo puede mostrar las dependencias entre las instancias y sus interfaces, y también puede modelar la migración de entidades entre nodos u otros contenedores(Sarmiento ,2013).

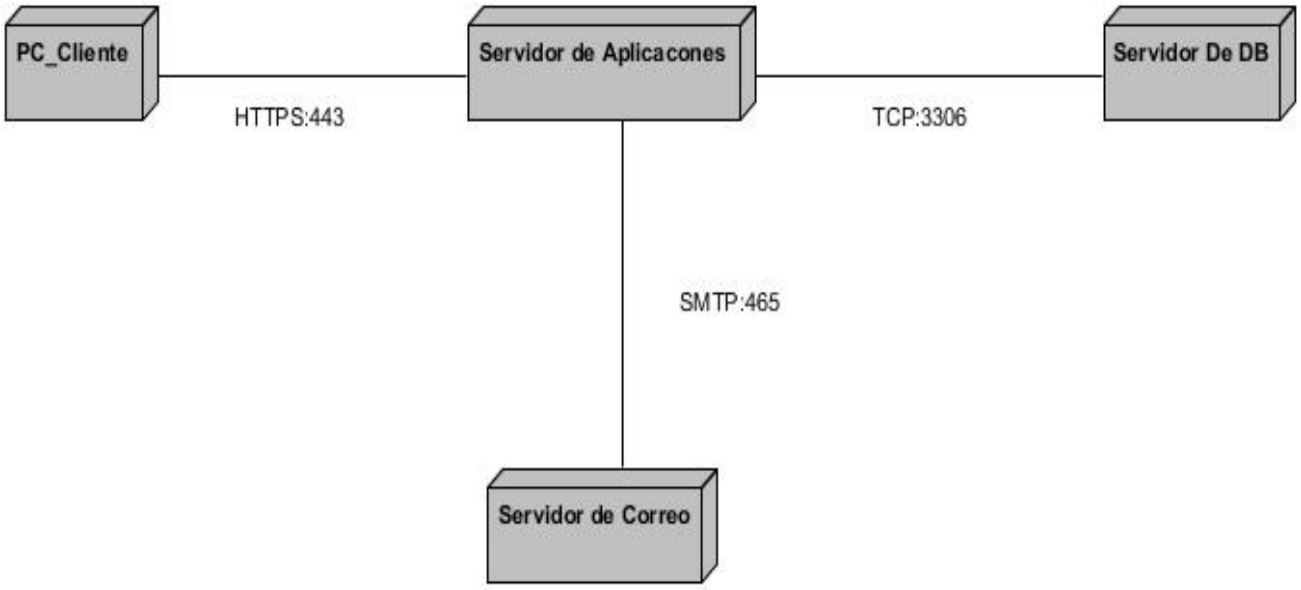

Figura 5 diagrama de despliegue

Se describe a cada nodo y su interrelación del Diagrama de Despliegue:

Servidor de Correo: Este nodo es el encargado del almacenamiento de los datos del envío y recepción del correo electrónico que genera las alertas del componente para generar alertas en el Sistema *Xilema Smart Keeper*. La comunicación se establece a través del protocolo SMTP.

Servidor de aplicaciones: Interfaz gráfica del sistema *Xilema Smart Keeper*, donde interactúa el Administrador y realiza todas las acciones.

PC_Cliente: Contiene la estación de trabajo cliente, necesita un navegador web que conecte con la aplicación web alojado en el servidor de aplicaciones, utiliza el protocolo de comunicación HTTP.

Servidor de BD: Contiene los datos necesarios para guardar y listar las alertas generadas.

2.10. Conclusiones del capítulo

Una vez realizada la fundamentación teórica que sustentó la presente investigación y concluido el flujo de análisis y diseño de la propuesta de solución, se puede arribar a las siguientes conclusiones:

- La identificación de los patrones de diseño y el estilo arquitectónico, evidencia que la solución propuesta tiene un alto grado de resistencia ante posibles modificaciones. La elaboración de diagramas de clases, facilitó la visión en cuanto a composición física y lógica del módulo.
- La realización del modelo de dominio ofrece una visión palpable de los componentes y los conceptos asociados a la creación del Componente para generar Alertas en el Sistema Xilema *Smart Keeper*, así como las relaciones entre estos.

Capítulo3: Implementación y validación del componente para generar alertas en el Sistema *Xilema Smart Keeper 3.0*

La fase de implementación en el desarrollo de un producto de *software*, es el mecanismo donde se ponen en práctica todas las descripciones y arquitecturas propuestas en las fases de análisis y diseño, es el complemento del trabajo de las fases que lo preceden dentro del proceso de desarrollo de *software*. La implementación ofrece una materialización precisa de los requisitos, con el objetivo de conformar el producto final requerido por el cliente.

Aparejado al proceso de implementación, el *software* que se crea debe ser sometido a determinadas pruebas que garanticen la correspondencia entre el producto y los requisitos definidos en las etapas anteriores. A esta etapa se le conoce como validación del sistema y en ella, pueden realizarse diferentes tipos de pruebas en función de los objetivos de las mismas.

3.2. Diagrama de componentes

El diagrama de componentes proporciona una visión física de la construcción del sistema de información. Muestra la organización de los componentes de *software*, sus interfaces y las dependencias entre ellos. Los elementos de estos diagramas son los componentes de software y las dependencias entre ellos.

Un componente es un módulo de *software* que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Una interfaz establece las operaciones externas de un componente, las cuales determinan una parte del comportamiento del mismo. Además, se representan las dependencias entre componentes o entre un componente y la interfaz de otro, es decir uno de ellos usa los servicios o facilidades del otro.

Estos diagramas pueden incluir paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías, etc.(Manuel, 2013).

Tabla 4 Descripción de los componentes del Diagrama de Componentes

Componentes					Descripción
Xilema Smart Keeper	Componente para generar alertas en el Sistema Xilema Smart Keeper 3.0	Modelo	Alertas.php		Clase entidad que almacena los datos para realizar las consultas necesarias para generar las alertas.
			AlertasEenviads.php		Clase entidad que almacena los datos para realizar las consultas necesarias para listar las alertas generadas.
		Controlador	AlertasController.php		Clase controladora que contiene las acciones relacionadas con la configuración de las alertas. Entre las funciones que realiza se encuentran la inserción en la base de datos y consultas a la misma para chequear los valores guardados.
		Vista	alertas	index.html.twig	Permite al administrador seleccionar los valores para la configuración de las alertas a generar por el componente
				alertas.html.twig	Muestra un listado con las alertas enviadas que se generaron
				alertas_sevicios.html.twig	Muestra un listado con las alertas enviadas que se generaron
		routing.yml			Contiene el mapa de rutas URL que se enlazan a las acciones de los controladores del módulo.
	config.yml				Contiene las configuraciones generales del sistema Xilema Smart Keeper.

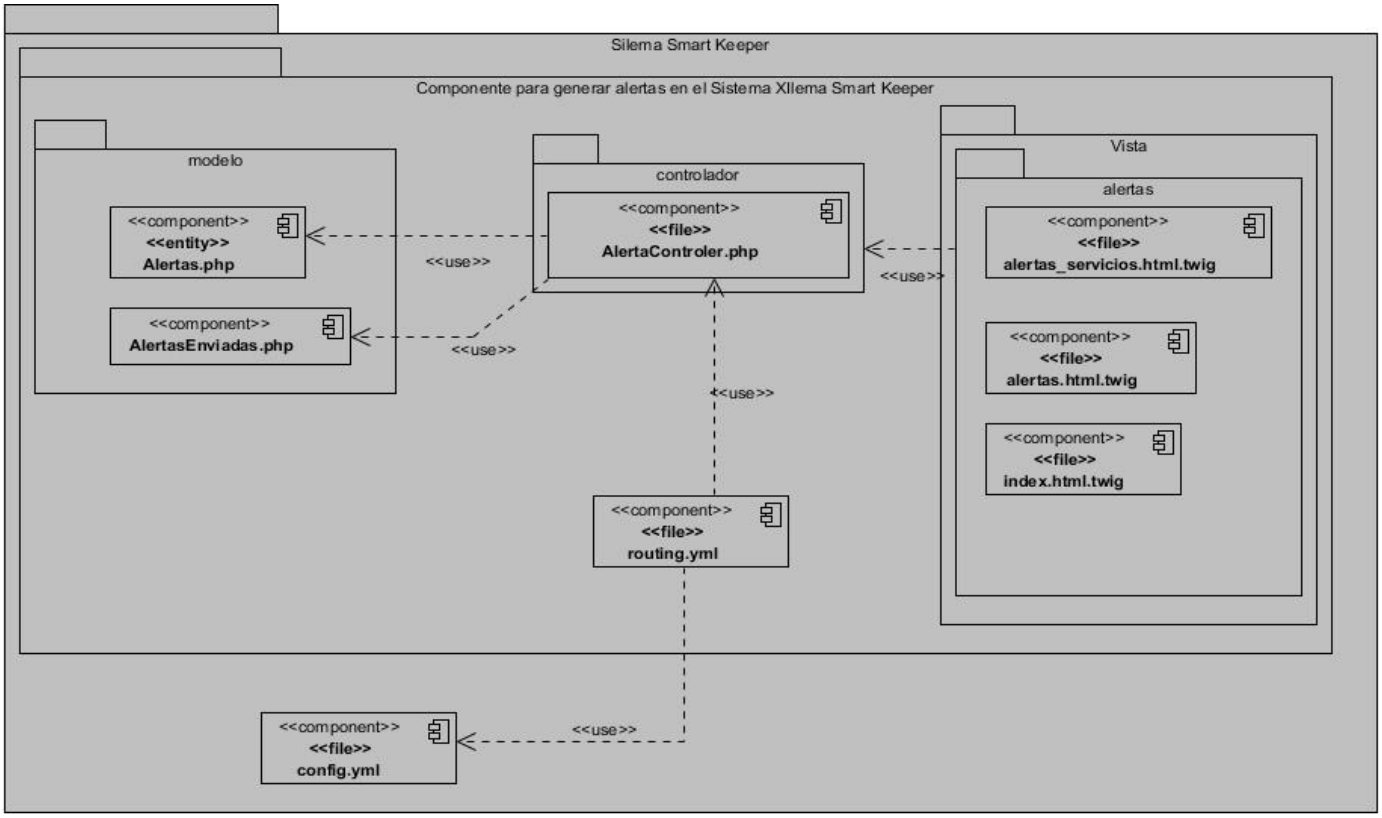

Figura 6 Diagrama de componentes

3.3. Estándares de codificación

Las convenciones de código básicamente son estándares que indican como se debe hacer algo, cuales son las reglas mínimas para realizar algún proceso, algo tan simple como declarar una variable, cuál debe ser el nombre de las clases, como se deben crear los paquetes etc.

Se debe saber que las convenciones de código en algunos casos sugieren como hacer las cosas, pero no obligan a hacerlo, no son camisa de fuerza. Por ejemplo, java dice que los paquetes deben llamarse todos en minúscula, pero eso no obliga a hacerlo, se puede crear un paquete con la primera letra mayúscula y aunque puede arrojar una advertencia no arroja un error y el programa funciona y funciona bien, pero a nivel de estándar estaría mal. Aparte de lo mencionado anteriormente algunas de las ventajas de aplicar las convenciones de Código son(Henao ,2014):

- Facilitan la lectura y entendimiento del código fuente.
- Reduce el costo del mantenimiento del código, costo se refiere tanto a nivel económico como esfuerzo y horas hombre.
- Por lo regular en todo software intervienen diferentes desarrolladores, si todos manejan el mismo lenguaje y convenciones, el desarrollo será mucho más ágil.
- Permite manejar un estándar de programación en el equipo de trabajo.
- Facilita agregar o la modificación de funcionalidades.
- El seguir las convenciones y aplicar los estándares habla muy bien de nosotros como desarrolladores.

Para facilitar el entendimiento del código y precisar un modelo a seguir, se adoptaron determinados estándares de codificación que a continuación se describen, agrupados por los aspectos en los que fueron utilizados. Cuando se trabaja con Symfony3.4 aconsejable seguir su estándar de codificación. Esto implica imitar el código ya existente.

Estructura

1. Añadir un solo espacio después de cada delimitador coma;
2. Añadir un solo espacio alrededor de los operadores (==, = &&, ...);

```
public function newAction(Request $request) {
    $em = $this->getDoctrine()->getManager();
    $alert = $em->getRepository("AppBundle:Alertas")->findOneBy(array("id" => 0));
    $alert->setCpu($request->get("cpu"));
    $alert->setRam($request->get("ram"));
    $alert->setHdd($request->get("hdd"));
    $alert->setUrl($request->get("url"));
    $alert->setConsumo($request->get("consumo"));
    $em->flush();

    return $this->redirectToRoute("alertas");
}
```

Figura 7 ejemplo de estándar de diseño

3. Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último;

4. Añadir una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración if);

```
public function newAction(Request $request) {
    $em = $this->getDoctrine()->getManager();
    $alert = $em->getRepository("AppBundle:Alertas")->findOneBy(array("id" => 0));
    $alert->setCpu($request->get("cpu"));
    $alert->setRam($request->get("ram"));
    $alert->setHdd($request->get("hdd"));
    $alert->setUrl($request->get("url"));
    $alert->setConsumo($request->get("consumo"));
    $em->flush();

    return $this->redirectToRoute("alertas");
}
```

Figura 8 Ejemplo de estándar de diseño 1

5. Usar llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga;
6. Definir una clase por archivo, esto no se aplica a las clases ayudante privadas, de las cuales no se tiene la intención de crear una instancia desde el exterior;
7. Declarar las propiedades de clase antes que los métodos;
8. Primero declarar los métodos públicos, luego los protegidos y finalmente los privados;
9. Usar paréntesis al crear instancias de clases independientemente de la cantidad de argumentos que tenga el constructor;
10. Las cadenas de mensajes de excepción deben concatenarse usando *sprintf*;

Convenciones de nomenclatura

1. Utilizar mayúsculas intercaladas —sin guiones bajos— en nombres de variable, función, método o argumentos;

```
public function alertasAction() {
   $em = $this->getDoctrine()->getManager();
   $alertas = $em->getRepository("AppBundle:AlertasEnviadas")->findBy(array(),array("fecha"=>"DESC"));
```

Figura 9 ejemplo de estándar de diseño 2

Usar guiones bajos para nombres de opción y nombres de parámetro;

```
return $this->render('alertas/alertas_servicios.html.twig');
```

Figura 10 ejemplo de estándar de diseño 3

2. Utilizar espacios de nombres para todas las clases;
3. Prefijar las clases abstractas con *Abstract*. Por favor, tener en cuenta que algunas de las primeras clases de *Symfony* no siguen esta convención y no se han rebautizado por razones de compatibilidad hacia atrás. No obstante, todas las nuevas clases abstractas tienen que seguir esta convención de nomenclatura;
4. Sufijar las interfaces con Interface;
5. Sufijo las características con *Trait*;
6. Sufijar las excepciones con *Exception;*
7. Utilizar caracteres alfanuméricos y guiones bajos para los nombres de archivo;

Documentación

1. Añadir bloques *PHPDoc* a todas las clases, métodos y funciones;
2. Omitir la etiqueta *@return* si el método no devuelve nada;
3. Las anotaciones *@package* y *@subpackage* no se utilizan.

3.4. Validación del sistema

Las pruebas son los procesos que permiten verificar y revelar la calidad de un producto de *software*, una vez terminado la implementación del producto es necesario realizarle pruebas, con el objetivo de detectar errores en la aplicación y la documentación, se identifican posibles fallos de implementación, calidad, o usabilidad de un *software*, este proceso resulta de gran importancia ya que da una medida de la calidad del mismo, siempre que se lleve a cabo de la forma correcta. En este proceso se ejecutan pruebas dirigidas al sistema de software en su totalidad, con el objetivo de medir el grado en que el *software* cumple con los requerimientos. Las mismas persiguen como objetivo fundamental, la detección de las no conformidades que este procedimiento arroja y que son de vital importancia en el proceso de desarrollo del *software*.

3.4.1. Pruebas funcionales

Toda prueba funcional se basa en los requerimientos entregados para el desarrollo de un software. Éstas se realizan mediante casos de prueba cuyo fin es validar que el software cumple con el nivel de calidad requerido para entrar en producción.

Marcela León, Jefa de Calidad de la Unidad Ingeniería en *Software* de *Kibernum*, señala que "las pruebas funcionales son importantes para el desarrollo del proyecto porque abarcan en su mayoría los requerimientos y solicitudes realizadas por los usuarios, por lo que mediante estas podemos determinar si lo que se está construyendo cumple con los niveles de aceptación descritos por el cliente".

Dentro de los principales beneficios que tienen este tipo de pruebas está la mitigación del riesgo de aparición de fallos en producción, el cumplimiento de los objetivos de los proyectos en términos de calidad y resultados esperados principalmente, pero también de plazos y costos. Además, la identificación temprana de riesgos y desviaciones asociadas a la calidad, permiten evitar problemas con proveedores, mayores costos para el cliente y finalmente generar confianza en el producto o sistema bajo test(*Kibernum* ,2015).

A continuación, se presenta el **caso de prueba correspondiente al HU configurar alertas** seguido del **Caso de prueba de la HU listar alertas enviadas** con el objetivo de comprobar el correcto funcionamiento del sistema al configuradas las alertas y mostrar el listado correspondiente a las mismas

Tabla 5 Caso de Prueba del RF1_Configurar alertas

Escenario	**Descripción**	**Valor CPU**	**Valor RAM**	**Valor HDD**	**Respuesta Flujo del sistema**	**Central**
EC1.1 configurar alertas de	El administrador introduce los datos en el sistema y este comprueba que	V 60	V 75	V 80	Valida los datos y si son correctos se guardan en la base de datos para realizar el monitoreo y	1.Se llenan los datos de entrada (CPU, RAM, HDD).

forma correcta .	los mismos sean correctos.				generar las posibles alertas.	2.Se selecciona la opción guardar. 3.Se guardan los datos son en la base de datos

Tabla 6 Caso de Prueba del RF8_Listar alertas enviadas

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC1.1 listar alertas enviadas	Se muestra una interfaz con las alertas enviadas	Muestra una interfaz con las alertas enviadas .	1.Se selecciona la opción ¨Alertas_hardware o Alertas_Servicios

Esta prueba comprueba el correcto funcionamiento del componente en cuanto a la generación del listado de las alertas enviadas.

Como resultado final de este tipo de pruebas se obtuvo que:

En una primera iteración se detectaron un total de 10 no conformidades, divididas en 7 de ortografía y 3 de funcionalidad. Las cuales fueron resueltas de forma satisfactoria. Para una segunda iteración se identificó una nueva no conformidad que se resolvió de forma inmediata y finalmente en una tercera iteración no se detectaron no conformidades obteniendo resultados satisfactorios (ver figura11).

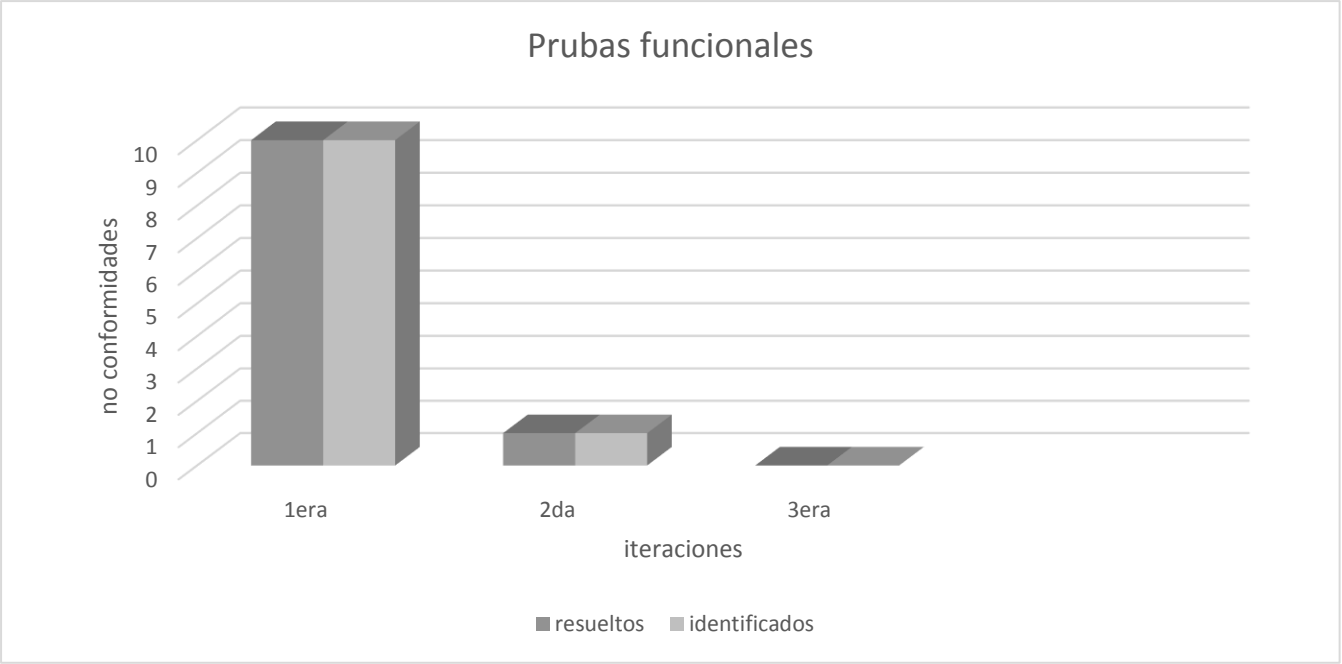

Figura 11 Resultado de las pruebas funcionales

3.4.2. Pruebas de integración

El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes(Cillero ,2014).

Una vez realizadas las pruebas funcionales a cada componente interno de manera independiente, y verificado que las funcionalidades implementadas se corresponden de acuerdo a los requisitos funcionales y no funcionales establecidos; se pudo comprobar el correcto funcionamiento de los componentes mediante el estudio del flujo de datos entre ellos. Posterior a estas pruebas, se hace necesaria la realización de pruebas de integración, con la finalidad de validar la compatibilidad y el funcionamiento de las interfaces que comunican las diferentes partes que componen la solución informática.

Algunas de las acciones que se llevaron a cabo para la realización de estas pruebas fueron:

- ✓ Comprobación del funcionamiento del enlace entre el Sistema *Xilema Smart Keeper 3.0* y el Componente para Generar Alertas.
- ✓ Verificación de la conexión entre el Componente y la base de datos en *MySQL*.
- ✓ Validación de las rutas de acceso, en el *routing* principal del sistema *Xilema Smart Keeper.*

3.4.3. Pruebas de carga y Estrés

Las pruebas de carga son proyectos que permiten detectar problemas de rendimiento en las aplicaciones, evaluar si satisfacen las necesidades del cliente antes de su publicación, y determinar las respuestas de la aplicación a diferentes niveles de uso una vez publicada. Una prueba de carga simula el acceso de varios usuarios al mismo tiempo, siguiendo el patrón de prueba que se haya creado(*ITblogsoget*i ,2017).

En el diseño del plan de pruebas de rendimiento para el componente, se tuvo en cuenta, las diferentes acciones que el usuario(administrador)puede realizar en el sistema. Con una muestra de 1000 peticiones concurrentes con un periodo de subida de un segundo la herramienta *JMeter* arrojo el reporte mostrado en la tabla 7.

Tabla 7 Resultados de las pruebas de rendimiento mediante el uso de Jmeter

Petición	Cantidad de peticiones	Tiempo de respuesta(ms)	Tasa de error(%)
Generar alerta de CPU	1000	16.45	0
Generar alerta de RAM	1000	16.45	0
Generar alerta de HDD	1000	16.45	0

Listar alertas generadas	1000	15.761	0
Total	4000	16.27	0

Como se puede observar en el análisis del resumen arrojado por la herramienta *JMeter*, para un total de 4000 muestras que se le realizaron al módulo, se alcanzó un rendimiento de 16.27 peticiones por segundo, con un porcentaje de 0% de errores para cada petición realizada. A partir de este resultado se considera que el componente para generar alertas en el sistema Xilema Smart Keeper responde correctamente ante situaciones de carga y estrés en función de la muestra utilizada teniendo en cuenta lo planteado por *Jakob Nielsen* en su libro *Usability Engineering*. En el cual plantea que 0.1 segundos es aproximadamente el límite para que el usuario sienta que el sistema está reaccionando instantáneamente, 1.0 segundo es aproximadamente el límite para que el flujo de pensamiento del usuario permanezca ininterrumpido y 10 segundos es aproximadamente el límite para mantener la atención del usuario centrada en el diálogo. Para retrasos más largos, los usuarios querrán realizar otras tareas mientras esperan que la computadora termine, por lo que se les debe dar una retroalimentación que indique cuándo se espera que la computadora termine(*stackoverflow* ,2017).

Para realizar la validación de los resultados de las pruebas se verificó, además, la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, elementos que pueden provocar cierto retraso en la generación de la carga en las pruebas e interferencia en la obtención de los resultados, dichos parámetros a tener en cuenta son:

Ambiente de prueba: El despliegue del componente se realizó en una computadora con *LinuxMint* 18.2 de 64 bits como sistema operativo, con 4 GB de memoria RAM y 4 procesadores a velocidad 3.30 GHz.

3.6. Conclusiones del capítulo

- ✓ Con el desarrollo de las pruebas se pudo verificar la calidad y el adecuado funcionamiento de la implementación de la propuesta de solución.

- ✓ En este capítulo se lograron desarrollar las funcionalidades propuestas para darle solución a la problemática existente respecto a la gestión de alertas en el Sistema Xilema Smart Keeper.
- ✓ La estandarización del código creado posibilitó agregar aspectos a la implementación que facilitan su entendimiento y reusabilidad en un futuro.

Conclusiones

Finalizada la investigación se obtiene el Componente para generar alertas en el sistema Xilema Smart Keeper, este facilita la administración y configuración de las mimas y agiliza el proceso de estructuración de dichas alertas. Esto se debe a que la herramienta permite gestionar los diferentes tipos de alertas de forma simultánea.

Otros aspectos significativos que se pueden destacar son:

- A partir del estudio y análisis de los fundamentos teóricos relacionados con la generación de alertas mediante filtros de contenido, se determinó que existen sistemas similares. Pero ninguno de ellos permite generar alertas de hardware y software al mismo tiempo, definiéndose una propuesta de solución de acuerdo a las necesidades existentes.
- La elaboración de los artefactos propuestos por la metodología de desarrollo y el levantamiento de requisitos permitieron una mayor comprensión del componente desarrollado, así como la identificación de los procesos y características del mismo.
- Las pruebas realizadas garantizan un correcto funcionamiento del componente implementado, así como una total integración al resto de los componentes del sistema Xilema Smart Keeper.
- La identificación de los patrones de diseño y el estilo arquitectónico evidencia que la solución propuesta tiene un alto grado de adaptabilidad ante posibles modificaciones. La elaboración de diagramas de clases y de secuencia, facilitó la visión en cuanto a composición física y lógica del componente.

Recomendaciones

Durante la realización del componente y como resultado de la investigación realizada, a pesar de haberse cumplido los objetivos de este trabajo, surgieron algunas ideas que pueden ampliar las prestaciones del software creado. El autor de este trabajo recomienda al equipo de desarrollo del Sistema Xilema Smart Keeper profundizar en:

- ✓ El desarrollo de funcionalidades que permitan a la aplicación generar las alertas desde servidores remotos.
- ✓ Valorar la posibilidad de permitir al administrador seleccionar desde la aplicación aquellos servicios que el considere que se deben monitorear.

Bibliografía

ALEGSA, 2015. Definición de Sistema informático (SI). [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.alegsa.com.ar/Dic/sistema_informatico.php.

analisis y diseño: determinación de los requerimientos de información. *analisis y diseño* [en línea], 2013. [Consulta: 30 mayo 2018]. Disponible en: http://2analisis01dise.blogspot.com/p/de.html.

AULA FORMATIVA, 2016. Definición, usos y ventajas del lenguaje HTML5. *Blog Aula Formativa* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://blog.aulaformativa.com/definicion-usos-ventajas-lenguaje-html5/.

AVALONINFORMATICA, 2017. Monitorización ED | Avalon Informática. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.avaloninformatica.com/monitorizacion-event-detect.

BAQUERO HERNÁNDEZ, L.R., 2016. *Extensión de la herramienta Visual Paradigm for UML para la UCI*. 2016. S.l.: s.n.

CALENDAMAIA, 2014. NetBeans. *Genbeta Dev* [en línea]. [Consulta: 21 mayo 2018]. Disponible en: https://www.genbetadev.com/herramientas/netbeans-1.

CEVALLOS, K., 2015. UML: Diagrama de Secuencia. *INGENIERÍA DEL SOFTWARE* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://ingsotfwarekarlacevallos.wordpress.com/2015/07/07/uml-diagrama-de-secuencia/.

CILLERO, 2014. Pruebas de Integración. *manuel.cillero.es* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/integracion/.

CULTURACION, 2014. Qué es y para que sirve MySQL. *Culturación* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://culturacion.com/que-es-y-para-que-sirve-mysql/.

DAWSON, A. y CHAD, A., [sin fecha]. squid : Optimising Web Delivery. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.squid-cache.org/.

DEFINICIÓN, 2016. Definición de Componente | Que es, Conceptos y Significados. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://definicionyque.es/componente/.

FERREIRA, I., 2017. IBM Tealeaf, introducción a los eventos. *Analítica web* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.analiticaweb.es/ibm-tealeaf-introduccion-eventos/.

GOMEZ, P., 2005. *PHP y MySQL Tecnologías para el desarrollo.* Ediciones Díaz de Santos. España: s.n.

GONZALO SÁNCHEZ, 2014. ¿Qué es MVC? *Tecnología digital* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://medium.com/tecnologia-digital/que-es-mvc-a84cdc2ed088.

HENAO, C., 2014. CoDejaVu: Qué son las Convenciones de Código? *CoDejaVu* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://codejavu.blogspot.com/2014/04/que-son-las-convenciones-de-codigo.html.

IREO, 2018. ManageEngine Firewall Analyzer - ManageEngine. *IREO - Mayorista de Soluciones TI* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://www.ireo.com/fabricantes-y-productos/manageengine/manageengine-firewall-analyzer.

ISRAELCCM, 2017. Lenguajes de programación. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://es.ccm.net/contents/304-lenguajes-de-programacion.

ITBLOGSOGETI, 2017. PRUEBAS DE CARGA Y ESTRÉS. *itblogsogeti* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://itblogsogeti.com/2017/03/23/pruebas-de-carga-y-estres/.

JAVA, 2015. ¿Qué es Java? [en línea]. [Consulta: 14 mayo 2018]. Disponible en: https://www.java.com/es/about/whatis_java.jsp?bucket_value=desktop-firefox59-ubuntulinux&in_query=no.

jQuery: Qué es, Orígenes, Ventajas y Desventajas. *Información práctica sobre Redes, Linux, Seguridad y Hacking para profesionales de TI. Capacity Academy* [en línea], 2013. [Consulta: 8 mayo 2018]. Disponible en: http://blog.capacityacademy.com/2013/03/16/jquery-que-es-origenes-ventajas-desventajas/.

JULIAN, P.P. y ANA, G., 2014. Definición de alerta - Qué es, Significado y Concepto. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://definicion.de/alerta/.

KIBERNUM, 2015. ¿Por qué son importantes las pruebas funcionales? *Kibernum Chile* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.kibernum.com/noticias/por-que-son-importantes-las-pruebas-funcionales-2/.

LARGMAN, C., 2011. *UML y Patrones*. Segunda Edicion. Madrid: Pearson Educacion. ISBN ISBN 84-205-3438-2.

MANAGEENGINE, 2017. Firewall Configuration & Log Management by ManageEngine Firewall Analyzer. *ManageEngine Firewall Analyzer* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://www.manageengine.com/products/firewall/.

MANUEL, C., 2013. Diagrama de Componentes. *manuel.cillero.es* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

MARKETING ONLINE, 2016. Metodologías del Desarrollo de Software. *OK HOSTING | Hospedaje Web, Dominios, Desarrollo de Software, Marketing Online, SEO* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://okhosting.com/blog/metodologias-del-desarrollo-de-software/.

MASHITA, 2012. Desarrollo Web y Curiosidades: 10 Ventajas de utilizar Modelo Vista Controlador (MVC) en tus proyectos. *Desarrollo Web y Curiosidades* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://blogdewebin.blogspot.com/2012/04/10-ventajas-de-utilizar-model-vista.html.

MGOEDTEL, 2016. Respuesta a eventos con las alertas de Azure Log Analytics. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://docs.microsoft.com/es-es/azure/log-analytics/log-analytics-tutorial-response.

NETBEANS, 2017. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [en línea]. [Consulta: 21 mayo 2018]. Disponible en: https://netbeans.org/index_es.html.

PÉREZ PORTO, J., 2008. Definición de sistema. *Definición.de* [en línea]. [Consulta: 30 mayo 2018]. Disponible en: https://definicion.de/sistema/.

PEREZ VALDEZ, D., 2007. ¿Qué es Javascript? *Maestros del Web* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.maestrosdelweb.com/que-es-javascript/.

PMOINFORMATICA.COM, P. por, 2018. ¿Qué es un requerimiento funcional? [en línea]. [Consulta: 30 mayo 2018]. Disponible en: http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html.

PRESMAN, R.S., 2011. *Ingeniería del software un enfoque práctico*. Septima Edicion. México: Educacion. ISBN ISBN 978-607-15-0314-5.

RODRÍGUEZ SÁNCHEZ, T., 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* 12 noviembre 2014. S.l.: s.n.

SALIXNETWORKS, 2015. Filtrado de Contenido Web - Control de Internet. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.salixnetworks.com/filtrado_web.html.

SARMIENTO, F., Johana, 2013. Visión General de los Diagramas de Despliegue. *UML* [en línea]. [Consulta: 10 mayo 2018]. Disponible en: http://umldiagramadespliegue.blogspot.com/.

SCHNEIER, B. y KELSEY, J., 2015. *Cryptographic Support for Secure Logs on Untrusted Machines*. 2 marzo 2015. S.l.: s.n.

STACKOVERFLOW, 2017. performance - What is considered a good response time for a dynamic, personalized web application? *Stack Overflow* [en línea]. [Consulta: 11 mayo 2018]. Disponible en: https://stackoverflow.com/questions/164175/what-is-considered-a-good-response-time-for-a-dynamic-personalized-web-applicat.

TELLO, javier, 2009. Requisitos No-Funcionales (NFR). *Softqanetwork.com* [en línea]. [Consulta: 30 mayo 2018]. Disponible en: http://www.softqanetwork.com/requisitos-no-funcionales-nfr.

TOLEDO, por D., 2016. JMeter, un viejo amigo en plena forma. *Solidgear* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://solidgeargroup.com/jmeter-un-viejo-amigo-en-plena-forma?lang=es.

AULA FORMATIVA, 2016. Definición, usos y ventajas del lenguaje HTML5. *Blog Aula Formativa* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://blog.aulaformativa.com/definicion-usos-ventajas-lenguaje-html5/.

AVALONINFORMATICA, 2017. Monitorización ED | Avalon Informática. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.avaloninformatica.com/monitorizacion-event-detect.

BAQUERO HERNÁNDEZ, L.R., 2016. *Extensión de la herramienta Visual Paradigm for UML para la UCI*. 2016. S.l.: s.n.

CALENDAMAIA, 2014. NetBeans. *Genbeta Dev* [en línea]. [Consulta: 21 mayo 2018]. Disponible en: https://www.genbetadev.com/herramientas/netbeans-1.

CEVALLOS, K., 2015a. UML: Diagrama de Secuencia. *INGENIERÍA DEL SOFTWARE* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://ingsotfwarekarlacevallos.wordpress.com/2015/07/07/uml-diagrama-de-secuencia/.

CEVALLOS, K., 2015b. UML: Diagrama de Secuencia. *INGENIERÍA DEL SOFTWARE* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://ingsotfwarekarlacevallos.wordpress.com/2015/07/07/uml-diagrama-de-secuencia/.

CILLERO, 2014. Pruebas de Integración. *manuel.cillero.es* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/integracion/.

CULTURACION, 2014. Qué es y para que sirve MySQL. *Culturación* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://culturacion.com/que-es-y-para-que-sirve-mysql/.

DAWSON, A. y CHAD, A., [sin fecha]. squid : Optimising Web Delivery. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.squid-cache.org/.

DEFINICIÓN, 2016. Definición de Componente | Que es, Conceptos y Significados. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://definicionyque.es/componente/.

FERREIRA, I., 2017. *IBM Tealeaf*, introducción a los eventos. *Analítica web* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.analiticaweb.es/ibm-tealeaf-introduccion-eventos/.

GOMEZ, P., 2005. *PHP y MySQL Tecnologías para el desarrollo.* Ediciones Díaz de Santos. España: s.n.

GONZALO SÁNCHEZ, 2014. ¿Qué es MVC? *Tecnología digital* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://medium.com/tecnologia-digital/que-es-mvc-a84cdc2ed088.

HENAO, C., 2014. *CoDejaVu*: ¿Qué son las Convenciones de Código? *CoDejaVu* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://codejavu.blogspot.com/2014/04/que-son-las-convenciones-de-codigo.html.

IREO, 2018. *ManageEngine Firewall Analyzer - ManageEngine. IREO - Mayorista de Soluciones TI* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://www.ireo.com/fabricantes-y-productos/manageengine/manageengine-firewall-analyzer.

ISRAELCCM, 2017. Lenguajes de programación. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://es.ccm.net/contents/304-lenguajes-de-programacion.

ITBLOGSOGETI, 2017. PRUEBAS DE CARGA Y ESTRÉS. *itblogsogeti* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://itblogsogeti.com/2017/03/23/pruebas-de-carga-y-estres/.

JAVA, 2015. ¿Qué es Java? [en línea]. [Consulta: 14 mayo 2018]. Disponible en: https://www.java.com/es/about/whatis_java.jsp?bucket_value=desktop-firefox59-ubuntulinux&in_query=no.

jQuery: Qué es, Orígenes, Ventajas y Desventajas. *Información práctica sobre Redes, Linux, Seguridad y Hacking para profesionales de TI. Capacity Academy* [en línea], 2013. [Consulta: 8 mayo 2018]. Disponible en: http://blog.capacityacademy.com/2013/03/16/jquery-que-es-origenes-ventajas-desventajas/.

JULIAN, P.P. y ANA, G., 2014. Definición de alerta - Qué es, Significado y Concepto. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://definicion.de/alerta/.

KIBERNUM, 2015. ¿Por qué son importantes las pruebas funcionales? *Kibernum Chile* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.kibernum.com/noticias/por-que-son-importantes-las-pruebas-funcionales-2/.

LARGMAN, C., 2011. *UML y Patrones*. Segunda Edicion. Madrid: Pearson Educacion. ISBN ISBN 84-205-3438-2.

MANAGEENGINE, 2017. Firewall Configuration & Log Management by ManageEngine Firewall Analyzer. *ManageEngine Firewall Analyzer* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://www.manageengine.com/products/firewall/.

MANUEL, C., 2013. Diagrama de Componentes. *manuel.cillero.es* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

MARKETING ONLINE, 2016. Metodologías del Desarrollo de Software. *OK HOSTING | Hospedaje Web, Dominios, Desarrollo de Software, Marketing Online, SEO* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://okhosting.com/blog/metodologias-del-desarrollo-de-software/.

MASHITA, 2012. Desarrollo Web y Curiosidades: 10 Ventajas de utilizar Modelo Vista Controlador (MVC) en tus proyectos. *Desarrollo Web y Curiosidades* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://blogdewebin.blogspot.com/2012/04/10-ventajas-de-utilizar-model-vista.html.

MGOEDTEL, 2016. Respuesta a eventos con las alertas de Azure Log Analytics. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://docs.microsoft.com/es-es/azure/log-analytics/log-analytics-tutorial-response.

NETBEANS, 2017. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [en línea]. [Consulta: 21 mayo 2018]. Disponible en: https://netbeans.org/index_es.html.

PEREZ VALDEZ, D., 2007. ¿Qué es Javascript? *Maestros del Web* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.maestrosdelweb.com/que-es-javascript/.

PERRY, joshua, 2015. MODELO DE DOMINIO.pdf. *Scribd* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://es.scribd.com/document/227682505/MODELO-DE-DOMINIO-pdf.

PRESMAN, R.S., 2011. *Ingeniería del software un enfoque práctico*. Septima Edicion. México: Educacion. ISBN ISBN 978-607-15-0314-5.

RODRÍGUEZ SÁNCHEZ, T., 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* 12 noviembre 2014. S.l.: s.n.

SALIXNETWORKS, 2015. Filtrado de Contenido Web - Control de Internet. [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://www.salixnetworks.com/filtrado_web.html.

SARMIENTO, F., Johana, 2013. Visión General de los Diagramas de Despliegue. *UML* [en línea]. [Consulta: 10 mayo 2018]. Disponible en: http://umldiagramadespliegue.blogspot.com/.

SCHNEIER, B. y KELSEY, J., 2015. *Cryptographic Support for Secure Logs on Untrusted Machines*. 2 marzo 2015. S.l.: s.n.

STACKOVERFLOW, 2017. performance - What is considered a good response time for a dynamic, personalized web application? *Stack Overflow* [en línea]. [Consulta: 11 mayo 2018]. Disponible en: https://stackoverflow.com/questions/164175/what-is-considered-a-good-response-time-for-a-dynamic-personalized-web-applicat.

TOLEDO, por D., 2016. JMeter, un viejo amigo en plena forma. *Solidgear* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://solidgeargroup.com/jmeter-un-viejo-amigo-en-plena-forma?lang=es.

Anexos

Anexo 1. Historias de Usuario

Tabla 8 HU_2 Enviar correo electrónico

Historia de Usuario	
Número: HU_2	**Nombre:** Enviar correo electrónico
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 4 horas	**Tiempo real:** 4horas
Descripción: El sistema envía un correo electrónico al administrador con la alerta generada.	
Observaciones:	
Prototipo de interfaz: no aplica	

Tabla 9 Generar alertas basadas en umbral relacionadas con CPU

Historia de Usuario	
Número: HU_3	**Nombre:** Generar alertas basadas en umbral relacionadas con CPU
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 3 horas	**Tiempo real:** 3horas
Descripción: El sistema permite al administrador seleccionar un umbral para generar la alerta del CPU.	
Observaciones: Si el administrador no selecciona ningún valor se establece por defecto 60% como umbral**.**	
Prototipo de interfaz: no aplica	

Tabla 10 HU_4 Generar alertas basadas en umbral relacionadas con HDD.

Historia de Usuario	
Número: HU_4	**Nombre:** Generar alertas basadas en umbral relacionadas con HDD.
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 3 horas	**Tiempo real:** 3horas
Descripción: El sistema permite al administrador seleccionar un umbral para generar la alerta del HDD.	
Observaciones: Si el administrador no selecciona ningún valor se establece por defecto 60% como umbral.	
Prototipo de interfaz: no aplica	

Tabla 11 HU_5 Generar alertas basadas en umbral relacionadas con RAM

Historia de Usuario		
Número: HU_5	**Nombre:** Generar alertas basadas en umbral relacionadas con RAM	
Programador responsable: Henry Ramírez Bullaín		**Iteración asignada:** 1
Prioridad: Alta		
Tiempo estimado: 3horas	**Tiempo real:** 3horas	
Descripción: El sistema permite al administrador seleccionar un umbral para generar la alerta del RAM.		
Observaciones: Si el administrador no selecciona ningún valor se establece por defecto 60% como umbral.		
Prototipo de interfaz:		

Tabla 12 HU_6 Listar alertas enviadas

Historia de Usuario	
Número: HU_6	**Nombre:** Listar alertas enviadas
Programador responsable: Henry Ramírez Bullaín	**Iteración asignada:** 1
Prioridad: Alta	
Tiempo estimado: 1 horas	**Tiempo real:** 1horas
Descripción: El sistema muestra una tabla con las alertas generadas.	
Observaciones:	
Prototipo de interfaz: