UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

Facultad 6

Solución para el chequeo a demanda de parámetros en sistemas de servidores y equipos de interconexión

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

Autor: Renier Ortiz Mondejar

Tutor: Ing. Adrián Misael Peña Montero.

La Habana, Junio de 2013

"Año 55 de la Revolución."

Declaración de Autoría

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

Para que así conste firmo la presente a los___ días del mes de_____ del año_____.

Autor:

_____

Renier Ortiz Mondejar.

Tutor:

_____

Ing. Adrián Misael Peña Montero.

Vivimos en un mundo donde nos escondemos para hacer el amor, mientras la violencia se practica a plena luz del día.

John Lennon.

Datos de contacto

Autor: Renier Ortiz Mondejar.

Universidad de las Ciencias Informáticas, Habana, Cuba.

E-mail: rortiz@estudiantes.uci.cu, sirortiz@gmail.com

Tutor: Ing. Adrián Misael Peña Montero.

Universidad de las Ciencias Informáticas, Habana, Cuba.

E-mail: ampena@uci.cu

Dedicatoria

…A mis seres queridos y en especial a mi madre por haber sido pilar fundamental en mi formación como persona y profesional.

Agradecimientos

Haber llegado hasta aquí y ver cumplido el sueño que tanto hemos deseado, no hubiera sido posible sin el apoyo de todas aquellas personas que desde pequeños y hasta los días de hoy han estado de una forma u otra a nuestro lado, familiares, amigos y profesores.

Todos deben recordar en la niñez la pereza con que nos levantábamos en las mañanas para asistir a la escuela, nunca imaginamos cuán importante sería en nuestras vidas ese 'apúrate que te coge tarde' o 'acábate de levantar y ni un minuto más', ni tampoco valoramos la paciencia de nuestros padres. Es en estos momentos que nos damos cuenta de su importancia. El hecho de verse convertido en un ingeniero útil a nuestra sociedad paga con intereses todos los momentos amargos que se han vivido durante la vida estudiantil. Por ello debemos agradecer a las personas que han hecho esto posible.

A mis profesores a lo largo de todo este tiempo por ser los responsables de mi formación como profesional y haberme inculcado valores para la vida. Agradezco a mis amigos, los cuales siempre han estado ahí para apoyarme en lo que he necesitado, también para compartir las alegrías. En estos dos últimos años he conocido gente excepcional las cuales no puedo dejar pasar por alto, ni mucho menos dejar de mencionarlas. Osmel, Diosbel, Chuchi, Merco, todos los compañeros del edificio 112, Pancho, Aliu, Sergio, Fabián, gracias por tanto conocimiento que me dieron del deporte, aún lo sigo odiando. Ramiro, Jose, Víctor, gracias por todos los momentos alegres y por las sesiones de tesis. En fin a todos estos nuevos amigos que me han demostrado que valió la pena todo el esfuerzo estos dos últimos años.

No puedo dejar de agradecer a mi tutor, en momentos que creía no poder con lo que tenía encima, me animó y encaminó en el trabajo que tenía que hacer. Gracias por todo eso.

A mis hermanos de guerrilla, Adrian, Aleko, Viruto, Elvis, Yelo, Lauren, Yohan, gracias por estar cerca, por estar ahí en los momentos necesarios, gracias por tener el placer de agradecerles.

Agradezco a mi novia, Madelin, fue mucha la paciencia que tuvo sobre todo en estos últimos tiempos. Cualquiera no lo hace, gracias por todo, te quiero.

Agradecimientos

A mi familia, la que siempre me apoyó en mis decisiones fueran buenas o malas, especialmente a mi mamá, esa que sacaba la paciencia de donde no hubiera para poder traerme hasta aquí. La que en los momentos cuando todo parecía que se iba a acabar, me dio ánimos, casi sin tenerlos ella misma, para que todo siguiera sobre ruedas. Gracias por haberme enseñado la necesidad de estudiar. En fin, gracias nuevamente, sin ella nada de esto hubiera sido posible.

Resumen

En la actualidad el control y supervisión de los sistemas encargados del manejo y procesamiento de datos constituye una tarea de vital importancia, evitando así el deterioro progresivo de los datos que son manejados por los mismos. Para poder llevar a cabo estas tareas se hace necesario el uso de un sistema de monitoreo. Estos sistemas de monitoreo se encargan de chequear los parámetros que le son configurados de un equipo determinado, además emiten alarmas en función de los valores obtenido en la tareas de monitorización. Estos sistemas no cuentan con un mecanismo capaz de determinar el origen de un fallo ocurrente en el sistema objeto de monitoreo. En el presente trabajo se propone desarrollar una aplicación que haciendo uso de un sistema de monitoreo ya existente sea capaz de chequear a demanda parámetros a monitorizar en un sistema, con el fin de detectar el origen de una falla. De esta forma se chequen parámetros en los equipos que se monitoricen en función de la falla que se detecte, evitando mantener activos un número elevado de parámetros a monitorizar, reduciendo el consumo de recursos por este concepto en los sistemas objeto de monitoreo. Como resultado se obtuvo una aplicación web que permite realizar un chequeo a demanda de parámetros a monitorizar en sistemas de servidores y equipos de interconexión. Para el desarrollo de esta aplicación se realiza un estudio de algunas de las soluciones existentes a nivel mundial que realicen el proceso de monitoreo de sistemas. Se estudian también los métodos de análisis de datos y los conceptos fundamentales asociados al dominio de la investigación. Se definen las principales herramientas, tecnologías y metodología a utilizar. Se genera la documentación técnica necesaria para el desarrollo de futuras versiones.

PALABRAS CLAVE

Sistema de monitoreo, tareas de monitorización, parámetros, falla, web, servidores.

Abstract

At present, the control and supervision of the systems responsible for the management and processing of data constitutes a task of vital importance, thus avoiding the progressive deterioration of the data that are managed by these systems.

In order to carry out this task, the use of a monitoring system is thus necessary. These monitoring systems are in charge of monitoring the parameters configured from a specific team; in addition, they emit alarms based on the values obtained in the monitoring tasks.

These systems do not have a mechanism capable of determining the origin of a recurring failure in the system, object of monitoring. The present work aims at developing an application which, by making use of an existing monitoring system, is capable of checking at demand parameters to be monitored within a system in order to detect the origin of a failure.

In this way parameters are checked in the equipment monitored, depending on the failure detected, thus preventing a large number of parameters to be monitored from staying active, reducing the consumption of resources by this concept in the systems, object of monitoring. As a result, there was obtained a web application that allows a check at request of parameters to be monitored in systems of servers and networking equipment.

KEY WORDS

Monitoring system, monitoring tasks, parameters, failure, web, servers.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Introducción

El creciente avance en las tecnologías ha permitido un considerable incremento en el desarrollo de la informática y las comunicaciones. Esto ha traído consigo un aumento de la información a procesar por los sistemas informáticos. La gran mayoría de los mismos necesitan de un servidor o dispositivo de interconexión para su buen funcionamiento, llegando a convertirse en una infraestructura de almacenamiento y procesamiento de datos. Hoy en día teniendo en cuenta el amplio desarrollo de estas tecnologías en el mundo, todas las empresas deben estar preparadas para administrar de mejor forma los recursos informáticos encargados del almacenamiento y procesamiento de datos. Esta tarea de administración consiste en mantener bajo un control adecuado todo lo que sucede en los equipos que soportan la gestión de datos, debido a la gran cantidad de información a procesar cada vez se requiere de mayor y mejor prestación de servicios del lado de las computadoras y equipos en función de servidores y agentes de las redes, por lo que la administración de los mismos se torna un tanto engorrosa.

La calidad del procesamiento de datos depende de las prestaciones por parte del hardware encargado, tratándose de capacidad y/o rapidez de procesamiento. Por otra parte están los servicios que pueda brindar el software encargado de las distintas tareas de procesamiento de datos como servidores web, servidores de base de datos, entre otros.

En la mayoría de las ocasiones no basta con tener un solo servidor o determinado equipo de interconexión para la gestión de una tarea, sino que para ello se requiere de un conjunto de sistemas conectados en red. Para su manejo y control se necesita de la mano del hombre. A las personas encargadas de realizar este proceso se les denomina administradores de redes. Ellos son los responsables de controlar, dar seguimiento, mantener y administrar toda la infraestructura tecnológica que soporta la gestión de información de una institución.

La necesidad de mantener periódicamente supervisado y controlar el estado de los sistemas de manera automática, ha provocado el surgimiento de varios mecanismos de monitoreo, con el objetivo de agilizar esta tarea.

Las soluciones de monitoreo más utilizadas en la actualidad están enfocadas a la monitorización de servidores y servicios principalmente. Estas soluciones deben permitir a los administradores realizar la

configuración de los parámetros que se deseen monitorizar, la forma en que van a ser chequeados los mismos, además de esto el sistema debe realizar automáticamente el análisis de los valores de los parámetros para conocer su estado y la generación de alarmas. Cuando un servidor o servicio falla, es tarea del administrador a cargo del mismo detectar qué componente es el que falló. Esto puede resultar un trabajo bastante engorroso cuando el objeto de monitoreo está formado por muchos componentes a la vez.

Cuando se desea detectar específicamente el componente que falla se necesita realizar el monitoreo a un nivel de detalle mucho mayor, lo que llevaría a un aumento significativo en el número de parámetros que es necesario chequear en cada objeto en cuestión. Para implementar esta solución con los sistemas de monitoreo actuales a los que se tiene acceso habría que chequear a la vez todos los parámetros que se necesite verificar, lo que puede aumentar de forma significativa el consumo de recursos por este concepto en los objetos que se monitoricen, provocando que la solución sea poco viable.

Para la detección de fallas de una forma más eficaz se necesita una solución de monitoreo que permita tener configurados todos los parámetros que se desean monitorizar, pero solo activos aquellos indispensables para conocer si el sistema está funcionando correctamente y que los otros puedan ser activados una vez que se necesite. Además, este principio se debe combinar con la organización de los parámetros en una estructura que optimice la búsqueda del componente que ha fallado. Los sistemas de monitoreo actuales a los que se tiene acceso no permiten configurar los parámetros que se desean chequear en los sistemas a monitorear siguiendo ninguno de los principios definidos anteriormente.

En el presente trabajo sólo van a ser abordadas las tareas de configuración y forma de chequeo de los parámetros a monitorear. Esto supone el **problema a resolver:** *¿Cómo realizar un chequeo a demanda de parámetros a monitorizar sobre un sistema de servidores y equipos de interconexión?*

Como **objeto de estudio** se define *el monitoreo de sistemas de servidores y equipos de interconexión.*

Ya definido el objeto de estudio, este queda acotado por el **campo de acción:** *el monitoreo a demanda de parámetros sobre sistemas de servidores y equipos de interconexión.*

Con la finalidad de darle cumplimiento a esta problemática se precisa el siguiente **objetivo general:** *desarrollar una solución que permita chequear el estado de los parámetros a monitorizar en sistemas de servidores y equipos de interconexión,* definiendo **como objetivos específicos:**

- Analizar el proceso de monitorización y chequeo de parámetros sobre sistemas de servidores y equipos de interconexión.
- Diseñar una solución que permita el chequeo a demanda de parámetros a monitorizar sobre sistemas de servidores y equipos de interconexión.
- Implementar una aplicación que permita realizar un chequeo a demanda de los parámetros a monitorear sobre sistemas de servidores y equipos de interconexión.
- Validar de la solución propuesta.

Para dar cumplimiento a los objetivos específicos se proponen las siguientes **tareas de la investigación:**

- Caracterización de las técnicas, métodos y tecnologías empleadas en la actualidad para el monitoreo de sistemas de servidores y equipos de interconexión.
- Análisis de las tecnologías utilizadas para el desarrollo de la solución.
- Identificación de las funcionalidades de la aplicación para el chequeo a demanda de parámetros a monitorizar sobre sistemas de servidores y equipos de interconexión.
- Realización del diseño de un repositorio de scripts de monitoreo que permita organizar el almacenamiento de los mismos.
- Realización del diseño de una aplicación que permita realizar un chequeo a demanda de parámetros a monitorizar sobre sistemas de servidores y equipos de interconexión.
- Implementación de las funcionalidades identificadas para realizar un chequeo a demanda de parámetros a monitorizar sobre sistemas de servidores y equipos de interconexión.
- Selección de las técnicas y métodos para la validación de la solución.
- Validación de la solución propuesta.

Para dar cumplimento a las tareas de la investigación se utilizaron diferentes métodos científicos, los cuales se muestran a continuación.

Métodos teóricos

Analítico Sintético: Se utiliza este método para realizar una búsqueda bibliográfica referente al tema de monitoreo de sistemas y extraer los elementos más importantes de la información consultada, que contribuyan a identificar la existencia de dificultades, carencias o limitaciones en cuanto al monitoreo de sistemas.

Histórico lógico: Utilizado para conocer los antecedentes y elementos de la investigación referidos al proceso de monitoreo de sistemas de servidores y equipos de interconexión, y la determinación de las tendencias actuales sobre la temática que se estudia. Su empleo permitió investigar acerca de los sistemas de monitoreo existentes a los que se tiene acceso para la monitorización de sistemas.

Métodos empíricos

Observación: Se utiliza para identificar las funcionalidades de los sistemas de monitoreo y herramientas de configuración de los mismos, existentes hoy en día y poder definir las características con que debe contar la solución.

Con el propósito de organizar el trabajo y lograr un mejor entendimiento del mismo, se organizó el documento en tres capítulos que recogen toda la información de la investigación realizada, así como información de la implementación de una aplicación para el chequeo a demanda de parámetros a monitorizar sobre sistemas de servidores y equipos de interconexión.

Capítulo I: Fundamentación teórica. Aborda los temas relacionados con el monitoreo de sistemas, así como algunos de los sistemas de monitoreo actuales. Se realiza un estudio sobre la detección de fallas en sistemas y el chequeo de parámetros para la detección, así como la configuración de los mismos para ser monitorizados. Se analizan las distintas metodologías, técnicas y tecnologías a usar en el desarrollo del presente trabajo. Se concluye el capítulo exponiendo los principales criterios acerca del tema estudiado y la selección de software y técnicas y herramientas para el desarrollo de la solución.

Capítulo II: Diseño e implementación de la solución. En este capítulo se hace referencia a las fases de exploración y planificación, propias de la metodología XP utilizadas para el desarrollo del sistema. Se especifican los procesos del negocio que se desean informatizar, así como los aspectos funcionales y no funcionales del software que se propone construir. Además se detallan las Historias de Usuario HU para

establecer luego el orden en que serán implementadas atendiendo a su prioridad. Estas constituyen los primeros pasos de la metodología de desarrollo seleccionada para organizar el ciclo de vida del proyecto.

Capítulo III: Validación de la aplicación. En este capítulo se definen un grupo de normas para la validación de los productos y/o artefactos, logrando que los mismos puedan ser probados para la verificación de su correcto funcionamiento. Se analizan las estrategias de pruebas que definen XP y las técnicas que se utilizarán para diseñar los casos de pruebas que guiarán la validación de la aplicación, además, mostrarán los resultados arrojados por las pruebas efectuadas.

Capítulo # 1 Fundamentación Teórica

Introducción

En los sistemas de servidores a menudo se hace necesario tener el control absoluto sobre los eventos que ocurren. Esto en un sistema sencillo puede parecer de cierto modo simple, pero cuando se cuenta con un sistema complejo, del cual dependen un conjunto de funcionalidades y tareas, se torna más engorroso el hecho de poder controlar todas sus actividades. Para esto surgen los sistemas de monitoreo, estos no son más que herramientas que permite al administrador de redes tener conocimiento de cualquier anomalía en un sistema de servidores y equipos de interconexión [1].

En el presente capítulo se realiza un estudio sobre las distintas formas y sistemas de monitoreo existentes, así como de las principales características de las herramientas para la configuración de los mismos. Además se hace un análisis de algunas de las metodologías y herramientas de desarrollo existentes, identificando la metodología que va a guiar todo el proceso de desarrollo de la aplicación, así como las herramientas empleadas.

1.1. Monitoreo

La detección oportuna de fallas y el monitoreo de los elementos que conforman una red de cómputo son actividades de gran relevancia para brindar un buen servicio a los usuarios que hagan uso de los mismos. De esto se deriva la importancia de contar con un esquema capaz de notificar las fallas de sistemas de servidores y equipos de interconexión y de mostrar su comportamiento. El término de monitoreo de sistemas describe el uso de un sistema que monitoriza una entidad, la misma puede ser un servidor o cualquier equipo de interconexión, en busca de cualquier comportamiento desfavorable en la infraestructura en cuestión. Los recursos usados en un sistema de servidores o equipos de interconexión deben ser verificados continuamente, en busca de cualquier falla que pueda conllevar al deterioro de un recurso específico [2].

Para llevar a cabo el proceso de monitorización de sistemas de servidores y equipos de interconexión mediante un mecanismo de monitoreo se hace uso de una estructura cliente-servidor, en el proceso, los mecanismos de monitoreo realizan tareas de control periódicamente. Se comunican con procesos que

supervisan el sistema operativo, que contiene información sobre el ordenador y para capturar los datos utilizan funciones de bajo nivel.

La monitorización puede ser de dos formas: monitorización remota y monitorización local. La primera se refiere a la monitorización que se realiza desde un servidor de monitoreo hacia uno o varios equipos. O sea, con un mismo mecanismo de monitoreo se pueden chequear muchos sistemas objeto de monitoreo. De esta forma se ejecuta en el servidor en cuestión un mecanismo de monitoreo y éste verifica el estado de los demás equipos. Cabe aclarar que en los equipos que se están monitorizando no se ejecutan tareas de monitorización.

Por otra parte en la monitorización local existe un mecanismo encargado de la tarea por cada equipo objeto de chequeo. Esta modalidad resulta poco efectiva ya que el equipo que se esté verificando debe dedicar parte de sus recursos a las tareas de monitorización, disminuyendo el rendimiento del equipo. Mientras que en la remota sólo el servidor de monitoreo es el encargado de realizar estas tareas, en este caso estos servidores deben ser capaces de brindar altas prestaciones dedicadas a la monitorización de los equipos remotos.

La actividad de monitoreo varía en cuanto a la necesidad de lo que se quiera controlar, pude ser el estado de los servicios de red, procesos de un ordenador o parámetros físicos del hardware. Teniendo en cuenta el propósito para el que se controla, el proceso puede ser visible o no al usuario del objeto de vigilancia.

En el proceso de monitoreo se brindan alarmas de cualquier evento inusual dentro del equipo que se esté monitorizando. Las alarmas son un elemento importante para la detección de problemas en la red. Cuando una alarma ha sido generada, esta debe ser detectada casi en el instante de haber sido emitida para poder atender el problema de una forma inmediata, incluso antes que el usuario del servicio pueda percibirla [3]. Para poder realizar esto se necesita disponer de un sistema de monitoreo, capaz de configurar parámetros a monitorizar y analizar el estado de los mismos, en función de esto debe ser capaz de emitir una alarma en caso de ocurrir alguna eventualidad en el equipo objeto de monitoreo.

1.2. Sistemas para el monitoreo

Tiempo atrás no se disponía de las tecnologías con las que en la actualidad contamos. Por tanto resultaba difícil el proceso de control y monitorización de sistemas de servidores y equipos de interconexión. A causa de esto eran los administradores de redes los responsables de llevar a cabo estas tares manualmente. Esto traía consigo que una parte del día estos sistemas carecieran de supervisión, trayendo problemas a la infraestructura implicada. Precisamente es esta una de las ventajas que traen los diferentes sistemas de monitoreo, que cubren la monitorización de los elementos las veinticuatro horas del día, además de una serie de características preventivas que los distinguen unos de otros.

En un principio, hablar de sistemas de monitoreo de servicios de red en sistemas operativos de red o desktop resultaba casi imposible, ya que no se contaba con las herramientas necesarias para hacerlo. Bastaba con saber que el servidor estaba operativo; y ¿cómo se hacía?, solo con ejecutar el comando ping y la dirección IP del servidor.

Las técnicas de monitoreo de servidores y equipos de interconexión han evolucionado, brindando resultados de control más efectivos que evitan futuros daños. Las técnicas más comunes que se utilizaban para monitorear una infraestructura tecnológica se llevaban a cabo con metodologías ineficientes que no resultaban confiables para el usuario. En cambio, el surgimiento de los sistemas de monitoreo permiten configurar hasta el más mínimo detalle para cumplir con las necesidades ambientales y de seguridad particulares de cada infraestructura.

1.2.1. Funcionalidades básicas de los sistemas de monitoreo

Los sistemas de monitoreo de servidores y equipos de interconexión tienen un conjunto de elementos clave. Asociado a ello cumplen con una serie de características que le permiten ejecutar diversas tareas.

Una de las tareas básicas de los sistemas de monitoreo se centra primeramente en la configuración de los parámetros a monitorizar. En este punto se configuran en el sistema de monitoreo los parámetros que el mismo va a chequear de un equipo determinado. Otra tarea principal es la de chequeo de los parámetros activos. El sistema chequea periódicamente los parámetros que le fueron configurados, verificando el estado de cada uno de los parámetros activos en cuanto a su funcionamiento o bien por umbrales [4].

Un sistema de monitoreo puede estar chequeando una gran cantidad de parámetros a la vez recolectando valores asociados a cada uno de ellos. El análisis de estos valores constituye otra de las tareas principales de los sistemas de monitoreo. Cuando se recibe un valor de un parámetro determinado se analiza y en función del mismo se detecta si el parámetro está en estado crítico, adecuado o fuera de funcionamiento.

De acuerdo al análisis realizado los sistemas son capaces de generar alarmas de cualquier evento inusual y a la vez estas alarmas se notifican al personal competente [5].

1.2.2. Formas de chequeo de parámetros

Las formas más comunes para el chequeo de parámetros son el uso del protocolo SNMP (Simple Network Manager) y la simulación de clientes. El protocolo SNMP permite a los administradores de red administrar dispositivos y diagnosticar problemas en la red. Es en principio, una típica aplicación cliente servidor donde el servidor (el agente SNMP) presenta información acerca de sí mismo en un árbol jerárquico, conteniendo información como, el nombre de la máquina, información acerca de la configuración de sus tarjetas de red, discos duros, etc. [6].

El servidor de SNMP, el SNMPD, usa para la comunicación con el cliente (el llamado manager), el protocolo UPD. El agente envía paquetes llamados "traps", encargados de informar de acontecimientos inusuales en su entorno, ya sea un reinicio del sistema, un exceso de tráfico en la red o un enrutador que deje de responder, por poner algunos ejemplos. Toda la información del agente se guarda en una base de datos denominada Management Information Base (MIB), formando una estructura de árbol y presentada en forma de objetos.

Ahora bien, si esto supone una ventaja para los administradores de redes, también se puede ver como una falla en la seguridad para el equipo que se esté monitorizando ya que esta información puede caer en manos incorrectas y más aún si se tiene la posibilidad de hacer algún cambio en la configuración de los dispositivos SNMP [7]. Para proteger los datos de un agente SNMP es necesario conocer el Comunity String.

La Comnunity String es una especie de contraseña de acceso que se transmite mediante el antes mencionado UDP en texto plano. En un agente SNMP se pueden configurar tantas comunidades como se deseen. Cuando se define una comunidad se puede definir la subred o el host desde el que se va a tener

acceso a esta. Además a las comunidades se le puede especificar los objetos a los cuales va a tener acceso en la MIB y qué permisos va a tener sobre estos objetos, los mismos pueden ser de lectura o escritura [8].

Para la recolección de información a través de este protocolo, se instala en el equipo que se desee monitorear el agente SNMP, el mismo se configura para recopilar la información. La MIB reside en el agente encargado de realizar el monitoreo de los objetos definidos en ella. Desde la estación de monitoreo se accede al agente SNMP a través de los clientes SNMP para obtener información de administración de cualquier dispositivo de red, ya sea switch, router o incluso impresoras u otra PC. Hace una petición de información al agente SNMP (dispositivo donde está instalado y configurado el protocolo), del que logra obtener por ejemplo el uso de CPU, el uso de la memoria, el uso de RAM o el uso del disco duro, etc [9].

Otra forma de recopilar información de dispositivos en la red es la simulación de clientes. Para esto se pueden confeccionar scripts que tengan acceso a dispositivos remotos para obtener información importante para el monitoreo. Los scripts normalmente lo que hacen es simular un cliente y la comunicación entre el agente de monitoreo y el cliente se realiza a través del protocolo que el cliente esté utilizando.

Los scripts para el chequeo de los parámetros básicos en un sistema de servidores generalmente forman parte del paquete del propio sistema de monitoreo, aunque se pueden programar infinidad de ellos en dependencia de los servicios o parámetros que se deseen monitorizar. Para ejecutar estos scripts en un equipo remoto es necesario que se encuentren instalados en el mismo equipo que se va a monitorizar. Este script es el que se encuentra procesándose en todas y cada una de las estaciones que se estén monitorizando mediante esta variante. El mismo se encarga de chequear un parámetro y devolver un valor en función del estado de dicho parámetro.

La responsabilidad de analizar los valores que devuelven los scripts para tomar una decisión no recae en las tareas de monitorización sino en el propio sistema de monitoreo, igualmente el mismo genera una alarma en caso de que los valores obtenidos en el chequeo rebasen los umbrales antes definidos, para los cuales debería estar en correcto funcionamiento el parámetro chequeado [10]. Esta es la forma más empleada para el monitoreo de sistemas, debido a la flexibilidad que brinda el poder diseñar e implementar scripts propios con funcionalidades particulares.

Por otra parte un tema polémico dentro del proceso de monitorización de sistemas sería la detección de fallas. Esto garantiza que el personal encargado de la administración y control de las diferentes infraestructuras pueda llevar un control más estricto sobre los eventos que suelen ocurrir en dichos escenarios. Para dar cumplimiento a esta tarea se realiza un tipo de monitorización en particular, la cual garantiza la integridad de los sistemas en cuestión, detectando qué componente o elemento, objetivo de monitoreo es el responsable de una falla determinada [11].

1.3. Sistemas de detección de fallas

La detección oportuna de fallas en un sistema de servidores y equipos de interconexión ha sido a través de los tiempos una forma de mantener a los mismos en un buen estado operativo. Para ello se hace necesario poder monitorizar cierta cantidad de parámetros en un sistema, en busca de cualquier anomalía que pudiera afectar el buen funcionamiento del sistema en cuestión y por ende el deterioro progresivo de la información que éste maneje.

Una característica de los sistemas formados por componentes que los difiere de los sistemas simples, es la noción para errores parciales. Un error parcial puede ocurrir cuando algún componente del sistema formado por éstos falla, el fallo puede afectar el correcto funcionamiento de algunos componentes, pero a la vez dejar otros sin afectarlos. Por ello la detección y corrección de situaciones anómalas en los servicios que brinda una red de computadoras es una tarea de suma importancia para su buena gestión. Los sistemas de detección de fallas al igual que los sistemas comunes de monitoreo facilitan diagnosticar y reportar cualquier anomalía en sistemas de servidores y equipos de interconexión.

Una característica que los diferencia es que los sistemas de detección de fallas son capaces de identificar el origen de la falla. Las principales fallas de hardware pueden ser fallas de memoria, calentamiento del microprocesador o fallas del disco duro. Mientras que las principales fallas de software pueden ser fallas del sistema operativo, presencia de virus o conflicto entre programas. Para poder efectuar una detección de fallas correctamente es necesario poder verificar en cada momento los componentes que conforman un sistema [12].

Para la detección de fallas es necesario tener identificados todos los componentes de un sistema, permitiendo esto la monitorización de cada uno de ellos, descomponiéndose a la vez en otros

subcomponentes con el fin de encontrar un componente que no pueda ser descompuesto en otros, éste sería el responsable de un fallo en el sistema [13]. En concreto, es esto en lo que consiste la detección de fallas, poder identificar el origen de una falla en sistemas de servidores y equipos de interconexión.

1.3.1. Chequeo de parámetros para la detección de fallas

Para la detección oportuna de fallas en un sistema de servidores y equipos de interconexión, se hace necesario chequear a cada instante sus componentes, tanto de hardware como de software en busca de anomalías en su funcionamiento. Para esto se emplea un sistema de monitoreo al cual se le configuran parámetros a monitorizar. La forma clásica de chequear parámetros en un sistema requiere de la monitorización periódica de los mismos, lo cual a la hora de detectar el origen de una falla en un sistema sería necesario mantener monitorizando a la vez un número elevado de estos parámetros. Esto trae consigo un aumento del consumo de recursos en el equipo objeto de monitoreo, ya que el mismo necesita dedicar una parte de estos a las tareas de monitorización.

Teniendo en cuenta lo antes expuesto, la forma más viable de realizar estas tareas sería activando y desactivando parámetros para monitorizar a medida que se vayan detectando fallas. Esto sería de forma tal que cuando se detecte una anomalía en un componente determinado se desactive el chequeo de los demás componentes y se comience a chequear por las ramas del componente en cuestión. Con esto se garantiza que periódicamente se esté monitorizando un número ínfimo de componentes a diferencia de los métodos tradicionales, los cuales se mantienen monitorizando a la vez todos los componentes del sistema. Es evidente entonces la ventaja que supone el chequeo a demanda frente al monitoreo tradicional.

Para una mejor organización de los componentes frente al chequeo a demanda se necesita ubicarlos en una estructura que permita localizar primero un componente y poder recorrer los demás subcomponentes asociados a él, de forma tal que se pueda tener en una primera visión el componente en sí y luego tomar los subcomponentes como uno nuevo, así sucesivamente hasta que se tenga uno que no contenga más subcomponentes asociados.

Para realizar esto la vía más prudente sería organizarlos en algún tipo de estructura de datos que permita optimizar la búsqueda del componente causante del fallo. Teniendo en cuenta lo antes expuesto la estructura de datos que más se ajusta a la solución es un árbol jerárquico. Este consta de formas de

recorrerlo que garantizan el objetivo que se pretende alcanzar. De los distintos recorridos que se pueden realizar sobre un árbol jerárquico el que más se ajusta a la solución sería un recorrido en anchura-primero donde el árbol es recorrido en orden por nivel (de nivel en nivel), donde se visita cada nodo en un nivel antes de ir a un nivel inferior [14], en este caso un nodo representa un componente.

De esta forma se garantiza que se chequeen los componentes padres y en caso de que uno de ellos presente alguna anomalía se procede a desactivar el chequeo de las demás ramas del árbol recorriendo la rama por la que se detectó la anomalía, repitiendo el proceso en cada uno de las nodos de la rama en cuestión del árbol, en este caso componente [15].

En la aplicación informática que da soporte a la solución propuesta en este trabajo se desea realizar un chequeo a demanda de parámetros en sistemas de servidores y equipos de interconexión. La organización de los componentes para este chequeo se hará de acuerdo a lo antes planteado, organizándolos en un árbol jerárquico y haciendo un recorrido en anchura-primero, para detectar qué componente específicamente es el responsable de una falla.

1.3.2. Configuración de los parámetros para la detección de fallas

Actualmente los métodos que se emplean para la configuración de parámetros para monitorizar un sistema determinado son enfocados al monitoreo del mismo como una entidad. Esto significa que a la hora de realizar las configuraciones para la monitorización de un sistema, se dejan listos una serie de parámetros del mismo y el sistema de monitoreo se encarga de analizar cada uno de ellos periódicamente.

Para el chequeo a demanda de los parámetros que requiere la detección de fallas, éstos deben estar listos pero inactivos, activándolos a la hora de efectuar el chequeo en sí. Es decir, partiendo de un sistema con una serie de parámetros configurados éstos se activan en el momento que se registre alguna eventualidad no constatada dentro de los umbrales predefinidos para los cuales el sistema debería estar funcionando correctamente.

Como se había explicado anteriormente teniendo en cuenta que dichos parámetros se encuentran organizados en forma de árbol jerárquico, si se detecta que en un nodo del árbol no hay ninguna eventualidad este parámetro se desactiva dándole paso al chequeo de los parámetros por las restantes

ramas del árbol. Esto trae como ventaja que no se necesita estar monitorizando una cantidad innecesaria de parámetros a la vez.

La aplicación que se va a desarrollar garantiza que se puedan llevar a cabo las tareas de configuración de los parámetros y organizarlos en la estructura que se menciona. Permite realizar el chequeo de estos parámetros, pero para que esto pueda hacerse se necesita de un motor de monitoreo el cual pueda monitorizar los parámetros activados y emitir alarmas.

1.4. Sistemas de monitoreo existentes

Algunos de los principales y más empleados sistemas de monitoreo son Nagios, OCS Inventory, Munin y Cacti. Aunque muy potentes en las funcionalidades para las que fueron diseñados ninguno de ellos soporta el chequeo a demanda de parámetros, principal motivo este, del trabajo en curso. En la investigación no se pretende realizar una aplicación que realice el monitoreo en su totalidad, solo se encarga de las tareas de chequeo de parámetros, despreocupándose de las funciones de monitorización. Dejándole estas tareas a otro sistema que sirva de apoyo para la solución propuesta. Atendiendo a esto es necesario seleccionar un motor de monitoreo, el cual sea capaz de monitorizar estos parámetros. Además debe poseer características particulares en función de facilitar el chequeo de parámetros a demanda, la organización de los componentes a monitorizar en forma de árbol jerárquico y permita tener configurados una serie de parámetros para su posterior activación o desactivación.

Con el fin de definir qué sistema de monitoreo emplear como motor de monitoreo de la aplicación, se analizan algunos de ellos en función de las características más significativas, que permitan la configuración de parámetros para la monitorización, y su correcta organización a fin de facilitar el chequeo a demanda; además debe brindar la posibilidad de obtener de alguna forma los datos del monitoreo. A continuación se hace un análisis de estos sistemas.

OCSInventory NG, este sistema está liberado bajo licencia GNU/GPL (General Public License) y se encuentra en la categoría de seguridad, redes y sistemas de administración.

Es un sistema diseñado para ayudar al administrador de red a dar de seguimiento de la configuración de las computadoras y el software que se instalan en la red. Surgió en el 2001 y ha sido modificado desde entonces, saliendo nuevas versiones cada cierto tiempo haciendo mejoras a sus funcionalidades.

OCS Inventory NG es capaz de detectar todos los dispositivos activos en la red tales como un servidor, conmutador, enrutador, impresora de red y dispositivos de vigilancia entre otros, para cada uno almacena la dirección, IP y MAC permitiendo clasificarlas. Este utiliza un agente que se encarga del inventario de los equipos cliente y un servidor de administración que centraliza los resultados de los inventarios y permite además visualizarlos. Una de las desventajas que presenta este sistema para el objetivo de esta investigación es el no ser capaz llevar a cabo un monitoreo exhausto de los equipos en cuestión, sino que como ya se ha explicado se encarga solo del inventario de los activos en la red y no de su funcionamiento [16].

Munin, es una herramienta de monitoreo de servidores y hosts pertenecientes a una red de computadoras que genera estadísticas sobre el funcionamiento de los mismos tales como, discos duros, memoria, y servicios que se encuentren en ejecución como un servidor PostgreSQL [17].

Cacti es otra solución completa para la monitorización de redes, diseñada para aprovechar el poder de almacenamiento y la posibilidad de graficar que poseen las RRDTool. Proporciona un esquema rápido de obtención de datos remotos, múltiples métodos de obtención de datos, un manejo avanzado de creación de plantillas, gráficos y una completa interfaz de gestión de usuarios. Además posee un servicio de alarmas mediante el manejo de umbrales, todo ello en una sola consola de administración fácil de configurar [18].

Nagios, es de código abierto, diseñado para monitorear sistemas de servidores y equipos de interconexión. Está implementado en lenguaje C. Asegura que las aplicaciones, servicios y procesos de negocio estén funcionando correctamente.

Nagios cuenta con una diversidad de características que se potencian con complementos como NagVis[1], NDOutils[2], NRPE (Nagios Remote Plugin Executor) y la gran cantidad de proyectos en paralelo que la comunidad Nagios desarrolla.

Este sistema hace uso de scripts para poder realizar el monitoreo de las diferentes estaciones. Los scripts para el monitoreo de los parámetros básicos en un sistema de servidores generalmente forman parte del paquete del propio sistema de monitoreo, aunque se pueden programar infinidad de ellos en dependencia

[1]NagVis no es más que un front-end para crear mapas de nuestra red el cual se vincula a Nagios perfectamente mostrando los hosts o servicios que agreguemos en el mapa en cualquiera de sus estados OK Warning o critical y así nos dará una idea delo que ocurre en nuestra red en tiempo real.
[2]NDOutils es un plugin se encarga de guardar los datos de Nagios en una base de datos mysql

de los servicios o parámetros que se deseen monitorear. Los scripts están escritos en leguaje Perl. Para ejecutar estos scripts en un equipo remoto es necesario que se encuentren instalados en el mismo equipo que se va a monitorear. Nagios simplemente toma el script que necesita y lo ejecuta, brindándole este la información para lo que fue creado [19].

Una de las peculiaridades de Nagios es que permite configurar cada parámetro de forma independiente y además posee mecanismos para almacenar la información que resulta del chequeo en una base de datos MySQL permitiendo tener acceso a ella en cualquier momento que se requiera.

Es válido destacar que de los sistemas analizados Nagios ofrece características y funcionalidades que se adaptan a los objetivos de este trabajo, motivo por el cual se selecciona para el desarrollo del mismo.

1.5. Herramientas para la configuración para sistema de monitoreo Nagios

Debido a la gran cantidad de funcionalidades que nos brindan los sistemas de monitoreo. Teniendo en cuenta que éstos están dirigidos principalmente a usuarios con experiencia en la administración de servidores, o bien en el trabajo con los diferentes equipos de interconexión existente; de los cuales ya se ha hablado en los apartados anteriores; los sistemas de monitoreo se hacen en ocasiones bastante engorrosos a la hora de configurarlos, ya que dependen de una serie de parámetros a tener en cuenta. Además que la configuración se guarda en ficheros de texto, lo cual en más de una ocasión ha resultado ser un contratiempo a la hora de poner en marcha un sistema de monitoreo. Los desarrolladores de software se han planteado diversas alternativas para darle solución a dicha problemática, estas han sido la implementación de herramientas de configuración de sistemas de monitoreo diseñadas de acuerdo a las necesidades de la infraestructura en que se vayan a emplear. Existen muchas de estas herramientas de configuración, cada una responde al sistema de monitoreo para el que haya sido creada. En esta investigación se hará referencia a algunas de las herramientas diseñadas para la configuración y gestión del sistema de monitoreo Nagios.

Aunque funcionales de acuerdo para lo que hayan sido diseñadas estas herramientas, no existe ninguna capaz de realizar una configuración que se ajuste al chequeo de parámetros a monitorizar para la detección de fallas, no soportan ninguna de las tres características que definen esta investigación, el chequeo a demanda de parámetros, organización de los componentes a monitorizar en forma de árbol jerárquico y la configuración de parámetros para su posterior activación o desactivación.

Partiendo de que estas herramientas no soportan estas características, se hace un análisis de las mismas en función de encontrar características que puedan ser de utilidad en el desarrollo de una nueva aplicación que satisfaga las necesidades de esta investigación.

Las herramientas que se analizaron fueron Nconf [20] y Centreon, identificando en esta última características significativas a tener en cuenta a la hora del desarrollo de la aplicación propuesta..

Centreon

Centreon es un paquete de software de código abierto bajo licencia GPL2 que permite supervisar todas las infraestructuras y aplicaciones que componen un sistema de información [21]. Esta es una herramienta que está destinada a la administración y monitorización de redes y dispositivos. Es un front-end para Nagios, es decir que le permite al usuario mediante una interfaz gráfica acceder a los servicios de Nagios de manera más cómoda. Esta herramienta necesita para su instalación de una gran cantidad de paquetes, entre ellos Nagios y con él ciertos plugins para lograr su funcionamiento.

Centreon ya trae por defecto algunas opciones como son:

- Información de rendimiento.
- Gráficas de Nagios.
- Información de procesos.
- Graficas estado del servidor.
- Generación de plantillas de configuración.

Permite la creación de usuarios y grupos de usuarios, así como hosts y grupos de éstos, también se pueden crear servicios, estos, son las órdenes que damos para que cada cierto tiempo se vayan haciendo comprobaciones en los hosts escogidos. Esto servicios se pueden aplicar a un host o grupo de ellos. Lo difícil de estos servicios es que son complicados de aprender a usar, principalmente porque se configuran mediante el uso de comandos [22].

Una desventaja de Centreon es que posee muy poca documentación y que muchos módulos son de pagos [23].

Partiendo de lo antes expuesto, se identificaron una serie de características en estas dos herramientas que son de utilidad en la aplicación que se pretende desarrollar, una de ellas es la implementación de la aplicación para la web. Se tiene en cuenta la generación de plantillas con que cuenta Centreon, siendo esta una de las características de relevancia para la aplicación motivo de este trabajo. También se tiene en cuenta la creación de usuario y grupos de usuarios permitiendo que en la aplicación que se desea desarrollar se tenga un control estricto del personal encargado de operar con la misma. Así mismo se analiza la posibilidad de la implementación de la aplicación basado en una aplicación web y sus peculiaridades que le dan ventaja sobre otro tipo de aplicaciones en este caso.

1.6. Desarrollo de aplicaciones web

En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador. La aplicación web contiene un conjunto de páginas estáticas y dinámicas. Una página web estática es aquella que no cambia cuando el usuario la solicita: el servidor web envía la página al navegador solicitante sin modificarla. Por el contrario, el servidor modifica las páginas dinámicas antes de enviarlas al navegador. Esta naturaleza cambiante es la que le da nombre de dinámica [1].

En cuanto al desarrollo de aplicaciones web existen dos grupos de lenguajes de programación para implementar una web, clasificándose en dependencia de dónde se implementan siguiendo la arquitectura cliente-servidor. De esta forma a un grupo se le llama lenguaje del lado de cliente y a otro, del lado del servidor. En el lado del cliente se encuentran Java Script, HTML, visual BASIC script y en el lado del servidor PERL, ASP, JSP, Java, aunque existen otras muchas sólo se hace alusión a las más empleadas.

Algunas de las ventajas que brinda el desarrollo de aplicaciones web es que poseen la característica de ser fácilmente accesibles, pudiendo acceder a ellas desde cualquier lugar que se cuente con una conexión a

internet o intranet y un navegador web, no necesita que se instale nada ni depende de ningún software en su mayoría.

1.6.1. Utilización de portales web como base para el desarrollo de aplicaciones web

En la actualidad el desarrollo de aplicaciones web se hace mucho más fácil y flexible gracias a las distintas tecnologías de gestión de contenido, como portales web, CMS (Content Management System – Sistemas de Gestión de Contenidos), entre otras. Estas agilizan el tiempo de desarrollo y garantizan la calidad del producto. Al tener una serie de módulos y componentes ya programados, no es necesario volver a escribir código para realizar funciones comunes y necesarias dentro de una aplicación web, como por ejemplo, el proceso de gestión de usuario y control de acceso.

En los últimos tiempos en el ambiente de Tecnología Web, surgió un término: portal. En general un portal web no es más que un Sitio Web que sirve de punto de partida para desarrollar aplicaciones de fácil acceso a través de internet o intranet [24]. Es un tipo de sitio de mayor tamaño, que puede pertenecer a un proveedor de internet o a una empresa.

De los Sistemas de Gestión de Contenidos se destacan por su amplia utilización y características particulares de cada uno de ellos en la elaboración de aplicaciones web, que van desde gestión empresarial, pasando por redes sociales hasta comercio electrónico, están Joomla, Drupal, Liferay, entre muchas otras.

Joomla: Es un sistema de gestión de contenido que permite construir potentes sitios web y aplicaciones en línea. Muchos aspectos, incluyendo su facilidad de uso y extensibilidad han hecho de Joomla uno de los software de construcción de sitios web disponibles más populares. Es una solución de código abierto disponible de forma gratuita para todo el mundo. Está programado mayoritariamente en lenguaje php bajo licencia GPL, con soporte para base de datos PostgreSQL y MySQL. Requiere preferentemente de un servidor HTTP Apache [25].

Drupal: Es un sistema de gestión de contenido modular y muy configurable.
Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitios web [26].

Liferay: Este posee un buen diseño arquitectónico basado en las mejores prácticas de J2EE que le permite ser utilizado con una gran variedad de servidores de aplicación, ejemplo Tomcat.

Liferay posee características de relevancia que lo sitúan en un lugar de alta preferencia por la mayoría de los desarrolladores de aplicaciones java. Este corre bajo la mayor parte de servidores de aplicaciones y contenedores de servlets, bases de datos, sistemas operativos y provee más de 700 combinaciones de despliegue. Posee una ventaja para la comunicación con sistemas externos ya que utiliza un framework abierto basado en arquitectura orientada a servicios. Esto le brinda una gran oportunidad a la solución que se desea desarrollar [27]. Liferay posee un amplio y completo mecanismo de gestión de usuarios basado en roles, lo cual permite mantener un control más estricto y garantizar la seguridad dentro de la aplicación, impidiendo que personal no autorizado pueda hacer uso de la misma, ya que esta maneja información sensible que puede afectar el correcto funcionamiento de los sistemas involucrados debido a un mal manejo de la misma. Liferay provee una alternativa viable de portal para el desarrollo de aplicaciones basadas en portlet [28].

Basándose en las características con que debe contar la solución propuesta, se propone usar Liferay como portal Web para el desarrollo de la aplicación.

1.6.2. Desarrollo basado en portlets

Atendiendo a que se va a utilizar Liferay para el desarrollo de la aplicación, y que el mismo ofrece un soporte significativo para el desarrollo basado en portlets, se decide desarrollar la aplicación en forma de estos.

Un portlet o colección de porlets se asemeja a una aplicación web que esta hospedada en un portal. Estos son manejados por un contenedor de portlets (liferay en este caso) el cual procesa peticiones y genera contenido dinámico, y son usados por los portales como un componente conectable de interfaz de usuario que posee una capa de presentación a un sistema informático [27].

Algunas características relevantes de los portlets, como que producen fragmentos de código de marcado que agregan a una página de un portal y que siguen típicamente la metáfora de escritorio, tributan a su selección ya que la aplicación que se propone desarrollar más que una aplicación comercial está orientada a la gestión y configuración de sistemas, buscando poseer una interfaz algo similar a como sería en una aplicación de escritorio.

1.7. Metodologías y herramientas propuestas

Las metodologías de desarrollo de software definen un conjunto de procedimientos, técnicas y ayuda a la documentación para el desarrollo de software. Una metodología organiza durante el proceso de desarrollo las actividades que se deben realizar para lograr la obtención de un producto final, definiendo las personas que participan, sus responsabilidades, cuándo y cómo cumplirlas. A continuación se hará una breve explicación de cuales se emplearon y en qué consisten.

1.7.1. Metodología de desarrollo

El proceso de desarrollo de este trabajo consta de un tiempo limitado para su culminación, además, el equipo de desarrollo se conforma por una sola persona, en constante intercambio con el cliente, permitiendo la retroalimentación de ambas partes. La solución final se considera innovadora ya que no existe un sistema único capaz de realizar las tareas del mismo, de manera automática. Durante el proceso se busca lograr un producto funcional más que una extensa documentación y los requisitos del sistema pueden estar sujetos a cambios, propiciado por la misma interacción cliente-desarrollador.

Atendiendo a lo antes expuesto y analizando algunas de las metodologías de desarrollo más utilizadas en los últimos tiempos, como SCRUM [29], RUP [30], (de las robustas) y Cristal [31], se decide emplear la metodología XP (Extreme Programming) como metodología de desarrollo para guiar el proceso. Esta al ser una metodología ágiles, centrada a potenciar las relaciones interpersonales como clave para el éxito del desarrollo de software; que se preocupa por el aprendizaje de los desarrolladores y propicia un buen clima de trabajo; que genera poca documentación y que agiliza el proceso de desarrollo [32], se adapta potencialmente a las circunstancias bajo las cuales se establece el desempeño de la solución propuesta. Por lo que se escoge como la metodología rectora del proceso de desarrollo de este trabajo.

1.7.2. Lenguaje de modelado

Aunque la metodología que se emplea en el desarrollo de este trabajo propone que todo proyecto que se realice tenga un diseño lo más sencillo posible, por lo que no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML (Lenguaje Unificado de Modelado, de sus siglas en inglés) [33], sí se hace uso del mismo, así como de un diagrama de modelo de dominio para lograr un mejor entendimiento de la solución. En la realización de los mismos se busca llegar a una notación sencilla y de fácil entendimiento, además de que a través del ciclo de desarrollo se está en constante intercambio con el cliente y que las funcionalidades del proyecto suelen ser cambiantes, se opta por la utilización de UML de sus siglas en inglés, como lenguaje de modelado de los diagramas pertinentes. Permitiendo darle una solución sencilla al desarrollo del trabajo.

El lenguaje unificado de modelado UML es el sucesor de la oleada de métodos de análisis y diseño orientado a objetos que surgió a finales de la década del 80 y principios de la siguiente. Decimos pues que el UML es un lenguaje de modelado y no un método. La mayor parte de los métodos consisten, al menos en principio, en un lenguaje y un proceso para modelar. El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar los diseños. El proceso es la orientación que nos dan sobre los pasos a seguir para realizar el diseño [34].

1.7.3. Herramientas de modelado

Debido a la poca documentación que genera la metodología de desarrollo XP, y a que el tiempo de desarrollo es limitado se optó por seleccionar una herramienta de modelado de fácil comprensión y grandes utilidades. Dentro de la suite de herramientas case para el modelado de diagramas UML existen varias herramientas dotadas de módulos que facilitan el trabajo durante la confección de un software. Todas poseen un lenguaje estándar. Teniendo en cuenta que se propone la utilización de herramientas libres, se optó por emplear **Visual Paradigm** como herramienta de modelado, el mismo soporta UML como lenguaje unificado de modelado en la confección de los diferentes diagramas y esquemas que acompañarán el desarrollo de la solución propuesta.

1.7.4. Lenguajes de programación

En epígrafes anteriores se abordaron temas referentes al desarrollo de aplicaciones web. En el mismo, se llegó a la conclusión de emplear Liferay como portal web para el despliegue de la solución propuesta. Teniendo en cuenta que este portal está desarrollado en java y que lo componen una serie de portlets, y que estos, se programan en java, se decidió optar por Java como lenguaje de programación para el desarrollo de la solución.

Java es un lenguaje de programación puramente orientado a objeto que está diseñado tanto para aplicaciones Web como de escritorio. El mismo fue desarrollado por la compañía Sun Microsystems en 1995, la cual desde el 2009 fue comprada por la compañía ORACLE, con el propósito de cubrir las necesidades tecnológicas de punta. Su principal característica y a su vez lo que lo diferencia ampliamente de los demás lenguajes de programación es la independencia que tiene a nivel de plataforma; esto se debe a que se le ha creado una máquina virtual para cada sistema operativo (SO). Cualquier aplicación que se desee hacer con acceso a través de la Web se puede hacer utilizando Java. Este lenguaje incorpora muchos aspectos que en cualquier otro son extensiones, propiedad de empresas de software o fabricantes de ordenadores, threads (hilos de procesamiento), ejecución remota, componentes, seguridad, acceso a bases de datos, etc. Por eso muchos expertos opinan que Java es el lenguaje ideal para aprender la informática moderna[35].

1.7.5. Sistemas de Gestión de Bases de Datos

PostgreSQL es un sistema de gestión de bases de datos que incorpora el modelo relacional para sus bases de datos y usa el lenguaje SQL como lenguaje de consulta. El sistema de gestión de bases de datos PostgreSQL, es una de las aplicaciones de código abierto con más éxito de los últimos años, seguido por muchos desarrolladores y usuarios. Es una buena herramienta para crear una aplicación con grandes cantidades de información. PostgreSQL es una excelente implementación de una base de datos relacional, con todo tipo de funcionalidades, de código abierto y de uso gratuito. Es una aplicación multiplataforma que cuenta con una amplia documentación y una gran comunidad de usuarios y desarrolladores. Altamente configurable y adaptable a las necesidades de la solución propuesta [36].

PostgreSQL puede ser usado desde cualquiera de los lenguajes de programación más comunes (C, C++, Perl, Python, Java, Tcl, PHP). Sigue muy de cerca los estándares de lenguajes de consulta (SQL: 2008), y sigue desarrollando estándares [37].

1.7.6. Entorno de desarrollo integrado (IDE)

Teniendo en cuenta lo antes expuesto con respecto al lenguaje de desarrollo a emplear, el tipo de aplicación que sería de tipo web y en la búsqueda de facilidades para la programación, como son, completamiento de código, fácil uso de las librerías disponibles e integración con el portal web Liferay, se optó por emplear el IDE Eclipse Índigo come entorno de desarrollo integrado. Este potente IDE cuenta con una integración de Liferay para su implementación, lo que facilita en gran medida el desarrollo de una aplicación para este portal web.

Eclipse es una plataforma abierta para el desarrollo de aplicaciones que corren sobre un amplio rango de sistemas operativos. Brinda la facilidad de asociar disímiles editores para cada tipo de ficheros además de posibilitar editarlos con los programas asociados por el sistema operativo. La arquitectura de plugins de Eclipse permite además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros. Existen versiones de Eclipse instalables para cualquier plataforma que incluyen el código fuente y los plugins más habituales. Una de las características más curiosas del IDE Eclipse es el modo en que se compilan los proyectos. La compilación es una tarea que se lanza automáticamente al guardar los cambios realizados en el código [38].

1.8. Conclusiones del capítulo

En el presente capítulo se detallaron las condiciones y problemas actuales que rodean el objeto de estudio, enmarcado en el monitoreo de sistemas de servidores y equipos de interconexión. Se indagó cómo se lleva a cabo el monitoreo de sistemas así como sus funcionalidades básicas, delimitando conceptos básicos como la detección de fallas en sistemas de servidores y equipos de interconexión, el chequeo a demanda de parámetros para la detección de fallas, configuración de parámetros para el chequeo a demanda de parámetros, y las formas de chequeo de los mismos. Se realizó un análisis de las características de la solución y la necesidad de contar con un motor de monitoreo para las tareas de monitorización de los

parámetros, identificando a Nagios como motor de monitoreo más adecuado. Se definió la metodología de desarrollo a utilizar, en este caso XP ajustándose a las peculiaridades de la solución propuesta, lo que la hacen idónea para guiar todo el flujo de trabajo en el desarrollo de la solución. Se definieron tecnologías y herramientas para el desarrollo. Luego de un estudio de las principales tecnologías para el desarrollo de aplicaciones para la web se seleccionó Liferay como portal web y el desarrollo de la aplicación basado en portlets debido al soporte que brinda Liferay para esto. Se concluye que es necesaria la implementación de un sistema de chequeo a demanda de parámetros a monitorizar en sistemas, con el fin de detectar el origen de fallas en el mismo.

Capítulo # 2 Diseño e implementación de la solución

Introducción

En este capítulo se hace referencia a las fases de exploración y planificación, propias de la metodología XP utilizada para guiar el proceso de desarrollo de la solución. Se especifican los procesos del negocio que se desean informatizar, así como los aspectos funcionales y no funcionales del software que se propone construir. Además se detallan las Historias de Usuario (HU) para establecer luego el orden del que serán implementadas atendiendo a su prioridad. Estas constituyen los primeros pasos de la metodología de desarrollo seleccionada para organizar el ciclo de vida del proyecto.

2.1. Descripción de la solución

Atendiendo a la amplia gama de servicios y mejoras que en la actualidad ofrecen los sistemas de servidores y equipos de interconexión, sobre todo en infraestructuras de gestión de datos, las cuales pueden llegar a ser de gran tamaño, se crean mecanismos para el control y seguimiento del estado de estos sistemas, con el objetivo de garantizar la integridad de los datos que los mismos manejan. Este proceso de control y seguimiento se traduce en monitoreo de servidores y equipos de interconexión. Para realizar esta tarea los sistemas de monitoreo chequean a cada instante el estado de los diferentes componentes para los cuales hayan sido configurado, notificando la ocurrencia de cualquier anomalía.

El sistema que se propone implementar hace uso de una de estas herramientas de monitoreo, específicamente Nagios, para llevar a cabo un chequeo a demanda de parámetros a monitorizar. Para cumplimentar esta tarea se hace uso de un repositorio de scripts y de una aplicación para la configuración de plantillas de monitoreo, la misma debe permitir activar o desactivar la monitorización de parámetros en función del chequeo a demanda. El repositorio de scripts se emplea para tener organizados los scripts en un sistema de directorios y así facilitar la búsqueda de los mismos, en función de lo que se desee monitorizar.

2.1.1. Repositorio para el almacenamiento de scripts

El sistema debe ser capaz de gestionar un repositorio de scripts, el cual va a estar organizado por categorías en dependencia de la función de cada script, estas están representadas en un sistema de directorios, el repositorio cuenta con un directorio raíz que contiene un primer nivel con otros directorios correspondientes a cada categoría. En una base de datos se almacenarán los metadatos de estos scripts, estos metadatos contienen una descripción detallada del script, así como la categoría a la que pertenece. Esto va a permitir la búsqueda de los scripts y escoger de entre ellos el que más se ajuste a las necesidades del monitoreo.

Categorías

Las categorías serán las encargadas de conformar la estructura que contendrá unívocamente los scripts que hayan sido gestionados en ella. Las categorías están conformadas de acuerdo a las funcionalidades para las que estén programados los scripts que se encuentren dentro de ella, las mismas pueden ser de hardware, de software entre otras. Al cargar un nuevo script en la aplicación deberá especificarse la categoría a la que pertenece, ubicándolo en la estructura de árbol adecuada para la funcionalidad a la que este responda.

Vale aclarar que la gestión del repositorio se hará desde la propia aplicación, esto incluye la creación, modificación y eliminación de categorías, así como de scripts.

2.1.2. Aplicación para la configuración de Nagios

La aplicación para el chequeo a demanda de parámetros a monitorizar en sistemas de servidores y equipos de interconexión, cuyo objetivo principal está enfocado a la detección de fallas, identificando el origen de una falla determinada en un sistema, cuenta con una serie de características que permiten desempeñar las funcionalidades para las que ha sido concebida. Para realizar este chequeo es necesario hacer una serie de configuraciones y la creación de plantillas de configuración, esto lo hace el administrador de sistemas encargado de operar con la aplicación, con el objetivo de garantizar una justa organización y rapidez a la hora de tener una serie de parámetros configurados y listos para ser monitorizados. Para garantizar estas tareas la aplicación se divide en varios módulos en los cuáles están agrupados todas las funcionalidades de la aplicación.

El módulo denominado **Módulo de Seguridad** será el encargado de controlar la autenticación de los usuarios en el sistema así como la gestión de usuarios, roles y niveles de acceso.

Para la autenticación el sistema debe:

- Permitir la autenticación en la aplicación mediante usuario y contraseña.

Para la gestión de los usuarios la aplicación debe ser capaz de:

- Añadir nuevos usuarios
- Actualizar los datos de los usuarios existentes.
- Eliminar usuarios existentes.

Para la gestión de roles la aplicación debe poder:

- Adicionar nuevos roles de usuario
- Actualizar los datos de los roles existentes
- Eliminar un determinado rol del sistema.

El módulo denominado **Módulo de Gestión de Componentes** de configuración es el encargado de la creación, modificación y eliminación de plantillas de configuración. Estas plantillas no van a ser más que una configuración genérica para monitorizar un componente determinado de un sistema. Para dar cumplimiento a esta funcionalidad la aplicación debe ser capaz de:

- Configurar la monitorización de un componente: Un componente dentro de un sistema de servidores o equipo de interconexión es aquel que se pueda monitorizar de forma global, como un servidor web o un servicio determinado. Estos componentes cuentan con otros parámetros que permiten conocer su estado en detalle.
- Configurar los parámetros asociados a un componente: Estos parámetros pueden formar otros componentes si de ellos se derivan otros parámetros más específicos.
- Determinar el método por el cual se va a realizar la monitorización. Para ello la aplicación debe permitir seleccionar el método.
- En caso de que sea mediante script o mediante NRPE, seleccionar de entre el repositorio de scripts el que se ajusta a las necesidades requeridas.

- Si se desea realizar el monitoreo por SNMP una aplicación externa debe brindar la información de los equipos que estén listos para ser monitoreados por SNMP, así como los componentes que puedan ser monitorizados de cada equipo.
- Probar que están bien hechas las configuraciones y guardar la plantilla en una base de datos PostgreSQL para su reutilización en próximos sistemas. Estas plantillas se le aplican a un sistema y se comprueba mediante un comando Nagios que están bien hechas las configuraciones y se activa por un tiempo breve la monitorización del mismo.

El módulo denominado **Gestión de Sistemas** es el encargado de aplicar una plantilla de configuración a un sistema determinado, para ello la aplicación debe poder:

- Visualizar los sistemas que están listos para añadirle una plantilla de configuración.
- Seleccionar de entre los sistemas y añadirle la plantilla.
- Editar la plantilla en caso de que requiera algún cambio para que se adapte al sistema en cuestión.
- Agregar datos de conexión como dirección ip, nombre dns, sistema operativo y descripción del sistema.
- Probar que estén bien hechas las configuraciones y activar la monitorización por un breve tiempo a fin de comprobar que se ha aplicado con éxito la plantilla de configuración.
- El sistema queda configurado y listo para ser monitorizado pero inactivo a la espera de la activación de los parámetros a monitorizar.

El módulo denominado **Gestión de activación y desactivación de los parámetros a monitorizar** debe ser capaz luego de tener una serie de sistemas configurados en el servidor de monitoreo y listos para ser monitorizados, de brindar una interfaz de comunicación con sistemas externos con un listado de los sistemas y los parámetros de cada uno en específico. De estos sistemas la aplicación debe activar o desactivar el monitoreo de parámetros de acuerdo a órdenes externas. Para ello la aplicación debe poder.

- Mediante servicios Rest [39] mostrar un listado de los sistemas listos para su monitorización.
- Activar el monitoreo de parámetros del sistema. En este caso se agrega un parámetro que va a decir qué tiempo va a estar activo el monitoreo de dichos parámetros.
- Desactivar el monitoreo de parámetros del sistema.

La aplicación debe contar con una interfaz capaz de brindar la información de monitoreo de un sistema o parámetro específico. Para ello debe ser capaz de:

- Establecer una comunicación con el sistema externo, esta se realiza mediante servicios Rest.
- Transferir datos de la aplicación al sistema externo en formato de XML.

2.2. Diseño de la aplicación

Se propone realizar una aplicación capaz de realizar un chequeo a demanda de parámetros a monitorizar de un sistema de servidores y equipos de interconexión con el objetivo de identificar el origen de una en los mismos. El sistema debe ser capaz de activar o desactivar parámetros a monitorizar de acuerdo a órdenes provenientes de un sistema externo.

2.2.1. Historias de usuarios

Una de las mejores prácticas adoptadas en el desarrollo de software es la administración de requerimientos. XP propone en este sentido hacer uso de las historias de usuario como técnicas para especificar las funcionalidades que brindará el sistema y constituye una manera muy dinámica de realizar una actividad [40]. Como su nombre lo indica son especificadas por el propio usuario y por tanto redactadas en su lenguaje; de manera sencilla y breve, evitando tecnicismos innecesarios que puedan complicar su comprensión, aunque los programadores pueden contribuir en la tarea. Además son la base para realizar las pruebas de aceptación, así como la estimación y planificación del proyecto [40].

Una vez identificadas las historias de usuario necesarias para liberar una primera versión operativa los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente, que darán solución a la historia correspondiente. Para definir cada HU se emplea una planilla, esta contiene todos los datos necesarios para desarrollar la funcionalidad descrita.

En el trabajo en curso se identificaron 9 HU, las mismas se listan a continuación y se muestra un ejemplo de planilla para HU.

Historias de usuarios

1. Gestionar repositorio de Scripts
2. Gestionar componentes
3. Gestionar equipos
4. Listar sistemas listos para monitorizar
5. Activar parámetros de monitoreo
6. Desactivar parámetros de monitoreo
7. Enviar estado del monitoreo

En la Tabla 1: HU Gestionar repositorio de Scripts se describe como se compone una HU y que significa cada campo de la misma.

Tabla 1: HU Gestionar repositorio de Scripts

Historia de Usuario		
Nombre: Gestionar repositorio de scripts.	**Puntos de estimación:** 3.	
No: 3.	**Usuario:** Usuario con permisos en la aplicación para ejecutar esta tarea.	**Iteración:** 1.
Descripción: El usuario debe cargar un script determinado en el repositorio indicando la categoría a la que pertenece así como los parámetros específicos que este recibe a la hora de ponerlo en función del monitoreo.		
Prioridad: Alta.	**Nivel de complejidad:** 4.	

Información adicional: Esta tarea es de vital importancia dentro de la aplicación ya que en fases posteriores se emplean los scripts para la configuración de los parámetros que se van a monitorizar, los scripts deben ser de fácil acceso dentro del repositorio.

2.2.2. Lista de reserva del producto

La lista de reserva del producto es una tabla que contiene los requisitos funcionales que debe cumplir la aplicación que se desea realizar, ordenado según la prioridad de implementación, catalogados en Muy Alta, Alta, Media y Baja. En esta última aparecen los requisitos de menor complejidad, además de los requisitos no funcionales del sistema a desarrollar. Indica la estimación de cada uno de ellos y su implementación por semana, además de quién hizo la estimación.

Tabla 2: Lista de reserva del producto

Ítem*	Descripción	Estimación	Estimado por
Requisitos funcionales			
Prioridad: Muy alta			

1	Adicionar script	0.5	Desarrollador
2	Modificar script	0.5	Desarrollador
3	Eliminar script	0.5	Desarrollador
4	Buscar script	0.5	Desarrollador
5	Crear componente	0.5	Desarrollador
6	Modificar componente	0.5	Desarrollador
7	Eliminar componente	0.5	Desarrollador

Prioridad: Alta			
14	Adicionar equipo	1	Desarrollador
15	Asignar componente a equipo	0.5	Desarrollador
16	Eliminar sistema	0.5	Desarrollador
Prioridad: Media			
17	Activar componente	1	Desarrollador
18	Desactivar componente	1	Desarrollador
19	Listar sistemas listos	0.5	Desarrollador
20	Listar parámetros	0.5	Desarrollador
21	Brindar estado de los parámetros	0.5	Desarrollador
Requisitos no funcionales			
Prioridad: Baja			
1	**Apariencia o interfaz externa:** La herramienta contará con una interfaz amigable, que brinde facilidades al usuario para propiciar un correcto uso de la misma y una mejor comprensión por parte de éste, con una navegabilidad flexible y de fácil comprensión.		

2	**Software:** Se implementará una herramienta que utilizará sistema operativo Linux, SGBD PostgreSQL 9.1, TOMCAT en su versión 7.0.27 como servidor de aplicaciones. Necesita de un servidor de monitoreo con Nagios instalado como sistema de monitoreo.
3	**Hardware:** Se necesita 2 GB de memoria RAM como mínimo, 512 MB de espacio libre en el disco duro para su instalación y un microprocesador a 2.40 GHz.
4	**Usabilidad:** Cualquier usuario con conocimientos de básicos en tareas de administración de redes podrá hacer uso de la aplicación.
5	**Rendimiento:** El sistema que se propone debe ser ágil en la obtención de datos de los parámetros monitoreados.
6	**Confidencialidad:** Sólo tendrán acceso a la aplicación los usuarios registrados con permisos de administración de servidores y equipos de interconexión.
7	**Restricciones del diseño y la implementación:** Para el diseño e implementación de la aplicación se utiliza la metodología XP, haciendo uso del Visual Paradigm 8.0 para el modelado. Se utiliza como lenguaje de programación Java, como gestores de BD PostgreSQL 9.1, y como IDE de desarrollo Eclipse Índigo, optimizado para Liferay.

Lugo de definidas las HU e identificados los requisitos funcionales y no funcionales, recogidos en la Tabla 2: Lista de reserva del producto, se hace necesario crear un plan de iteraciones.

2.2.3. Plan de iteraciones

El plan de iteraciones especifica las historias de usuario desarrolladas en cada iteración del proceso de implementación en el orden preestablecido y las fechas en que serán liberadas [32]. Para el desarrollo del

sistema propuesto de se han definido 4 iteraciones, las cuales se describen en la Tabla 3: Plan de Iteraciones.

Tabla 3: Plan de Iteraciones

Iteraciones	Descripción de la iteración
Iteración 1	La primera iteración del desarrollo del sistema va a tener como objetivo la implementación de las funcionalidades básicas o prioridad baja para la puesta en marcha del mismo. Estas tributan a la HU 1. Concluida esta iteración, será posible gestionar el repositorio de scripts.
Iteración 2	En esta iteración está centrada la implementación de las HU de prioridad media para el sistema, estas son las número 2 y 7. Al finalizar esta iteración se podrá gestionar un componente y enviar el estado del chequeo.
Iteración 3	En esta iteración es implementan las HU 3 y 4. Al finalizar esta iteración el sistema podrá gestionar la configuración de un sistema y listar los sistemas preparados para chequear sus parámetros.
Iteración 4	En esta iteración se lleva a cabo las funcionalidades más importantes del sistema, estas se recogen en las HU de prioridad muy alta, estas son las 5 y 6. Con la culminación de esta iteración de complementan las funcionalidades para las que fue diseñado el sistema, pudiendo activar o desactivar los parámetros a chequear.

2.2.4. Tarjetas CRC

El uso de tarjetas CRC permite al programador centrarse y apreciar el desarrollo orientado a objetos.

Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en la columna de la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad [41].

En la Tabla 4: Tarjeta CRC Gestionar Script se muestra un ejemplo de tarjeta CRC generadas para el diseño de la aplicación.

Tabla 4: Tarjeta CRC Gestionar Script

Tarjeta CRC	
Clase: GestionarScript	
Responsabilidades	**Colaboraciones**
Adicionar Modificar Eliminar Buscar	Script

Tabla 5: Tarjeta CRC Gestionar plantilla

Tarjeta CRC	
Clase: GestionarComponente	
Responsabilidades	**Colaboraciones**
Adicionar	Plantilla

Modificar Eliminar Buscar	

2.2.5. Diagrama de clases

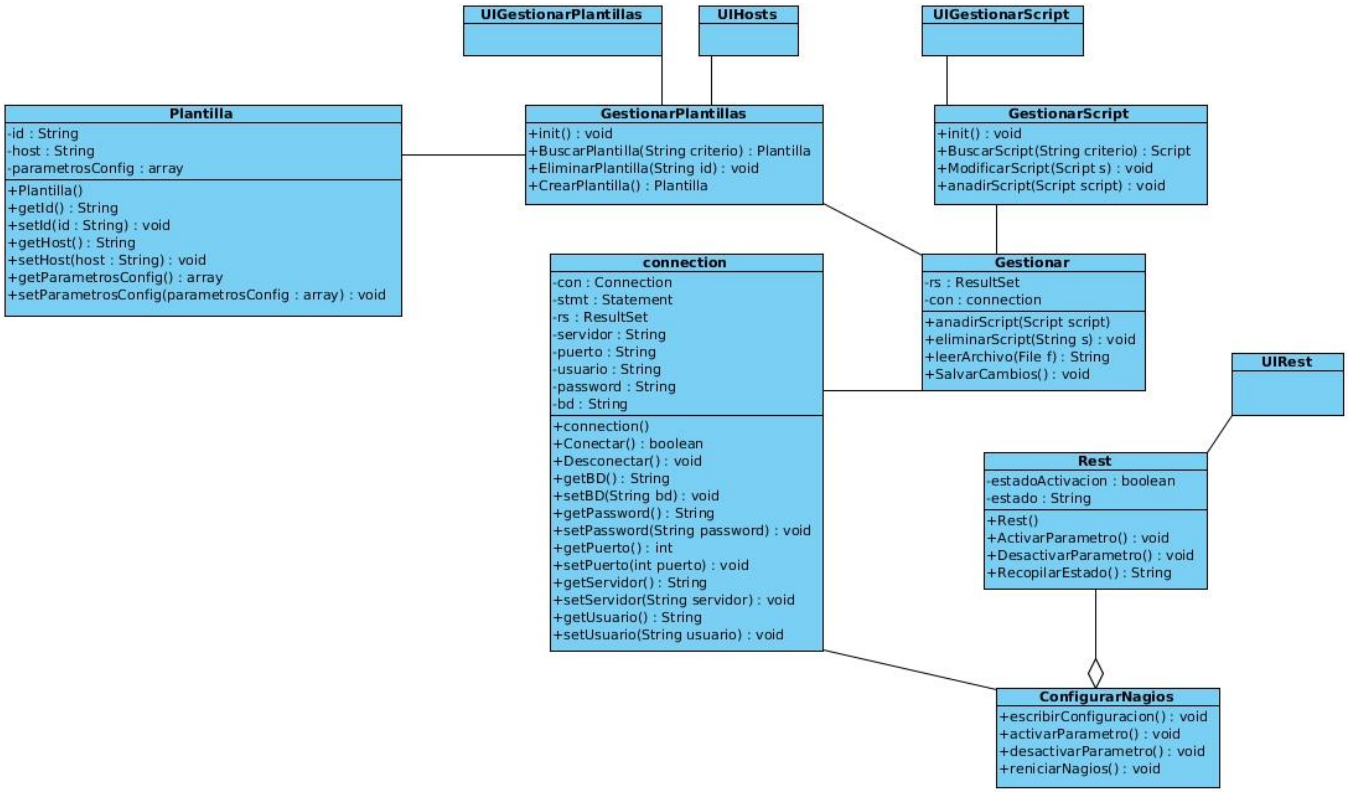

Ilustración 1: Diagrama de Clases

2.3. Arquitectura base de la aplicación

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, presenta un esquema genérico y probado de su solución [42].

2.3.1. Patrones de arquitectura

Un estilo arquitectónico describe una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes [43].

Atendiendo a que la aplicación que se desea desarrollar emplea a Liferay como portal web y que el mismo no define patrón de arquitectura alguno, sino que está basado en Struts [44] para seguir el patrón modelo-vista-controlador [45], se decide usar este patrón en el desarrollo de la aplicación. Este patrón permite dividir los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

Vista: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario. En este punto Liferay ofrece una arquitectura de temas que permite llevar a cabo cambios en la apariencia del portal sin necesidad de modificar el código fuente. En la aplicación que se desea implementar se hace uso de esta característica ya que los temas que se emplean son propios de Liferay.

Controlador: Recibe las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsación de teclas, entra otras. Los eventos son traducidos a solicitudes de servicio para el modelo de la vista. Es el que debe controlar los eventos, en la aplicación se emplea Vaadin [46], como framework web creado para desarrollar aplicaciones RIA (Rich Internet Applications) [47]. Este es el encargado de atender todos los eventos que ocurran en la aplicación.

Modelo: Encapsula los datos de las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos [43]. Esto se pone de manifiesto con el uso de los services builders de Liferay, este es el encargado de realizar todo el acceso a datos y la persistencia de los mismos en la base de datos.

En la Ilustración 2: Patrón modelo-vista-controlador pone de manifiesto de una forma más clara el uso del patrón modelo-vista-controlador en la solución propuesta.

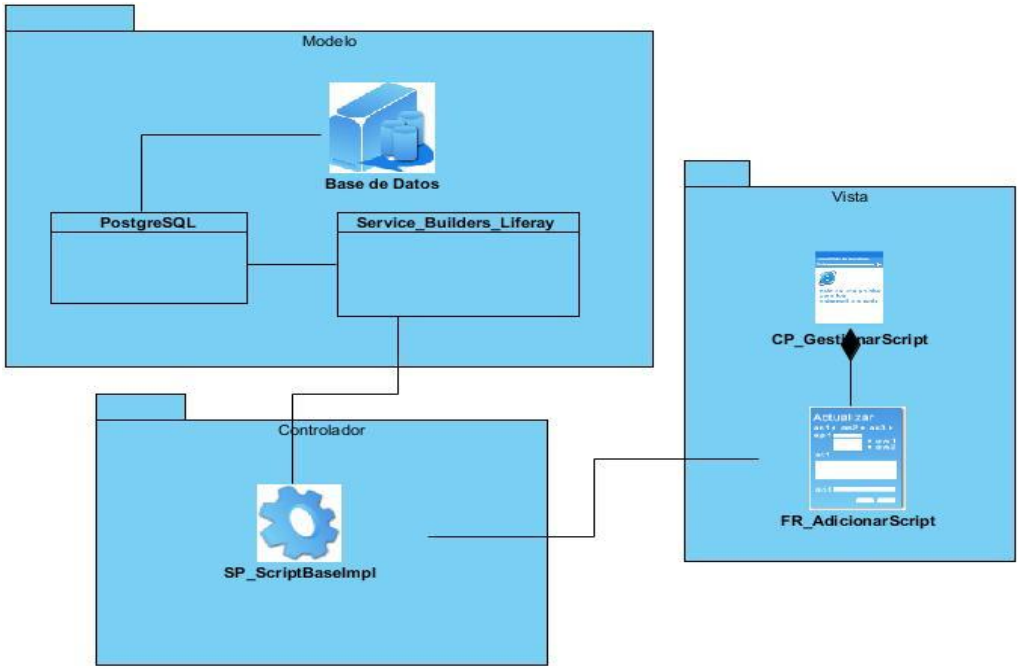

Ilustración 2: Patrón modelo-vista-controlador

2.3.2. Patrones de diseño

Un patrón de diseño (en programación orientada a objetos, POO), es una descripción de diversos objetos y clases preparados para resolver un problema de diseño general aplicado a un contexto específico. Un patrón de diseño identifica las instancias y clases que participan en dicho patrón además de sus papeles, sus relaciones y sus responsabilidades para llevar a cabo la tarea a resolver. Cada patrón se centra en resolver un problema particular en la POO. Describe cuando se puede aplicar, si puede ser aplicado desde el punto de vista de las limitaciones del diseño y las consecuencias, tanto positivas como negativas que tiene su utilización [48].

2.3.3 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza, que ayuda a entender el diseño del objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable [49].

Los patrones GRASP se dividen en 6 de ellos: bajo acoplamiento, creador, experto, alta cohesión, controlador.

Patrón Bajo Acoplamiento

En el patrón de bajo acoplamiento las clases deben estar los menos ligadas entre sí, que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre ellas [50]. Esto se puede evidenciar en las clases que manejan los datos a través de los service builders de Liferay ejemplo, en la clase **ScriptBaseImp.java**.

Patrón creador

Se utiliza cuando se quiere a partir de una clase con alta jerarquía obtener clases descendientes o instancias a partir de las clases obtenidas (8). Un ejemplo donde se pone de manifiesto este patrón es en la clase **ConfigurarSistema.java** ya que esta es la encargada de crear objetos de tipo plantilla para asignárselos a la configuración de un sistema [50].

Patrón experto

El patrón experto indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo; el cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos [50]. Este patrón se evidencia en la clase Controladora ya que es esta la encargada de contener lo necesario para gestionar las actividades sobre las plantillas, por ejemplo la creación de una de ellas.

Patrón alta cohesión

El patrón Alta Cohesión se utiliza para que una clase tenga responsabilidad moderada en un área funcional y colabore con otras clases para llevar a cabo las tareas.

En cuanto al diseño de objetos, la cohesión (o de manera más específica, la cohesión funcional) es una medida de la fuerza y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión

[50]. La utilización de este patrón se refleja en la clase **GestionarRepositorio.java**, ya que la misma para poder llevar a cabo el proceso de adicionar scripts al repositorio depende de otras clases con las funcionalidades necesarias que identifican a un script.

Patrón controlador

Es el patrón que controla quien debería encargarse de atender un evento del sistema. Este asigna la responsabilidad del manejo de un mensaje a los eventos de un sistema a una clase que representa un conjunto de opciones. Un corolario importante del patrón Controlador es que los objetos de la interfaz (por ejemplo, objetos de ventanas, applets) y la capa de presentación no deberían encargarse de manejar los eventos del sistema [51]. Se puede observar el uso de este patrón en la clase **Container_Util,** la cual se encarga de procesar los datos que serán mostrados al usuario.

2.4. Implementación de la aplicación

2.4.1. Tareas de la ingeniería

Las historias de usuario se evalúan y se dividen cada una en tareas, cada una de las mismas representa una característica del sistema. En la Tabla 6: Tarea de la Ingeniería No 3, aparece una tarea de la ingeniería, mostrándose el miembro del equipo de desarrollo encargado de programarla y una descripción breve de lo que se necesita.

Tabla 6: Tarea de la Ingeniería No 3

Tarea de la ingeniería	
Número Tarea: 3	**Número Historia de Usuario: 3**
Nombre de la tarea: Gestionar repositorio de script	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3

Fecha inicio:	Fecha fin:
Programador responsable: Renier Ortiz Mondejar	
Descripción: El usuario podrá crear, modificar y eliminar scripts dentro del repositorio, organizados por categoría en dependencia de las funciones para las que hayan sido programados.	

2.4.2. Estándares de codificación

Emplear técnicas de codificación y realizar buenas prácticas de programación con vista a generar un código de alta calidad, es de gran importancia para el software que se desarrolla y para obtener un rendimiento del mismo. Además, si se aplica de forma correcta el estándar definido facilita añadir nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. El propósito de las siguientes reglas y recomendaciones es lograr que los programadores del proyecto PostgreSQL tengan un estilo de código común [52].

Identación

La unidad de identado es de 4 espacios. El uso de la tabulación debe ser evitado porque (tal como se escribía en el siglo pasado) no existe un estándar que determine con precisión el ancho que va a producir la tabulación.

Longitud de la línea

Deben evitarse las líneas con más de 80 caracteres. Cuando una sentencia sobrepase una línea simple del editor, esta debe ser separada en otras. Realizar la separación después de un operador, preferentemente luego de una coma, pues de esta forma disminuye la posibilidad de que al realizar un copiado de código se cometa un error por olvido de la inserción del punto. La siguiente línea debe ser identada con 5 espacios.

Comentarios

Es muy importante dejar alguna información que pueda ser leída después por aquellas personas (posiblemente usted mismo) que necesitan entender que fue lo que se hizo en el fragmento de código, sirviéndole el comentario para tener un mayor entendimiento de lo implementado. Se debe tener en cuenta que a la hora de escribirlos, los mismos deben tener la mayor claridad posible para lograr un mayor entendimiento del usuario. Es recomendable la utilización de comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código.

Declaración de variables

La declaración de cada una de las variables debe realizarse en una línea y ser comentada, además deben aparecer ordenadas alfabéticamente: El nombre de las variables debe comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula.

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de la declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

Identificadores

Los identificadores pueden estar formados por cualesquiera de las 26 letras minúsculas o mayúsculas (A... Z, a... z), los 10 dígitos (0... 9) y el carácter subrayado “_”. Debe evitarse el uso de caracteres internacionales porque no siempre pueden ser leídos o entendidos correctamente en todos los lugares. No se debe usar el símbolo de dólar “$” o la barra invertida “\” en los identificadores.

Sentencias

Sentencias simples

Cada línea debe contener como máximo una sentencia. Debe poner un punto y coma “;” al final de cada sentencia simple. Tenga en cuenta que una sentencia de asignación puede resultar una asignación de una

función o de un objeto como literal y en todo los casos como sentencia de asignación debe estar finalizada con un punto y coma.

Sentencias compuestas

Son aquellas que contienen una lista de sentencias encerradas entre llaves:

Las sentencias encerradas deben ser identadas a 4 espacios.

La llave que inicia la lista de sentencias debe estar al final de la línea de la sentencia compuesta.

La llave que determina la lista de sentencias debe estar al comienzo de una línea y guardar la misma identación que la sentencia compuesta en correspondencia con la llave que inicia.

Las llaves siempre serán usadas para listar todas las sentencias, aunque se trate de una sola, cuando son parte de una estructura de control como if o for. Ello facilita agregar nuevas sentencias sin la introducción accidental de errores.

Etiquetas

Las sentencias etiquetadas son opcionales. Solo estas sentencias deben ser etiquetadas: while, do, for, foreach, switch.

Sentencia return

Una sentencia return no debe utilizar paréntesis" ( )" alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminando con un punto y coma.

Sentencia if

La sentencia if debe ser escrita de esta manera:

```
if (condition){
        statements
}
```

```
if (condition){
        statements
} else{
        statements
}
if (condition){
statements
}else if (condition){
        statements
}else{
        statements
}
```

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera:

```
for (initialization; condition; update){
        statements
}
while (condition){
        statements
}
forecah (valor1, valor2){
        statements
}
```

Sentencia switch

La sentencia switch debe ser escrita de las forma:

```
switch (expression){
        case expression:
        statements
        case expression:
        statements
        default:
        statements
}
```

Espacios en blanco

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionada. Los espacios en blanco pueden ser (o no deben ser) utilizados en las siguientes circunstancias:

No se debe utilizar un espacio en blanco entre el identificador de una función y el paréntesis que abre a la lista de parámetros. Ello ayuda a distinguir palabras reservadas y llamadas a funciones.

Para cualquier operador binario excepto el punto ".", el paréntesis que abre y el corchete que abre, todos los demás deben ser separado por un espacio entre operando y operador.

No se debe utilizar el espacio para separar un operador unario de su operando excepto cuando ese operador es una palabra como typeof.

Cada punto y coma ";" en una sentencia de control debe ser seguido por un espacio.

Debe dejarse un espacio después de cada coma ",".

2.4.3. Interfaces principales de la aplicación

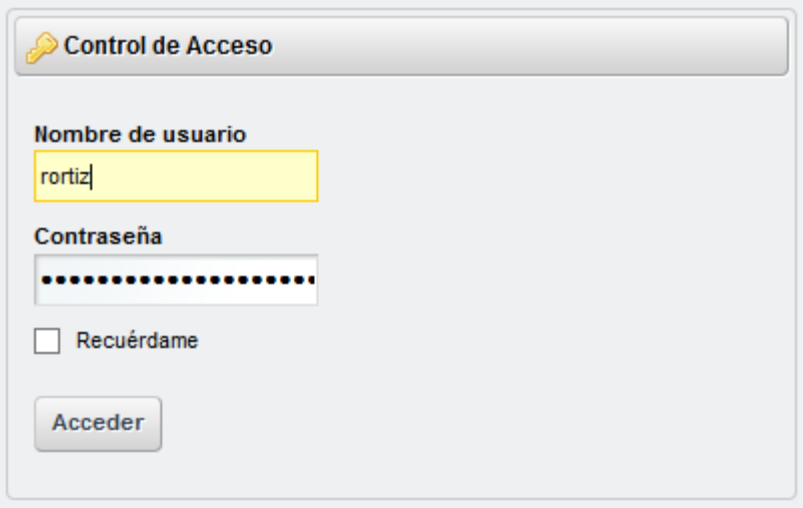

Ilustración 3: Control de acceso

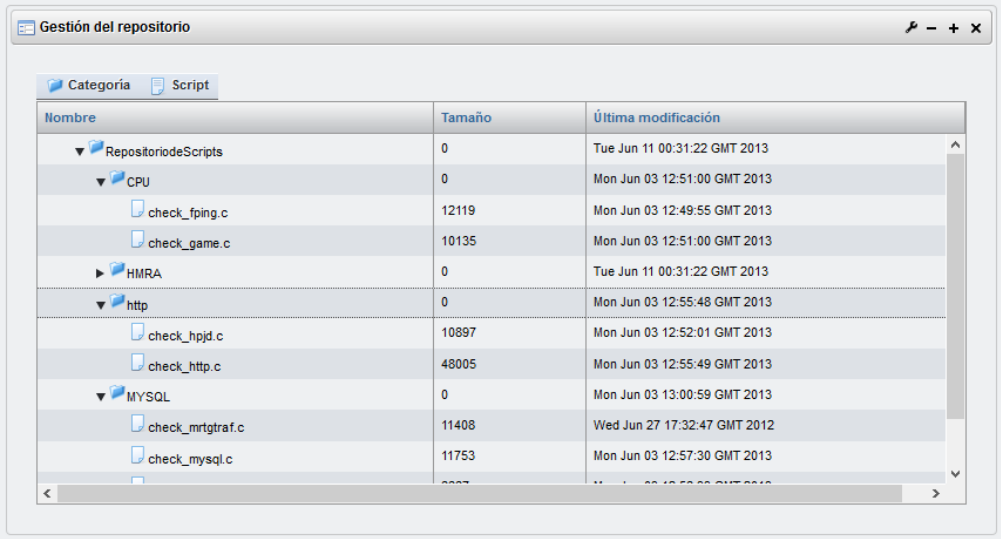

Ilustración 4: Gestionar repositorio

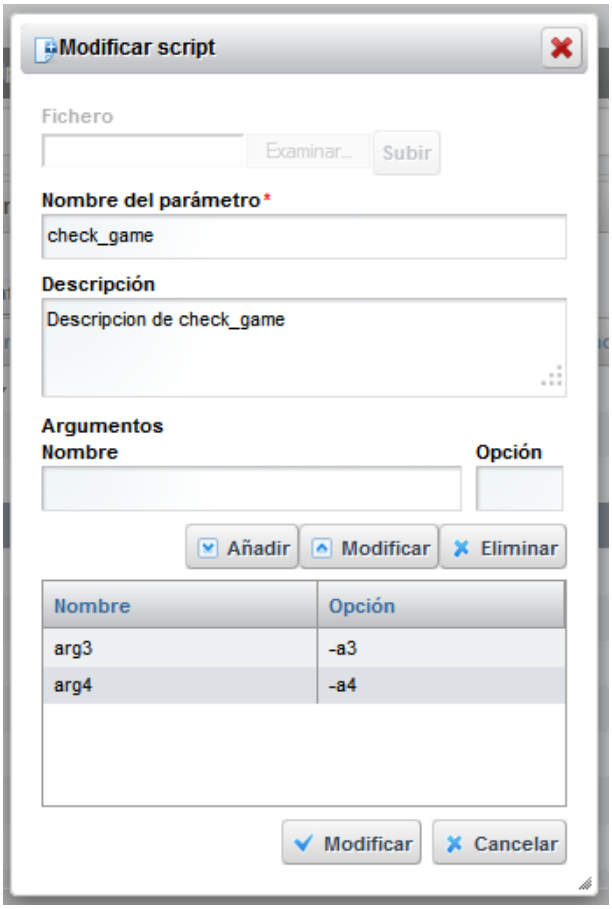

Ilustración 5: Modificar Script

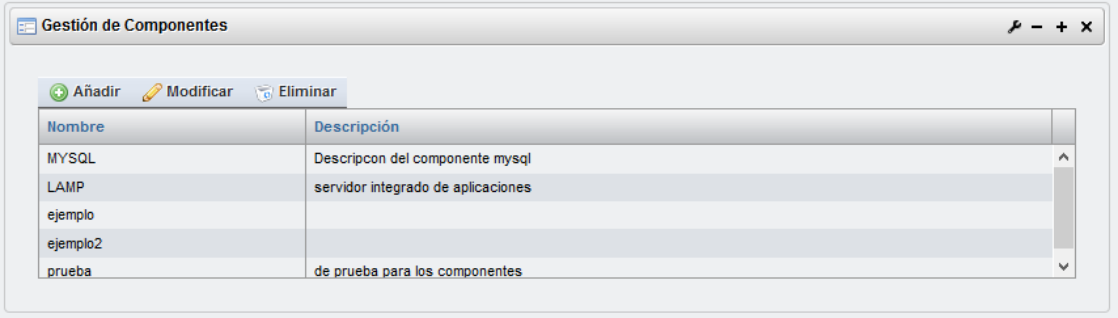

Ilustración 6: Gestión de componentes

Ilustración 7: Añadir componente

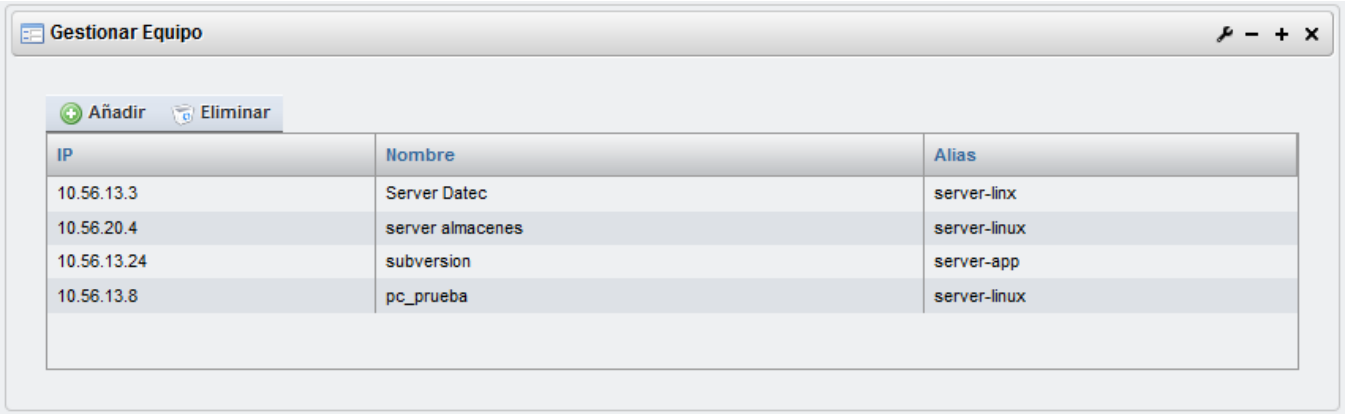

Ilustración 8: Gestionar equipo

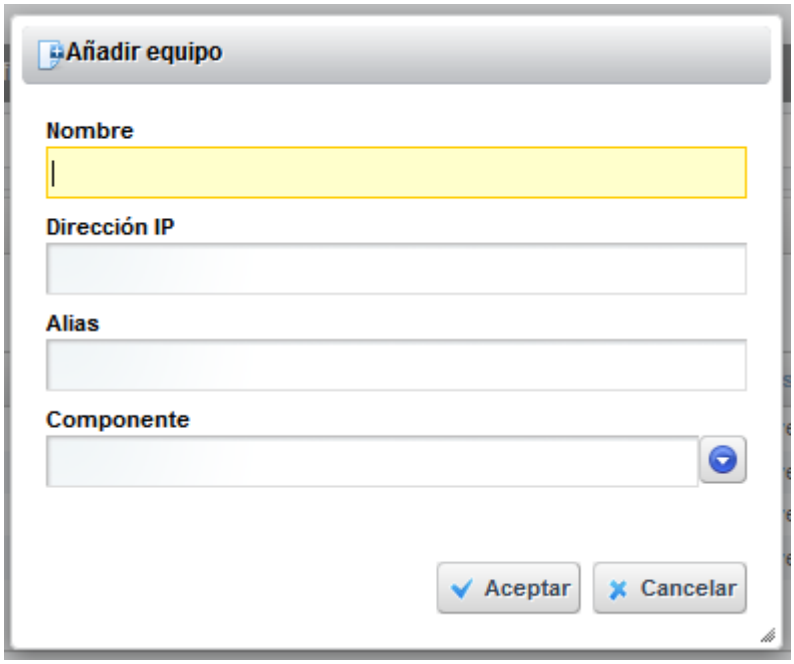

Ilustración 9: Añadir equipo

2.5. Conclusiones del capítulo

En el presente capítulo se hizo alusión a la descripción de la solución propuesta, quedando descompuesta en historias de usuarios, permitiendo especificar los aspectos funcionales y no funcionales de la solución, así como las tareas ingenieriles. Aunque la metodología XP no lo define, se realizó el diagrama modelo del dominio para un mejor entendimiento. Quedó expuesta en un diagrama de clases la solución propuesta, pudiendo evidenciarse las distintas clases con las que contará el sistema. Se abordó acerca de los patrones arquitectónicos empleados, definiendo el modelo-vista-controlador para el desarrollo de la solución, así como los patrones de diseño. Se especificaron los estándares de código a utilizar. Se espera que teniendo en cuenta la descripción de todo lo anterior la implementación sea efectuada de la mejor forma, permitiendo un producto final con buena calidad.

Capítulo #3 Validación de la aplicación

Introducción

En este capítulo se definen un grupo de normas para la validación de los productos y/o artefactos, logrando que los mismos puedan ser probados para la verificación de su correcto funcionamiento. Se analizan las estrategias de pruebas que definen XP y la técnica que se utilizará para diseñar los casos de pruebas que guiarán la validación de la aplicación, además de mostrarse los resultados arrojados por éstas pruebas.

3.1. Validación de la aplicación

La validación es un proceso general. Se debe asegurar que el software cumpla con las expectativas del cliente. Este proceso va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado. Es importante llevar a cabo la validación de los requerimientos del sistema de forma inicial. Es fácil cometer errores y omisiones durante la fase de análisis de requerimientos del sistema, y en tales casos el software final no cumplirá las expectativas del cliente. Dentro del proceso de validación se utilizan dos técnicas de comprobación y análisis de sistemas, las inspecciones del software y las pruebas del software [53].

3.2. Estrategias de Pruebas

Una de las primacías fundamentales de la metodología XP. Es proporcionar al cliente la posibilidad de concretar las funcionalidades de las HU y verificarlas en el proceso de pruebas. Las mismas son la manera de comprobar el código que se vaya implementando. Las pruebas de software de consisten en la verificación dinámica del comportamiento de un programa en un conjunto finito de casos de prueba [54].

Existes diversas estrategias de pruebas, en el desarrollo de este trabajo se hizo un estudio acerca del proceso de prueba, correspondiente a la metodología de desarrollo empleado en el presente trabajo. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñadas por los propios programadores, y pruebas de aceptación o pruebas funcionales, destinadas a evaluar si al final de una iteración se logró la funcionalidad requerida, estas son diseñadas por el cliente final [54].

Pruebas unitarias

Las pruebas unitarias se escriben antes y detallan antes de escribir el código. Por cada historia de usuario, el cliente escribe una prueba unitaria. Estas son ejecutadas entonces, antes de concluir el proceso de integración, continua con la modificación del sistema. Todo este proceso asegura que el nuevo código implementa la funcionalidad correcta y se integra correctamente con el resto sin necesidad de cambios en este sentido. [55].

Pruebas de aceptación

Las pruebas de aceptación, al igual que las de sistema, se realizan sobre el producto terminado e integrado, pero a diferencia de aquellas están concebidas para que sea un usuario final quien detecte los posibles errores.

Estas son definidas por el cliente en cada HU, y tienen como objetivo asegurar que las funcionalidades del sistema cumplan con lo que se espera de ellas. Las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración. Estas pruebas no se realizan durante el desarrollo ya que sería impresentable al cliente, sino que se realizan ya terminado el producto o luego de una iteración pactada previamente con el cliente [56].

Para la validación de la aplicación se realizaran pruebas de aceptación.

3.3. Técnicas para la realización de las pruebas

Cualquier proyecto que se trace una estrategia de prueba debe contar con estrategias y técnicas para la aplicación de cada una de estas pruebas. A continuación se realiza una breve caracterización de dos técnicas importantes para de esta forma seleccionar una de estas técnicas para la correcta realización de las pruebas.

3.3.1. Pruebas estructurales o de caja blanca

La prueba de caja blanca es una técnica de diseño de casos de prueba que realiza las pruebas al código de la aplicación y que usa la estructura de control del diseño procedimental para derivar los casos de prueba.

Ellas garantizan que el programador pueda ejecutar los caminos independientes de cada módulo al menos una vez y que utilice todas las estructuras de datos internas. En las pruebas estructurales las pruebas se seleccionan en función del conocimiento que se tiene de la implementación del módulo. Se suelen aplicar a módulos pequeños. El probador analiza el código y deduce cuántos y qué conjuntos de valores de entrada han de probarse para que al menos se ejecute una vez cada sentencia del código [57].

3.3.2 Pruebas funcionales o de caja negra

También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas.

Las pruebas de caja negra pretenden demostrar que las funciones del software son operativas y que funcionan correctamente aceptando de forma adecuada la entrada de datos y produciendo una salida correcta. Este tipo de prueba nos permite demostrarle al cliente que la aplicación satisface las necesidades del mismo [58].

Para la validación de la aplicación se van a realizar pruebas de caja negra.

3.4. Descripción de los casos de prueba

Los casos de prueba no son más que un conjunto de variables y condiciones de las que se vale el analista para determinar el correcto funcionamiento o no de la funcionalidad que se esté probando. Para verificar que todos los requisitos funcionales sean cumplidos e implementados correctamente debe realizarse un caso de prueba por cada uno de ellos, y en caso de que un requisito contenga otros en sí, también debe hacerse un caso de prueba para cada uno de ellos. Los casos de prueba se componen de los elementos.

Tabla 7: Descripción de los campos de un caso de prueba

Caso de Prueba de Aceptación	
Código:	Historia de Usuario (No y Nombre):

Nombre:
Descripción:
Condiciones de ejecución:
Entradas / Paso de ejecución:
Resultado esperado:
Evaluación de la prueba:

Campos de la tarjeta de caso de prueba

Historia de Usuario (No y Nombre): Número y nombre de la historia de usuario a la que corresponde.

Código: Número asignado a cada caso de prueba perteneciente a una historia de usuario determinada.

Nombre: Nombre del caso de prueba. Debe describir lo que se quiere comprobar.

Descripción: Descripción de lo que se desea probar, debe ser corta y precisa.

Condiciones de ejecución: Condiciones a tener en cuenta para ejecutarse el caso de prueba.

Entradas / Paso de ejecución: Entradas al caso de prueba en caso de necesitarlas.

Resultado esperado: Descripción breve de lo que debe suceder.

Evaluación de la prueba: Se evalúa si el caso de prueba tuvo éxito o no.

En la Tabla 8: Gestionar repositorio de Scripts, Tabla 9: Gestionar componentes y Tabla 10: Gestionar equipos se definen los casos de prueba principales de los que se van a utilizar, con los mismos se pretende testear un flujo completo de trabajo en la aplicación, guiado por las historias de usuario, desde la HU 1 hasta la HU 7.

Tabla 8: Gestionar repositorio de Scripts

Caso de Prueba de Aceptación	
Código: 1	Historia de Usuario (No y Nombre): HU#1 Gestionar Repositorio de Scripts.
Nombre: Gestionar el repositorio de scripts.	

Descripción: Prueba para la funcionalidad gestionar repositorio de scripts, en la misma se añade una categoría y un script dentro de la categoría.
Condiciones de ejecución: Añadir una categoría dentro del repositorio y dentro de ella un script.
Entradas / Paso de ejecución: Se introduce una categoría al repositorio, pasando el nombre de la categoría, luego se añade un script a la categoría, subiendo el fichero del script, así como el nombre del comando, descripción y los argumentos del script.
Resultado esperado: La categoría y el script se añaden satisfactoriamente.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 9: Gestionar componentes

Caso de Prueba de Aceptación	
Código: 2	Historia de Usuario (No y Nombre): HU#2 Gestionar componentes.
Nombre: Gestionar componentes.	
Descripción: Se añaden componentes especificando nombre, descripción, categoría, parámetro general, así como los parámetros específicos y los componentes asociados. Además se modifica y elimina un componente.	
Condiciones de ejecución: Añadir un componente y modificarlo y modificarlo.	
Entradas / Paso de ejecución: Se introduce un componente pasándole el nombre, descripción, categoría, parámetro general, parámetros específicos y los componentes asociados. Para modificar el componente se le pasan los nuevos datos (descripción, parámetro general, parámetros específicos y componentes asociados).	
Resultado esperado: El componente se añade, modifica y elimina satisfactoriamente.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 10: Gestionar equipos

Caso de Prueba de Aceptación

Código: 3	Historia de Usuario (No y Nombre): HU#3 Gestionar equipos.
Nombre: Gestionar equipos para monitorizar.	
Descripción: Se añaden equipos para monitorizar pasándole el nombre, dirección ip, alias y el componente, además se puede eliminar el equipo.	
Condiciones de ejecución: Añadir un equipo y eliminarlo.	
Entradas / Paso de ejecución: Se introduce un equipo pasándole el nombre, dirección ip, alias y un componente.	
Resultado esperado: El equipo se añade y elimina satisfactoriamente.	
Evaluación de la prueba: Prueba satisfactoria.	

3.4.1. Resultado de las pruebas

Al realizar las pruebas de aceptación durante la primera iteración se obtuvo un 15 no conformidades, implicadas las historias de usuarios, **gestionar componentes** y **enviar estado del monitoreo**. Estas no conformidades fueron corregidas al concluir la iteración.

Al realizar las pruebas de aceptación durante la segunda iteración no se obtuvieron no conformidades, implicadas las historias de usuario, gestionar configuración de equipos y listar sistemas listos para monitorizar.

Al realizar las pruebas de aceptación durante la tercera y última iteración se obtuvieron 7 no conformidades, implicadas las historias de usuario, activar parámetros a chequear y desactivar parámetros a chequear. Algunas de estas no conformidades fueron corregidas al concluir la iteración, persistiendo 2 no conformidades.

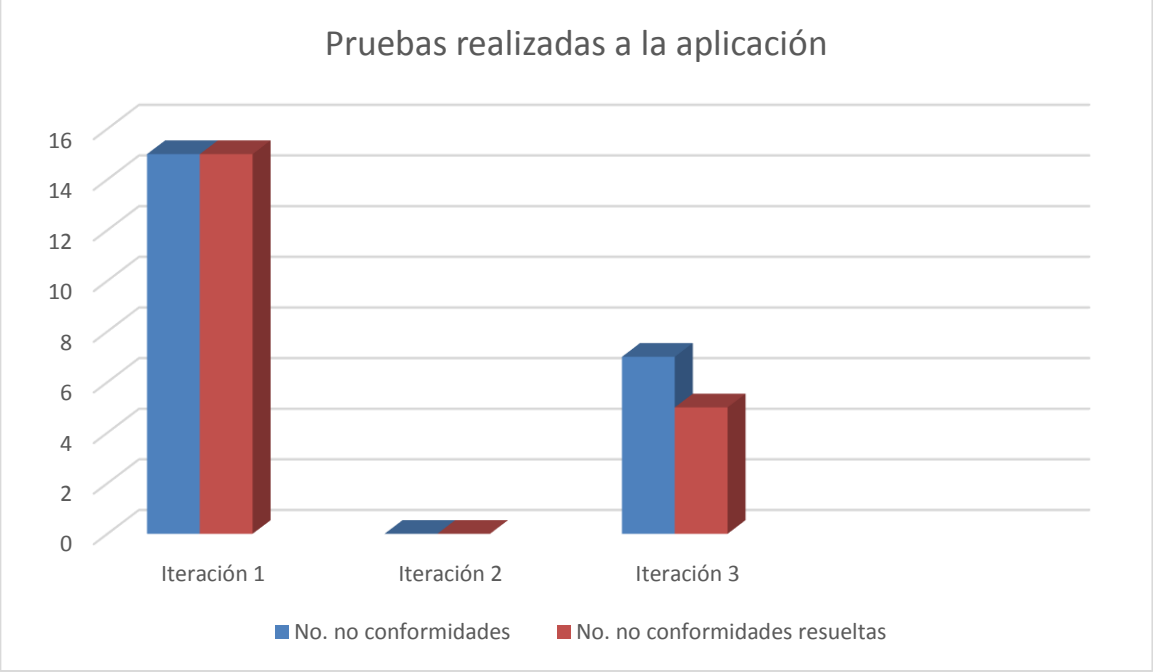

Gráfico 1: Resultado de la pruebas realizadas a la aplicación

3.5. Conclusiones del capítulo

Las pruebas de software permiten la verificación de la calidad del producto. Son empleadas para identificar posibles fallos de una aplicación.

- Fueron probadas todas las funcionalidades y fueron solucionadas las no conformidades encontradas en el transcurso de las pruebas a la aplicación.
- Se logró presentar los resultados arrojados en cada iteración logrando obtener una herramienta que responde a todos los requisitos funcionales identificados.

Conclusiones generales

Para la realización de este trabajo se analizó el proceso de monitorización y chequeo de parámetros sobre servidores y equipos de interconexión, permitiendo identificar características que aporten a la solución del problema. Como resultado del trabajo realizado:

- Se diseñó una aplicación que permite el chequeo a demanda de parámetros a monitorizar en sistemas de servidores y equipos de interconexión, mediante la organización de los parámetros en forma de árbol jerárquico, permitiendo activar y desactivar el chequeo de parámetros de tal forma que conlleve a la detección de fallas en sistema de servidores y equipos de interconexión.
- Se implementó una aplicación que permite realizar el chequeo a demanda de los parámetros a monitorizar en sistemas de servidores y equipos de interconexión, permitiendo organizar scripts de monitoreo en un repositorio de scripts, y crear perfiles de configuración para el chequeo de parámetros.
- Se validó la solución mediante las pruebas y técnicas que propone la metodología XP, detectando algunas no conformidades las cuales fueron resueltas al final de cada iteración.

Recomendaciones

- Implementar una aplicación que se comunique con esta y realice las tareas de análisis de parámetros y generación de alarmas.
- Crear un mecanismo de limpieza de la base de datos para purgar los datos de monitoreo cada cierto tiempo.

Referencias bibliográficas.

1. Ávila, A.R., *Iniciación a la red Internet: Concepto, funcionamiento, servicios y aplicaciones de Internet*2010: Ideaspropias Editorial SL.

2. Diana Lilian Antúnez Ginarte, M.L.G.P., *Personalización de la herramienta de gestión de inventarios OCS Inventory NG*, in *Facultad 1*2011, Universidad de las Ciencias Informáticas. p. 78.

3. Carvajal Dávila, E.C., *Elaboración de una guía de procedimientos de medición y monitoreo en istemas de comunicación SDH par que tengan la característica de tolerancia a fallas*, 2008, QUITO/EPN/2008.

4. Conterón Tene, M.F. and X. DT-López, *Técnica Sniffer para detectar vulnerabilidades en el servidor WEB, MAIL y FTP del Hospital Regional Docente Ambato*, 2012.

5. Iszaevich, G.E.W., *Monitoreo de PostgreSQL con Munin*. Revista Cubana de Ciencias Informáticas, 2011. 5(1).

6. Stallings, W., *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*1998: Addison-Wesley Longman Publishing Co., Inc.

7. Valarezo Saldarriaga, G.G., J.C. Simisterra Huila, and M.C. Yépez Flores, *Implementación de un Sistema de Gestión y Administración de Redes Basados en el Protocolo Simple de Monitoreo de Redes SNMP en la Red ESPOL-FIEC*. 2011.

8. Jiang, G., *Multiple vulnerabilities in SNMP*. Computer, 2002. 35(4): p. 2-4.

9. Administrativas, C.U.d.C.E.-. *ADMINISTRACIÓN DE REDES*. 2000 [cited 2013 17/01/2013]; Disponible desde: http://html.rincondelvago.com/administracion-de-redes.html.

10. Barcelona, U.a.d., *SISTEMA DE MONITORIZACIÓN DE SERVIDORES LINUX*, in *Memoria del proyecto de Ingeniería Técnica en Informática de Sistemas*2010.

11. Cerdas, G.E.G., *Sistema de monitoreo, detección y notificación de fallas en las condiciones de operación de los equipos de laboratorio de Componentes Intel de Costa Rica*.

12. Coulouris, G., J. Dollimore, and T. Kindberg, *Sistemas distribuidos*2001: Addison Wesley.

13. Miranda, J.J., A. Paca-Palao, and L. Najarro, *Evaluación situacional, estructura, dinámica y monitoreo de los sistemas de información en accidentes de tránsito en el Perú-2009*. Rev. perú. med. exp. salud publica, 2010. 27(2): p. 273-287.

14. Brida, J., D. Matesanz Gómez, and W.A. Risso, *Contagion, Correlations and Topology: Analysis of the Currency Dynamics in the Latin American Markets (Contagio, Correlaciones Y Topología. Análisis De La Dinámica Cambiaria En Los Mercados Latinoamericanos)*. Investigación Económica, 2009. 68(267): p. 115-146.

15. Denegri, E. and G. Frontera Sánchez, *Algoritmos para grafos y programación de propósito general en CUDA*. 2009.

16. Annioldys Uranga González, E.M.C., *Adaptación y configuración de OCS Inventario NG para el Inventario centralizado de Hardware y Software en la Universidad de las Ciencias Informáticas*, in *Faculta 2*2008, UCI.

17. Wolf, G., *Monitoreo de PostgreSQL con Munin*. Revista Cubana de Ciencias Informáticas, 2011. 5(1): p. 1-8.

18. Ojeda, O. and D.A. Valdivieso Carrión, *Identificación y levantamiento de las plataformas de gestión ymonitoreo para la elaboración de un manual de usuario que seráutilizado en la aplicación y ejecución de procesos en la red BackboneIP/MPLS de la Corporación Nacional de Telecomunicaciones*. 2012.

19. Cayuqueo, S., *Monitoria y análisis de Red con Nagios*, 2012.

20. Gandul Soto, A., *Sistema de motorización y control de un proceso de programación de vehículos*. 2011.
21. Cienfuegos, G.d.T.L. *Centreon 2.1.11, Nagios 3.2.3, Nagios-plugins,NDOutils desde los fuentes, Nagvis 1.5.8 en Debian Lenny*. 2011; Disponible desde: http://www.cfg.jovenclub.cu/gulcf/?p=595.
22. Gómez, J., *Monitorización con Centreon*. Todo Linux: la revista mensual para entusiastas de GNU/Linux, 2010(116): p. 38-41.
23. Sánchez, A.L. *slideshare*. 2011 [cited 2012 martes 11 de diciembre]; Presentacion sobre Centreon]. Disponible desde: www.slideshare.net/aitortersio/centreon-breve-explicacin.
24. Durán, C., et al., *Desarrollo de un portal WEB para la Superintendencia de Servicios de Certificación Electrónica,(SUSCERTE) capaz de generar la acreditación, renovación o suspensión a los proveedores de servicios de certificación electrónica*. 2012.
25. Joomla. *What is Joomla?* . 2013; Disponible desde: http://www.joomla.org/about-joomla.html.
26. Tramullas, J., *Drupal para bibliotecas y archivos*2010: [Zaragoza]: Grupo de investigación sobre Gestión de Recursos de Información en las Organizaciones (Universidad de Zaragoza)-Fundación Zaragoza Ciudad del Conocimiento, 2010.
27. Miño Viteri, P.M., *Desarrollo del portal interno de GESTORINC SA mediante la creación extensión e implementación de portlets, utilizando liferay portal como gestor de contenidos*. 2010.
28. Bonilla López, J.M., *Projecte Intranet Liferay*. 2012.
29. Barrios, W.G., et al., *SCRUM: application experience in a software development PyME in the NEA*. Journal of Computer Science & Technology, 2012. 12.
30. Parra Castrillón, E., *Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje-MESOVA*. Revista Virtual Universidad Católica del Norte, 2011. 1(34): p. 113-137.
31. Mendes Calo, K., E.C. Estevez, and P.R. Fillotranni. *Evaluación de metodologías ágiles para desarrollo de software*. in *XII Workshop de Investigadores en Ciencias de la Computación*. 2010.
32. Letelier, P., *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. 2006.
33. Gil, G.D., et al. *Metodologías ágiles y desarrollo basado en el conocimiento, evaluación cuantitativa de F/OSS para la reutilización, Normas ISO y su aplicación en centros educativos*. in *XIV Workshop de Investigadores en Ciencias de la Computación*. 2012.
34. Quintero, J.B., et al., *Un estudio comparativo de herramientas para el modelado con UML*. revista universidad eafit, 2012. 41(137): p. 60-76.
35. Ochoa, I.S. and J.M. Herrera, *Sistema para la extracción de información desde Servicios Web y su visualización en móviles*. Serie Científica, 2009. 2(4).
36. Pérez, P.Y.P., et al., *Experiencias en el uso de PostgreSQL en el sistema GESPRO, un enfoque práctico*. Revista Cubana de Ciencias Informáticas, 2011. 5(1).
37. NOVELLA LATORRE, J., *Sistema de gestión de base de datos PostgreSQL*. 2012.
38. Dayana de las Mercedes Fuentes Rodríguez, P.M.C.B., *Juego Multijugador Dominó para Celulares*, in *UCI*2010, UCI: Cuba. p. 78.
39. Costello, R.L., *Building Web Services the REST Way*. 2008.
40. José H. Canós, P.L.y.M.C.P., *Métodologías Ágiles en el Desarrollo de Software*.
41. Stevens, P., et al., *Utilización de UML en Ingeniería del Software con Objetos y Componentes*2007: Addison Wesley.
42. Valencia, A.M. and M. Ferro González, *Documentacion y analisis crítico de algunas arquitecturas de software en aplicaciones empresariales*. 2011.
43. Reynoso, C.B., *Introducción a la Arquitectura de Software*. Documento de la Universidad de Buenos Aires. Descargado de http://www.willydev.net/descargas/prev/IntroArq.pdf, 2004. 21(01): p. 2009.

44. Husted, T., et al., *Struts in Action: Building web applications with the leading Java framework*2003: Manning.
45. Toubes Tova, E., *Gestor de comunidades online: GCO*. 2010.
46. Grönroos, M., *Book of Vaadin*2011: Vaadin Limited.
47. Moreno Marín, J., *Artículos de colección-Aplicaciones RIA*. 2012.
48. Barnes, D.J. and M. Kölling, *Programación orientada a objetos con Java*. MANEJO EFICIENTE DEL TIEMPO, 2007.
49. Masanet, M.I., E. Zavalla, and A. Fernández. *Un enfoque integrado para las prácticas de laboratorio en la educación a distancia*. in *XV Congreso Argentino de Ciencias de la Computación*. 2009.
50. Tabares, R.B., *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación*. REVISTA ENTRE CIENCIA E INGENIERÍA, 2011(8): p. 161-173.
51. Lott, J. and D. Patterson, *ActionScript 3. Patrones de diseño*2007.
52. Eidelman, A.P., *eXtreme Programming*. Tendencias Tecnológicas en Arquitecturas y Desarrollo de Aplicaciones: p. 5.
53. Kasiak, T. and D.A. Godoy. *Simulación de Proyectos de Software desarrollados con XP: Subsistema de Desarrollo de Tareas*. in *XIV Workshop de Investigadores en Ciencias de la Computación*. 2012.
54. Chicaiza Molina, J.A. and L.A. Quispe Quispe, *Desarrollo del sistema integrado parametrizable contable y financiero online para empresas comerciales y de servicios (scfonline) utilizando la metodología midas (metodología para el desarrollo de sistemas de información web)*. 2007.
55. Rosales, C.L.S., *PRÁCTICAS DE EXTREME PROGRAMMING (XP) EN LA ENSEÑANZA DE LA INGENIERÍA DE SOFTWARE*.
56. MÉNDEZ NAVA, E. and G. RAMÓN, *Modelo de Evaluación de Metodologías para el Desarrollo de Software*, 2006, Tesis de especialización no publicada, Universidad Católica Andrés Bello, Caracas.
57. Rodriguez, C.A. and J.E. Bonilla. *Pruebas en Programación Extrema*. in *IV SIMPOSIO INTERNACIONAL DE SISTEMAS DE INFORMACIÓN E*.
58. Sanz, L.F., *Tutorial: pruebas funcionales y trabajo en equipo*. Universidad Europea de Madrid, Grupo de calidad de software de ATI, 2007.

Bibliografía

1. Ávila, A.R., Iniciación a la red Internet: Concepto, funcionamiento, servicios y aplicaciones de Internet2010:Ideaspropias Editorial SL.

2. Diana Lilian Antúnez Ginarte, M.L.G.P., Personalización de la herramienta de gestión de inventarios OCS Inventory NG, in Facultad 12011, Universidad de las Ciencias Informáticas.

3. Carvajal Dávila, E.C., Elaboración de una guía de procedimientos de medición y monitoreo en istemas de comunicación SDH par que tengan la característica de tolerancia a fallas, 2008.

4. Conterón Tene, M.F. and X. DT-López, Técnica Sniffer para detectar vulnerabilidades en el servidor WEB, MAIL y FTP del Hospital Regional Docente Ambato, 2012.

5. Iszaevich, G.E.W., Monitoreo de PostgreSQL con Munin. Revista Cubana de Ciencias Informáticas, 2011.

6. Stallings, W., SNMP, SNMPv2, SNMPv3, and RMON 1 and 21998: Addison-Wesley Longman Publishing Co., Inc.

7. Valarezo Saldarriaga, G.G., J.C. Simisterra Huila, and M.C. Yépez Flores, Implementación de un Sistema de Gestión y Administración de Redes Basados en el Protocolo Simple de Monitoreo de Redes SNMP en la Red ESPOL-FIEC. 2011.

8. Jiang, G., Multiple vulnerabilities in SNMP. Computer, 2002.

9. Administrativas, C.U.d.C.E.-. ADMINISTRACIÓN DE REDES. Disponible desde: http://html.rincondelvago.com/administracion-de-redes.html.

10. Barcelona, U.a.d., SISTEMA DE MONITORIZACIÓN DE SERVIDORES LINUX, in Memoria del proyecto de Ingeniería Técnica en Informática de Sistemas 2010.

11. Cerdas, G.E.G., Sistema de monitoreo, detección y notificación de fallas en las condiciones de operación de los equipos de laboratorio de Componentes Intel de Costa Rica.

12. Coulouris, G., J. Dollimore, and T. Kindberg, Sistemas distribuidos 2001: Addison Wesley.

13. Miranda, J.J., A. Paca-Palao, and L. Najarro, Evaluación situacional, estructura, dinámica y monitoreo de los sistemas de información en accidentes de tránsito en el Perú-2009. Rev. perú. med. exp. salud publica, 2010.

14. Brida, J., D. Matesanz Gómez, and W.A. Risso, Contagion, Correlations and Topology: Analysis of the Currency Dynamics in the Latin American Markets (Contagio, Correlaciones Y Topología. Análisis De La Dinámica Cambiaria En Los Mercados Latinoamericanos). Investigación Económica, 2009.

15. Denegri, E. and G. Frontera Sánchez, Algoritmos para grafos y programación de propósito general en CUDA. 2009.

16. Annioldys Uranga González, E.M.C., Adaptación y configuración de OCS Inventario NG para el Inventario centralizado de Hardware y Software en la Universidad de las Ciencias Informáticas, in Facultad 2 2008, UCI.

17. Wolf, G., Monitoreo de PostgreSQL con Munin. Revista Cubana de Ciencias Informáticas, 2011.

18. Ojeda, O. and D.A. Valdivieso Carrión, Identificación y levantamiento de las plataformas de gestión y monitoreo para la elaboración de un manual de usuario que será utilizado en la aplicación y ejecución de procesos en la red BackboneIP/MPLS de la Corporación Nacional de Telecomunicaciones. 2012.

19. Cayuqueo, S., Monitoria y análisis de Red con Nagios, 2012.

20. Gandul Soto, A., Sistema de motorización y control de un proceso de programación de vehículos. 2011.

21. Cienfuegos, G.d.T.L. Centreon 2.1.11, Nagios 3.2.3, Nagios-plugins,NDOutils desde los fuentes, Nagvis 1.5.8 en Debian Lenny. 2011; Disponible desde: http://www.cfg.jovenclub.cu/gulcf/?p=595.

22. Gómez, J., Monitorización con Centreon. Todo Linux: la revista mensual para entusiastas de GNU/Linux, 2010.

23. Sánchez, A.L. slideshare. 2011 Presentacion sobre Centreon. Disponible desde: www.slideshare.net/aitortersio/centreon-breve-explicacin.

24. Durán, C., et al., Desarrollo de un portal WEB para la Superintendencia de Servicios de Certificación Electrónica, (SUSCERTE) capaz de generar la acreditación, renovación o suspensión a los proveedores de servicios de certificación electrónica. 2012.

25. Joomla. What is Joomla? . 2013; Disponible desde: http://www.joomla.org/about-joomla.html.

26. Tramullas, J., Drupal para bibliotecas y archivos2010: [Zaragoza]: Grupo de investigación sobre Gestión de Recursos de Información en las Organizaciones (Universidad de Zaragoza)-Fundación Zaragoza Ciudad del Conocimiento, 2010.

27. Miño Viteri, P.M., Desarrollo del portal interno de GESTORINC SA mediante la creación extensión e implementación de portlets, utilizando liferay portal como gestor de contenidos. 2010.

28. Bonilla López, J.M., Projecte Intranet Liferay. 2012.

29. Barrios, W.G., et al., SCRUM: application experience in a software development PyME in the NEA. Journal of Computer Science & Technology, 2012. 12.

30. Parra Castrillón, E., Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje-MESOVA. Revista Virtual Universidad Católica del Norte, 2011.

31. Mendes Calo, K., E.C. Estevez, and P.R. Fillotranni. Evaluación de metodologías ágiles para desarrollo de software. in XII Workshop de Investigadores en Ciencias de la Computación. 2010.

32. Letelier, P., Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2006.

33. Gil, G.D., et al. Metodologías ágiles y desarrollo basado en el conocimiento, evaluación cuantitativa de F/OSS para la reutilización, Normas ISO y su aplicación en centros educativos. in XIV Workshop de Investigadores en Ciencias de la Computación. 2012.

34. Quintero, J.B., et al., Un estudio comparativo de herramientas para el modelado con UML revista universidad eafit, 2012.

35. Ochoa, I.S. and J.M. Herrera, Sistema para la extracción de información desde Servicios Web y su visualización en móviles. Serie Científica, 2009.

36. Pérez, P.Y.P., et al., Experiencias en el uso de PostgreSQL en el sistema GESPRO, un enfoque práctico. Revista Cubana de Ciencias Informáticas, 2011.

37. NOVELLA LATORRE, J., Sistema de gestión de base de datos PostgreSQL. 2012.

38. Dayana de las Mercedes Fuentes Rodríguez, P.M.C.B., Juego Multijugador Dominó para Celulares, in UCI 2010, UCI: Cuba.

39. Costello, R.L., Building Web Services the REST Way. 2008.

40. José H. Canós, P.L.y.M.C.P., Métodologías Ágiles en el Desarrollo de Software.

41. Stevens, P., et al., Utilización de UML en Ingeniería del Software con Objetos y Componentes 2007: Addison Wesley.

42. Valencia, A.M. and M. Ferro González, Documentacion y analisis crítico de algunas arquitecturas de software en aplicaciones empresariales. 2011.

43. Reynoso, C.B., Introducción a la Arquitectura de Software. Documento de la Universidad de Buenos Aires. Descargado de http://www.willydev.net/descargas/prev/IntroArq.pdf, 2004.

44. Husted, T., et al., Struts in Action: Building web applications with the leading Java framework2003: Manning.

45. Toubes Tova, E., Gestor de comunidades online: GCO. 2010.

46. Grönroos, M., Book of Vaadin2011: Vaadin Limited.

47. Moreno Marín, J., Artículos de colección-Aplicaciones RIA. 2012.

48. Barnes, D.J. and M. Kölling, Programación orientada a objetos con Java. MANEJO EFICIENTE DEL TIEMPO, 2007.

49. Masanet, M.I., E. Zavalla, and A. Fernández. Un enfoque integrado para las prácticas de laboratorio en la educación a distancia. in XV Congreso Argentino de Ciencias de la Computación. 2009.

50. Tabares, R.B., Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. REVISTA ENTRE CIENCIA E INGENIERÍA, 2011.

51. Lott, J. and D. Patterson, ActionScript 3. Patrones de diseño 2007.

52. Eidelman, A.P., eXtreme Programming. Tendencias Tecnológicas en Arquitecturas y Desarrollo de Aplicaciones

53. Kasiak, T. and D.A. Godoy. Simulación de Proyectos de Software desarrollados con XP: Subsistema de Desarrollo de Tareas. in XIV Workshop de Investigadores en Ciencias de la Computación. 2012.

54. Chicaiza Molina, J.A. and L.A. Quispe Quispe, Desarrollo del sistema integrado parametrizable contable y financiero online para empresas comerciales y de servicios (scfonline) utilizando la metodología midas (metodología para el desarrollo de sistemas de información web). 2007.

55. Rosales, C.L.S., PRÁCTICAS DE EXTREME PROGRAMMING (XP) EN LA ENSEÑANZA DE LA INGENIERÍA DE SOFTWARE.

56. MÉNDEZ NAVA, E. and G. RAMÓN, Modelo de Evaluación de Metodologías para el Desarrollo de Software, 2006, Tesis de especialización no publicada, Universidad Católica Andrés Bello, Caracas.

57. Rodriguez, C.A. and J.E. Bonilla. Pruebas en Programación Extrema. in IV SIMPOSIO INTERNACIONAL DE SISTEMAS DE INFORMACIÓN E.

58. Sanz, L.F., Tutorial: pruebas funcionales y trabajo en equipo. Universidad Europea de Madrid, Grupo de calidad de software de ATI, 2007.

59. Yoandy Pérez Villazón, Metodología para la migración a Software Libre de las universidades del Ministerio Superior (MES). Universidad de las Ciencias Informáticas. 2008.
60. Lisandra Cala Hernández, Propuesta de Integración y Nuevas Herramientas para el Desarrollo de La Plataforma Cubana de Migración a Software Libre. Universidad de las Ciencias Informáticas. 2010.
61. ProactivaNET® Inventario. Disponible en: < http://www.proactivanet.com/proactivanet-inventario > .
62. ProactivaNET® Service Desk. Disponible en: < http://www.proactivanet.com/proactivanet-service-desk> .
63. Integración con el inventario de red. Disponible en: < http://www.proactivanet.com/integracion-con-el-inventario-de-red> .
64. ProactivaNET® Service Desk - Gestión de incidencias. Disponible en: < http://www.proactivanet.com/proactivanet-service-desk-gestion-de-incidencias> .
65. ProactivaNET® Service Desk - Gestión de Problemas. Disponible en: < http://www.proactivanet.com/proactivanet-service-desk-gestion-de-problemas> .
66. ProactivaNET® Service Desk – Gestión de Cambios y Entregas.

Glosario de términos

Chequeo a demanda: En el presente trabajo se denomina chequeo a demanda al tipo de chequeo de según peticiones de una aplicación externa.

Detección de fallas: En el presente trabajo se emplea el término detección de fallas para denominar la detección de una falla en sistemas de servidores y equipos de interconexión.

Umbrales: Se emplea el término umbrales para definir los valores críticos entre los cuales un parámetro está en correcto funcionamiento.

Árbol jerárquico: Se emplea el término para hacer referencia a la forma en que se organizan los parámetros configurados de un sistema determinado.

Metadatos: Se hace uso de este término para definir un conjunto de datos que esclarecen las funcionalidades de un elemento específico.