Universidad de las Ciencias Informáticas

Facultad 6

Sistema integrado para la monitorización de MongoDB y CouchDB

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

Autor(es): Pedro Enrique Hernández Zamora
Mario Luis Fuente Hernández

Tutor(es): MsC. Anthony R. Sotolongo León
Ing. Rosnel Venero Acosta

La Habana, Cuba
"Año 56 de la Revolución"
Junio de 2014

El verdadero hombre no mira de qué lado se vive mejor, sino de qué lado está el deber.

JOSÉ MARTÍ

Declaración de Autoría

Declaramos ser los únicos autores de la presente tesis y reconocemos 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 _____.

_____	_____
Pedro Enrique Hernández Zamora	**Mario Luis Fuente Hernández**
Firma del Autor	**Firma del Autor**
_____	_____
MsC. Anthony R. Sotolongo León	**Ing. Rosnel Venero Acosta**
Firma del Tutor	**Firma del Tutor**

Datos de Contacto

Autor:
Pedro Enrique Hernández Zamora
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: pehernandiz@estudiantes.uci.cu

Autor:
Mario Luis Fuente Hernández
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: mlfuente@estudiantes.uci.cu

Tutor:
MsC. Anthony Rafael Sotolongo León
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: asotolongo@uci.cu

Tutor:
Ing. Rosnel Venero Acosta
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: rvacosta@uci.cu

Agradecimientos

Quiero darle gracias a dios por ser mi fuerza y mi refugio en los momentos que pensaba que no iba a graduarme. A nuestro comandante en jefe Fidel Castro y a la Revolución por darme la oportunidad de estudiar en esta maravillosa universidad. A mis tutores Anthony y Rosnel por guiarme y aconsejarme de la mejor manera posible y por su tiempo y dedicación. Agradecerle también al tribunal de tesis y a todos los profesores que contribuyeron en mi formación profesional. A todos mis compañeros que de una u otra forma me han ayudado especialmente Eduar, Marcos Michel, Yadian Juan Carlos y Darián que más que un amigo es como mi hermano. Quiero agradecer por encima de todo a mi familia que no ha dejado nunca de darme su apoyo, y por hacerme saber que nunca voy a estar solo, especialmente a mi madre por su preocupación, sus consejos, por estar siempre presente ante cualquier situación, por su cariño amor y comprensión por ser madre y padre al mismo tiempo en mi vida, y por muchas cosas que no alcanzarían las palabras para mencionar. A mis otros padres Mary y Roberto por educarme y forjarme como una persona de bien y demostrarme que todo en la vida tiene su momento que debemos luchar para alcanzar las metas que nos proponemos y que todo sacrificio tiene su recompensa. A mi hermana Thalía por ser mi alegría y la razón de mi esfuerzo por ser un mejor ejemplo cada día para ella. A mi abuela Rosa por quererme tanto y siempre apoyarme en todo. A mi tía Mariolis por sus consejos y su preocupación porque siempre hiciera las cosas correctamente. A mi otro hermano Eduardo por ser mi ejemplo y por su preocupación por que todo me saliera bien A todos los que de una forma u otra han hecho posible la realización de este sueño de muchos años Muchas Gracias.

Pedro Enrique.

Dedicatoria

Dedico este trabajo de diploma a las personas que de alguna manera han marcado mi vida de manera especial, a Dalgis por ser la persona más importante en mi mundo, por creer en mí siempre, porque siempre vas a estar a mi lado en cualquier situación de la vida, por traerme al mundo y darme la vida por todo esto estaré eternamente agradecido y por ser tu hijo, nunca podré pagarte todo lo que haz echo por mí. A mi padre Leonides, a mi abuela Sonia y a mi primo Osmany, a ellos que no se encuentran físicamente pero siempre estarán en mi corazón, yo sé que desde el cielo están conmigo en cada paso que doy, en cada decisión y en cada pensamiento, que dios los tenga en la gloria. También dedico este trabajo a toda mi familia que siempre me ha apoyado en las buenas y las malas y nunca han dudado y siempre han tenido fe en mi de que puedo lograr lo que me proponga.

Pedro Enrique.

Resumen

La necesidad de almacenar la información en forma digital y bajo niveles de seguridad imprescindibles le ha dado un protagonismo esencial a las bases de datos. Entre sus clasificaciones se encuentran las bases de datos NoSQL orientadas a documentos, entre las que se pueden mencionar MongoDB y CouchDB. A pesar de la eficiencia del manejo de los datos con estas aplicaciones existe, como inconveniente, el desbordamiento del uso de algunos recursos del sistema donde se ejecutan. Una manera de prevenir estas sobrecargas es a través de la monitorización de algunas métricas que son definidas de acuerdo a las necesidades identificadas.

El sistema integrado para la monitorización de MongoDB y CouchDB está enfocado a prevenir los inconvenientes, que pueden causar, el desconocimiento de la utilización de algunos recursos del sistema por parte de los Sistemas Gestores de Bases de Datos MongoDB y CouchDB. Está concebido como una aplicación web a través de la cual el administrador de bases de datos o algunos usuarios específicos puedan ver el comportamiento de las bases de datos y ser notificados sobre la ocurrencia de posibles errores o la detección de algún umbral alcanzado.

Palabras clave: bases de datos NoSQL, Sistemas Gestores de Base de Datos, métricas.

Abstract

The need to store information in digital form and under security essential has given an essential role to the databases. Its rankings are NoSQL document-oriented data among which may be mentioned MongoDB and CouchDB. Despite the efficiency of data management with these exist, as drawback, the overflow using some system resources where running applications. One way to prevent these surges is through monitoring certain metrics that are defined according to identified needs.

The integrated monitoring MongoDB and CouchDB system is aimed at preventing problems that can cause, ignorance of the use of certain system resources by Management Systems Databases MongoDB and CouchDB. It is designed as a web application through which the databases administrator or some specific users can see the behavior of the database and be notified of the occurrence of errors or detection of a threshold reached.

Keywords: NoSQL databases, Management Systems Databases, metrics.

Índice de Contenido

Introducción 1
Capítulo 1: Fundamentación teórica 8
1.1. Introducción 8
1.2.1. Monitorización 8
1.2.2. Sistema de gestión de bases de datos (SGBD) 9
1.3. Sistemas de gestión de bases de datos NoSQL 9
1.3.1. Sistema de gestión de bases de datos MongoDB 11
1.3.2. Sistema de gestión de bases de datos CouchDB 11
1.4. Herramientas que monitorizan servidores de aplicaciones y bases de datos 12
1.4.1. Nagios XI ™ 13
1.4.2. Munin 14
1.4.3. Análisis crítico 14
1.5. Definición de métricas para el proceso de monitorización de los gestores MongoDB y CouchDB 16
1.5.1. Definición de métricas 17
1.6. Tecnologías y herramientas propuestas para el desarrollo de la solución 22
1.6.1. Lenguaje unificado de modelado 22
1.6.2. Herramienta de modelado 23
1.6.3. Lenguaje de programación Python 23
1.6.4. Lenguaje de programación PHP 25
1.6.5. Framework para el desarrollo 26
1.6.6. Entorno de desarrollo integrado (IDE) 29
1.7. Metodología de desarrollo de software 30
1.7.1. XP (eXtreme Programming) 31
1.8. Conclusiones del capítulo 33
Capítulo 2: Diseño e implementación de la solución 34
2.1. Introducción 34
2.2. Descripción de la solución 34
2.2.1. Solución propuesta 34
2.3. Diseño del sistema 36

Índice de Contenido

2.3.1. Historias de usuario....36
2.3.2. Lista de reserva del producto....38
2.3.3. Plan de iteraciones....40
2.3.4. Tarjetas CRC....41
2.3.5. Modelo conceptual....41
2.3.6. Patrones de arquitectura....43
2.3.7. Patrones de diseño....44
2.4. Implementación del sistema....45
2.4.1. Tareas de ingeniería....45
2.4.2. Estándares de codificación....46
2.5. Conclusiones del capítulo....48
Capítulo 3: Validación de la solución....49
3.1. Introducción....49
3.2. Validación del sistema....49
3.2.1. Pruebas unitarias....49
3.2.2. Pruebas de aceptación....50
3.2.3. Técnicas de prueba....51
3.3. Casos de pruebas basados en Historias de Usuario....52
3.4. Análisis de los resultados del proceso de pruebas....54
3.5. Conclusiones del capítulo....55
Conclusiones....56
Recomendaciones....57
Referencias bibliográficas....58
Bibliografía....62
Glosario de Términos....66
Anexos....67

Índice de Figuras

Fig. 1 Métrica para los procesos en el servidor con Munin. 17
Fig. 2 Métrica para medir el consumo de CPU en el servidor con Munin. 18
Fig. 3 Métrica para medir el consumo de la memoria RAM en el servidor con Munin. 18
Fig. 4 Métrica para medir las entradas y salidas al disco duro en el servidor con Munin. 18
Fig. 5 Estructura interna del sistema de monitorización. 36
Fig. 6 Diagrama conceptual. 43
Fig. 7 Patrón experto 44
Fig. 8 Patrón creador 44
Fig. 9 Patrón bajo acoplamiento 45
Fig. 10 Patrón alta cohesión 45
Fig. 11 Ejemplo de indentación de código. 47
Fig. 12 Ejemplo de imports. 47
Fig. 13 Ejemplo de comentario en línea. 47
Fig. 14 Ejemplo de comentario en bloque. 47
Fig. 15 Ejemplo de nombre de función. 47
Fig. 16 Ejemplo de nombres de variables. 48
Fig. 17 Resultado de las pruebas por iteración. 55

Índice de Tablas

Tabla. 1 HU Capturar el consumo de memoria RAM en el servidor. 37
Tabla. 2 Lista de reserva del producto. 38
Tabla. 3 Plan de iteraciones. 40
Tabla. 4 Tarjeta CRC de la clase Monitorización. 41
Tabla. 5 Tarea de Ingeniería de HU 1. 46
Tabla. 6 PA de la HU 1 53
Tabla. 7 Ejemplo de no conformidades detectadas. 54

Introducción

Los sistemas de información[1] han creado un alto nivel de dependencia en el mundo, repercutiendo considerablemente en la gestión de los datos[2]. Unido a esto, el auge alcanzado por las tecnologías ha provocado que los sistemas informáticos sean cada vez más empleados en todas las esferas de la sociedad. Los beneficios de la informática, como una rama de la ciencia que posibilita automatizar los procesos de una institución, han permitido el crecimiento gradual de la productividad y disminuye, a su vez, el costo de las actividades productivas. Estas ventajas hacen que el uso de las Tecnologías de la Información y las Comunicaciones, en lo adelante TIC, esté presente en gran parte del mundo.

> *"Las TIC se definen colectivamente como herramientas que las personas usan para compartir, distribuir y reunir información, y comunicarse entre sí, o en grupos, por medio de las computadoras o las redes de computadoras interconectadas. Se trata de medios que utilizan tanto las telecomunicaciones como las tecnologías de la computación para transmitir información."* (Romaní, 2009)

Es una realidad que las TIC están cada vez más presente en las prácticas cotidianas, producto de la ya referida masificación de dispositivos, así como de la disminución de sus costos e incremento de sus capacidades (Romaní, 2009). Esta diversificación crea una relación estrecha entre estos elementos y la gestión de los datos, en todos sus términos, influyendo en el almacenamiento digital de la información. Algunas entidades dedicadas a la mercadotecnia y el análisis de los datos que se gestionan a través del uso de las TIC, muestran lo relevante de esta relación.

El flujo de datos generado cada año crece de forma exponencial, "según un informe de *International Data Corparation*[3] *(IDC)*, en 2007 existían 281 exabytes (281.000 millones de gigabytes) de datos. Por su parte

[1] Conjunto organizado de datos, que constituyen un mensaje sobre un determinado ente o fenómeno.

[2] Son símbolos que describen hechos, condiciones, valores o situaciones. Pueden asociarse dentro de un contexto para convertirse en información

[3] International Data Corporation (IDC) es una corporación que se dedica a la investigación de mercado de América, análisis y consultoría especializada en tecnologías de la información, telecomunicaciones y tecnología de consumo.

la consultora *McKinsey Global Institute*[4] *(MGI)* informó en 2011 sobre el aumento del volumen de información a escala mundial, en 2010 se generaron y almacenaron más de 6 500 peta bytes de datos (aproximadamente 6 500 millones de gigabytes) y se estima que para el 2020 el mundo va a generar 50 veces la cantidad de información existente" (Group, 2014).

Por la velocidad en que se incrementan los datos, cada día se hace más difícil el manejo de grandes volúmenes de información en todos los sectores. Por tal motivo es necesario contar con la tecnología capaz de almacenar, procesar y analizar la información de manera rápida y eficiente, con el menor costo posible. Comúnmente para el almacenamiento y control de la información se utilizan las bases de datos que según la Ing. María Pinto Molina se definen como "recursos que recopilan todo tipo de información, para atender las necesidades de un amplio grupo de usuarios. Su tipología es variada y se caracterizan por una alta estructuración y estandarización de la información" (Molina, 2011).

Dentro de sus disímiles categorías se encuentran las bases de datos relacionales y las NoSQL. Estas últimas son definidas por el Ing. William Sepúlveda como:

> *"Las bases de datos* NoSQL *son un conjunto de bases de datos que no tienen esquemas, no usan SQL como lenguaje de consulta ni permiten joins*[5]*, no intentan garantizar la propiedad ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad)*[6]*, escalan horizontalmente, hacen uso amplio de la memoria principal del computador, resuelven el problema de los altos volúmenes de información y la inmensa cantidad de consultas y transacciones diarias" (Sepúlveda, 2013).*

Este tipo de bases de datos, a su vez, se distinguen en diferentes tipos de acuerdo a la forma en que se organizan. Entre ellas se encuentran las orientadas a documentos que "son libres de esquemas, almacenan diferentes tipos de información en las mismas estructuras y, se auto-describen en la estructura de cada

[4]McKinsey Global Institute o McKinsey&Company es una consultora global que se focaliza en resolver problemas concernientes a administración estratégica. McKinsey trabaja prestando sus servicios a las mayores empresas de negocios del mundo, gobiernos e instituciones.

[5] Sentencia que permite combinar registros de dos o más tablas en una base de datos relacional.

[6]Hace referencia a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción

documento usando JSON[7]. Entre ellas se encuentran MongoDB y CouchDB" (Yudisney Vazquez Ortíz, 2013).

Entre las bases de datos NoSQL orientadas a documentos una de las más usadas a nivel mundial según el ranking de *DB-engines* es MongoDB. Especialmente para la gestión de contenidos y la entrega de información algunas de las empresas que utilizan esta herramienta son: *MTV Networks, SourceForge, SunlightLabs y Omnidimensional*. En el caso de la gestión de datos de usuario se pueden mencionar a *Copper Egg Reveal Cloud, Appboy, ChannelMeter* y *CityBus*. Además en la industria y las noticias la usan el periódico *The guardian, Examiner.com, Forbes, The New York Times, The Chicago Tribune, CNN Turk y Business.com* (IT, 2014).

En Cuba, la Universidad de las Ciencias Informáticas (UCI), creada en el año 2002, ejerce como centro de estudios y a su vez como empresa productora de software. Está compuesta por diferentes centros de desarrollo que poseen perfiles tecnológicos afines a los productos software que implementan. Dentro de ellos se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC) compuesto por diferentes departamentos y uno de ellos es PostgreSQL. En el mismo se emplean bases de datos relacionales entre ellas PostgreSQL y se han identificado para el trabajo y el almacenamiento de la información las NoSQL MongoDB y CouchDB, esta última es la segunda en al ranking de DB-engines.

Las bases de datos, como pieza fundamental de una infraestructura que gestione volúmenes elevados de información, especialmente desde el punto de vista digital, necesitan como un paso de vital importancia la correcta monitorización de métricas relacionadas a ellas. La monitorización se realiza enfocada a dos aristas fundamentales: En el servidor para chequear la integridad de los datos y al gestor de bases de datos con el fin de evaluar los tiempos de respuesta de las consultas, el espacio ocupado en disco, el nivel de uso de memoria, la cantidad de conexiones que se realizan, entre otros.

Con el objetivo de conocer cómo se evaluaban esas métricas en el departamento de PostgreSQL se realizó una entrevista a los especialistas del departamento para conocer sobre las herramientas de monitorización existentes. Durante el estudio se observó que para las bases de datos NoSQL, como caso puntual de la investigación las herramientas para monitorizar este tipo de actividades poseen varias dificultades para su

[7] Formato ligero para el intercambio de datos.

uso integrado a varios gestores NoSQL. Esto puede provocar desconocimiento respecto al consumo de los recursos del sistema y de posibles fallas de los Sistemas Gestores de Base de Datos (SGBD).

Afecta factores como la disponibilidad de la información, pues desconocer que la capacidad de almacenamiento del disco duro del servidor ha llegado a su límite provocaría que no se almacenen los datos generados después de alcanzado ese umbral. El consumo desmedido de los recursos del procesador podría paralizar el funcionamiento del SGBD e impedir que se acceda a la información en el momento que se necesite. En cuanto al desempeño de las bases de datos se analizó que en caso de que los SGBD requieran más memoria RAM de la que posee el servidor donde se ejecutan podría hacer más lento los procedimientos a realizar, como son el almacenamiento de información y la petición de datos. De esta manera se comprometen la productividad del negocio y la satisfacción del usuario final.

Conocer el estado de estos recursos le permite al administrador de bases de datos prever posibles errores antes que ocurran y tomar medidas sobre ellos. El envío de alertas programadas o notificaciones automáticas ayuda a mantener la información disponible en todo momento y agiliza el funcionamiento de los servidores de bases de datos. Para dar solución a la problemática planteada se ha identificado para la presente investigación el **problema a resolver:** ¿Cómo monitorizar bases de datos NoSQL orientadas a documentos?

Se define como **objeto de estudio:** El proceso de monitorización de bases de datos.

Enmarcado en el **campo de acción:** El proceso de monitorización de bases de datos NoSQL orientadas a documentos.

Para guiar la presente investigación se define como **objetivo general:** Desarrollar un sistema que permita la monitorización de bases de datos NoSQL orientadas a documentos.

Para el cumplimiento del objetivo de la investigación se definen los siguientes **objetivos específicos:**

1. Definir el marco teórico conceptual y el estado del arte referente a las herramientas para la monitorización de MongoDB y CouchDB.
2. Realizar el análisis y diseño del sistema para la monitorización de MongoDB y CouchDB.
3. Implementar el sistema para la monitorización de MongoDB y CouchDB.
4. Realizar pruebas al sistema para la monitorización de MongoDB y CouchDB.

Además de los objetivos específicos se plantea el problema a través de una serie de **preguntas de investigación**:

- ¿Cuáles serían las bases para definir el marco teórico conceptual y el estado del arte referente a las herramientas para la monitorización de MongoDB y CouchDB?
- ¿Cómo guiar el diseño del sistema a través de la realización de artefactos orientados por una metodología de desarrollo?
- ¿Cómo caracterizar el desarrollo de la propuesta de solución y los resultados de las pruebas efectu adas para validar las funcionalidades implementadas?
- ¿Cuáles son las herramientas y tecnologías más adecuadas para desarrollar el sistema de monitorización para MongoDB y CouchDB?
- ¿Qué pruebas son más factibles aplicar para la validación del sistema a desarrollar?

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

1. Definición del marco teórico conceptual de la investigación sobre la monitorización de las bases de datos NoSQL orientadas a documentos.
2. Descripción del estado del arte de los principales sistemas homólogos referentes a las herramientas para la monitorización de las bases de datos NoSQL orientadas a documentos.
3. Análisis de las tecnologías y herramientas a utilizar en la construcción de la solución propuesta.
4. Análisis y diseño de los artefactos generados por la metodología seleccionada.
5. Elaboración de los artefactos diseñados según lo establecido por la metodología seleccionada.
6. Definición de métricas para la implementación del sistema para la monitorización de MongoDB y CouchDB.
7. Implementación del sistema para la monitorización de MongoDB y CouchDB.
8. Diseño y ejecución de los casos de pruebas al sistema para la monitorización de MongoDB y CouchDB.

Con el fin de guiar el proceso de elaboración de la investigación propuesta se utilizaron como **métodos científicos**:

Métodos teóricos

- **Histórico – lógico:** Se utilizó en la introducción para elaborar el contenido conceptual de la investigación y denotar de forma estructurada los temas a tratar y la relación entre ellos. Además de brindarle a los autores de la presente investigación los datos necesarios para comprender las bases teóricas de la investigación de manera consecuente. Se realizará una investigación que permitirá tener conocimiento de los gestores NoSQL más utilizados a nivel mundial con el objetivo de realizar una selección de estos para realizar la monitorización de los mismos. También se investigará sobre la existencia de herramientas capaces de realizar este proceso.

- **Analítico – sintético:** Se utilizó en el capítulo de fundamentación teórica para seleccionar las herramientas y tecnologías adecuadas para el correcto desarrollo de la solución propuesta, así como para representar de manera breve sus principales características. A través de este método se realizó un estudio preciso de alguno de los sistemas homólogos y se seleccionó la metodología de desarrollo a utilizar. Será utilizado para realizar el análisis de toda la información que se obtenga durante todo el proceso de desarrollo de la investigación y el mismo permitirá obtener una síntesis de los contenidos fundamentales que puedan resultar relevantes para este trabajo de diploma.

- **Modelación:** Se utiliza para seleccionar en que parte del objeto de estudio se basará el análisis del marco teórico de acuerdo al nivel de importancia dentro de la investigación. Es el método mediante el cual se crean abstracciones para representar la realidad compleja, como por ejemplo diagramas o figuras que representen el problema o la solución para un mejor entendimiento.

Métodos empíricos

- **Observación:** Se utilizó para conocer la manera en que trabajan las bases de datos NoSQL en el departamento de PostgreSQL, a través de la elaboración de indicadores que permitieron conocer las particularidades del proceso de monitorización.

- **Entrevista:** Se utilizó para detectar las problemáticas existentes en el departamento de PostgreSQL respecto a la monitorización de las bases de datos NoSQL, a través de preguntas realizadas a los especialistas del departamento.

La presente investigación se encuentra estructurada en tres capítulos divididos de la siguiente manera:

Capítulo 1: Fundamentación teórica

En este capítulo se realiza la fundamentación teórica sobre las herramientas de monitorización de bases de datos NoSQL orientadas a documentos. Se abordan los principales conceptos asociados al problema general y se realiza un estudio de las herramientas y técnicas existentes que permiten seleccionar las que serán utilizadas para la implementación de la solución que se propone. Además se identificarán las métricas que serán utilizadas para realizar una monitorización eficiente del desempeño de las bases de datos.

Capítulo 2: Diseño e implementación de la solución

En este capítulo se presenta la solución que se propone, evidenciando las funcionalidades del sistema y los datos generales que apoyen el entendimiento por parte de los usuarios, desarrolladores y clientes. Además se manifiestan de forma clara los requisitos funcionales y no funcionales que están establecidos para que el sistema cumpla con las expectativas del cliente que la solicita. Se presentarán las Historias de Usuario, Tareas de Ingeniería, las Tarjetas CRC (Clase, Responsabilidad, Colaboración) y el modelo conceptual como parte de los pasos imprescindibles en el diseño del sistema propuesto.

Capítulo 3: Validación de la solución

En este capítulo se investigarán las diferentes estrategias de pruebas definidas por la metodología XP, que permiten verificar el correcto funcionamiento del sistema. Se diseñan y aplican las pruebas que le serán realizadas a la aplicación. Las funcionalidades del sistema serán sometidas a diferentes escenarios de prueba y se comprobará que sus resultados son los esperados y en caso contrario serán corregidos para la entrega final.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En este capítulo se realiza la fundamentación teórica sobre las herramientas de monitorización de bases de datos NoSQL orientadas a documentos. Se abordan los principales conceptos asociados al problema general y se realiza un estudio de las herramientas y técnicas existentes que permiten seleccionar las que serán utilizadas para la implementación de la solución que se propone. Además se identificarán las métricas que serán utilizadas para realizar una monitorización eficiente del desempeño de las bases de datos.

1.2. Principales conceptos

1.2.1. Monitorización

Monitorizar puede verse como una forma de prevensión de riegos. ”El término monitorizar se asocia a la acción de controlar o supervisar cuidadosamente una actividad durante un tiempo acordado, además de vigilar el funcionamiento de un sistema, servicio o actividad. Existen dos tipos de monitorización” (Martinez, 2011):

- **Ad Hoc:** Monitorización específica en caso de problemas o pruebas. Se utiliza generalmente para investigar una situación puntual en la que se intenta encontrar una explicación a un suceso, cambio o problema.
- **Preventiva:** Detecta interrupciones de servicios, alerta sobre posibles problemas y crea gráficos con tendencias y datos históricos sobre los sistemas. Este tipo de monitorización está automatizada y ayuda a descubrir cambios en los sistemas que provocan o pueden provocar problemas en un futuro cercano.

La Real Academia de la Lengua Española define monitorizar como “la acción de observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías.”

Cohen & Franco en su libro Gestión social: cómo lograr eficiencia e impacto en las políticas sociales, expresan que: “la monitorización es una comparación de lo que se va realizando con lo que se ha programado, es muy importante que se esté siempre observando el proceso.”

Luego de citados y analizados algunos conceptos los autores de la investigación coinciden con lo expresado por el Dr. Emilio Coni al definir monitorización como:

> *"El seguimiento rutinario de programas usando los datos de los insumos, los procesos y los resultados obtenidos. Se utiliza para evaluar si las actividades programadas se están llevando o no a cabo en el tiempo y forma establecidos. Las actividades de monitorización revelan el grado de progreso del programa hacia las metas identificadas" (Dr.Emilio Coni, 2007).*

1.2.2. Sistema de gestión de bases de datos (SGBD)

> *"Es el conjunto de programas que permiten definir, manipular y utilizar la información que contienen las bases de datos para realizar todas las tareas de administración necesarias para mantenerlas operativas, mantener su integridad, confidencialidad y seguridad"* (Sánchez, 2012).

Un SGBD debe cumplir al menos con las siguientes características:

- **Definición de los datos:** debe ser capaz de recibir los datos en versión fuente y convertirlo en objetos.
- **Manipulación de datos:** debe atender las solicitudes de los usuarios para extraer, actualizar adicionar y suprimir datos.
- **Seguridad e integridad de los datos:** debe supervisar las solicitudes de los usuarios y rechazar los intentos de violar las medidas de seguridad e integridad definidas por el administrador de bases de datos.

Existen varios SGBD en el mundo con diferentes características e indicados a sectores distintos, ya sea por su robustez, por el nivel de carga de datos que posee o por las posibilidades de adquisición que presenten, en dependencia de la licencia bajo la cual se distribuyen. Entre sus clasificaciones se encuentran los relacionales y NoSQL, siendo estos últimos los de interés para la presente investigación.

1.3. Sistemas de gestión de bases de datos NoSQL

Los sistemas de gestión de bases de datos NoSQL son una categoría general de SGBD que difiere de los modelos relacionales clásicos en diferentes modos (Artaza, 2012):

- No requieren esquemas de información fija.
- Evitan las operaciones join.

- Escalan horizontalmente.

Las bases de datos NoSQL están concebidas para obtener una altísima capacidad de volumen de almacenamiento y velocidad de procesamiento de la información. Las ventajas más significativas de la arquitectura de datos NoSQL son (Leo-Revilla, 2013):

- **Escalabilidad:** Se pueden escalar con relativa facilidad ante demandas puntuales de sobrecarga de datos.
- **Rendimiento:** Se hace necesario añadir más recursos en la plataforma hardware o priorizar cuales son los servicios más críticos en cada momento.
- **Activación/Desactivación:** Debido a la naturaleza distribuida de los datos, los modelos NoSQL responden muy bien ante la activación/desactivación de los servicios en base a las necesidades puntuales de demanda por parte de los usuarios o del mismo sistema.

De manera general es válido aclarar que el surgimiento de las bases de datos NoSQL no significa el fin para las bases de datos relacionales. De manera diferente cada una de ellas se debe usar según las características de la entidad que las utiliza y los datos que se manejan, atendiendo a las particularidades que ofrecen. Un ejemplo de ello es que cuando el volumen de datos crece en momentos precisos, las necesidades de procesos específicos no se pueden prever o cuando existen picos de usos de sistema por usuarios en varias ocasiones entonces la mejor alternativa para gestionar la información es las bases de datos NoSQL.

Las bases de datos NoSQL se clasifican principalmente en cuatro grupos (Sepúlveda, 2013):

- De clave Valor.
- Documentos
- Familia de columnas
- Grafos.

La presente investigación está enfocada especialmente a las bases de datos NoSQL orientadas a documentos basándose en el tipo de información que se maneja en el departamento de PostgreSQL. Este tipo de base de datos permite la gestión de información semi-estructurada orientadas a documentos direccionados por una clave única posibilitando recuperar su contenido.

1.3.1. Sistema de gestión de bases de datos MongoDB

> *"MongoDB es una base de datos de código abierto utilizada por diversas empresas, en todos los sectores y para una amplia variedad de aplicaciones. Se trata de una base de datos ágil construida para la escalabilidad, el rendimiento, la alta disponibilidad, la ampliación de las implementaciones, desde servidores individuales hasta grandes y complejas arquitecturas multi-sitio. Proporciona, además, un alto rendimiento tanto para lecturas como para la escritura de los datos" (Mongodb.org, 2013).*

Características principales:

- JSON[8] para modelo de datos con esquemas dinámicos.
- Auto-Sharding[9] para Escalabilidad Horizontal.
- Built-In[10] de replicación para alta disponibilidad.
- Búsqueda de texto.
- Marco de agregación.
- Integración Hadoop[11].
- Conductores: 13 Controladores soportados por MongoDB; 37 Controladores soportados por la Comunidad.
- Puede utilizarse en diferentes y múltiples ámbitos: archivos, infraestructura nube, gestión de contenidos, comercio electrónico, educación, juegos, almacenamiento de metadatos, estadísticas en tiempo real, redes sociales, entre otras.

1.3.2. Sistema de gestión de bases de datos CouchDB

> *"CouchDB es una base de datos que abarca completamente la web. Almacena sus datos a través de documentos JSON y realiza las Consultas, combinación y transformación de los datos con JavaScript. CouchDB funciona bien con la web moderna e incluso para aplicaciones móviles. Permite un alto nivel de distribución de datos o aplicaciones, de manera eficiente, mediante la replicación incremental que brinda este sistema de gestión de base de datos NoSQL. La transformación de*

[8] Notación literal de objetos de JavaScript para el intercambio de datos.
[9] Proceso de almacenar registros de datos a través de múltiples máquinas.
[10] Permite distintos niveles de acceso de acuerdo a las necesidades del sistema de base de datos,
[11] Framework de software que permite a las aplicaciones trabajar con miles de nodos y petabytes de datos.

documentos sobre la marcha y las notificaciones de cambios en tiempo real hace que el desarrollo de aplicaciones web sea más fácil. CouchDB es altamente disponible y tolerante a la partición, pero también es consistente con el tiempo. Una de sus grandes ventajas es poseer un motor de almacenamiento tolerante a fallos que pone la seguridad de sus datos en primer lugar" (apache.Couchdb.org, 2013).

Características principales:

- **Permite Relajarse:** busca que las personas entiendan fácilmente los conceptos y el funcionamiento de la herramienta.
- **Sin esquema:** brinda gran flexibilidad ya que en el mundo real no todas las cosas poseen las mismas propiedades ni siguen un modelo tan estricto.
- **Replicación:** adicionalmente soporta la replicación en múltiples bases de datos.
- **Mejora la latencia con datos locales**: tiene la capacidad de sincronizarse con una base de datos central para que en caso de que se pierda la conexión con el servidor los datos aun persistan en el cliente.

1.4. Herramientas que monitorizan servidores de aplicaciones y bases de datos

La monitorización permite coleccionar mediciones de rendimiento que permitan analizar la eficiencia de los sistemas y los servicios brindados. Al realizar la monitorización en los Sistemas Gestores de Bases de Datos se pueden detectar algunas anomalías y posibles interferencias que ocurren mientras se ejecuta alguna acción. La detección a tiempo de un problema permite que se corrija el procedimiento antes de que llegue a ocurrir la falla, en consecuencia con el posible error. En la prevención de estas problemáticas el administrador de bases de datos juega un papel de suma importancia. Entre sus tareas principales están la de realizar la monitorización de todos los sistemas a los cuales controla y supervisa para saber cómo se está desarrollando su funcionamiento, planificar lo que se debe hacer y conocer las modificaciones y actualizaciones que necesita realizar en el sistema.

Existen herramientas capaces de adaptarse a monitorizar complejas aplicaciones, permitiendo facilitar y hacer más sencillo el servicio de monitorización. Algunas de estas herramientas están basadas en la monitorización específica de sistemas operativos, servidores de aplicaciones, virtualización y plataformas distribuidas. Además existen algunas que proporcionan estadísticas de rendimiento de los servidores de bases de datos, la web y servicios de red más comunes.

Con la intención de identificar las principales funcionalidades de un sistema de monitorización y conocer algunas de las herramientas, dedicadas a estas tareas, se realizó un estudio de los sistemas homólogos. No se puede precisar que los sistemas citados sean los únicos pero si mencionar que son las herramientas de monitorización más utilizadas en diferentes sectores.

1.4.1. Nagios XI ™

> *"Es una poderosa solución de monitorización de infraestructura de TI (Tecnologías de la Información) en el mercado. Posee control a nivel empresarial y soluciones de alerta que proporcionan a las organizaciones, con una visión ampliada de su infraestructura de TI, los problemas antes que afecten a los procesos críticos del negocio" (Rojas, 2012).*

Nagios XI proporciona a las organizaciones muchos beneficios, entre ellos (Rojas, 2012):

- Monitorización Integral de la Infraestructura TI.
- Conocimiento.
- Planificación proactiva.
- Personalización.
- Facilidad de uso.
- Arquitectura extensible.

Nagios Enterprise posee versiones de software y documentación bajo diferentes licencias. La licencia específica para cualquier paquete de software determinado se indicará en la documentación del software, el código fuente, o la información adjunta (Enterprises, 2014).

Los precios de las licencias se determinan por el número de hosts (nodos) que se tiene la intención de controlar. Los términos de las diferentes licencias son (Enterprises, 2014):

- **Licencia de código abierto:** vigilará hasta siete (7) anfitriones / nodos con servicios limitados. Sólo se tiene que seleccionar la licencia libre en la interfaz de administración de Nagios XI.Los servicios de apoyo no se incluyen cuando se utiliza una licencia libre.
- **Licencia de compra:** reciben acceso especial a un servicio de asistencia por correo electrónico sólo para clientes que incluye hasta diez (10) incidentes de soporte de cada año, en función del nivel de licencia comprada. Incluye soporte técnico, descargas para clientes, formación, entre otras ventajas.

1.4.2. Munin

> *"Es una herramienta orientada al trabajo en red de monitorización histórica de recursos que puede ayudar –a través de la graficación–a analizar tendencias en su uso, y a bosquejar cómo desarrollar agentes de monitorización específicos a elementos no contemplados por un desarrollo genérico. Entre sus criterios de diseño se encuentra el de ofrecer una instalación muy simple y al mismo tiempo facilitar a sus usuarios el desarrollo de agentes recolectores de información para adecuarlo a sus necesidades" (Iszaevich, 2011).*

Se divide en tres componentes principales (Iszaevich, 2011):

1. **Servidor:** El único proceso que se ejecuta constantemente en todas las máquinas monitorizadas. Su función es responder a las solicitudes del cliente, y configurar y llamar a los plugins[12], manejando todos los aspectos relacionados con la comunicación en red. El servidor debe ejecutarse con privilegios de súper usuario, para poder lanzar a cada uno de los plugins con los privilegios que requieran. Al servidor se hace referencia también como munin-node, al ser el componente que se ejecuta en todos los nodos.
2. **Plugins:** Cada uno de los agentes de recolección de datos que son invocados por munin-node. Entregan los niveles de la información que monitorizan, y son también capaces de describir su función y configuración.
3. **Cliente:** Proceso que se ejecuta periódicamente (típicamente cada 5 minutos) desde un nodo central, interrogando a cada uno de los servidores, y generando las páginas web estáticas.

1.4.3. Análisis crítico

En el caso de Nagios, a pesar de que el sistema posee ventajas a tener en cuenta por su estabilidad y potencia, está enfocado a la monitorización de los recursos de la red. Esto hace que su utilización como herramienta para monitorizar servidores de base de datos brinde resultados muy pobres de acuerdo a la necesidad de conocer el uso de recursos del sistema, por no ser esta su especialidad. En cuanto a su obtención, aun cuando puede distribuirse en ocasiones bajo una licencia libre, se tendría que limitar la cantidad de servidores a monitorizar y no se contaría con mantenimiento y soporte para el sistema. En el caso de la licencia de compra se debería también especificar la cantidad de nodos para los cuales se utilizará

[12] Es un complemento en una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

y pagar de acuerdo a ello. Todo esto interfiere en la utilización de Nagios como la herramienta para monitorizar las bases de datos NoSQL.

Munin por su parte si permite monitorizar las bases de datos a través de la creación de plugins, enfocándose especialmente al trabajo que estas realizan en la red. Se pueden mencionar, entre otros, los plugins que posibilitan que se puedan monitorizar las acciones que se realizan en el SGBD de PostgreSQL, Oracle y MySQL. Para las bases de datos como MongoDB y CouchDB los plugins que existen solo permiten establecer conexiones con ellos a través de conexión SSL[13]. Además es importante señalar que Munin no emite alertas, simplemente se dedica a la detección de irregularidades en intervalos específicos de tiempo.

La monitorización de servidores de bases de datos también puede ser realizada por herramientas específicas que trae el propio SGBD. En el caso de MongoDB existe la funcionalidad mongostat que es un comando que brinda reportes sobre la actividad en el gestor, dentro de la información que brinda se encuentra la gestión de documentos, la cantidad de consultas y comandos así como la cantidad de conexiones y el tráfico de red. Los conocedores de estos procesos no aconsejan su uso pues el trabajo a través de ellos es a nivel de consola. Esto trae como inconveniente que dificulta el trabajo de los administradores por su complejidad, haciéndolo más lento. Por su parte en el gestor CouchDB se pueden conocer reportes sobre la actividad en el mismo accediendo a la dirección http://localhost:5984/_stats, pero no proporciona en sus reportes gran cantidad de información sobre el desempeño del gestor, no obstante se puede conocer información como las lecturas y escrituras en las bases de datos así como la cantidad de vistas leídas. Estas herramientas por sus características no son las más idóneas para monitorizar un conjunto de servidores de forma centralizada.

A nivel internacional existen varios sistemas que se dedican a la monitorización, aunque no lo hacen de manera específica para bases de datos NoSQL. En el caso de los sistemas encontrados se detectaron la falta de disponibilidad, para su estudio, de documentación reflejada de manera oficial, obstaculizando el proceso de análisis de ellos. En el estudio de estas herramientas, se observó que están enfocadas a otras áreas de monitorización como las redes, no contando con una configuración apropiada para la monitorización de bases de datos NoSQL. Junto a esto se denota la necesidad del cliente de poseer un software de monitorización propio, creado a partir de métricas definidas específicamente para los aspectos que se necesitan controlar. Es por ello que se decidió crear una herramienta que abarque las necesidades

[13] Proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

reales del departamento de PostgreSQL para monitorizar las bases de datos NoSQL. No obstante, del estudio realizado se tomaron ideas generales que podrían contribuir a la creación del sistema de monitorización para MongoDB y CouchDB que se propone.

1.5. Definición de métricas para el proceso de monitorización de los gestores MongoDB y CouchDB

Para realizar una monitorización eficiente y adecuada a servidores de bases de datos, es de suma importancia definir métricas, con el objetivo de verificar su rendimiento y desempeño. A través de las métricas se deben realizar estudios comparativos entre los resultados obtenidos, mostrando un estado de mejora o decadencia y optimizar así los procedimientos en el servidor. El uso de métricas proporciona un dato cuantitativo sobre el proceso de monitorización, dando estimaciones sobre el estado de los diferentes servicios de las bases de datos en los servidores y su comportamiento de manera general.

La definición de métricas permitirá, además, establecer patrones de comportamiento para los servidores de bases de datos que serán monitorizados, partiendo de que estas no pueden interpretarse por sí solas sino que lo hacen a través de indicadores. Estos son de gran importancia y tienen gran utilidad porque sirven de base para cuantificar conceptos medibles a métodos cuantitativos de evaluación o predicción y a su vez ofrecen información para la toma de decisiones.

Las diversas clasificaciones que se le otorgan a las métricas van aparejado de las necesidades particulares y las prioridades que tenga cada administrador y cada empresa, otorgándole mayor importancia a las operaciones y aplicaciones de misión crítica para asegurar el correcto funcionamiento de los servidores de bases de datos. Algunos de los elementos más importantes para los que se pueden definir métricas y que se suelen monitorizar son (Martinez, 2011):

- **CPU:** Carga del sistema y uso de la CPU (Unidad Central de procesamiento).
- **Servidor:** Disponibilidad y posibles problemas del hardware.
- **Memoria:** Carga y uso de la memoria, uso de la memoria de intercambio (swap).
- **Red:** Disponibilidad de los componentes de red, tráfico de entrada y salida.
- **Discos / almacenamiento:** Espacio utilizado, *I/O (Input/Output)*[14] del sistema.

[14] Se refiere a los procesos de entrada (Input) y salida (Output).

- **Bases de datos:** Número de conexiones, número de transacciones, transacciones abiertas, bloqueos, espacio usado, tipo de comandos usados.

La definición de las métricas para el desarrollo del sistema para la monitorización de MongoDB y CouchDB se basó en la clasificación realizada anteriormente de los tipos de métricas más utilizadas. Además se estudiaron las métricas que realizan las herramientas analizadas en los sistemas homólogos y que tienen relación con el sistema propuesto. También se tuvo en cuenta algunas de las métricas que se miden en las herramientas que traen por defecto los gestores. El resto de las métricas se realizaron a partir de la solicitud realizada por los especialistas del departamento de PostgreSQL.

1.5.1. Definición de métricas

A pesar que el sistema Munin está orientado al análisis de los recursos respecto a la red, utiliza algunas métricas generales que poseen una amplia utilización y se encuentran validadas a nivel internacional. Las seleccionadas fueron (Wiesel, 2014):

- Métrica para el análisis de procesos en el servidor.
- Métrica para el análisis del consumo de CPU.
- Métrica para el análisis de las entradas y salidas en el disco duro del servidor.
- Métrica para el análisis del consumo de la memoria RAM.

Las siguientes figuras muestran las métricas mencionadas anteriormente capturadas por munin.

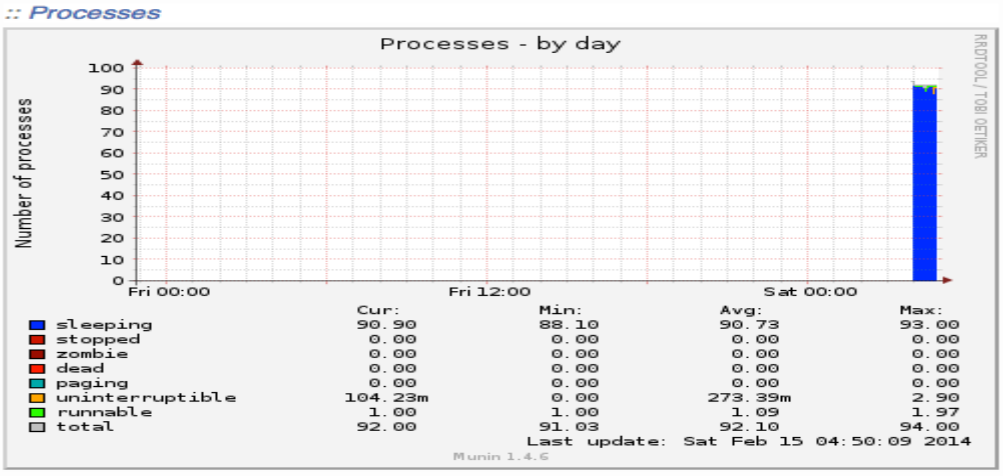


Fig. 1 Métrica para los procesos en el servidor con Munin.

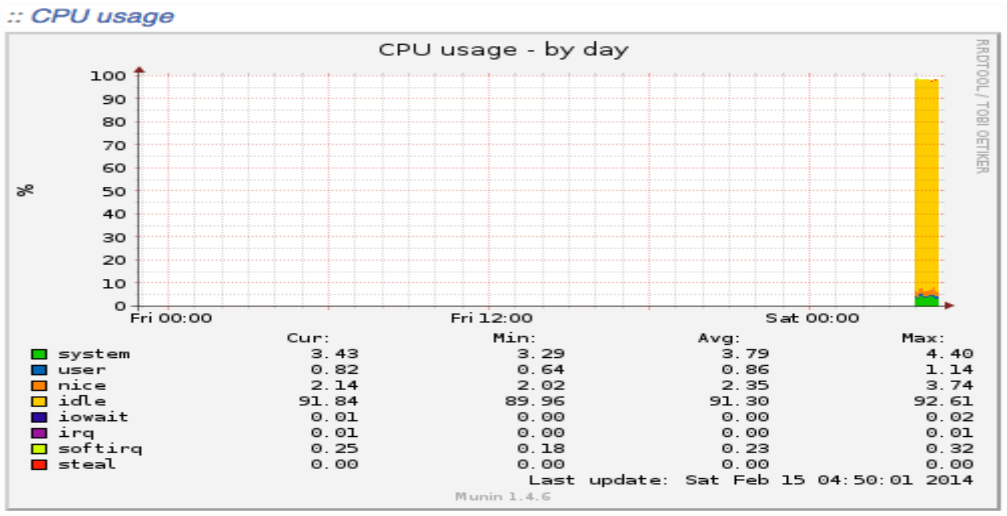


Fig. 2 Métrica para medir el consumo de CPU en el servidor con Munin.

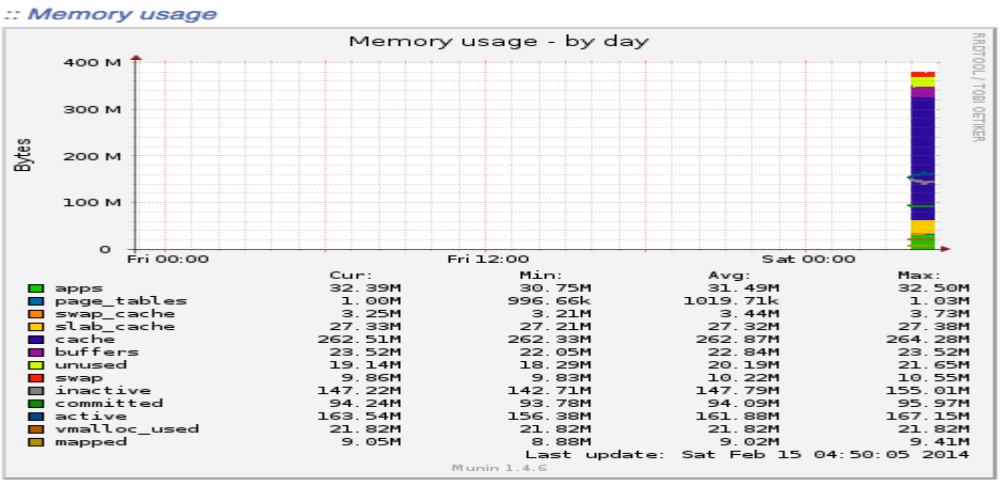


Fig. 3 Métrica para medir el consumo de la memoria RAM en el servidor con Munin.

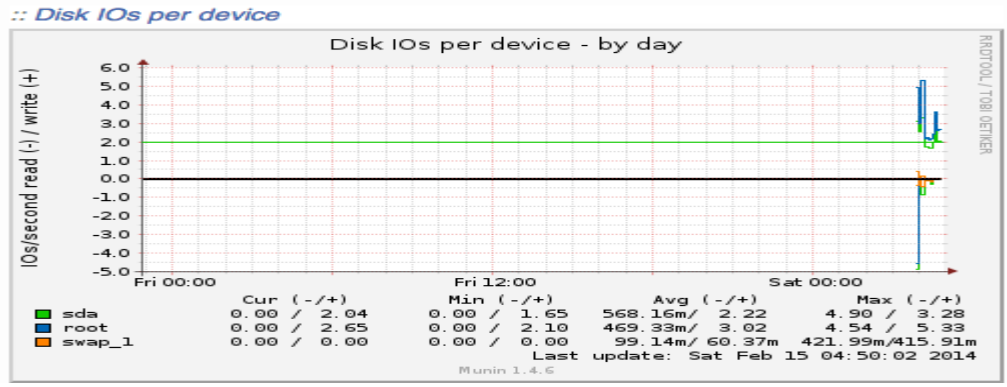


Fig. 4 Métrica para medir las entradas y salidas al disco duro en el servidor con Munin.

En el caso de Nagios las métricas identificadas fueron también las relacionadas con la utilización del procesador y memoria del sistema. Además de validar el análisis de las conexiones lógicas, lo que a pesar de ser para la red, puede ser modificada para la monitorización de las bases de datos.
Teniendo en cuenta lo anteriormente planteado se definen las siguientes métricas para aplicar al sistema a desarrollar, clasificadas en dos categorías fundamentales las métricas generales y las de bases de datos.

Métricas generales:

- **Uso de espacio en disco.**
  - **Descripción:** Esta métrica se refiere al porciento de utilización del disco duro.
  - **Indicadores:**
    1. Espacio libre en disco.
    2. Espacio en uso por el sistema.
    3. Espacio total.
- **Consumo de memoria RAM.**
  - **Descripción:** Esta métrica se refiere al porciento de utilización de la memoria RAM.
  - **Indicadores:**
    1. Espacio libre en memoria.
    2. Espacio total de memoria.
    3. Espacio usado por el sistema.
- **Consumo de memoria swap.**
  - **Descripción:** Esta métrica se refiere al porciento de utilización de la memoria swap.
  - **Indicadores:**
    1. Espacio libre.
    2. Espacio usado.
    3. Espacio total.
- **Uso del CPU.**
  - **Descripción:** Esta métrica se refiere al porciento de utilización del CPU.
  - **Indicadores:**
    1. Porciento utilizado por cada CPU.
- **Estadística de E/S de datos en el disco duro**

- **Descripción:** Esta métrica se refiere a las acciones de entrada y salida realizadas en el disco duro.
- **Indicadores:**
    1. Datos leídos.
    2. Datos escritos.

Métricas de bases de datos:

Métricas comunes para ambos gestores:

- ✓ **Cantidad de bases de datos.**

  Descripción: Esta métrica se refiere a la cantidad de bases de datos del gestor, proporcionando una estadística sobre el crecimiento diario de las mismas.
- ✓ **Procesos activos.**

  Descripción: Esta métrica se refiere a la cantidad de procesos activos por el gestor, proporcionando la cantidad de procesos que se encuentran en ejecución.
- ✓ **Cantidad de documentos por bases de datos.**

  Descripción: Esta métrica se refiere a la cantidad de documentos por bases datos en el gestor, proporcionando un reporte detallado sobre el número de documentos que posee cada base de datos.
- ✓ **Cantidad total de documentos.**

  Descripción: Esta métrica se refiere a la cantidad total de documentos en el gestor, proporcionando un reporte detallado sobre el número total de documentos que existe en el mismo.

➢ **Métricas para el gestor MongoDB.**

- ✓ **Cantidad de conexiones.**

  Descripción: Esta métrica se refiere a la cantidad de conexiones al gestor MongoDB, proporcionando el número de conexiones abiertas que presenta el gestor.
- ✓ **Estadística de documentos.**

  Descripción: Esta métrica se refiere a la cantidad de documentos insertados, actualizados y eliminados en el gestor MongoDB, proporcionando un reporte detallado sobre las operaciones de inserción, actualización y eliminación que se realizan en el gestor.
- ✓ **Cantidad de consultas.**

Descripción: Esta métrica se refiere a la cantidad de consultas realizadas en el gestor MongoDB, proporcionando un reporte detallado sobre el número de consultas que se realizan en el gestor.

- ✓ **Cantidad de comandos.**

  Descripción: Esta métrica se refiere a la cantidad de comandos realizados en el gestor MongoDB, proporcionando un reporte detallado sobre el número de comandos que se realizan en el gestor.

- ✓ **Cantidad de cursores abiertos.**

  Descripción: Esta métrica se refiere a la cantidad de cursores abiertos del gestor MongoDB, proporcionando un reporte detallado sobre el número de cursores que se han abierto en el gestor.

- ✓ **Tráfico de datos.**

  Descripción: Esta métrica se refiere a la cantidad de datos de entrada y salida en el gestor MongoDB, proporcionando un reporte detallado sobre los bytes de entrada y de salida en el gestor contribuyendo al conocimiento del tráfico de datos existente en el mismo.

- ✓ **Cantidad de colecciones por bases de datos.**

  Descripción: Esta métrica se refiere a la cantidad de colecciones por bases datos en el gestor MongoDB, proporcionando un reporte detallado sobre el número de colecciones que posee cada base de datos en el gestor.

- ➢ **Métricas para el gestor CouchDB.**
  - ✓ **Estadística de peticiones.**

    Descripción: Esta métrica se refiere a la cantidad de peticiones http (Put, Delete, Get) en el gestor CouchDB, proporcionando un reporte detallado sobre las peticiones http put, delete y get que se realizan en el gestor.

  - ✓ **Estadística de bases de datos.**

    Descripción: Esta métrica se refiere a la cantidad de bases de datos abiertas, lecturas y escrituras en las bases de datos del gestor CouchDB, proporcionando un reporte detallado sobre la cantidad de bases de datos abiertas, la cantidad de veces que se lee y la cantidad de veces que se escribe en las bases de datos del gestor.

  - ✓ **Cantidad de vistas leídas.**

Descripción: Esta métrica se refiere a la cantidad de vistas leídas en el gestor CouchDB, proporcionando un reporte detallado sobre el número total de vistas leídas en el gestor.

1.6. Tecnologías y herramientas propuestas para el desarrollo de la solución

El estudio de las tecnologías y herramientas es imprescindible para el desarrollo de cualquier aplicación. Seleccionar las adecuadas según las necesidades del equipo de desarrollo y las características del software a implementar es vital para el correcto cumplimiento de los requisitos. A continuación se describen, luego de una valoración previa con el cliente del departamento de PostgreSQL, las que serán usadas para el diseño e implementación del sistema para la monitorización de MongoDB y CouchDB.

1.6.1. Lenguaje unificado de modelado

> *"El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales" (James Rumbaugh, 1998).*

UML no pretende seguir paso a paso el desarrollo de un sistema, sino que se presenta como un lenguaje de modelado de propósito general. Su objetivo se centra, precisamente, en brindar una forma tan simple de modelar que pueda ser útil para diseñar cualquier tipo de sistema y que cualquier modelador, sea cual fuese su experiencia en el área, pudiera utilizarlo como vía para visualizar los productos ideados.

Este lenguaje se divide fundamentalmente en 5 áreas conceptuales (James Rumbaugh, 1998):

- **Estructura estática:** Cualquier modelo preciso debe primero definir el universo del discurso, esto es, los conceptos clave de la aplicación, sus propiedades internas, y las relaciones entre cada una.
- **Comportamiento dinámico:** Hay dos formas de modelar el comportamiento. Una es la historia de la vida de un objeto y la otra son los patrones de comunicación.
- **Construcciones de implementación:** Los modelos de UML tienen significado para el análisis lógico y para la implementación física.

- **Organización del modelo**: Los paquetes son unidades organizativas, jerárquicas, y de propósito general de los modelos de UML.
- **Mecanismos de extensión:** No importa cuán completo sea el conjunto de "facilidades" de un lenguaje, la gente querrá hacer extensiones.

1.6.2. Herramienta de modelado

> *"Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación" (González, 2012).*

Alguna de las características representativas de esta herramienta son (Zapata, 2005):

- Posee dos tipos de licencia: Gratuita y Comercial.
- Permite modelar productos de calidad.
- Soporta aplicaciones web.
- Escrito en varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones

1.6.3. Lenguaje de programación Python

> *"Python es un lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas. Python permite dividir un programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándares que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay*

módulos incluidos que proporcionan E/S[15] de ficheros, llamadas al sistema y sockets" (Rossum, 2000).

Entre las principales características de Python se encuentran (Antonio Arauzo Azofra, 2004):

- **Sintaxis simple:** evita muchos problemas.
- **Intérprete:** respuesta inmediata.
- **Indentado de código:** buenas prácticas programación.
- **Rápida expansión:** Alguien con experiencia en programación lo aprende en unas horas.
- **Es multiplataforma:** GNU/Linux, Windows, Mac, PalmOS, WindowsCE, RiscOS, VxWorks.
- **Extensible:** C, C++, Fortran, Java (Jython).
- **Incrustable:** Se puede incorporar en otras aplicaciones.

En la lista que cada año se realiza sobre la utilización de los lenguajes de programación en el mundo Python ha logrado ubicarse en lugares alentadores gracias a las múltiples ventajas que posee (Antonio Arauzo Azofra, 2004):

- Lenguaje de alto nivel.
- Permite centrarse en el problema específico.
- Desarrollo más rápido.
- Código muy legible.
- Facilidad de mantenimiento.
- Potente.
- Es Software Libre, incluyendo librerías, códigos de ejemplo.
- Gran cantidad de librerías para múltiples usos.
- Gran documentación.
- Buen soporte.
- Hay lenguajes diseñados para ser fáciles de aprender y hay lenguajes enfocados al desarrollo profesional de aplicaciones. Python es ambas cosas.
- Soporta diferentes paradigmas de programación (funcional, orientada a objetos).

[15] Entrada y salida.

Se seleccionó el lenguaje Python para la realización del daemon[16] (demonio en español) a través del cual se realizará la captura de los datos de configuración de los SGBD en la solución que se propone. Esta decisión se basó en la particularidad de que un proceso de este tipo requiere gran potencia, por su característica de ejecución continua. Python le brindara mayor robustez y un alto nivel de estabilidad que posibilitará que la captura de los datos que manejará el daemon sea más precisa.

1.6.4. Lenguaje de programación PHP

> *"PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor" (Martínez, 2002).*

PHP es un lenguaje de programación dirigido a la creación de páginas web dinámicas. Es un lenguaje de programación procedural con una sintaxis similar a la del leguaje C, aunque puede utilizarse una sintaxis de programación orientada a objetos similar a la de java. Desde sus inicios, PHP ha sido posiblemente el lenguaje más utilizado en entornos de desarrollo web, y desde 2001 está situado el Top 10 del índice Tiobe[17] de lenguajes de programación (Marco, 2013).

Cuando se realiza un análisis para seleccionar un lenguaje de programación de script existen algunas pautas a tener en cuenta de manera especial. A continuación se da una representación del funcionamiento de PHP en cada una de ellas (Mariño, 2008):

- **Velocidad:** Es importante no crear demoras en las computadoras por esta razón no debe requerir muchos recursos del sistema. PHP se integra muy bien con otro software, especialmente en ambientes Unix, cuando se configura como módulo de Apache, ya está listo para usarse.
- **Estabilidad:** PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** PHP provee diferente niveles de seguridad, estos pueden ser configurados desde el archivo.ini.
- **Simplicidad:** Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

[16] Es un tipo especial de proceso informático no interactivo, es decir, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario.
[17] Es una medida de la popularidad de los lenguajes de programación, calculado a partir de número de resultados en los motores de búsqueda.

- **Conectividad:** PHP dispone de una amplia gama de librerías, y agregarle extensiones es muy fácil. Esto le permite a PHP ser utilizado en muchas áreas diferentes, tales como encriptado, gráficos, XML y otras.

Entre las ventajas propias que posee el lenguaje se encuentran (Mariño, 2008):

- PHP se ejecuta en (casi) todas las plataformas utilizando el mismo código fuente, pudiendo ser compilado y ejecutado de manera independiente al Sistema Operativo.
- Es completamente expansible.
- Posee muchas interfaces distintas para cada servidor web.
- Puede interactuar con muchos motores de Base de dato como MySQL, Oracle, PostgreSQL, CouchDB, MongoDB, entre otras.
- Una gran variedad de módulos cuando el programador necesite una interface para una librería en particular, fácilmente podrá crear una API[18] (Application Programming Interface, por sus siglas en inglés) para esta.
- PHP es OpenSource, lo cual significa que el usuario no depende de una compañía para arreglar cosas que no funcionan, además no está forzado a pagar actualizaciones anuales para tener una versión que funcione.

La solución que se propone esta ideada para la web por lo cual la selección de un correcto lenguaje de programación es vital. PHP se seleccionó porque ofrece un amplio nivel de configuración, permitiendo a los usuarios acceder y modificar aquellas funcionalidades que no se adecuen a sus necesidades. El proceso de instalación es fácil y ligero pues no consume muchos recursos del sistema y entre los SGBD que soporta se encuentran CouchDB y MongoDB. Además posee una amplia comunidad de desarrolladores que posibilitan eliminar bugs[19](error de software) de manera rápida.

1.6.5. Framework para el desarrollo

> *"Un framework[20] simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona*

[18] Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
[19] Es un error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado.
[20] Marco de trabajo.

estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas" (Fabien Potencier, 2008).

Los framework no son específicos sino que se crean de acuerdo a lenguajes de programación y la cantidad de funcionalidades que se deseen. Esto implica que una vez seleccionado un lenguaje específico el framework debe estar en concordancia con este. Además, en el caso del desarrollo de aplicaciones web deben seleccionarse framework para el desarrollo del lado del servidor y otro para el desarrollo de las vistas de lado del cliente.

Symfony

"Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web" (Fabien Potencier, 2008).

Algunas de las principales características de Symfony son (Fabien Potencier, 2008):

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

Symfony es un framework que, además de estar escrito totalmente en PHP 5 y soportar gran parte de los SGBD, es muy popular. Desde su creación muchos usuarios alrededor del mundo se documentaron y comenzaron a explotar esta biblioteca. Aun cuando existen más framework para el desarrollo en la web, es este de los más conocidos. Posee gran estabilidad debido a que los colaboradores enseguida crean parches y mejoras, detectando los errores de la documentación y realizando otras tareas muy importantes.

JQuery

> *"Es una librería de JavaScript, rápida y concisa que simplifica el trabajo con documentos HTML[21]. No es un extenso e inflamado framework que prometa lo mejor de AJAX[22], ni un conjunto de innecesarias y complicadas mejoras: JQuery ha sido diseñado para cambiar la forma de escribir JavaScript[23]. Utiliza un interesante concepto para hacer código corto y simple, tiene manejadores de eventos. Otro tema que JQuery resuelve con facilidad es el de los efectos, añade dinamismo visual a la presentación del sitio, como son añadirle funcionalidad, tanto al código como al resto de los elementos" (Torres, 2011).*

Entre las características más significativas que definen a jQuery se encuentran (Leyva, 2008):

- Es orientada a eventos.
- Manipulación de la hoja de estilos CSS[24].
- Efectos y animaciones.
- Selección de elementos del DOM[25].

Sobre todas las cosas es necesario decir que la selección de este framework de presentación no se basa en su popular uso, ni siquiera en que sea el mejor. Indiscutiblemente JQuery es rechazado por algunos desarrolladores por no poseer componentes predefinidos tan buenos como los que trae consigo ExtJS, por ejemplo. Sin embargo esta "desventaja" que algunas personas le encuentran brinda un menor uso de

[21] HyperText Markup Language: define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.
[22] Acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML es una técnica de desarrollo web para crear aplicaciones interactivas.
[23] Es un lenguaje de programación interpretado que permite mejorar la interfaz de usuario en páginas web dinámicas.
[24] Cascading Style Sheets es el lenguaje de hojas de estilo utilizado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcado.
[25] Es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes JavaScript.

recursos del sistema, logrando de esta manera que las aplicaciones sean más ligeras. JQuery es también muy utilizado para realizar gráficas con mayor facilidad.

Cuando se trata de un sistema, que más allá de brindar una interfaz lo más agradable posible, necesita fortaleza, es cuando JQuery se hace necesario. En el caso de una herramienta para monitorizar bases de datos, como la que se propone, el consumo de recursos es inevitable por lo que debe evitarse sobrecargar el sistema con una aplicación pesada. Además, el hecho de que exista una funcionalidad que emita alertas provoca que la realización de gráficos sea inevitable. Es por estos motivos que se seleccionó JQuery como el framework de presentación.

1.6.6. Entorno de desarrollo integrado (IDE)

Aptana Studio es un IDE de desarrollo para aplicaciones de la web que facilita el desarrollo integrado de Ajax con las tecnologías emergentes. Está basado en el conocido entorno de desarrollo Eclipse (IDE = Integrated Development Environment), también de código abierto. Mientras Eclipse está focalizado en el desarrollo para Java, Aptana Studio es una distribución focalizada en el desarrollo web, con soporte a HTML, CSS y JavaScript, así como opcionalmente a otras tecnologías como PHP, Adobe Air o Ruby on Rails. Aptana Studio está disponible como una aplicación independiente o como plugins para Eclipse.

El programa se distribuye para todos los sistemas operativos más comunes: Windows, Linux y Mac OS X en dos versiones (Álvarez, 2007):

- **Aptana Studio Community Edition**: Es la versión gratuita, que contiene la mayoría de las funcionalidades del IDE, como edición, debugging, sincronización y administración de proyectos. Posee soporte para todas las tecnologías ya mencionadas.
- **Aptana Studio Profesional Edition**: Esta otra versión, de pago, tiene además algunas funcionalidades extras, útiles aunque no necesarias para empezar a trabajar con Aptana. Por ejemplo, la versión "Pro" incluye: Soporte para JSON, un motor de reportes, debug en Internet Explorer (la versión community sólo tiene debug en Firefox), gestión remota de proyectos y soporte para FTPS[26] y SFTP[27].

[26]Es una extensión de FTP mediante SSL para el cifrado de los datos.
[27] SSH File Transfer Protocol es un protocolo que sólo usa un canal de comunicación y envía y recibe los mensajes en código binario.

A pesar de que la edición profesional del IDE posee algunas funcionalidades extras estas no son significativas para desarrollar con Aptana. Entre las más relevantes de la edición profesional puede mencionarse la gestión remota de proyectos, pero el propio Aptana Community dispone de una herramienta para conexión por FTP[28] para el trabajo de archivos.

Las principales características de este IDE son (Aptana Inc., 2011):

- Asistente de código HTML, CSS Y JavaScript.
- Asistente de despliegue.
- Depurador integrado.
- Integración Git[29].
- Incorporado a un terminal.
- Permite personalización del IDE.

Posee además (Álvarez, 2007):

- Ayudas visuales para la escritura de scripts en diversos lenguajes, como coloreado y auto escritura del código, ayudas contextuales de referencia a medida que se escribe, entre otras.
- Visualización de errores de sintaxis a medida que se escribe.
- Soporte para hacer FTP a servidores remotos, con herramientas para sincronización.
- Debug en Firefox (Debug Internet Explorer también con la versión Profesional)
- Librerías de funciones en JavaScript populares en Ajax/Javascript para utilizar en los proyectos.
- Ejemplos ya creados para empezar a conocer las posibilidades de desarrollo rápidamente.
- Pre visualización de estilos CSS con el editor CSS.
- Extensible a partir de *plugins* que puede crear Aptana u otras empresas y herramientas para estar al tanto de cualquier nuevo añadido.
- Extensible por JavaScript. Los usuarios pueden escribir scripts para realizar acciones y macros.

1.7. Metodología de desarrollo de software

Las metodologías de desarrollo de software constituyen un elemento esencial en el desarrollo de software. Según Piattini (1996), no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto

[28] **F**ile **T**ransfer **P**rotocol es un protocolo de red para la transferencia de archivos entre sistemas conectados.
[29] Es un software de control de versiones diseñado por Linus Torvalds.

no existe una definición universalmente aceptada. Sí hay un acuerdo en considerar a la metodología como "un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software".

Por lo tanto, una metodología es un conjunto de componentes que especifican: (Cataldi, 2000)

- Cómo se debe dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué salidas se producen y cuándo se deben producir.
- Qué restricciones se aplican.
- Qué herramientas se van a utilizar.
- Cómo se gestiona y controla un proyecto.

> *"Históricamente, las metodologías tradicionales en el campo de desarrollo de software, han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. En su lugar las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas a proyectos pequeños y a la gestión de riesgos, las metodologías ágiles constituyen una solución a la medida para ese entorno" (Mitaritonna, 2010).*

La solución propuesta en este trabajo de diploma cuenta, para su construcción, con un equipo de desarrollo conformado por dos personas. El margen de tiempo para el desarrollo es relativamente corto y dado que no se cuenta con la experiencia necesaria en la implementación de este tipo de aplicaciones, se precisa entonces de una estrecha relación entre el cliente y el desarrollador. Por la complejidad de las funcionalidades, resulta imprescindible la posibilidad de regresar atrás en el proceso de desarrollo, en caso de detectarse alguna anomalía, que pudiera afectar la entrega a tiempo de una primera versión funcional. Todo esto inclina a los autores a seleccionar una metodología ágil para guiar el proceso de desarrollo del software propuesto.

1.7.1. XP (eXtreme Programming)

> *"Más que una metodología, XP se considera una disciplina, la cual está sostenida por valores y principios propios de las metodologías agiles. Existen 4 valores que sirven como pilares en el desarrollo de estas metodologías" (Luis Miguel Echeverry Tobón, 2007):*

- **La comunicación:** En XP la relación con el cliente es tan estrecha, que es considerado parte del equipo de desarrollo.

- **La simplicidad:** No utiliza recursos para la realización de actividades complejas, solo se desarrolla lo que el cliente demanda, de la forma más sencilla.
- **La retroalimentación:** Está presente desde el principio del proyecto, ayuda a encaminarlo y darle forma.
- **El coraje:** El equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios que se presentarán en el transcurso de la actividad.

Además de estos valores XP centra su desarrollo en 12 prácticas específicas que la caracterizan. Estas son (Luis Miguel Echeverry Tobón, 2007):

- El desarrollo está dirigido por pruebas.
- El juego de la planificación.
- Cliente in-situ[30].
- Programación por parejas.
- Entregas pequeñas.
- Refactorización[31] sin piedad.
- Integración continúa del código.
- Diseño simple.
- Utilización de metáforas del sistema.
- Propiedad colectiva del código.
- Convenciones de código.
- No trabajar horas extras.

Teniendo en cuenta las principales características que posee la metodología XP y comparándolas con la del equipo de desarrollo que implementará la herramienta propuesta, se corroboró la similitud en la mayoría de estas. Es por ello que los autores del presente trabajo toman como referencia para el desarrollo de la solución propuesta la metodología XP.

[30] Es una expresión latina que significa «en el sitio» o «en el lugar».
[31] Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

1.8. Conclusiones del capítulo

El análisis de algunas herramientas dedicadas a la monitorización en el mundo definió la importancia de crear una herramienta propia de la UCI y de manera más específica, para el departamento de PostgreSQL. Se definieron 23 métricas necesarias para realizar la monitorización según pautas ya establecidas tanto por sistemas homólogos como por el departamento de PostgreSQL, lo que sirvió como una guía para el desarrollo de la solución propuesta. Se seleccionó la metodología XP brindándole al desarrollo de la herramienta mayor flexibilidad y agilidad en los procesos vinculados a ella. Se utiliza el lenguaje de programación PHP lo que posibilitó mayor interactividad en la web y hacer la herramienta más sencilla y ligera. En el caso del lenguaje de programación Python, se utilizó para la elaboración del demonio de configuraciones permitiendo que sea lo suficientemente potente según las peculiaridades que un módulo de este tipo posee. Además se manifestaron las características principales de las herramientas y tecnologías vinculadas al desarrollo de la solución lo que les brindó a los autores de la investigación los conocimientos necesarios para su utilización.

Capítulo 2: Diseño e implementación de la solución

2.1. Introducción

En este capítulo se presenta la solución que se propone evidenciando las funcionalidades del sistema y los datos generales que apoyen el entendimiento por parte de los usuarios, desarrolladores y clientes. Además se manifiestan de forma clara los requisitos funcionales y no funcionales que están establecidos para que el sistema cumpla con las expectativas del cliente que la solicita. Se presentarán las Historias de Usuario, Tareas de Ingeniería, las Tarjetas CRC (Clase, Responsabilidad, Colaboración) y el diagrama conceptual como parte de los pasos imprescindibles en el diseño del sistema propuesto.

2.2. Descripción de la solución

2.2.1. Solución propuesta

En el departamento de PostgreSQL se trabaja con distintos Sistemas Gestores de Bases de Dato (SGBD). Cada SGBD tiene características distintas y funcionan, a su vez, de manera diferente. Aun así, también poseen similitudes y una de ellas es que todos realizan un trabajo pesado que consume muchos recursos del sistema. Por este motivo pueden ocurrir errores que paralicen el funcionamiento de las bases de datos, provocando que la información que contienen pierda la disponibilidad y por tanto se afecten procesos que dependen de ella para realizarse.

Una forma de prevenir estos errores es brindándole a los administradores de bases de datos una herramienta que les permita supervisar el comportamiento de los gestores respecto a los principales recursos del sistema. La manera de realizar esta tarea es a través de las herramientas de monitorización que contenga las métricas específicas que se desean evaluar. En el departamento de PostgreSQL existen algunas de estas aplicaciones pero están desarrolladas para otros SGBD. Las bases de datos CouchDB y MongoDB tienen la característica de no ser relacionales y se utilizan para propósitos distintos a los que tiene otros SGBD como los conocidos MySQL, PostgreSQL u Oracle. Es por esto que precisan de su propia herramienta de monitorización, acorde al trabajo que realizan y sus características específicas.

El sistema para la monitorización de MongoDB y CouchDB que se propone debe brindar, entre sus funcionalidades, la posibilidad de:

- Mostrar el rendimiento de los recursos del sistema:

- ✓ Uso de memoria.
- ✓ Uso del Disco de almacenamiento

- Mostrar la gestión de los documento que se realiza en los SGBD No SQL CouchDB y MongoDB
- Definir el momento en que el administrador desea o considera que se debe notificar del uso de un recurso específico.
- Notificar a los usuarios sobre errores o inconvenientes en el uso de algunos de los recursos evaluados.

El sistema de monitorización propuesto a desarrollar es el encargado de recolectar un grupo de métricas, clasificadas en generales y de bases de datos, que permitirá registrar el rendimiento de los gestores MongoDB y CouchDB. El administrador debe instalar el sistema en cada uno de los servidores a monitorizar, una vez instalado se comportará como un proceso del sistema operativo, que de acuerdo a sus características estará enviando constantemente, en el intervalo de tiempo definido por el administrador, el resultado de las métricas capturadas a la base de datos NoSQL MongoDB.

La estructura interna del sistema estará compuesta por diferentes paquetes, uno de ellos es el paquete capturador de métricas donde se va a definir cada una de las métricas e indicadores que permiten analizar el comportamiento tanto de los gestores MongoDB y CouchDB como del sistema operativo. El paquete con nombre Demonio contiene una serie de módulos encargados de definir todas las funciones que pueden realizarse con el sistema, tales como: start, stop, restart, reload, force-reload y status. El mismo se instala en cada una de las pc que serán monitorizadas desde donde se enviará la información a un servidor MongoDB al cual se accede desde la aplicación web y se muestran los reportes de las métricas al usuario.

SIM.log es un archivo o registro al que se le va añadiendo líneas a medida que se realizan acciones sobre el sistema. Las incidencias, errores de conexión, entradas de datos inválidos, cualquier problema con el sistema se guardan en el mismo. SIM.cfg es un archivo de configuración donde se definen parámetros necesarios para la conexión con el servidor MongoDB, además de parámetros que especifican que se desea monitorizar así como el intervalo de tiempo para salvar los datos en la base de datos. En la clase Principal es donde se ejecutan todas las métricas e indicadores capturados de los gestores y del sistema operativo enviando las mismas de forma concurrente hacia el servidor MongoDB. En la base de datos MongoDB se almacena toda la información capturada de cada uno de los servidores monitorizados. La Figura 5 representa gráficamente la interacción de los paquetes, módulos, archivos, clases y MongoDB.

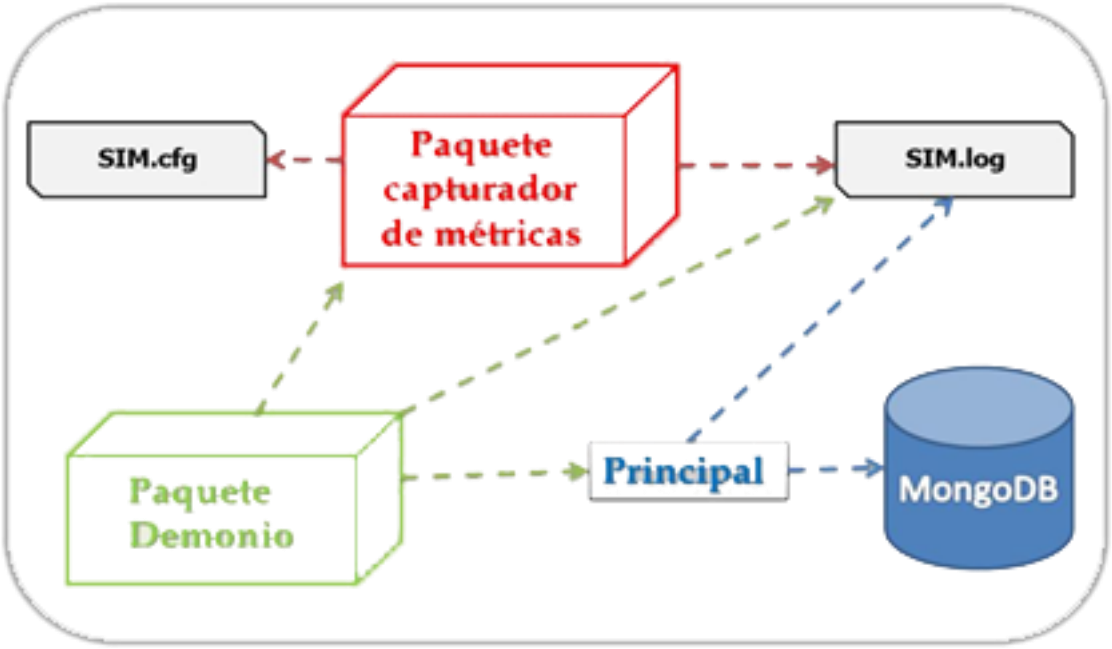


Fig. 5 Estructura interna del sistema de monitorización.

2.3. Diseño del sistema

2.3.1. Historias de usuario

La metodología de desarrollo XP comienza con la exploración. Durante esta fase se realiza el proceso de organización de lo que debe desarrollarse y como se debe hacer. Una de las principales actividades de esta fase es la identificación de las Historias de Usuario (HU) y en correspondencia las Tareas de Ingeniería. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Las historias de usuario (HU), como parte de la metodología XP, son descripciones cortas escritas en el lenguaje del usuario, sin terminología técnica. Su principal diferencia respecto a los conocidos casos de uso está en el nivel de detalle, pues estas solo brindan la información necesaria para que los programadores estimen el tiempo de desarrollo en el momento ideal. Lo principal de las HU es que recogen los puntos de vista del cliente respecto a cómo ellos valoran que debe desarrollarse en los requisitos funcionales definidos.

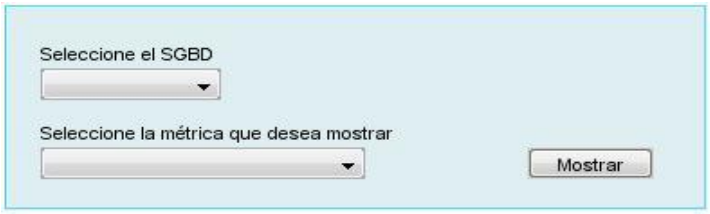
Las Historias de usuario cuentan con la siguiente estructura:

- **Número:** A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.
- **Nombre:** Nombre descriptivo de la HU.
- **Prioridad en Negocio:** Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que puede tomar son (Alta, Media o Baja).
- **Riesgo en Desarrollo:** Grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. (Alto, Medio o Bajo).
- **Puntos Estimados:** Unidades de tiempo estimadas, en semanas, por el equipo de desarrollo para darle cumplimiento a la HU.
- **Puntos Reales:** Unidades de tiempo reales, en semanas, que el equipo de desarrollo necesitó para darle cumplimiento a la HU.
- **Iteración Asignada:** Número de la iteración, de acuerdo al ciclo de desarrollo establecido, en la cual será implementada la HU.
- **Descripción:** Descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.
- **Observaciones:** Condiciones que deben tenerse en cuenta para el desarrollo de la funcionalidad.

En total se realizaron 24 HU de las cuales solo se muestran en el capítulo un ejemplo de ellas.

Tabla. 1 HU Capturar el consumo de memoria RAM en el servidor.

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Capturar el consumo de memoria RAM en el servidor.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Pedro Enrique Hernández Zamora Mario Luis Fuente Hernández	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 0.7 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.7 semanas
Descripción: Permite obtener el consumo de la memoria RAM, el cantidad de memoria libre, la utilizada por el sistema y el total de memoria.	

Observaciones: Para obtener los datos sobre el uso de la memoria RAM debe seleccionarse la métrica a través de la interfaz de la aplicación.
Prototipo de interfaz: 

2.3.2. Lista de reserva del producto

Con la metodología XP los requisitos del sistema se plasman en un documento denominado Lista de Reserva del Producto (LRP) y a través de él se definen algunos atributos para organizar el desarrollo de los requisitos como son:

- **Prioridad:** Según la importancia del requisito para el desarrollo del sistema se le asigna un nivel de prioridad que puede variar en: Muy alta, Alta, Media y Baja.
- **Ítem:** Define una numeración o identificador para el requisito planteado.
- **Descripción:** Plantea un nombre sugerente que de una idea previa de lo que significa el requisito.
- **Estimación:** Tiempo en el que se estima que se puede desarrollar el requisito planteado.
- **Estimado por:** Persona o rol dentro del equipo de desarrollo que se encarga de estimar el tiempo en el que debe desarrollarse el requisito.

Tabla. 2 Lista de reserva del producto.

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
	RF1	Capturar el consumo de memoria RAM en el servidor.	0.7 semanas	Analista
	RF2	Capturar el consumo de memoria Swap en el servidor.	0.7 semanas	Analista
	RF3	Capturar el uso de CPU en el servidor.	0.7 semanas	Analista
	RF4	Capturar el uso de espacio en disco del servidor.	0.7 semanas	Analista
	RF5	Capturar la estadística de E/S de datos en el disco duro.	0.7 semanas	Analista
	RF6	Mostrar las notificaciones o alertas.	0.7 semanas	Analista
Alta				

	RF7	Capturar la cantidad de bases de datos del gestor CouchDB.	0.5 semanas	Analista
	RF8	Capturar la cantidad de procesos activos por CouchDB.	0.4 semanas	Analista
	RF9	Capturar la estadística de peticiones http (put, delete, get) en el gestor CouchDB.	0.5 semanas	Analista
	RF10	Capturar la estadística de bases de datos en el gestor CouchDB.	0.4 semanas	Analista
	RF11	Capturar la cantidad de documentos por bases de datos en el gestor CouchDB.	0.4 semanas	Analista
	RF12	Capturar la cantidad total de documentos en el gestor CouchDB	0.4 semanas	Analista
	RF13	Capturar la cantidad de vistas leídas en el gestor CouchDB.	0.4 semanas	Analista
	RF14	Capturar la cantidad de bases de datos del gestor MongoDB.	0.4 semanas	Analista
	RF15	Capturar la cantidad de procesos activos por MongoDB.	0.4 semanas	Analista
	RF16	Capturar la Cantidad de conexiones al gestor MongoDB.	0.5 semanas	Analista
	RF17	Capturar la estadística de documentos del gestor MongoDB.	0.5 semanas	Analista
	RF18	Capturar la cantidad de consultas en el gestor MongoDB.	0.5 semanas	Analista
	RF19	Capturar la cantidad de comandos en el gestor MongoDB.	0.5 semanas	Analista
	RF20	Capturar la cantidad de cursores abiertos en el gestor MongoDB.	0.4 semanas	Analista
	RF21	Capturar el tráfico de red en el gestor MongoDB.	0.4 semanas	Analista
	RF22	Capturar la cantidad de documentos por bases de datos en el gestor MongoDB.	0.4 semanas	Analista
	RF23	Capturar la cantidad total de documentos en el gestor MongoDB.	0.4 semanas	Analista
	RF24	Capturar la cantidad de colecciones por bases de datos del gestor MongoDB.	0.4 semanas	Analista
Media				
Baja				
Requisitos No Funcionales				
	Seguridad	Garantizar que la aplicación solicite usuario y contraseña como datos credenciales necesarios para garantizar la autenticación en el sistema.		
	Rendimiento	La velocidad de procesamiento y la capacidad de respuesta de la aplicación web deben ser de 5 segundos, en dependencia del tipo de conexión que esté		

		establecida con el servidor y las características de la red.		
	Software	Para el cliente: Sistema gestor de bases de datos MongoDB y CouchDB. Se aconseja tener instalado como sistema operativo Ubuntu 12.04 o Debían 6. Se requiere tener instaladas las librerías python-couchdb,python-couchdbkit, python-pymongo,python-psutil,python, sysstat.Debe ser compatible con los navegadores Mozilla Firefox, Google Chrome, entre otros de los navegadores más usados. Para el servidor: Se aconseja tener instalado como sistema operativo Debían 6.Se requiere tener instalado el gestor MongoDB.		
	Hardware	Para el cliente: Microprocesador Pentium 4 o Superior.1 GB de RAM o superior.5 GB de espacio libre en disco. Para el servidor: Microprocesador Pentium 4 o Superior.1 GB de RAM o superior.80 a 120 GB de espacio libre en disco.		

2.3.3. Plan de iteraciones

En la metodología XP, la creación del sistema se divide en etapas para facilitar su realización. Por lo general, los proyectos constan de 3 iteraciones. Para cada iteración se define el conjunto de Historias de Usuario que, según la necesidad del desarrollador, se van a implementar. Al final de cada iteración se obtiene como resultado la entrega de los requisitos funcionales correspondientes a las HU definidas para ella. Las tareas que no se realicen en una iteración se tienen en cuenta para la próxima iteración. De manera general el tiempo que demorara la aplicación en construirse, según estimaciones, es 90 días.

Tabla. 3 Plan de iteraciones.

No	Descripción de la iteración	Orden para implementar las HU	Duración total
1	Se desarrollan 6 historias de usuario con una prioridad muy alta en su desarrollo que permiten obtener información sobre las métricas del uso del sistema.	HU_1, HU_2, HU_3, HU_4, HU_5, HU_6	4.2 semanas

2	Se desarrollan 7 historias de usuario con una prioridad alta en su desarrollo que permiten obtener información sobre las métricas para el gestor CouchDB.	HU_7, HU_8, HU_9, HU_10, HU_11, HU_12, HU_13	3.0 semanas
3	Se desarrollan 11 historias de usuario con una prioridad alta en su desarrollo que permiten obtener información sobre las métricas para el gestor MongoDB.	HU_14, HU_15, HU_16, HU_17, HU_18, HU_19, HU_20, HU_21, HU_22, HU_23, HU_24	4.8 semanas

2.3.4. Tarjetas CRC

Las tarjetas CRC permiten ver las clases como algo más que depósito de datos y conocer el comportamiento de cada una de ellas en un mayor nivel. La metodología XP estipula el uso de la misma como un artefacto obligatorio durante el desarrollo de un proyecto debido a los beneficios que aportan a los desarrolladores. A continuación se muestra la tarjeta de la clase monitorización que responde a funcionalidades de alta prioridad en el sistema.

Tabla. 4 Tarjeta CRC de la clase Monitorización.

Tarjeta CRC	
Clase: Monitorización	
Responsabilidades	**Colaboraciones**
➢ Ejecutar métricas. ➢ Comprobar conexión al servidor MongoDB.	Demonio

2.3.5. Modelo conceptual

La metodología XP describe el sistema utilizando las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Esto no implica que no se utilicen los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando su complejidad no sea alta y defina información importante. En el caso del diseño mediante diagramas la metodología utiliza el modelo conceptual como representación de la estructura del producto que se desarrolla.

Capítulo 2: Diseño e Implementación de la Solución

Cuando se desarrolla un producto software es difícil mostrarle a los usuarios e incluso a otros desarrolladores una vista previa de lo que será el sistema. En la ingeniería de software esta meta se logra a través de la realización de diagramas UML. Entre ellos se encuentra el modelo conceptual como la vía de representar lo que se desea construir en un lenguaje poco técnico. Sobre todo muestra las relaciones que existen entre las principales entidades con que se trabaja durante el desarrollo. Este tipo de modelo es usualmente utilizado cuando los requisitos pueden variar o no están claros en su totalidad. Es válido aclarar que no es un diagrama específico de cada clase o componente del sistema sino que es la vía de visualizar lo esencial para entender el comportamiento del software.

El sistema de monitorización debe ser manejado por el administrador de base de datos de la entidad donde se despliegue. Es por ello que este actor inicia las funcionalidades del sistema. El administrador es la persona que monitoriza los servidores de MongoDB y CouchDB.

El concepto de métrica se refiere a las evaluaciones que se realizan a los SGBD, para lograr conocer los datos deseados. A través de estas métricas se gestionan las notificaciones y alertas que se desean activar para monitorizar el estado de los recursos del servidor. Cada métrica definida le brinda al administrador una o más informaciones de acuerdo a la evaluación que realiza a los SGBD NoSQL.

La información, en el caso de la solución que se propone, se refiere al conjunto de datos que una o más métricas pueden brindar al administrador del SGBD. Este concepto está relacionado, además, con los SGBD pues son estos los que poseen la información que se necesita monitorizar. Los datos brindados se almacenan en el SDBD MongoDB

EL flujo general a seguir es evaluar los recursos que manejan los SGBD NoSQL a través de las métricas definidas, aportando la información necesaria que estos poseen y que permite conocer el estado de estos recursos. Las informaciones obtenidas a través de las métricas de guarda en un SGBD definido según la estructura de la herramienta de monitorización.

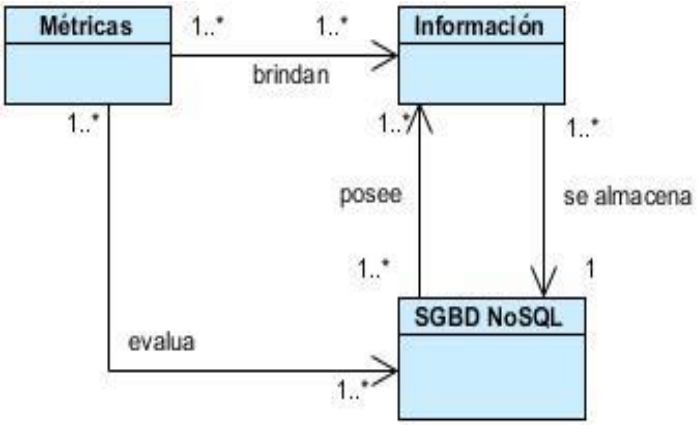


Fig. 6 Diagrama conceptual.

2.3.6. Patrones de arquitectura

Los patrones de arquitectura expresan un esquema organizativo fundamental para sistemas de software. En muchas ocasiones más allá de definir la utilización de alguno de ellos se selecciona una tecnología específica que ya posee una estructura. En el desarrollo de la solución que se propone se utiliza el framework de desarrollo Symfony. Esto indica que por correspondencia se utiliza el patrón MVC[32] para la organización del sistema y su elaboración.

El patrón de arquitectura MVC permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación. En la vista el desarrollador implementa las funcionalidades que solamente se encarga de recoger las peticiones del usuario y mostrar la información que se genera en el software. El modelo es el que lleva los datos recogidos a la base de datos y consume cada una de las opciones que sobre esta se desean realizar. El controlador, por su parte, es una capa intermedia entre el modelo y la vista. En él se encuentra la información fuerte del software y los métodos principales.

El Patrón MVC se ve frecuentemente en aplicaciones Web, donde la Vista es la página HTML y el código que provee de datos dinámicos a la página; el Modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio; el Controlador es el responsable de recibir los eventos de entrada desde y hacia la Vista.

[32] Modelo, Vista, Controlador

2.3.7. Patrones de diseño

El uso de patrones proporciona una estructura conocida para todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Los patrones GRASP[33] son los más utilizados en el entorno del desarrollo de productos informáticos. Son reconocidos por ser una serie de buenas prácticas de aplicación recomendable en el diseño de software.

Experto

El patrón experto en información es el principio básico de asignación de responsabilidades. Consiste en asignar la responsabilidad de creación de un objeto o la implementación de un método al contiene la información específica. La utilización de este patrón evita que deban solicitarse datos entre las clases de forma innecesaria.

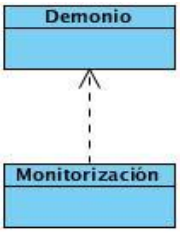


Fig. 7 Patrón experto

Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Esto posibilita que no se creen más objetos de lo necesario y por tanto evita que se sobrecargue la ejecución del software.

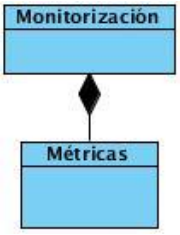


Fig. 8 Patrón creador

Bajo acoplamiento

[33] Patrones Generales de Software para Asignar Responsabilidades.

Es la idea de tener las clases lo 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, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

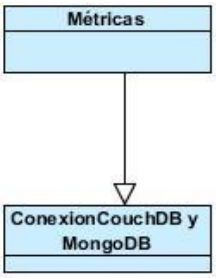


Fig. 9 Patrón bajo acoplamiento

Alta Cohesión

Se encarga de posibilitar que la información que almacena una clase deba de ser coherente y está, en la medida de lo posible, relacionada con la clase. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión, compartirá la responsabilidad de una operación con otras clases.

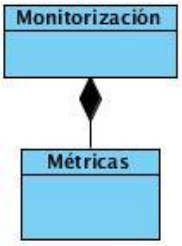


Fig. 10 Patrón alta cohesión

Diseño de la base de datos MongoDB.

2.4. Implementación del sistema

2.4.1. Tareas de ingeniería

Las Tareas de Ingeniería se realizan muy de la mano de las Historias de Usuario. Estas describen de manera más exacta los pasos que deben realizarse para organizar el trabajo de desarrollo de cada uno de los requisitos funcionales plasmados en las HU. Definen además la fecha, según el tiempo estimado, en que debe comenzar y concluir la realización de la tarea. El correcto tratamiento de este artefacto brinda al equipo

de desarrollo una visión estructurada de lo realmente necesario para obtener el producto en el tiempo especificado y con la calidad mínima deseada.

En el sistema de monitorización se definieron 24 Tareas de Ingeniería, una por cada HU. En este acápite se presenta solo un ejemplo de ellas.

Tabla. 5 Tarea de Ingeniería de HU 1.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario: HU_1**
Nombre Tarea: Implementación de la funcionalidad Capturar el uso de memoria RAM en el servidor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 0.7 semanas
Fecha Inicio: 20/01/2014	**Fecha Fin:** 24/01/2014
Programador Responsable: Pedro Enrique Hernández Zamora Mario Luis Fuente Hernández	
Descripción: A través de esta funcionalidad se obtiene el uso de la memoria RAM en el servidor de base de datos.	

2.4.2. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Este sería el mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. El estándar de codificación que se utilizó en la presente investigación es el que define el lenguaje de programación Python, debido a que este lenguaje tiene su propio estilo ante los demás lenguajes. El estándar de codificación utilizado se describe a continuación (Guido van Rossum, 2007).

Indentación: En el sistema se usan 4 espacios por cada nivel de indentación.

```
doc = col.find_one({"Reporte": date.today().strftime("%d-%m-%Y")})
if doc is None:
    col.insert(reporte)
else:
    doc.get("Metricas").get("Sistema").get("RAM").get("Uso de RAM")
```

Fig. 11 Ejemplo de indentación de código.

Imports: Los imports normalmente pueden ser colocados en distintas líneas. Se colocan siempre en la parte superior del archivo, justo después de cualquier comentario o cadena de documentación del módulo, y antes de las variables globales y las constantes del módulo. Es muy desaconsejable el uso de imports relativos para importar código de un paquete.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Deamon import Daemon
from metricas.Metricas import Metricas
import sys
import time
import os
```

Fig. 12 Ejemplo de imports.

Comentarios: Existen dos tipos de comentario en Python, el comentario de una línea y el de bloque. El comentario de línea generalmente se aplican al código que se encuentra a continuación, y se indentan al mismo nivel que dicho código. Cada línea de un comentario de bloque comienza con un # y un único espacio. Los comentarios de bloque se realizan encerrando los párrafos entre tres comillas simples.

```
#Inicializa la conexión hacia el gestor MongoDB donde se van a salvar los datos
def __init__(self):
    self.conn = pymongo.Connection(self.IP_MONGO, int(self.PUERTOM))
```

Fig. 13 Ejemplo de comentario en línea.

```
class Daemon:
        '''
        A generic daemon class.
        Usage: subclass the Daemon class and override the run() method
        '''
```

Fig. 14 Ejemplo de comentario en bloque.

Nombre de funciones: Los nombres de funciones se escriben en letra mayúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad y a partir de la segunda palabra se comienza con minúscula las demás.

```
def Uso_de_cpu(self):
```

Fig. 15 Ejemplo de nombre de función.

Nombre de variables: Las variables son declaradas con letra minúscula y de ser necesario con un guion bajo para dividir las palabras.

```
result = ruta.communicate()[0]
m_total = int(str(result.lstrip().splitlines()[3]).split()[1])
```

Fig. 16 Ejemplo de nombres de variables.

2.5. Conclusiones del capítulo

Se definieron los principales requisitos del sistema, tanto los funcionales como los no funcionales, a través de la Lista de Reserva del Producto (LRP), en la cual se documentaron 24 requisitos funcionales y 4 no funcionales, lo que permitió definir las funcionales que debe realizar la herramienta propuesta. Se describieron, además, sus características específicas a través de 24 Historias de Usuario y fue organizado el desarrollo de ellos mediante la realización de 24 Tareas de Ingeniería lo que le brindó a la solución mayor claridad y organización. Se diseñó el Modelo conceptual con el fin de dar un entendimiento no técnico acerca de los principales conceptos a tener en cuenta para entender el flujo de actividades de la herramienta. Se describieron las principales clases, utilizando 4 tarjetas CRC, para describir las funcionalidades esenciales de la herramienta. Se especificaron y ejemplificaron los patrones de diseño y de arquitectura utilizados en el desarrollo de la solución propuesta, a través de los cuales se obtuvo un nivel mayor de organización y un flujo estructurado de los datos de la herramienta.

Capítulo 3: Validación de la solución

3.1. Introducción

En este capítulo se investigaran las diferentes estrategias de pruebas definidas por la metodología XP, que permiten verificar el correcto funcionamiento del sistema. Se diseñan y aplican las pruebas que le serán realizadas a la aplicación. Las funcionalidades del sistema serán sometidas a diferentes escenarios de prueba y se comprobará que sus resultados son los esperados y en caso contrario corregirlos para la entrega final.

3.2. Validación del sistema

Es de gran importancia realizar pruebas en el proceso de desarrollo de software con el objetivo de medir la calidad del software que se desarrolle. Las mismas tienen como objetivo detectar el máximo de errores existentes tanto en el código que se implementa como en cualquier otra fase para garantizar la calidad del producto desarrollado. El proceso de pruebas se realiza de diferentes maneras teniendo en cuenta la metodología de desarrollo utilizada. Para la presente investigación se definió como metodología de desarrollo XP la cual divide el proceso de pruebas en dos grandes grupos, Pruebas Unitarias y Pruebas de aceptación.

3.2.1. Pruebas unitarias

> *"Las Pruebas Unitarias son aquellas que permiten comprobar el funcionamiento de cada componente del Sistema de Información por separado, lo que aplicado al Paradigma de la Orientación a Objetos es equivalente a hablar de pruebas que permiten comprobar el funcionamiento correcto de una Clase de modo que, para cada uno de sus métodos, o al menos para cada método no trivial, se pueden ejecutar casos de prueba independientes." (Manager, 2013)*

Algunas de las ventajas del uso de las pruebas unitarias son (Manager, 2013):

- Dan soporte a la refactorización y a la implementación de mejoras en el código.
- Simplifican la integración del proyecto.
- Sirven como documentación sobre el uso del código implementado.
- Independencia.
- Automatización.
- Completitud.

La realización de este tipo de pruebas posibilita ir eliminando errores paso a paso para no acumularlos para las pruebas de aceptación o cualquier tipo de prueba que se realice después de integrada la aplicación. Su principal objetivo es el aislamiento del código para evitar posibles errores en ellos.

3.2.2. Pruebas de aceptación

> *"Una Prueba de Aceptación, en lo adelante PA, tiene como propósito demostrar al cliente el cumplimiento de un requisito del software" (Letelier, 2007).*

Precisando un poco más, una PA (Letelier, 2007):

- Describe un escenario (secuencia de pasos) de ejecución o uso del sistema desde la perspectiva del Cliente.
- Puede estar asociada a requisitos funcionales o no funcionales
- Un requisito tiene una o más PAs asociadas
- La PAs cubren desde escenarios típicos/frecuentes hasta los más excepcionales.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que las mismas verifican y controlan el grado de aceptación del cliente con el producto que se ha desarrollado, marcando así el final de una iteración y el inicio de la próxima. El cliente final del software es quien debe desarrollar estas pruebas para corroborar que se cumplan correctamente las funcionalidades exigidas por él. (Pérez, 2007)

Estas pruebas tienen como objetivo verificar las funcionalidades, y por tal motivo, los requisitos definidos en el sistema son la principal fuente de información a la hora de construir las pruebas de aceptación. Durante una iteración la Historia de Usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar cuando una Historia de Usuario ha sido correctamente implementada. (Pérez, 2007)

Una Historia de Usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse una nueva prueba de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. La garantía de calidad es una parte esencial en la metodología XP. La realización de este tipo de pruebas y la

publicación de los resultados deben ser lo más rápido posible, para que los desarrolladores puedan realizar con la mayor rapidez posible los cambios que sean necesarios. (Pérez, 2007)

3.2.3. Técnicas de prueba

Hay dos maneras de probar cualquier producto construido, 1) si se conoce la función específica para la que se diseñó el producto, donde se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores de cada función, 2) si se conoce el funcionamiento interno del producto, se aplican pruebas para asegurarse de que "todas las piezas encajan". Es decir, que las operaciones internas se realizan de acuerdo con las especificaciones, y que se han probado todos los componentes internos de manera adecuada. Al primer enfoque de prueba se le denomina "prueba de caja negra" y al segundo "prueba de caja blanca" (Pressman, 2005).

Pruebas de caja blanca: Las pruebas de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que: 1) garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos 1 vez, 2) ejerciten los lados verdaderos y falsos de todas las decisiones lógicas, 3) ejecuten todos los bucles en sus límites y dentro de sus límites operacionales, y 4) ejerciten estructuras de datos internos para asegurar su validez (Pressman, 2005).

Pruebas de caja negra: Las pruebas de caja negra, también denominadas, "pruebas de comportamiento", se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero del software derivar un conjunto de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. Las pruebas de caja negra no es una opción frente a las técnicas de caja blanca, es, en cambio, un enfoque complementario que tiene probabilidades de descubrir una clase diferente de errores que se descubrirían con los métodos de caja blanca (Pressman, 2005).

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías: 1) funciones incorrectas o faltantes, 2) errores de interfaz, 3) errores en estructuras de datos o en acceso a bases de datos externas, 4) errores de comportamiento o desempeño, y 5) errores de inicialización y término (Pressman, 2005).

Método de prueba de caja negra

Partición equivalente: Es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. Un caso de prueba

ideal de manejo simple descubre una clase de errores, por ejemplo, (procesamiento incorrecto de todos los datos de caracteres) que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse (Pressman, 2005).

Para la realización de las pruebas al sistema desarrollado por este trabajo de diploma, se tuvieron en cuenta algunos aspectos que permitieron definir finalmente cuales pruebas eran las más idóneas para aplicar.

Entre los aspectos analizados están:

- No resulta de prioridad el analizar cómo fueron desarrolladas cada una de las funcionalidades ni la verificación de cada una de ellas.
- El desarrollo está más bien orientado a la entrega temprana de la primera versión funcional de una solución que permita a los administradores de bases de datos gestionar y conocer el comportamiento de los gestores MongoDB y CouchDB así como de la utilización de los recursos del sistema. Además está orientado a ser cumplidas correctamente todas las funcionalidades acordadas al inicio del proceso, en las que no existan errores de interfaz, de estructuras de datos o de acceso a bases de datos.

Debido a las características analizadas anteriormente para el desarrollo del sistema propuesto por este trabajo de diploma, se decide entonces utilizar como Tipo de prueba, las pruebas de aceptación, como Técnica de prueba, las pruebas de caja negra y dentro de estas el método de partición de equivalencia.

3.3. Casos de pruebas basados en Historias de Usuario

Para la ejecución de las pruebas que validarán el correcto funcionamiento de la solución, se realizará un flujo completo que comprende un caso de prueba para cada uno de los escenarios generales que dan cumplimiento a cada una de las Historias de Usuarios identificadas. Este flujo describirá el proceso necesario para desarrollar las pruebas con el objetivo de verificar cada escenario por los que está compuesto dicha solución. Para el correcto funcionamiento de cada uno de los escenarios de prueba el usuario debe seguir el flujo central de operaciones a fin de lograr el resultado de la prueba seleccionada.

La ejecución de las pruebas al sistema fueron guiados por un total de 24 casos de pruebas, los cuales fueron diseñados para probar cada una de las funcionalidades realizadas por la aplicación.

Dada la relación existente entre las Historias de Usuario y las pruebas de aceptación y el nivel de complejidad de los requisitos establecidos se decidió realizar una PA por cada HU. De esta manera quedaron conformadas 24 PA. A continuación se muestra una de ellas como ejemplo.

Tabla. 6 PA de la HU 1

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU_1-1	**Nombre Historia de Usuario:** Capturar el consumo de memoria RAM en el servidor.
Nombre de la persona que realiza la prueba: Pedro Enrique Hernández Zamora Mario Luis Fuente Hernández	
Descripción de la Prueba: El objetivo de este caso de prueba es mostrar el estado del recurso del sistema asociado a la memoria RAM, dígase espacio que consume el sistema, el espacio total y espacio que queda libre. Con esta prueba se puede conocer cómo se ejecuta esta métrica en la herramienta.	
Condiciones de Ejecución: El usuario debe ser el administrador del servidor de Bases de Datos para tener acceso al sistema y conocer de antemano la métrica que desea ejecutar.	
Entrada / Pasos de ejecución: El administrador del servidor de Bases de Datos debe entrar al sistema. Para poder ejecutar la prueba debe seleccionar la métrica que desea probar, en este caso la de capturar el consumo de la memoria RAM. Para esta métrica en específico no debe seleccionarse ningún SGBD pues se está analizando el servidor de manera general.	
Resultado Esperado: Una notificación del sistema donde muestre los siguientes datos. - Espacio de la memoria RAM total. - Espacio de la memoria RAM utilizado. - Espacio de la memoria RAM libre.	
Evaluación de la Prueba: Satisfactoria	

Las pruebas realizadas al sistema con el objetivo de verificar el correcto funcionamiento de cada una de las historias de usuario, proporcionaron resultados satisfactorios, no así las historias de usuarios detectadas con algún tipo de error, las cuales pasaron a ser no conformidades quedando documentadas en la planilla

de no conformidades del expediente de proyecto, adjunto a este trabajo de diploma. Como resultado de la aplicación de las pruebas, se detectaron un conjunto de no conformidades. Se muestran en la tabla siguiente las asociadas a la historia de usuario Capturar el consumo de memoria RAM en el servidor.

Tabla. 7 Ejemplo de no conformidades detectadas.

No.	NC	Aspecto asociado	Etapa de detección	Clasificación	Estado NC	Respuesta
1	El sistema no se conecta al gestor MongoDB.	Conexión con el gestor MongoDB	Pruebas	S	29/04/2014 PD 30/04/2014 RA	Se modificó el archivo de configuración del gestor para que el mismo pudiera aceptar conexiones externas desde otros host.

La plantilla de no conformidades recoge además los errores que son detectados durante la revisión de la documentación del sistema. Se elabora un documento por cada revisión que se haga y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba. Además de estar plasmado en la planilla de diseño de casos de prueba, estas no conformidades se van registrando en un documento aparte para luego enviarlo al equipo de desarrolladores. De igual manera, aun trabajando paralelamente con los casos de prueba, el documento emitido a los desarrolladores es independiente, porque los casos de prueba son para consumo y único uso de los probadores.

3.4. Análisis de los resultados del proceso de pruebas

Se realizó la implementación de la solución propuesta, a través de lo cual se construyó una herramienta desarrollada de manera completa que respondía a los requisitos funcionales identificados. Las funcionalidades obtenidas fueron probadas Además de realizaron pruebas de aceptación para verificar el sistema desde el punto de vista exterior, evaluando el comportamiento de la herramienta en la ejecución de

las funciones que posee. Se diseñó una PA para cada HU, lográndose un total de 24 pruebas ejecutadas en 3 iteraciones. En el gráfico de la Figura 17 Resultado de las pruebas por iteración se puede observar el proceso de desarrollo de pruebas en cada iteración así como las no conformidades detectadas en cada una de ellas.

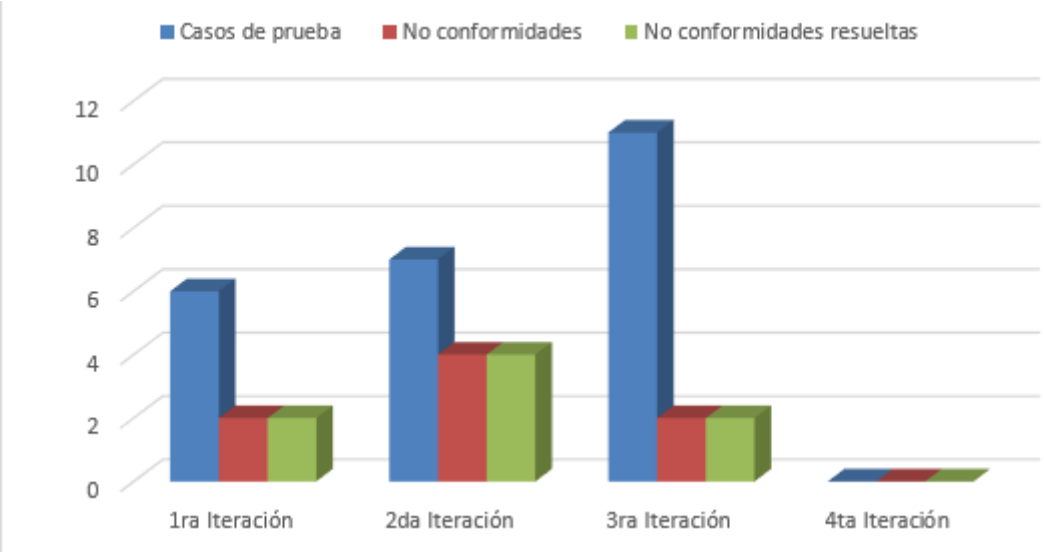


Fig. 17 Resultado de las pruebas por iteración.

La figura anterior muestra los casos de pruebas ejecutados, las no conformidades detectadas en cada etapa de iteración y al mismo tiempo las que fueron corregidas por el equipo de desarrollo, evidenciándose en la última iteración la no existencia de no conformidades terminándose la fase de pruebas con un total de 8 no conformidades detectadas las cuales fueron resueltas por el equipo de desarrollo de forma satisfactoria.

3.5. Conclusiones del capítulo

Se realizaron las pruebas definidas por la metodología XP que especifica el diseño y ejecución de las pruebas de aceptación. Las mismas se realizaron según la cantidad de historias de usuario diseñadas, comprobando la ejecución de los requisitos identificados de acuerdo a las normativas establecidas. La correcta realización de las pruebas permitió identificar varios errores que fueron corregidos, posibilitando obtener un sistema de monitorización funcional. Se aplicaron las pruebas seleccionadas permitiendo contar luego de 3 iteraciones con una versión final del sistema para la monitorización de MongoDB y CouchDB.

Conclusiones

Al concluir el desarrollo de este trabajo de diploma y cumplidos cada uno de los capítulos en los que está dividido se arriba a las siguientes conclusiones:

- El análisis de diferentes formas para monitorizar bases de datos NoSQL a través del estudio de definiciones y conceptos asociados a la investigación le permitió a los autores del presente trabajo obtener una base teórica profunda acerca del tema. Relacionado a esto se identificaron un conjunto de sistemas homólogos relacionados con el objetivo de encontrar una posible solución u obtener una guía para el desarrollo del sistema propuesto. El estudio de estas herramientas arrojó como resultado la necesidad de crear un sistema para la monitorización de bases de datos NoSQL orientadas a documentos.
- El diseño del sistema de monitorización se realizó de acuerdo a las pautas establecidas por la metodología XP para la construcción de diagramas UML, logrando obtener una visión previa de lo que debe hacer el sistema y guiar el desarrollo en la siguiente fase.
- La implementación del sistema de monitorización para los SGBD NoSQL dió como resultados el desarrollo de un conjunto de métricas de evaluación que respondieron a los requisitos funcionales establecidos por el cliente. La utilización de patrones de diseño como experto, creador, alta cohesión, bajo acoplamiento y de arquitectura como el modelo vista controlador agilizó el proceso de desarrollo y le brindó al sistema mayor organización.
- El proceso de pruebas realizado se basó en las pruebas unitarias y las pruebas de aceptación definidas por la metodología, lo que permitió corroborar el cumplimiento de los requisitos implementados y evaluar el funcionamiento del sistema.

Recomendaciones

Con el objetivo de mejorar el funcionamiento y utilidad del sistema integrado para la monitorización de MongoDB y CouchDB, además de extender el alcance de la solución se recomienda:

- Continuar con el proceso de desarrollo del sistema con el fin de incrementar la cantidad de métricas para evaluar el comportamiento de las bases de datos NoSQL orientadas a documentos de acuerdo a las necesidades que se tenga.
- Implementar un mecanismo que permita el envío de las notificaciones a través de mensajes al usuario vía correo o mensajería instantánea.

Referencias bibliográficas

1. **Alvarez, Miguel Angel. 2007.** desarrolloweb.com. [En línea] 16 de noviembre de 2007. [Citado el: 26 de noviembre de 2013.] http://www.desarrolloweb.com/articulos/aptana-studio.html.
2. **Antonio Arauzo Azofra, Ernesto Revilla. 2004.** *Una pequeña introducción a Python.* Granada : s.n., 2004.
3. **Aptana Inc. 2011.** aptana.com. [En línea] 2011. [Citado el: 02 de diciembre de 2013.] http://aptana.com/products/studio3.
4. **2011.** aptana.com. [En línea] 2011. [Citado el: 02 de diciembre de 2013.] http://aptana.com/products/studio3.
5. **Cataldi, Ing. Zulma. 2000.** *Metodología de diseño, desarrollo y evaluación de software educativo.* Argentina : s.n., 2000. ISBN 960-34-0204-2.
6. **Dr.Emilio Coni, Dr. Mario F. Chaben, Dr. Carlos Malbrán. 2007.** *Indicadores para el monitoreo y evaluación de las actividades de control de la tuberculosis.* Alemania : s.n., 2007.
7. **Romaní, Juan Cristóbal Cobo. 2009.** *El concepto de tecnologías de la información. Benchmarking sobre las definiciones de las TIC en la sociedad del conocimiento* Mexico : s.n., 2009. Vol. Vol. 14. ISSN: 1137-1102.
8. **Enterprises, Nagios. 2014.** Nagios XI. [En línea] 2014. [Citado el: 24 de enero de 2014.] http://www.nagios.com/legal/licenses/.
9. **Fabien Potencier, François Zaninotto. 2008.** *Symfony, la guía difinitiva.* 2008.
10. **Godoy, Mayreils. 2011.** *La educación inicial.* 2011.
11. **González, Lianet Cabrera. 2012.** 10, Cuba : RCCI, 2012, Vol. 5.
12. **Group, International Data. 2014.** IDC Analyze the future. [En línea] 2014. [Citado el: 19 de diciembre de 2013.] http://www.idc.com/about/about.jsp.
13. **Hellkamp, Marcel. 2014.** Bottle. *Bottle.* [En línea] 04 de febrero de 2014. [Citado el: 16 de abril de 2014.] http://bottlepy.org/.

14. **Iszaevich, Gunnar Eyal Wolf. 2011.** 1, La Habana : Ediciones Futuro, 2011, Vol. Vol.5. ISSN: 1994-1536.

15. **IT, Solid. 2014.** DB-Engine. [En línea] Solid IT, 2014. [Citado el: 13 de enero de 2014.] http://db-engines.com/en/ranking.

16. **ITI. 2010.** ITI. [En línea] Instituto Tecnlógico de Informática, 30 de diciembre de 2010. [Citado el: 14 de abril de 2014.] http://www.iti.es/servicios/servicio/resource/7240/index.html.

17. **James Rumbaugh, Ivar Jacobson, Grady Booch. 1998.** *El Lenguaje Unificado de Modelado. Manual de Refeerncias.* s.l. : Series Editors, 1998.

18. **José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2010.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : DSIC -Universidad Politécnica de Valencia, 2010.

19. **Ken Schwaber, Jeff Sutherland. 2011.** La Guía Definitiva de Scrum:Las Reglas del Juego. *La Guía de Scrum.* s.l. : Scrum. org, 2011.

20. **Lerman, Craig. 1999.** *UML y Patrones.* Mexico : s.n., 1999. 970-17-0261-1.

21. **Letelier, Patricio. 2007.** *Pruebas de Aceptación como conductor del Proceso Software.* España : Universidad Politécnica de Valencia, 2007.

22. **Luis Miguel Echeverry Tobón, Luz Elena Delgado Carmona. 2007.** *Caso práctico de la metodología ágil XP al desarrollo de software.* Colombia : s.n., 2007.

23. **Manager, Scrum. 2013.** Scrum Manager. [En línea] 23 de enero de 2013. [Citado el: 11 de 04 de 2014.] http://www.scrummanager.net/bok/index.php?title=Pruebas_unitarias.

24. **Marco, Bartolomé Sintes. 2013.** Páginas web con PHP. [En línea] 17 de septiembre de 2013. [Citado el: 16 de enero de 2014.] http://www.mclibre.org.

25. **Mariño, Carlos Vázquez. 2008.** *Programación en PHP 5. Nivel Básico.* 2008.

26. **Martínez, Rafael. 2002.** *Manual de PHP.* 2002.

27. **Martinez, Rafael. 2011.** postgreSQL.org.es. [En línea] 18 de febrero de 2011. [Citado el: 04 de diciembre de 2013.] http://www.postgresql.org.es/. 582.

28. **Mitaritonna, Ing. Alejandro Daniel. 2010.** *CAPA-3: Una innovadora metodología para el desarrollo de software en ambientes de trabajo virtuales .* Buenos Aires : s.n., 2010.

Referencias Bibliográficas

29. **Molina, Maria Pinto. 2011.** e-COMS. [En línea] 13 de abril de 2011. [Citado el: 19 de diciembre de 2013.] http://www.mariapinto.es/e-coms/bases_datos.htm.

30. **Mora, Ing. Jhon Freddy Montes. 2008.** *Puebas del software "Caja Blanca y Caja Negra".* Colombia : Universidad Nacional Abierta y a Distancia UNAD, 2008.

31. **Rodríguez, Dr. Eduardo. 2011.** *Importancia de las pruebas de software.* Mexico : CINVESTAV, 2011.

32. **Rojas, Pablo A. Muñoz. 2012.** [En línea] 2012. [Citado el: 23 de noviembre de 2013.] http://www.nagios-cl.org/nagios-xi..

33. **Rossum, Guido van. 2000.** *Guía de aprendizaje de Python.* Amsterdam : BeOpen PythonLabs, 2000. 1895.22/1012.

34. **Sánchez, Ing. Javier. 2012.** Ecuador : s.n., 2012.

35. **Sepúlveda, Ing.William Díaz. 2013.** Bases de Datos No SQL. [En línea] 27 de mayo de 2013. [Citado el: 21 de noviembre de 2013.] http://basesdedatosnosql.blogspot.com/2013/05/bases-de-datos-nosql-llegaron-para.html.

36. **Peñalver Romero, Ing. Gladys Marsi, García De La Puente, Ing. Sergio Jesus y Ing. Abel Meneses Abad, Ing. Abel. 2010.** *SXP.* Chile : s.n., 2010.

37. **UNICEF, GICHD. 2005.** *IMAS de Educación en el Riesgo de las Minas, Guía de Mejores Prácticas 7.* Ginebra : s.n., 2005. ISBN-13: 978-92-806-3978-0.

38. **Wiesel, Jonathan. 2014.** *Monitorear los recursos de un servidor utilizando Munin.* Colombia

39. **Yudisney Vazquez Ortíz, Anthony Rafael Sotolongo León. 2013.** Num. 9, La Habana : RCCI, 2013, Vol. Vol. 6. ISSN: 2306-2495.

40. **Zapata, Luis Giraldo. 2005.** 2005.

41. **Guido van Rossum, Barry Warsaw. 2007.** [Online] agosto 10, 2007. [Cited: junio 19, 2014.] http://mundogeek.net/traducciones/guia-estilo-python.htm.

42. **Pressman, Roger S. 2005.** Ingeniería del Software. Un enfoque práctico.Quinta Edición. 2005.

43. **Pérez, Maite Rodríguez Corbea Meylin Ordóñez. 2007.** *La metodología XP aplicable al desarrollo del software educativo en Cuba.* s.l. : Universidad de las Ciencias Informáticas, 2007.

Bibliografía

1. **Alvarez, Miguel Angel. 2007.** desarrolloweb.com. [En línea] 16 de noviembre de 2007. [Citado el: 26 de noviembre de 2013.] http://www.desarrolloweb.com/articulos/aptana-studio.html.
2. **Antonio Arauzo Azofra, Ernesto Revilla. 2004.** *Una pequeña introducción a Python.* Granada : s.n., 2004.
3. **Aptana Inc. 2011.** aptana.com. [En línea] 2011. [Citado el: 02 de diciembre de 2013.] http://aptana.com/products/studio3.
4. **2011.** aptana.com. [En línea] 2011. [Citado el: 02 de diciembre de 2013.] http://aptana.com/products/studio3.
5. **Cataldi, Ing. Zulma. 2000.** *Metodología de diseño, desarrollo y evaluación de software educativo.* Argentina : s.n., 2000. ISBN 960-34-0204-2.
6. **Cayuqueo, Sergio. 2014.** *Monitorización y análisis de Red con Nagios.*
7. **Dr.Emilio Coni, Dr. Mario F. Chaben, Dr. Carlos Malbrán. 2007.** *Indicadores para el monitoreo y evaluación de las actividades de control de la tuberculosis.* Alemania : s.n., 2007.
8. *El concepto de tecnologías de la información. Benchmarking sobre las definiciones de las TIC en la sociedad del conocimiento.* **Romaní, Juan Cristóbal Cobo. 2009.** Mexico : s.n., 2009. Vol. Vol. 14. ISSN: 1137-1102.
9. **Enterprises, Nagios. 2014.** Nagios XI. [En línea] 2014. [Citado el: 24 de enero de 2014.] http://www.nagios.com/legal/licenses/.
10. **Fabien Potencier, François Zaninotto. 2008.** *Symfony, la guía difinitiva.* 2008.
11. **Godoy, Mayreils. 2011.** *La educación inicial.* 2011.
12. **González, Lianet Cabrera. 2012.** 10, Cuba : RCCI, 2012, Vol. 5.
13. **Group, International Data. 2014.** IDC Analyze the future. [En línea] 2014. [Citado el: 19 de diciembre de 2013.] http://www.idc.com/about/about.jsp.
14. **Hellkamp, Marcel. 2014.** Bottle. *Bottle.* [En línea] 04 de febrero de 2014. [Citado el: 16 de abril de 2014.] http://bottlepy.org/.

15. **Iszaevich, Gunnar Eyal Wolf. 2011.** 1, La Habana : Ediciones Futuro, 2011, Vol. Vol.5. ISSN: 1994-1536.

16. **IT, Solid. 2014.** DB-Engine. [En línea] Solid IT, 2014. [Citado el: 13 de enero de 2014.] http://db-engines.com/en/ranking.

17. **ITI. 2010.** ITI. [En línea] Instituto Tecnlógico de Informática, 30 de diciembre de 2010. [Citado el: 14 de abril de 2014.] http://www.iti.es/servicios/servicio/resource/7240/index.html.

18. **James Rumbaugh, Ivar Jacobson, Grady Booch. 1998.** *El Lenguaje Unificado de Modelado. Manual de Refeerncias.* s.l. : Series Editors, 1998.

19. **José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2010.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : DSIC -Universidad Politécnica de Valencia, 2010.

20. **Ken Schwaber, Jeff Sutherland. 2011.** La Guía Definitiva de Scrum:Las Reglas del Juego. *La Guía de Scrum.* s.l. : Scrum. org, 2011.

21. **Lerman, Craig. 1999.** *UML y Patrones.* Mexico : s.n., 1999. 970-17-0261-1.

22. **Letelier, Patricio. 2007.** *Pruebas de Aceptación como conductor del Proceso Software.* España : Universidad Politécnica de Valencia, 2007.

23. **Luis Miguel Echeverry Tobón, Luz Elena Delgado Carmona. 2007.** *Caso práctico de la metodología ágil XP al desarrollo de software.* Colombia : s.n., 2007.

24. **Manager, Scrum. 2013.** Scrum Manager. [En línea] 23 de enero de 2013. [Citado el: 11 de 04 de 2014.] http://www.scrummanager.net/bok/index.php?title=Pruebas_unitarias.

25. **Marco, Bartolomé Sintes. 2013.** Páginas web con PHP. [En línea] 17 de septiembre de 2013. [Citado el: 16 de enero de 2014.] http://www.mclibre.org.

26. **Mariño, Carlos Vázquez. 2008.** *Programación en PHP 5. Nivel Básico.* 2008.

27. **Martínez, Rafael. 2002.** *Manual de PHP.* 2002.

28. **Martinez, Rafael. 2011.** postgreSQL.org.es. [En línea] 18 de febrero de 2011. [Citado el: 04 de diciembre de 2013.] http://www.postgresql.org.es/. 582.

29. **Mitaritonna, Ing. Alejandro Daniel. 2010.** *CAPA-3: Una innovadora metodología para el desarrollo de software en ambientes de trabajo virtuales .* Buenos Aires : s.n., 2010.

30. **Molina, Maria Pinto. 2011.** e-COMS. [En línea] 13 de abril de 2011. [Citado el: 19 de diciembre de 2013.] http://www.mariapinto.es/e-coms/bases_datos.htm.

31. **Mora, Ing. Jhon Freddy Montes. 2008.** *Puebas del software "Caja Blanca y Caja Negra".* Colombia : Universidad Nacional Abierta y a Distancia UNAD, 2008.

32. **Rodríguez, Dr. Eduardo. 2011.** *Importancia de las pruebas de software.* Mexico : CINVESTAV, 2011.

33. **Rojas, Pablo A. Muñoz. 2012.** [En línea] 2012. [Citado el: 23 de noviembre de 2013.] http://www.nagios-cl.org/nagios-xi..

34. **Rossum, Guido van. 2000.** *Guía de aprendizaje de Python.* Amsterdam : BeOpen PythonLabs, 2000. 1895.22/1012.

35. **Sánchez, Ing. Javier. 2012.** Ecuador : s.n., 2012.

36. **Sepúlveda, Ing.William Díaz. 2013.** Bases de Datos No SQL. [En línea] 27 de mayo de 2013. [Citado el: 21 de noviembre de 2013.] http://basesdedatosnosql.blogspot.com/2013/05/bases-de-datos-nosql-llegaron-para.html.

37. *SXP.* **Peñalver Romero, Ing. Gladys Marsi, García De La Puente, Ing. Sergio Jesus y Ing. Abel Meneses Abad, Ing. Abel. 2010.** Chile : s.n., 2010.

38. **UNICEF, GICHD. 2005.** *IMAS de Educación en el Riesgo de las Minas, Guía de Mejores Prácticas 7.* Ginebra : s.n., 2005. ISBN-13: 978-92-806-3978-0.

39. **Yudisney Vazquez Ortíz, Anthony Rafael Sotolongo León. 2013.** Num. 9, La Habana : RCCI, 2013, Vol. Vol. 6. ISSN: 2306-2495.

40. **Zapata, Luis Giraldo. 2005.** 2005.

41. **Oré, Ing. Alexander. 2009.** UNIT testing – pruebas unitarias –cap. [En línea] 2013. [Citado el: 13 de abril de 2014.] http://www.calidadysoftware.com/testing/pruebas_unitarias2.php

42. **Hamon, Yann. 2014**. Munin. [En línea] 2014 [Citado: 22 de noviembre de 2013.] http://munin-monitoring.org/

43. **Universidad Veracruzana. 2012.** Programación extrema: “Metodología para desarrollo ágil de aplicaciones”. Universo: el periódico de los universitarios. 2012. No. 486.

44. **Guido van Rossum, Barry Warsaw. 2007.** [Online] agosto 10, 2007. [Cited: junio 19, 2014.] http://mundogeek.net/traducciones/guia-estilo-python.htm.

45. **Pressman, Roger S.2005.** Ingeniería del Software. Un enfoque práctico.Quinta Edición. 2005.

46. **Pérez, Maite Rodríguez Corbea Meylin Ordóñez. 2007.** *La metodología XP aplicable al desarrollo del software educativo en Cuba.* s.l. : Universidad de las Ciencias Informáticas, 2007.

Glosario de Términos

No SQL: En informática, es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales. Los datos almacenados no requieren estructuras fijas como tablas.

Información: Desde el punto de vista de la ciencia de la computación es un conocimiento explícito extraído por seres vivos o sistemas expertos como resultado de interacción con el entorno o percepciones sensibles del mismo entorno. En principio la información, a diferencia de los datos o las percepciones sensibles, tienen estructura útil que modificará las sucesivas interacciones del ente que posee dicha información con su entorno.

Dato: Es una representación simbólica (numérica, alfabética, algorítmica) de un atributo o variable cuantitativa. Los datos describen hechos empíricos, sucesos y entidades. Los datos aisladamente pueden no contener información humanamente relevante. Sólo cuando un conjunto de datos se examina conjuntamente a la luz de un enfoque, hipótesis o teoría se puede apreciar la información contenida en dichos datos.

SGBD: Es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.

Servidor: Una computadora en la que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicaciones llamadas clientes, tanto si se trata de un ordenador central (*mainframe*), un miniordenador, una computadora personal, una PDA o un sistema embebido; sin embargo, hay computadoras destinadas únicamente a proveer los servicios de estos programas: estos son los servidores por antonomasia.

Servidor de base de datos: Provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.

Administrador de base de datos: Profesional que administra las tecnologías de la información y la comunicación, siendo responsable de los aspectos técnicos, tecnológicos, científicos, inteligencia de negocios y legales de bases de datos

Anexos

Entrevista

(Realizada al jefe de departamento, especialistas).

Objetivo: comprobar cómo se desarrolla la utilización y monitorización de las bases de datos NoSQL orientadas a documentos en el departamento de PostgreSQL.

Compañero o compañera.

Se necesita su valiosa colaboración para la realización de esta investigación, de sus respuestas dependen los resultados de la misma. **MUCHAS GRACIAS**.

Datos generales.

- ✓ Nombre y Apellidos_____
- ✓ Años de experiencia_____

Aspecto a encuestar

1. ¿Qué bases de datos NoSQL orientadas a documentos se utilizan en el departamento?

2. ¿La información que se almacena y genera respectivamente desde esas bases de datos es vital para el departamento?

3. ¿Qué cantidad de personas del departamento tienen acceso a la información que se maneja?

4. ¿Qué recurso de los que utiliza el servidor de base de datos son más críticos?

5. ¿Qué funciones de las bases de datos son más críticas?

6. ¿Conoce el administrador de bases de datos el estado o comportamiento de los recursos tanto del servidor como de la bases de datos específicamente? Si la respuesta es afirmativa ¿A través de que herramienta obtiene esa información?

7. ¿Qué beneficios encuentra en la monitorización de las bases de datos NoSQL orientadas a documentos a través de una herramienta automatizada?

8. ¿Considera que la utilización de un sistema para monitorización de las bases de datos NoSQL orientadas a documentos beneficiaría el trabajo con ellas en el departamento?