Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Sistema de Monitoreo y Generación de Notificaciones sobre Servidores GNU/Linux

Aplicación web para generación de notificaciones y estadísticas para servidores GNU/Linux

Modalidad: Proyecto Tecnológico

Trimestre 2016 Invierno

Viridiana Matías Hernández
Matrícula 209303318
al209303318@alumnos.azc.uam.mx

Asesor: M en C. Hugo Pablo Leyva
Profesor titular "A" de tiempo completo
Departamento de Sistemas
hpl@correo.azc.uam.mx

Coasesora: Dra. Rafaela Blanca Silva López
Profesor titular "B" de tiempo completo
Departamento de Sistemas
rbsl@correo.azc.uam.mx

Fecha de entrega: 08 de abril de 2016

Declaratoria:

Yo, Hugo Pablo Leyva y Rafaela Blanca Silva López, declaramos que aprobamos el contenido del presente Reporte de Proyecto de Integración y damos nuestra autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

M en C. Hugo Pablo Leyva
Asesor

Dra. Rafaela Blanca Silva López
Coasesora

Yo, Viridiana Matías Hernández, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Viridiana Matías

Resumen

El presente documento contiene información sobre el desarrollo e implementación del proyecto que lleva por nombre “Sistema de Monitoreo y Generación de Notificaciones sobre Servidores GNU/Linux” del cual se desprende el subsistema llamado “Aplicación web para generación de notificaciones y estadísticas para servidores GNU/Linux” el cuál se detallará en las siguientes secciones. Dicho subsistema tiene como objetivo principal facilitar la actividad de monitoreo al usuario mediante el uso una aplicación web capaz de realizar el monitoreo hacia algunos servicios de cada servidor que se encuentren en una misma red local.

Contenido

Índice de imágenes

Introducción

La actividad de monitoreo en el ámbito profesional es una tarea esencial, la cual se encarga de vigilar el funcionamiento ya sea de servicios, aplicaciones e inclusive los propios servidores que son parte de la infraestructura.

La información obtenida de los eventos producidos en el monitoreo permite a los administradores de sistemas mantenerlos informados de qué está sucediendo en los servidores, aplicaciones y servicios que estos alojan, también puede ser de gran ayuda si lo que se desea es encontrar la causa de un problema que se esté presentando.

En la actualidad existen diversas herramientas tanto de código abierto como comerciales que ofrecen servicio de monitoreo, desde servicios básicos hasta la infraestructura completa, entre ellas se encuentranNagios [1], Logstash [2] y Kibana [3] por mencionar algunas, que si bien son de gran utilidad resultan ser complejas a la hora de configurar las acciones que se deberán realizar en caso de presentarse algún fallo lo que limita que la herramienta sea aprovechada en su máxima capacidad.

La solución propuesta en este proyecto ofrece el monitoreo en tiempo real de los servicios alojados en servidores de una red local mediante una aplicación web que

brinda una configuración sencilla para la implementación del monitoreo, dicha aplicación también se compone de un módulo de envío de notificaciones vía correo electrónico en el caso de detectar fallos. Con el fin de ofrecer un mejor servicio se integra un tercer módulo que brinda la opción de consultar actividades de monitoreo sobre períodos de tiempo especificados por el usuario.

Antecedentes

Proyectos terminales:

1. Sistema de Filtrado para redes de computadoras con herramientas Gráficas en Linux

   Este proyecto realiza la configuración de una red local (asignación, traducción y filtrado de IP's) mediante el uso de una herramienta gráfica que se enfoca a usuarios no especializados en el área de administración de sistemas de tal manera que toda la configuración mediante protocolos de enrutamiento o técnicas de traducción de direcciones IP's es por medio de una interfaz gráfica sin necesidad de usar comandos.

   La relación entre este sistema y el propuesto radica en el propósito de facilitar la configuración en este caso sobre redes locales sin necesidad de tener extensos conocimientos técnicos sobre el tema presentando así una interfaz gráfica que facilite dichas tareas. Por el contrario la diferencia principal es el servicio que ofrece cada sistema, en el que se menciona su objetivo es la asignación de direcciones IP mientras que el sistema propuesto ofrece un monitoreo sobre los servidores que se encuentran en una red local.

2. Sistema generador de reportes del puerto Ethernet

   Este proyecto genera reportes de los datos que han sido recabados del puerto Ethernet y almacenadas en bases de datos relacionales utilizando el lenguaje java para la generación de los reportes mediante el uso de Crystalreports[1]. En el sistema propuesto la realización de reportes será con tecnologías web y se consumirá de base de datos no relacionales siendo éstas últimas las principales diferencias entre proyectos, además de que no sólo se monitoreará un puerto sino un conjunto de servicios, por otro lado similitudes se tiene la generación de reportes basados en la información recabada.

[1]Aplicación de inteligencia empresarial utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos (bases de datos).

3. Control y Monitoreo de una Red de Microcontroladores por PC

   El proyecto se centra en el monitoreo de microcontroladores mediante acceso ssh, implementado en visual basic. Aunque este proyecto no realiza monitoreo sobre servidores de aplicaciones, se hace referencia por la semejanza del objetivo final que es el monitoreo en una red local. Además de la creación de reportes sobre los daros recabados de dicho monitoreo. Por otro lado principal diferencia que presentan ambos proyectos es la infraestructura de la red, en el proyecto citado la red que se maneja es de microcontroladores mientras que la solución propuesta es sobre servidores de servicios.

Tésis:

4. Monitoreo del comportamiento de servidores de aplicaciones

   Este proyecto tiene el mismo enfoque que el que sistema propuesto, monitorear servicios sobre servidores en una red local, las diferencias radican en la implementación de cada proyecto. En el proyecto actual la extracción de información se hace mediante la implementación de maestro-esclavos en los servidores para establecer comunicación mientras que el sistema propuesto realiza dicha comunicación mediante web sockets, agregando una funcionalidad extra que son los reportes generados a partir de la información de actividad recabada de los servidores.

5. Monitor de calidad de servicio de servidores en el ámbito educativo

   El proyecto se centra en el monitoreo a servidores de aplicaciones educativas dentro de una red local, mientras que el sistema propuesto se enfoca a monitorear cualquier tipo de servicio y no sólo aplicaciones educativas. Las diferencias principales que se presentan en ambos proyectos radican en sus propósitos, para el monitor de calidad como su nombre lo dice se centra en realizar reportes orientados a la calidad del servicio que prestan las aplicaciones educativas, mientras que el sistema propuesto generará reportes con fines estadísticos sobre los eventos generados por los servicios en un periodo de tiempo.

Sotware:

6. Logstash

   Es una herramienta que permite la extracción de datos sobre eventos de servicios o aplicaciones en específico. Su uso requiere de un mayor grado de conocimientos sobre sistemas operativos y que se maneje en su totalidad la

herramienta para que se exploten sus beneficios, la configuración suele ser confusa y no tiene interfaz gráfica para dicha acción, mientras que en el proyecto propuesto las configuraciones necesarias sean a través de portales web que permitirá un mejor manejo de las utilidades del sistema.

Justificación

La implementación de la aplicación web propuesta facilitará la tarea de un administrador de sistemas en el monitoreo de servicios en la red, siendo de gran utilidad para aquellas personas que tienen conocimientos básicos sobre gestión de servidores y cómo herramienta de apoyo para aquellos que tienen dominio sobre el tema.

La facilidad al realizar las configuraciones de implementación del monitoreo de dicha herramienta y el uso de tecnologías web permitirán que su ejecución sea multiplataforma en cualquier tipo de navegador sobre cualquier sistema operativo.

La arquitectura de tipo API[2] REST[3] que trabaja con el protocolo HTTP que va muy de la mano con programación web lo que resulta una ventaja sobre las herramientas que se encuentran disponibles actualmente, ya que no necesita de un instalador para su funcionamiento.

Para un administrador de sistemas es importante garantizar el buen funcionamiento de la infraestructura de un sistema, detectar de manera pronta los fallos que se generen conlleva tomar acciones inmediatas para solucionarlos, y tener indicios que lo conduzcan a la causa evita perder tiempo en encontrar el origen del error pues acota el espacio de búsqueda en la detección del problema.

Las funcionalidades que se agregan al sistema propuesto ayudan a tener un histórico sobre los eventos que han ido ocurriendo en cierto período de tiempo, de tal manera que se puedan hacer consultas sobre los servicios monitoreados con la opción de generar reportes que también pueden ser bajados en formato PDF.

Objetivos

Objetivo general

- Desarrollar una aplicación web que monitoree servicios en servidores GNU/Linux en tiempo real sobre una red local generando notificaciones en caso de presentarse fallas.

2 Interfaz de Programación de Aplicaciones (ApplicationProgramming Interface)
3Transferencia de Estado Representacional (RepresentationalState Transfer)

Objetivos específicos

- Desarrollar módulo de monitoreo sobre el estado de los servicios en tiempo real.
- Implementar API para el consumo y clasificación de la información obtenida sobre eventos obtenidos de los servicios en los servidores monitoreados.
- Implementar módulo para generación de configuraciones en el monitoreo de servicios.
- Implementar módulo de notificaciones vía correo electrónico de fallos ocurridos en servicios monitoreados.

Marco teórico

La Aplicación web para generación de notificaciones y estadísticas para servidores GNU/Linux fue desarrollado bajo la metodología de desarrollo ágil SCRUMutilizando tecnologías enfocadas al desarrollo web lo que hace del sistema más escalable en poco tiempo.

Metodología SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

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

Se determinó elegir esta tecnología debido al tiempo que se requería para el desarrollo del proyecto y la cantidad de requerimientos a cubrir.

Debian 8

El sistema operativo que aloja el sistema es Debian en su versión 8. Se eligió esta distribución por la sencilla configuración en algunos servicios (ssh, apache) que se monitorean con la aplicación web desarrollada.

MongoDB 2.4.10

Es un gestor de base de datos no relacional que muestra flexibilidad en cuanto a su modelado dejando libre la opción de modificaciones en cualquier momento. MongoDB fue elegido por la compatibilidad que tiene con el framework utilizado para el desarrollo del sistema

Node.js 4.6.8
Node es un intérprete Javascript del lado del servidor que cambia la noción de cómo debería trabajar un servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sólo una máquina física.

Para la gestión de información recabada del monitoreo se utilizó nodejs, ya que junto con sailsjs, facilita la integración de vistas con los servicios web desarrollados.

SailsJs 0.11
Es un framework para desarrollo web con nodejs basado en el modelo vista controlador, brinda la facilidad necesaria para crear sistemas en poco tiempo y brindando escalabilidad.
El framework conforma la parte medular del sistema siendo este el gestor entre los servicios web que se desarrollan, base de datos, sockets, vistas HTML y la interacción entre todos estos así como los módulos externos.

Angular js 1.2.13
Framework orientado al desarrollo de aplicaciones web, su función principal es mejorar el HTML con el fin de que sea entendible y fácil de manipular para realizar cambios de manera pronta sin necesidad de manejar demasiado código.
Se hizo uso de este framework para la interacción del usuario con los servicios de tal manera que la aplicación sea dinámica y responda a las peticiones hechas por el usuario a través de las vistas.

Python 2.7
Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Para la creación de los sockets se utilizó este lenguaje por su versatilidad y amplia compatibilidad con otros lenguajes compilados y gestores de base de datos como mongoDB.

Desarrollo

Módulos del sistema

La aplicación web desarrollada se compone de tres módulos que integran la funcionalidad completa de este proyecto.

A continuación se muestran los diagramas de arquitectura tecnológica para el desarrollo de cada uno de los módulos que componen el sistema.

Es el módulo encargado de realizar el monitoreo de los servicios alojados en los distintos servidores, la comunicación con la API de monitoreo no es directa, a través de sockets implementados en python alojan los datos que recaban en la base del dato del servidor de aplicación web para que éste consuma los datos y posteriormente a través de websockets los muestre en la vista.

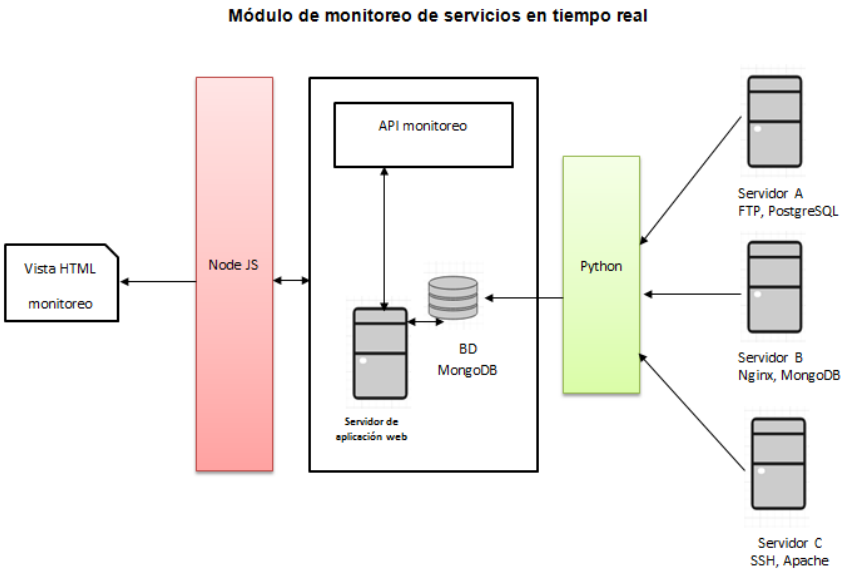


Imagen 1 Diagrama de arquitectura tecnológica: Módulo de monitoreo en tiempo real

En la siguiente imagen se muestra la interacción entre servidores cliente y servidor maestro, la comunicación resulta muy similar a la de la imagen 1, sin embargo en este módulo el flujo es bilateral.

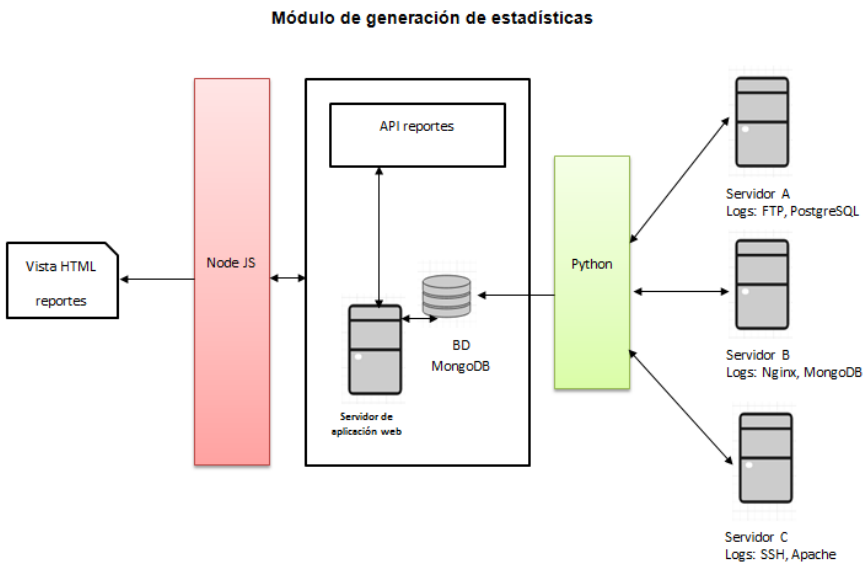


Imagen 2 Diagrama de arquitectura tecnológica: Módulo de generación de estadísticas

La siguiente imagen muestra los componentes del módulo de notificaciones del cual se desprenden dos APIS: notificaciones y alertas. Dependiendo del tipo de error se procede a enviar una alerta a la página web o mandar correo electrónico con el motivo del fallo.

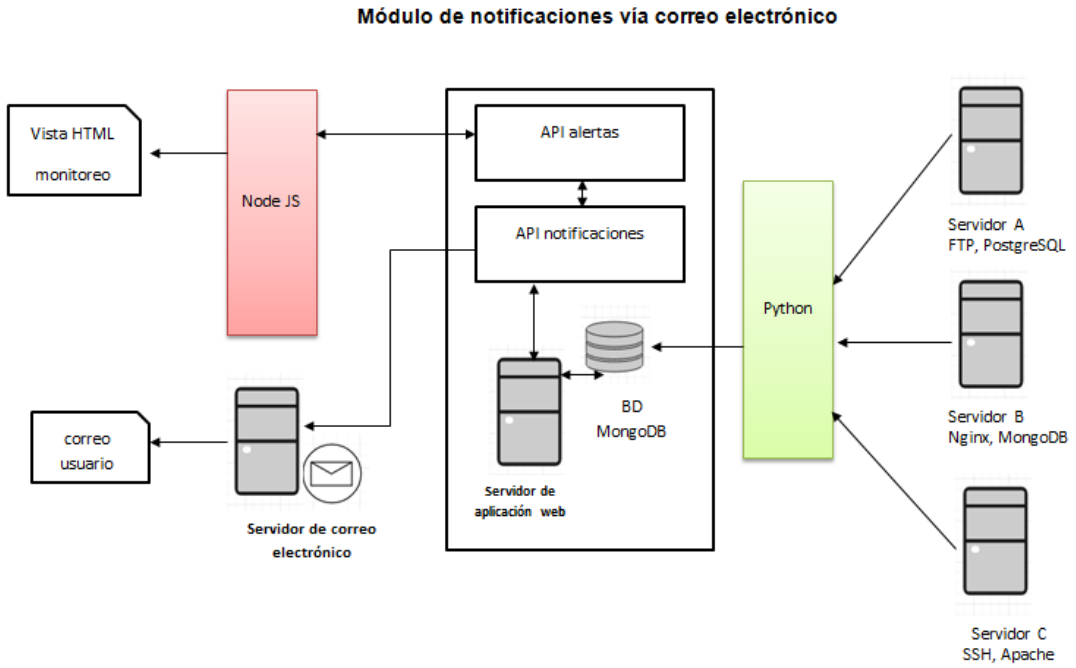


Imagen 3 Diagrama de arquitectura tecnológica: Notificaciones

Módulo de monitoreo de servicios en tiempo real

Este módulo es el encargado de monitorear en tiempo real el estado de los servicios que se encuentran en cada uno de los servidores de la red local informando de los eventos ocurridos al servidor web.
Para la realización de este módulo se implementó la comunicación con sockets TCP/IP en python siguiendo la arquitectura cliente-servidor.
Para este módulo se explicarán dos partes fundamentales que lo componen:

- Servidor
  Es el encargado de levantar el puerto de comunicación y atender las peticiones de todos y cada uno de los clientes que pide conectarse con él.
  En la siguiente imagen se puede observar la clase que se declara con los datos que serán necesarios para establecer la conexión; IP del servidor (192.168.0.50) y puerto (5000).

```
class ClientThread(threading.Thread):

    def __init__(self,ip,port,clientsocket):
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.csocket = clientsocket
        print "Se ha conectado  "+ip+":"+str(port)

    def run(self):
        print "conexión de : "+ip+":"+str(port)

        clientsock.send("conexión exitosa")

        data = "clientsocket"

        while len(data):
            data = self.csocket.recv(2048)
            print "Recibiendo datos de (%s:%s) servicio : %s"%(self.ip, str(self.port), data)
            self.csocket.send("Enviaste : "+data)

        print "Cliente "+self.ip+" desconectado"

host = "192.168.0.50"
port = 5000
```

Imagen 4 Parámetros servidor

Que entra en un ciclo para estar en escucha y recibir las peticiones que los clientes puedan hacer:

```
while len(data):
    data = self.csocket.recv(2048)
    print "Recibiendo datos de (%s:%s) servicio : %s"%(self.ip, str(self.port), data)
    self.csocket.send("Enviaste : "+data)
```

Imagen 5 Servidor en escucha de cliente

Ejecución:
En la siguientes imágenes se muestra la terminal del proceso del servidor corriendo, es espera de alguna petición y sucesivamente recibiendo datos del cliente que se conecta.

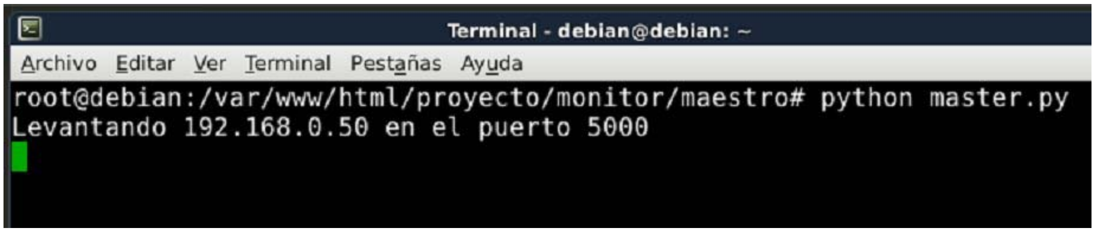


Imagen 6 Servidor corriendo

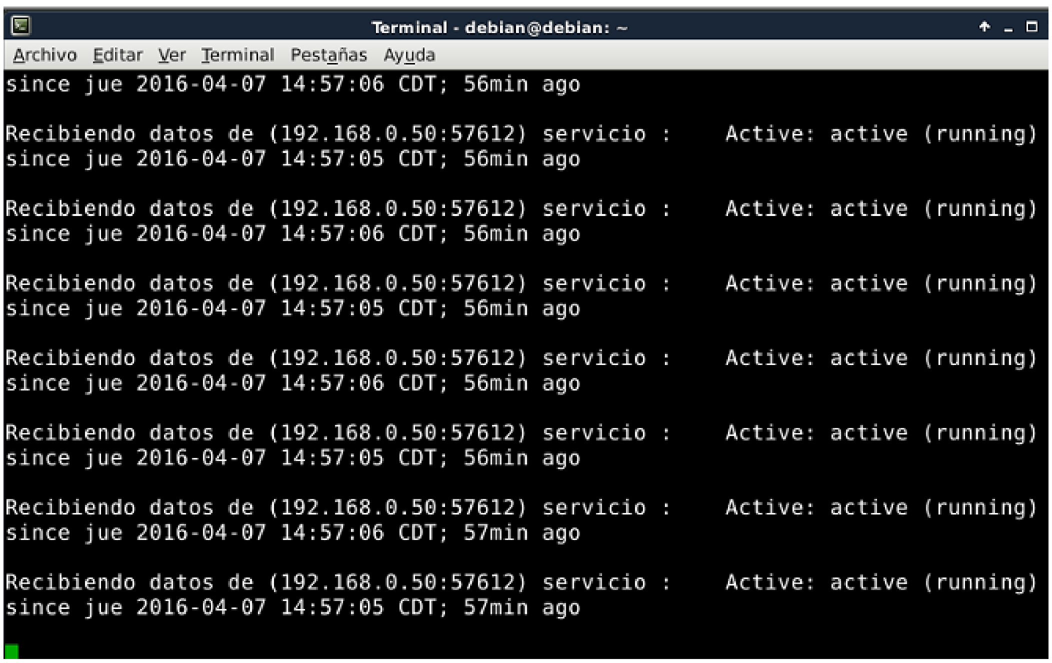


Imagen 7 Servidor recibiendo datos de cliente

- Cliente
  Habrá un cliente por cada servidor a monitorear. Cada cliente será el encargado de recabar los datos de los servicios del servidor que esté monitoreando, se comunicará con el maestro a través del puerto en escucha que se haya establecido, que en este caso es el puerto 5000 y almacenará en la base de datos de la aplicación web los datos recabados de su ejecución.

  En la siguiente imagen las partes esenciales del código del cliente muestran los datos del servidor al que se conectarán y su respectivo puerto, así como las colecciones en las que deben guardar los datos recabados del servidor en el que se alojan.

```
#Direccón IP de la BD
mongoClient = pymongo.MongoClient("192.168.0.50",27017)
#Conexión a la base de datos
db = mongoClient.eventos

#Colecciones que se manejarán
collection = db.dato
procesos = db.recursos
alertas = db.alertas


# Creando un socket TCP/IP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conecta el socket en el puerto cuando el servidor esté escuchando
#192.168.0.50
server_address = ('192.168.0.50', 5000)
print >>sys.stderr, 'Conectando a %s puerto %s' % server_address
sock.connect(server_address)
```

Imagen 8 Conexión de cliente a servidor

El cliente genera los datos de cada servicio y los aloja en la base de datos donde se aloja el servidor, el código que se muestra en la imagen siguiente contiene un fragmento de los datos que se guardan en la colección dato, dicha colección contiene la información de los servicios que cada cliente obtiene.

```
#Guardar servicios en BD
ssh1 = subprocess.Popen("service ssh status | grep -e 'Active'", shell=True,
essh = ssh1.stdout.readlines()
ssh = essh[0]
envssh = {"ip":ip,"servicio":"ssh","estado":ssh,"fecha":fecha,"hora":hora}
collection.insert(envssh)
```

Imagen 9 Obtención de datos y envío

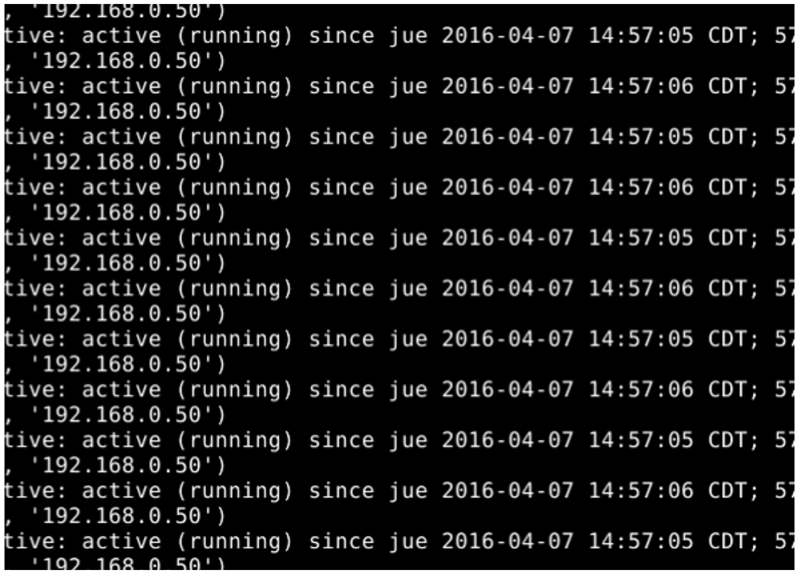
Si el envío es exitoso el cliente se seguirá ejecutando indeterminadamente, en caso de ocurrir un fallo se detendrá.

```
try:
    envia()
except sock.error:
    print('connection error')
    sys.exit(0)
```

Imagen 10 Enviar datos de manera indeterminada

Ejecución:

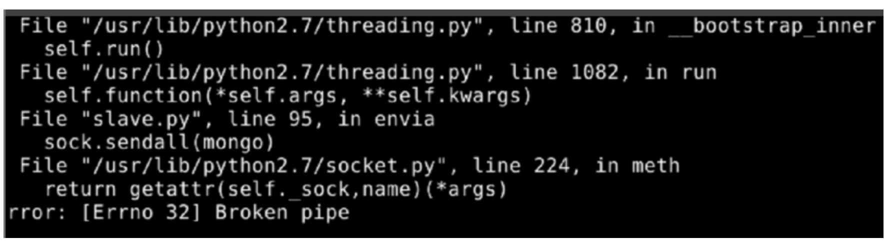
Esta comunicación entre cliente-servidor quedará fija a menos que surja algún fallo de alguna de las dos partes.



```
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:06 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:06 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:06 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:06 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:06 CDT; 57
, '192.168.0.50')
tive: active (running) since jue 2016-04-07 14:57:05 CDT; 57
, '192.168.0.50')
```

Imagen 11 Cliente corriendo

Si el proceso de uno de los clientes falla, el maestro cierra la conexión del cliente y sigue funcionando, pero si el maestro falla, se cierran todas las conexiones que se tienen con los clientes y en ese momento el módulo de monitoreo dejará de funcionar.



```
 File "/usr/lib/python2.7/threading.py", line 810, in __bootstrap_inner
   self.run()
 File "/usr/lib/python2.7/threading.py", line 1082, in run
   self.function(*self.args, **self.kwargs)
 File "slave.py", line 95, in envia
   sock.sendall(mongo)
 File "/usr/lib/python2.7/socket.py", line 224, in meth
   return getattr(self._sock,name)(*args)
rror: [Errno 32] Broken pipe
```

Imagen 12 Error de cliente, cerrando conexión

Módulo de generación de estadísticas

Este módulo extrae los datos que son almacenados en la base de datos por el "Sistema de monitoreo de Servicios en sistemas GNU/Linux" [4] , de la cual obtiene la información sobre un rango de tiempo determinado (fecha actual, la última semana o por una fecha específica) sobre el estado de los servicios de cada uno de los servidores que serán monitoreados, mostrando una gráfica con el número de eventos exitosos contra el número de eventos fallidos según el servicio que se haya elegido.
Para la realización de este módulo se implementaron servicios web realizan las búsquedas por criterios como tipo de servicio, rango de fecha, entro otros, que alimentan las vistas HTML.

Ejecución:
La siguiente imagen muestra la gráfica generada de los eventos exitosos contra los fallidos de una fecha en específico que el usuario proporcionó.

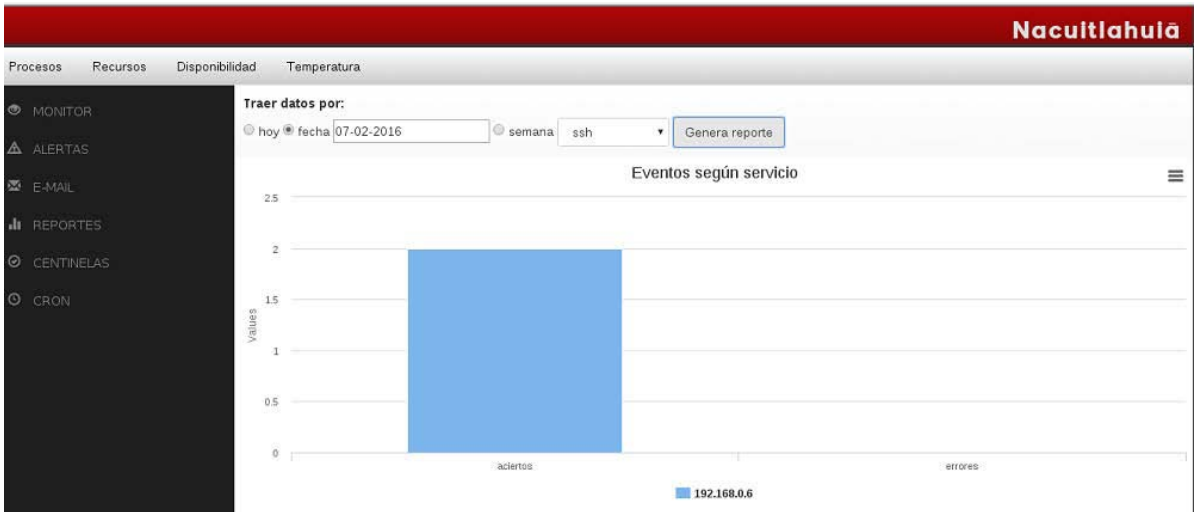


Imagen 13 Eventos contabilizados de ssh por fecha

Si la aplicación web cuenta con acceso a internet el resultado de las tablas puede ser exportado ya sea en formato png, jpg o PDF como se muestra en la siguiente imagen:

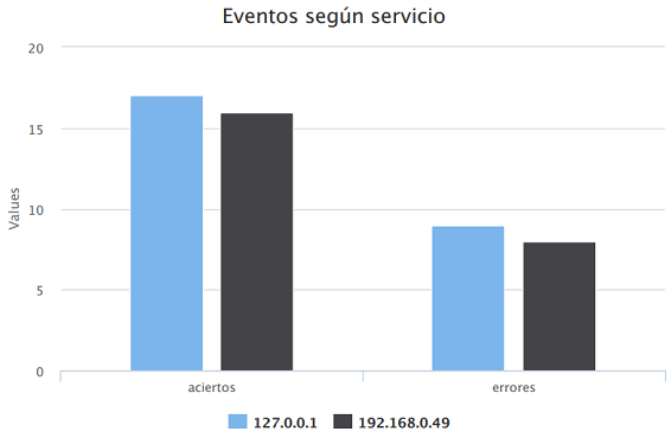


Imagen 14 Exportación de gráficos formato PDF

Módulo de notificaciones vía correo electrónico

Este módulo va de la mano con el módulo de monitoreo de servicios en tiempo real, en caso de presentarse alguna falla en una aplicación que está siendo monitoreada se notifica al personal encargado del área a través de un correo electrónico sobre el fallo sucedido.

El sistema cuenta con una pantalla para dar de alta a los usuarios que se desee notificar como se muestra en la siguiente pantalla:

Imagen 15 Pantalla para agregar cuentas de correo electrónico

En la siguiente imagen se muestra el correo electrónico que se recibe cuando sucede algún fallo catalogado como crítico con los datos correspondientes al mismo como lo son: *IP del servidor*, *servicio en el que ocurrió la falla*, *causa*, *fecha* y *hora:*

Imagen 16 Notificación de fallo

Para los fallos que no son considerados críticos se muestran en la sección de “Alertas” de la aplicación web como ocurre en la siguiente imagen:

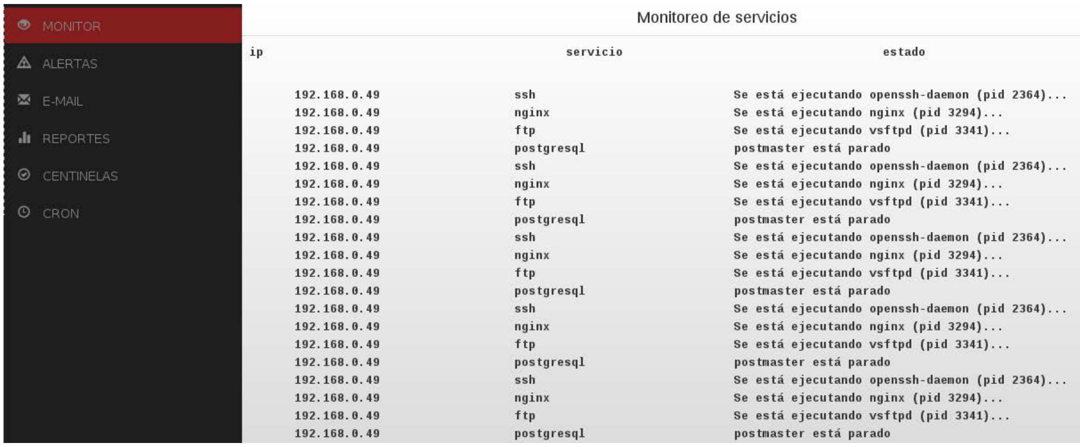


Imagen 17 Vista del monitoreo de servicios

Instalación

Para la instalación del sistema se requiere tener Nodejs como entorno de ejecución en su versión más reciente y estable v4.6.85 y npm 3.8.3 para poder descargar las dependencias que se requieran. Una vez situado en el proyecto maestro se ejecuta el comando ***node app.js*** o ***sailslift.***

Para la ejecución del maestro el comando es el siguiente: ***python master.py***

Para la ejecución de cada cliente: ***python slave.py***

Ejecución:

Si el sistema funciona correctamente se verá en terminal una pantalla como a siguiente:

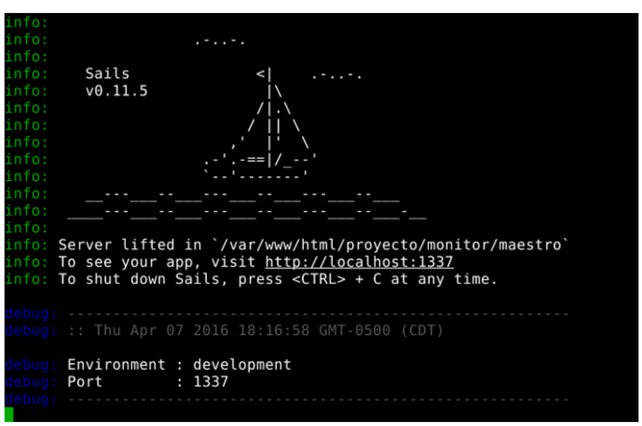


Imagen 18 Ejecutando sails.js

Análisis de resultados

Las pruebas se realizaron sobre distribuciones de Linux; Debian 7, Debian 8 y Centos 6.2, encontrando diferencias notables en la forma de ejecución de comandos para su monitoreo por lo que se decidió realizar una versión de cliente para Debian (7 y 8) y otra para Centos.

Los servicios que se monitorearon fueron nginx, FTP, mongoDB, postgreSQL, apache y ssh. La manera en que se encuentra integrado el sistema permite que sean añadidos más servicios para su monitoreo, se decidieron estos servicios pues sirven de ejemplo para el funcionamiento del sistemas pero queda claro que para realizar un buen trabajo de monitoreo es necesario incluir los servicios clave que se alojen en cada servidor.

Debido a la ejecución asíncrona por la que se caracteriza nodejs, se tuvieron difcultades al mostrar los datos del monitoreo en tiempo real por que se optó por agregar una función que controle el tiempo de en el que deba mostrar los datos del monitoreo.

Por otro lado, de los resultados obtenidos durante el monitoreo se pudo observar que la tasa de eventos por error es relativamente baja, la mayoría de los errores obtenidos fueron generados de manera intencional para probar el funcionamiento de los módulos.

Conclusiones

Durante el desarrollo del presente proyecto se obtuvieron los resultados deseados cumpliendo con los objetivos indicados, incluso se pudieron generar módulos extra para complementar el funcionamiento del sistema como lo fueron el mostrar la temperatura del procesador y los procesos que corren en los servidores. Al mismo tiempo determiné que para que el sistema sea lo suficientemente robusto y confiable en cuanto a los resultados que obtenga, necesita de un exhaustivo y profundo estudio sobre cada uno de los recursos y servicios que aloje, ya que en este proyecto se delimitó el monitoreo a unos cuantos servicios sería necesario explorar mayores opciones y tener en cuenta todas las posibilidades de situaciones que pueden ocurrir en un entorno de producción y con diferentes configuraciones.

En conclusión el desarrollo de esta aplicación brinda la facilidad de uso al usuario para un monitoreo básico de servicios sin la necesidad de que éste tenga que realizar alguna configuración, a su vez permite que se puedan integrar nuevos módulos de funcionalidad sin mayores problemas.

Referencias bibliográficas

[1] Nagios [online] disponible en: https://www.nagios.org/

[2] Logstash [online] disponible en: https://www.elastic.co/products/logstash

[3] Kibana [online] disponible en: https://www.elastic.co/guide/kibana/en/kibana/current/index.html

[4] Israel I. Montes de Oca, “Sistema de monitoreo de Servicios en sistemas GNU/Linux”, [software]

[5] Bermúdez Silva Irving A., “Debian GNU/Linux Para El Usuario Final”, Ed. Grupo ICCO, 2007.

[6] Alegre Ramos Ma. Del Pilar, García-Cervigón Hurtado Alfonso, “SISTEMAS OPERATIVOS EN RED”, Ed. Paraninfo, 1ª edición 2011.

[7] Azaustre Carlos, “Desarrollo web ágil con Angular.js”, 1ª edición formato PDF.

[8] Béjar Heredia María de la Cruz, “Selección, instalación, configuración y administración de los servidores de transferencia de archivos”, IC Editorial,1ª edición 2014.

[9] Gómez López Julio, “Administración de Sistemas Operativos. Un enfoque práctico”, Ra-Ma Editorial, 2ª edición 2011.

[10] Adelstein Tom, Lubanovic Bill, “Administración de sistemas Linux”, Ed. Anaya Multimedia/O’ Reilly, 2007.

[11] Gift Noah, Jones Jeremy, “Python for Unix and Linux System Administration”, Ed. O’ Reilly, 2008.

[12] Cameron Jamie, “LINUX SYSTEMS WITH WEBADMIN.System Administration and Module Development”, Ed. Prentice Hall, 2004.

[13] Métodologías ágiles, [online] disponible en: http://proyectosagiles.org/que-es-scrum/

[14] Node.js, https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

[15] Sails.js, [online] disponible en: http://sailsjs.org/

[16] Angular js, [online] disponible en: https://angularjs.org/

[17] Python, [online] disponible en: https://www.python.org/

Entregables

De acuerdo a la propuesta establecida para el desarrollo de este proyecto, los entregables comprometidos a realizar fueron los siguientes:

- Módulo de generación de estadísticas
- Módulo de monitoreo de servicios en tiempo real
- Módulo para enviar notificaciones vía correo electrónico
- Documentación del sistema*
- Guíarápida de usuario*
- Código fuente. Se incluye dentro de la carpeta ***Anexos*** que contiene a su vez la carpeta *maestro* y *esclavo* las cuales corresponden a los módulos especificados en este proyecto.
- Reporte general

Los puntos marcados con asterisco (*) se incluyen en la siguiente sección.

Documentación del sistema

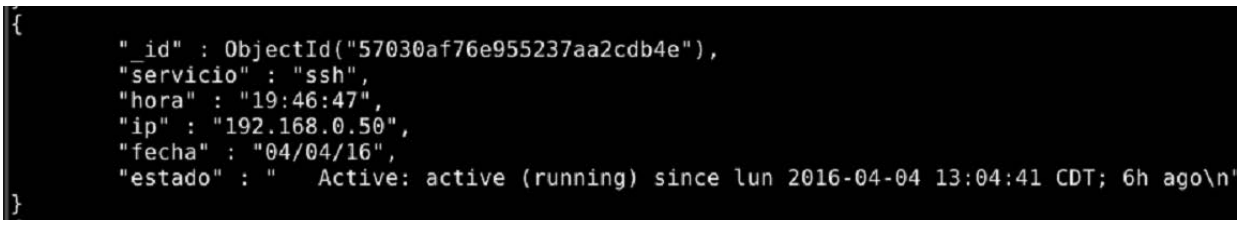
Las colecciones que se utilizaron para el desarrollo del sistema son las que se detallan a continuación:

Colección: **`Dato`**

```
{
    servicio: String,
    hora: Date,
    fecha: Date,
    ip: String,
    estado: String
}
```

Atributo	Tipo de dato	Descripción
servicio	String	Indica que tipo de servicio se está monitoreando: *ssh, ftp, nginx, mongo, apache o sql*
hora	Date	Indica la hora en el que se realizó el registro
fecha	Date	Indica la fecha en la que se realizó el registro
ip	String	Indica la ip del servidor que se está monitoreando
estado	string	Indica el estado del servicio que se monitorea

Ejemplo:



Colección: `Alertas`

```
{
    ip: String,
    proc: String,
    valor: String,
    hora: Date,
    fecha: Date
}
```

Atributo	Tipo de dato	Descripción
proc	String	Indica que tipo de servicio se está monitoreando: *ssh, ftp, nginx, mongo, apache o sql*
hora	Date	Indica la hora en el que se realizó el registro
fecha	Date	Indica la fecha en la que se realizó el registro
ip	String	Indica la ip del servidor que se está monitoreando
valor	string	Indica el valor obtenido del log de error del servicio que se monitorea

Ejemplo:

Colección: `Recursos`

```
{
    ip: String,
    proc: String,
    valor: Object
}
```

Atributo	Tipo de dato	Descripción
proc	String	Indica que tipo de recurso se está monitoreando: *uptime,temperatura,psaux*
ip	String	Indica la ip del servidor que se está monitoreando
valor	Object	Contiene el/los valores recabados de los recursos de cada servidor

Ejemplo:

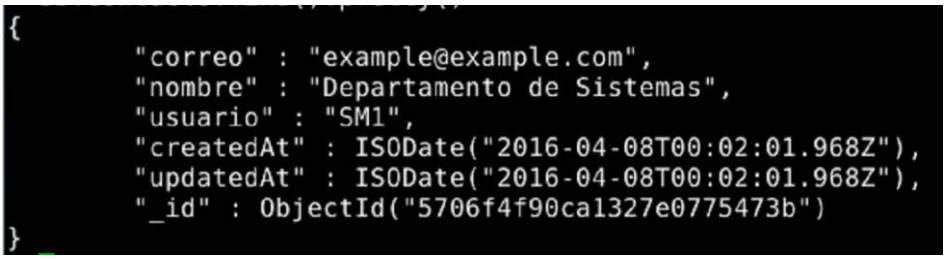

```
{
        "_id" : ObjectId("56eb05da6e955207d2c1e750"),
        "ip" : "192.168.0.50",
        "proc" : "uptime",
        "valor" : "23min,4\n"
}
{
        "_id" : ObjectId("56eb05da6e955207d2c1e751"),
        "ip" : "192.168.0.50",
        "proc" : "temperatura",
        "valor" : "+43.0°C\n"
}
{
        "_id" : ObjectId("56eb05da6e955207d2c1e752"),
        "ip" : "192.168.0.50",
        "proc" : "psaux",
        "valor" : [
                "/opt/google/chrome/chrome\n",
                "node\n",
                "/opt/google/chrome/chrome\n",
                "/opt/google/chrome/chrome\n",
                "grunt\n",
                "/usr/bin/mongod\n",
                "/usr/bin/sublime_text\n",
                "/opt/google/chrome/chrome\n",
                "/usr/bin/X\n",
                "Thunar\n"
        ]
}
```

Colección: `Contacto`

```
{
    nombre: String,
    correo: String,
    usuario: String
}
```

Atributo	Tipo de dato	Descripción
correo	String	Correo electrónico que se proporciona para recibir notificaciones
nombre	String	Nombre que se asocia al correo proporcionado para recibir notificaciones
usuario	String	Valor que agrega por defecto el sistema, se utiliza como identificador de cada usuario.

`Ejemplo:`

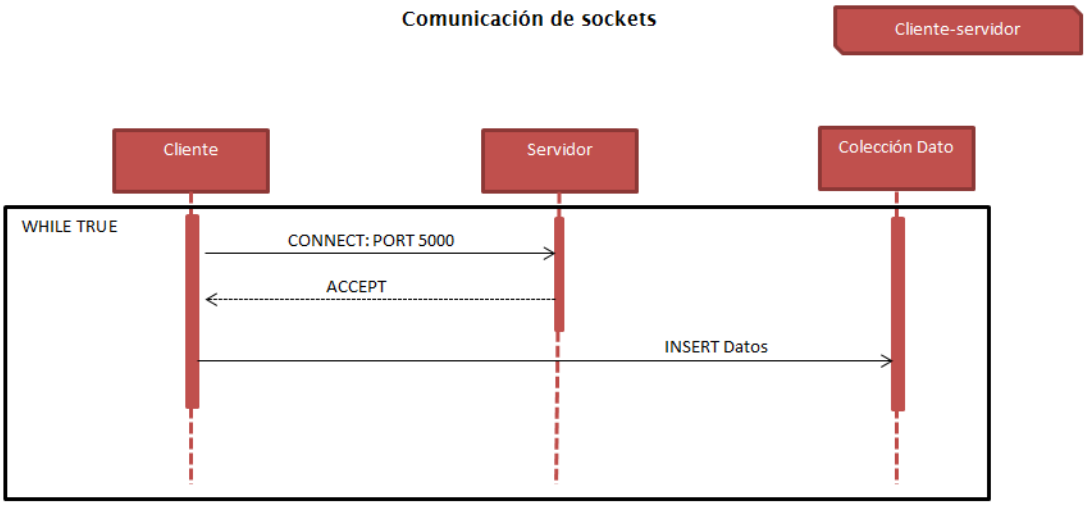


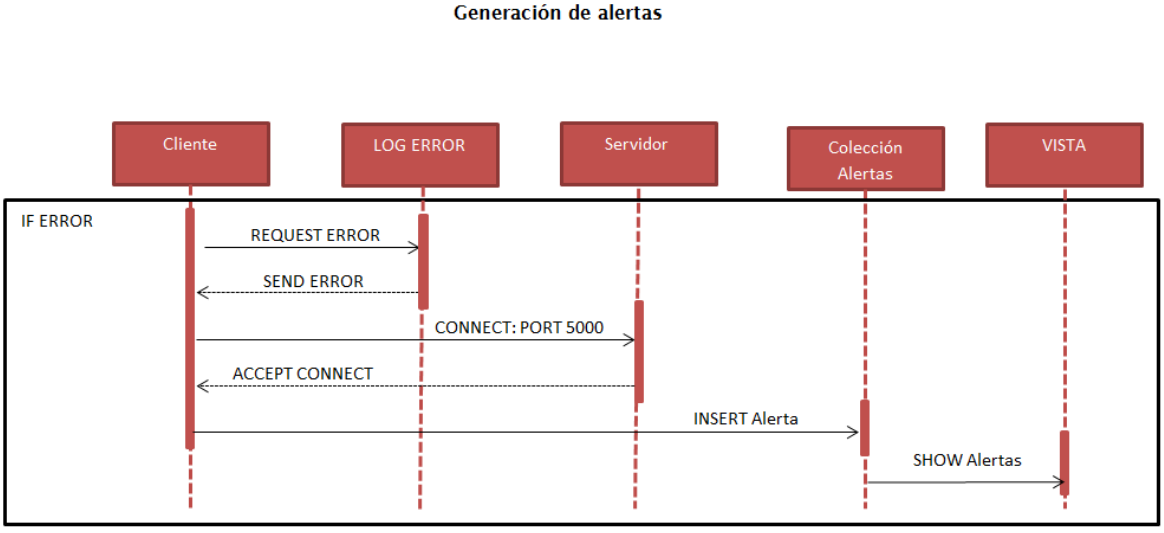
Diccionario de tipo de datos:

El último estándar ECMAScript.

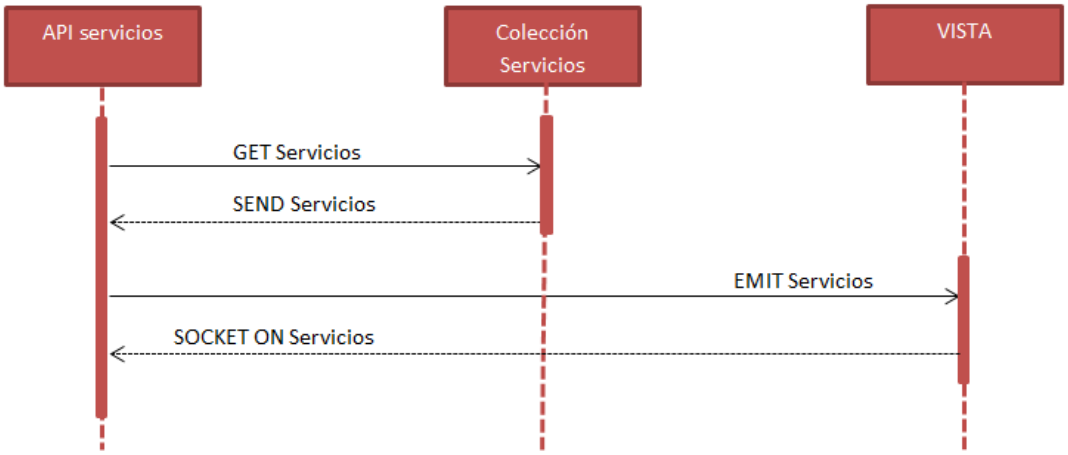
Tipo de dato	descripción	Referencia
String	Secuencia de caracteres usado para representar el texto	https://developer.mozilla.org/es/docs/Glossary/String
Object	Se refiere a una estructura de datos que contiene datos e instrucciones para trabajar con los datos	https://developer.mozilla.org/es/docs/Glossary/Objeto
Date	Cadena representando una fecha	https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/parse

Diagramas de interacción

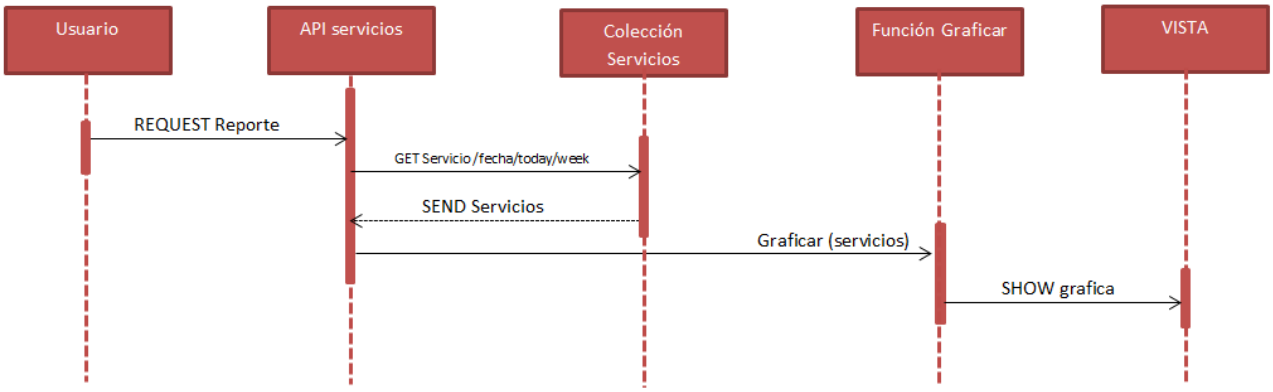




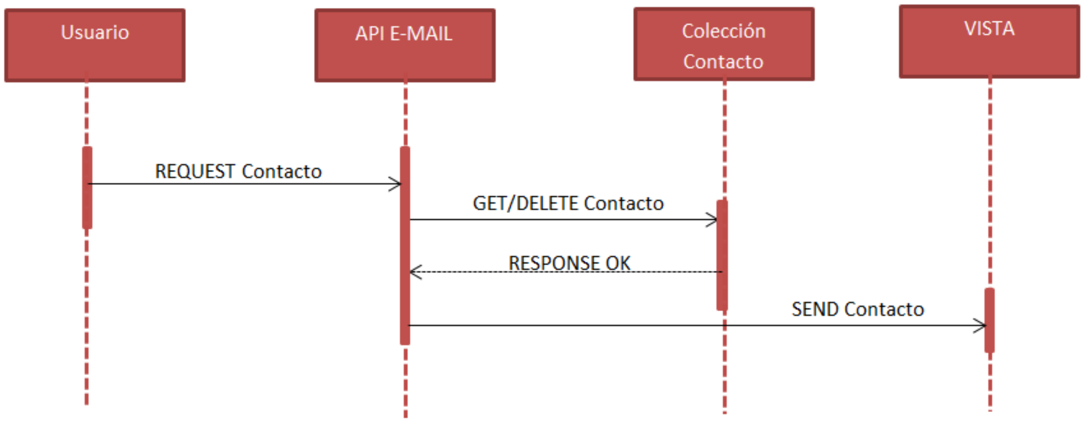
Mostrar monitoreo de servicios



Reporte Servicios



CRUD Contacto



Manual de usuario

El presente documento describe las características de cada sección que compone la ***Aplicación web para generación de notificaciones y estadísticas para servidores GNU/Linux***con el fin de que el usuario comprenda el funcionamiento de éste y conozca el porqué de los valores obtenidos.

Monitor
En esta pantalla se muestran los servicios que son monitoreados en tiempo real con la descripción del proceso que siguen cada uno de ellos:

Pantalla 1 Monitor

Alertas

En esta sección se muestran las alertas que son generadas por los archivos de log de error de cada uno de los servicios.

Pantalla 2 Alertas

Correo electrónico

En la parte de configuración de correo electrónico el usuario puede dar de alta en el sistema las cuentas que desea que reciban notificación en caso de presentarse algún fallo.

El botón de borrar que se encuentra en la parte inferior izquierda vacía la colección de contactos que se encuentren registrados hasta el momento.

Pantalla 3 E-mail

Reportes

El usuario puede visualizar a través de esta sección los resultados obtenidos de los eventos que se han monitorizado según un rango de tiempo, existen tres opciones a elegir “hoy”,”fecha” y “semana”.

Si se elige *hoy*, el sistema buscará todos los registros del servicio que se desea visualizar y contabilizará el total de eventos exitosos y fallidos para generar un gráfico como el que se muestra en la imagen.

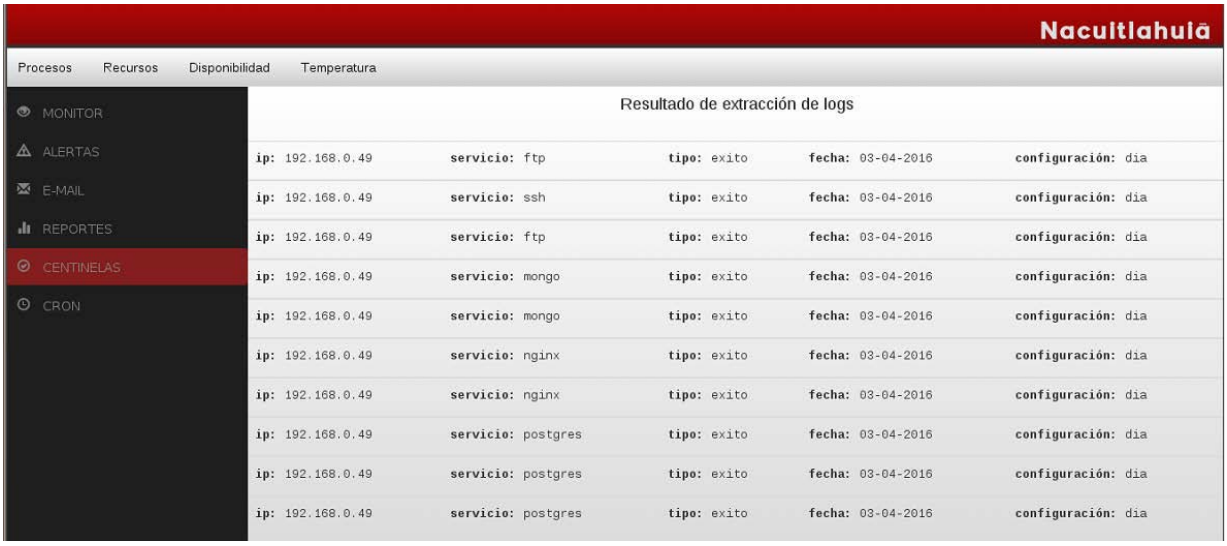
Si la opción elegida fue fecha, se desplegará un cuadro de texto para que el usuario ingrese una fecha de su elección, es importante mencionar que debe ingresar el dato con el siguiente formato “dd-mm-aaaa” de lo contrario la búsqueda no será exitosa, los datos recabados que coincidan con la fecha indicada mostraran el gráfico correspondiente.

Cuando la opción elegida es “semana” se contabilizarán todos los eventos que se hayan generado en los últimos 8 días a partir del día que se haga la consulta.

Pantalla 4 Reportes según fecha en específico

Centinelas

Esta sección contiene los resultados de la extracción de logs de cada uno de los servidores así como de sus servicios, en caso de presentarse algún fallo el sistema lo mostrará en la pantalla como la de la imagen siguiente:



Pantalla 5 Resultado de extracción de logs

Cron

Esta sección muestra la configuración actual de tiempo que se tiene para hacer la extracción de logs, existen dos opciones: *diario y semanal*

Si la configuración elegida es diario indica al sistema que el usuario desea hacer la extracción de logs diariamente.

Por el contrario si la opción elegida es semana, la extracción se hará semanalmente según el día indicado por el usuario.

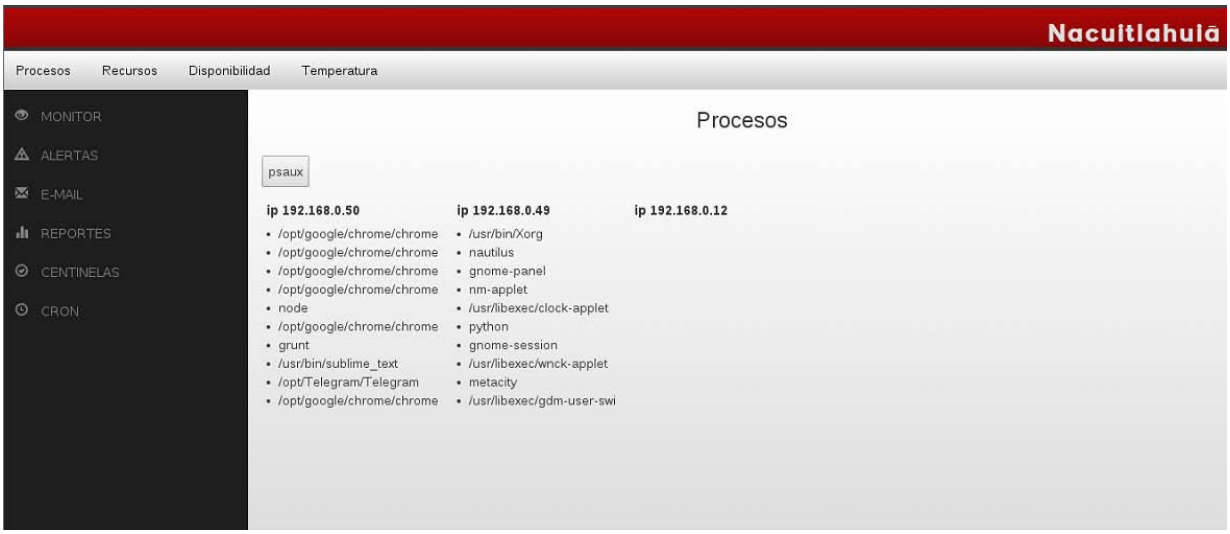
Pantalla 6 Configuración de cron

Las siguiente tres imágenes muestran datos de apoyo que complementan el funcionamiento de la aplicación web, en la sección de disponibilidadmuestra el tiempo que llevan levantados los servidores.

Pantalla 7 Comando uptime

Procesos

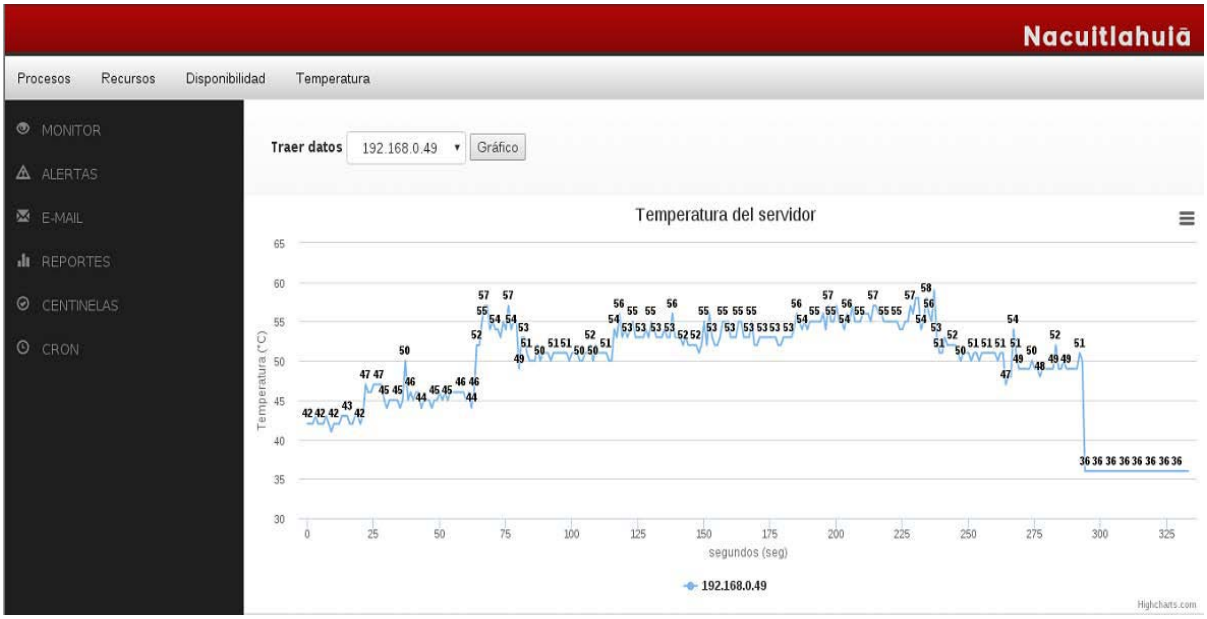
Se muestran los procesos que corren en cada servidor.



Pantalla 8 Comando psaux

Temperatura

En esta sección se muestra la temperatura del procesador recabada en el último minuto



Pantalla 9 Temperatura del procesador