

Título: Sistema para la configuración y monitoreo de alta disponibilidad, balanceo de carga en aplicaciones con HAProxy y Keepalived.

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

Autores:

Hansel Tellez Milián

Angel Danny Aguila Jerez

Tutores:

Ing. Dania Carmenate Cantero

Ing. Claudia Liset Stincer Torres

La Habana, 2016

“Año 57 de la Revolución”



facebook.

"El mayor riesgo es no correr ningún riesgo. En un mundo que cambia muy rápidamente, la única estrategia que garantiza fallar es no correr riesgos."

Mark Zuckerberg.

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Telemática de la Universidad de las Ciencias informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Angel Danny Aguila Jerez

Hansel Tellez Milián

Firma del Autor

Firma del Autor

Ing. Claudia Liset Stincer Torres

Ing. Dania Carmentate Cantero

Firma del Tutor

Firma del Tutor

Datos de Contacto

Datos del Autor:

Hansel Tellez Milián

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: hftellez@estudiantes.uci.cu

Datos del Autor:

Angel Danny Aguila Jerez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: adaguila@estudiantes.uci.cu

Datos del Tutor:

Ing. Dania Carmenate Cantero. Pertenece al área de TLM, Dpto. de Desarrollo de Componentes, Facultad 2, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dcarmenate@uci.cu

Datos del Tutor:

Ing. Claudia Liset Stincer Torres. Pertenece al área de TLM, Dpto. de Desarrollo de Componentes, Facultad 2, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: clstincer@uci.cu

Agradecimientos

Quiero agradecer de una forma u otra a todas aquellas personas que han hecho posible que este sueño se realizara.

A mis padres Sonia y Cesar, a mi hermana Gretel, a mis abuelos y a mi familia en general por que cada uno de ellos me apoyo siempre aun cuando las cosas no anduvieron bien, cada uno de ellos siempre creyó en mí.

A mi compañero de tesis, por ser un excelente compañero y estar ahí cuando fue necesario, por toda su entrega.

A mis tutoras, a Claudia por toda la ayuda brindada y a Dania por estar siempre al pendiente de que todo estuviera en tiempo, en general a las dos por soportarnos durante este tiempo y por su preocupación.

A todos los especialistas del departamento que nos ayudaron muchísimo aún sin ser nuestros tutores.

A mis amistades del aula, por el apoyo incondicional.

A mi novia, por soportarme durante todo este tiempo y aún más durante el periodo de desarrollo de la tesis, por su ayuda incondicional aun cuando tenía otras cosas que atender, por todo el apoyo que me brindo en todo momento, por darme ánimos cuando las cosas se ponían difíciles, por ser la persona que me ayudó a levantarme en los peores momentos.

A toda la familia de mi novia, por tratarme como si fuera uno de ellos durante todo este tiempo.

A todos los profesores, que durante los cinco años me transmitieron sus experiencias ayudándome a llegar hasta aquí.

Hansel

A mi madre, que simplemente sin ella no estaría aquí hoy. Por estar ahí cada vez que necesité apoyo a pesar de tener sus propios problemas por resolver y estar siempre armando broncas por ello, siendo yo el principal objetivo de la descarga. Por ser la ayuda sin la cual no pudiese haber dedicado el tiempo que le dediqué a cumplir esta meta de mi vida, haciéndose cargo de lo que hiciera falta, porque siempre ha estado muy orgullosa de mí.

A Patani (mi hermana), que piensa que la odio, pero no es así, solo que a veces se excede un poco, pero la quiero igual.

A mi Tata, que supo entenderme cuando más lo necesité, y me apoyó como pudo en cada momento, comprendiendo qué tiempo era para ella y cual, para el trabajo, siempre conté con su apoyo.

A Hansel, mi compañero de tesis, por encargarse siempre de lo que hiciera falta para que esto saliera adelante.

A mis tutoras, que a pesar de estar ocupadas con deberes supieron encontrar tiempo para dedicarnos.

A los míos, ellos saben quiénes son, por cada momento bueno y por estar en los malos también. Por cada fiesta, cada playa, cada Vine.

A cada uno de los miembros de mi grupo durante la carrera, que no fueron pocos ni mucho menos, todos formaron una pequeña parte de mi camino hasta aquí, los que siguieron y los que desgraciadamente no también.

A los profesores cuyo trabajo, paciencia y dedicación me permitieron llegar hasta este momento, preparado para la vida profesional.

A todos los que se me quedan, y que saben que estuvieron ahí, gracias.

Danny

Dedicatoria

Le dedico esta tesis a mi familia:

A mis padres Sonia y Cesar por estar ahí en todo momento, por todos los sacrificios que han hecho para que yo pudiera estar hoy aquí, por todas las cosas que me han enseñado, por haber ayudado a convertirme en la persona que soy, por ser los padres que son que para mí son perfectos.

Hansel

A mi madre

Danny

RESUMEN

El balanceo de carga y la alta disponibilidad son aspectos críticos en centros donde se ofrezcan variados servicios de red; mantener un buen monitoreo de estos aspectos es fundamental para lograr una mayor estabilidad en cada uno de los servicios brindados.

En el Departamento de Componentes del Centro de Telemática de la Universidad de las Ciencias Informáticas no se cuenta con un sistema que permita la configuración y monitoreo de los servicios de balanceo de carga y alta disponibilidad.

La presente investigación está encaminada al estudio de los sistemas que utilizan el balanceo de carga y la alta disponibilidad para comprender su funcionamiento de forma que se puedan corregir las deficiencias detectadas con el desarrollo de un nuevo sistema que permita configurar y monitorear estos servicios mantenidos con las herramientas HAProxy y Keepalived. Dicho sistema debe ser capaz de mantener registros históricos de los balanceadores y de graficar los mismos para su posterior análisis.

En el transcurso de la investigación se exponen los pasos necesarios para desarrollar el sistema. Se presentan los conceptos relacionados con la investigación. Además, se seleccionan las herramientas, tecnologías y metodología que son usadas para el desarrollo del sistema.

Índice

| | |
|--|----|
| Introducción | 1 |
| Capítulo 1: Fundamentación Teórica de la Investigación. | 4 |
| 1.1. Introducción | 4 |
| 1.2. Conceptos fundamentales | 4 |
| 1.3. Análisis de soluciones | 6 |
| 1.3.1. Sistemas para la configuración de HAProxy | 6 |
| 1.3.2. Sistemas para la configuración de Keepalived | 7 |
| 1.3.3. Sistemas para el monitoreo de HaProxy | 7 |
| 1.4. Metodología de desarrollo | 12 |
| 1.5. Tecnologías y herramientas | 13 |
| 1.5.1. HAProxy | 13 |
| 1.5.2. Keepalived | 13 |
| 1.5.3. Lenguajes de programación: | 13 |
| • Python 2.7 | 13 |
| • HTML 5 | 13 |
| • CSS 3 | 14 |
| • JavaScript (JS) | 14 |
| 1.5.4. Framework de desarrollo | 14 |
| • JQuery 1.9 | 14 |
| • Backbone 1.1 | 14 |
| • Twitter Bootstrap 3.0 | 14 |
| • Django 1.7 | 15 |
| 1.5.5. Herramienta CASE: Visual Paradigm 5.0 | 15 |
| 1.5.6. Lenguaje de modelado: BPMN 2.0 | 15 |
| 1.5.7. Editor de código: Aptana 3.6 | 15 |
| 1.5.8. Sistema Gestor de Base de Datos (SGBD) | 15 |
| 1.5.9. Servidor web | 16 |
| 1.5.10. Herramientas auxiliares | 16 |
| 1.6. Conclusiones del capítulo | 17 |

| | |
|--|----|
| Capítulo 2: Propuesta de Solución | 18 |
| 2.1. Introducción | 18 |
| 2.2. Propuesta de solución | 18 |
| 2.3. Lista de Funcionalidades | 22 |
| 2.3.1. Funcionalidades del módulo de configuración: | 22 |
| 2.3.2. Funcionalidades del servicio:..... | 23 |
| 2.4. Lista de Reserva del Producto | 23 |
| 2.5. Personal relacionado con el sistema..... | 24 |
| 2.6. Fase de Exploración | 24 |
| 2.6.1. Historias de Usuario | 24 |
| 2.7. Fase de planificación | 26 |
| 2.7.1. Estimación de esfuerzo por HU | 26 |
| 2.7.2. Plan de iteraciones..... | 27 |
| 2.7.3. Plan de duración de las iteraciones | 27 |
| 2.7.4. Plan de entrega | 28 |
| 2.8. Conclusiones del capítulo | 28 |
| Capítulo 3: Diseño, Implementación y Pruebas | 29 |
| 3.1. Introducción | 29 |
| 3.2. Arquitectura del Sistema | 29 |
| 3.2.1. Arquitectura N-Tiers | 29 |
| 3.3. Patrón Arquitectónico..... | 31 |
| 3.3.1. Patrón de arquitectura Model Template View (MTV) | 31 |
| 3.4. Patrones de Diseño | 34 |
| 3.4.1. Patrones para Asignar Responsabilidades (GRASP) | 34 |
| 3.5. Tarjetas Clase – Responsabilidad – Colaborador (CRC) | 35 |
| 3.5.1. Tarjetas CRC pertenecientes al Módulo de Configuración..... | 35 |
| 3.5.2. Tarjetas CRC pertenecientes al Servicio Intérprete de Registros de HAProxy..... | 36 |
| 3.6. Diagrama de Clases Persistentes | 36 |
| 3.7. Diagrama Entidad-Relación | 37 |
| 3.8. Tareas de Ingeniería..... | 37 |
| 3.9. Pruebas del Sistema..... | 39 |

| | |
|---------------------------------------|-----------|
| 3.9.1. Pruebas de unidad | 39 |
| 3.9.2. Pruebas de Aceptación | 40 |
| 3.10. Conclusiones del capítulo | 43 |
| Conclusiones Generales | 44 |
| Recomendaciones | 45 |
| Referencias | 46 |
| Bibliografía Consultada | 48 |

Índice de figuras

| | |
|---|----|
| Figura #1: Fichero de configuración de HAProxy..... | 6 |
| Figura #2: Fichero de configuración de Keepalived Sistemas para el monitoreo de HAProxy | 7 |
| Figura #3: Ejemplo de Interfaz de Zabbix | 8 |
| Figura #4: Ejemplo de interfaz de Nagios | 9 |
| Figura #5: Ejemplo de interfaz de Datadog, integrado con HAProxy | 11 |
| Figura #6: Ejemplo de interfaz de Stats de HAProxy | 11 |
| Figura #7: Sistema para la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones. | 18 |
| Figura #8: Propuesta de sistema | 19 |
| Figura #9: Proceso de Negocio Gestionar Nodos del Clúster. | 20 |
| Figura #10: Proceso de Negocio Procesar Estadísticas de HAProxy | 21 |
| Figura #11: Arquitectura del sistema | 30 |
| Figura #12: Arquitectura Cliente-Servidor..... | 30 |
| Figura #13: Arquitectura de Flujo de Datos | 31 |
| Figura #14: Esquema de funcionamiento del patrón MTV de Django | 32 |
| Figura #15: Clases de la capa Modelo | 33 |
| Figura #16: Clases de la capa Plantilla..... | 33 |
| Figura #17: Clases de la capa Vista | 34 |
| Figura #18: Diagrama de Clases Persistentes..... | 36 |
| Figura #19: Diagrama Entidad-Relación..... | 37 |
| Figura #20: Interfaz de salida de pytest luego de las pruebas unitarias..... | 39 |

Figura #21: Ejemplos de las pruebas de unidad realizadas40

Figura #22: Resultados de las pruebas de aceptación43

Índice de tablas

Tabla #1: Ficha de Proceso Insertar Nodo20

Tabla #2: Ficha de Proceso Procesar Estadísticas de HAProxy22

Tabla #3: Historia de Usuario #3 Gestionar nodos del clúster25

Tabla #4: Estimación de esfuerzo por Historia de Usuario26

Tabla #5: Plan de duración de las iteraciones27

Tabla #6: Plan de entrega28

Tabla #7: Tarjeta CRC: Clase Node35

Tabla #8: Tarjeta CRC: Clase sis36

Tabla #9: Tarea de Ingeniería #16 Configurar los parámetros básicos de HAProxy38

Tabla #10: Tarea de Ingeniería #21 Traducir estadísticas38

Tabla #11: Caso de prueba Insertar Nodo.....41

Introducción

En la actualidad las tecnologías de la información forman parte indispensable en la vida del ser humano. La informática, a nivel mundial, supone una nueva revolución tecnológica, pues han sido numerosos los avances alcanzados en dicha ciencia, debido a la necesidad de información y de nuevos conocimientos que tiene el mundo actual, no únicamente para el desarrollo de los pueblos, sino también para el de las nuevas tecnologías.

El uso de las TIC (Tecnologías de la Información y las Comunicaciones) no para de crecer y de extenderse, tendiendo a ocupar un lugar creciente en la vida humana y el funcionamiento de las sociedades, como por ejemplo la Internet. (1)

La Internet se define como la red informática mundial, descentralizada, formada por la conexión directa entre computadoras mediante un protocolo especial de comunicación. (2) En ella se mantienen desde sitios web para la consulta de información hasta aplicaciones que ya son prácticamente parte de la vida de muchos, como las redes sociales (Facebook, Twitter, etc).

A medida que aumenta el número de usuarios de los sitios web, se hace más complicado atender todas las peticiones de cada uno de ellos. Para asegurar el mantenimiento del servicio se utiliza en muchos casos una técnica conocida como balanceo de carga, que se encarga de repartir equitativamente el trabajo entre distintos servidores o procesos, logrando así atender mayor cantidad de peticiones simultáneamente, o bien, atender las peticiones que se hacen normalmente en menor rango de tiempo, evitando los llamados cuellos de botella. Por otra parte, es necesario también garantizar que los servicios prestados estén disponibles en todo momento, ya que si ante cualquier contingencia estos dejaran de prestarse influiría negativamente en la calidad de los mismos y en la imagen de sus proveedores. Esto se conoce como alta disponibilidad (3).

Cuba va aparejada al creciente desarrollo de las tecnologías de la información, aportando resultados significativos en las diversas esferas. Se han desarrollado varios proyectos productivos en las diferentes provincias del país que revelan el conocimiento y habilidades de los profesionales que se forman hasta el momento en Cuba.

La Universidad de las Ciencias Informáticas (UCI) ocupa un lugar destacado en la producción de software en el país. En ella se han desarrollado múltiples proyectos productivos que benefician el desarrollo económico y tecnológico del país, y ayudan a la adquisición de nuevos conocimientos. El departamento de Desarrollo de Componentes del Centro Telemática (TLM) de dicha universidad brinda variados servicios telemáticos tanto a la universidad como a empresas u organizaciones estatales. Uno de ellos es el aseguramiento de la alta disponibilidad y el balanceo de carga en aplicaciones utilizando las herramientas de software libre HAProxy y Keepalived.

El departamento actualmente no cuenta con una herramienta que permita la configuración de dicho servicio. Esta se realiza a través de una consola de comandos en la cual el administrador de los

servicios debe conocer donde se encuentran los ficheros de configuración de cada herramienta, conocer cada parámetro, sus posibles valores y uso, tornando el proceso bastante engorroso. Por otra parte, HAProxy ofrece una interfaz llamada Stats que permite realizar un monitoreo de informaciones que son de interés para el administrador, como son: número total de sesiones, máximo concurrente de sesiones, sesiones por segundo, sesiones activas en cada nodo, cantidad de bytes de entrada y de salida, errores en las conexiones y en las respuestas desde el servidor, incluyendo el estado actual de los servidores. Pero esta información se muestra de manera acumulativa, o sea, no admite acotar los datos a un determinado periodo de tiempo. Esto impide realizar estudios más precisos del comportamiento de los servidores y clústeres, como en qué días hay mayor número de usuarios conectados, cuantos errores como promedio ocurren cada semana o si existió algún instante de tiempo donde se alcanzó el máximo de sesiones permitidas abiertas en los servidores.

Por lo anteriormente planteado se presenta el siguiente **problema a resolver**: ¿cómo configurar y monitorear los servicios de alta disponibilidad y balanceo de carga en aplicaciones mediante las herramientas HAProxy y Keepalived? Considerando como **objeto de estudio** la configuración y monitoreo de alta disponibilidad y balanceo de carga y tomando como **campo de acción** la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones mediante las herramientas HAProxy y Keepalived. Se definió el siguiente **objetivo general**: desarrollar un sistema para la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones mediante las herramientas HAProxy y Keepalived.

Las siguientes **tareas de la investigación** están dirigidas a dar cumplimiento a los objetivos del trabajo:

- Caracterización de las herramientas más utilizadas en Cuba y el mundo para la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones con HAProxy y Keepalived.
- Selección de la metodología de desarrollo de software y las herramientas a utilizar para el desarrollo del sistema.
- Definición de las funcionalidades que debe brindar el sistema para la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones con HAProxy y Keepalived.
- Diseño e implementación del sistema para la configuración y monitoreo de alta disponibilidad y balanceo de carga y en aplicaciones con HAProxy y Keepalived.
- Realización de las pruebas del sistema para comprobar que las respuestas del mismo sean las esperadas.

Para apoyar el desarrollo de la investigación se emplean los siguientes **métodos científicos**:

Métodos Teóricos:

- **Analítico-Sintético:** este método permitirá analizar las teorías presentadas en las bibliografías consultadas y extraer los elementos más importantes relacionados con el objeto de estudio, necesarios para dar solución al problema planteado.
- **Modelación:** este método se utilizará para crear modelos y diagramas con el objetivo de entender el funcionamiento del proceso de interpretación de los Stats generados por HAProxy.

Métodos Empíricos:

- **Observación:** este método permitirá observar cómo se realiza el proceso de interpretación de los Stats generados por HAProxy. Permite analizar y comparar los resultados obtenidos de las pruebas realizadas al sistema desarrollado.
- **Entrevista:** este método se utilizará para recopilar información mediante conversaciones con el cliente, para satisfacer sus necesidades y obtener un producto con la calidad requerida.

El trabajo está estructurado en 3 capítulos, los cuales se describen a continuación:

Capítulo 1: Fundamentación teórica de la investigación. En este capítulo se estudian los conceptos fundamentales que se tratan a lo largo de la investigación, los diferentes sistemas similares existentes y su funcionamiento, además se definen las herramientas, tecnologías y la metodología a utilizar.

Capítulo 2: Propuesta de Solución. En este capítulo se propone una solución a partir de una descripción detallada de las funcionalidades implementadas. Se describen las historias de usuario que se desarrollarán en cada iteración y se define un plan de entrega del producto.

Capítulo 3: Diseño, Implementación y Pruebas. En este capítulo se realiza la descripción del sistema, se definen los patrones de diseño, la arquitectura y las clases de diseño. También se describen los elementos de la implementación y las pruebas aplicadas al sistema para validar su correcto funcionamiento.

Capítulo 1: Fundamentación Teórica de la Investigación.

1.1. Introducción

En este capítulo se realiza un estudio sobre el balanceo de carga y la alta disponibilidad en clústeres de aplicaciones, se exponen los conceptos fundamentales del balanceo de carga y la alta disponibilidad, las tecnologías y herramientas a utilizar, así como la metodología de desarrollo.

1.2. Conceptos fundamentales

- **Balanceo de carga**

El balanceo de carga es una técnica cuyo objetivo fundamental es dividir el total de trabajo que un sistema tiene que hacer entre dos o más sistemas. Lo cual permite realizar el mismo trabajo en una porción de tiempo más reducida, lo que es equivalente a realizar mayor cantidad de trabajo en el mismo tiempo total. Existen varias formas de realizar el balanceo de carga entre las que destaca el balanceo de carga por software. El balanceo de carga está indicado para sistemas en los que es muy difícil prever la cantidad de trabajo que este debe realizar (3).

- **Balanceador de carga**

Un balanceador de carga fundamentalmente es un dispositivo de hardware o software que se pone al frente de un conjunto de servidores que atienden una aplicación y, tal como su nombre lo indica, asigna o balancea las solicitudes que llegan de los clientes a los servidores usando algún algoritmo (desde un simple Round Robin hasta algoritmos más sofisticados).

El servidor proxy inverso protege a los servidores HTTP internos proporcionando un punto de acceso único a la red interna, a partir de ello ofrece ventajas de seguridad y características de acceso a red:

- El administrador puede utilizar las características de autenticación y control de acceso del servidor proxy inverso para controlar quién puede acceder a los servidores internos y controlar a qué servidores puede acceder cada usuario individual. Cuando se despliega un servidor proxy inverso, el proceso de autenticación y derechos de acceso a varios servidores internos puede controlarse desde una sola máquina, lo cual simplifica la configuración de la seguridad.
- Todo el tráfico hacia los servidores de la intranet parece dirigido a una única dirección de red (la dirección del servidor proxy inverso).
- Cuando se despliega un servidor proxy inverso, sólo aquellos URL asociados con el servidor proxy inverso aparecerán como públicos de cara a los usuarios del navegador web. Los

usuarios de Internet utilizarán estos URL para acceder al servidor proxy inverso. El servidor proxy inverso maneja estas solicitudes de los usuarios de Internet y las redirige al servidor HTTP interno correspondiente.

- El administrador realiza configuraciones de correlación de URL en el servidor proxy inverso que hace esta redirección posible. Cuando se configura el servidor proxy inverso, el administrador correlaciona los URL utilizados para acceder al servidor proxy inverso con los URL reales de los servidores HTTP internos. Cuando un usuario de Internet envía un URL al servidor proxy inverso, el servidor proxy inverso examina el URL y utiliza estas configuraciones de correlación (o reglas) para reescribir el URL.
- El servidor proxy inverso reescribe el URL sustituyendo la dirección del servidor proporcionada por el usuario de Internet (una dirección de proxy inverso) con la dirección real del servidor interno. La solicitud HTTP se envía entonces a la red interna desde el servidor proxy inverso al servidor interno.
- Todo el tráfico enviado a los usuarios de Internet desde los servidores internos parece proceder de una única dirección de red.
- Cuando un servidor HTTP interno responde a una solicitud de un usuario de Internet, el servidor interno envía la respuesta al servidor proxy inverso y el servidor proxy inverso envía la respuesta al usuario de Internet. La respuesta enviada en Internet al usuario de Internet contiene la dirección del servidor proxy inverso, no la dirección del servidor HTTP interno.

(4)

Los balanceadores se utilizan para incrementar la capacidad de procesamiento y confiabilidad, estos aseguran la disponibilidad monitorizando el estado de las aplicaciones y enviando las peticiones hacia los servidores que puedan responder (5).

- **Balanceo de carga por software**

El balanceo de carga por software se proporciona en una de dos maneras: como parte de un sistema operativo o como una aplicación complemento. El balanceo de carga que se implementa como una solución basada en software a menudo puede ofrecer la facilidad de despliegue y mantenimiento, así como un rendimiento similar a la de soluciones basadas en hardware (3).

- **Alta disponibilidad**

La disponibilidad es una de las características de los sistemas, que mide el grado en que los recursos del mismo están disponibles para su uso por el usuario final a lo largo de un tiempo dado. Esta no solo se relaciona con la prevención de caídas del sistema, sino incluso con la percepción de “caída” desde el punto de vista de los usuarios, las cuales van desde tiempos de respuestas prolongados hasta la escasa asistencia técnica. (5)

- **Clúster de Alta Disponibilidad**

Están diseñados para garantizar el funcionamiento permanente de ciertas aplicaciones. Su propio funcionamiento es brindar un servicio las 24 horas del día los siete días de la semana. Su arquitectura está compuesta por un grupo de dos o más máquinas que se caracterizan por la redundancia de sus recursos y porque se monitorean constantemente entre sí, en caso de que exista un fallo de la aplicación en alguna de las máquinas o del hardware en algún lugar del clúster, el software de alta disponibilidad debe ser capaz de migrar automáticamente los servicios que han fallado hacia las otras máquinas que componen el clúster. (5)

1.3. Análisis de soluciones

1.3.1. Sistemas para la configuración de HAProxy

El proceso de configuración de HAProxy se realiza modificando el fichero de configuración mediante líneas de comando en la terminal. Como resultado del estudio de los métodos para la configuración de HAProxy se identificaron los parámetros básicos y avanzados que necesitan ser configurados para el correcto funcionamiento de HAProxy. A continuación, se muestra una figura con los parámetros de configuración de la herramienta:

```
global                                     #Configuración básica
    log 127.0.0.1   local0
    log 127.0.0.1   local1 notice
    #log loghost    local0 info
    maxconn 4096
    #debug
    #quiet
    user haproxy
    group haproxy

defaults
    log          global
    mode http
    option httplog
    option dontlognull
    retries 3
    redispatch
    maxconn 2000
    contimeout 5000
    clitimeout 50000
    srvtimeout 50000

listen webfarm *:80                       #Aquí empieza la configuración del clúster de aplicación en específico, con un nombre y puerto para escuchar
    mode http
    stats enable                            #Para poder acceder a los stats desde el navegador
    stats auth admin:admin                  #Usuario y contraseña para acceder a los stats
    balance roundrobin                     #Algoritmo de balanceo usado
    cookie JSESSIONID prefix
    option httpclose
    option forwardfor
    option httpchk GET /
    server webA 192.168.0.11:80 cookie A check #Nombre, dirección, puerto y cookie de cada servidor a revisar para la obtención de los stats
    server webB 192.168.0.12:80 cookie B check
```

Figura #1: Fichero de configuración de HAProxy

1.3.2. Sistemas para la configuración de Keepalived

Para llevar a cabo la configuración de Keepalived es necesario modificar el archivo de configuración. La configuración de este archivo se realiza mediante líneas de comando ejecutadas en la terminal del sistema. Luego de realizar el estudio se identificaron los parámetros básicos que necesitan ser configurados para el correcto funcionamiento de Keepalived. Dichos parámetros se muestran en la siguiente figura:

```
vrrp_script chk_haproxy {          #Configuración básica para la integración con HAProxy
    script "killall -0 haproxy"
    interval 2
    weight 2
}

vrrp_instance VI_1 {
    interface eth0
    state MASTER                   #Define si el servidor es MASTER o BACKUP. Solo puede haber un único servidor MASTER
    virtual_router_id 51
    priority 101                   # 101 en master, 100 en backup
    virtual_ipaddress {           # Dirección IP flotante para el clúster de aplicación
        192.168.0.10
    }
    track_script {
        chk_haproxy
    }
}
```

Figura #2: Fichero de configuración de Keepalived Sistemas para el monitoreo de HAProxy

1.3.3. Sistemas para el monitoreo de HaProxy

Para el monitoreo de redes existen variados sistemas. En este caso se analizan solamente los que se pueden integrar con HAProxy y que constituyen software libre, enfocando sobre todo aquellos más utilizados en entornos empresariales. Los sistemas existentes que cumplen con dichas características, de acuerdo a fuentes como Google Trends y Capterra, son los presentados aquí:

- **Zabbix**

Zabbix es un Sistema de Monitoreo de Redes creado por Alexei Vladishev. Está diseñado para monitorear y registrar el estado de varios servicios de red, servidores, y hardware de red.

Zabbix controla todos los informes y estadísticas, así como los parámetros de configuración y se accede a través de una interfaz basada en una web final. Estar basado en la web asegura que el estado de su red y la salud de los servidores pueden ser evaluados desde cualquier ubicación.

Correctamente configurado, Zabbix puede desempeñar un papel importante en la supervisión de la infraestructura de las tecnologías de la información. Esto es igualmente cierto en el caso de pequeñas organizaciones con pocos servidores y para las grandes empresas con una multitud de los servidores.

Zabbix es libre de costo, está escrito y distribuido bajo la licencia GPL (General Public License) versión 2. Esto significa que su código fuente es distribuido libremente y se encuentra disponible para el público en general.

Ventajas de Zabbix

- Auto-descubrimiento de servidores y dispositivos de red.
- Servidor para Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X.
- Garantiza la autenticación de los usuarios.
- Flexibles permisos de usuarios.
- Interfaz basada en la web.
- Posee un sistema de notificación de eventos a través de E-mail. (6)

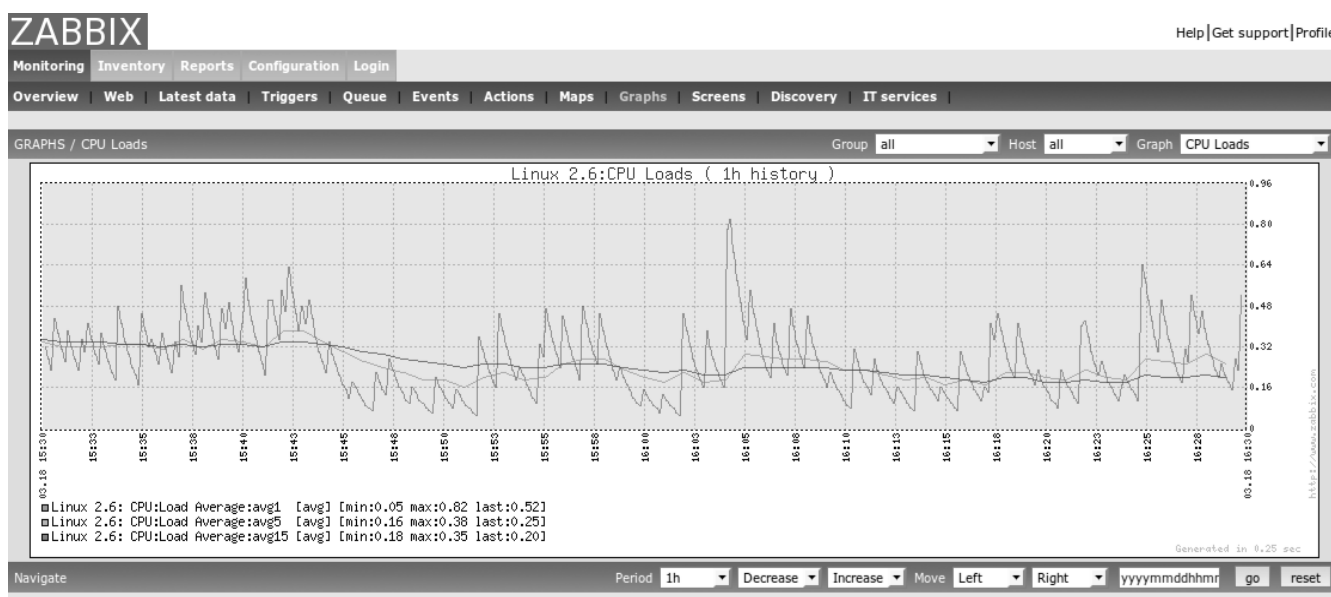


Figura #3: Ejemplo de Interfaz de Zabbix

Es capaz de descubrir automáticamente frontends y backends y agregarlos a la configuración de HAProxy, pero para ello es necesaria la configuración de un script. Dicho script presenta complicaciones para su composición, pues son necesarios conocimientos avanzados de la herramienta y de programación, a la vez de que se debe conocer donde colocar dicho script con permisos de ejecución. Sumado a este, se deben programar y copiar en determinados directorios: un script para acceder a los Stats de HAProxy y otro que constituye una plantilla HTML para especificar qué datos almacenar en la caché de la herramienta para su posterior monitoreo, que debe importarse utilizando para ello la interfaz que ofrece Zabbix. También es necesario configurar HAProxy para que escuche en una ruta que corresponde a un socket determinado, el cual debe conocerlo el usuario a priori. Se puede ver un ejemplo de estos scripts en el [Anexo 6](#).

Se puede observar que el proceso se mantendría complicado y bastante engorroso y no cumpliría con las necesidades del departamento. Sumándose a esto, no mantiene registros históricos de la información que va recopilando.

- **Nagios**

Nagios es un sistema de monitorización de redes ampliamente utilizado, de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. (7)

Nagios tiene la siguiente estructura:

Un núcleo de la aplicación que forma la lógica de control de negocio de la aplicación contiene el software necesario para realizar la monitorización de los servicios y máquinas de la red para la que ofrece soporte. Hace uso de diversos componentes que vienen con la aplicación, y puede hacer uso de otros componentes realizados por terceras personas.

Ventajas

- La verificación de disponibilidad se delega en plugins.
- Programación de chequeos inteligente.
- Configuraciones muy detalladas y basadas en plantillas.
- Utiliza información topológica para determinar dependencias.
- Permite definir políticas de notificación. (7)

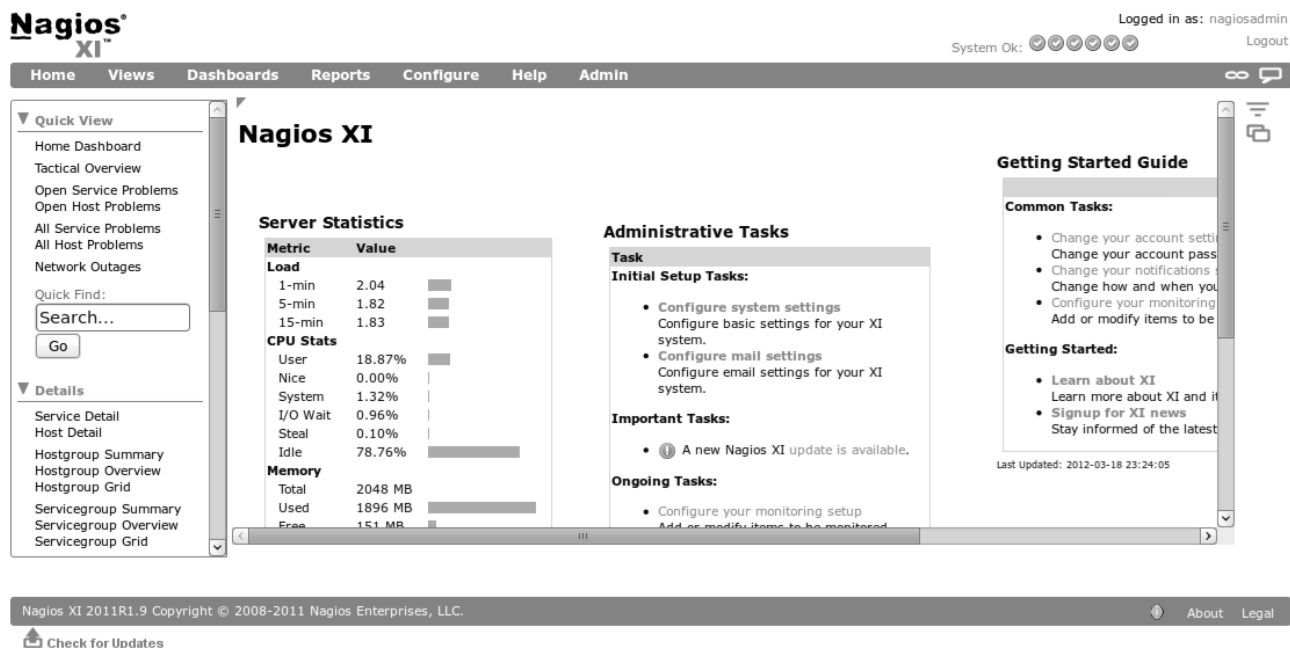


Figura #4: Ejemplo de interfaz de Nagios

En este caso, la integración con HAProxy se puede realizar mediante el uso de un plugin para ello. Actualmente hay varios ya implementados y que son libres. Sin embargo, esta integración llega solamente hasta la obtención de los Stats para generar una salida de los mismos en el sistema sin la posibilidad de graficarlos para su análisis y comparación. Además, los Stats obtenidos son los

ofrecidos por HAProxy de forma acumulativa. Este es un ejemplo de salida usando uno de los plugins disponibles para la integración con HAProxy:

```
HAPROXY OK - cluster1 (Active: 60/60) cluster2 (Active: 169/169) |
t=0.131051s;2;10;0; sess_cluster1=0sessions;;;0;20000
sess_cluster2=78sessions;;;0;20000
```

- **Datadog**

Datadog es una plataforma de monitoreo y análisis basada en SaaS (Software as a Service) para operaciones y equipos de desarrollo. Une datos de servidores, bases de datos, aplicaciones, herramientas y servicios para presentar una vista unificada de las aplicaciones que corren a escala en la nube. Datadog usa un agente de código abierto escrito en Python para recoger métricas y eventos. (8)

Más que paneles sumarios, Datadog permite ver los gráficos de todas las fuentes en tiempo real, dividir los datos por el host o dispositivo, y consultar las estadísticas sobre el uso de recursos de hardware. Datadog ofrece la posibilidad de consultar las estadísticas de forma acumulativa, o como promedio, así como personalizar visualizaciones, de forma interactiva o en código.

Configurar Datadog no es una tarea difícil, y su integración con HAProxy se realiza creando un fichero de configuración para el sistema, especificando la URL y credenciales para acceder a los Stats de HAProxy. Luego de reiniciar el sistema se puede establecer la integración desde la interfaz de Datadog. A continuación, un ejemplo del contenido de un fichero de configuración para la obtención de métricas de HAProxy por parte de Datadog:

```
init_config:

instances:
  - url: http://localhost/admin?stats
    # username: username
    # password: password
```

Datadog muestra las gráficas con los Stats que desee monitorear el usuario, además de brindar otras que muestran comportamientos promedio, pero solo permite mostrar 8 de los más de 20 datos distintos que ofrece la interfaz de HAProxy. Se puede analizar el comportamiento de los servidores desde hace una determinada cantidad de tiempo hasta el día actual, lo que a veces genera gráficas muy extensas, haciendo el análisis de estas engorroso para el usuario. A continuación, se muestra un ejemplo de interfaz de Datadog en integración con HAProxy, monitoreando 7 de los 8 Stats permitidos por el sistema:

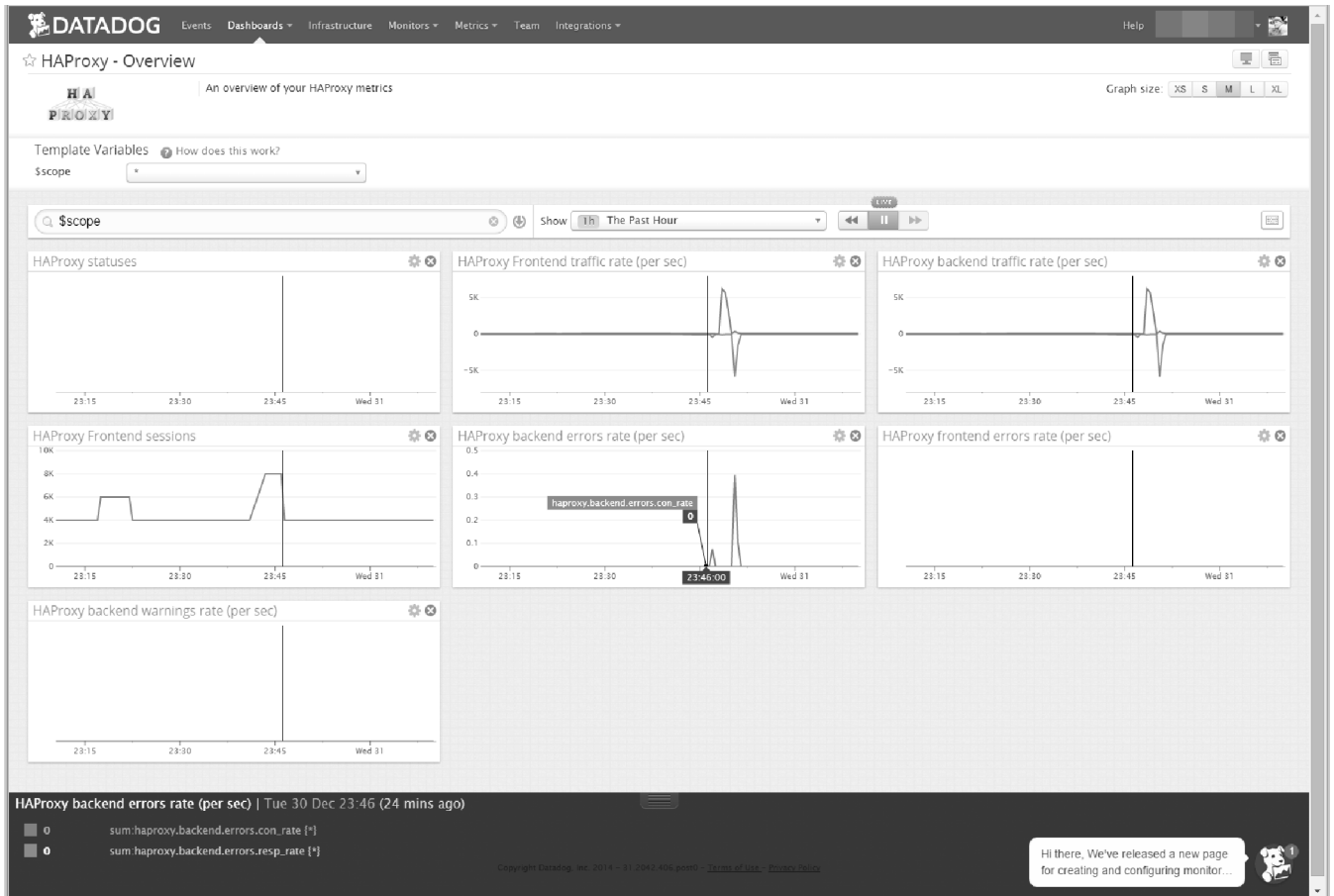


Figura #5: Ejemplo de interfaz de Datadog, integrado con HAProxy

- **Stats de HAProxy**

HAProxy ofrece una interfaz donde se pueden observar las estadísticas del comportamiento de los servidores. Estas estadísticas son almacenadas de forma acumulativa desde el instante en que se inicia el servicio. HAProxy tiene dos formas de obtener dichas estadísticas: mediante la consola o mediante una interfaz web que se habilita para este fin. Dicha herramienta no mantiene un historial de la información recogida y esta es mostrada de forma poco atractiva visualmente (mediante tablas), dificultando su entendimiento.

HAProxy version 1.3.15.2, released 2008/06/21

Statistics Report for pid 1802

> General process information

pid = 1802 (process #1, nbproc = 1)
 uptime = 0d 0h06m00s
 system limits : memmax = unlimited ; ulimit-n = 8205
 maxsock = 8205
 maxconn = 4096 (current conns = 1)

active UP backup UP
 active UP, going down backup UP, going down
 active DOWN, going up backup DOWN, going up
 active or backup DOWN not checked

Note: UP with load-balancing disabled is reported as "NOLB".

Display option:
 • Hide 'DOWN' servers
 • Refresh now
 • CSV export

External resources:
 • Primary site
 • Updates (v1.3)
 • Online manual

| | Queue | | | Sessions | | | LbTot | Bytes | | Denied | | Errors | | Warnings | | Status | Server | | | | | | |
|----------|-------|-----|-------|----------|-----|-------|-------|-------|-------|--------|------|--------|------|----------|------|--------|--------|------|-----|-----|-----|------|---------|
| | Cur | Max | Limit | Cur | Max | Limit | | In | Out | Req | Resp | Req | Conn | Resp | Retr | | Redis | Wght | Act | Bck | Chk | Dwn | Downtme |
| Frontend | | | | 1 | 2 | 2000 | 13 | 5027 | 17201 | 0 | 0 | 0 | 0 | 0 | 0 | OPEN | | | | | | | |
| webA | 0 | 0 | - | 0 | 1 | - | 5 | 1 | 2162 | 1519 | | 0 | 0 | 0 | 0 | 6m UP | 1 | Y | - | 0 | 0 | 0s - | |
| webB | 0 | 0 | - | 0 | 1 | - | 4 | 1 | 1637 | 1370 | | 0 | 0 | 0 | 0 | 6m UP | 1 | Y | - | 0 | 0 | 0s - | |
| Backend | 0 | 0 | | 0 | 1 | 2000 | 9 | 2 | 5027 | 17201 | 0 | 0 | 0 | 0 | 0 | 6m UP | 2 | 2 | 0 | 0 | 0 | 0s | |

Figura #6: Ejemplo de interfaz de Stats de HAProxy

Una vez finalizado el estudio de los sistemas para el monitoreo de HAProxy se concluye que es necesario implementar un nuevo sistema para llevar a cabo dicho monitoreo debido a que los existentes, en general, ofrecen estadísticas de forma acumulativa, a excepción de Datadog que, aunque muestra registros históricos, lo hace en la misma gráfica, lo que puede provocar que los gráficos se vuelvan difíciles de interpretar. También se observa que las interfaces son poco atractivas visualmente, como es el caso de Nagios y Stats de HAProxy, dificultando el análisis del comportamiento de los servidores por parte de los usuarios, y que la configuración se puede tornar compleja para estos tratándose de Zabbix.

1.4. Metodología de desarrollo

Las metodologías de desarrollo son un conjunto de procedimientos y técnicas para el desarrollo de productos de software. Estas se pueden clasificar en ágiles o tradicionales.

Las metodologías tradicionales están enfocadas al estricto control del proceso, estas han demostrado ser eficientes para proyectos de gran tamaño. Dentro de las metodologías tradicionales se pueden encontrar a METRICA y Proceso Unificado Relacional (RUP por sus siglas en inglés).

En las metodologías ágiles las relaciones entre los individuos son más importantes que las herramientas empleadas en el proceso de desarrollo del software. Las metodologías ágiles se enfocan más en crear un software que funcione que en mantener una documentación extensa y detallada de todo el proceso de desarrollo. El intercambio colaborativo con el cliente debe ser intenso. Es más importante la reacción ante un posible cambio que darle seguimiento a un plan estricto de desarrollo. Algunos ejemplos de estas metodologías son SCRUM y Programación Extrema (XP por sus siglas en inglés). (9)

- **Programación Extrema (XP)**

XP es una metodología ágil que se basa en las relaciones interpersonales como clave para el desarrollo de software propiciando un buen ambiente de trabajo. Basa su funcionamiento de la retroalimentación continua entre el cliente y el equipo de desarrollo además de la simplicidad en las soluciones implementadas. (9).

Para el presente trabajo se selecciona la metodología XP ya que el mismo es un proyecto pequeño y de poco tiempo de desarrollo; el equipo de trabajo está formado por dos programadores entre los cuales existe un alto nivel de interacción debido a su buena comunicación y entendimiento, además el cliente radica en el mismo centro de producción lo que posibilita un mayor intercambio de opiniones con el objetivo de lograr un producto que satisfaga las necesidades del cliente.

1.5. Tecnologías y herramientas

El lenguaje y las herramientas empleadas en el desarrollo del sistema fueron definidas por el centro TLM, que siguiendo las pautas de diseño establecidas por la universidad cuenta con el framework Xilema-Base-Web para el desarrollo de sus aplicaciones.

1.5.1. HAProxy

HAProxy es una solución rápida y eficaz para ofrecer balance de carga y funciones de proxy para las aplicaciones basadas en HTTP y TCP (Transmission Control Protocol). Es particularmente adecuado para sitios web de tráfico elevado. (10)

1.5.2. Keepalived

Keepalived es un software escrito en C cuyo principal objetivo es proveer medios para mantener alta disponibilidad en sistemas e infraestructuras basadas en Linux de una manera sencilla y robusta, basado en el protocolo VRRP (Virtual Router Redundancy Protocol, Protocolo de Redundancia de Enrutador Virtual) (11)

1.5.3. Lenguajes de programación:

- **Python 2.7**

Es un lenguaje de programación interpretado cuya filosofía fomenta una sintaxis que permite mantener un código enteramente legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y en menor medida programación funcional.

Python posee un tipado dinámico además de ser multiplataforma, es administrado por la Python Software Foundation, el lenguaje posee una licencia de código abierto que es compatible con la Licencia Pública General de GNU. (12)

Se utiliza por ser un lenguaje potente para el desarrollo de aplicaciones web, posee una gran cantidad de librerías disponibles; la sencillez en la sintaxis de Python hace que el proceso de mantenimiento de las aplicaciones sea más fácil y rápido. Se usa esa versión por ser base del framework Xilema-base-web.

- **HTML 5**

Es la evolución de HTML (Hypertext Markup Language) el lenguaje en el que es creada la web, este agrupa las nuevas tecnologías para el desarrollo de aplicaciones web. HTML5 introduce etiquetas que permiten la publicación de archivos de audio y video. Se utilizará HTML5 para el desarrollo de las interfaces del sistema para así aprovechar su soporte para CSS3. (13)

- **CSS 3**

Es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML (Extensible Hypertext Markup Language). CSS (Cascade Style Sheet) es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas (14). El lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista.

- **JavaScript (JS)**

Es un lenguaje ligero, interpretado y orientado a objetos. Utilizado como lenguaje de scripts para las páginas web, es un lenguaje multiparadigma, se utiliza mayormente del lado del cliente creando efectos atractivos en las páginas web. (15) Se utilizará este lenguaje para manejar los datos del lado del cliente.

1.5.4. Framework de desarrollo

Un framework de desarrollo es un ambiente que contiene librerías y módulos que pueden ser reutilizados para facilitar el desarrollo de una aplicación. En el centro TLM se decidió utilizar Django para el desarrollo del sistema.

- **JQuery 1.9**

JQuery es un framework para el lenguaje JavaScript, es de código abierto que está licenciado bajo la Licencia Pública General de GNU v2, cuyo objetivo es simplificar la creación de páginas web responsivas, además ofrece una infraestructura que aporta facilidad para el desarrollo de aplicaciones web complejas del lado del cliente. JQuery aporta mucha ayuda para crear interfaces de usuario, además provee a nuestra página web efectos dinámicos (16).

- **Backbone 1.1**

Backbone es un pequeño framework que permite construir aplicaciones usando JavaScript, basado en el paradigma de diseño de aplicaciones Modelo Vista Controlador. Su objetivo consiste en probar y definir un conjunto de estructuras de datos junto al manejo de la interfaz por medio de vistas que son útiles cuando se construyen aplicaciones JavaScript. (17)

- **Twitter Bootstrap 3.0**

Bootstrap es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Es un framework potente con numerosos componentes web que ahorrará mucho

esfuerzo y tiempo. (18) Se empleará para la creación de las interfaces, ofreciendo una buena integración con librerías como JQuery.

- **Django 1.7**

Django es un framework de desarrollo web de código abierto escrito en Python que utiliza permite la construcción de aplicaciones web de forma rápida utilizando la menor cantidad de código. Implementa una modificación del Modelo-Vista-Controlador (MVC), llamada MTV (Modelo-Vista-Plantilla por sus siglas en inglés). Django se centra en automatizar todo lo posible para facilitar el desarrollo (19). Para el desarrollo del sistema informático se utiliza en esa versión debido a que es la base de Xilema-base-web.

1.5.5. Herramienta CASE: Visual Paradigm 5.0

Visual Paradigm es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Es una herramienta con disponibilidad en múltiples plataformas y soporta varios Sistemas Gestores de Bases de Datos como Oracle, Microsoft SQL Server, MySQL PostgreSQL y SQLite. También permite el uso de las distintas metodologías propias de la Ingeniería de Software. (9)

1.5.6. Lenguaje de modelado: BPMN 2.0

BPMN (Business Process Model and Notation) provee a los negocios con la capacidad de entender su funcionamiento interno en una notación gráfica y les da a las organizaciones la habilidad de comunicar sus procedimientos de manera estandarizada. (20)

1.5.7. Editor de código: Aptana 3.6

Para el desarrollo del sistema se empleará el editor de texto Aptana Studio, es un editor de texto que ofrece características avanzadas de programación para lenguajes como HTML, JavaScript, CSS y Python. Aptana ofrece herramientas para el trabajo con bases de datos y un marcado de sintaxis mediante colores. (21)

1.5.8. Sistema Gestor de Base de Datos (SGBD)

Un SGBD es un conjunto de programas que permiten la creación de base de datos y proporciona herramientas para añadir, borrar, modificar, y eliminar datos, además de mantener la integridad de estos.

- **PostgreSQL 9.3**

PostgreSQL es un potente SGBD objeto relacional de código abierto, tiene soporte completo para claves foráneas, vistas y disparadores. Incluye la mayoría de los tipos de datos de SQL 2008

incluyendo integer, boolean y varchar entre otros. También soporta el almacenamiento de objetos binarios grandes como imágenes, sonido o video. Algunas de las características de PostgreSQL son:

- Arquitectura cliente servidor con un amplio rango de drivers.
- Diseño de alta concurrencia donde lectores y escritores no se bloquean.
- Altamente confiable y extensible para muchos tipos de aplicación.
- Optimizador de consultas sofisticado.
- Incluye herencia entre tablas. (22)

1.5.9. Servidor web.

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. (23)

- **NGINX 1.2**

Se trata de un **servidor web/proxy** completamente inverso, que tiene como principal característica ser sumamente ligero, lo que lleva a su otro gran atractivo, su velocidad, lo que nos permite servir **aplicaciones web** con una velocidad muy superior a la de sus competidores más directos. Podemos decir entonces, que es un **servidor web de alto rendimiento**, ideal para realizar todo tipo de trabajos, ya sean profesionales o aficionados.

Características de NGINX:

- Capacidad para manejar más de 10,000 conexiones simultáneas con bajo uso de memoria
- Tolerancia a Fallos
- Servidores Virtuales basados en nombre y dirección IP
- Autenticación de acceso
- Manejo de ancho de banda

Como servidor web se selecciona NGINX por ser definido por el cliente, ser multiplataforma, poseer un alto rendimiento y consumir pocos recursos en el ordenador.

1.5.10. Herramientas auxiliares

- **Paramiko**

Paramiko es una implementación Python del protocolo SSHv2 (Secure Shell), proveyendo las funcionalidades de cliente y de servidor. Se usa para el desarrollo del sistema ya que lo dota de una herramienta para realizar conexiones a los balanceadores, permitiendo la configuración de HAProxy y Keepalived de manera remota.

- **BeautifulSoup 3**

Se trata de una librería Python para obtener información de ficheros HTML y XML. Provee formas de navegar, buscar y modificar el árbol gramatical de estos ficheros. (24) Se usa para el desarrollo del sistema a la hora de obtener los Stats de HAProxy.

- **Twisted**

Es un framework de red para programación dirigida por eventos escrito en Python. Soporta una gran variedad de protocolos, incluyendo TCP, UDP, HTTP y SSH. (25) Se usa en el desarrollo del sistema para ofrecerle a sus componentes aislados físicamente la posibilidad de interacción entre sí.

1.6. Conclusiones del capítulo

Luego de realizar un estudio de la tecnología de clúster, los sistemas balanceadores de carga que existen, además del software necesario para su implementación se concluye:

- El balanceo de carga es tratado por HAProxy. Keepalived asegura la alta disponibilidad de los clústeres de HAProxy.
- Es necesaria la implementación de un nuevo sistema para configurar y monitorear los servicios de balanceo de carga y alta disponibilidad en aplicaciones con HAProxy y Keepalived.
- Como servidor de aplicaciones web se podrá usar cualquiera de los disponibles ya que todos usan el protocolo HTTP y por ello ofrecen integración con HAProxy.
- Se definió la metodología de desarrollo a usar para la implementación de la solución, así como las herramientas y lenguajes para la misma.

Capítulo 2: Propuesta de Solución

2.1. Introducción

El siguiente capítulo está dedicado al diseño de la propuesta de solución del sistema para la configuración de balanceo de carga y alta disponibilidad en clústeres de aplicaciones. Se definirán las funcionalidades del sistema, además de las historias de usuario en correspondencia con la metodología de desarrollo seleccionada.

2.2. Propuesta de solución

La solución propuesta está diseñada para todas aquellas instituciones que necesiten configurar la alta disponibilidad, balanceo de carga y mantener un monitoreo del comportamiento de sus servidores de aplicación. La propuesta está conformada por dos módulos. El primer módulo es un servicio capaz de traducir el registro Stats generado por HAProxy a un formato entendible y almacenarlos en una base de datos. El segundo módulo permite la configuración de los parámetros fundamentales de HAProxy y Keepalived, así como la monitorización de los servidores web. El presente sistema permitirá a los administradores mantener un monitoreo constante de los servicios brindados, así como tomar decisiones basadas en el estudio de los datos recopilados por el sistema. En la figura 1 se muestra cómo está conformado el Sistema para la Configuración, Monitoreo de Alta Disponibilidad y Balanceo de Carga en Clústeres de Aplicaciones.

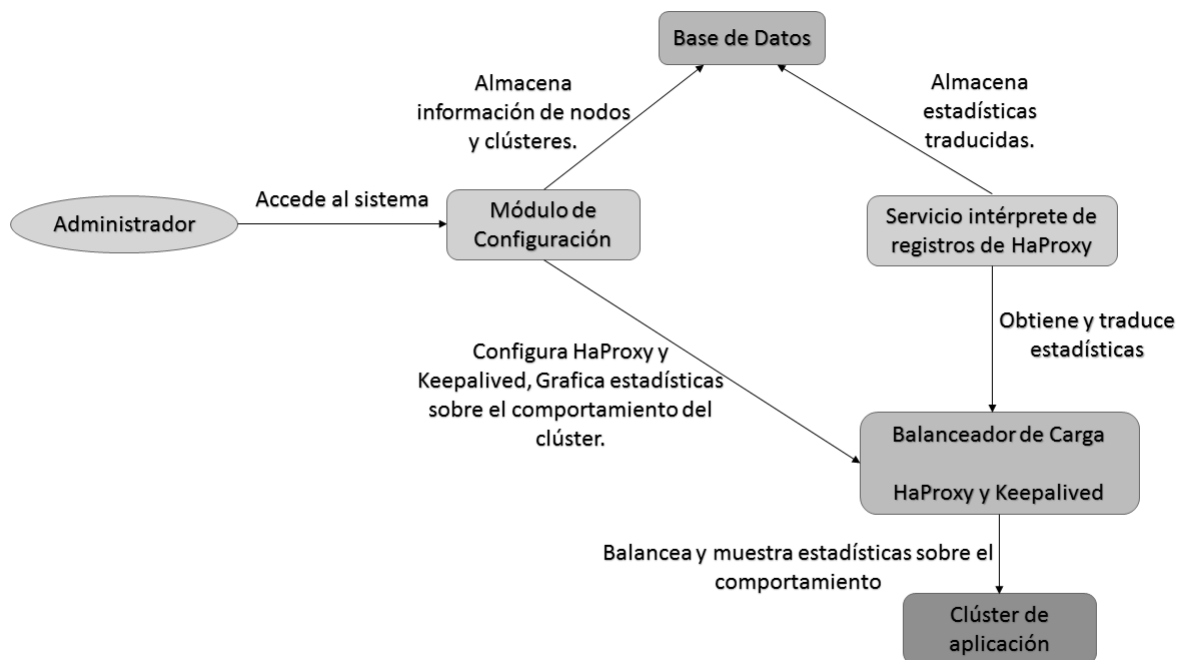


Figura #7: Sistema para la configuración y monitoreo de alta disponibilidad y balanceo de carga en aplicaciones.

Para un mejor entendimiento de la propuesta de sistema se muestra la siguiente figura:

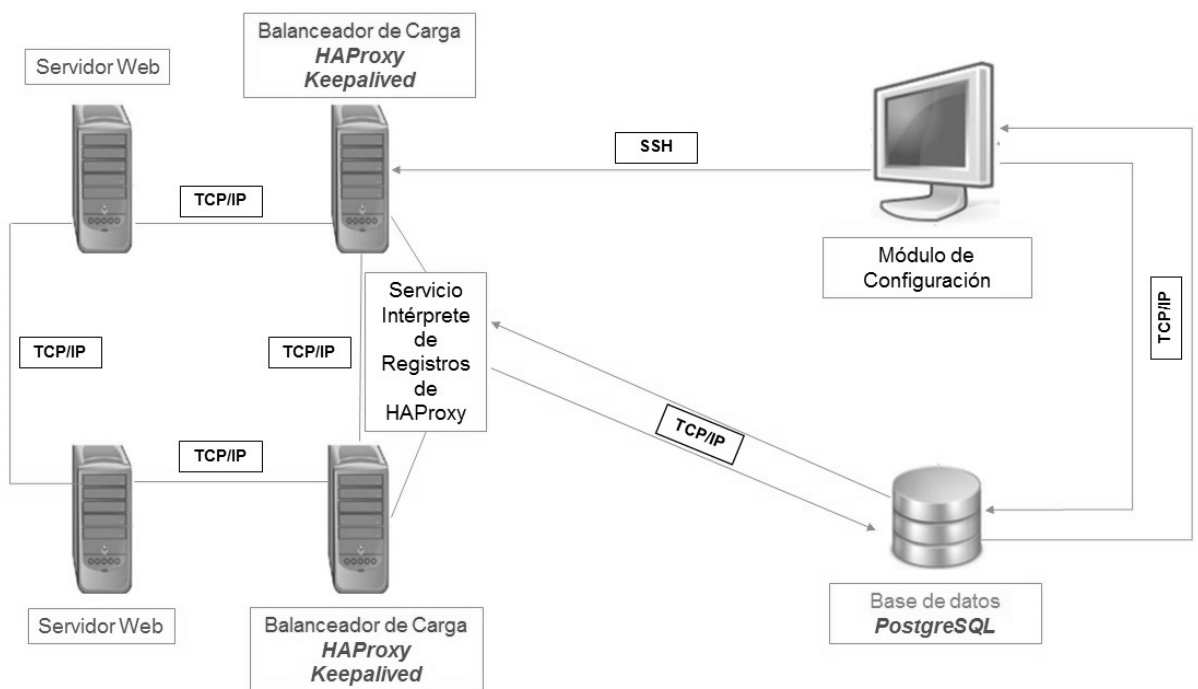


Figura #8: Propuesta de sistema

El módulo de configuración es el encargado de gestionar todo lo referente a los clústeres tanto de aplicaciones como de HAProxy. Gestiona todo lo relativo a los nodos que conforman el clúster. Se configuran los parámetros básicos de HAProxy y Keepalived, y es capaz de mostrar gráficas a partir de estadísticas recopiladas de todos los clústeres de HAProxy.

Proceso del Negocio Gestionar Nodos del Clúster

A través de este proceso el administrador puede gestionar los nodos de los diferentes clústeres. Una vez que el administrador accede a la interfaz Gestionar Nodos del Clúster podrá insertar, eliminar, modificar o listar los nodos de un clúster. En la siguiente figura se puede observar el proceso Gestionar Nodos del Clúster:

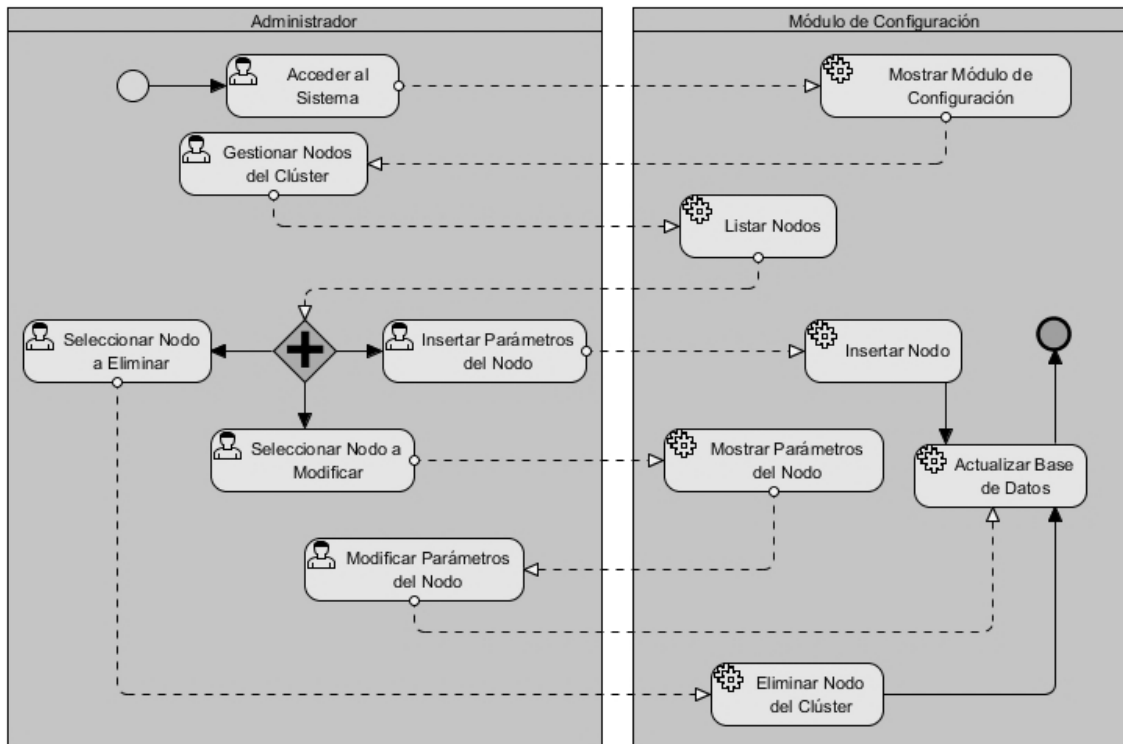


Figura #9: Proceso de Negocio Gestionar Nodos del Clúster.

A continuación, la ficha de proceso de Insertar Nodo perteneciente al proceso de negocio anteriormente expuesto, donde se recoge toda la información de dicho proceso y se genera como parte del procedimiento con el lenguaje BPMN:

Tabla #1: Ficha de Proceso Insertar Nodo

| | |
|-----------------|-----------------------------------|
| Nombre: | Insertar Nodo |
| Autores: | Hansel Tellez, Angel Danny Aguila |
| Fecha: | 5/02/2016 |

Descripción: Permite insertar un nuevo nodo de aplicación o balanceador.

Actores: Administrador

Precondiciones: El administrador debe haberse logueado en el sistema

Flujo Normal:

El actor pulsa sobre el botón Insertar Nodo

El sistema muestra un formulario para llenar los datos del nodo en cuestión.

El actor introduce los datos del nodo.

El sistema comprueba la validez de los datos y los almacena

Flujo Alternativo:

4. El sistema comprueba la validez de los datos, si los datos no son correctos, se notifica al actor de ello permitiéndole que los corrija.

Poscondiciones: El nodo queda almacenado en el sistema

Proceso del Negocio Procesar Estadísticas de HAProxy

Este servicio es el encargado de interactuar directamente con los clústeres de HAProxy para capturar, traducir, y almacenar los Stats generados por HAProxy. Una vez que el servicio se pone en ejecución comienza a capturar periódicamente los Stats generados por HAProxy de su interfaz web. Este periodo puede ser especificado por el administrador a través del Módulo de Configuración, así como el resto de los parámetros necesarios para el correcto funcionamiento del servicio, como la información para el acceso a la base de datos. Luego comienza el proceso de traducción del registro. Dicho proceso consiste en convertir la información obtenida a un formato uniforme y compatible con los modelos de base de datos para poder ser posteriormente almacenados y utilizados correctamente por el Módulo de Configuración (como la mayoría de datos son numéricos, estos son convertidos a tipo de dato integer). Por último, se guardan los datos en la base de datos. En la siguiente figura se puede observar el proceso Procesar Estadísticas de HAProxy:

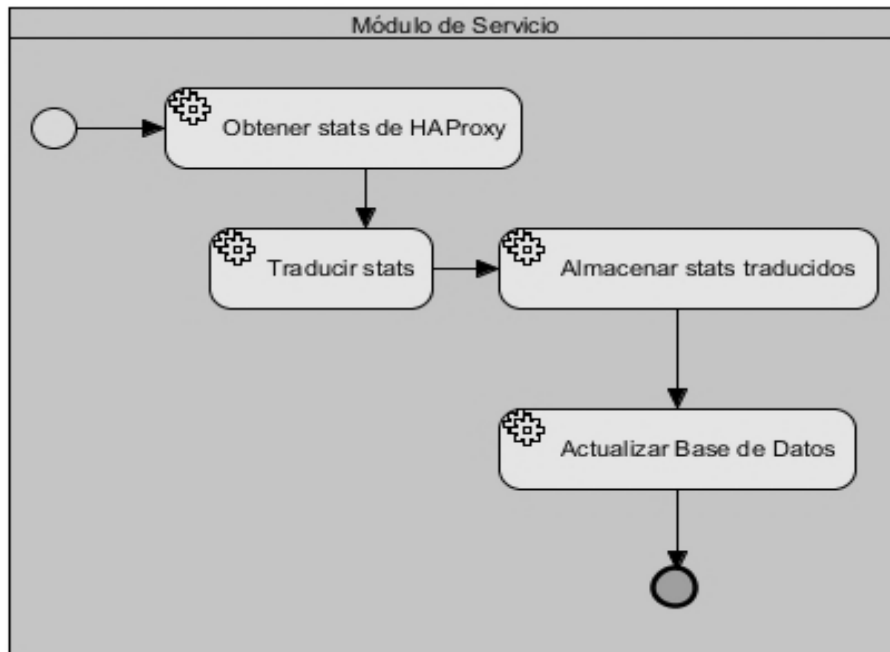


Figura #10: Proceso de Negocio Procesar Estadísticas de HAProxy

A continuación, la ficha del proceso de negocio Procesar Estadísticas de HAProxy:

Tabla #2: Ficha de Proceso Procesar Estadísticas de HAProxy

| | |
|---|-----------------------------------|
| Nombre: | Procesar Estadísticas de HAProxy |
| Autores: | Hansel Tellez, Angel Danny Aguila |
| Fecha: | 05/02/2016 |
| Descripción: Permite al sistema obtener, traducir y almacenar las estadísticas generadas por HAProxy | |
| Actores: N/A | |
| Precondiciones: Tener instalados, configurados y activos los servicios de HAProxy, Keepalived y el Servicio Intérprete de Registros | |
| Flujo Normal: Se obtienen las estadísticas generadas por HAProxy de la página de Stats. Se convierten (traducen) las estadísticas obtenidas a un formato que corresponde con los modelos de base de datos. Se almacenan las estadísticas traducidas en la base de datos | |
| Flujo Alternativo: N/A | |
| Poscondiciones: Los datos para cada nodo del clúster quedan almacenados para el instante de tiempo actual. | |

El resto de las fichas de procesos se pueden encontrar en el [Anexo 1](#).

2.3. Lista de Funcionalidades

2.3.1. Funcionalidades del módulo de configuración:

- Gestionar clústeres de aplicación.
 1. Adicionar clúster de aplicación.
 2. Modificar clúster de aplicación.
 3. Eliminar clúster de aplicación.
 4. Listar clúster de aplicación.
- Gestionar clústeres de HAProxy.
 1. Adicionar clúster de HAProxy.
 2. Modificar clúster de HAProxy.

3. Eliminar clúster de HAProxy.
 4. Listar clúster de HAProxy.
- Gestionar nodos de clúster.
 1. Adicionar nodo.
 2. Modificar nodo.
 3. Eliminar nodo.
 4. Listar nodos.
 5. Comprobar conexión.
 6. Instalar servicio.
 7. Configurar servicio intérprete de registros de HaProxy.
 - Configurar los parámetros básicos de HAProxy.
 - Configurar los parámetros básicos de Keepalived.
 - Graficar las estadísticas obtenidas del HAProxy.

2.3.2. Funcionalidades del servicio:

- Iniciar servicio.
- Detener servicio.
- Reiniciar servicio.
- Procesar fichero de estadísticas de HAProxy
 1. Obtener estadísticas.
 2. Traducir estadísticas.
 3. Almacenar estadísticas.

2.4. Lista de Reserva del Producto

En la lista de reserva se definen las cualidades o propiedades que el producto debe tener para un correcto funcionamiento.

Usabilidad

El usuario que interactuará con el sistema, necesitará una preparación previa para poder operar el mismo. Deberá poseer al menos un nivel medio en conocimientos de computación y tener nociones básicas de balanceo de carga y aseguramiento de la alta disponibilidad mediante HAProxy y Keepalived.

Software

Los ordenadores desde donde se acceda al sistema deben tener instalado un navegador web con total soporte para HTML 5.0, XHTML, CSS 3 y JavaScript (se recomiendan Mozilla Firefox 23 o superior, Google Chrome 27 o superior). Los nodos balanceadores deben tener instalado HAProxy

y Keepalived. La estación donde se ejecutará el balanceo de carga debe tener el Servicio Intérprete del Registro de HAProxy activo para la obtención, traducción y almacenamiento en base de datos de las estadísticas de HAProxy en todo momento. Este servicio podrá ser instalado por el módulo de gestión mediante conexión ssh.

Hardware

Cada computadora que ejecute algún módulo del sistema debe tener al menos 2 GB de RAM y 8 GB de disco duro, y el servidor de base de datos además un disco duro de 250 GB como mínimo.

Disponibilidad

El sistema estará disponible las 24 horas del día para que el administrador pueda acceder al mismo en el momento que desee. Es necesario que el servicio se encuentre en constante ejecución para poder almacenar las estadísticas de HAProxy brindadas por cada balanceador de carga en tiempo real. Para lograr esta disponibilidad se utilizará la herramienta de HAProxy que será la encargada de garantizar que el sistema esté disponible el tiempo requerido; a su vez se utilizará la herramienta Keepalived para lograr la disponibilidad del servicio en todo momento, lo cual garantiza que el servicio pueda almacenar las estadísticas brindadas por cada balanceador en todo momento.

2.5. Personal relacionado con el sistema

Una persona relacionada con el sistema es aquella que está vinculada al desarrollo o uso del mismo. En el sistema en cuestión existirán los siguientes roles:

-Desarrolladores: personas que implementan el sistema.

-Administrador: persona que hará uso del módulo de configuración para configurar los parámetros de HAProxy y Keepalived, así como la gestión de los clústeres y sus respectivos nodos. También podrá obtener información de HAProxy en forma gráfica en un determinado intervalo de tiempo.

2.6. Fase de Exploración

Es la fase en la que se define el alcance general del proyecto. El cliente define lo que necesita mediante la redacción de sencillas historias de usuario. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado. (26)

2.6.1. Historias de Usuario

Las Historias de Usuarios (HU) sustituyen a los documentos de especificación funcional, y a los casos de uso. Estas historias son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. (26)

Las HU serán representadas mediante tablas divididas por las siguientes secciones:

- **Número:** número de la historia de usuario incremental en el tiempo.
- **Nombre de Historia de Usuario:** el nombre de la historia de usuario.
- **Modificación de Historia de Usuario Número:** si sufrió alguna modificación anterior.
- **Usuario:** persona que va a interactuar con la HU.
- **Programadores Responsables:** involucrados en el desarrollo de la HU.
- **Iteración Asignada:** número de la iteración.
- **Prioridad en Negocio:**
 - Las historias de usuarios que son de funcionalidades imprescindibles en el desarrollo del sistema tienen prioridad alta.
 - Las historias de usuarios que son de funcionalidades que debe de tener el sistema, pero que no son necesarias para su funcionamiento, tienen prioridad media.
 - Las historias de usuarios que son de funcionalidades auxiliares y que son independientes del sistema, tienen prioridad baja.
- **Riesgo en Desarrollo:**
 - Las historias de usuarios que, en caso de tener algún error de implementación, puedan afectar la disponibilidad del sistema, tienen riesgo de desarrollo alto.
 - Las historias de usuarios que puedan presentar errores y retrasan la entrega de la versión, tienen riesgo de desarrollo medio.
 - Las historias de usuario que puedan presentar errores, pero estos son tratados con facilidad y no afectan en desarrollo del proyecto, tienen riesgo de desarrollo bajo.
- **Puntos Estimados:** tiempo estimado que se demorará el desarrollo de la HU.
- **Puntos Reales:** tiempo que se demoró en realidad el desarrollo de la HU.
- **Descripción:** breve descripción de la HU.
- **Observaciones:** señalamiento o advertencia del sistema.
- **Prototipo de Interfaz:** prototipo de interfaz si aplica.

A continuación, un ejemplo de HU perteneciente al sistema:

Tabla #3: Historia de Usuario #3 Gestionar nodos del clúster

| | |
|--|--|
| Historia de Usuario | |
| Número: 3 | Nombre: Gestionar nodos del clúster |
| Modificación de Historia de Usuario: Ninguna | |
| Usuario: Administrador | Iteración asignada: 4 |
| Programadores Responsables: Hansel Tellez, Angel Danny Aguila | |

| | |
|--|----------------------------|
| Prioridad en Negocio: Media | Puntos Estimados: 3 |
| Riesgo en Desarrollo: Alta | Puntos Reales: 3 |
| Descripción: Permite registrar los datos de un nuevo nodo a un clúster determinado, ya sea de aplicación o de HAProxy, así como modificar, eliminar y mostrar los nodos existentes. También posibilita instalar el Servicio Intérprete de Registros de HAProxy en ellos a través de conexión ssh y comprobar el estado de dicho servicio. | |

El resto de las Historias de Usuario se encuentran en el [Anexo 2](#).

2.7. Fase de planificación

Luego de identificar las HU, se realiza la planificación de la entrega. En esta fase el cliente establece la prioridad de cada HU, y los programadores realizan una estimación del esfuerzo necesario para implementar cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

2.7.1. Estimación de esfuerzo por HU

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto, lo que se conoce como Estimación de Esfuerzo por Puntos. Para ello los programadores definen una puntuación para cada HU de acuerdo a la complejidad de implementación de la misma para ellos, teniendo en cuenta una medida de tiempo por cada punto de estimación. Se considera un total de 5 días de programación real por cada punto de estimación.

Tabla #4: Estimación de esfuerzo por Historia de Usuario

| Historias de Usuario | Puntos de Estimación |
|--|-----------------------------|
| 1- Gestionar clúster de aplicación | 2 |
| 2- Gestionar clúster de HAProxy | 2 |
| 3- Gestionar nodos del clúster | 3 |
| 4- Configurar los parámetros básicos de HAProxy | 1 |
| 5- Configurar los parámetros básicos de Keepalived | 1 |
| 6- Graficar las estadísticas obtenidas del HAProxy | 2 |
| 7- Iniciar servicio | 0.2 |

| | |
|---|-----|
| 8- Detener servicio | 0.2 |
| 9- Reiniciar servicio | 0.2 |
| 10- Procesar fichero de estadísticas de HAProxy | 3 |

2.7.2. Plan de iteraciones

El plan de iteraciones consiste en seleccionar las historias de usuario que corresponden a cada iteración. Para el desarrollo de las historias de usuario se planificaron 4 iteraciones.

Iteración 1

En esta primera iteración se implementan las HU 4, 5 y 6 pertenecientes al Módulo de Configuración.

Iteración 2

En esta segunda iteración se implementa la HU 10, perteneciente al Módulo de Servicio.

Iteración 3

En esta tercera iteración se implementan las HU 1 y 2 pertenecientes al Módulo de Configuración.

Iteración 4

En esta cuarta iteración se implementan las HU 3, 7, 8 y 9, complementando así el Módulo de Configuración.

2.7.3. Plan de duración de las iteraciones

El plan de duración de las iteraciones es el encargado de relacionar las HU que van a ser implementadas con cada una de las iteraciones. También se muestra el tiempo de duración de cada iteración definida.

Tabla #5: Plan de duración de las iteraciones

| Iteración | Historias de Usuario a implementar | Duración |
|-----------|--|-----------|
| 1 | 4- Configurar los parámetros básicos de HAProxy | 4 Semanas |
| | 5- Configurar los parámetros básicos de Keepalived | |
| | 6- Graficar las estadísticas obtenidas del HAProxy | |
| 2 | 10- Procesar fichero de estadísticas de HAProxy | 3 Semanas |

| | | |
|---|------------------------------------|-------------|
| 3 | 1- Gestionar clúster de aplicación | 4 Semanas |
| | 2- Gestionar clúster de HAProxy | |
| 4 | 3- Gestionar nodos del clúster | 3.6 Semanas |
| | 7- Iniciar servicio | |
| | 8- Detener servicio | |
| | 9- Reiniciar servicio | |

2.7.4. Plan de entrega

El Plan de entrega se obtiene a través de las reuniones realizadas entre el equipo de desarrollo y los clientes, para definir el tiempo que va a demorar el desarrollo del sistema. Este Plan detalla la fecha de fin de cada una de las iteraciones definidas y los productos obtenidos en cada una de ellas.

MC: Módulo de Configuración.

SIS: Servicio Intérprete de Stats de HAProxy.

CS: complementos del Servicio Intérprete de Stats de HAProxy.

SCAB: Sistema para la Configuración de Alta Disponibilidad y Balanceo de Carga.

Tabla #6: Plan de entrega

| Sistema | Final de Iteración 1 (11 de marzo de 2016) | Final de Iteración 2 (1 de abril de 2016) | Final de Iteración 3 (29 de abril de 2016) | Final de Iteración 4 (25 de mayo de 2016) |
|-------------|---|--|---|--|
| SCAB | MC v0.1 | SIS | MC v0.2 | CS |

2.8. Conclusiones del capítulo

En el presente capítulo se realizó una descripción de la solución propuesta. Se abordaron los aspectos referentes a las fases de exploración y planificación de la metodología XP. Se describieron las características y funcionalidades del sistema a desarrollar. Con la descripción de las historias de usuario, la planificación y la obtención del plan de entregas se ha logrado delimitar el ciclo de desarrollo del sistema.

Capítulo 3: Diseño, Implementación y Pruebas

3.1. Introducción

En este capítulo se describen las fases de diseño, implementación y pruebas propias de la metodología de desarrollo XP. Se define la arquitectura del sistema, así como los patrones arquitectónicos y de diseño utilizados en el desarrollo del mismo. También son detalladas las iteraciones realizadas durante la implementación, así como las tareas de ingeniería y las clases que van a conformar el sistema. Además, son descritas las pruebas que se le van a realizar al sistema, con el objetivo de verificar su correcto funcionamiento.

3.2. Arquitectura del Sistema

La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. (27)

Los estilos arquitectónicos son la herramienta básica de un arquitecto a la hora de dar forma a la arquitectura de una aplicación. Pueden organizarse en torno al aspecto de la aplicación sobre el que se centran. Lo normal en una arquitectura es que no se base en un solo estilo arquitectónico, sino que combine varios de dichos estilos para obtener las ventajas de cada uno. (28) A continuación los estilos arquitectónicos para el desarrollo del sistema:

3.2.1. Arquitectura N-Tiers

El estilo arquitectónico N-Tiers define la separación de la funcionalidad en niveles físicos separados, o sea, que separa cada segmento del sistema en un ordenador distinto. Es un estilo para definir el despliegue de las capas de la aplicación.

Características:

- Se caracteriza por la descomposición funcional de las aplicaciones, componentes de servicio y su despliegue distribuido, que ofrece mejor escalabilidad, disponibilidad, rendimiento, manejabilidad y uso de recursos.
- Cada nivel es completamente independiente de los otros niveles excepto del inmediatamente inferior.
- Tiene al menos 3 niveles lógicos o capas separados. Cada capa implementa una funcionalidad específica y está físicamente separada en distintos servidores. (28)

Para el desarrollo del presente sistema se propone una arquitectura N-Tiers, debido a que tanto el módulo de configuración como el servicio intérprete interactúan con la base de datos del sistema, y a su vez los mismos pueden estar separados físicamente en diferentes ordenadores para lograr un

mejor uso de recursos y un mejor rendimiento. En este caso el sistema estará compuesto por 3 niveles físicos, los cuales se muestran en la siguiente figura:



Figura #11: Arquitectura del sistema

Módulo de Configuración. Arquitectura Cliente-Servidor

La arquitectura Cliente-Servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura (9).

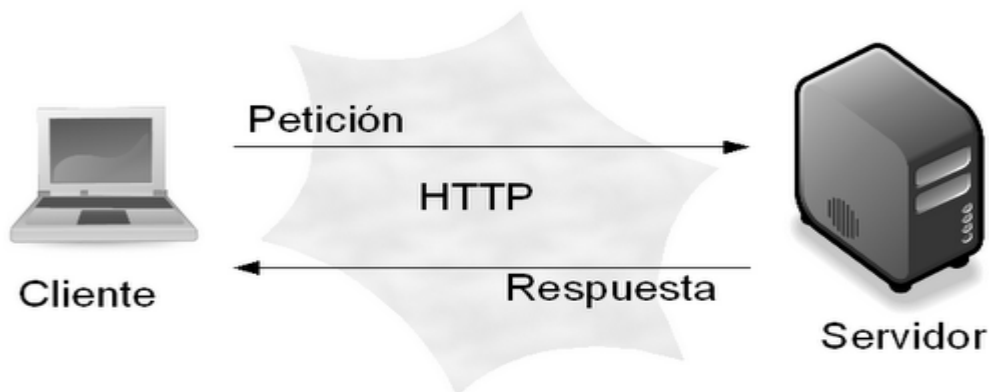


Figura #12: Arquitectura Cliente-Servidor

Se decide utilizar la arquitectura Cliente-Servidor para el desarrollo del módulo de configuración debido a que el administrador puede acceder y guardar los datos de los clústeres y sus configuraciones desde cualquier computadora que esté conectada a la red.

Servicio Intérprete de Registros de HAProxy. Arquitectura de Flujo de Datos

Esta arquitectura se aplica cuando los datos de entrada se han de transformar en datos de salida mediante una serie de componentes para el cálculo o la manipulación. Dichos componentes son denominados filtros y están conectados por tuberías que transmiten datos de cada uno de ellos al siguiente. Está diseñado para esperar la entrada de datos con cierta forma y producir su salida de forma específica (9).

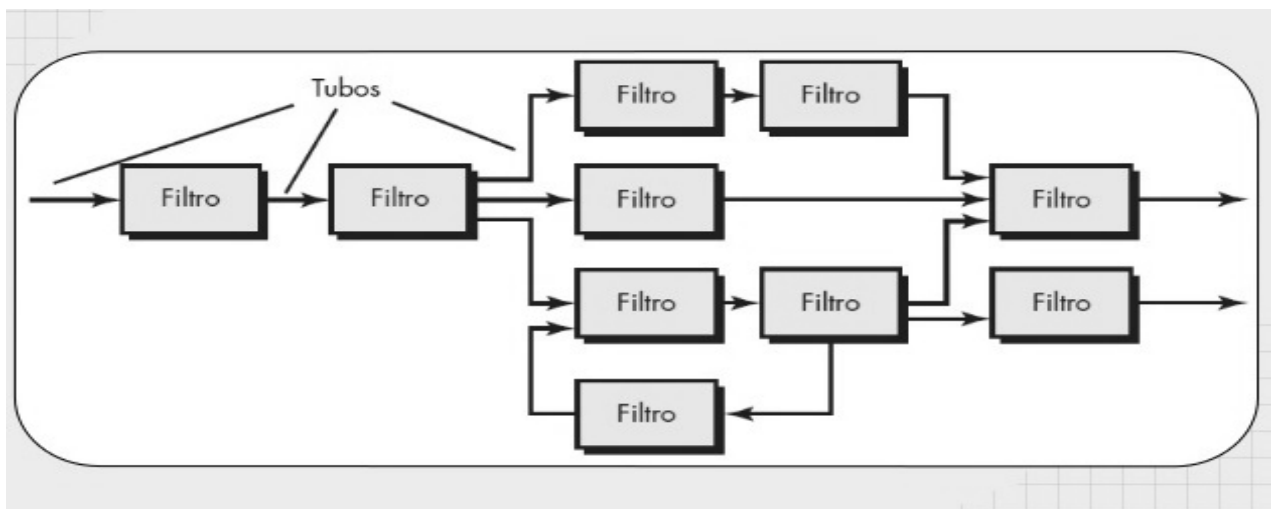


Figura #13: Arquitectura de Flujo de Datos

Tomado de www.emaze.com

Se decide utilizar la arquitectura de Flujo de Datos para el desarrollo del módulo de servicio ya que después de obtener los registros de HAProxy es necesario transformarlos para su posterior almacenamiento y uso.

3.3. Patrón Arquitectónico

Un patrón arquitectónico es la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específicos. Provee un conjunto de subsistemas predefinidos y especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. (27)

3.3.1. Patrón de arquitectura Model Template View (MTV)

Django es conocido como un framework MTV (por sus siglas en inglés Model Template View, Modelo Plantilla Vista). Está basado en el patrón MVC (por sus siglas en inglés Model View Controller, Modelo Vista Controlador), MTV es un patrón de arquitectura de software que describe

una forma de organizar el código de una aplicación separando los datos, la interfaz de usuario y la lógica de la aplicación en tres componentes:

El **modelo** incorpora la capa del dominio y persistencia, es el encargado del acceso a datos.

La **plantilla** se encarga de presentar los datos al usuario.

La **vista** es el componente que se encarga de recibir e interpretar la interacción del usuario, actuando sobre el modelo y la vista de manera adecuada para provocar cambios en la representación interna de los datos, así como en su visualización.

El propósito del MTV es aislar los cambios. Es una arquitectura preparada para los cambios, que separa los datos y la lógica de negocio de la lógica de presentación.

En el caso del framework Django esto ocurre en los Modelos, las Plantillas y las Vistas (MTV).

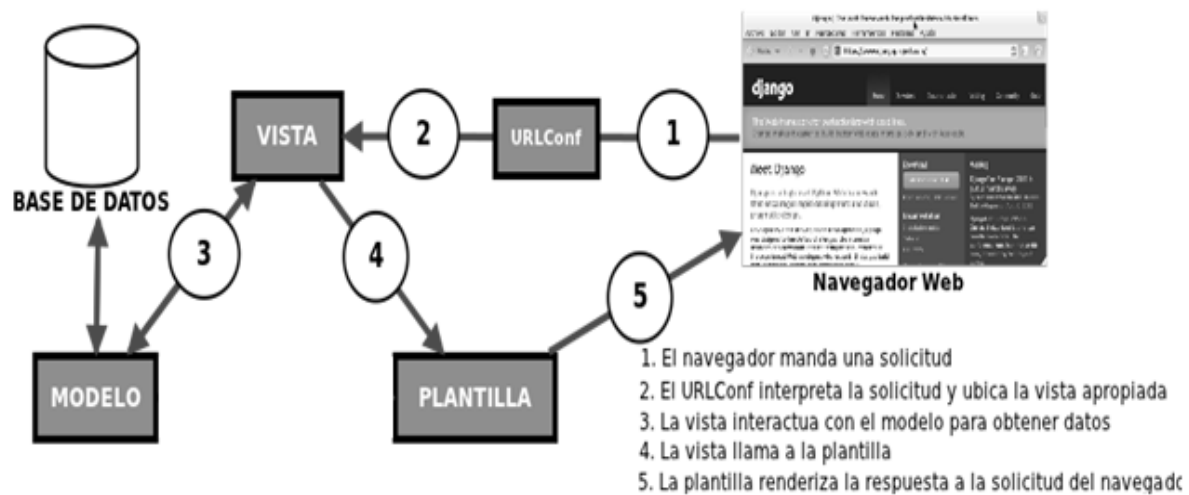


Figura #14: Esquema de funcionamiento del patrón MTV de Django

Tomado de www.maestrosdelweb.com

M significa “**Model**” (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

T significa “**Template**” (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento

V significa “**View**” (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.

A continuación, se muestran los ficheros y clases que forman parte de cada una de las capas de la arquitectura MTV:

Capa Modelo

Estas clases modelo se encuentran declaradas en el fichero models.py.

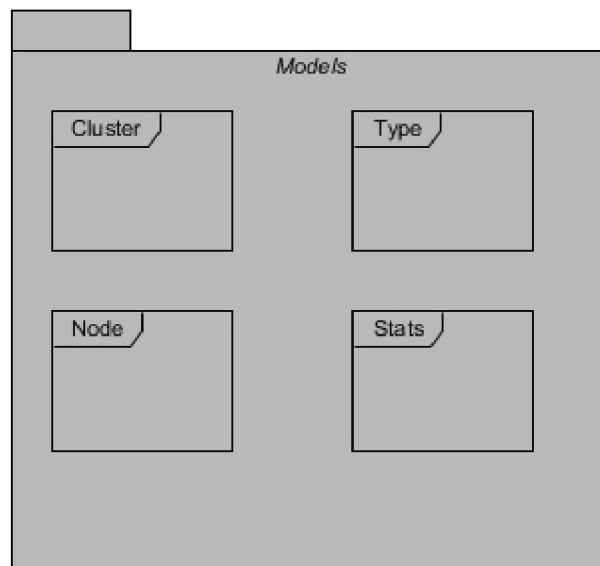


Figura #15: Clases de la capa Modelo

Capa Plantilla

La plantilla recibe los datos de la vista y luego los organiza para la presentación al navegador web. Las plantillas se encuentran ubicadas dentro de la carpeta templates.

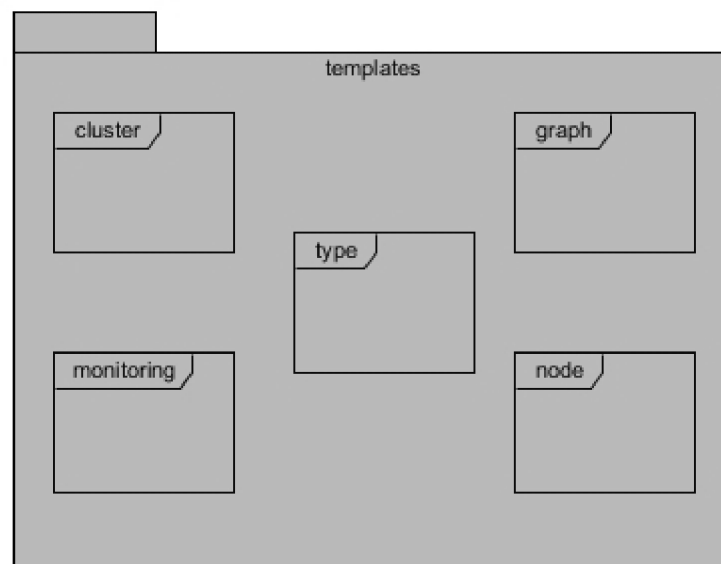


Figura #16: Clases de la capa Plantilla

Capa Vista

Las vistas en Django se muestran como funciones que determinan qué datos serán visualizados. Estas se encuentran declaradas en el fichero views.py.

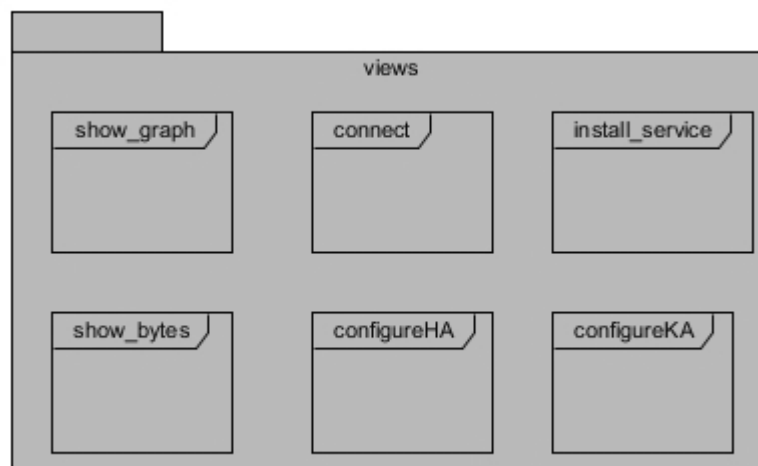


Figura #17: Clases de la capa Vista

3.4. Patrones de Diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón. La finalidad de cada patrón de diseño es proporcionar una descripción que le permita al diseñador determinar:

- 1) Si el patrón es aplicable al trabajo actual
- 2) Si el patrón se puede reutilizar
- 3) Si el patrón puede servir como guía para desarrollar un patrón similar, pero difiere en cuanto a la funcionalidad o estructura. (9)

3.4.1. Patrones para Asignar Responsabilidades (GRASP)

Los patrones GRASP (acrónimo en inglés General Responsibility Assignment Software Patterns) son patrones de diseño para la asignación de responsabilidades. Su uso es fundamental para un correcto diseño del software. Ayudan a la reutilización de diseño gráfico, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones.

A continuación, se describen los patrones GRASP utilizados en el desarrollo de este sistema.

Patrón Experto

Problema: ¿Cuál es un principio general para asignar responsabilidades a los objetos?

Solución: Asignar una responsabilidad al experto en información. (29)

Cada clase creada por el ORM de Django a partir de una entidad, es experta en manejar su información, al ser este ORM una librería externa que utiliza Django para realizar su capa de abstracción al modelo de datos.

A las clases Cluster, Type, Node, Stats y show_graph se les asignarán las responsabilidades que le corresponden a cada una, de acuerdo a la información que poseen.

Alta Cohesión

Problema: ¿Cómo mantener la complejidad manejable?

Solución: Asignar una responsabilidad de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. (29)

Bajo Acoplamiento

Problema: ¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?

Solución: Asignar una responsabilidad de manera que el acoplamiento permanezca bajo. (29)

A las clases Cluster, Type, Node y Stats se les asignarán responsabilidades con el objetivo que una clase no dependa de muchas clases.

3.5. Tarjetas Clase – Responsabilidad – Colaborador (CRC)

Las tarjetas CRC (Clase-Responsabilidad-Colaborador) son fichas, una por cada clase, en las que se escriben brevemente las responsabilidades de la clase, y una lista de objetos con los que colabora para llevar a cabo esas responsabilidades. (29)

3.5.1. Tarjetas CRC pertenecientes al Módulo de Configuración

Tabla #7: Tarjeta CRC: Clase Node

| Clase Node | |
|------------------------|--------------------|
| Responsabilidad | Colaborador |
| Insertar un nuevo nodo | Model |
| Modificar un nodo | |
| Eliminar un nodo | |
| Buscar un nodo | |

3.5.2. Tarjetas CRC pertenecientes al Servicio Intérprete de Registros de HAProxy

Tabla #8: Tarjeta CRC: Clase sis

| Clase sis | |
|--|--------------------|
| Responsabilidad | Colaborador |
| Establecer conexión con HAProxy Stats | config |
| Capturar los datos propiciados por HAProxy Stats cada un determinado intervalo de tiempo | DB helpers |
| Traducir datos capturados | soupParser |
| Insertar datos traducidos en la base de datos | |

El resto de las Tarjetas CRC se pueden encontrar en el [Anexo 3](#).

3.6. Diagrama de Clases Persistentes

El diagrama de clases persistentes muestra las clases que perduran en el tiempo, así como la relación que existe entre cada una de ellas. En la siguiente figura se observa el diagrama de clases persistentes:

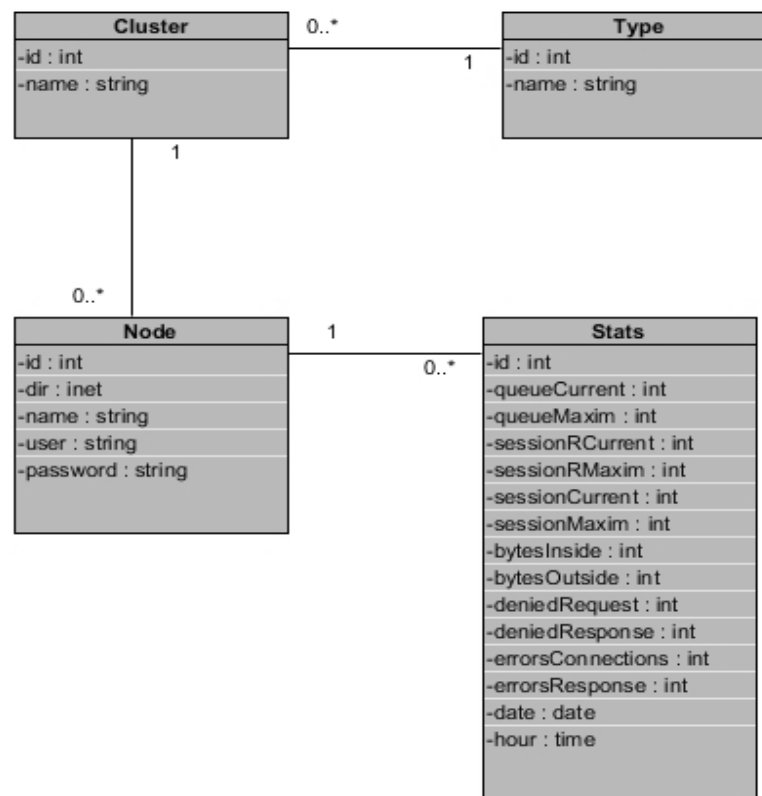


Figura #18: Diagrama de Clases Persistentes

3.7. Diagrama Entidad-Relación

Un diagrama entidad-relación es una herramienta para el modelado de datos de un sistema de información, expresando entidades relevantes para este y las relaciones entre sí. A continuación, el diagrama entidad-relación del sistema:

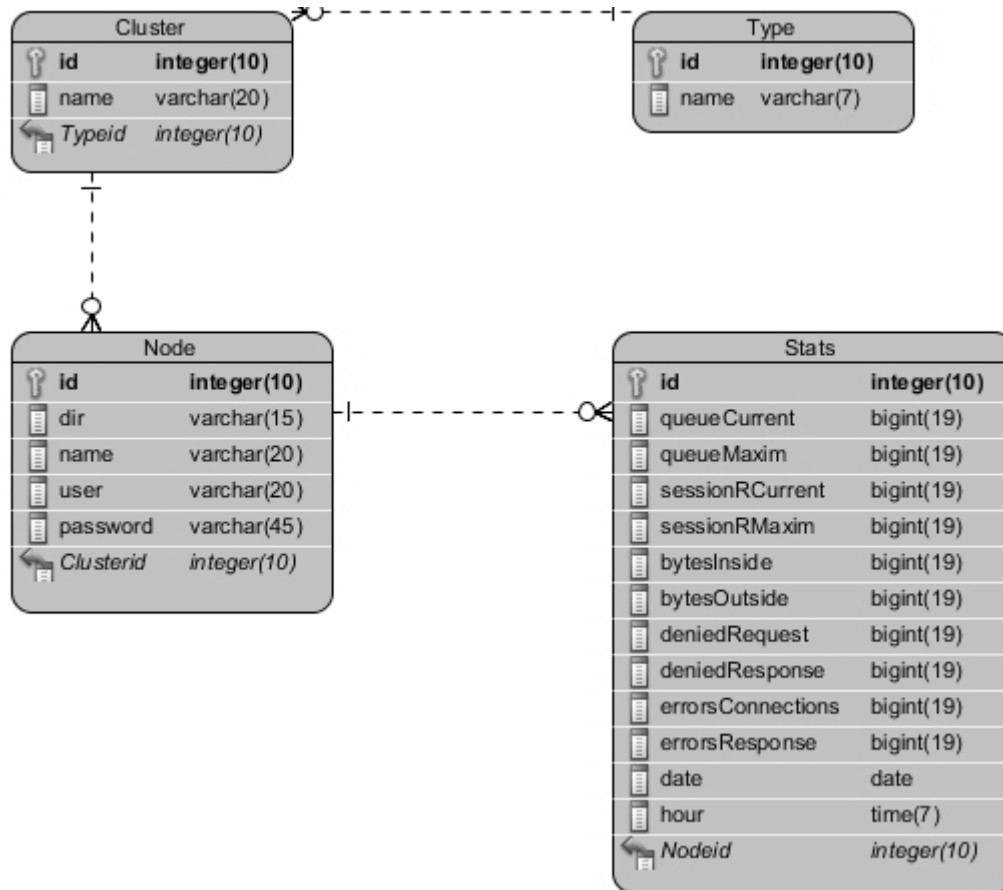


Figura #19: Diagrama Entidad-Relación

3.8. Tareas de Ingeniería

La metodología de software XP plantea que la implementación de un software se hace iterativamente. Durante cada iteración se desarrollan un conjunto de HU definidas por el cliente y descritas por el equipo de desarrollo. En esta fase de implementación las HU se dividen en tareas de ingeniería, las cuales son asignadas a los programadores para ser implementadas durante la iteración correspondiente.

Las tareas de ingeniería serán representadas mediante tablas divididas por las siguientes secciones:

- **Número de Tarea:** los números de cada tarea deben ser consecutivos.

- **Nombre de la Tarea:** nombre que identifica a la tarea.
- **Número de Historia de Usuario:** número de la historia de usuario a la que pertenece la tarea.
- **Tipo de Tarea:** las tareas pueden ser de Desarrollo, Corrección, Mejora.
- **Puntos Estimados:** tiempo estimado en días que se le asignará a cada tarea.
- **Fecha Inicio:** fecha en que comienza la tarea.
- **Fecha Fin:** fecha en que se concluye la tarea.
- **Programadores Responsables:** nombre y apellidos de los programadores que van a desarrollar la tarea.
- **Descripción:** descripción de la tarea.

A continuación, se muestran algunas de las tareas correspondientes a las HU de prioridad alta:

Tabla #9: Tarea de Ingeniería #16 Configurar los parámetros básicos de HAProxy

| | |
|---|---|
| Tarea de Ingeniería | |
| Número de Tarea: 16 | Número de Historia de Usuario: 4 |
| Nombre de la Tarea: Configurar los parámetros básicos de HAProxy | |
| Tipo de Tarea: Desarrollo | Puntos estimados: 5 |
| Fecha Inicio: 15/02/2016 | Fecha Fin: 19/02/2016 |
| Programadores Responsables: Angel Danny Aguila Jerez, Hansel Téllez Milián | |
| Descripción: Permite la configuración básica de HAProxy para un clúster determinado. | |

Tabla #10: Tarea de Ingeniería #21 Traducir estadísticas

| | |
|---|--|
| Tarea de Ingeniería | |
| Número de Tarea: 21 | Número de Historia de Usuario: 10 |
| Nombre de la Tarea: Traducir estadísticas | |
| Tipo de Tarea: Desarrollo | Puntos estimados: 5 |
| Fecha Inicio: 21/03/2016 | Fecha Fin: 25/03/2016 |
| Programadores Responsables: Angel Danny Aguila Jerez, Hansel Téllez Milián | |
| Descripción: Permite al servicio intérprete traducir las estadísticas obtenidas de HAProxy a un lenguaje apto para almacenar posteriormente. | |

El resto de las Tareas de Ingeniería se pueden encontrar en el [Anexo 4](#).

3.9. Pruebas del Sistema

La prueba del software es un elemento de un tema más amplio que suele denominarse verificación y validación. Verificación es el conjunto de actividades que aseguran que el software implemente correctamente una función específica. Validación es un conjunto diferente de actividades que aseguran que el software construido corresponde con los requisitos del cliente. (9)

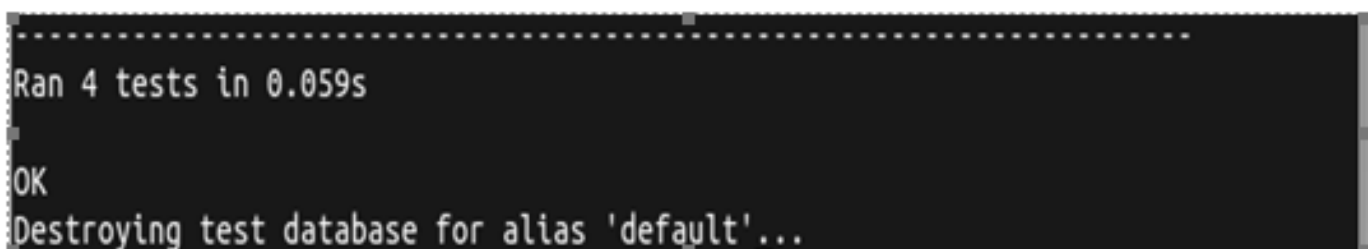
La metodología XP divide las pruebas en dos grupos: pruebas de unidad, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

3.9.1. Pruebas de unidad

La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el módulo de software. Tomando como guía la descripción del diseño al nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo.

Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente. Este tipo de prueba se puede aplicar en paralelo a varios componentes. (9)

Para la realización de las pruebas unitarias se utilizó la herramienta `pytest-django` en su versión 2.9.1. Esta es una herramienta que se utiliza para hacerle pruebas a las aplicaciones escritas en el lenguaje Python que utilizan el framework Django. Las pruebas unitarias se fueron realizando a medida que se terminaba de implementar una funcionalidad, en busca de posibles errores. Todos los errores encontrados en las pruebas realizadas fueron corregidos de forma correcta.



```
Ran 4 tests in 0.059s
OK
Destroying test database for alias 'default'...
```

Figura #20: Interfaz de salida de pytest luego de las pruebas unitarias

Las pruebas de unidad fueron aplicadas a las clases correspondientes a los modelos, para ello se realizaron búsquedas de datos a partir de parámetros identificativos de los elementos. A continuación, algunos ejemplos de las pruebas de unidad realizadas:

```

class ModelsTestCase(TestCase):
    def setUp(self):
        name1 = "MyCluster"
        ip = '192.168.0.214'
        name = 'newNode'
        user = 'root'
        password = 'root'
        Type.objects.create(id=13, name='newType')
        Cluster.objects.create(id=15, name=name1, type_id=13)
        Node.objects.create(id=12, dir=ip, name=name, user=user, password=password, cluster_id=15, type_id=13)

    def test_cluster_name(self):
        c1 = Cluster.objects.get(name="MyCluster")
        self.assertEqual(c1.name, 'MyCluster')

    def test_cluster_type(self):
        c2 = Cluster.objects.get(type_id=13)
        self.assertEqual(c2.type_id, 13)

    def test_node_dir(self):
        n = Node.objects.get(dir='192.168.0.214')
        self.assertEqual(n.dir, '192.168.0.214')

```

Figura #21: Ejemplos de las pruebas de unidad realizadas

3.9.2. Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que se realizan a partir de las historias de usuario. (30) Una historia de usuario puede tener todas las pruebas de aceptación que desee para asegurar su funcionamiento. El objetivo específico de esta prueba es garantizar que los requerimientos han sido cumplidos y que el sistema ha sido aceptable. (31)

Las pruebas de aceptación correspondientes a cada una de las funcionalidades del sistema son representadas mediante tablas con los siguientes campos:

- **Clases Válidas:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Clases Inválidas:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado Esperado:** se hará una breve descripción del resultado que se espera, tanto para entradas válidas como para entradas inválidas.
- **Resultado de la Prueba:** se hará una breve descripción del resultado que se obtiene.
- **Observaciones:** se hará la descripción de cualquier señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

A continuación, se muestra una de las pruebas de aceptación realizadas al sistema:

Tabla #11: Caso de prueba Insertar Nodo

| Clases Válidas | Clases Inválidas | Resultado Esperado | Resultado de la Prueba | Observaciones |
|---|---|--|------------------------|---------------|
| <p>El administrador accede a la interfaz de Nodos y presiona el botón Adicionar. Luego introduce el nombre, la dirección IP, el clúster al que va a pertenecer, el tipo, el usuario y la contraseña.</p> <ul style="list-style-type: none"> • Web1 • 10.8.15.11 • Webfarm • Web • Root • Root | | <p>El módulo de configuración verifica que los datos de entrada son válidos y almacena una nueva conexión Telnet en la base de datos del sistema</p> | Satisfactorio | |
| | <p>El administrador accede a la interfaz de Nodos y presiona el botón Adicionar. Luego introduce el nombre, la dirección IP, el clúster al que va a pertenecer, el tipo, el usuario y la contraseña.</p> <ul style="list-style-type: none"> • Web1 • 10.8.15.11 • Webfarm • Web • Root | <p>El módulo de configuración debe notificar que ya existe un nodo con ese IP asignado. No se puede insertar el nodo.</p> | Satisfactorio | |

| | | | | |
|--|---|---|---------------|--|
| | <ul style="list-style-type: none"> • Root | | | |
| | <p>El administrador accede a la interfaz de Nodos y presiona el botón Adicionar. Luego introduce el nombre, la dirección IP, el clúster al que va a pertenecer, el tipo, el usuario y la contraseña.</p> <ul style="list-style-type: none"> • Web1 • 10.8.15. • Webfarm • Web • Root • Root | <p>El módulo de configuración debe notificar que la dirección IP es inválida. No se puede insertar el nodo.</p> | Satisfactorio | |
| | <p>El administrador accede a la interfaz de Nodos y presiona el botón Adicionar. Luego introduce el nombre, la dirección IP, el clúster al que va a pertenecer, el tipo, el usuario y la contraseña.</p> <ul style="list-style-type: none"> • • • • • • | <p>El módulo de configuración debe notificar que debe llenar todos los campos obligatorios. No se puede insertar el nodo.</p> | Satisfactorio | |

Las restantes pruebas realizadas pueden consultarse en el [Anexo 5](#).

A través de la siguiente figura se exponen los resultados alcanzados luego de realizar las pruebas de aceptación, donde se obtuvo en una primera iteración un total de 12 no conformidades, de las

cuales 10 fueron resueltas, 2 no procedía y ninguna quedó pendiente por resolver. En una segunda iteración se obtuvo un total de 7 no conformidades, de las cuales 5 fueron resueltas, 2 no procedía y ninguna quedó pendiente por resolver. En una tercera iteración se obtuvo un total de 4 no conformidades, las cuales fueron resueltas, para arribar a una cuarta iteración donde no se detectaron no conformidades.

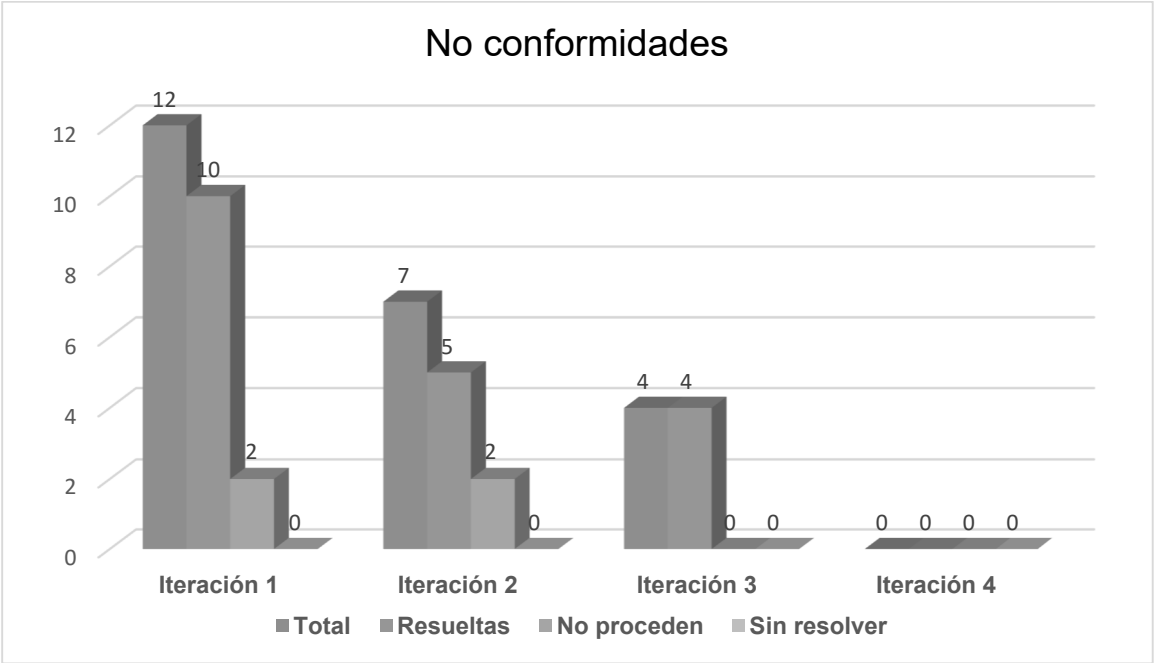


Figura #22: Resultados de las pruebas de aceptación

3.10. Conclusiones del capítulo

En este capítulo se describieron las fases de Diseño, Implementación y Pruebas correspondientes a la metodología XP. Se describió la arquitectura que va a tener el sistema, así como también los patrones de diseño y las tarjetas CRC que fueron definidas por cada módulo del sistema. Se implementaron las tareas de ingeniería correspondientes a las HU definidas en cada iteración. Por último, se realizaron las pruebas unitarias y de aceptación, las cuales arrojaron ciertas no conformidades que fueron solucionadas.

Conclusiones Generales

En el presente trabajo se investigaron las principales características y funcionamiento de los balanceadores de carga. Se analizaron los diferentes sistemas para el balanceo de carga y la alta disponibilidad, y los sistemas para la configuración y el monitoreo de dichos servicios en aplicaciones con HAProxy y Keepalived. Se aplicaron las herramientas y tecnologías de trabajo teniendo en cuenta la metodología de desarrollo de software XP. Se hizo un profundo análisis y comprensión del estilo arquitectónico y los patrones de diseño utilizados para lograr un correcto diseño e implementación del sistema. También se realizaron pruebas al sistema para comprobar su correcto funcionamiento.

Como resultado de la investigación se obtuvo:

- ✓ Un sistema que permite almacenar los datos de los clústeres de balanceo de carga y de aplicación, así como de cada uno de sus nodos componentes.
- ✓ Un sistema que permite configurar los parámetros de HAProxy y Keepalived en los balanceadores de carga.
- ✓ Un módulo capaz de obtener, traducir y almacenar los datos que HAProxy brinda sobre los clústeres de aplicación.
- ✓ Un sistema que permite graficar los datos de HAProxy acotándolos a un determinado intervalo de tiempo.

El Sistema para la Configuración de Alta Disponibilidad y Balanceo de Carga permite a los clientes mantener registros históricos del comportamiento de sus aplicaciones, posibilitando graficar dichos registros para realizar mejores análisis y comparaciones. También les brinda una herramienta que es capaz de configurar HAProxy y Keepalived de una manera más sencilla y rápida.

Por todo lo anteriormente expuesto se concluye que el objetivo trazado al inicio de la investigación se cumplió satisfactoriamente, logrando como resultado un sistema que cumple con los requerimientos y especificaciones del cliente.

Recomendaciones

Teniendo en cuenta los resultados obtenidos, se recomienda:

- Implementar una funcionalidad que permita realizar configuraciones avanzadas a las herramientas HAProxy y Keepalived, con el objetivo de hacerlas más personalizables.
- Implementar la internacionalización del sistema para aumentar el número de posibles usuarios.

Referencias

1. Soler Pérez, Vicente. *El uso de las TIC (Tecnologías de la Información y la Comunicación) como herramienta didáctica en la escuela*. Sevilla : Centro Educativo de Sevilla, España, 2008.
2. Diccionario de la Lengua Española. *Diccionario de la Lengua Española*. [En línea] Real Academia Española. <http://dle.rae.es>.
3. Palacio Vargas, David y Abril Romero, Felipe Andres. *Introducción al balanceo de carga en Aplicaciones web*. Quindío : s.n., 2012.
4. IBM. IBM Knowledge Center. [En línea]
http://www.ibm.com/support/knowledgecenter/es/SSKTXQ_8.5.0/com.ibm.help.sametime.v85.doc/config/st_adm_port_rvprxy_overview_c.html.
5. Mora Rodriguez, Orisbel. *Diseño e implementación de una solución de clúster para los . La Habana : s.n., 2011.*
6. *The Watcher Knows*. Ramm, Mark. 2005.
7. Barth, Wolfgang. *Nagios: System And Network Monitoring*. 2006.
8. Fall, Susan. *Datadog Adds Big Data Analytics*. 2015.
9. Pressman, Roger. *Ingeniería de Software. Un enfoque practico*. 2007.
10. HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer. [En línea] 2016.
<http://www.haproxy.org/>.
11. Keepalived. [En línea] 2016.
12. Gonzáles Duque, Raúl. *Python para todos*. Madrid : s.n.
13. Freddy Vega, John y Van Der Henst, Christian. *HTML 5 el presente de la web*. 2011.
14. Eguíluz Pérez, Javier. *Introduccion a CSS*. Madrid : s.n., 2007.
15. Mozilla developer. *Mozilla developer*. [En línea] [Citado el: 12 de Diciembre de 2015.]
<https://developer.mozilla.org/es/docs/Web/JavaScript>.
16. Angel Alvarez, Miguel. *Manual de JQuery*. 2012.
17. genbetadev. [En línea] [Citado el: 16 de Enero de 2016.]
<http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construiraplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0>.
18. bligoo. [En línea] [Citado el: 16 de Enero de 2016.] <http://bloginformatico.bligoo.com/sistema-informatico..>
19. Holovaty, Adrian y Kaplan Mosss, Jacob. *La guia definitiva de Django*. 2015.

20. Object Management Group. Object Management Group. *Business Process Model and Notation*. [En línea] <http://www.bpmn.org/>.
21. Aptana inc. *Aptana inc.* [En línea] [Citado el: 26 de Diciembre de 2015.] www.aptana.com.
22. Microbufer. *Microbufer*. [En línea] [Citado el: 18 de Enero de 2015.] <https://microbuffer.wordpress.com/2011/05/04/que-es-postgresql/>.
23. Servicios de Internet. *Servicios de Internet*. [En línea] <https://sites.google.com/site/servdeinternet3nm40/home/Servidor-web>.
24. Richardson, Leonard. Beautiful Soup Documentation. [En línea] 2015. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.
25. McKellar, Jessica. The Architecture of Open Source Applications (Volume 2): Twisted. [En línea] <http://www.aosabook.org/en/twisted.html>.
26. Joskowicz, José. *Reglas y Prácticas en XP*. Vigo, España : s.n., 2008.
27. Billy Reinoso, Carlos. *Introducción a la Arquitectura de Software*. Buenos Aires : s.n., 2004.
28. Llorente, Cesar de la Torre y otros. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0*. 2010. ISBN 978-84-936696-3-8.
29. Larman, Craig. *UML y Patrones*. s.l. : Prentice Hall.
30. R, Allende. *Desarrollo de Portales y Extranet con Plone*. 2006.
31. BECK, F. M. Y. K. *Planeando en Programación Extrema*. 2000.

Bibliografía Consultada

1. Soler Pérez, Vicente. *El uso de las TIC (Tecnologías de la Información y la Comunicación) como herramienta didáctica en la escuela*. Sevilla : Centro Educativo de Sevilla, España, 2008.
2. Diccionario de la Lengua Española. *Diccionario de la Lengua Española*. [En línea] Real Academia Española. <http://dle.rae.es>.
3. Palacio Vargas, David y Abril Romero, Felipe Andres. *Introducción al balanceo de carga en Aplicaciones web*. Quindío : s.n., 2012.
4. IBM. IBM Knowledge Center. [En línea]
http://www.ibm.com/support/knowledgecenter/es/SSKTXQ_8.5.0/com.ibm.help.sametime.v85.doc/config/st_adm_port_rvprxy_overview_c.html.
5. Mora Rodriguez, Orisbel. *Diseño e implementación de una solución de clúster para los .* La Habana : s.n., 2011.
6. *The Watcher Knows*. Ramm, Mark. 2005.
7. Barth, Wolfgang. *Nagios: System And Network Monitoring*. 2006.
8. Fall, Susan. *Datadog Adds Big Data Analytics*. 2015.
9. Pressman, Roger. *Ingeniería de Software. Un enfoque practico*. 2007.
10. HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer. [En línea] 2016.
<http://www.haproxy.org/>.
11. Keepalived. [En línea] 2016.
12. Gonzáles Duque, Raúl. *Python para todos*. Madrid : s.n.
13. Freddy Vega, John y Van Der Henst, Christian. *HTML 5 el presente de la web*. 2011.
14. Eguíluz Pérez, Javier. *Introduccion a CSS*. Madrid : s.n., 2007.
15. Mozilla developer. *Mozilla developer*. [En línea] [Citado el: 12 de Diciembre de 2015.]
<https://developer.mozilla.org/es/docs/Web/JavaScript>.
16. Angel Alvarez, Miguel. *Manual de JQuery*. 2012.
17. genbetadev. [En línea] [Citado el: 16 de Enero de 2016.]
<http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construiraplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0>.
18. bligoo. [En línea] [Citado el: 16 de Enero de 2016.] <http://bloginformatico.bligoo.com/sistema-informatico..>
19. Holovaty, Adrian y Kaplan Mosss, Jacob. *La guía definitiva de Django*. 2015.

20. Object Management Group. Object Management Group. *Business Process Model and Notation*. [En línea] <http://www.bpmn.org/>.
21. Aptana inc. *Aptana inc.* [En línea] [Citado el: 26 de Diciembre de 2015.] www.aptana.com.
22. Microbufer. *Microbufer*. [En línea] [Citado el: 18 de Enero de 2015.] <https://microbuffer.wordpress.com/2011/05/04/que-es-postgresql/>.
23. Servicios de Internet. *Servicios de Internet*. [En línea] <https://sites.google.com/site/servdeinternet3nm40/home/Servidor-web>.
24. Richardson, Leonard. Beautiful Soup Documentation. [En línea] 2015. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.
25. McKellar, Jessica. The Architecture of Open Source Applications (Volume 2): Twisted. [En línea] <http://www.aosabook.org/en/twisted.html>.
26. Joskowicz, José. *Reglas y Prácticas en XP*. Vigo, España : s.n., 2008.
27. Billy Reinoso, Carlos. *Introducción a la Arquitectura de Software*. Buenos Aires : s.n., 2004.
28. Llorente, Cesar de la Torre y otros. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0*. 2010. ISBN 978-84-936696-3-8.
29. Larman, Craig. *UML y Patrones*. s.l. : Prentice Hall.
30. R, Allende. *Desarrollo de Portales y Extranet con Plone*. 2006.
31. BECK, F. M. Y. K. *Planeando en Programación Extrema*. 2000.
32. Linux Virtual Server project. *Linux Virtual Server project*. [En línea] [Citado el: 12 de Diciembre de 2015.] <http://www.linuxvirtualserver.org/>.
33. Pen - Slag Office. *Pen - Slag Office*. [En línea] [Citado el: 12 de Diciembre de 2015.] <http://siag.nu/pen/>.
34. UltraMonkey. *UltraMonkey*. [En línea] [Citado el: 15 de Enero de 2016.] www.ultramonkey.org.
35. Facilcloud. *Facilcloud*. [En línea] [Citado el: 23 de Diciembre de 2015.] http://facilcloud.com/docs/es_ES/haproxy/.
36. Sampier, Roberto Hernández. *Metodología de la Investigación*. La Habana : Félix Varela, 2008.
37. ITU. La UIT publica los datos sobre las TIC de 2015. [En línea] http://www.itu.int/net/pressoffice/press_releases/2015/17-es.aspx.