BALANCEO DE CARGA DE SERVIDORES CON MOD_PROXY_BALANCER

Cristian David Agredo: Juan Esteban Alarcón; Juan Diego Caicedo; Esteban Sardi

Facultad de Ingeniería, Departamento de informática
Universidad Autónoma de Occidente
Cali, Colombia

cristian.agredo@uao.edu.co; juan_esteban.alarcon@uao.edu.co; juan_diego.caicedo@uao.edu.co; esteban.sardi@uao.edu.co

s

Abstract — In this project, the implementation of a cluster of web servers with load balancing will be carried out, this will work as frontend of the web server and each time a request is sent to the load balancer and it will be in charge of redirecting the request to one of the servers of the cluster hosted on the backed, these will have the necessary resources to resolve the requests.

The web requests are made to the balancer, which will be in charge of delegating the server in charge of processing the request, the backed servers will be running a web service. Finally, the tests were carried out implementing different load balancing algorithms to analyze the results obtained and see how each of these solves the requests made.

I. INTRODUCCIÓN

Con la tendencia actual de los entornos web, y la gran cantidad de personas que necesitan acceder a ellos de forma rápida y en cualquier momento el utilizar un balanceador de carga se volvió una práctica esencial si se desea que una página web con una alta demanda de usuarios pueda siempre mantenerse disponible para responder a todas las peticiones que se realizan sobre ella.

En esta práctica se explica la manera de crear un balanceador de carga local usando apache mod proxy balancer desde el sistema operativo de CentOS 7, como hacer las distintas pruebas de carga con Jmeter y analizarlas para poder seleccionar el algoritmo de balanceo que mejor se adapte a nuestra página.

Para proceder con la práctica es necesario tener un conocimiento básico en sistemas operativos tipo GNU/Linux, se recomienda también usar una box en Vagrant con el fin de optimizar los recursos de la máquina en la que se proceda con la práctica.

II. OBJETIVOS

A. General

Implementar un clúster de servidores web con balanceo de carga, el cual funcionará como frontend y se encargará de redirigir las peticiones a uno de los servidores alojados en el backend para balancear la carga de peticiones.

B. Específicos

- Realizar la configuración de los servidores requeridos para el desarrollo del proyecto y su balanceador
- Probar la ejecución de múltiples peticiones simultáneas al cluster de servidores web
- Realizar pruebas de carga variando los parámetros de las mismas
- Obtener los resultados y conclusiones de los diferentes algoritmos de balanceo de carga

II. MARCO TEÓRICO

A. Cluster/Balanceador de carga

Consiste en distribuir de forma eficiente el tráfico de red que reciben los sitios web por medio de los servidores backend, puesto que en la actualidad los sitios web modernos de alto tráfico se ven en la obligación de atender millones de peticiones simultáneas de todos sus usuarios ya sea para acceder a videos, texto, datos, etc. Todo esto de la forma más rápida y confiable posible, se puede apreciar como funciona de forma gráfica en la figura 1.

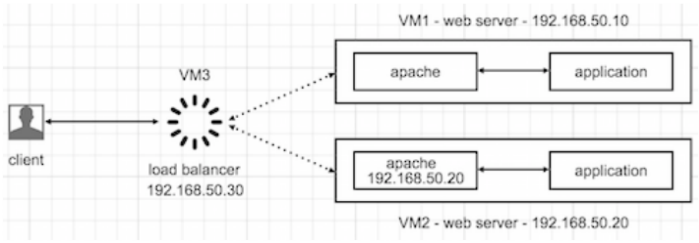


Figura 1. Diagrama de un balanceador de carga

El balanceador actúa como un policía de tránsito entre los servidores y las peticiones entrantes para enrutarlas con el servidor mejor capacitado para atender la solicitud, de esta forma se evita que algún servidor reciba más peticiones de las que puede atender y se empiece a saturar reduciendo su rendimiento y el de todo el entorno.

B. Apache mod_proxy_balancer:

Es el módulo de apache que nos proveé el servicio de balanceo de carga para los protocolos soportados como HTTP, FTP, AJP13, WebSocket. Con el fin de no tener que realizarlo desde 0 y que no sea tan complejo crear e implementar el balanceador de carga en los servidores.

C. Máquina virtual

Consiste en un entorno virtual que nos permite utilizar un sistema operativo aprovechando y particionando los recursos de memoria, disco, etc, de la máquina local.

Esto es posible gracias a la tecnología de la virtualización que utiliza el software para simular el hardware de la máquina virtual lo que permite según la capacidad de la máquina local correr una o varias máquinas virtuales al mismo tiempo gracias a un hipervisor que es el encargado de virtualizar los recursos locales y repartirlos en las máquinas virtuales que se necesiten.

D. Apache

Es el software de servidor web más utilizado en la actualidad. Desarrollado y mantenido por Apache Software Foundation, apache es un software de código abierto y gratuito, es ejecutado en el 67% de los servidores web del mundo. Su gran capacidad de personalización gracias a los diferentes entornos y módulos que tiene le permite satisfacer las necesidades de muchas compañías como por ejemplo WordPress.

.

E. Servidor Web

El término servidor web puede referirse a software o hardware, o ambos trabajan conjuntamente:

Por el lado del hardware, un servidor web es una computadora que almacena un software de servidor web y unos archivos que componen un sitio web (por ejemplo, documentos HTML, imágenes, hojas de estilo CSS y archivos JavaScript). Un servidor web se conecta a internet y soporta intercambio físico de datos con otros dispositivos conectados a la web.

Por el lado del software, un servidor web incluye varias partes que controlan cómo los usuarios de la web acceden a los archivos alojados. Como mínimo, este es un servidor HTTP. Un servidor HTTP es un software que entiende URLs (direcciones web) y HTTP (el protocolo que el navegador usa para ver páginas web). Un servidor HTTP puede ser accedido a través de nombres de dominio de los sitios web que lo almacena, y este entrega el contenido a estos sitios web alojados hacia el dispositivo del usuario final.

F. Protocolo HTTP

Es un protocolo para extraer recursos en formato de documentos HTML. Es la base de cualquier intercambio de datos en la web y es un protocolo cliente-servidor, lo que significa que las solicitudes son iniciadas por el destinatario, que usualmente es un navegador web. Para completar la petición, un documento completo es reconstruido desde diferentes subdocumentos extraídos, tales como texto, descripción de capas, imágenes, videos, scripts y demás.

III. MATERIALES Y MÉTODOS

Para el desarrollo del proyecto se requirió de un PC (Equipo de cómputo personal), un software de virtualización (VirtualBox), Máquinas virtuales con el sistema operativo CentOS 7.9 y una herramienta para generar entornos de desarrollo (Vagrant).

A. PC (Equipo de cómputo personal)

El computador es una herramienta que nos facilita mucho las tareas que realizamos en el día a día. Para el desarrollo del proyecto este se ha utilizado para alojar los diferentes servidores configurados, el sistema de virtualización en donde se han instalado los sistemas operativos y hacer uso de la herramienta de vagrant.

Figura 3 . Equipo personal.

Las características del equipo han variado según los integrantes del equipo. Sin embargo, las características recomendadas para realizar el desarrollo del proyecto son 8GB de memoria ram, ya que se requieren diferentes servidores en funcionamiento. Se ha probado con 4GB pero lamentablemente no dio los resultados esperados. Almacenamiento en disco duro de 2GB para las configuraciones realizadas y un procesador compatible con la arquitectura Intel X86 de Intel X86 64.

B. Software de virtualización (VirtualBox)

El software de virtualización de VirtualBox nos ayudará a montar máquinas virtuales con instalaciones de sistemas operativos, en este caso cuatro máquinas virtuales (servidores) con el sistema operativo de CentOS 7.9. Nos permite probar aplicaciones o servicios en otro sistema operativo sin la necesidad de realizar particiones e instalarlos junto a nuestro sistema operativo principal.

Figura 4.Logo de VirtualBox.

Instalar un sistema operativo en una máquina virtual puede ser muy útil a la hora de realizar configuraciones más profundas en el sistema

operativo o instalar alguna aplicación peligrosa que puede hacer que tu máquina principal falle.

C. *Sistema operativo CentOS 7.9*

Este sistema operativo es una distribución de linux apoyado en RHEL (Red Hat Enterprise Linux) su uso principal está destinado para el sector empresarial y organizaciones de gran tamaño.

Figura 5. Logo de CentOS.

La razones por las que se ha escogido este sistema operativo es porque los requerimientos de hardware para su correcto funcionamiento son bastante accesibles, además de que está siendo implementado en un sistema de virtualización, este funciona asignándole 512 MB de memoria ram. Sin embargo para una mayor fluidez en el desarrollo del proyecto se le asignó 1024MB de memoria ram. debe tener 20GB de espacio en disco duro o un poco menos si se usa desde línea de comandos. Toda la práctica se llevó a cabo sin interfaz gráfica del sistema operativo, todo a través de línea de comandos desde el Vagrant

D. *Vagrant*

Vagrant es una herramienta que nos ayuda a crear y manejar máquinas virtuales con un mismo entorno de trabajo, este como tal no tiene la capacidad para correr una máquina virtual, se encarga de las características con las que se debe crear esta máquina y los complementos a instalar.

Figura 6. Logo de Vagrant.

Sabemos que puede ser muy tedioso cuando debemos desarrollar un proyecto en equipo desde diferentes máquinas y la configuración que realizamos no es la misma de todos los integrantes o simplemente por otras configuraciones ajenas no funciona de la manera en como se esperaba. Desde el desarrollo del proyecto se utilizaron funciones de vagrant como el "Vagrant Cloud" en donde se ha montado la configuración final de las máquinas para que el colectivo no presente los problemas anteriormente mencionados.
La gran ventaja de esto es que posee un archivo de configuración llamado Vagrantfile donde se centraliza toda la configuración que queremos asignarle a la máquina virtual.

IV. METODOLOGÍA

Este proyecto consta de implementar tres tipos de servidores, uno de ellos tiene la función de ser el balanceador (Frontend) de carga, el cual es el que delega cual servidor resuelve las peticiones realizadas por el cliente, y otros dos, encargados de correr un servicio web en el Backend. Sin embargo, existen diferentes algoritmos de balanceo de carga para resolver las peticiones los cuales implementamos uno de los más utilizados que es byrequest (equilibrado por el número de peticiones).

En el primer servidor, el cual se ha llamado como loadbalancer, está alojado el cluster de balanceo de carga asignado con la ip 192.168.50.30, este es el encargado de redirigir las peticiones a los servidores del backed con el servicio web. Este balanceo está realizado mediante el algoritmo byrequest (equilibrado por el número de peticiones) y mejor llamado como Round Robin

Para los otros dos servidores, estos son los encargados de implementar un servicio web en el backend con el lenguaje de programación html. Estos fueron llamados como servidor1 y servidor2, con las ip's 192.168.50.10 y 192.168.50.20 respectivamente.

Los integrantes del colectivo se han delegado el trabajo de manera equitativa realizando cada uno la configuración de los servidores para realizar pruebas y analizar resultados y errores encontrados durante el proceso.

V. INSTALACION Y CONFIGURACION

Configuración del loadbalancer. En la configuración explicamos las variaciones respecto al segundo balanceador, ya que solo se requiere cambiar el algoritmo con el que se resuelven las peticiones

```
Vagrant.configure("2") do |config|
if Vagrant.has_plugin?("vagrant-vbguest")
    config.vbguest.auto_update = false
end
config.vm.define :servidor1 do |servidor1|
servidor1.vm.box = "bento/centos-7.9"
servidor1.vm.network :private_network, ip: "192.168.50.10"
servidor1.vm.hostname = "servidor1"
end
config.vm.define :servidor2 do |servidor2|
servidor2.vm.box = "bento/centos-7.9"
servidor2.vm.network :private_network, ip: "192.168.50.20"
servidor2.vm.hostname = "servidor2"
end
config.vm.define :loadbalancer do |loadbalancer|
loadbalancer.vm.box = "bento/centos-7.9"
loadbalancer.vm.network :private_network, ip: "192.168.50.30"
loadbalancer.vm.hostname = "loadbalancer"
end

end
```

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

```
<VirtualHost *:80>
<Proxy balancer://clusterServicios>
BalancerMember http://192.168.50.10
BalancerMember http://192.168.50.20
ProxySet lbmethod=bytraffic
</Proxy>
ProxyPreserveHost On
ProxyPass "/" "balancer://clusterServicios/"
ProxyPassReverse "/" "balancer://clusterServicios/"
</VirtualHost>
```

```
sudo -i
yum install vim httpd
```

```
service httpd restart
```

Instalación y configuración de las máquinas 1 y 2. La configuración para estos dos servidores es igual, pues son los que alojan el servicio web. Sin embargo, estos servicios web son diferentes pero se pueden configurar a gusto de cada desarrollador.

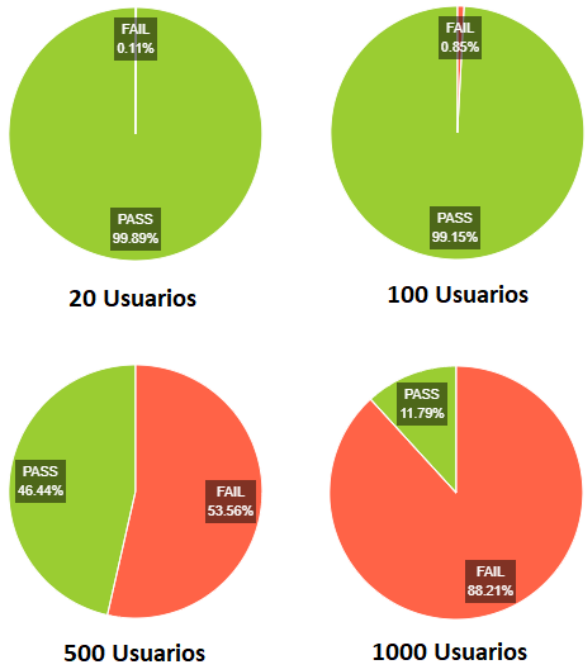
```
//Ingresamos como usuario root
#sudo -i
//Instalamos el editor de texto Vim
#yum install vim -y
//Instalamos el servicio http
#yum install httpd -y
//Nos dirigimos a la siguiente ruta
#cd /var/www/html/
//Creamos y editamos el index del servicio web
#vim index.html
//Colocamos el código dentro del archivo
//Guardamos el archivo ctrl + c y :wq
//Reiniciamos el servicio http
#systemctl httpd restart
```

VI. ANÁLISIS Y RESULTADOS

Luego de realizar la instalación de los diferentes servidores, mediante el software de JMeter se han analizado los resultados obtenidos a partir de los algoritmos de balanceo de Round Robin .Para las pruebas se hicieron pruebas con 20, 100, 500 y 1000 usuarios, durante cada intervalo de 5 segundos se ingresaban un número de usuarios simultáneamente hasta completar el total de usuarios de la prueba.

A continuación se presentarán los resultados obtenidos de las pruebas realizadas:

Gráfica 1. Resumen de peticiones con algoritmo Round Robin.



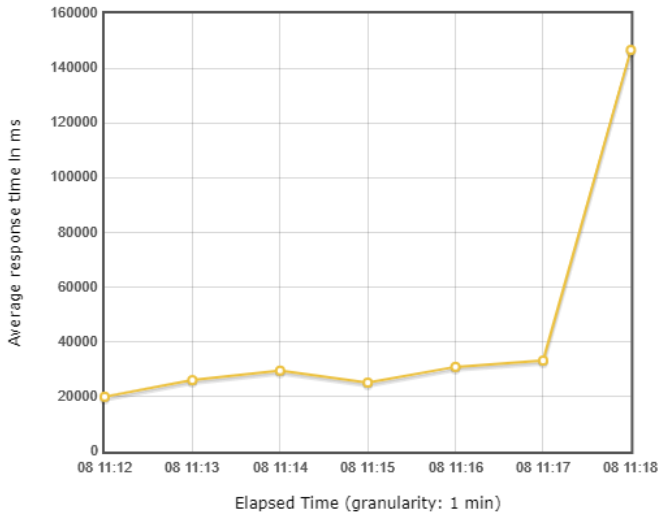
En las gráficas 1 se puede observar el resultado de las pruebas ejecutadas para 20, 100, 500 y 1000 usuarios en donde se puede mostrar el porcentaje de peticiones exitosas versus las que fallan. Con los resultados obtenidos podemos observar que el algoritmo round robin empieza a tener una tasa de más del 50% de fallos después de los 500 usuarios, ya para los 1000 usuarios la tasa de fallos es mayor al 88% lo cual sería inconcebible para la ejecución de un servidor web que requiera gran cantidad de flujo de usuarios.

Tabla 1. Porcentajes de tipos de errores generados usando el algoritmo Round Robin para 1000 usuarios.

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to 192.168.50.30:80 [/192.168.50.30] failed: Connection timed out: connect	5135	77.50%	68.36%
Non HTTP response code: java.net.SocketTimeoutException/Non HTTP response message: Read timed out	1420	21.43%	18.90%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	42	0.63%	0.56%
502/Proxy Error	29	0.44%	0.39%

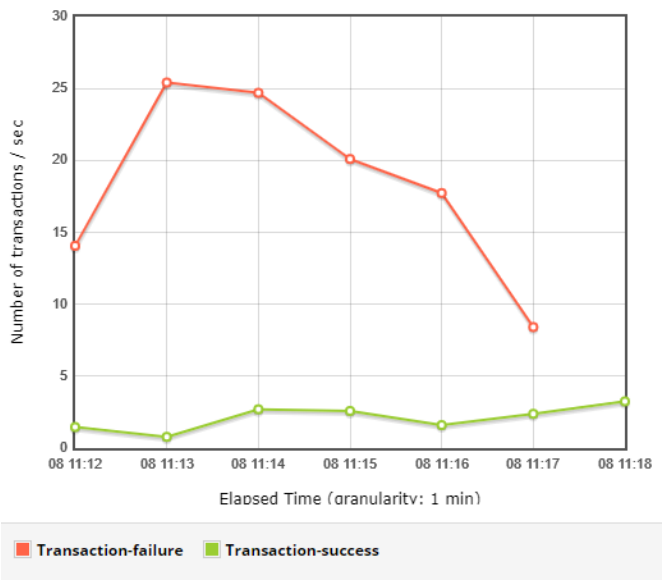
Como se observa en la tabla 1, los errores comunes arrojados en el algoritmo son HttpHostConnectException, SocketTimeoutException y SocketException, de los cuales el más común es el HttpHostConnectException que se genera debido a que un usuario trata de acceder al servidor a través de TCP pero el servidor no dispone en ese momento de ningún listener para aceptar la conexiones solicitada. Podemos observar además que el agoritomo Round Robin genera un error adicional que es 502/Proxy Error.

Gráfica 2. Promedio de tiempos de respuesta (en milisegundos) en el periodo de duración de la prueba usando el algoritmo Round Robin para 1000 usuarios.



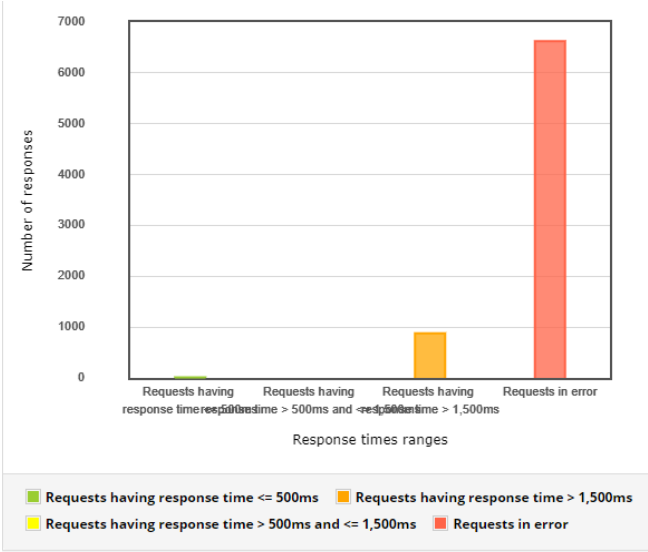
Podemos observar en la gráfica 2 los tiempos de respuesta en promedio que tarda en responder las solicitudes del balanceador del algoritmo de Round Robin en cada prueba, los resultados se muestran en cada minuto exactamente, y se puede evidenciar que al pasar de las solicitudes el tiempo de respuesta va aumentando.

Gráfica 3. Total de transacciones por segundo en el periodo de duración de la prueba usando el algoritmo Round Robin para 1000 usuarios.



La gráfica 3 muestra el total de transacciones por cada segundo que son exitosas versus las que fallan en el tiempo de duración de las pruebas del balanceador.Esta gráfica nos muestra que maneja un número mucho mayor de transacciones fallidas,sin embargo aquí no se habla de porcentajes sino de número de transacciones a través del tiempo de duración, lo que nos permite evidenciar el comportamiento de los fallos con respecto a los éxitos en cada periodo de tiempo (en este caso en cada minuto). Esto nos permite observar que desde un comienzo se tienen un mayor de porcentajes de fallas que de éxitos, siendo esta diferencia bastante considerable (aprox 14 fallas y 2 exitos), y esta cantidad de fallas sube considerablemente en el siguiente minuto, sin embargo a partir de este los fallos empiezan a disminuir de manera constante hasta no tener ninguno a en el último minuto.

Gráfica 4. Resumen del total de respuestas de las peticiones clasificadas en tiempos de respuesta (en milisegundos) usando el algoritmo Round Robin para 1000 usuarios.



Las gráfica 4 , muestra cada una una gráfica en barras en las cuales se puede evidenciar el número total de respuestas de las peticiones realizadas por los 1000 usuarios clasificadas en intervalos de tiempos de respuesta en milisegundos y adicional una barra para las peticiones que terminaron en error.

VII. CONCLUSIONES

- Teniendo en cuenta la implementación y configuración del proyecto podemos apreciar el algoritmo Round Robin que se puede utilizar para redireccionar el tráfico de las peticiones realizadas al servidor,con este implementado podemos poner a prueba su efectividad con la herramienta JMeter.

- Sería una buena práctica tener otro servidor de balanceador, pero con otro algoritmo como lo es Weighted east-connection para así, implementar las mismas pruebas y ver qué algoritmo se adapta mejor a la solución planteada.

- Hablando de la configuración del servidor, este se ha realizado haciendo uso de herramientas vistas anteriormente durante el desarrollo de las clases como los paquetes httpd, que en este vienen incluidos los módulos con los que trabajamos (mod_proxy_balancer), byrequest .Además de esto para mantener la configuración igual en las máquinas de cada integrante del grupo se utilizó Vagrant Cloud, en donde se montaron cada una de las máquinas y descargarlas y configurarlas mediante el archivo de configuración Vagrantfile, una herramienta muy práctica y de gran valor a la hora de trabajar en equipo.

- Con las pruebas realizadas se puede evidenciar que el balanceador falla en gran medida en las 500 solicitudes ,entonces se concluye que este algoritmo no es muy efectivo.

REFERENCIAS

[1] NGINX, Inc. (2021, 26 enero). What Is Load Balancing? How Load Balancers Work. NGINX. Recuperado 17 de mayo de 2022, de https://www.nginx.com/resources/glossary/load-balancing/

[2] The Apache Software Foundation. (s. f.). mod_proxy_balancer - Apache HTTP Server Version 2.4. Apache. Recuperado 17 de mayo de 2022,, de https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html

[3] Citrix, Inc. (s. f.). What is a Virtual Machine and How Does it Work? - Citrix Mexico. Citrix.com. Recuperado 17 de mayo de 2022,, de https://www.citrix.com/es-mx/solutions/vdi-and-daas/what-is-a-virtual-machine.html

[4] About the Editorial Staff. (s. f.). What is Apache? - What is a Web Server? WPBeginner. Recuperado 17 de mayo de 2022,, de https://www.wpbeginner.com/glossary/apache/

[5] GCFGlobal. (s. f.). Informática Básica: ¿Qué es un computador? GCFGlobal.org. Recuperado 17 de mayo de 2022,, de https://edu.gcfglobal.org/es/informatica-basica/que-es-un-computador/1/

[6] Fernández, Y. (2020, 1 junio). VirtualBox: qué es y cómo usarlo para crear una máquina virtual con Windows u otro sistema operativo. Xataka. Recuperado 17 de mayo de 2022,, de https://www.xataka.com/basics/virtualbox-que-como-usarlo-para-crear-maquina-virtual-windows-u-otro-sistema-operativo

[7] Ionos. (2020, 30 enero). ¿Qué es CentOS? Versiones CentOS y requisitos del sistema. IONOS Digitalguide. Recuperado 17 de mayo de 2022, de https://www.ionos.es/digitalguide/servidores/know-how/que-es-centos-versiones-y-requisitos-del-sistema/

[8] An overview of HTTP - HTTP | MDN. (s. f.). MDN Web Docs. Recuperado 17 de mayo de 2022,, de https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[9] What is a web server? - Learn web development | MDN. (s. f.). MDN Web Docs. Recuperado 17 de mayo de 2022,, de https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server