UD 1 – Tarea 2

Ejercicios

Elena de Antón Barrio – DAW2

Índice

Introducción

A continuación, se expondrán diferentes ejercicios con el objetivo de adquirir contenidos y conceptos básicos en relación con el desarrollo de aplicaciones-web.

1. Especificaciones técnicas de tu ordenador personal

Sistema Operativo: Windows 10

Procesador: Intel® Core™ i7-4510u cpu @2.00GHz

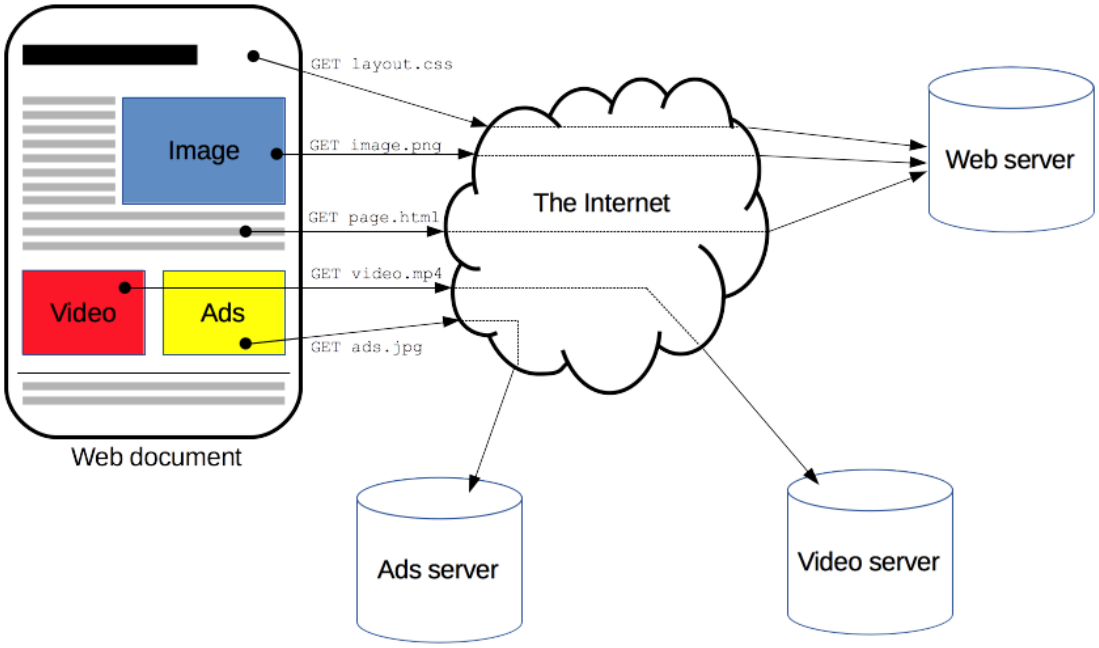
Memoria RAM: 16GB

Tarjeta gráfica: NVIDIA GEFORCE

Tipo de sistema: Sistema operativo de 64bits, procesador x64

2. Estudio detallado del protocolo HTTP.

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc...



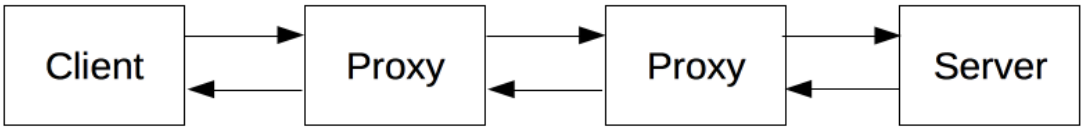
Protocolo HTTP

Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas.

Arquitectura de los sistemas basados en HTTP

HTTP es un protocolo basado en el principio de cliente-servidor: las peticiones son enviadas por una entidad: el agente del usuario (o un proxy a petición de uno). La mayoría de las veces el agente del usuario (cliente) es un navegador Web, pero podría ser cualquier otro programa, como por ejemplo un programa-robot, que explore la Web, para adquirir datos de su estructura y contenido para uso de un buscador de Internet.

Cada petición individual se envía a un servidor, el cuál la gestiona y responde. Entre cada petición y respuesta, hay varios intermediarios, normalmente denominados proxies, los cuales realizan distintas funciones, como: gateways o caches.



Relación entre el Cliente y el Servidor

En realidad, hay más elementos intermedios, entre un navegador y el servidor que gestiona su petición: hay otros tipos de dispositivos: como routers, módems ... Es gracias a la arquitectura en capas de la Web, que estos intermediarios, son transparentes al navegador y al servidor, ya que HTTP se apoya en los protocolos de red y transporte. HTTP es un protocolo de aplicación, y por tanto se apoya sobre los anteriores. Aunque para diagnosticar problemas en redes de comunicación, las capas inferiores son irrelevantes para la definición del protocolo HTTP .

Cliente: el agente del usuario

El agente del usuario, es cualquier herramienta que actué en representación del usuario. Esta función es realizada en la mayor parte de los casos por un navegador Web. Hay excepciones, como el caso de programas específicamente usados por desarrolladores para desarrollar y depurar sus aplicaciones.

El navegador es siempre el que inicia una comunicación (petición), y el servidor nunca la comienza (hay algunos mecanismos que permiten esto, pero no son muy habituales).

Para poder mostrar una página Web, el navegador envía una petición de documento HTML al servidor. Entonces procesa este documento, y envía más peticiones para solicitar scripts, hojas de estilo (CSS), y otros datos que necesite (normalmente vídeos y/o imágenes). El navegador, une todos estos documentos y datos, y compone el resultado final: la página Web. Los scripts, los ejecuta también el navegador, y también pueden generar más peticiones de datos en el tiempo, y el navegador, gestionará y actualizará la página Web en consecuencia.

Una página Web, es un documento de hipertexto (HTTP), luego habrá partes del texto en la página que puedan ser enlaces (links) que pueden ser activados

(normalmente al hacer click sobre ellos) para hacer una petición de una nueva página Web, permitiendo así dirigir su agente de usuario y navegar por la Web. El navegador, traduce esas direcciones en peticiones de HTTP, e interpretara y procesará las respuestas HTTP, para presentar al usuario la página Web que desea.

El servidor Web

Al otro lado del canal de comunicación, está el servidor, el cual "sirve" los datos que ha pedido el cliente. Un servidor conceptualmente es una unica entidad, aunque puede estar formado por varios elementos, que se reparten la carga de peticiones, (load balancing), u otros programas, que gestionan otros computadores (como cache, bases de datos, servidores de correo electrónico, ...), y que generan parte o todo el documento que ha sido pedido.

Un servidor no tiene que ser necesariamente un único equipo físico, aunque si que varios servidores pueden estar funcionando en un único computador. En el estándar HTTP/1.1 y Host , pueden incluso compartir la misma dirección de IP.

Proxies

Entre el cliente y el servidor, además existen distintos dispositivos que gestionan los mensajes HTTP. Dada la arquitectura en capas de la Web, la mayoria de estos dispositivos solamente gestionan estos mensajes en los niveles de protocolo inferiores: capa de transporte, capa de red o capa física, siendo así transparentes para la capa de comunicaciones de aplicación del HTTP, además esto aumenta el rendimiento de la comunicación. Aquellos dispositivos, que sí operan procesando la capa de aplicación son conocidos como proxies. Estos pueden ser transparentes, o no (modificando las peticiones que pasan por ellos), y realizan varias funciones:

- Caching (la caché puede ser pública o privada, como la caché de un navegador)
- Filtrado (como un anti-virus, control parental, ...)
- Balanceo de carga de peticiones (para permitir a varios servidores responder a la carga total de peticiones que reciben)

- Autentificación (para el control al acceso de recursos y datos)
- Registro de eventos (para tener un histórico de los eventos que se producen)

Mensajes HTTP

n las versiones del protocolo HTTP/1.1 y anteriores los mensajes eran de formato texto y eran totalmente comprensibles directamente por una persona. En HTTP/2, los mensajes estan estructurados en un nuevo formato binario y las tramas permiten la compresión de las cabeceras y su multiplexación. Así pues, incluso si solamente parte del mensaje original en HTTP se envía en este formato, la sematica de cada mensaje es la misma y el cliente puede formar el mensaje original en HTTP/1.1. Luego, es posible interpretar los mensajes HTTP/2 en el formato de HTTP/1.1.

Existen dos tipos de mensajes HTTP: peticiones y respuestas, cada uno sigue su propio formato.

Peticiones

Una petición de HTTP, está formado por los siguientes campos:

Un método HTTP, normalmente pueden ser un verbo, como: GET, POST o un nombre como: OPTIONS o HEAD, que defina la operación que el cliente quiera realizar. El objetivo de un cliente, suele ser una petición de recursos, usando GET, o presentar un valor de un formulario HTML, usando POST, aunque en otras ocasiones puede hacer otros tipos de peticiones.

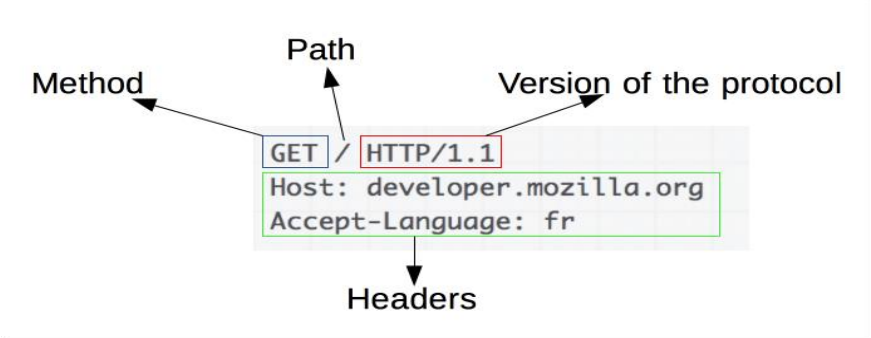
La dirección del recurso pedido; la URL del recurso, sin los elementos obvios por el contexto, como pueden ser: sin el protocolo (http://), el dominio (aquí developer.mozilla.org), o el puerto TCP (aquí el 80).

La versión del protocolo HTTP.

Cabeceras HTTP opcionales, que pueden aportar información adicional a los servidores.

O un cuerpo de mensaje, en algún método, como puede ser POST, en el cual envía la información para el servidor.

Ejemplo:



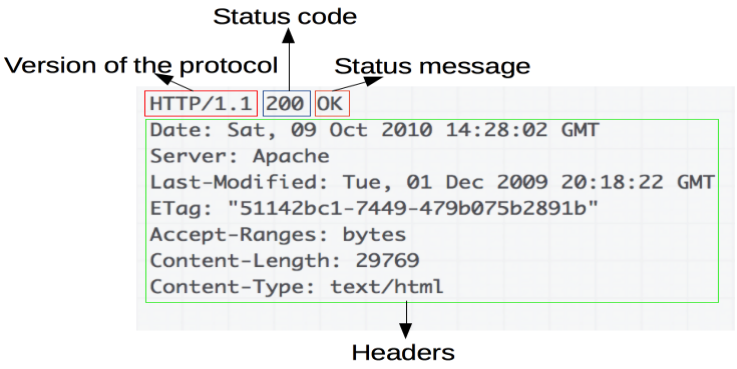
Ejemplo Petición

Respuestas

Las respuestas están formadas por los siguientes campos:

- La versión del protocolo HTTP que están usando.
- Un código de estado, indicando si la petición ha sido exitosa, o no, y debido a que.
- Un mensaje de estado, una breve descripción del código de estado.
- Cabeceras HTTP, como las de las peticiones.
- Opcionalmente, el recurso que se ha pedido.

Ejemplo:



Ejemplo Respuesta

3. Se pide buscar en Internet diferentes aplicaciones web y clasificarlas según su tipo, como mínimo dos páginas de cada tipo.

- Aplicación Web Estática:

https://www.freiduriacondal.com/

https://devilwearszara.vogue.es/

- Aplicación Web Dinámica

https://www.zara.com/es/

https://www.huffingtonpost.es/

- Aplicaciones Web del Portal

https://es.yahoo.com

http://www.madrid.org

- Aplicaciones Web Animadas

Sayviget.com

www.kaiserosa.com

- Aplicaciones de comercio electrónico

https://www.ebay.es/

https://www.zalando.es/

- Aplicaciones web con gestor de contenidos

WordPress.com

Joomla.com

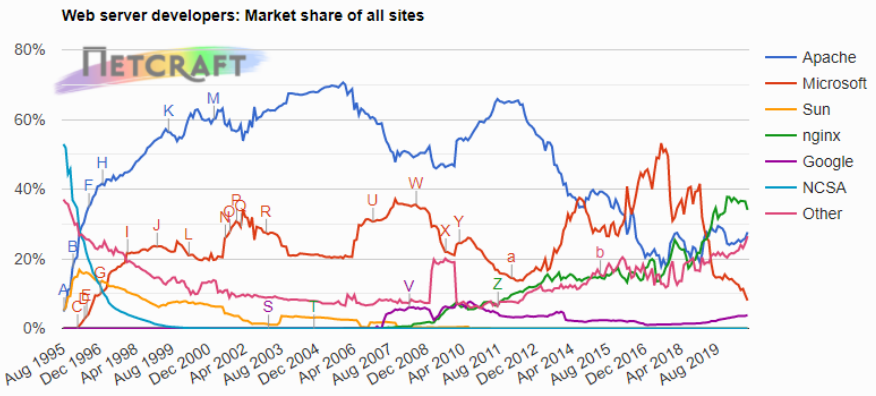
4. ¿Qué tecnologías se utilizan en cada una de las capas de la arquitectura web?

Capa de presentación: HTML, CSS, DOM, JavaScript, AJAX.

Capa de negocio: PHP, Java con sus tecnologías JavaServelts y JavaServer Pages (JSP), Perl, Ruby, Python, Node.js, C# y Visual Basic con sus tecnologías ASP/ASP.NET.

Capa de acceso a datos: MySQL.

5. Existen multitud de servidores web en el mercado. La empresa Netcraft se encarga de hacer estadísticas y mediciones sobre la utilización de los distintos servidores web. Se pide una gráfica con la utilización de los principales servidores web y las principales características de los cinco primeros.



Developer	August 2020	Percent	September 2020	Percent	Change
nginx	448,602,806	36.45%	407,086,442	34.03%	-2.43
Apache	318,307,245	25.87%	330,682,809	27.64%	1.78
Microsoft	118,126,662	9.60%	95,944,600	8.02%	-1.58

Gráfica principales servidores web

6. Diferencias y semejanzas entre Saas, PaaS e IaaS

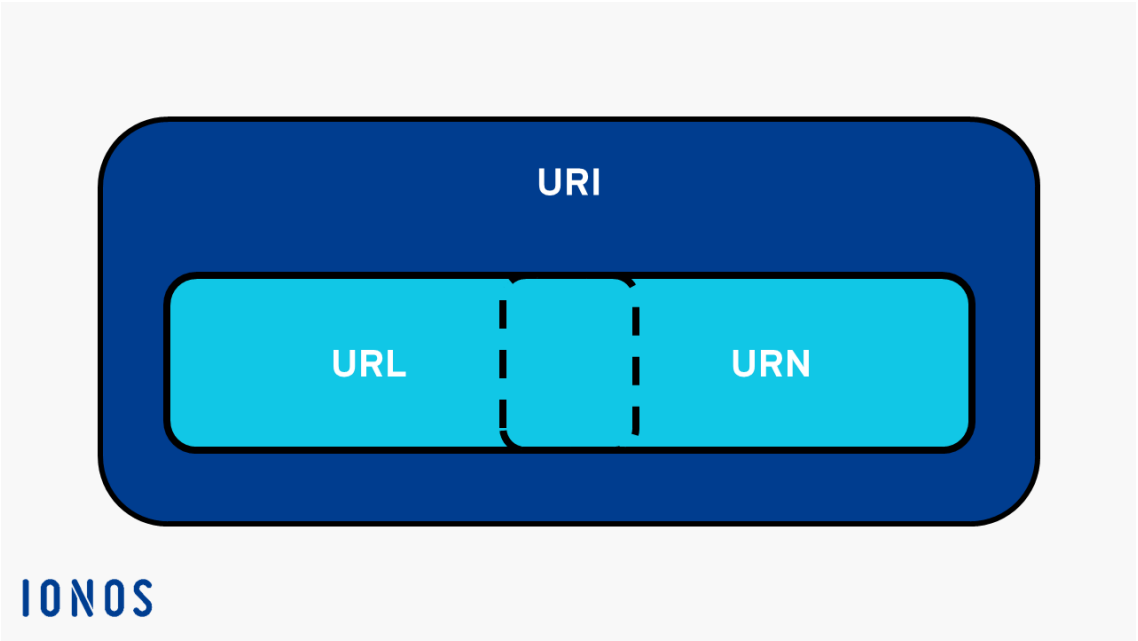
Comparación entre Saas, Paas e IaaS Semejanzas .- Todos ellos ofrecen servicio en línea, cuyo funcionamiento se da a través de la nube.. Además, ofrecen la posibilidad de pagar únicamente en función de los recursos a consumir y del uso que se les vaya a dar. Todos ellos son sistemas escalables, de modo que permite aumentar la capacidad siempre y cuando sea necesario. La principal diferencia entre SaaS, PaaS e Iaas tiene que ver con el mantenimiento y soporte ofrecidos por el proveedor. Mientras que en IaaS es el desarrollador de las aplicaciones el que debe ocuparse de todo, en SaaS, los usuarios ni siquiera tienen acceso al software. Por su parte en PaaS pueden gestionar la plataforma, pero no el servidor

7. Diferentes versiones disponibles de Apache Tomcat y sus diferentes versiones de las especificaciones

Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Authentication (JASIC) Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
5.0	3.0	4.0	2.0	2.0	10.0.x	10.0.0-M8 (alpha)	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.38	8 and later
3.1	2.3	3.0	1.1	1.1	8.5.x	8.5.58	7 and later
3.1	2.3	3.0	1.1	N/A	8.0.x (superseded)	8.0.53 (superseded)	7 and later
3.0	2.2	2.2	1.1	N/A	7.0.x	7.0.106	6 and later (7 and later for WebSocket)
2.5	2.1	2.1	N/A	N/A	6.0.x (archived)	6.0.53 (archived)	5 and later
2.4	2.0	N/A	N/A	N/A	5.5.x (archived)	5.5.36 (archived)	1.4 and later
2.3	1.2	N/A	N/A	N/A	4.1.x (archived)	4.1.40 (archived)	1.3 and later
2.2	1.1	N/A	N/A	N/A	3.3.x (archived)	3.3.2 (archived)	1.1 and later

Veriones de Apache Tomcat

8. URI, URN Y URL: definición, sintaxis (estructura) y diferentes ejemplos.



Ejemplo visual del concepto de URI, URL y URN

Los URL y URN pueden considerarse subcategorías del URI.

El URL y el URN presentan la misma sintaxis que el URI. Ambos tipos de identificadores son, por lo tanto, subcategorías del URI. Los URL y URN son un tipo de URI. Del mismo modo, ni URL ni URN son identificadores uniformes recursos.

El URL o uniform resource locator se utiliza para indicar dónde se encuentra un recurso. Por lo tanto, también sirve para acceder a algunas páginas web por Internet. Por el contrario, el URN o uniform resource name es independiente de la ubicación y designa un recurso de forma permanente. Por lo tanto, si el URL se conoce principalmente como una forma de identificar un dominio web, el URN también puede tratarse de un ISBN que identifique un libro de manera indefinida.

9. Escalabilidad horizontal: Ventajas y inconvenientes

¿Qué es la escalabilidad horizontal?

Cuando hablamos de escalabilidad horizontal nos estamos refiriendo a la capacitación de un sistema informático en nube para hacer que la calidad del mismo siga intacta aunque se haya producido un crecimiento desmesurado e imprevisto. Se trata de una de las soluciones más potentes, aunque antes de la llegada de la nube su implantación no resultaba un proceso sencillo ni económico.

Para hacer que un sistema se escalable horizontalmente debemos tener varios servidores, a los que también se les denomina nodos. Estos servidores tienen que trabajar al unísono y de esta manera crear una red (Cluster) que tiene como objetivo dividir todo el trabajo. Así, a medida que el número de usuarios va aumentando, la solución pasa por ir añadiendo más nodos. Dentro de este entorno que hemos creado, es importante que siempre haya un servidor principal que adopte el rol de administrador de los demás nodos.

¿Cuáles son las ventajas de la escalabilidad horizontal?

La potencia que representa a este tipo de solución en nube hace que pueda aportar un gran número de ventajas a nuestro sistema informático. Una de las principales es que el crecimiento del sistema casi no va a tener fin, puesto que vamos a poder ir añadiendo nodos a medida que los vayamos necesitando. Otra cuestión que hay que tener en cuenta es que la escalabilidad horizontal sigue funcionando aunque alguno de los nodos falle, asegurando así que la red no se caiga de forma repentina.

Por otro lado, es capaz de dar soporte a cualquier movimiento que se produzca en las cargas y a todo ello se le suma que la escalabilidad horizontal es 100% compatible con la vertical en el caso de que necesitemos hacer uso de ambas para que nuestro sistema informático tenga su correcto funcionamiento.

¿Tiene desventajas?

Si bien hay desventajas en la aplicación de esta tecnología, la nota positiva es que son minoritarias. En este caso, debemos decir que este tipo de escalabilidad necesita un gran mantenimiento para que todo funcione a la perfección. Trabajando en nube no se requiere una infraestructura tan compleja como antaño y la implementación de sus herramientas tampoco es tan exigente,

aunque sigue teniendo cierto margen de dificultad a tener en cuenta. En cualquier caso, la escalabilidad horizontal se presenta hoy en día como una de las mejores soluciones para mantener un entorno de cloud computing sin que el crecimiento pueda terminar afectando a la estabilidad del sistema.

10. Escalabilidad vertical: Ventajas y inconvenientes

¿Qué es el escalado vertical?

La escalabilidad vertical (o Scaling Up) es la manera más común de crecimiento en la nube. Básicamente se refiere a la instalación de más memoria o poder de procesamiento.

En términos de cloud podemos entenderlo como: redimensionar un servidor sin cambiar su código. Permite incrementar la capacidad del hardware o las instancias cloud al añadir más recursos. Sin embargo, habrá que tener en cuenta que sólo se puede escalar de acuerdo al tamaño del servidor actual (RAM y espacio del disco duro).

Ventajas de este modelo

A pesar de que este modelo de escalado puede resultar, en ocasiones, en mayor inversión en infraestructura, los beneficios del escalado vertical se enfocan en la rapidez y sencillez de su implantación:

Simple: No se necesita configurar más que el servidor con el que ya se cuenta. Esto nos evita cambios de código y programaciones extra y la manipulación de instancias múltiples. Todo se gestiona desde una sola instancia.

Ganancia de rendimiento: en cada actualización se obtiene mayor rapidez RAM y capacidad de memoria.

No se requiere cambio en el código de las instancias previamente instaladas. No requiere un diseño específico ni para su aplicación, ni para la arquitectura que se desee implementar.

Se adapta a la mayoría de las aplicaciones y es compatible con un gran número de licencias de software.

Si se trata de un entorno On-premise, este tipo de escalado es el adecuado en caso de no contar con un espacio físico de grandes dimensiones.

¿Tiene alguna desventaja?

Comparado con el modelo de escalado horizontal, podemos encontrar desventajas como la dificultad de llevar a cabo múltiples tareas o solicitudes de forma simultánea. Por otro lado, si el servidor excede las cargas de trabajo, los tiempos de disponibilidad podrían ser más bajos por los que el monitoreo de las instancias debe ser constante.

Conclusión

Tras la realización de estos ejercicios, he aprendido diferentes contenidos y conceptos básicos en relación con el desarrollo de aplicaciones-web.

Webgrafía

- https://developer.mozilla.org/es/docs/Web/HTTP/Overview
- https://tomcat.apache.org/whichversion.html
- https://www.heflo.com/es/blog/saas/saas-paas-iaas-diferencias/
- https://blog.mdcloud.es/que-es-la-escalabilidad-horizontal-en-un-servicio-de-cloud-computing/
- https://apser.es/entendiendo-la-escalabilidad-vertical/