Arquitectura. Contenidos Generales:

Arquitectura:

Arquitectura distribuida. Protocolo HTTP. Microservicios. Servidores. Concepto y uso. Modelo Cliente/Servidor.

Datacenters y Servidores:

Datacenters.

Servidores. Concepto y uso.

Protocolos de comunicación.

Tipos de servidores y protocolos soportados (mails, bases de datos, web´s, FTP).

Qué trabajo realiza un servidor

Cómo funciona la lógica de un servidor

Protocolos Cliente/Servidor

Sistemas operativos y conectividad.

Modelo Cliente Servidor

Tipos de servidores.

Desde el procesamiento: FTP - Correo - Web - IMAP - Streaming

Desde la estructura: Hosting -VPS - Servidor dedicado Cloud - Co-Location.

Comparativas: Amazon - Google Cloud - Microsoft Azure.

Arquitectura

Arquitectura distribuida

La arquitectura web trabaja con dos nodos:

- el **cliente** tiene un programa ejecutable (application client, el *web browser* o navegador es el más común)
- y el **servidor** tiene otro programa ejecutable: será nuestro server.

Estos nodos son lógicos: pueden estar ubicados físicamente en la misma máquina, pero igualmente tendremos una separación de componentes en cliente y servidor.

El cliente hace pedidos a través de un puerto contra el servidor, el servidor responde.

El flujo de mensajes siempre comienza en el cliente:

- **cliente** pide servicio (**request**)
- **servidor** responde (**response**)

Algunas consecuencias

- Nuestra aplicación pasa a ser una **aplicación distribuida**: va a tener una parte corriendo en el servidor y otra parte corriendo en el cliente.
  Dependiendo de la arquitectura que elijamos
  - podemos tener la mayor parte de la lógica en el servidor y tener un cliente liviano (thin) o ZAC (Zero Administration Client). Entonces lo que le llega al cliente es sólo un documento HTML, y es fácil mantener la aplicación cuando tengo muchos clientes ubicados
  - bien podemos poner gran parte de la lógica en el cliente y utilizar la parte server solamente para sincronizar la información entre sesiones de usuario

- de todas maneras, por más liviano que sea el cliente, los *browsers* no son uniformes, entonces si queremos que una aplicación ande en todos ellos muchas veces vamos a tener que manejar código específico para cada plataforma (browser, versión, sistema operativo y a veces hasta el hardware).
- como el cliente es el que dispara los pedidos, todas las interacciones entre el usuario y la aplicación deben ser iniciadas por el usuario, la aplicación no puede tomar la iniciativa. Ej: si tengo una lista de tareas pendientes, para que aparezca una nueva tarea hay que obligar al cliente a que dispare el refresh.

Pedido/respuesta

Antes de meternos más de lleno, nos preguntamos: la tecnología de objetos ¿es consistente con la metáfora "pedido-respuesta" (request/response)? Sí, en definitiva es la representación de lo que es un mensaje.

En la tecnología web siempre es el cliente el que pide y siempre el servidor el que responde.

Cómo se implementa la comunicación

El cliente dice: "necesito x". Esto se traduce en una dirección de una página en particular, esa dirección recibe el nombre de **URL** (Uniform Resource Locator, o forma de encontrar un recurso en el servidor):

http://localhost:8080/html-css/index.html

donde

- **http** es HyperText Transfer Protocol, el **protocolo de comunicación** por defecto que usan los navegadores
  - otros protocolos son: **https** (donde los datos viajan encriptados), **ftp**, etc.
- localhost es el **servidor web** hacia el que vamos a conectarnos
  - en este caso localhost es el web server que está en la PC local, que equivale a la dirección IP 127.0.0.1
  - el servidor web puede ser una dirección IP o un nombre que luego es convertido a una dirección IP a través de un DNS (Domain Name Server)
- 8080 es el puerto donde el servidor está "escuchando" pedidos
- y finalmente la página que queremos cargar, que recibe el nombre de **recurso**
- La forma en que publicaremos las páginas como **rutas** depende de la

tecnología en la que trabajamos, lo importante es entender que una página html es accesible para un usuario con una ruta única llamada URL.

Fuente:

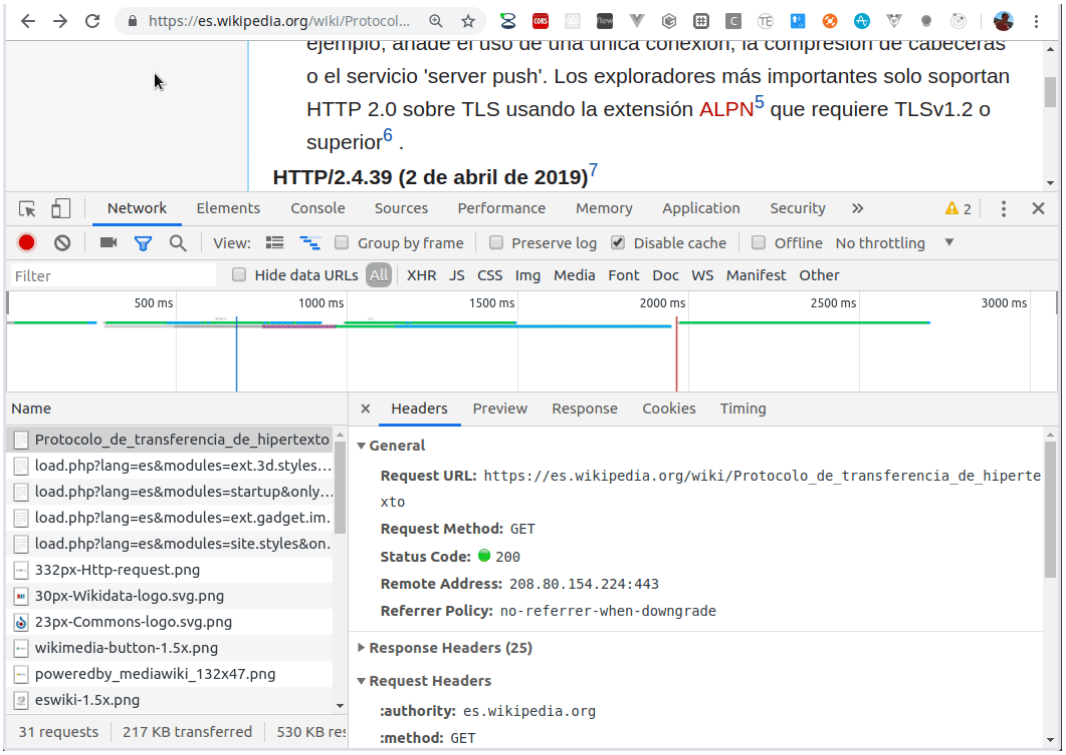
https://www.evaluandosoftware.com
https://codigofacilito.com
https://www.escuelapython.com
https://wiki.uqbar.org
https://anexsoft.com

HTTP

Http es un protocolo **no orientado a conexión** que define la forma de comunicación entre el cliente y el servidor.

No-orientado a conexión significa que no guarda ninguna información sobre conexiones anteriores, por lo que no tenemos el concepto de sesión de usuario, es un protocolo sin estado (*stateless protocol*). Esto tiene varias implicancias, la más fuerte es que requiere que la aplicación mantenga la información necesaria para mantener una sesión (por ejemplo, sabiendo qué usuario es el que está haciendo una operación).

Un mensaje http tiene formato de texto, por lo que es legible al usuario y fácilmente depurable, como vemos en el siguiente video:



Nota: ver video-1.gif

Abrimos en un navegador las herramientas de desarrollo (por lo general es la tecla F12), y en la solapa *Network* podemos inspeccionar las distintas respuestas que procesa el navegador, con el pedido http original que hace un mensaje de tipo `GET`.

Tipos de mensaje

Un cliente puede enviar un pedido al servidor utilizando diferentes métodos

- **GET**: asociada a una operación de lectura, sin ningún otro efecto

- **HEAD**: es exactamente igual al pedido vía GET pero enviando únicamente el resultado de la operación en un header, sin el contenido o *body*
- **POST**: se suele asociar a una operación que tiene efecto colateral, no repetible
- **PUT**: está pensado para agregar información o modificar una entidad existente
- **DELETE**: se asocia con la posibilidad de eliminar un recurso existente
- **OPTIONS**: permite ver todos los métodos que soporta un determinado servidor web
- **TRACE**: permite hacer el seguimiento y depuración de un mensaje http (se agrega información de debug)
- **CONNECT**: equivalente a un ping, permite saber si se tiene acceso a un host

Envío mediante GET method

Aquí los parámetros viajan dentro de la URL como par clave=valor:

http://www.appdomain.com/users?size=20&page=5

- ? delimita el primer parámetro
- & delimita los siguientes parámetros

La ventaja de utilizar este método es que dado que http es un protocolo no orientado a conexión, podemos reconstruir todo el estado que necesita la página a partir de sus parámetros (es fácil navegar hacia atrás o adelante). Por otra parte es el método sugerido para operaciones sin efecto, que recuperan datos de un recurso.

Por otra parte, no es conveniente para pasar información sensible (como password o ciertos identificadores), algunos navegadores imponen un límite máximo de caracteres para estos pedidos y necesita codificar los caracteres especiales (p. ej. el espacio a %20) dado que el request solamente trabaja con el conjunto de caracteres ASCII.

Envío mediante POST method

Los parámetros viajan en el BODY del mensaje HTML, no se ven en la URL del browser. Aquí no hay restricciones de tamaño para pasaje de información y tampoco se visualizan los parámetros en la URL del browser.

GET vs. POST

La recomendación W3C (World Wide Web Consortium) dice que deberíamos usar

- **GET** cuando sepamos que se trata de consultas que no van a tener efecto colateral (no habrá modificación en el estado del sistema)

- **POST** cuando sepamos que el procesamiento de la página causará una alteración del estado del sistema (al registrar el alquiler de una película, al modificar los datos de un socio o al eliminar un producto de la venta). Otros métodos posibles que veremos son PUT y PATCH, para modificaciones y alteraciones parciales, respectivamente.

El camino de un pedido http

- el browser se conecta con el servidor a partir del dominio o IP (localhost = 127.0.0.1) y puerto
- se envía la petición al servidor en base a dirección, método, parámetros, etc.
- el servidor responde a ese pedido: esa respuesta es una nueva página con un código de estado HTTP:
    - 200 : OK
    - 401 : Unauthorized
    - 403 : Forbidden
    - 404 : Not Found
    - 405 : Method not allowed
    - 500 : Internal Server Error

El lector puede buscar la lista de códigos de error HTTP (las especificaciones RFC 2616 y RFC 4918) y formas de resolverlos.

- la aplicación cliente o *user agent* se desconecta del servidor una vez procesada la respuesta

Consecuencias del mensaje http para las aplicaciones web

La página es la mínima unidad de información entre cliente y servidor, lo que implica:

- **problemas en la performance**: no siempre debería refrescar toda la página si sólo necesito actualizar parcialmente la información de dicha página
- **problemas en el diseño**: tengo dificultades para poder particionar una pantalla en componentes visuales
- **problemas de usabilidad**: para que la página sea dinámica necesitamos forzar una comunicación con el servidor

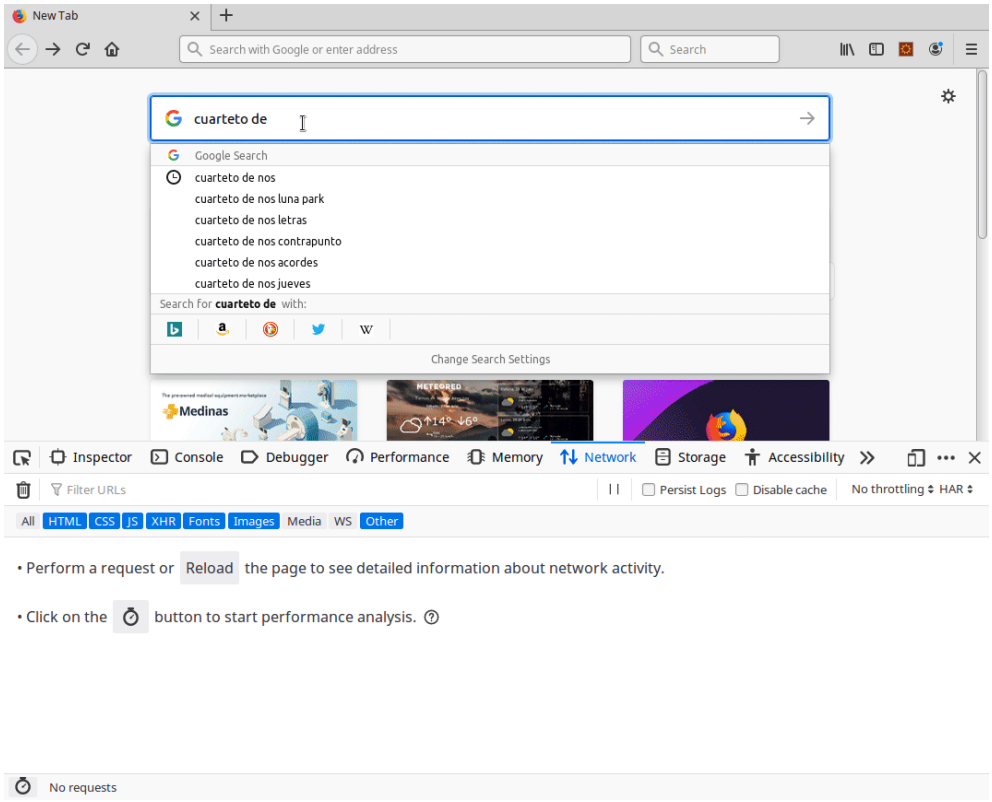
String oriented programming: la comunicación entre cliente y servidor involucra solo texto, necesitamos adaptar fechas, números, booleanos y también los objetos de

negocio (socios de un videoclub, alumnos, materias, vehículos de una flota, etc.) así como las colecciones.

Procesamiento de la respuesta en el cliente

El servidor contesta con un string que tiene

- un ***header*** donde indica el resultado del pedido
- un **contenido**, que forma parte del body, que puede ser HTML, json o cualquier otro formato que el cliente entienda



Arriba vemos la respuesta del navegador al buscar “Cuarteto de Nos”.

Nota: ver video-2.gif

Fuente:

https://www.evaluandosoftware.com
https://codigofacilito.com
https://www.escuelapython.com
https://wiki.uqbar.org
https://anexsoft.com

REST

"Representational State Transfer" o traducido a "Transferencia de presentación de estado" es lo que se domina a REST. ¿**Y eso es**?, una técnica de arquitectura de software usada para construir APIs que permitan comunicar a nuestro servidor con sus clientes usando el protocolo HTTP mediante URIs (https://es.wikipedia.org/wiki/Identificador_de_recursos_uniforme) lo suficientemente inteligentes para poder satisfacer la necesidad del cliente.

- REST es **STATELESS**, es decir que cada petición que reciba nuestra API **debe perecer**. Por ejemplo, **no podemos RECORDAR** un usuario logueado en el API usando una sesión, esto es un PECADO ya que agotaría la memoria RAM de nuestro servidor (10 mil usuarios conectados a nuestra API). Lo que correcto es pasar un TOKEN para cada petición realizada al API, y el API deberá validar si esta es correcta o no (por ahora no vamos hablar de técnicas para generar el TOKEN, pero lo más común es usar una COOKIE).
- Se implementan **RECURSOS** para generar comunicación, es decir crea URIs únicas que permiten al cliente entender y utilizar lo que está exponiendo. Por ejemplo:
    - api.anexsoft.com/users
    - api.anexsoft.com/users/1405
- Cada petición realizada a nuestra API responde a un verbo, y dicho verbo a una operación en común. Mediante los métodos HTTP hacemos las peticiones, lo común es GET y POST, PUT y DELETE.
    - **POST (create):** cuando mandamos información para insertar por ejemplo un registro en la base de datos. La información es enviada en el cuerpo de la petición, es decir que los datos no son visibles al usuario. api.anexsoft.com/users
    - **GET (read)**: es usado para modo lectura, por ejemplo: cuando queremos listar a todos los usuarios de nuestra base de datos. Los parámetros son enviados por la URL.
- api.anexsoft.com/users
    - **PUT (update):** cuando queremos actualizar un registro. Actualizar la información de un usuario X.
- api.anexsoft.com/users

    - **DELETE (delete):** cuando queremos eliminar un registro. Borrar un usuario X de nuestra base de datos.
- api.anexsoft.com/users

Con esto hemos mencionado algunas características básicas de lo que es REST, la cual podríamos decir que es un estándar para crear una REST API o RESTFul (https://www.codigonaranja.com/restful-web-service).

¿QUÉ ES RESTFUL?

Es un servicio que disponemos al público usando REST. REST es el concepto, RESTFul es la implementación y al crear un RESTFul creamos una API, la cual una API es un conjunto de funciones o procedimientos para que sea utilizado por otro software.

Ejemplo: la API de Google Maps, Youtube, Facebook, etc.

¿BAJO QUÉ CIRCUNSTANCIAS DEBERÍAMOS IMPLEMENTAR UN SERVICIO RESTFUL?

Se dice que hoy en día debemos pensar orientado al servicio. Se recomienda trabajar directamente con una REST API bajo estas situaciones.

- Si vamos a trabajar una aplicación del tipo SPA, si o si debemos crear una REST API. Un framework recomendado puede ser AngularJS, EmberJS, Vue.js, React, backbone entre otros.
- Si es una Web comercial y vas a exponer recursos al público, es muy recomendable crear una REST API.
- Si tu sistema no solo va a ser accedido desde una PC, también deberíamos pensar en crear una REST API, ya que permite crear escalabilidad.

En resumen, crear una REST API es una manera muy profesional de desacoplar tu sistema de la capa de persistencia.

PUNTOS A FAVOR

- **Separación entre el cliente y el servidor**
    - Nuestros proyectos se vuelven autónomos, por lo tanto no nos interesa si dicha tecnología es compatible con la otra ya que usaremos como un

medio de comunicación JSON.

- **No importa la tecnología**
  - Si eres PHP, .NET, Java, Python, etc. da igual, al final solo necesitas saber cómo consumir/responder al servicio.
- **Escalabilidad y Flexibilidad**
  - Realiza los cambios que quieras dentro de tu API, lo que interesa es que se respete el mismo mensaje o respuesta que le brindas al cliente para mantener la misma lógica.
- **¿Mejora de recursos consumidos por el servidor?**
  - REST no debe usar sesiones, por lo tanto disponemos de más memoria RAM.
  - Lo correcto es trabajar con formatos estandarizados como JSON no haremos uso de HTML para responder al cliente. En este caso ganamos velocidad.

PUNTOS EN CONTRA

- Mayor tiempo en desarrollo debido a que hay que plantear y estandarizar las respuestas de nuestra API para que se tornen amigables para quien la consume.
- Curva de aprendizaje incrementada, ya que como muchos están acostumbrados a trabajar enviado HTML por las peticiones AJAX, podría resultar un poco tedioso trabajar con JSON puro, entre ellas, entender el uso del framework que implementa la API o tener claro en el concepto si queremos desarrollar un framework desde cero.

Fuente:

https://www.evaluandosoftware.com
https://codigofacilito.com
https://www.escuelapython.com
https://wiki.uqbar.org
https://anexsoft.com

Microservicios

Los microservicios, o arquitectura de microservicios, es un enfoque para el desarrollo de software en el que una aplicación grande se construye como un conjunto de componentes o servicios modulares.
Cada módulo admite una tarea específica o un objetivo de negocio y utiliza una interfaz simple y bien definida, como una **interfaz de programación de aplicaciones (API)**, para comunicarse con otros conjuntos de servicios.

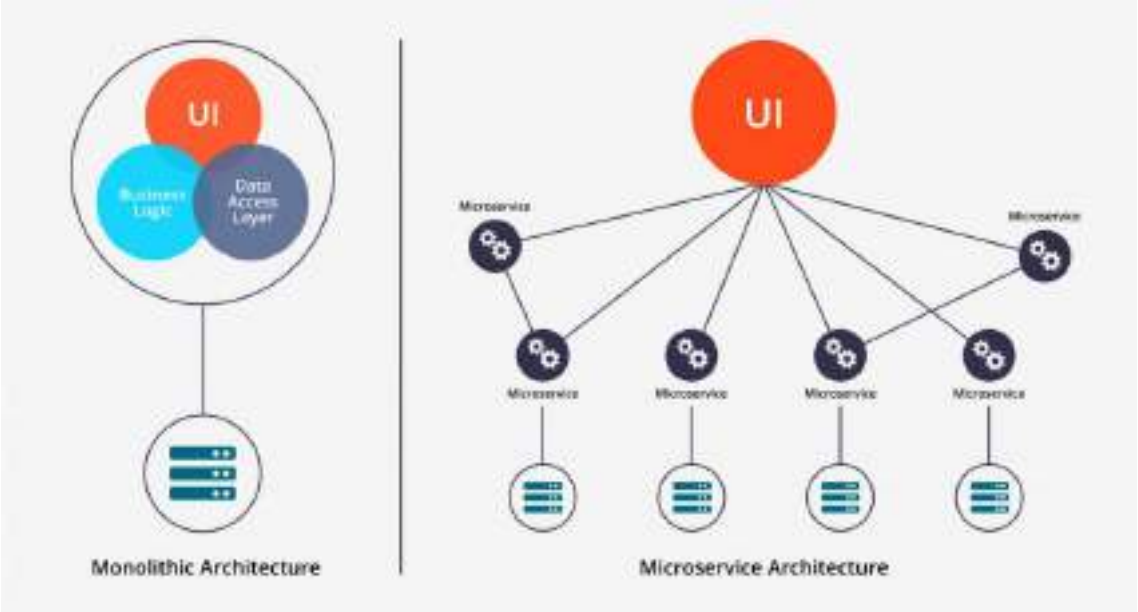
Cómo funcionan los microservicios

En la arquitectura de microservicios, una aplicación se divide en servicios. Cada uno ejecuta un proceso único y generalmente administra su propia base de datos. Un servicio puede generar alertas, registrar datos, admitir **interfaces de usuario** (UI), manejar la identificación o autenticación del usuario y realizar otras tareas.

El paradigma de microservicio proporciona a los equipos de desarrollo un enfoque más descentralizado para construir software. Los microservicios permiten que cada servicio sea aislado, reconstruido, re desplegado y administrado de forma independiente. Por ejemplo, si un programa no genera informes correctamente, puede ser más fácil rastrear el problema a ese servicio específico. Ese servicio específico podría luego probarse, reiniciarse, parchearse y volver a implementarse según sea necesario, independientemente de otros servicios.

Microservicios vs. Arquitectura monolítica

En una arquitectura monolítica, todo el código está en un archivo ejecutable principal, que puede ser más difícil de solucionar, probar y actualizar. Si hay un problema en una base de código, ese problema podría ubicarse en cualquier lugar dentro del software. Habría más pruebas, y éstas tardarían más debido a la cantidad de código monolítico involucrado. En una aplicación monolítica, cualquier pequeño cambio o actualización requiere compilar e implementar una versión completamente nueva de la aplicación. Esto significa que cualquier desarrollo de aplicaciones monolíticas implica una planificación, preparación, tiempo y gasto significativos.

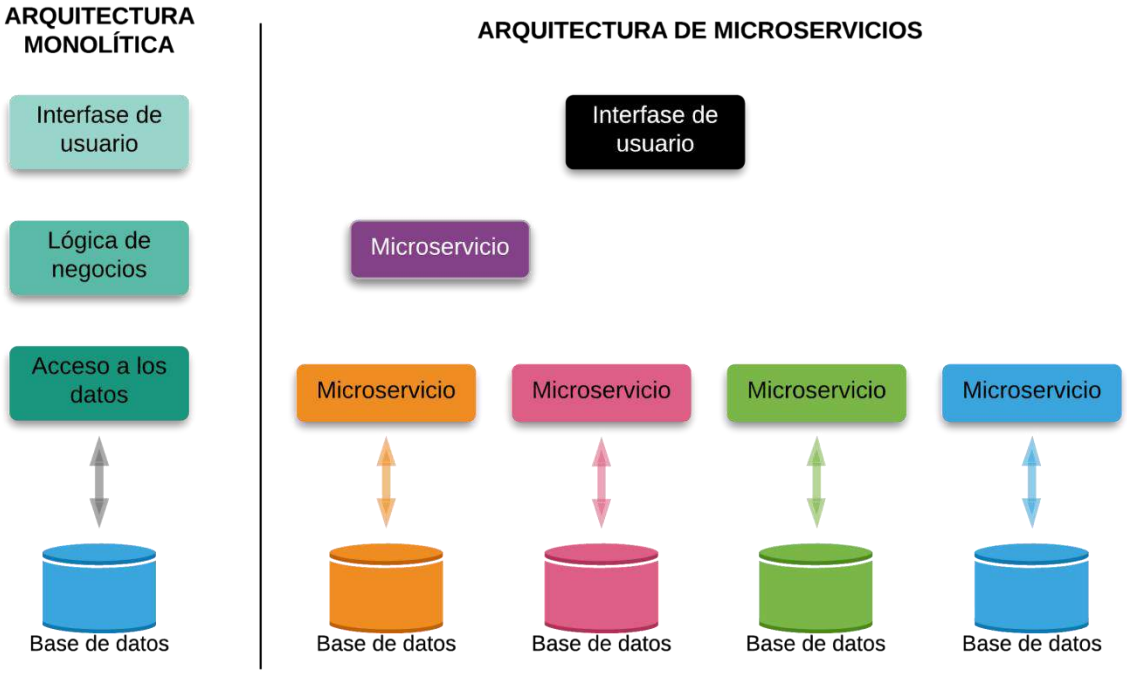


Además, las aplicaciones monolíticas son más difíciles de escalar. Cuando una aplicación monolítica alcanza una limitación de su capacidad, como el rendimiento de datos o algún otro cuello de botella, la única alternativa práctica es implementar otra iteración completa de toda la aplicación monolítica: administrar el tráfico entre las instancias utilizando balanceadores de carga.

En comparación, es posible escalar solo los servicios de una aplicación de microservicio agregando instancias de contenedor de solo esos servicios. Esto hace que el microservicio de escala sea mucho más eficiente en recursos que las aplicaciones de escala utilizando una arquitectura monolítica.

Los microservicios facilitan la prueba y el despliegue de cambios. Debido a que cada uno está separado de los otros, se mejora el aislamiento de fallas. Si hay un problema en el software, el servicio problemático puede ser aislado, remediado, probado y redistribuido sin la necesidad de realizar una prueba de regresión de toda la aplicación como ocurre con las arquitecturas de aplicaciones monolíticas tradicionales. La arquitectura de microservicios mejora la agilidad empresarial con un desarrollo e implementación de software más rápido en comparación con la arquitectura de software monolítica.

Sin embargo, los microservicios no son libres de gestión. Con la misma cantidad de servicios que un producto de software monolítico, la administración requerida en una arquitectura de microservicios podría ser más compleja ya que cada servicio está separado uno del otro. Esto puede llevar a dificultades para manejar todas las partes de un todo. Por ejemplo, se necesita una cuidadosa supervisión y administración para rastrear la disponibilidad y el rendimiento de todos los servicios de componentes que operan dentro de una aplicación de microservicio.



Ciclo de vida del desarrollo en microservicios

Para obtener los beneficios potenciales y esperados del desarrollo en microservicios (esto es aplicable a casi cualquier nueva forma de trabajo o gestión), es necesario asegurar los **Factores Críticos de Éxito** a desarrollar bajo este modelo.

El primer Factor Crítico de Éxito que debe asegurar cualquier organización (pública o privada), es descubrir los factores críticos de éxito que son aplicables a su contexto. En otras palabras, lo primero es asegurar la correcta "**Definición e implantación de una estrategia de desarrollo en microservicios**, que asegure la viabilidad del cambio y la obtención de los beneficios según el Ciclo de Vida previsto".

Una estrategia suele ser la visión de una situación futura con unos beneficios asociados, que nos transporta a un estado "ideal" en el que nos vemos trabajando con soltura y en un ambiente de inmensa alegría compartida bajo el nuevo modelo de desarrollo en el que todo es sencillo, funciona a la primera y los equipos están eufóricos y en armonía. De hecho, las estrategias no suelen fallar (son "perfectas") lo que falla es la implantación de la estrategia.

Dicho esto, nuestra recomendación es definir una **estrategia para microservicios** que se pueda implantar de forma real. Para ello nos basamos en el **Ciclo de Vida del desarrollo en microservicios**. Se muestra un esquema a continuación.

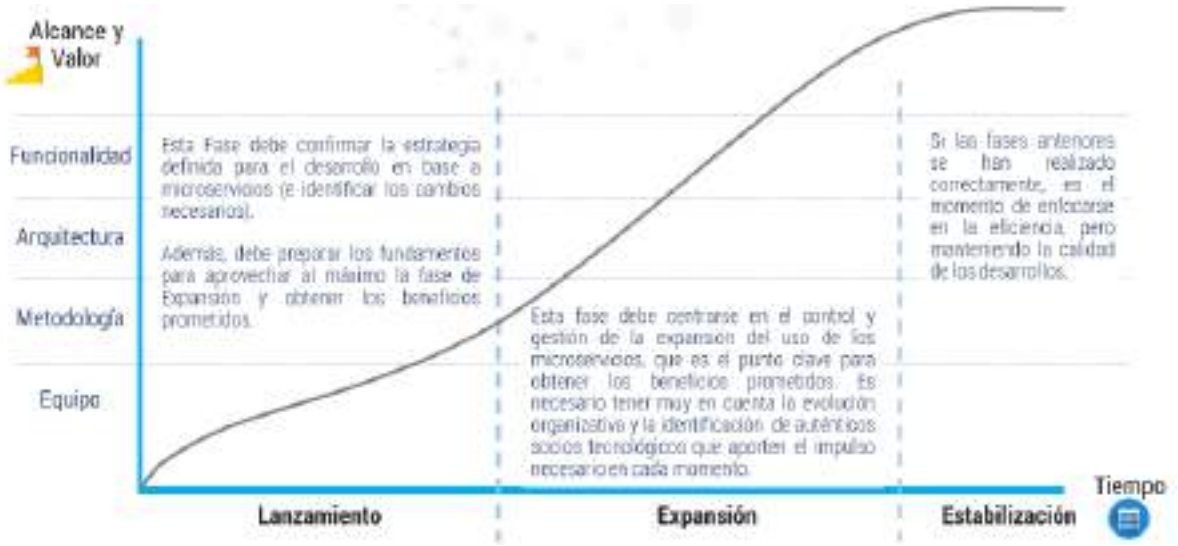


Figura: Microservicios. Ciclo de Vida. Implantación de estrategia de desarrollo en microservicios.

La **Estrategia de Desarrollo en microservicios** debe ser definida para cada entidad y debe ser implantable.

En general, el Ciclo de Vida se divide en tres Fases: Lanzamiento, Expansión y Estabilización. Cada una de estas fases necesita de cierto tiempo y otros recursos, a la vez que nos aportan cierto valor y alcance.

La **Fase de Expansión** es la que mayor beneficio nos da en el tiempo, por lo que cuanto antes lleguemos a ella antes obtendremos el valor del desarrollo en microservicios. Sin embargo, si la **Fase de Lanzamiento** no se realiza correctamente en la Fase de Expansión puede que nos de valor, pero a un coste insostenible. Adicionalmente, si la Fase de Expansión no se realiza correctamente, no podremos llegar a la “eficiencia prometida”, la intuiremos, pero no la obtendremos.

Nuestra recomendación es definir la Estrategia a la vez que definimos su implantación utilizando la metodología del Ciclo de Vida. Este enfoque supone hacernos las preguntas adecuadas y aportar respuestas viables a los ámbitos funcionales, de arquitectura, de metodología y de equipo (u organización) para cada fase.

Así obtendremos los Factores Críticos de Éxito (de la Estrategia de Desarrollo basada en microservicios) específica para cada organización (pública o privada) y para cada Fase del Ciclo de Vida. No hay que olvidar el definir qué resultados suponen la finalización de las fases y realizar una previsión del comportamiento de las principales magnitudes de la fase siguiente.

De forma práctica, este ejercicio no es complicado y se debe realizar independientemente del tamaño de los equipos, de los volúmenes de desarrollo o del alcance funcional que se quiere tener con el desarrollo en microservicios. Además, nos permitirá estimar la inversión necesaria (no solo la económica) y el retorno esperado.

La teoría es fácil y los conceptos también lo son, lo que ya no es tan fácil, y es en donde la experiencia y la visión global aportan su valor diferencial, es identificar los "**Factores Críticos de Éxito Reales**", los que de verdad hay que asegurar y anticipar su gestión en el Ciclo de Vida. ***¡Cuidado!, la tecnología debe estar al servicio de la estrategia y nunca la estrategia debe estar supeditada al uso de una tecnología determinada***.

Es importante seguir esta metodología antes de empezar con las iniciativas de desarrollo en microservicios, pero aún es mucho más importante si ya se ha empezado la iniciativa y no está funcionando como se espera. Si no se revisa la estrategia y su implantación podemos tener una Fase de Lanzamiento "eterna" con el sufrimiento organizacional que conlleva.

Claves de una arquitectura de microservicios

Adoptar un enfoque de microservicios cuando se construye un sistema desde cero o cuando se adapta uno que ya existe puede traer grandes beneficios. Sin embargo, hay una gran cantidad de **retos** a tener en cuenta que deben abordarse antes de que decidamos construir una arquitectura de microservicios.

Este tipo de sistemas tienden a permitir la construcción de servicios que son simples si utilizamos alguna de las soluciones disponibles, aunque generalmente la arquitectura que los respalda no es tan simple.

Procesos de negocio

Una aplicación más pequeña hace que puedas tener equipos o personas responsables con una mejor coordinación. La desventaja es que el ecosistema puede evolucionar en muchas aplicaciones diferentes a implementar y administrar.

Es importante recordar que un solo microservicio es casi siempre sólo una parte de una imagen más grande.

"Los microservicios son un medio para lograr un fin, un medio para ser más innovadores, flexibles y para reaccionar ante los nuevos requisitos o solicitudes de funciones."

Escalabilidad

Los elementos más importantes de un microservicio en sí son su nivel de granularidad y su accesibilidad a través de las API. **Los microservicios deben desarrollarse, implementarse y escalarse de manera independiente**, permitiendo una entrega más rápida de la funcionalidad con poco impacto en otros sistemas. Como tales, deben abordar un dominio lo suficientemente amplio como para evitar desafíos en torno a la

orquestación de «nanoservicios», pero lo suficientemente limitados para evitar dependencias innecesarias.

Una vez que se implementa un microservicio con la funcionalidad adecuada, debe ser consumible por otros servicios y aplicaciones, por lo que debe tener una API bien definida que los clientes del servicio puedan descubrir, comprender y ejecutar fácilmente.

Proceso Ágil

Establecer objetivos de medición para determinar si se está obteniendo valor de los microservicios. El objetivo es mejorar la capacidad de hacer cambios en cualquier aplicación de la compañía y llegar a los clientes rápidamente. Para tener éxito, debe trabajar dentro de los límites del equipo de no más de 10 a 12 personas por equipo.

A medida que cambie la arquitectura de sus aplicaciones a una **arquitectura de microservicios**, deberá modificar la forma en que organiza sus equipos para que estén alineados.

APIs

Las garantías de aislamiento y las APIs son los elementos más importantes de los microservicios.

El aislamiento permite a los desarrolladores iterar de forma rápida e independiente. Sin embargo, los desarrolladores deben mantener la compatibilidad con la API para que otros microservicios no se vean afectados (y crean dependencias que ralentizan las iteraciones).

Hay varios puntos que tenemos que tener en cuenta:

- **No todos los servicios son microservicios**. Al comenzar a trabajar en microservicios debes asegurarte que estén adecuadamente desacoplados. Si están fuertemente acoplados no será posible escalarlos de manera individual.

- **No empieces con muchos servicios al mismo tiempo**. Empieza poco a poco, con las funcionalidades clave y hazlo evolucionar.

- Conforme se vaya evolucionando el ecosistema de aplicaciones, **comprende dónde se encuentra el proyecto en la actualidad y qué objetivos se están intentando lograr**. Diseña aplicaciones para que los equipos pequeños puedan tener un control total del servicio o que la aplicación sea totalmente independiente de todo lo demás.

Microservicio pros y contras

La arquitectura de microservicios plantea una serie de concesiones para los desarrolladores de software. En términos de ventajas, se observan:

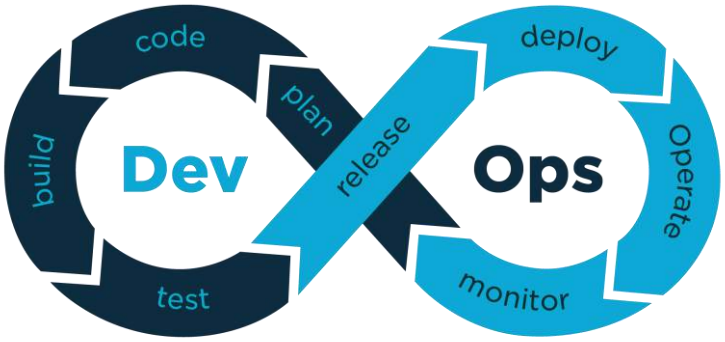
- Se despliegan fácilmente.
- Requieren menos tiempo de desarrollo.
- Puede escalar rápidamente
- Se puede reutilizar en diferentes proyectos.
- Contienen mejor aislamiento de fallas.
- Puede ser desplegado en equipos relativamente pequeños.
- Trabaja bien con los contenedores.

Sin embargo, también hay inconvenientes con la arquitectura de microservicios, tales como:

- Potencialmente demasiada granularidad.
- Esfuerzo extra de diseño para la comunicación entre servicios.
- Latencia durante el uso pesado.
- Pruebas complejas.

Microservicios y DevOps

DevOps (https://es.wikipedia.org/wiki/DevOps) combina tareas entre la aplicación y los equipos de operaciones del sistema. Con el aumento de las comunicaciones entre los desarrolladores y el personal de operaciones, un equipo de IT puede crear y administrar mejor la infraestructura. Las operaciones de IT aseguran el presupuesto para capacidades, tareas operativas, actualizaciones y más. Los desarrolladores y los equipos de aplicaciones pueden administrar bases de datos, servidores, software y hardware utilizados en la producción.



El trabajo en equipo y la colaboración entre los equipos de desarrollo y operaciones son necesarios para soportar el ciclo de vida de los microservicios, prestándose a los equipos de DevOps. También es la razón por la que los equipos de DevOps experimentados están bien equipados para emplear arquitectura de microservicios en proyectos de desarrollo de software.

DevOps: https://www.youtube.com/watch?v=p-bOnV8FRMQ&ab_channel=Exceltic

Arquitectura de microservicios vs. SOA

SOA (https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios) es una arquitectura de software donde cada uno de sus servicios utiliza protocolos. Esto permite a los usuarios combinar funcionalidades y dar vida a aplicaciones creadas a partir de servicios anteriores. SOA ha sido la práctica de desarrollo estándar durante casi dos décadas. Sin embargo, el ingenio de SOA se pone en tela de juicio cuando se trabaja con la computación en la nube. Con la nube, SOA carece de escalabilidad y se ralentiza con los cambios de solicitud de trabajo, lo que limita el desarrollo de la aplicación.

Muchos desarrolladores consideran que la arquitectura de microservicios es un enfoque más granular de SOA. Los defensores del modelo SOA creen que la arquitectura de microservicios es la evolución natural de SOA necesaria para adaptarse a la **computación en la nube** y satisfacer las crecientes demandas de ciclos de desarrollo de software más rápidos.

Otros creen que los microservicios son un enfoque más independiente de la plataforma para el desarrollo de aplicaciones y, por lo tanto, deben tener un nombre único. Este grupo podría argumentar que SOA vive en las capas de administración de microservicio.

Microservicios y contenedores

Un contenedor es un paquete de software individual y ejecutable, que incluye todas las dependencias que se necesitan para funcionar de manera independiente. Los contenedores están separados del resto del software que los rodea, y muchos contenedores pueden emplearse en el mismo entorno. En una arquitectura de microservicios, cada servicio se crea en contenedores individualmente en el mismo entorno, como el mismo servidor o los servidores relacionados.

Una máquina virtual (VM) se puede utilizar como alternativa a los contenedores para crear microservicios. Una máquina virtual simula los sistemas informáticos para producir funcionalidades de una computadora física. Cada servicio podría potencialmente utilizar una máquina virtual para alojar una característica prevista. Sin embargo, las máquinas virtuales a menudo se evitan para los microservicios debido al sistema operativo individual (SO) y otros gastos generales necesarios para cada máquina virtual. Los contenedores son mucho más eficientes en el uso de recursos porque solo se requieren el código subyacente y las dependencias relacionadas para operar el servicio.

Seguridad de la arquitectura de microservicios

La arquitectura de microservicios puede aliviar algunos problemas de seguridad que surgen con las aplicaciones monolíticas. Los microservicios simplifican el monitoreo de seguridad porque las diversas partes de una aplicación están aisladas. Una brecha de seguridad podría ocurrir en una sección sin afectar otras áreas del proyecto. Los microservicios ofrecen resistencia contra los ataques distribuidos de denegación de

servicio (DDoS) cuando se usan con contenedores al minimizar una toma de control de la infraestructura con demasiadas solicitudes de servidor.

Sin embargo, todavía existen desafíos al proteger aplicaciones de microservicios, que incluyen:

- Más áreas de red están abiertas a vulnerabilidades.
- Una menor coherencia general entre las actualizaciones de la aplicación permite más violaciones de seguridad.
- Hay una mayor área de ataque, a través de múltiples puertos y **APIs**.
- Hay una falta de control de software de terceros.
- La seguridad debe mantenerse para cada servicio.

Los desarrolladores de microservicios han ideado estrategias para aliviar los problemas de seguridad. Para ser proactivo, use un escáner de seguridad, utilice limitaciones de control de acceso, redes internas seguras, etc.

Despliegue de aplicaciones de microservicio

Se han producido tres avances principales para hacer viable la arquitectura de microservicios.

El primer avance, los contenedores, permite un medio consistente y eficiente de recursos para empaquetar servicios individuales. Docker (https://www.dataart.com.ar/news/para-que-nos-sirve-docker/) es una herramienta popular para desarrolladores, que permite el uso de contenedores en las instalaciones o en la nube pública o privada. Esto ofrece una amplia variedad de alternativas de implementación para desarrolladores de microservicios y empresas.

El segundo desarrollo importante es la aparición de herramientas de orquestación, como Kubernetes (https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/). Estas herramientas ayudan a automatizar el escalado, la implementación y la administración de contenedores.

Un tercer avance importante para los microservicios es la evolución de la malla. Una malla de servicios es una capa de infraestructura dedicada a la comunicación entre servicios individuales. Las mallas de servicio hacen que estas comunicaciones sean más rápidas, más seguras, visibles y confiables.

Cuando cientos de servicios se comunican entre sí, se vuelve complicado saber cuáles interactúan entre sí. Linkerd (https://es.slideshare.net/paradigmatecnologico/linkerd-a-fondo-1) es una herramienta de orquestación que logra comunicaciones seguras, rápidas y visibles entre los servicios mediante la captura de comportamientos, como el equilibrio de carga compatible con la latencia o el descubrimiento del servicio.

Los niveles de complejidad en los microservicios están disminuyendo constantemente debido a estos avances, como las complejidades de la supervisión y el registro, a medida que más organizaciones y equipos de desarrollo adoptan los microservicios.

Fuente:
https://www.evaluandosoftware.com
https://codigofacilito.com
https://www.escuelapython.com
https://wiki.uqbar.org
https://anexsoft.com

Datacenters y Servidores

¿Qué es un Datacenter?

El término Data Center es un término habitual para muchos. Sin embargo entrar en la definición del mismo puede ser interesante para otros muchos y, como una imagen vale más que mil palabras, los más interesados pueden ver en este video: https://youtu.be/XZmGGAbHqa0

algunas de las instalaciones del Data Center de la empresa Google y como es un el personal que trabaja en esos centros de información.

Un Data Center es, tal y como su nombre indica, un “centro de datos” o “Centro de Proceso de Datos” (CPD). Esta definición engloba las dependencias y los sistemas asociados gracias a los cuales:

- Los datos son almacenados, tratados y distribuidos al personal o procesos autorizados para consultarlos y/o modificarlos.
- Los servidores en los que se albergan estos datos se mantienen en un entorno de funcionamiento óptimo.

Los primeros Data Centers se diseñaron siguiendo las arquitecturas clásicas de informática de red, en las que los equipos eran “apilables” en mesas, armarios o racks como la siguiente imagen:

La necesidad de fácil gestión y de optimización del espacio han hecho que se evolucione hacia sistemas basados en equipos cuyas dimensiones permiten aprovechar al máximo el volumen disponible en los racks (equipos “enracables”), logrando una alta densidad de equipos por unidad de espacio.

Un Datacenter físico puede alojar Datacenters virtuales, cuyo costo es menor gracias a la virtualización. Cada Centro de Datos virtual es independiente del resto y dispone de máximas garantías de seguridad, disponibilidad y flexibilidad.

Los Data Center iniciales tampoco estaban diseñados para proporcionar facilidades de red avanzadas, ni los requerimientos mínimos de ancho de banda y velocidad de las arquitecturas actuales. La rápida evolución de Internet y la necesidad de estar conectados en todo momento han obligado a las empresas a requerir un alto nivel de fiabilidad y seguridad, de tal forma que se proteja la información corporativa y esté disponible sin interrupciones o degradación del acceso, con el objetivo de no poner en peligro sus negocios, sean del tamaño que sean. El cumplimiento de estos requisitos, cada día mas demandados, es posible dentro de un Data Center. Igual que un banco es el mejor sitio para guardar y gestionar el dinero, un centro de datos lo es para albergar los equipos y sistemas de información.

Los datos almacenados, no son datos estáticos, están en constante movimiento, se interrelacionan unos con otros y dan como resultado nuevos datos. Su crecimiento es constante y ello implica no solo que deben estar protegidos mediante las medidas de seguridad adecuadas, sino también dotados de estupendos “motores que les permitan moverse ágilmente por las autopistas de la información”.

El crecimiento exponencial del número de usuarios de los servicios online ha llevado a las empresas a subcontratar la gestión, mantenimiento y administración de sus equipos informáticos y de comunicaciones en los Data Center. Esto les permite centrarse en el desarrollo de su propio negocio y olvidarse de complejidades tecnológicas derivadas de las características anteriormente comentadas, así como prestar el servicio sin la necesidad de realizar una inversión elevada en equipamiento dedicado a este fin.

Servidores externos

¿Qué busca una empresa contratando infraestructura externa?

El modelo de Datacenter se ha replicado y mejorado con el tiempo para que las empresas clientes puedan solicitar:

· **Ancho de banda**: Es necesario contar con una gran capacidad de transferencia de datos, de tal forma que no sea apreciable ningún tipo de “retardo” provocado por la utilización de la red.

· **Fiabilidad y Alta disponibilidad**: Los sistemas deben responder a cualquier situación crítica, haciendo posible la prestación del servicio sin pérdida apreciable de calidad incluso cuando es necesario atender gran cantidad de peticiones de forma puntual (“pico”) o continuada. Resulta imprescindible contar con sistemas de alta disponibilidad y redundancia a través de modernas arquitecturas de red y servicios.

· **Seguridad**: Cubriendo fundamentalmente tres aspectos:

1 – Seguridad física: Comprendiendo la seguridad de los sistemas hardware, soportes, dependencias y demás entidades «tangibles» del entorno del Data Center.

2 – Seguridad lógica: Incluyendo los aspectos de protección aportados por aplicaciones, protocolos y procesos que intervienen en el sistema, y complementado por elementos de seguridad de red (cortafuegos), detección de intrusos (IDS), y análisis a nivel de aplicación (antivirus).

3 – Seguridad político-corporativa: Formada por los aspectos de seguridad relativos a política general de la organización, normas, procedimientos y convenciones internas aplicables. En este aspecto se debe tener en cuenta el cumplimiento de la legislación aplicable.

Dichas áreas están interrelacionadas, y la existencia coherente de medidas de seguridad en cada una de ellas garantiza el nivel de protección óptimo frente a las posibles amenazas de seguridad.

· Escalabilidad, flexibilidad y rapidez a la hora de implementar sus proyectos (time–to-market): Los equipos y los web sites del Data Center tienen que funcionar con la misma agilidad y rapidez que Internet para ser competitivos. La infraestructura existente tiene que permitir poner en marcha un proyecto en el mínimo tiempo posible, así como ampliar el número de elementos o la capacidad de los existentes de forma rápida y sin impacto en el servicio. Cada proyecto requerirá una solución específica cuyos requisitos variarán en el tiempo, si la infraestructura y el Data Center no permiten efectuar estos cambios con la celeridad necesaria nunca se logrará estar a la altura de la demanda del usuario o el servicio prestado por los posibles competidores.

· Arquitecturas sofisticadas de comunicaciones: Para cubrir la demanda de los usuarios es necesario contar con los últimos desarrollos que la industria Internet lanza al mercado (firewalls, balanceadores de carga, sistemas de replicación de contenidos, etc).

· La gestión y administración por personal especializado: Los sistemas implicados en la prestación del servicio requieren un alto nivel de especialización. Así mismo, la

monitorización continuada 24×7 y detección precoz de errores se han convertido en requisitos imprescindibles para garantizar la calidad del servicio prestado, y lograr detectar los “cuellos de botella” antes de que representen un problema.

Tipos de Servidores Web

VPS + Cloud + Hosting Compartido + Dedicado + Hosting Wordpress

VPS

Un servidor virtual privado (VPS, por sus siglas en inglés) es una partición virtual dentro de un servidor físico que le asigna recursos exclusivos a cada partición. Éstas otorgan acceso raíz que permite instalar un sistema operativo y trabajar con alto grado de libertad.

El hosting VPS es uno de los servicios de alojamiento más populares que puedes elegir para tu sitio web. Utiliza tecnología de virtualización para proporcionarte recursos dedicados (privados) en un servidor con múltiples usuarios.

Es una solución más segura y estable que el hosting compartido donde no se obtiene espacio de servidor dedicado. Sin embargo, es de menor escala y más barato que alquilar un servidor completo.

El hosting VPS generalmente es elegido por los propietarios de sitios web que tienen un tráfico de nivel medio que excede los límites de los planes de hosting compartido pero que aún no necesitan los recursos de un servidor dedicado.

Las soluciones de VPS generalmente ofrecen varios planes de alojamiento. Por ejemplo, una empresa de alojamiento, tiene seis planes de hosting VPS que se adaptan a las diferentes necesidades y te permiten escalar tu sitio sin problemas cuando necesites más recursos.

¿Cuándo usar VPS y cómo decidir si es hora de una mejora?
¿Hay alguna alternativa al VPS? ¿Qué es un VPS administrado?
En resumen, ¿qué es un VPS?
¿Cómo funciona el hosting VPS?

Un servidor es una computadora en la que tu proveedor de alojamiento web almacena los archivos y las bases de datos necesarios para tu sitio web. Cada vez que un visitante en línea quiere acceder a tu sitio web, su navegador le envía una solicitud a tu servidor y transfiere los archivos necesarios a través de Internet. El hosting VPS te proporciona un servidor virtual que simula un servidor físico; sin embargo, en realidad, la máquina se comparte entre varios usuarios.

Al usar la tecnología de virtualización, tu proveedor de alojamiento web instala una capa virtual sobre el sistema operativo del servidor. Esta capa divide el servidor en particiones y le permite a cada usuario instalar su propio sistema operativo y software.

Por lo tanto, un servidor privado virtual (VPS) es tanto virtual como privado porque tienes control absoluto. Está separado de otros usuarios del servidor a nivel del sistema operativo. De hecho, la tecnología VPS es similar a la creación de particiones en tu computadora cuando quieres ejecutar más de un sistema operativo (por ejemplo, Windows y Linux) sin tener que reiniciar.

Un VPS te permite configurar tu sitio web dentro de un contenedor seguro con recursos garantizados (memoria, espacio en disco, núcleos de CPU, etc.) que no tienes que

compartir con otros usuarios. Con el hosting VPS, tienes el mismo acceso de nivel raíz que si alquilaras un servidor dedicado, pero a un costo mucho más bajo.

Comparación de VPS con otros tipos de alojamiento web
Los diferentes tipos de alojamiento web te permiten realizar diferentes niveles de personalización en tu servidor. Varían en precio, rendimiento (por ejemplo, tiempo de carga de la página) y la disponibilidad del servicio (por ejemplo, tiempo de actividad). A continuación, cómo el hosting VPS se compara con otras soluciones de hosting.

Hosting Compartido
El hosting compartido es la solución para los propietarios de sitios web con menos tráfico. Es el punto de partida de la mayoría de las pequeñas empresas y bloggers. Con el hosting compartido, se divide el mismo servidor físico entre varios clientes de la empresa de hosting. No obtienes recursos dedicados para ti, ya que tu sitio se ejecuta en el mismo sistema operativo que el de los demás.

Por lo tanto, la memoria y el poder de computación que tu sitio puede usar se ven afectados por las necesidades de los otros usuarios del servicio. Por ejemplo, si hay un pico de tráfico repentino en un sitio web alojado en el mismo servidor, puede aumentar el tiempo de carga de tu página. Tampoco puedes elegir tu sistema operativo ni otro software de servidor, ya que todos los usuarios usan la misma configuración. En definitiva, es tu proveedor de hosting el que se encarga de cada aspecto de tu entorno de hosting compartido.

Puedes pensar en hosting compartido como un alquiler en el que compartes el mismo apartamento con varios compañeros. El hosting VPS sigue siendo una solución compartida, sin embargo, cada quien tiene su propio cuarto donde pueden personalizar el espacio de acuerdo a sus necesidades. Por ejemplo, pueden elegir la pintura, los muebles, la decoración, etc.

Cloud Hosting
Con el cloud hosting o hosting en la nube no usas un solo servidor, sino un clúster (varias computadoras interconectadas) que se ejecuta en la nube. Cada servidor en el clúster almacena una copia actualizada de tu sitio web. Cuando uno de los servidores está demasiado ocupado, el clúster redirige automáticamente el tráfico a uno que no lo esté tanto. Como resultado, el hosting en la nube no tiene tiempo de inactividad (downtime), ya que siempre hay un servidor en el clúster que puede atender las solicitudes de los visitantes de tu sitio web.

El alojamiento en la nube y el alojamiento VPS no son mutuamente excluyentes. En cambio, muchas empresas de hosting ofrecen alojamiento VPS en una infraestructura en la nube. Esta es también una solución que hemos elegido para comentar, ya que descubrimos que la combinación de VPS con las tecnologías en la nube conduce al rendimiento y la confiabilidad más altos posibles en el entorno del servidor virtual.

Hosting WordPress
El hosting WordPress es un servicio específicamente ofrecido para los propietarios de sitios de WordPress. Viene con varias características relacionadas con WordPress que sólo puedes usar si tienes un sitio de WordPress, como instalación con un solo clic, plugins preinstalados o una interfaz de línea de comandos de WP. Los servidores están configurados para las necesidades de WordPress. Por lo tanto, los proveedores de hosting ofrecen hosting para WordPress como parte de su servicio de hosting compartido.

Aunque también es posible configurar un sitio de WordPress en un servidor privado virtual, no puedes obtener acceso a los servidores personalizados que se han configurado con WordPress en mente. Sin embargo, si aún eliges VPS para tu sitio de WordPress, puedes configurar tu entorno de alojamiento según las necesidades de tu negocio.

Hosting dedicado
Con hosting dedicado, alquilas un servidor físico completo para tu negocio. Si tienes un sitio web de alto tráfico, el hosting dedicado puede ser la mejor solución para ti, ya que los servidores dedicados son rápidos, flexibles y totalmente personalizables. Sin embargo, el servicio también viene con un precio, por eso no es la mejor solución para todos, especialmente si tienes un sitio web pequeño o mediano.

Si bien el hosting VPS te permite elegir y configurar tu sistema operativo y aplicaciones del servidor, el hosting dedicado va un paso más allá. No solo te permite configurar el software sino también el hardware, ya que todo el servidor es tuyo y nadie tiene voz en cuanto a la configuración. También puedes ejecutar un servidor dedicado en el sitio (por ejemplo, en tu oficina), sin embargo, no obtendrás el apoyo de un equipo de hosting profesional en ese caso.

Pros y contras del VPS
El hosting VPS puede ser una solución ideal para ti si es el servicio que realmente necesitas. A continuación, puedes leer los pros y contras de tener un servidor privado virtual.

Pros
- Es más rápido y más confiable que un servidor de hosting compartido.
- Dado que los recursos del servidor, como la memoria o la potencia de procesamiento, están garantizados, existe una fluctuación de cero a mínima en los recursos disponibles.
- Los problemas y aumentos de tráfico de otros usuarios del servidor no afectan a tu sitio.
- Obtienes acceso de superusuario (raíz) a tu servidor.
- Obtienes mayor privacidad, ya que tus archivos y bases de datos están bloqueados para otros usuarios del servidor.
- Es un servicio fácil de escalar. A medida que tu sitio web crece, puedes mejorar fácilmente los recursos de tu servidor (RAM, CPU, espacio en disco, ancho de banda, etc.).

Contras
- Es más caro que el hosting compartido.
- Se requiere más conocimiento técnico para administrar tu servidor.
- Los servidores mal configurados pueden generar vulnerabilidades de seguridad.

¿Cuándo usar VPS y cómo decidir si es hora de una mejora?
El hosting VPS generalmente se ve como el paso siguiente después de que tu sitio web crezca fuera de los límites de recursos del hosting compartido. Si el plan de hosting web compartido más avanzado ya no es suficiente para ejecutar tu sitio web sin problemas, vale la pena pasarse a un plan de VPS. En tales casos, el hosting VPS puede proporcionarte lo mejor de dos mundos: hosting compartido y dedicado.

Sin embargo, hay otros casos de uso en los que puede ser una buena idea comenzar con un plan de VPS desde el principio. Por ejemplo, los sitios web de comercio electrónico

donde se necesita garantizar pagos seguros con un entorno rápido y estable pueden beneficiarse mucho de un servidor privado virtual. De hecho, si almacenas cualquier tipo de información confidencial o tienes que procesar pagos en línea, un VPS puede ayudarte a reducir el riesgo de violaciones de seguridad y robo de identidad.

Si esperas recibir picos de tráfico en determinados momentos en tu sitio, por ejemplo, en el caso de un sitio web de planificación de eventos o de venta de boletos, un plan de hosting VPS puede ser un regalo del cielo. Los servidores de juegos y otros sitios con muchos recursos también pueden ejecutarse con un rendimiento mucho mejor en un servidor privado virtual.

¿Hay alguna alternativa al VPS? ¿Qué es un VPS administrado?
Uno de los mayores escollos del hosting VPS es tener que gestionar todo el entorno del servidor por tu cuenta. Si tu servidor virtual no está bien configurado y mantenido, podrías enfrentarte con serios riesgos de seguridad y de pérdida de rendimiento. Los servicios administrados de hosting VPS han aparecido en el mercado como una solución viable a este problema. Puedes considerarlo como una alternativa mejorada al hosting VPS, que viene con soporte técnico completo.

El VPS administrado sigue siendo una novedad en el mercado, sin embargo cada vez más proveedores de hosting lo ofrecen a sus clientes. Por ejemplo, proporcionar un entorno de hosting totalmente administrado con planes de cloud hosting.

En resumen, ¿qué es un VPS?
El hosting VPS te brinda acceso a recursos garantizados y un control total sin tener que manejar tu propio servidor. Es una excelente solución para sitios web de mediano a alto tráfico, de recursos pesados y de comercio electrónico. Sin embargo, las empresas que esperan un crecimiento rápido también pueden beneficiarse de la estabilidad del servicio. Si quieres un entorno de hospedaje confiable y de escala empresarial a un precio amigable, definitivamente vale la pena considerar el hosting VPS para tu sitio web.

Cómo funciona un servidor

¿Qué es un servidor?
El término servidor tiene dos significados en el ámbito informático. El primero hace referencia al ordenador que pone recursos a disposición a través de una red, y el segundo se refiere al programa que funciona en dicho ordenador. En consecuencia aparecen dos definiciones de servidor:

Definición Servidor (hardware): un servidor basado en hardware es una máquina física integrada en una red informática en la que, además del sistema operativo, funcionan uno o varios servidores basados en software. Una denominación alternativa para un servidor basado en hardware es "host" (término inglés para "anfitrión"). En principio, todo ordenador puede usarse como "host" con el correspondiente software para servidores.

Definición Servidor (software): un servidor basado en software es un programa que ofrece un servicio especial que otros programas denominados clientes (clients) pueden usar a nivel local o a través de una red. El tipo de servicio depende del tipo de software del servidor. La base de la comunicación es el modelo cliente-servidor y, en lo que concierne al intercambio de datos, entran en acción los protocolos de transmisión específicos del servicio.

¿Cómo funciona un servidor?
La puesta a disposición de los servicios del servidor a través de una red informática se basa en el modelo cliente-servidor, concepto que hace posible distribuir las tareas entre los diferentes ordenadores y hacerlas accesibles para más de un usuario final de manera independiente. Cada servicio disponible a través de una red será ofrecido por un servidor (software) que está permanentemente en espera. Este es el único modo de asegurar que los clientes como el navegador web o los clientes de correo electrónico siempre tengan la posibilidad de acceder al servidor activamente y de usar el servicio en función de sus necesidades.

Modelo cliente-servidor
Tipos de servidores
La comunicación entre cliente y servidor depende de cada servicio y se define por medio de un protocolo de transmisión. Este principio puede aclararse partiendo de los siguientes tipos de servidores:

Servidor web: la tarea principal de un servidor web es la de guardar y organizar páginas web y entregarlas a clientes como navegadores web o crawlers. La comunicación entre servidor (software) y cliente se basa en HTTP, es decir, en el protocolo de transferencia de hipertexto o en HTTPS, la variante codificada. Por regla general, se transmiten documentos HTML y los elementos integrados en ellos, tales como imágenes, hojas de estilo o scripts. Los servidores web más populares son el servidor HTTP Apache, los servicios de Internet Information Server de Microsoft (ISS) o el servidor Nginx.

Servidor de archivos: un servidor de archivos se encarga de almacenar los datos a los que acceden los diferentes clientes a través de una red. Las empresas apuestan por dicha gestión de archivos para que sea mayor el número de grupos de trabajo que tengan acceso a los mismos datos. Un servidor de archivos contrarresta los conflictos originados por las diferentes versiones de archivos locales y hace posible tanto la creación automática de las diferentes versiones de datos como la realización de una copia de seguridad central de la totalidad de datos de la empresa. En el acceso al servidor de

archivos por medio de Internet entran en juego protocolos de transmisión como FTP (File Transfer Protocol), SFTP (Secure File Transfer Protocol), FTPS (FTP over SSL) o SCP (Secure Copy). Los protocolos SMB (Server Message Block) y NFS (Network File System) se encuentran habitualmente en las redes de área locales (LAN).

Servidor de correo electrónico: un servidor de correo electrónico consta de varios módulos de software cuya interacción hace posible la recepción, el envío y el reenvío de correos electrónicos, así como su puesta a punto para que estén disponibles. Por regla general funciona mediante el protocolo de transferencia simple de correo (SMTP). Los usuarios que quieran acceder a un servidor de correo electrónico necesitan un cliente de correo electrónico que recoja los mensajes del servidor y los entregue en la bandeja de entrada, proceso que tiene lugar a través de los protocolos IMAP (Internet Message Access Protocol) o POP (Post Office Protocol).
Servidor de base de datos: un servidor de base de datos es un programa informático que posibilita que otros programas puedan acceder a uno o varios sistemas de bases de datos a través de una red. Las soluciones de software con una elevada cuota de mercado son Oracle, MySQL, Microsoft SQL Server, PostgreSQL y DB2. Los servidores de bases de datos ayudan a los servidores web, por regla general, a la hora de almacenar y entregar datos.

Servidor de juegos: los servidores de juegos son servidores (software) creados específicamente para juegos multijugador online. Estos servidores gestionan los datos del juego online y permiten la interacción sincrónica con el mundo virtual. La base de hardware de un servidor de juegos se encuentra en el centro de datos de los proveedores especializados o está disponible en una red doméstica local.

Servidor proxy: el servidor proxy sirve como interfaz de comunicación en las redes informáticas. En su papel de intermediario, el servidor proxy recibe las solicitudes de red y las transmite a través de su propia dirección IP. Los servidores proxy se usan para filtrar la comunicación, para controlar el ancho de banda, para aumentar la disponibilidad a través del reparto de cargas, así como para guardar datos temporalmente (caching). Además, los servidores proxy permiten una amplia anonimización, ya que la dirección IP del cliente queda oculta en el proxy.

Servidor DNS: el servidor DNS o servidor de nombres permite la resolución de nombres en una red. Los servidores DNS son de vital importancia para la red informática mundial (WWW), ya que traducen los nombres de host como www.example.com en la correspondiente dirección IP. Si quieres saber más sobre los servidores de nombres y sobre el sistema de nombres de dominio (DNS), visita nuestra guía digital.

En teoría, un único dispositivo físico puede alojar diferentes tipos de servidores. Sin embargo, es habitual alojar cada uno de los servidores en un ordenador independiente o que estos se repartan en más de un ordenador. De esta manera, se evita que la utilización del hardware de un servicio repercuta en el rendimiento de otros servicios.

¿En qué consiste el alojamiento de servidores?
Mientras que a las grandes empresas les sale rentable la adquisición de hardware de servidores, los autónomos y los particulares que quieren desarrollar proyectos en un servidor propio recurren normalmente al alquiler. Los proveedores especializados ofrecen diferentes modelos de servidores de alquiler en los que los usuarios no tienen que preocuparse por el funcionamiento de la máquina física. La gama de productos abarca desde servidores dedicados cuyos componentes de hardware se ponen a disposición de

los usuarios de manera exclusiva, hasta servicios de hosting compartido para alojar a varios clientes virtuales en una base de hardware común. Para obtener más información, visita nuestra guía sobre las ventajas y los inconvenientes de los diferentes modelos de alojamiento.

Funcionamiento de un servidor Web

Un servidor es un dispositivo virtual que le brinda espacio y estructura a los sitios web para que almacenen sus datos y manejen sus páginas.

¿Alguna vez te has preguntado dónde están localizados todos los contenidos de tu estrategia de marketing digital que subes a tu página web?

Es, pues, en los servidores web. Estos son fundamentales para Internet y si ellos no existieran, navegar por la web sería muy diferente a lo que conocemos hoy en día.

Con la transformación digital y el creciente uso de Internet era más que necesario crear servidores que fueran capaces de almacenar y emitir la biblioteca de información (casi infinita) que encontramos en la web.

¿Qué es un servidor?
Un servidor web (server) es un ordenador de gran potencia que se encarga de "prestar el servicio" de transmitir la información pedida por sus clientes (otros ordenadores, dispositivos móviles, impresoras, personas, etc.)

Los servidores web (web server) son un componente de los servidores que tienen como principal función almacenar, en web hosting, todos los archivos propios de una página web (imágenes, textos, videos, etc.) y transmitirlos a los usuarios a través de los navegadores mediante el protocolo HTTP (Hipertext Transfer Protocol).

¿Para qué sirve un servidor web en Internet?
El rol principal de un servidor web es almacenar y transmitir el contenido solicitado de un sitio web al navegador del usuario.

Este proceso, para los internautas no dura más que un segundo, sin embargo, a nivel del web server es una secuencia más complicada de lo que parece.

Para cumplir con sus funciones el servidor deberá tener la capacidad de estar siempre encendido para evitar interrumpir el servicio que le ofrece a sus clientes. Si dicho servidor falla o se apaga, los internautas tendrán problemas al ingresar al sitio web.

¿Cómo funciona un servidor web?
La comunicación entre un servidor y sus clientes se basa en HTTP, es decir, en el protocolo de transferencia de hipertexto o en su variante codificada HTTPS.

Para saber cómo funciona, primero es necesario conocer que el web server está permanentemente en espera de una solicitud de información.

Además, ten en cuenta que toda computadora, smartphone o tablet tiene una dirección IP única e irrepetible que lo identifica de otro dispositivo en la red, así es como el servidor web envía la información exacta que el internauta está esperando.

Ahora bien, para que el web server pueda cumplir con su función es necesario que reciba la petición por parte de un navegador, en otras palabras, se envía un pedido desde una dirección IP hacia la dirección IP del servidor que aloja los archivos del sitio en cuestión.

A continuación, el servidor web busca en sus archivos la información que se le está solicitando, procede a interpretar las líneas de código y a enviar el resultado al navegador cuya dirección IP fue la solicitante.

Este resultado se le muestra a los internautas y es lo que siempre sucede cuando se navega en sitios de Internet. Cuando este proceso se completa podemos decir que el web server ha cumplido con su función.

¿Cuáles son las características de un servidor web?
Como características necesarias de un servidor web a nivel de software y hardware, podemos encontrar:

A nivel de software
Sistema Operativo
Se encarga de que el hardware funcione y logre interactuar con los servicios que corre el sistema. Algunos ejemplos son:

Unix,
Linux,
o Windows.
Sistemas de archivos
Es una guía lógica que permite que el sistema pueda ubicar, ordenar y filtrar datos en el disco duro, con el fin de que podamos leerlos, modificarlos o eliminarlos.

Software servidor HTTP
Son los diferentes tipos de servidores web especializados en transmitir el contenido vía web (Apache, Nginx, IIS, Caddy, etc.).

Virtual Hosting
Permite que bajo el mismo web server e IP se alojen en varios sitios web distinto.

Despacho de ficheros estáticos y dinámicos
Los ficheros estáticos brindan soporte para alojar y despachar archivos como:

JPG,
GIF,
PNG,
BMP,
CSS,
TXT,
HTML,
Javascript,
MP3
y MP4.

Los ficheros dinámicos funcionan para información y se basan en archivos que serán procesados por el servidor dependiendo generalmente de un lenguaje de desarrollo, por ejemplo: en PHP, ASP, Python, Ruby y GO.

Monitoreo de Red y Límites
Permite monitorear el tránsito de red, paquetes que entran y salen, así como servicios de sistema y uso de hardware como:

1. el uso del Almacenamiento;
2. consumo de RAM;
3. porcentaje de ocupación del CPU;
4. velocidad de la red;
5. rendimiento de escritura/lectura en disco.

Sistema de seguridad
El sistema de seguridad de un servidor debe:

- imponer límites de acceso por dirección IP;
- denegar o permitirle acceso a ciertos archivos o URLs;
- solicitar usuario y contraseña para autenticación básica HTTP;
- realizar un filtrado de peticiones inseguras;
- dar soporte para despachar información cifrada con certificados de seguridad SSL vía HTTPS.

A nivel de hardware
Rack y gabinete
El rack se refiere al lugar donde se alojan los servidores físicamente y el gabinete es el armazón que sostiene los componentes de hardware de una computadora.

CPU
Es el centro de procesamiento de datos del servidor desde donde se realizan todos los cálculos lógicos y matemáticos para que el usuario pueda manipular y acceder a los datos como necesita.

Memoria RAM
Se utiliza para almacenar información y datos de forma temporal dependiendo de la demanda del usuario a través del sistema operativo.

Unidades de almacenamiento
El almacenamiento de servidores web se hace en discos duros, los cuales permiten almacenar la información del sistema operativo, los servicios de sistema, y en última instancia los datos cargados por el usuario.

Puerto de red
El ancho de banda es el que te permite tener un volumen suficiente para transmitir información de ida y vuelta hacia y desde tu servidor web.

¿Qué tipos de servidores web existen?
Existen muchos tipos de servidores web, conoce cuáles son los servidores web más usados en la actualidad:

Apache
Es el más común y utilizado en el mundo, sin embargo, ha perdido popularidad frente a Microsoft IIS y Nginx.

Entre las ventajas de Apache está que es un código abierto, con software gratuito y multiplataforma, y entre sus desventajas su bajo rendimiento cuando recibe miles de requests (peticiones) simultáneas en procesamiento de contenido dinámico o archivos estáticos.

Nginx

Conocido y popularizado como una de las mejores alternativas de Apache. Nginx es un servidor web de código abierto y gratuito (aunque también existe una versión comercial) que se destaca por su alto rendimiento.

Entre sus beneficios resalta una configuración simple, ligera, rápida y excelente en cuanto a seguridad y rendimiento, además permite ser configurado para integrarse nativamente con casi cualquier tecnología y lenguaje de programación moderno.

Como desventajas podemos encontrar que no soporta los archivos .htaccess (de Apache), aunque incluye su propio lenguaje de rewrites.

LiteSpeed
Es un software de despacho HTTP desarrollado por LiteSpeedTech, existe una versión de código abierto (open source) y una versión comercial que incluye diferentes tipos de licencia.

Algunos de los beneficios de LiteSpeed es que soporta grandes cantidades de conexiones simultáneas con un consumo de recursos realmente bajo (incluso con aplicaciones demandantes como las que utilizan PHP) y a nivel de archivos estáticos está a la altura de Nginx.

Microsoft IIS
Internet Information Services o IIS se ha popularizado para ofrecer servicios en la nube, principalmente en Azure (la plataforma de Cloud Hosting de Microsoft).

Además, su perfecta integración con Windows (claro está), Visual Studio y sus herramientas hicieron que este web server se posicionara como el servidor líder en el mundo empresarial.

Otros servidores web conocidos
Lighttpd;
Caddy;
Cherokee;
NodeJS;
Sun Java System Web Server;
Google Web Server (GWS), es el servidor privado de Google por lo que no puede ser dascargado.

Cada servidor web es indicado para ciertas funciones, por lo que escoger cuál vas a usar va a depender de lo que pretendas hacer con ellos.

En resumen:
No olvides que al considerar elegir un web server debes tener en cuenta:

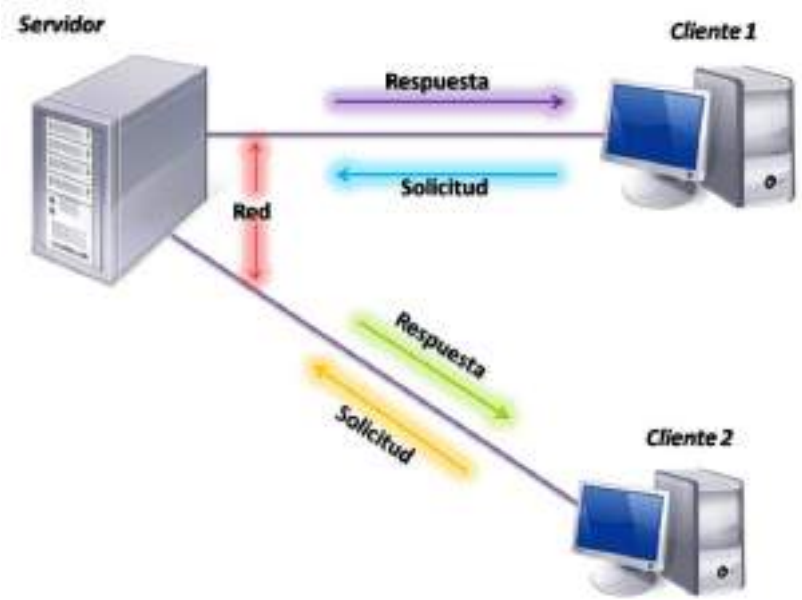
- que tan bien funciona con el sistema operativo y otros servidores;
- su capacidad para manejar la programación del servidor;
- las características de seguridad;
- las herramientas particulares de publicación;
- motor de búsqueda;
- la creación de sitios que vienen con él.

Los servidores Web no solo permiten publicar tu sitio en Internet, sino que también entregan el contenido a tus lectores. Conocer cuál se adapta más a ti y a tus lectores es fundamental para que el web server que elijas cumpla adecuadamente sus funciones.

Arquitectura Cliente-Servidor

PROTOCOLO TCP/IP

Arquitectura cliente/servidor
Según TIC, (2016)La estructura cliente - servidor es una arquitectura de computación en la que se consigue un procesamiento cooperativo de la información por medio de un conjunto de procesadores, de tal forma que uno o varios clientes, distribuidos geográficamente o no, solicitan servicios de computación a uno o más servidores.



De esta forma, y gracias a esta arquitectura, la totalidad de los procesadores, clientes y servidores, trabajan de forma cooperativa para realizar un determinado tratamiento de la información.

Atendiendo a esta visión descentralizada, la arquitectura cliente - servidor consiste en una arquitectura distribuida de computación, en la que las tareas de cómputo se reparten entre distintos procesadores, obteniendo los usuarios finales el resultado final de forma transparente, con independencia del número de equipos (servidores) que han intervenido en el tratamiento. Se puede decir por tanto que la arquitectura cliente - servidor es un tipo de arquitectura distribuida, posiblemente la más extendida.

Elementos que forman parte de una arquitectura cliente - servidor
Un sistema Cliente/Servidor es un Sistema de Información distribuido basado en las siguientes características:

Servicio: unidad básica de diseño. El servidor los proporciona y el cliente los utiliza.

Recursos compartidos: Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos.

Protocolos asimétricos: Los clientes inician "conversaciones". Los servidores esperan su establecimiento pasivamente.

Transparencia de localización física de los servidores y clientes: El cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.

Independencia de la plataforma hardware y/o software que se emplee.

Sistemas débilmente acoplados. Interacción basada en envío de mensajes.

Encapsulamiento de servicios. Los detalles de la implementación de un servicio son transparentes al cliente.

Escalabilidad horizontal (añadir clientes) y vertical (ampliar potencia de los servidores).

Integridad: Datos y programas centralizados en servidores facilitan su integridad y mantenimiento.

En el modelo usual Cliente/Servidor, un servidor, (daemon en la terminología sajona basada en sistemas UNIX/LINUX, traducido como "demonio") se activa y espera las solicitudes de los clientes.

Lo normal es que los servicios de un mismo servidor puedan ser utilizados por múltiples clientes distintos. Tanto los programas cliente como los servidores son con frecuencia parte de un programa o aplicación mayores.

ESQUEMA DE FUNCIONAMIENTO DE UN SISTEMA SEGÚN LA ARQUITECTURA CLIENTE - SERVIDOR.

El Esquema de funcionamiento de un Sistema Cliente/Servidor sería:

1) El cliente solicita una información al servidor.
2) El servidor recibe la petición del cliente.
3) El servidor procesa dicha solicitud.
4) El servidor envía el resultado obtenido al cliente.
5) El cliente recibe el resultado y lo procesa.

COMPONENTES DE LA ARQUITECTURA CLIENTE - SERVIDOR

El modelo Cliente/Servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos.

Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

Nivel de Presentación: Agrupa a todos los elementos asociados al componente Cliente.

Nivel de Aplicación: Agrupa a todos los elementos asociados al componente Servidor.

Nivel de comunicación: Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.

Nivel de base de datos: Agrupa a todas las actividades asociadas al acceso de los datos.

Elementos principales
CLIENTE

Un cliente es todo proceso que reclama servicios de otro. Una definición un poco más elaborada podría ser la siguiente: cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor. Se lo conoce con el término front-end.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

De este modo el cliente se puede clasificar en:

Cliente basado en aplicación de usuario. Si los datos son de baja interacción y están fuertemente relacionados con la actividad de los usuarios de esos clientes.

Cliente basado en lógica de negocio. Toma datos suministrados por el usuario y/o la base de datos y efectúa los cálculos necesarios según los requerimientos del usuario.

SERVIDOR

Un servidor es todo proceso que proporciona un servicio a otros. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las principales funciones que lleva a cabo el proceso servidor se enumeran a continuación:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

MIDDLEWARE

El middleware es un módulo intermedio que actúa como conductor entre sistemas permitiendo a cualquier usuario de sistemas de información comunicarse con varias fuentes de información que se encuentran conectadas por una red. En el caso que nos concierne, es el intermediario entre el cliente y el servidor y se ejecuta en ambas partes.

El middleware se estructura en tres niveles:

- Protocolo de transporte.
- Network Operating System (NOS).
- Protocolo específico del servicio.

Las principales características de un middleware son:

- Simplifica el proceso de desarrollo de aplicaciones al independizar los entornos propietarios.
- Permite la interconectividad de los Sistemas de Información del Organismo.
- Proporciona mayor control del negocio al poder contar con información procedente de distintas plataformas sobre el mismo soporte.
- Facilita el desarrollo de sistemas complejos con diferentes tecnologías y arquitecturas.

COMUNICACIÓN ENTRE LOS ELEMENTOS (NOS)

Como se ha comentado en el apartado anterior, el middleware es un conjunto de aplicaciones encargadas de enlazar al cliente con el servidor. Para ello se estructura en tres capas diferentes:

Protocolo de transporte: comunes a otras aplicaciones.

Network Operating System (NOS).

Protocolo específico del servicio: especiales para distintos tipos de sistemas Cliente/Servidor.

Él NOS es el encargado de proporcionar una apariencia de sistema único a un sistema Cliente/Servidor. Se trata pues, de una extensión del Sistema Operativo:

El cliente realiza una llamada a un servicio como si fuera local.

Él NOS:

- Intercepta la llamada.
- Redirige la llamada al servidor apropiado.
- Devuelve la contestación.

Él NOS debe proporcionar transparencia a los procesos Cliente/Servidor con respecto a:

- **Localización:** Los recursos sólo se conocen por su nombre. El sistema en el que se ejecutan es irrelevante.
- **Espacio de nombres:** Las convenciones de los nombres de los recursos deben ser iguales, independientemente del sistema que los soporte.
- **Conexión:** Un único usuario y contraseña para todo el sistema.
- **Replicación:** No se debe diferenciar entre copias de un mismo recurso.
- **Acceso local / remoto:** El acceso a un recurso se debe realizar como si estuviera localizado en el mismo sistema que el programa cliente.
- **Tiempo**: Los relojes de todos los elementos del sistema deben estar sincronizados.
- **Fallos**: El sistema debe proporcionar servicios de detección de fallos, redundancia y reconexión tras un fallo.
- **Administración**: Un único sistema de gestión de todos los recursos.

- **Protocolos**: Idéntica interfaz de programación para todos los protocolos de transporte.

Tipos de arquitectura cliente / servidor

Uno de los aspectos claves para entender la tecnología Cliente/Servidor, y por tanto contar con la capacidad de proponer y llevar a cabo soluciones de este tipo, es llegar a conocer la arquitectura de este modelo y los conceptos o ideas asociados al mismo.

Un esquema de clasificación basado en los conceptos de Fat Client/Thin Client, Fat Server/Thin Server, es decir, basado en el tamaño de los componentes. En segundo lugar tenemos una clasificación según la naturaleza del servicio que nos ofrecen.

TIPOS DE ARQUITECTURA CLIENTE - SERVIDOR POR TAMAÑO DE COMPONENTES.

Este tipo de clasificación se basa en los grados de libertad que brinda el modelo Cliente/Servidor para balancear la carga de proceso entre los niveles de presentación, aplicación y base de datos.

Dependiendo de qué segmento de las capas de software tenga que soportar la mayor o menor carga de procesamiento, se habla de Fat Client (Thin Server) o Fat server (Thin Client).

Consideraciones de este tipo son importantes en el momento de decidir una plataforma de desarrollo, al mismo tiempo que pueden definir la viabilidad o no de las mismas para enfrentar un cierto número de restricciones impuestas por una problemática a resolver.

FAT CLIENT (THIN SERVER)

En este esquema de arquitectura el peso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un administrador de base de datos.

En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System), y como se concluirá más adelante, tiene pocas posibilidades de aplicarse en sistemas de misión crítica.

FAT SERVER (THIN CLIENT)

Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el peso de la aplicación corre por el lado del servidor de aplicación.

En general este tipo de arquitectura presenta una flexibilidad mayor para desarrollar una gran variedad de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

TIPOS DE ARQUITECTURA CLIENTE - SERVIDOR SEGÚN

LA NATURALEZA DE SERVICIO PROPORCIONADO.

SERVIDORES DE FICHEROS

Con un servidor de archivos, un cliente lo que hace es requerimientos de los mismos sobre una red. Esta es una forma muy primitiva de servicios de datos, la cual necesita intercambio de muchos mensajes sobre una red para hallar el dato requerido.

Los servidores de archivos usan recursos compartidos sobre la red y son necesarios para crear repositorios de documentos, imágenes y archivos grandes sobre la red.

SERVIDORES DE BASES DE DATOS

Este análisis está elaborado desde el punto de vista del modelo Cliente/Servidor, y está directamente relacionado con la arquitectura en dos planos, que se describirá en el apartado siguiente.

Obviamente la creación de aplicaciones Cliente/Servidor está asociada a la utilización de servidores de bases de datos relacionales SQL, y dependiendo de los requerimientos y restricciones se debe elegir entre una arquitectura dos o tres planos.

SERVIDORES DE TRANSACCIONES

Estos tipos de sistemas se pueden implementar con cualquiera de las modalidades Cliente/Servidor en dos o tres planos, pero incorporan un elemento principal sobre el cual se elabora y basa toda la fortaleza de este modelo, el concepto de transacción.

Con un servidor de transacciones el proceso cliente llama a funciones, procedimientos o métodos que residen en el servidor, ya sea que se trate de un servidor de bases de datos o un servidor de aplicaciones.

SERVIDORES DE OBJETOS

Con un servidor de objetos, las aplicaciones Cliente/Servidor son escritas como un conjunto de objetos que se comunican. Los objetos cliente se comunican con los objetos servidores usando un Object Request Broker (ORB). El cliente invoca un método de un objeto remoto. El ORB localiza el método del objeto en el servidor, y lo ejecuta para devolver el resultado al objeto cliente.

Los servidores de objetos deben soportar concurrencia. La parte central de la comunicación en los servidores de objetos es el ORB:

- Elemento central y principal de esta arquitectura.
- Bus de objetos. Permite la comuniación entre ellos.
- Middleware avanzado: Permite llamadas estáticas y dinámicas a objetos.
- Lenguaje de descripción de interfaces independiente del lenguaje de programación.

SERVIDORES WEB

La primera aplicación cliente servidor que cubre todo el planeta es el World Wide Web. Este nuevo modelo consiste en clientes simples que hablan con servidores Web. Un servidor Web devuelve documentos cuando el cliente pregunta por el nombre de los mismos. Los clientes y los servidores se comunican usando un protocolo basado en RPC,

llamado HTTP. Este protocolo define un conjunto simple de comandos, los parámetros son pasados como cadenas y no provee tipos de datos. La Web y los objetos distribuidos están comenzando a crear un conjunto muy interactivo de computación Cliente/Servidor.

Modelos cliente/servidor
Una de las clasificaciones mejor conocidas de las arquitecturas Cliente/Servidor se basa en la idea de planos (tier), la cual es una variación sobre la división o clasificación por tamaño de componentes.

A nivel de software
Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura Cliente/Servidor y en la madurez y popularidad de la computación distribuida

MODELO CLIENTE/SERVIDOR 2 CAPAS

Esta estructura se caracteriza por la conexión directa entre el proceso cliente y un administrador de bases de datos. Dependiendo de donde se localice el grupo de tareas correspondientes a la lógica de negocios se pueden tener a su vez dos tipos distintos dentro de esta misma categoría:

IMPLEMENTADO CON SQL REMOTO

En este esquema el cliente envía mensajes con solicitudes SQL al servidor de bases de datos y el resultado de cada instrucción SQL es devuelto por la red, no importando si son uno, diez, cien o mil registros. Es el mismo cliente quien debe procesar todos los registros que le fueron devueltos por el servidor de base de datos, según el requerimiento que él mismo hizo.

Ventajas:

Presenta una estructura de desarrollo bastante simple ya que el programador maneja un único ambiente de desarrollo (es más simple respecto al Cliente/Servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

Inconvenientes:

La gran cantidad de información que viaja al cliente congestiona demasiado el tráfico de red, lo que se traduce en bajo rendimiento.

Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.

IMPLEMENTADO CON PROCEDIMIENTOS ALMACENADOS

En este esquema el cliente envía llamadas a funciones que residen en la base de datos, y es ésta quien resuelve y procesa la totalidad de las instrucciones SQL agrupadas en la mencionada función.

Ventajas: Presenta las mismas ventajas de una arquitectura dos planos con procedimientos almacenados, pero mejora considerablemente el rendimiento sobre ésta,

dado que reduce el tráfico por la red al procesar los datos en la misma base de datos, haciendo viajar sólo el resultado final de un conjunto de instrucciones SQL.

Inconvenientes: Si bien la complejidad de desarrollo se ve disminuida, se pierde flexibilidad y escalabilidad en las soluciones implantadas. Obliga a basar el peso de la aplicación en SQL extendido, propios del proveedor de la base de datos que se elija. Debiera considerarse que sí bien los procedimientos almacenados (stored procedures), los desencadenantes (triggers) y las reglas (constraint) son útiles, en rigor son ajenos al estándar de SQL

MODELO CLIENTE/SERVIDOR 3 CAPAS

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

Ventajas:

Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto a la estructura en dos planos).

Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones. Provee escalabilidad horizontal y vertical.

Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.

Inconvenientes:

Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con Cliente/Servidor dos planos.

Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo Cliente/Servidor dos planos, y normalmente son de alto costo.

A nivel de hardware
Esta clasificación del modelo Cliente/Servidor se basa igualmente en la distribución de los procesos y elementos entre sus componentes, pero centrándose en la parte física del mismo, en el que la administración de la interfaz gráfica se asocia a los clientes PC y la seguridad e integridad de los datos quedan asociados a ambientes mainframe o por lo menos a servidores locales y/o centrales.

MODELO CLIENTE / SERVIDOR 2 CAPAS

Los clientes son conectados vía LAN a un servidor de aplicaciones local, el cual, dependiendo de la aplicación puede dar acceso a los datos administrados por él.

MODELO CLIENTE / SERVIDOR 3 CAPAS

Los clientes son conectados vía LAN a un servidor de aplicaciones local, el cual a su vez se comunica con un servidor central de bases de datos. El servidor local tiene un comportamiento dual, dado que actúa como cliente o servidor en función de la dirección de la comunicación.

Tipos de Servidores según el servicio realizado FTP - Correo - Web - IMAP - Streaming

Un servidor o server, en el lenguaje informático, es un ordenador y sus programas, que están al servicio de otros ordenadores.

El servidor atiende y responde a las peticiones que le hacen los otros ordenadores. Los otros ordenadores, que le hacen peticiones, serán los "clientes" del servidor.

Precisamente se llaman servidores porque sirven cosas y están al servicio de otros ordenadores.

Por ejemplo si tienes un correo electrónico, lo recibes de un servidor de correo electrónico, si deseas ver una página web, la recibes de un servidor web, si trabajas en una red de ordenadores todos los servicios compartidos de la red estarán en un servidor de red y así otros muchos servicios y tipos de servidores que veremos.

El modelo o arquitectura que siguen los servidores es el de cliente-servidor, es decir el cliente/s pide y el servidor proporciona los recursos o servicios.

Los servidores se utilizan para gestionar los recursos de una red.

Un servidor deberá estar siempre encendido, ya que si se apaga dejará de dar servicio a los demás. Cuando un servidor falla (se apaga o tiene errores) hace que los demás usuarios de la red tengan problemas, porque no disponen de los servicios que proporciona ese servidor.

Por ejemplo, un usuario puede configurar un servidor para controlar el acceso a una red, enviar/recibir correo electrónico, gestionar los trabajos de impresión, o alojar un sitio web.

La red más conocida y más grande es Internet, y está llena de servidores. Pero ojo hay servidores dentro de redes pequeñas y particulares, incluso tu puedes hacer que tu propio ordenador sea un servidor.

Dependiendo del servicio que de el servidor, tiene que disponer de software (programas) específicos capaces de ofrecer esos servicios. El hardware es simplemente un ordenador, aunque es recomendable que sea de gama alta, para dar respuesta a las peticiones lo más rápido posible.

En la siguiente imagen vemos el apilamiento de los servidores de una empresa que se dedica a proporcionar almacenamiento de información, también llamado Cloud Computing.

Servidores de internet

Normalmente, la mayoría de los servidores están diseñados para operar sin ninguna intervención manual durante su funcionamiento. Eso sí, antes se deberán configurar correctamente.
Tipos de Servidores
Vamos a ver los principales tipos de servidores y explicar para qué sirve cada uno:

- **Servidor de Correo Electrónico o Mail Server**: Es un ordenador dentro de una red que funciona como una oficina de correo virtual. Transfiere y almacena los mensajes de correo electrónico a través de una red.

Estos servidores tienen programas capaces de almacenar correos para los usuarios locales y con un conjunto de reglas definidas por el usuario que determinan cómo el servidor de correo debe reaccionar ante el destino de un mensaje específico.

Normalmente estos servidores se dividen en otros 2 diferentes, una para el correo entrante (llamados POP3) y otro para el correo saliente (llamados SMTP).

Los servidores POP3 retienen los mensajes de correo electrónico entrantes hasta que el usuario compruebe su correo y entonces los transfieren al equipo cuando el usuario lo pide.

Los servidores SMTP administran el envío de los mensajes de correo electrónico a Internet. El servidor SMTP administra el correo electrónico saliente y se utiliza en combinación con un servidor POP3 o IMAP de correo electrónico entrante. Cuando el usuario da la orden de enviar, el servidor lo envía.

Otro tipo de servidores de correo son los IMAP que permiten trabajar con los mensajes de correo electrónico sin necesidad de descargarlos antes al equipo. Puedes obtener una vista previa, eliminar y organizar los mensajes directamente en el servidor de correo sin descargarlos en tu equipo. Ejemplos son los correos de yahoo, Hotmail, etc.

También están los servidores Fax que hacen lo mismo que los de correo, pero para la recepción y transmisión de faxes.

- Servidor FTP: Se trata de uno de los más antiguos en Internet, "file transfer protocol" o en Español Protocolo Para la Transferencia de Archivos. Se utilizan para realizar una transferencia segura de archivos entre ordenadores (envío de archivos de un sitio a otro). Los FTP garantiza la seguridad de los archivos y control de su transferencia.

En este caso el cliente 1 envía una petición al servidor FTP para que le envíe un archivo al cliente 2. El servidor se lo envía y el cliente 2 lo recibe. Todo este proceso se realiza mediante un programa llamado FTP instalado en el cliente 1 y en el 2. El servidor dispondrá de otro programa (software) que se encargará de la recepción y el envío.

Este tipo de servidores se utilizan para subir archivos de páginas web a los servidores web, archivos de imágenes, videos, para hacer backup (copias de seguridad), etc.

- Web Server o Servidor Web: Todas las páginas web que puedes ver por internet están almacenadas en servidores, llamados servidores web.

Un servidor web almacena los archivos de una web y los proporciona a los clientes que los solicitan haciendo la transferencia de los archivos a través de la red mediante los navegadores. El cliente lo pide a través de su navegador y el servidor web lo envía al mismo navegador del cliente pare que este lo pueda visualizar.

Los archivos web incluyen texto, imágenes, videos, etc.. y que solo los navegadores pueden visualizar.

El servidor "sirve" (envía) el archivo web (por ejemplo una web en formato html) al navegador del cliente para que lo pueda visualizar. El servidor, el navegador y la comunicación a través de la red seguirán unas normas llamadas "protocolo HTTP".

El espacio que te dejan estos servidores para alojar tu web se llama Hosting. Hay dos tipos principales de hosting:

Hosting Compartido: en el servidor web hay varias páginas alojadas de distintos clientes.

Hosting Dedicado: tienes un servidor para ti solito donde puedes alojar tus webs. Lógicamente son más caros.

Muchas veces se dice servidor web compartido o dedicado para hacer referencia a este tipo de hosting.

Los servidores web utilizan programas específicos para administrar sus servicios. En función del programa que utiliza el servidor web para administrar y servir las páginas web pueden ser de varios tipo. Todos los tipos que vamos a ver a continuación son realmente programas de gestión del servidor web (software).

Tipos de Servidores Web

- Servidor Apache HTTP: Este es el servidor web más popular del mundo desarrollado por la Apache Software Foundation. El servidor web Apache es un software de código abierto y se puede instalar en casi todos los sistemas operativos incluyendo Linux, Unix, Windows, FreeBSD, Mac OS X y más. Alrededor del 60% de los ordenadores usados como servidor web ejecuta el servidor Web Apache.

- Microsoft IIS es un Servidor Web de alto rendimiento de Microsoft. Este servidor Web se ejecuta en plataformas Windows NT / 2000 y 2003 (y en la próximas nuevas versiones de Windows también). IIS viene incluido con Windows NT / 2000 y 2003; Dado que IIS está estrechamente integrado con el sistema operativo, es relativamente fácil administrarlo.

- El Lighttpd , pronunciado lighty es también un servidor web gratuito que se distribuye con el sistema operativo FreeBSD. Este servidor web de código abierto es rápido, seguro y consume mucha menos energía de la CPU. Lighttpd también se puede ejecutar en los sistemas operativos Windows, Mac OS X, Linux y Solaris.

- Sun Java System Web Server es un servidor web adecuado para grandes sitios web de medianas y grandes empresas. Aunque el servidor es libre no es de código abierto. Sin embargo, se ejecuta en plataformas Windows, Linux y Unix. El servidor web de Sun Java System soporta varios idiomas, guiones y tecnologías necesarias para la Web 2.0, tales como JSP, servlets Java, PHP, Perl, Python, Ruby on Rails, ASP y ColdFusion, etc.

- Jigsaw (Servidor de W3C) proviene del World Wide Web Consortium. Es de código abierto y libre y puede ejecutarse en varias plataformas como Linux, Unix, Windows, Mac OS X Free BSD, etc. Jigsaw ha sido escrito en Java y se puede ejecutar scripts CGI y programas PHP.

- El servidor Ngnix es un servidor Web muy ligero y trabaja sobre sistemas Unix y Windows. Se ha convertido en el 4º servidor HTTP más popular de la red y también se distribuye bajo licencia BSD. Se utiliza en el 19% de los servidores web.

Dentro de la red de internet hay unos servidores que se llaman DNS que son los que se encargan de gestionar los nombres de los dominios de las páginas web (las direcciones de las webs). Estos servidores se llaman Servidores DNS. Para saber más sobre esto visita el siguiente enlace: ¿Qué es el DNS y servidores DNS?.

- Servidores Proxy o Servidores de Red: Se utilizan para administrar una red de ordenadores, permitiendo el acceso o no a la red de los clientes. Suelen incluir protección de la red como por ejemplo un firewall (cortafuegos).

- Servidores de Bases de Datos: Son ordenadores preparados para alojar bases de datos para ser utilizadas por uno o más clientes. Además estos servidores realizan tareas como el análisis de los datos, el almacenamiento, la manipulación de datos, y otras tareas específicas.

- Servidores de Audio/Video: Permiten transmitir contenido multimedia en streaming. El streaming es una técnica de envío continuo de información, que permite por ejemplo, ir viendo una película según se va descargando, sin necesidad de descargarla por completo para visualizarla.

- Chat Server o Servidor Chat: Es un equipo dedicado a manejar y mantener un chat y sus usuarios. Los más famosos son los IRC. Ahora también se les conoce como servidores en tiempo real, porque permiten intercambiar información de forma instantánea.

- Servidores Groupware: Son servidores que facilitan el trabajo en grupo de varios ordenadores, con un objetivo común (por ejemplo un proyecto).

Estos servidores disponen de software que permite colaborar a los usuarios del servidor independientemente de donde están ubicados, permitiéndoles así hacer un trabajo colaborativo.

Los archivos y datos almacenados en un servidor groupware pueden ser alterados, acceder y recuperados por los miembros del grupo de trabajo. Groupware también se conoce como software de colaboración.

- Servidor Telnet: Son servidores que nos permiten iniciar sesión en cualquier ordenador y realizar tareas en otro ordenador. Podemos trabajar con nuestro ordenador de forma remota, es decir desde otro ordenador.

- Servidor SIP: Se encargan de gestionar el establecimiento de las llamadas telefónicas por internet. Los SIP almacenan la dirección IP donde deben acceder para realizar la comunicación con un usuario. No transmite ni audio ni video, solo establece la comunicación.

- List Server o Servidores Lista: Permite gestionar listas de correos.

- Servidores Cloud: Realmente estos servidores lo único que hacen es dejarte o alquilarte un espacio del servidor. La mayoría se utilizan para almacenar grandes

cantidades de información en el servidor y tenerla protegida fuera de nuestro ordenador. Muchas empresas alquilan servidores cloud (en la nube) para tener en ellos toda la valiosa información de la empresa, utilizándose cuando quieran y realizando el propio servidor copias de seguridad.

- Cluster de Servidores: Un clúster de servidores es la agrupación de varios servidores dedicados a la misma tarea, Hay veces que un solo servidor se queda pequeño para toda la demanda de los clientes y es necesario más. En estos casos se agrupan en lo que se conoce como Cluster de Servidores.