FrontEnd y BackEnd

Introducción

El **Frontend** son aquellas tecnologías de desarrollo ***web del lado del cliente***, es decir, las que corren en el navegador del usuario y que son básicamente tres: *HTML, CSS y JavaScript*. El frontend *se enfoca en el usuario*, en todo con lo que puede interactuar y lo que ve mientras navega. Una buena experiencia de usuario, inmersión y usabilidad son algunos de los objetivos que busca un buen desarrollador frontend, y hoy en día hay una gran variedad de **Frameworks**, preprocesadores y librerías que ayudan en esta tarea.

El **Backend** es aquello que ***se encuentra del lado del servidor*** y se encarga de *interactuar con bases de datos, verificar maniobras de sesiones de usuarios, montar la página en un servidor y servir todas las vistas creadas por el desarrollador frontend*. En este caso el número de tecnologías es mucho menos limitado, puesto que la programación backend puede alcanzar *lenguajes como PHP, Python, .NET, Java, etc*., y las bases de datos sobre las que se trabaja pueden ser SQL, MongoDB, MySQL, entre otras.

La idea de esta abstracción es mantener separadas las diferentes partes de un sistema web o software para tener un mejor control. En pocas palabras, el objetivo es que el frontend recoja los datos y el backend los procese.

Estas dos capas que forman una aplicación web son independientes entre sí (no comparten código), pero intercambian información. Esta división permite que el acceso a las bases de datos solo se haga desde el backend y el usuario no tenga acceso al código de la aplicación, mientras que la programación del lado del cliente permite que el navegador pueda, por ejemplo, controlar dónde el usuario hace clic o acceder a sus ficheros.

Con esta separación de entornos el usuario de una aplicación web lo que hace es, por ejemplo, iniciar sesión escribiendo su usuario y contraseña en un formulario; a continuación, los datos se envían y el backend toma esta información que viene desde el HTML y busca las coincidencias de usuario en la base de datos con una serie de procesos invisibles para el usuario. En este punto, el servidor mandaría un mensaje al frontend dándole acceso (o no) a la aplicación.

Lenguajes Web Frontend

A pesar de que hay varios lenguajes que se usan en el frontend, nosotros nos basaremos en tres, HTML, CSS y JavaScript, aunque HTML y CSS no son lenguajes de programación, no se debe confundir lenguajes de programación como

ejemplo JavaScript, ActionScript o Java, con lenguajes de marcado como HTML o lenguaje de hojas de estilo cómo CSS. También existen otros lenguajes frontend, como por ejemplo ActionScript, Java, Silverlight, VBScript u otros lenguajes XML, pero se usan poco en comparación con HTML, CSS y JavaScript.

HTML es un lenguaje de marcado de los contenidos de un sitio web, es decir, para designar la función de cada elemento dentro de la página: titulares, párrafos, listas, tablas, etc. Es el esqueleto de la web y la base de todo el frontend.
CSS es un lenguaje de hojas de estilo creado para controlar la presentación de la página definiendo colores, tamaños, tipos de letras, posiciones, espaciados, etc.
JavaScript es un lenguaje de programación interpretado que se encarga del comportamiento de una página web y de la interactividad con el usuario.
Aparte, junto al cliente también tenemos los frameworks, las librerías, los preprocesadores, los plugins... pero todo gira alrededor de HTML, CSS y JavaScript.

Lenguajes Web Backend

Aquí encontramos unos cuantos, por ejemplo, PHP, Python, Rails, Go, C#, Java, NodeJS (JavaScript) entre otros. Como vemos, mientras que para el frontend se acostumbra a trabajar solo con tres lenguajes, en el backend hay unos cuantos más. Por suerte, un desarrollador backend no necesita saberlos todos.
Quizás habréis notado que tenemos JavaScript tanto por el lado del cliente como por el lado del servidor. JavaScript se creó originalmente como lenguaje para el frontend, pero los últimos años se ha creado su lugar dentro del backend con NodeJS, un motor que interpreta JavaScript en el servidor sin necesidad de un navegador. Esto no quiere decir que el JavaScript que tenemos en el cliente tenga alguna conexión con el que se encuentra en el servidor: cada uno corre por su parte de manera independiente. El JavaScript del cliente corre en el navegador y no tiene ningún enlace ni ninguna conexión con el que hay en el servidor y no le interesa saber cómo está montada la arquitectura del servidor ni cómo se conecta a la base de datos.
Ahora se puede utilizar el mismo lenguaje en todos los contextos del desarrollo: JavaScript en el cliente de escritorio (DOM), en el cliente móvil (Cordova, React Native), en el servidor (Node.js) o en la BBDD (MongoDB). La posibilidad de trabajar frontend y backend con un mismo lenguaje desde el punto de vista del desarrollador es muy cómoda, especialmente para aquellos que trabajan ambos mundos.
En cuanto a la tecnología, las herramientas que se utilizan en el backend son: editores de código, compiladores, algunos depuradores para revisar errores y seguridad, gestores de bases de datos y algunas otras cosas.
Uno de los stacks o pila de tecnologías más utilizado por los desarrolladores es el que se conoce por **LAMP**: Linux, Apache, MySQL y PHP. Cualquier web hecha con

Wordpress, Drupal o Prestashop, por ejemplo, están hechas sobre estos cuatro pilares.
Pero se pueden hacer las variaciones que se crean convenientes, puesto que muchas de estas tecnologías son intercambiables por otras similares. Por ejemplo NginX en lugar de Apache, PostgreSQL en lugar de MySQL o Ruby on Rails en lugar de PHP.
Otro stack muy utilizado es el llamado **MEAN**, que se compone de MongoDB, Express, Angular y NodeJS. A diferencia del conjunto anterior, esta pila de trabajo busca entregar la mayor cantidad de carga junto al cliente pero requiere una forma muy diferente de pensar las cosas.
También existe un equivalente en Microsoft que sería Windows + Microsoft IIS + .NET + SQL Server.

Desarrolladores Full Stack

Lo más habitual es que los desarrolladores se especialicen bien con frontend o bien con backend, pero hay una tercera especie que son los llamados «Full-Stack», unos programadores con un perfil técnico muy completo que conocen bien tanto lo referente a backend como a frontend, así como la administración de servidores. Tienen un buen conocimiento de todas las áreas del desarrollo y esto les permite construir proyectos complejos por ellos mismos, sin la ayuda de terceras personas. Se trata de un perfil cada vez más demandado y bien remunerado.

Anna Ferry Mestres,
http://cv.uoc.edu/annotation/a9c35c372dcee6e6b92afad6993cd048/620334/PID_00250214/PID_00250214.html

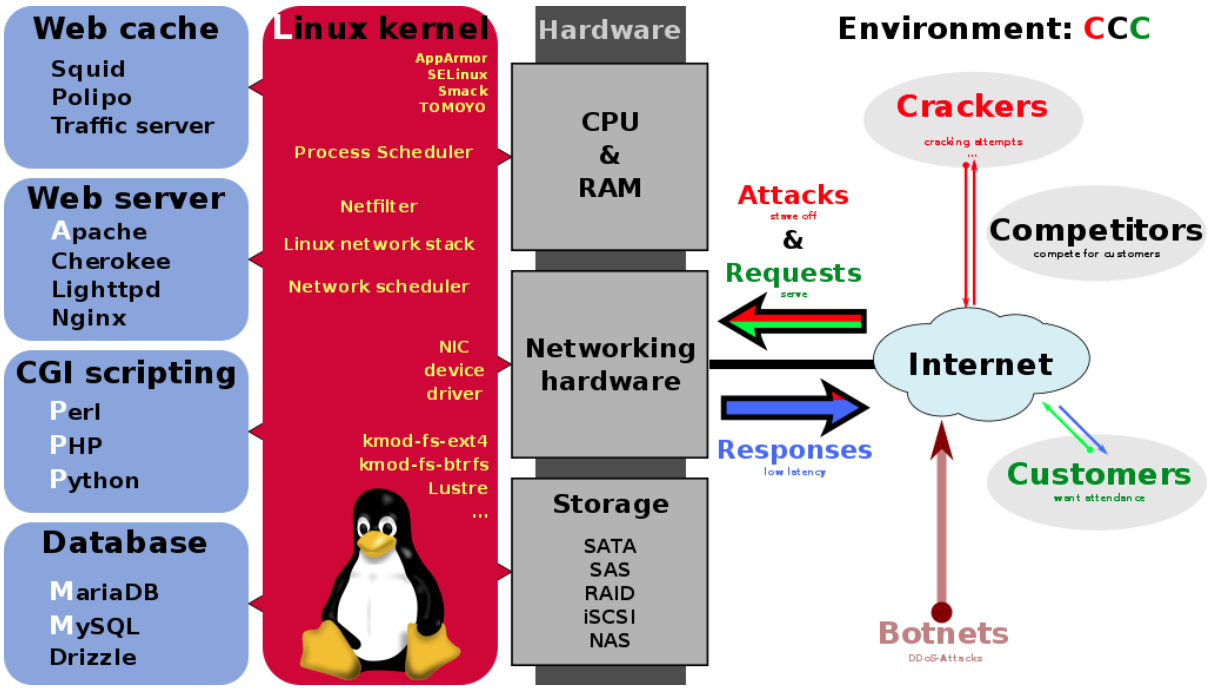
Arquitectura LAMP

LAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- Linux, el sistema operativo.
- Apache, el servidor web.
- MySQL/MariaDB, el gestor de bases de datos.
- PHP, el lenguaje de programación.

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor web, utilizando un paradigma de programación para el desarrollo del sistema.

Desarrollador Full Stack

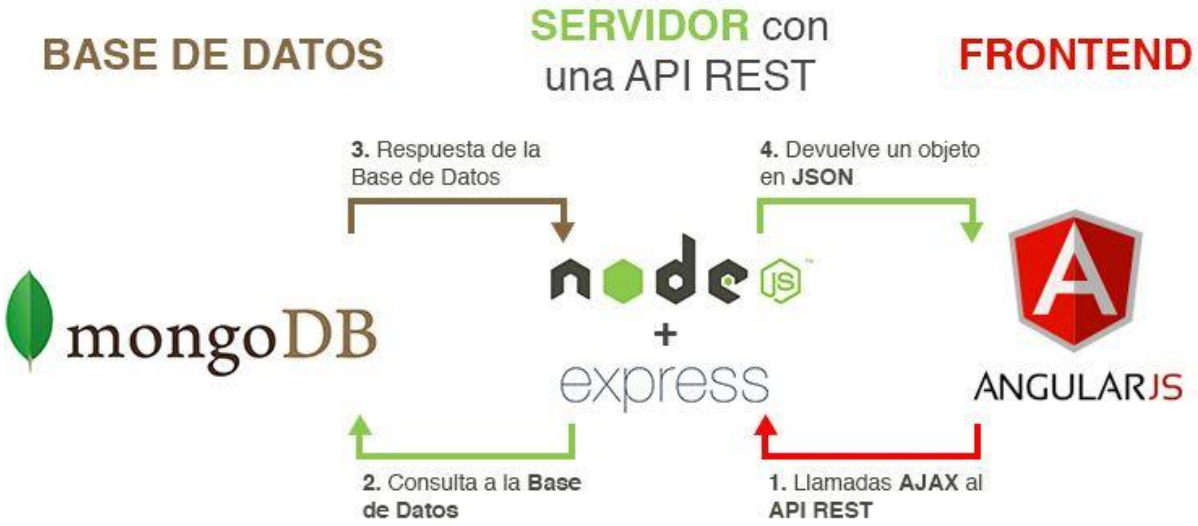


Arquitectura MEAN

MEAN es desarrollo full-stack en JavaScript, es decir, es el conjunto de tecnologías necesarias para el desarrollo de todas las capas de una aplicación web con JavaScript. Está compuesto fundamentalmente por cuatro tecnologías, MongoDB, Express, Angular y Node.js.
Cada subsistema del MEAN stack es de código abierto y de uso gratuito. A continuación se describen estas tecnologías.

- **MongoDB:** es un sistema de base de datos NoSQL, que almacena los datos en estructuras o “documentos”, los cuales están definidos con la notación JSON (Notación simple de objeto tipo JavaScript), lo que permite una rápida manipulación y transferencia de los datos. La mayor característica de esta plataforma es su escalabilidad, lo que significa que puede aumentar en forma considerable la cantidad de datos que almacena sin que esto afecte su funcionamiento en general.
- **ExpressJS:** este framework está escrito en JavaScript para Node.js, es decir es un módulo de NodeJS y como tal funciona sobre esta plataforma; este módulo ofrece los métodos suficientes en JavaScript, para poder manejar las solicitudes o peticiones que se hacen por medio de los métodos del protocolo HTTP (GET, POST, etc.). También ofrece un sistema simple de enrutamiento, que dentro del mean stack es aprovechado en el back-end o en el lado del servidor.

- **AngularJS:** es un framework que facilita la manipulación del DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos), y por lo tanto en el mean stack es la plataforma que se usa para trabajar en el front end. Este framework permite crear una gran variedad de efectos, de una forma sencilla, reduciendo contundentemente la cantidad de código, lo que permite que sea mucho más sencillo de mantener.
- **NodeJS:** es un entorno de ejecución o runtime para JavaScript construido con el motor de JavaScript V8 de Chrome y es la plataforma encargada del funcionamiento del servidor. Funciona totalmente con JavaScript, un lenguaje de programación que en un principio era dedicado a correr en el lado del cliente, pero su uso se ha ampliado considerablemente en todos los aspectos de un sitio web.



Arquitectura BFF

Se puede usar una arquitectura Backend for Frontend (BFF) para crear backends para aplicaciones web o móviles orientadas al cliente. Los BFF pueden ayudar a respaldar una aplicación con múltiples clientes al mismo tiempo que mueven el sistema a un estado menos acoplado que un sistema monolítico. Este code pattern ayuda a los equipos a iterar las funciones más rápido y tener control sobre los backends para aplicaciones móviles sin afectar la experiencia de la aplicación móvil o web correspondiente.

Descripción

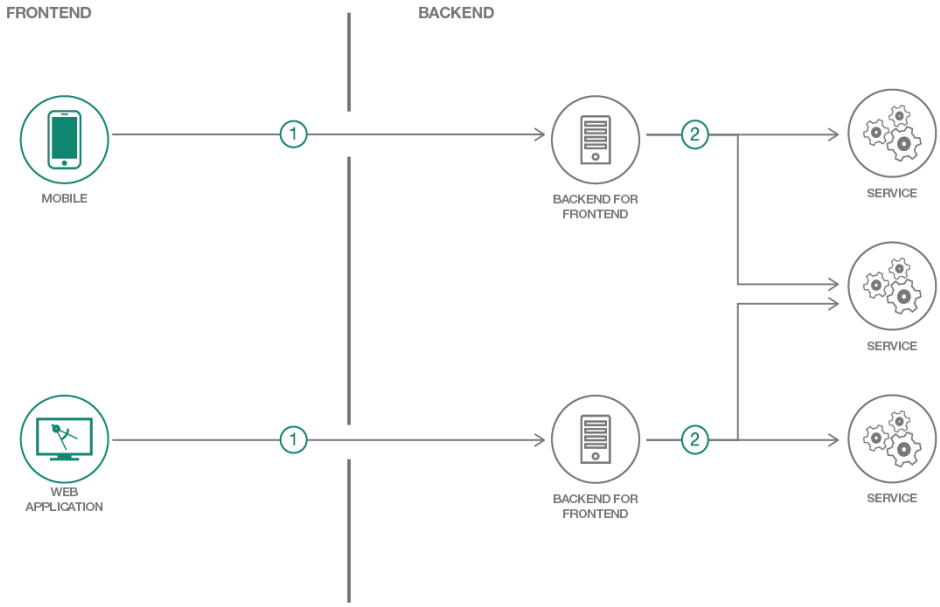
Una arquitectura de microservicio permite a los equipos iterar rápidamente y desarrollar tecnología para escalar rápidamente. La arquitectura Backend for Frontend (BFF) es un tipo de patrón creado con microservicios. El componente principal de este patrón es una aplicación que conecta el frontend de su aplicación con el backend. Este Code Pattern BFF lo ayudará a crear ese componente de acuerdo con las mejores prácticas de IBM.

Este code pattern lo ayudará a:

- Crear el patrón de arquitectura Backend for Frontend (BFF)
- Generar una aplicación en Node.js, Swift o Java
- Generar una aplicación con archivos para implementar en Kubernetes, Cloud Foundry o DevOps Pipeline
- Generar una aplicación con archivos para monitoreo y seguimiento distribuido
- Conectar a servicios aprovisionados

Este code pattern también facilita el seguimiento de un modelo de programación Cloud Native que utiliza las mejores prácticas de IBM para el desarrollo de aplicaciones BFF. Verá cosas como casos de prueba, chequeo de funcionamiento y métricas en cada lenguaje de programación.

Si hace clic en **Desarrollar en IBM Cloud** en la parte superior del Code Pattern, podrá aprovisionar dinámicamente los servicios de Cloud. Esos servicios se inicializarán automáticamente en su aplicación generada. Adicione un servicio administrado de MongoDB, un servicio de Watson o pruebas automáticas en un pipeline de DevOps personalizado.



1. Las plataformas de experiencia del usuario, como Mobile and Web Apps, que pueden ser soportadas en lenguajes como Node.js, Java o Swift, se comunican de su propio servidor backend para el servidor frontend, con el fin de reunir las APIs adecuadas y las solicitudes de servicio necesarias.
2. Cada Backend for Frontend llama a los servicios necesarios que son solicitados por el frontend.

Instrucciones

Vea las instrucciones detalladas en los archivos README:

- Node.js
- Java
- Spring
- Swift

Referencias:

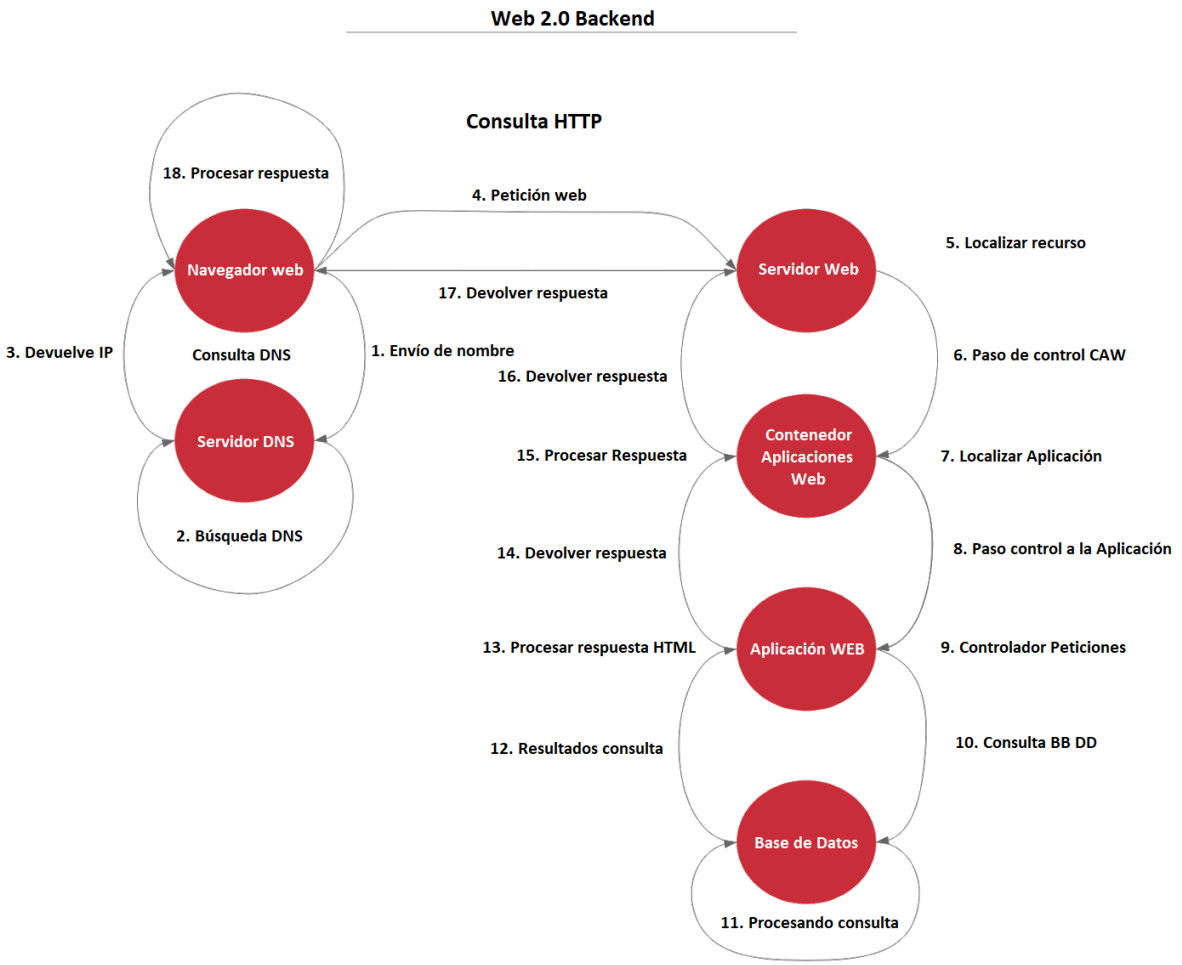
https://nodejs.org/es/docs/
https://jarroba.com/mean-mongo-express-angular-node-ejemplo-de-aplicacion-web-parte-ii/
https://es.wikipedia.org/wiki/MEAN
https://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx

https://es.wikipedia.org/wiki/LAMP

Elementos de la arquitectura Web

- Cliente Web/DNS: Navegador Web. Por ejemplo: Chrome, Safari, Firefox.
- Servidor DNS: Se ocupa de la administración del espacio de nombres de dominio. Es el encargado de conocer la IP asociada a cada nombre de dominio de internet. Por lo tanto, al consultarlo con un nombre de dominio, devuelve la dirección IP asociada a ese nombre de dominio. En Argentina, la oficina encargada de administrar el Espacio de Nombres de Dominio de nivel superior .ar es «.ar - NIC Argentina». Network Information Center Argentina, o NIC Argentina (en español: Centro de Información de la Red para Argentina), es una oficina dependiente de la Secretaría Legal y Técnica de la Presidencia de la Nación bajo la órbita de la Dirección Nacional de Registro de Dominios de Internet, responsable de administrar el dominio de nivel superior .ar, además del registro de nombres de dominio de internet de las personas físicas y jurídicas. https://es.wikipedia.org/wiki/NIC_Argentina
- Servidor Web: Es un programa informático que procesa una aplicación del lado del servidor. Por lo tanto, almacena y procesa la Información Web. Entre distintas tecnologías, se destacan, Apache, Nginx, IIS o Internet Information Services, entre otros.
- Contenedor de aplicaciones Web: módulo o servidor que permite la ejecución de aplicaciones web. Por ejemplo, módulo PHP o Python del Servidor Web. Componente ASP o ASPX de IIS. Servidor o Contenedor de Aplicaciones Web Java: Tomcat, Weblogic, Websphere, JBoss, Geronimo, etc.

Proceso de una Petición Web 2.0



1. Cliente Web: Solicita la resolución de nombres al servidor DNS. Por ejemplo: google.com
2. Servidor DNS: Recibe y trata la solicitud. Una vez recibida la petición realiza las consultas necesarias para resolver y obtener la dirección IP.
3. Servidor DNS: Devuelve al navegador Web la dirección IP que corresponde al Servidor Web.
4. Cliente Web: Conecta con el servidor web mediante la dirección IP y el puerto. Realiza la petición mediante una URL (Método GET) o un formulario (Método POST). Dicha solicitud incluye: la dirección IP del servidor web, el puerto del servidor web, URL y parámetros.
5. Servidor Web: Control de Acceso, Análisis de la petición y localización del recurso. Como detecta que es el acceso a un fichero o ruta de aplicación tiene que traspasar el control al Contenedor de aplicaciones Web
6. Paso de la petición del servidor web al contenedor de aplicaciones web
7. El contenedor analiza la petición y en base a la ruta traspasa el control a la aplicación web.
8. Paso del control de la petición desde el CAW a la aplicación.

9. La aplicación recibe la petición y decide qué hacer en base a ella, es decir, elegir la funcionalidad que se encargará de gestionar esa petición, normalmente en base a la ruta, el método HTTP y los parámetros de entrada por URL. Una vez elegida ejecutará esa funcionalidad.
10. La aplicación realiza una petición SQL a la base de datos.
11. La Base de Datos recibe la petición SQL y la procesa realizando los cambios que tenga que hacer, si corresponde.
12. Una vez procesada la petición devuelve los datos a la aplicación web, normalmente un conjunto de datos. Ej. los 10 últimos clientes.
13. La aplicación web recibe estos datos y tiene que generar una salida, normalmente HTML, donde estructura el contenido de los datos devueltos por la BBDD en etiquetas HTML.
14. La aplicación web devuelve una respuesta al Contenedor de Aplicaciones Web
15. El contenedor procesa la respuesta, para controlar la ejecución de la aplicación por si esta falla.
16. El Contenedor de Aplicaciones Web devuelve el fichero al servidor web.
17. El servidor Web devuelve los datos dentro de la respuesta HTTP al navegador web.
18. Cliente Web: Presenta (renderiza) el contenido HTML resultante.
19. Repite los pasos 4-18 para obtener los ficheros relacionados: CSS, JS, Imágenes, etc.

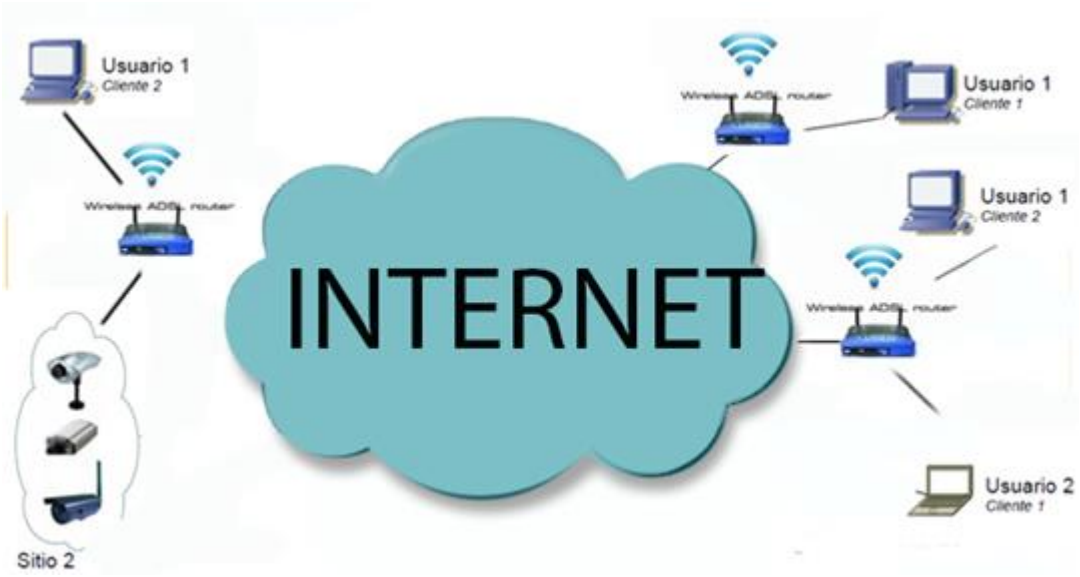
Conceptualización de las estructuras involucradas en la transferencia de datos entre ordenadores

Para entender la programación web es necesario primero conocer o tener alguna noción sobre las estructuras involucradas en la transferencia de datos entre computadoras.

Esta transferencia de datos, es decir, la comunicación entre dos o más computadoras se lleva a cabo a través de lo que se llama comúnmente red de computadoras. La más conocida red es Internet o, como también se la denomina, la red de redes.

Vamos a ver conceptos de red de computadoras, de Internet y cómo funcionan los elementos que las constituyen, a modo introductorio para entender cómo funcionan los sistemas para la Web.

Para comenzar vamos a definir que "*(...) una red de computadoras, también llamada red de ordenadores, red de comunicación de datos o red informática, es un conjunto de equipos nodos y software conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos con la finalidad de compartir información recursos y ofrecer servicios (...)"* (Tanenbaum, 2003)



La finalidad principal para la creación de una red de ordenadores es la de compartir recursos e información a grandes distancias, asegurar la confiabilidad y la disponibilidad de la información (características propias que definen lo que es información), aumentar la velocidad de transmisión de los datos y reducir los costos. (CENS, 2018)

Si bien este no es un curso de redes de computadora es necesario conocer los elementos físicos en los que trabajan los sistemas que vamos a realizar. Con ello queremos decir que si bien podemos abstraernos de esta tecnología es muy útil tener alguna noción de lo que pasa detrás de escena.

La comunicación por medio de una red se lleva a cabo en dos diferentes categorías: una capa denominada física y otra lógica.

La capa física incluye todos los elementos de los que hace uso un equipo para comunicarse con otro equipo dentro de la red, como por ejemplo, tarjetas de red, los cables, las antenas, etc. esto es todo lo que diremos de la capa física si te interesa saber más podés leer autores como Tanenbaum en donde habla de redes y el modelo OSI (modelo de interconexión de sistemas abiertos, protocolos de comunicación, etc.).

Con respecto a la capa lógica la comunicación se rige por normas muy rudimentarias que por sí mismas resultan de escasa utilidad. Sin embargo, haciendo uso de dichas normas es posible construir los denominados protocolos que son normas de comunicación más complejas (de alto nivel) capaces de proporcionar servicios útiles.

Los protocolos son un concepto muy similar al de los idiomas de las personas. Es decir si dos personas hablan el mismo idioma, y respetan ciertas reglas (tales como hablar y escucharse por turnos), es posible comunicarse y transmitir ideas e información. Ese es el modelo de un protocolo.

Para formar una red se requieren elementos de hardware, software y protocolos. los elementos físicos se clasifican en dos grupos: los dispositivos de usuario final (llamados también Hosts) y dispositivos de red. Entre los dispositivos de usuario final podemos enumerar computadoras, impresoras, escáneres, y demás elementos que brindan servicios directamente al usuario. los segundos (dispositivos de red) son todos aquellos que conectan entre sí a los dispositivos de usuario final posibilitando su intercomunicación. (Farías, 2013)

Concluyendo vamos a decir que el fin de una red es interconectar los componentes Hardware de una red, y por tanto, principalmente, los ordenadores individuales también denominados Host, a los equipos que ponen los servicios en la red, los servidores Web, entre otros.

Ahora bien, hemos hablado de la tecnología de lo que es redes de computadoras pero no hemos nombrado a la Internet.

Internet es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen, constituyan una red lógica única de alcance mundial. (Guillamón, 2009)

Uno de los servicios que más éxito ha tenido en Internet ha sido world wide web, www o la web. La www es un conjunto de protocolos que permite de forma sencilla la consulta remota de archivos de hipertexto y utiliza Internet como medio de transmisión («Internet, n.». Oxford English Dictionary (Draft edición). Marzo de 2009).

Para hacer más notable esta distinción vamos a nombrar algunos otros servicios y protocolos en Internet aparte de la web por ejemplo el envío de correo electrónico, protocolo SMTP, transmisión de archivos, protocolo FTP, para nombrar algunos.

Conceptos de HTTP, IP y Protocolos

Bueno ya hemos dejado en claro que significa la palabra protocolo. Por lo tanto, ahora podemos hablar de la familia de protocolos de Internet.
Es necesario tener nociones de estos protocolos ya que al programar aplicaciones para la web tenemos que tener presentes algunas reglas que los caracteriza.
Esta familia de protocolos es un conjunto de protocolos de red en los que se basa Internet y permite la transmisión de datos entre computadoras como hemos dicho.
Entre los principales podemos nombrar el conjunto de protocolos TCP/IP que hace referencia a los dos protocolos más importantes que componen la Internet, que fueron los primeros en definirse y qué son los más utilizados.
Por un lado TCP (protocolo de control de transmisión). Sin entrar en mucho detalle, TCP se usa para crear conexiones entre computadoras a través de las cuales pueden enviarse un flujo de datos. Por la forma en la que está implementado este protocolo los datos serán entregados en su destino sin errores y en el mismo orden en el que se transmitieron. Esto qué quiere decir, que es un protocolo orientado a conexión, ya que el cliente y el servidor deben anunciarse y aceptar la conexión antes de comenzar a transmitir los datos entre ellos. O sea, hay un intercambio de mensajes entre ellos para abrir una línea de conexión que permanece abierta durante toda la comunicación (por eso es orientado a conexión). (dsi.uclm.es, 2007)

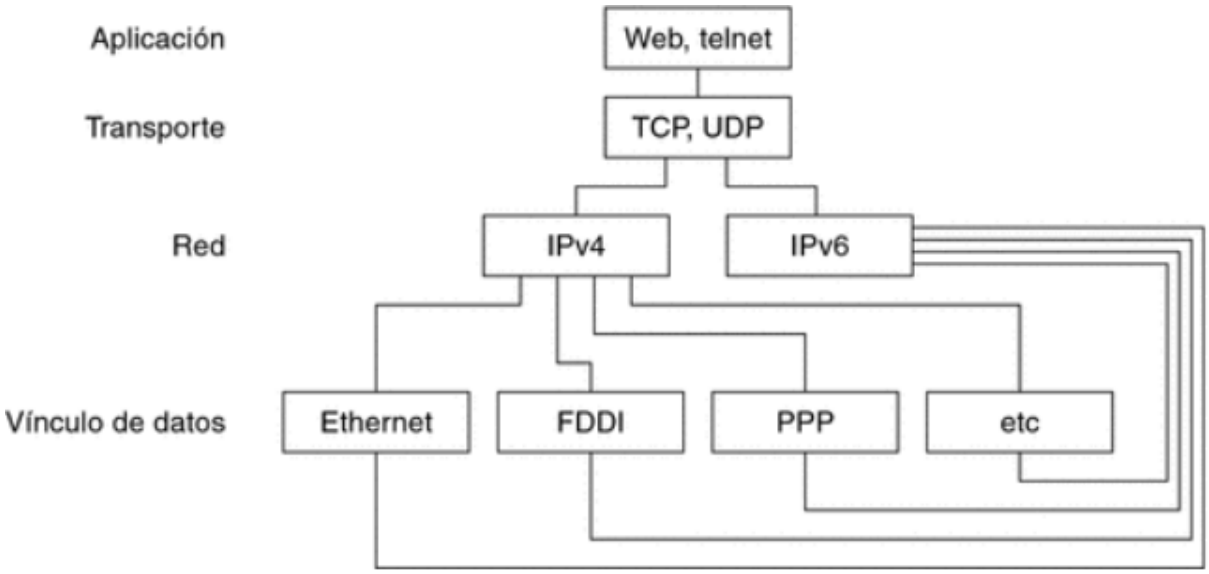


Figura 1: Oracle (10/3/2020) Parte II Administración de TCP/IP. Recuperado de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0sr/index.html

Por otro lado el protocolo IP (Internet Protocol) es un protocolo cuya función principal es el uso direccional en origen o destino de comunicación para transmitir datos mediante un protocolo no orientado a conexión que transfiere paquetes

conmutados a través de distintas redes físicas previamente enlazadas según la norma OSI. (Wikipedia, 2020)

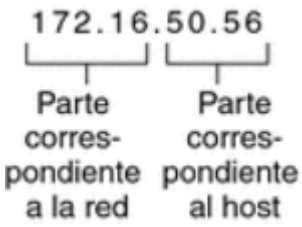


Figura 2: Oracle (10/3/2020) Parte II Administración de TCP/IP. Recuperado de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0sr/index.html

Algo importante del diseño del protocolo IP es que se realizó suponiendo que la entrega de los paquetes de datos sería no confiable por eso se tratara de realizar del mejor modo posible mediante técnicas de enrutamiento sin garantías de alcanzar el destino final pero tratando de buscar la mejor ruta entre las conocidas por la máquina que está usando IP.
Para entender mejor esta distinción entre dos protocolos uno TCP y otro IP habría que analizar el modelo de capas OSI que no vamos a ver. Lo que sí vamos a decir es que hay una jerarquía entre capas y el protocolo IP pertenece a una capa denominada de red que está por encima de una capa denominada de transporte en dónde se encuentra TCP.
Entonces, en conclusión, se utiliza la combinación de estos dos protocolos para la comunicación en Internet, en dónde TCP aporta la fiabilidad entre la comunicación e IP la comunicación entre distintas computadoras ya que las cabeceras de IP (cabecera por ser una parte el protocolo, que aquí no tiene importancia) contienen las direcciones de las máquinas de origen y destino, llamadas direcciones IP. estas direcciones serán usadas por los enrutadores (routers) para decidir el tramo de red por el que se enviarán en los paquetes.
Para entender mejor el funcionamiento de la Internet vamos a decir que dentro de la red de redes que es Internet debe existir un mecanismo para conectar dos computadoras. Este mecanismo lo provee el protocolo de Internet, el cual hace que un paquete de una computadora, llegue a la otra de manera segura a través del protocolo TCP y que llegue a destino a través de las direcciones IP.
Para terminar, una dirección IP es un número que identifica de manera lógica y jerárquica una interfaz de un dispositivo dentro de una red que utiliza el protocolo de internet.
Todas las computadoras en internet tienen una dirección de IP. A modo informativo vamos a decir que existe otro sistema que se denomina sistema de nombres de dominio o DNS que asocia nombres comunes a direcciones IP por ejemplo la dirección www.cba.gov.ar tiene asociado un número de IP correspondiente, pero

este mecanismo existe para que sea más fácil llegar a esa página web sin tener que recordar el número de IP.

Figura 3: Oracle (10/3/2020) Parte II Administración de TCP/IP. Recuperado de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0sr/index.html

Con esto concluimos una introducción de lo que es la comunicación entre computadoras en internet. Si bien parece algo complejo es de mucha importancia conocer estos mecanismos con los que trabajamos.
Por ejemplo si queremos comunicarnos con una base de datos en Internet y nos pide un número de IP ya sabemos de qué se trata. también sabemos que podemos comunicarnos a cualquier computadora conectada en red a través de estos mecanismos y que nos abstraemos de cómo se hace esto. Es decir no nos tenemos que preocupar de manejar errores o interferencias en la comunicación.
Por otro lado, antes de continuar con el desarrollo web tradicional tenemos que aprender un nuevo protocolo que es imprescindible para la comunicación en la web.
Ya vimos que la estructura de red se maneja en capas. También mencionamos que hay una capa de red en dónde está el protocolo IP, una capa Superior de transporte en dónde está el protocolo TCP y ahora vemos una nueva capa que es la de aplicación en dónde se usa el protocolo HTTP.

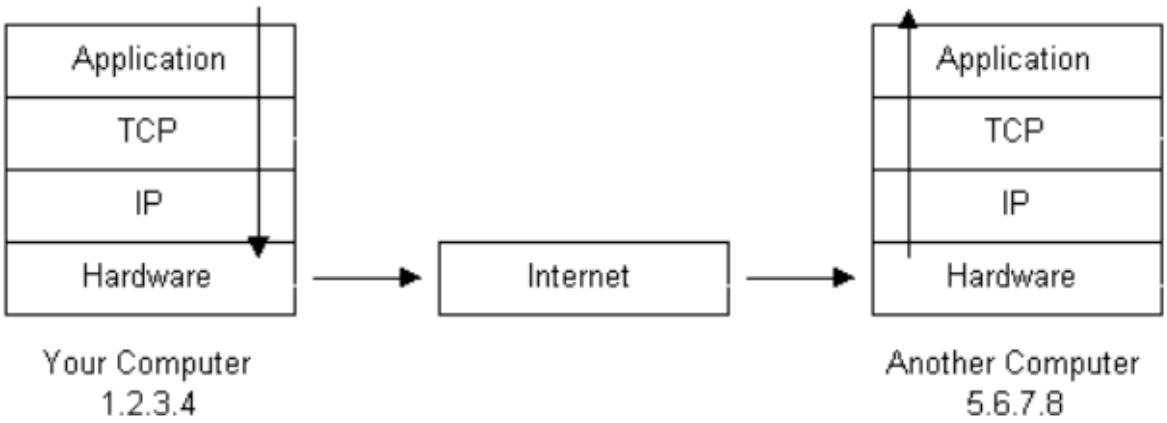


Figura 4: Oracle (10/3/2020) Parte II Administración de TCP/IP. Recuperado de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0sr/index.html
El protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) Es el protocolo de comunicación que permite las transferencias de información en la web.

Es muy importante saber qué HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. Luego veremos las implicancias de esto.

Una descripción importante del protocolo es que es orientado a transacciones y sigue el esquema de petición/respuesta entre un cliente y un servidor. El cliente realiza una petición enviando un mensaje con cierto formato al servidor. El servidor le envía un mensaje de respuesta. Para hacerlo más concreto un cliente podría ser un navegador web y un servidor podría ser una aplicación en un servidor web corriendo en Internet.

Para que entendamos la forma de cómo se programa una aplicación web necesitamos en alguna medida entender los mensajes HTTP.

Los mensajes HTTP son en texto plano lo que los hace más legible y fácil de depurar. Estos tienen la siguiente estructura:

Primero, hay una línea inicial en donde se diferencian todos modos dependiendo de si son peticiones y respuestas. Para las peticiones la línea comienza con una acción requerida por el servidor, a esto se le denomina método de petición (ya veremos cuáles son a continuación) seguido de la url del recurso y la versión http que soporte al cliente. Lo importante, el método de petición y la URL (Uniform Resource Locator o localizador de recursos uniforme.

Para las respuestas la línea comienza con la versión de HTTP seguido por un código de respuesta y con una frase asociada a dicho retorno.

También los mensajes tienen una cabecera que son metadatos con información diversa y el cuerpo de mensaje que es opcional. Típicamente este cuerpo tiene los datos que se intercambian entre el cliente y el servidor.

Los métodos de petición (o también llamados verbos) son varios. Cada método indica la acción qué desea que se efectúe sobre el recurso identificado lo que este recurso representa depende de la aplicación del servidor.

Los métodos que vamos usar son los de **get** y **post**. Cabe destacar que hay convenciones en donde se utilizan otros métodos. En este punto entran en juego el estándar REST y las API's.

El método get solicita una representación del recurso especificado. las solicitudes que usan get sólo deben recuperar datos y no deben tener ningún otro efecto.

El método post envía los datos para que sean procesados por el recurso identificado. Los datos enviados se incluirán en el cuerpo de la petición. esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes.

Para finalizar con la explicación de HTTP vamos a nombrar algunos códigos de respuesta, los que empiezan con 2, por ejemplo el 200 representa una respuesta correcta es decir que indica que la petición ha sido procesada correctamente. Otro ejemplo es la respuesta 404 Qué significa errores causados por el cliente por ejemplo que el recurso no se encuentre. Finalmente las respuestas que comienzan

con 5 por ejemplo el 500 son errores causados por el servidor. Esto indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.
Todo lo que hablamos de HTTP lo vamos a usar al momento de programar una aplicación web. Vamos a ver que al presionar por ejemplo un link se hace una petición get al servidor para buscar otra página lo que resultará en una respuesta 200 si se encuentra la página o en un 404 si no se encuentra. También veremos que al llenar un formulario y enviarlo al servidor lo haremos a través de una petición post.
Los otros protocolos el de TCP/IP son necesarios para configuraciones de conexiones pero a nivel aplicación no los utilizaremos. Eso no significa que no haya que entenderlos pues nos facilitarán la solución de problemas, por ejemplo, al conectarnos a una base de datos desde la aplicación web del servidor.

Referencias:

Tanenbaum, A. S., & Tanenbaum, A. S. (2003). Computer networks, fourth edition: Problem solutions. Prentice Hall PTR.

Medium (10/3/2020). Red de computadoras. Recuperado de: https://medium.com/@KVTacoa/red-de-computadoras-e3941854e4d3

Oracle (10/3/2020) Parte II Administración de TCP/IP. Recuperado de: https://docs.oracle.com/cd/E19957-01/820-2981/6nei0r0sr/index.html

Tecnologías y protocolos de red (10/3/2020). Recuperado de: https://www.dsi.uclm.es/personal/miguelfgraciani/mikicurri/Docencia/LenguajesInternet0910/web_LI/Teoria/Protocolos%20de%20bajo%20nivel/Protocolo%20TCP.htm

Wikipedia (10/3/2020). Protocolo de internet. Recuperado de: https://es.wikipedia.org/wiki/Protocolo_de_internet

Anna Ferry Mestres, http://cv.uoc.edu/annotation/a9c35c372dcee6e6b92afad6993cd048/620334/PID_00250214/PID_00250214.html

Material de lectura complementario:

https://www.wikizero.com/es/Familia_de_protocolos_de_Internet