# Diferenciemos Frontend de Backend (diapos 1 al 4)

**Frontend** es la parte de una aplicación con la que el usuario interactúa. Es un conjunto de tecnologías que se ejecutan en el lado del cliente y que se encargan de la interacción con los usuarios.

Digamos que en una arquitectura cliente-servidor es el conjunto de tecnologías que usamos en el cliente, quien es el encargado de solicitar recursos a una máquina que proporciona (sirve) datos.

Los lenguajes fundamentales que usamos en Frontend son: HTML, CSS y Javascript. De estos se desprenden muchas herramientas y utilidades que nos permiten crear interfaces de usuario de una manera ordenada y rápida.

**Backend** es la capa que trabaja con los datos que se despliegan en el frontend. Es encargado de conectarse con la base de datos y conectarse al servidor, que es una aplicación que realiza la comunicación con el Frontend.

En una arquitectura cliente-servidor es el conjunto de tecnologías que usamos en el servidor, quien es el encargado de proveer recursos a una máquina que los solicita (cliente).

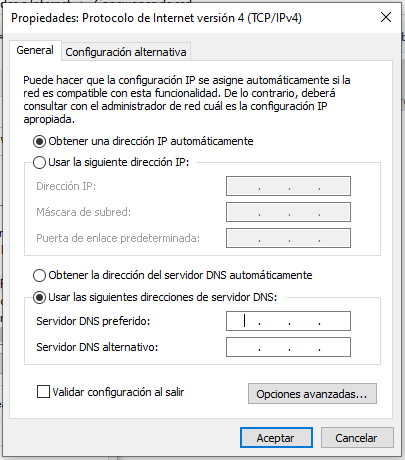
Los lenguajes fundamentales que usamos en Backend son: PHP, Python, Ruby, Java, C#, Java. De estos también se desprenden herramientas para el desarrollo de backends más robustos, seguros y más rápidos de programar; Django, Laravel, Ruby on Rails, etc.

# 

# Flujo de datos en una aplicación web (diapo 5)

Todo empieza cuando ingresamos el nombre de una página web en la barra de direcciones. Nuestro computador busca en sus registros de hosts el nombre de la página y verifica si está registrada, si lo está, obtiene su dirección IP, si no está, nuestra red se conecta a nuestro proveedor de servicios de internet (ISP), por medio de él nos conectamos al servidor DNS configurado en nuestro sistema operativo (imagen de más abajo), si no está configurado en el sistema operativo, se conecta al servidor DNS configurado en nuestro router.

En Windows la pantalla de configuración del servidor DNS luce así:



El servidor DNS le entrega la dirección IP del sitio al cual queremos acceder a nuestro navegador, quien realiza una solicitud al servidor web que tiene la dirección IP que el servidor DNS nos entregó. Si no especificamos (junto al nombre de la página web) el puerto por el cual vamos a acceder al servidor, la solicitud establece conexión por el puerto 80.

Nuestro navegador se conecta a nuestro servidor por medio de solicitudes de datos, estas solicitudes pueden ir acompañado de datos provistos por el navegador y/o el usuario quien interactúa como con la parte cliente. Nuestro servidor se encarga de gestionar la petición y le entrega los datos provistos por el navegador a nuestro backend. El backend es quien realiza la gestión y cálculos usando los datos que ingresan. Nuestro backend, entre otras cosas, indica el formato en que deben ir los datos de vuelta al cliente según cómo lo haya solicitado el cliente (en nuestro caso el navegador). Opcionalmente, nuestro backend puede extraer datos estableciendo una conexión con una base de datos, luego de que la conexión se haya realizado con éxito y los datos fueron extraídos de la base de datos, estos son enviados a nuestro backend quien procesa esos datos, luego, estos datos procesados son formateados según cómo lo solicitó el navegador para ser enviados por nuestro servidor hacia el cliente.

# Pero, ¿Qué pasa en el navegador?

Nuestro navegador está en la parte cliente. Nosotros (por medio del navegador) como clientes, solicitamos recursos a un servidor usando el Protocolo de Transferencia de Hipertexto, también conocido como protocolo HTTP. Quien define la manera en que se realiza la comunicación entre servidores y navegadores.

El servidor perfectamente puede ser puede ser nuestra misma máquina (servidor local) o puede ser otra máquina ubicada físicamente lejos (servidor remoto).

En cada petición al servidor, nuestro navegador recibe datos en distintos formatos. Los más conocidos y actualmente usados son; texto plano, texto HTML, texto CSS, aplicación JSON (tenemos más como audio, video, fuente, imagen, aplicación, etc). Cuando los datos son recibidos por el navegador, son analizados por su motor de renderizado, quien lee los datos y en el caso de que se haya transferido una estructura HTML (texto HTML) se conforma la estructura del sitio en el navegador.

# Stacks de programación (diapos 7 y 8)

Ok, a grandes rasgos este sería el flujo de información entre un navegador y un servidor. Todo bien hasta aquí, pero ahora entremos a conocer cómo nosotros podemos construir un backend y un frontend que sirva para nuestros propósitos. Pero para eso primero debemos conocer las tecnologías que tenemos al alcance.

Estas tecnologías están divididas según a qué nos referimos, o sea, según si estamos hablando de el servidor, el frontend, backend y la base de datos.

Hay que tener presente que estos stacks de programación son un conjunto de tecnologías que se han popularizado pero no son guías estrictas sobre qué tecnologías debemos usar en sistemas o aplicaciones que deseemos construir. Pero si son patrones recomendables debido a la naturaleza de las tecnologías, ya que se basan, por ejemplo, en un lenguaje de programación en común o existen diversas librerías y herramientas que facilitan la conexión entre ellas.

Así, tenemos las tres más populares; LAMP, MEAN y MERN.

## LAMP

LAMP es un acrónimo y hace referencia a:

* **L**inux, como sistema operativo del servidor
* **A**pache, el servidor web instalado en la máquina servidor.
* **M**ySQL, como gestor de bases de datos
* **P**HP, el lenguaje de programación de nuestro backend y frontend

Cabe señalar que el stack LAMP, hoy en día es uno de los más antiguos que podemos ver en sistemas que actualmente están en producción.

¿Vale la pena usarlo en nuestros días? Probablemente no, a menos que por necesidades específicas se indique, pero hoy tenemos mejores opciones más eficientes.

## 

## MEAN

MEAN es un acrónimo y hace referencia a:

* **M**ongoDB, como gestor de bases de datos.
* **E**xpressJS, como herramienta para construir nuestro backend
* **A**ngular, como herramienta (framework, en estricto rigor) para construir nuestro frontend.
* **N**odeJS, como servidor web instalado en la máquina servidor.

¿Qué tienes de bueno este stack? Que utiliza el lenguaje Javascript tanto en el frontend y backend y el formato JSON para comunicación entre ellos (ya veremos qué es JSON). Lo cual acelera nuestro desarrollo no teniendo que aprender y lidiar con dos lenguajes de programación distintos.y JSON que es un formato que Javascript entiende de forma nativa.

## MERN

MERN es un acrónimo y hace referencia a:

* **M**ongoDB, como gestor de bases de datos.
* **E**xpressJS, como herramienta para construir nuestro backend
* **R**eactJS, como herramienta (librería, en estricto rigor) para construir nuestro frontend.
* **N**odeJS, como servidor web instalado en la máquina servidor

Al igual que el MEAN, este stack trabaja con Javascript en el frontend y backend. Se diferencia porque para desarrollar nuestro frontend usamos ReactJS como librería base.

# Links complementarios:

Modelo cliente-servidor

[**https://es.wikipedia.org/wiki/Cliente-servidor**](https://es.wikipedia.org/wiki/Cliente-servidor)

Protocolo HTTP

[**https://developer.mozilla.org/es/docs/Web/HTTP**](https://developer.mozilla.org/es/docs/Web/HTTP)

Formatos de documentos enviados entre cliente y servidor

[**https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\_of\_HTTP/MIME\_types**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types)

Stack MERN

[**https://platzi.com/blog/que-es-mern-stack-javascript/**](https://platzi.com/blog/que-es-mern-stack-javascript/)