APLICACIONES WEB

- *Sara del Pino Cabrera Sánchez* *2º DAW*

BROWSER WARS

- *User Agent (Agente de Usuario):* es una cadena de texto establecida por los creadores de los navegadores para identificar la aplicación que estamos usando.
- *¿Qué ocurre si preguntamos por este dato en la consola de Chrome?*

  > navigator.userAgent
  < '***Mozilla****/5.0 (…)* ***AppleWebKit****/537.36 (…)* ***Chrome****/105.0.0.0* ***Mobile Safari****/537.36*'

- Obtendremos información confusa, ¿por qué al buscar en Chrome nos salen datos de Mozilla? Esto mismo ocurre en otros navegadores.
  Esto no es más que problemas de retrocompatibilidad

Con el comienzo de los navegadores nació *Mosaic*, con este, el *User Agent* era algo muy simple, tan solo identificaba el navegador que se utilizaba:

< '***NCSA_Mosaic****/1.0*'

Más adelante se dieron cuenta de que los usuarios podían acceder al navegador desde diferentes plataformas, por lo que el *User Agent* evolucionó:

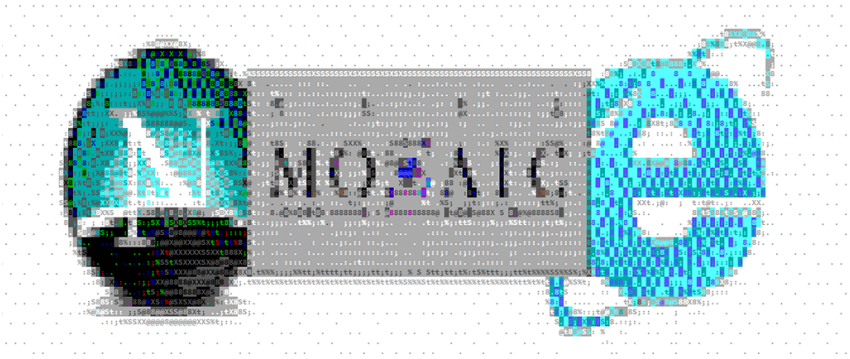
< '***NCSA_Mosaic****/2.0 (Windows 3.1)*'

Mientras tanto, apareció *Netscape Navigator* (nombre código: *Mozilla*) que, con el tiempo, acabó siendo el navegador más utilizado, reemplazando a *Mosaic*. El *User Agent* de sus primeras versiones era algo así:

< '***Mozilla****/3.0 (Win 95; I)*'

Utilizaba este dato para hacer referencia al navegador, sistema operativo y otros datos adicionales.

Fue entonces cuando *Internet Explorer* hacía su aparición, intentando competir contra *Netscape*. En ese momento comenzó la guerra entre navegadores.



Ambos exploradores se centraron tanto en crear nuevas características y funcionalidades que terminaron por dejar de lado la corrección de errores. Esto terminó siendo un dolor de cabeza para los usuarios y programadores de páginas webs, teniendo que decidir entre crear un diseño para uno u otro.

En aquel entonces Google no existía, por lo que *Netscape* era el navegador predominante. Esto hizo que Microsoft tomara la decisión de añadir como palabra clave en su *User Agent* "Mozilla" para no ser excluidos como navegador y poder visualizar el contenido específico para *Netscape*.

< '***Mozilla****/4.0 (compatible; MSIE 4.01, Windows 98)*'

Tras varios años, por fin *Internet Explorer* consiguió adelantar a *Netscape* que, por varios atrasos y problemas, terminó dando pie a un nuevo navegador llamado *Mozilla Firefox*. Su *User Agent* tiene algunas diferencias:

< '***Mozilla****/5.0 (X11; U; Linux i686; en-US; rv:1.7.6)* ***Gecko****/20050317* ***Firefox****/1.0.1*'

Con *Firefox* surge una nueva guerra que hace mucho daño a *Internet Explorer*, que se había quedado obsoleto en funcionalidades y características, con muchísimos errores sin corregir

que *Mozilla Firefox* superaba con creces. Desde características básicas como las pestañas en el navegador, a la velocidad del mismo o el buen rendimiento de su motor interno de renderizado de páginas.

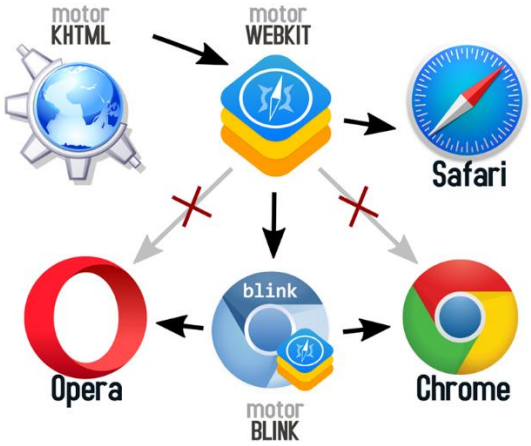
Internet Explorer se actualiza sólo cuando ve que *Mozilla Firefox* está funcionando tan bien que podría llegar a amenazar su situación predominante. Es entonces cuando este comienza a coleccionar todo tipo de información en el *User Agent*:

< '***Mozilla**/4.0 (compatible; **MSIE** 7.0; **Windows NT** 5.1; **Trident**/4.0; **FDM**; **.NET CLR** 2.0.50727; **Alexa Toolbar***'

Más adelante nace *Konqueror*, utilizando el motor JavaScript KJS y el motor de renderizado KHTML (principal aliciente para la creación de *Safari*). Su *User Agent* sigue el esquema de los anteriores:

< '***Mozilla**/5.0 (compatible; **Konqueror**/3.0; i686 **Linux**; 20021219)*'

Con el paso del tiempo, Apple se basó en el motor KHTML para crear un nuevo motor de renderizado llamado Webkit, que sería el núcleo de *Safari*. Aunque comenzó siendo exclusivo de Apple, terminó siendo liberado como software libre, convirtiéndolo en uno de los principales motores del panorama web.



El *User Agent* de *Safari* sería el siguiente:

< '***Mozilla**/5.0 (**Macintosh**; **Intel Mac OS X** 10_10_2 **AppleWebKit**/600.3.18 (**KHTML, like Gecko**) **Version**/8.0.3 **Safari**/600.3.18)*'

Un poco más tarde Google lanzó su propio navegador, *Chrome*. Un éxito rotundo que desbancó a *Firefox* como el navegador favorito de los usuarios.

Chrome comenzó usando Webkit, al igual que *Safari*, pero con el tiempo terminó creando un fork de este para crear su propio motor de renderizado: Blink.

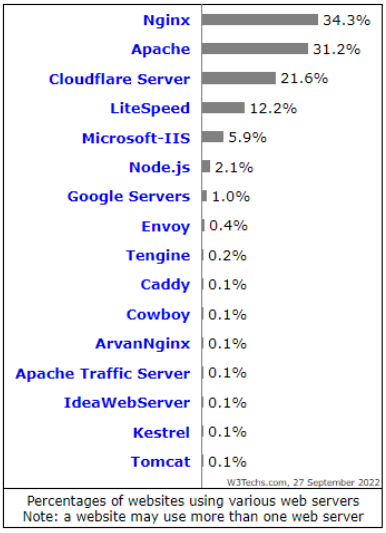
- *¿En qué nos afecta todo esto como desarrolladores?*
- Al haber tantos navegadores y estos luchar incesantemente entre sí, terminan por falsificar su *User Agent* y a descuidar apartados claves en sus navegadores. Como desarrolladores esto nos influye muchísimo, ya que encontraremos incompatibilidades y que, debido a un falso *User Agent*, nuestra web no será correctamente visible en todas partes.
- Al final siempre tenderemos a obtener un Cross-Browser, una web que funciona y se visualiza de igual manera en cualquier navegador, pero es una tarea bastante complicada.

Servidores Web

Un servidor web es un software que forma parte del servidor y tiene como misión principal devolver información (páginas) cuando recibe peticiones por parte de los usuarios. En otras palabras, es el software que permite que los usuarios que quieren ver una página web en su navegador puedan hacerlo.

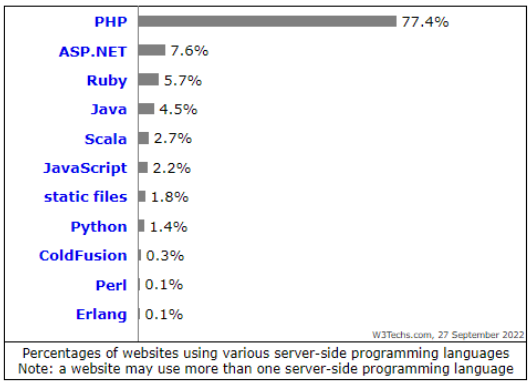
Esto claramente nos afecta como programadores web, pues tendremos que hacer peticiones a nuestro servidor para obtener información y poder mostrársela correctamente a los usuarios.

Como podemos observar, hay tres claros ganadores en lo que a Servidores Web se refiere:



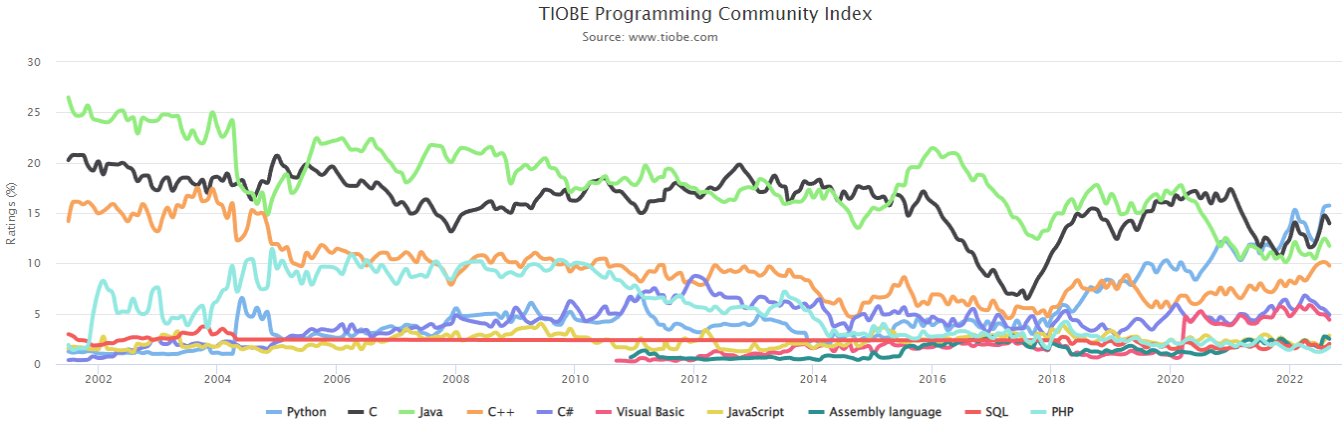
- *Nginx:*
  - *Sirve archivos estáticos y dinámicos.*
  - *Sirve como proxy inverso.*
  - *Soporte de autentificación HTTP.*
  - *Balanceo de carga y tolerancia a fallos.*
  - *Alta disponibilidad.*
- *Apache:*
  - *Gratuito y de fuente abierta.*
  - *Instalación y configuración sencilla.*
  - *Extensible y adaptable mediante módulos.*
  - *Soporte para PHP, Perl y Python.*
- *Cloudflare Server:*
  - *Protección contra amenazas y bots.*
  - *Modo de navegación Offline.*
  - *Disminución del uso de la CPU.*
  - *Alerta a visitantes de ordenadores infectados.*
  - *Desempeño mejorado.*

Programación Lado Servidor



Esto nos afecta como programadores principalmente a la hora de buscar trabajo y aprender nuevas tecnologías. Siempre es importante conocer que es lo que más se usa, pero también lo que menos, pues esto podría abrir más puertas de trabajo al haber menos personas interesadas.

Lenguajes Populares



Conocer los lenguajes más populares es recomendable, pues podremos hacernos una idea de que nos solicitaran cuando accedamos a un puesto de trabajo. Por lo que siempre es bueno tener unas nociones básicas sobre estos lenguajes.

Apps Nativas (Móviles)

Las aplicaciones nativas de las plataformas móviles se implementan con diferentes tecnologías:

- *Android:* Kotlin.
- *iPhone y iPad:* Swift.
- *Desarrollo híbrido:* Ionic, Flutter, React Native y Xamarin.

APLICACIONES DE INTERNET (MÓVILES)

- *Aplicaciones Progresivas (PWA).*
- Para poder hacer uso de algunas páginas web como *PWA* deberemos ir a la dirección de la web en concreto, ir al menú (los tres puntos) y seleccionar la opción de "*Añadir a pantalla de inicio*". En caso de disponer de aplicación web progresiva se nos instalará en el móvil.
- Para que puedan ser publicadas en la Play Store deberá:
    - Cumplir con los criterios de instalación de *PWA* (que nombramos anteriormente).
    - Tener al menos una puntuación de 80/100 en *Lighthouse*.
    - El *PWA* debe cumplir con la política de Play Store.

FRAMEWORKS POPULARES