D1-UD1 Ejecución de código tras un servidor web

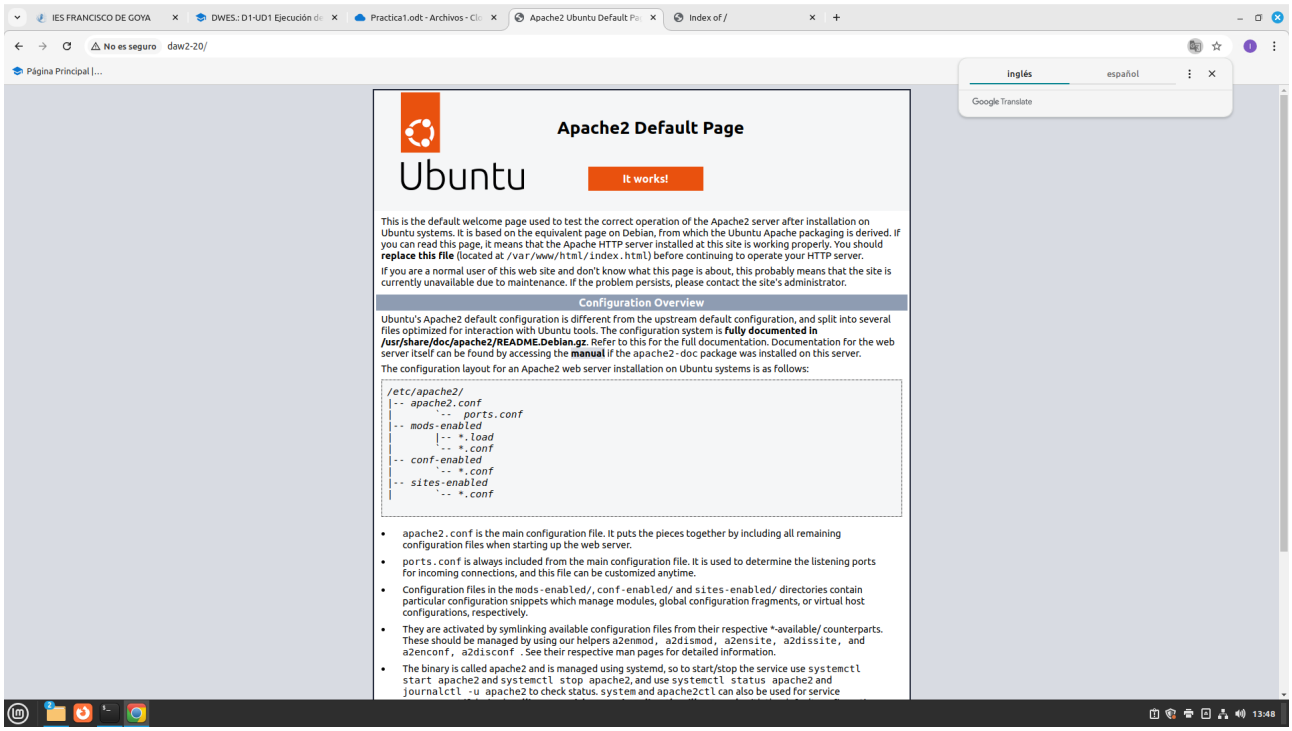
Autores:

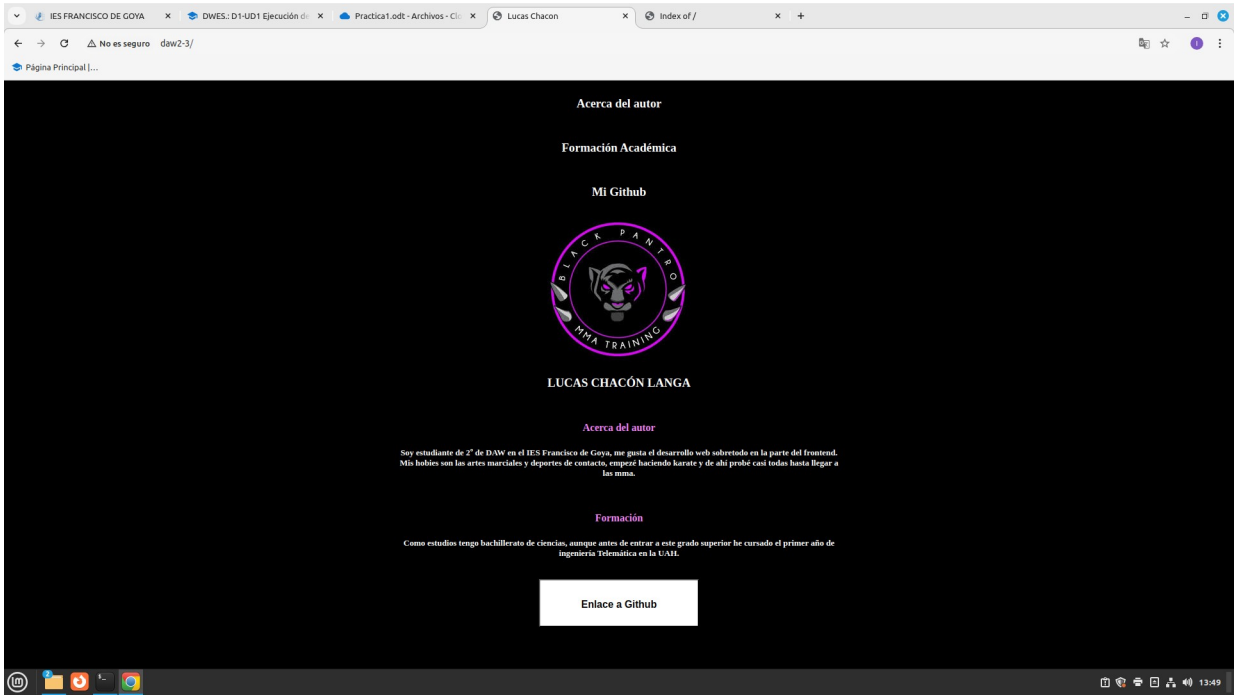
Iván Turro Arroyo, Lucas Chacón Langa, David Barroso Martínez y Sergio Berrendero

Desarrollo: en grupos de dos a cuatro alumnos, se trata de empezar a experimentar con un servidor web (Apache) mediante los siguientes pasos:

1. En vuestras máquinas reales ya tenéis instalado el servidor web Apache y el profesor os ha dado permisos en la carpeta /var/www/html para añadir o cambiar páginas. Podéis probarlo desde el navegador con http://localhost/ o de una máquina a otra con http://dirección-ip/

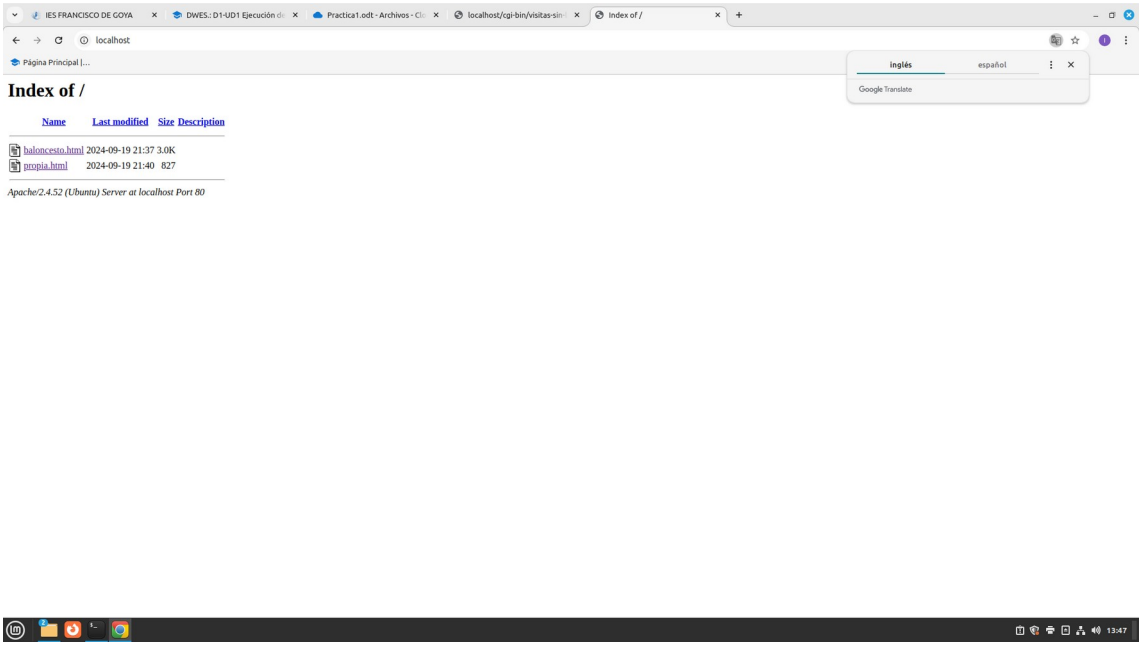
   Comprobamos en las maquinas de varios compañeros para ver los cambios que hemos hecho cada uno en nuestra página principal





1.1 Tras comprobar que el servidor funciona, personaliza la página web que sirve por defecto, crea subcarpetas y añade nuevas páginas. Comprueba que funcionan los enlaces (link) relativos entre páginas web obtenidas a través del servidor: en el navegador no abres la página web desde el almacenamiento, si no a través de http://localhost/carpeta/página ... (o la IP en vez de localhost, o el nombre ...)

Tras comprobar que el servidor funciona, personalizamos la página web por defecto, creamos subcarpetas y añadimos nuevas páginas. Verificamos que los enlaces relativos entre las páginas funcionaran correctamente accediendo a ellas mediante la IP o nombre de dominio.

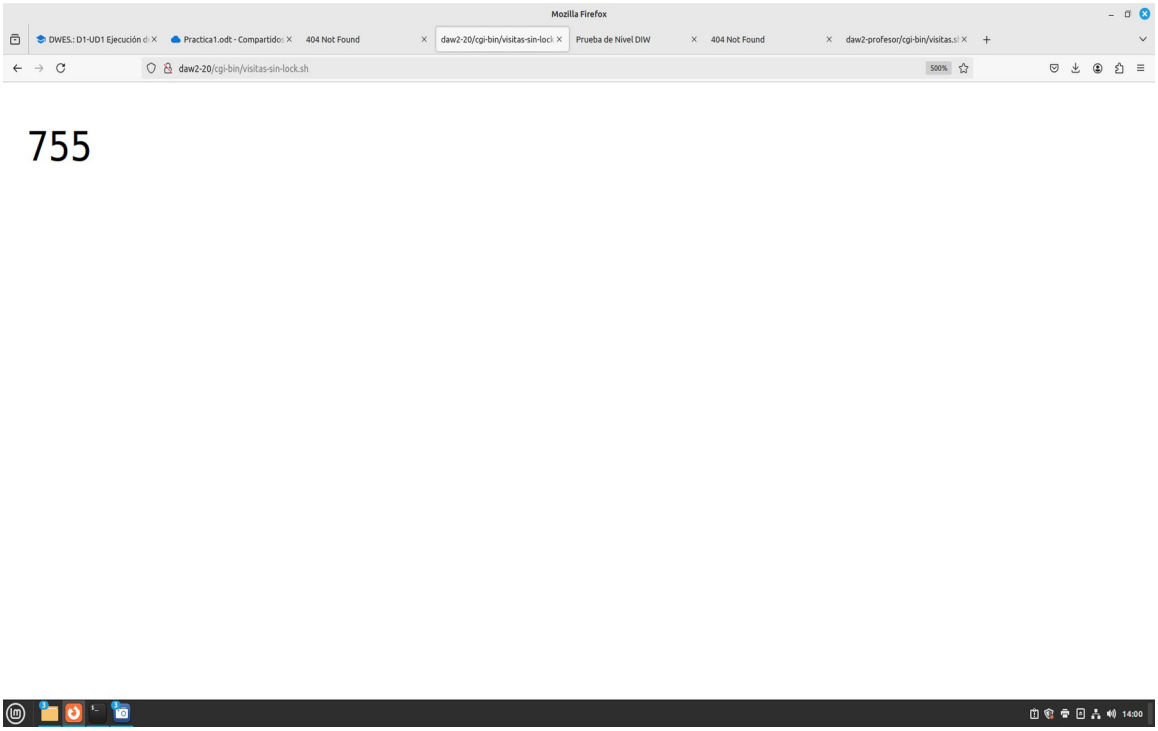


1.2 Prueba http://localhost/cgi-bin/hola.sh , también terminado en visitas-sin-lock.sh

Probamos hola.sh y nos aparece el Hola la dirección ip local y el puerto que utiliza el servidor apache en el caso que lo hagamos a la pagina local, sin embargo si consultamos a otro ordenador del aula aparecerá su respectiva ip privada, también aparecerá la fecha y hora de cuando se realizo la petición.


```
Hola 127.0.0.1:38620, la hora es: Fri Sep 20
14:09:03 CEST 2024
```

Probamos visitas sin lock desde local y va añadiendo de 1 en 1 en el ordenador 20 para probarlo en remoto desde el ordenador 18.

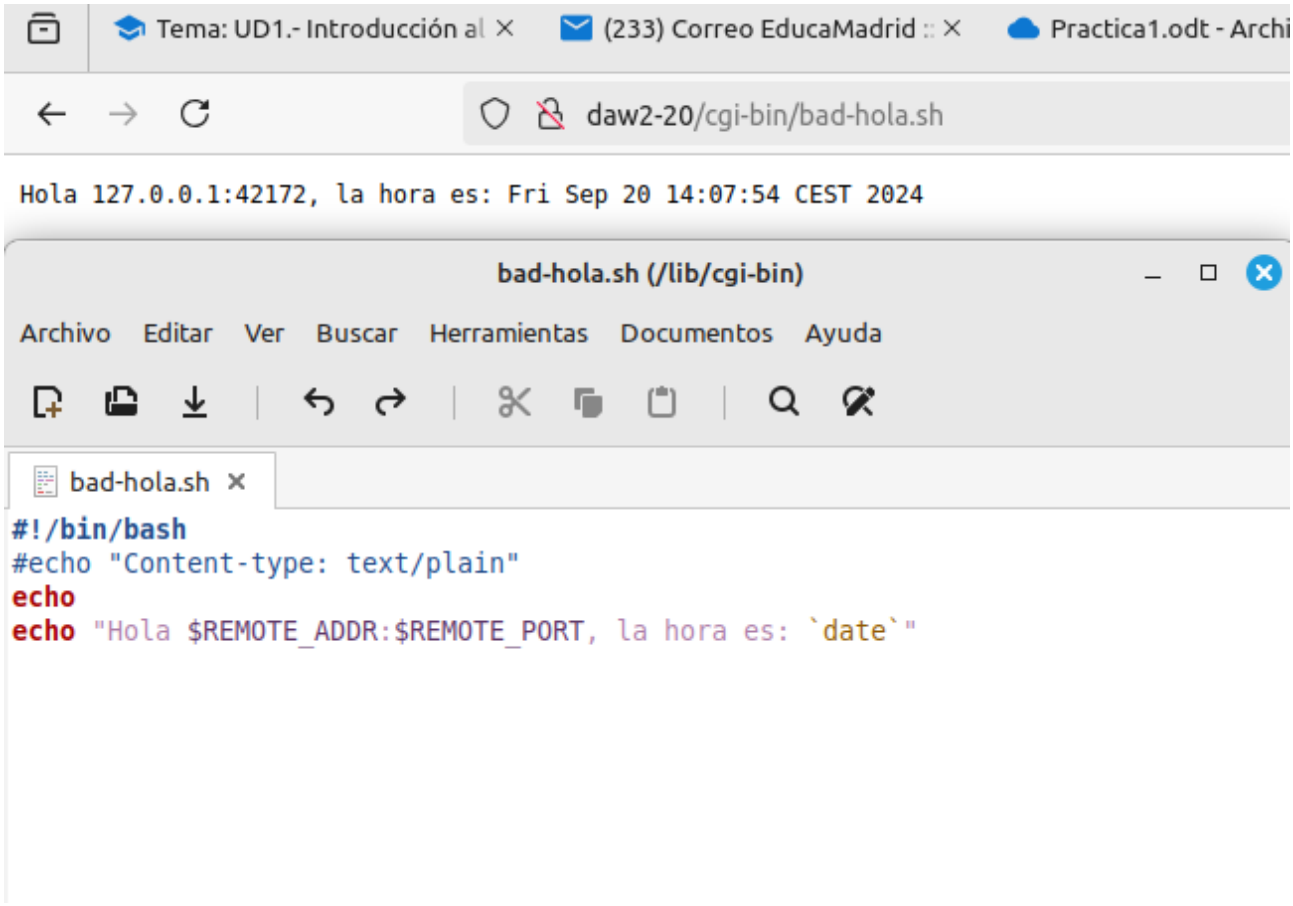


1.3 Accede desde varios navegadores y desde varias máquinas. ¿Cual es el resultado? ¿Dónde guarda el número de visitas? ¿Qué pasa si quitas al script el echo de una línea en blanco debajo del Content/Type? ¿Y si quitas el Content/Type? (versión bad-hola.sh)

Si quitamos el echo de la linea en blanco lo que ocurre es que aparece un *“Internal Server Error”.*

Si quitamos el “*Content-type”* o toda la linea no ocurre nada y todo seguirá funcionando de manera normal como si lo abriésemos sin modificar.

De manera curiosa el numero después de la ip va cambiando, que es el puerto de respuesta a la petición.



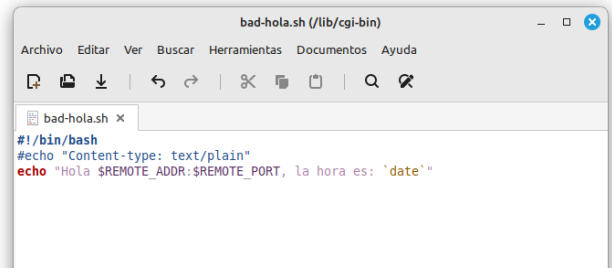
Internal Server Error

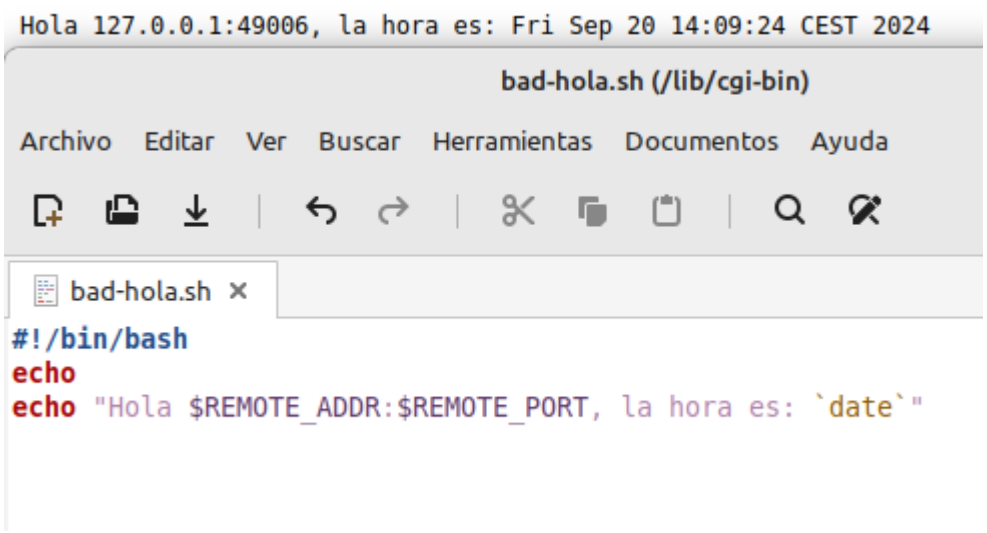
The server encountered an internal error or misconfiguration and was unable to complete your request.

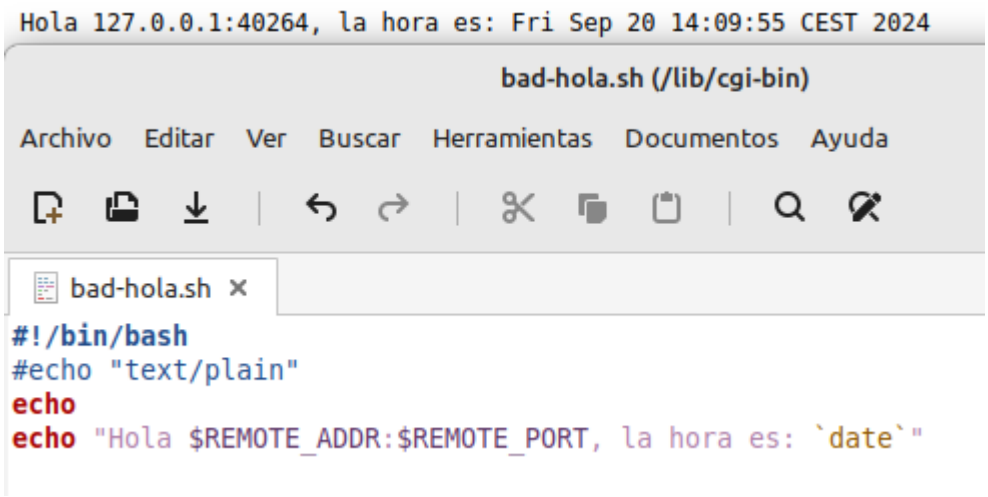
Please contact the server administrator at webmaster@localhost to inform them of the time this error occurred, and the actions you performed just before this error.

More information about this error may be available in the server error log.

Apache/2.4.52 (Ubuntu) Server at daw2-20 Port 80







2.4 Optativa: peticiones masivas con ab (ApacheBench, comando ab que se instala junto con apache, ej ab -n 1000 -c 10 http://localhost/cgi-bin/visitas.sh → realiza mil peticiones de 10 en 10 a la dirección dada) Analiza la diferencia entre los resultados producidos por las distintas versiones del script.

ab -n 1000 -c 10 http://localhost/cgi-bin/visitas.sh

para enviar 1000 peticiones simultáneas, 10 a la vez. Notamos una diferencia clara entre los dos scripts:

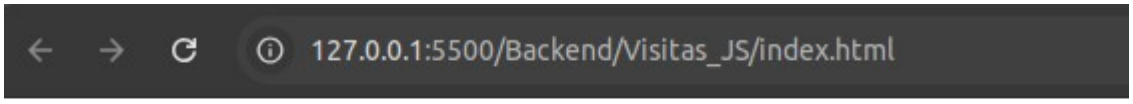
Script con lock: las visitas se registran correctamente, procesando cada petición de manera secuencial. El archivo de visitas no se sobrescribe ya que el lock garantiza que solo una petición se ejecute a la vez.

Script sin lock: al recibir múltiples peticiones desde varios dispositivos, las visitas se sobrescriben. Algunas peticiones más lentas registran correctamente las visitas, pero la mayoría pierde datos debido a la falta de control.

Se llega a la conclusión de que el script sin lock falla ante peticiones simultáneas, mientras que el script con lock mantiene la integridad de los datos

3. Reflexiona y escribe sobre la siguiente cuestión¿Puedes hacer un contador de visitas en Javascript para Frontend? ¿Funcionaría igual que el anterior en el backend? Justifica la respuesta.

Tras hacer el programa en JavaScript, podemos ver que se usa la herramienta localStorage que es una zona en el navegador donde se guardan diferentes datos, en este caso el número de visitas al acceder a la web, que es un contador. La diferencia que hay frente al backend es que ese contador va asociado a una pagina con su ip y puerto en concreto es el la ip y puerto 127.0.0.1:5500 que levanta la extensión live server de Visual Studio Code, es decir, que si cambiamos de navegador ese contador volverá a estar a 0. El JavaScript queda adjuntado a la tarea. Este ejercicio sirve para que quede claro la importancia del servidor.



Has visitado esta página 13 veces.

Conclusión:
A través de esta práctica, hemos adquirido una comprensión más clara y visual de las diferencias fundamentales entre las páginas dinámicas y estáticas. Las páginas dinámicas, al interactuar con el servidor y actualizar contenido en tiempo real, ofrecen una experiencia más personalizada y flexible en comparación con las páginas estáticas, que se caracterizan por su contenido fijo. Además, hemos experimentado la relevancia de ciertos mecanismos de control, como el atributo "lock", que aunque puede ralentizar el tiempo de respuesta del sistema, es esencial para garantizar la seguridad y consistencia en la ejecución de procesos concurrentes funcionando como embudo para que las visitas no se sobrescriban.

Asimismo, al implementar la lógica en JavaScript, hemos comprendido la importancia crítica de una base de datos en el backend. Depender exclusivamente del almacenamiento local (local storage) del navegador no es suficiente para gestionar de manera centralizada información esencial, como el número de visitas u otros datos clave. Esto subraya la necesidad de un sistema de almacenamiento persistente y robusto que permita compartir y sincronizar datos entre diferentes usuarios y dispositivos, lo que en última instancia mejora la escalabilidad y funcionalidad de las aplicaciones web dinámicas.