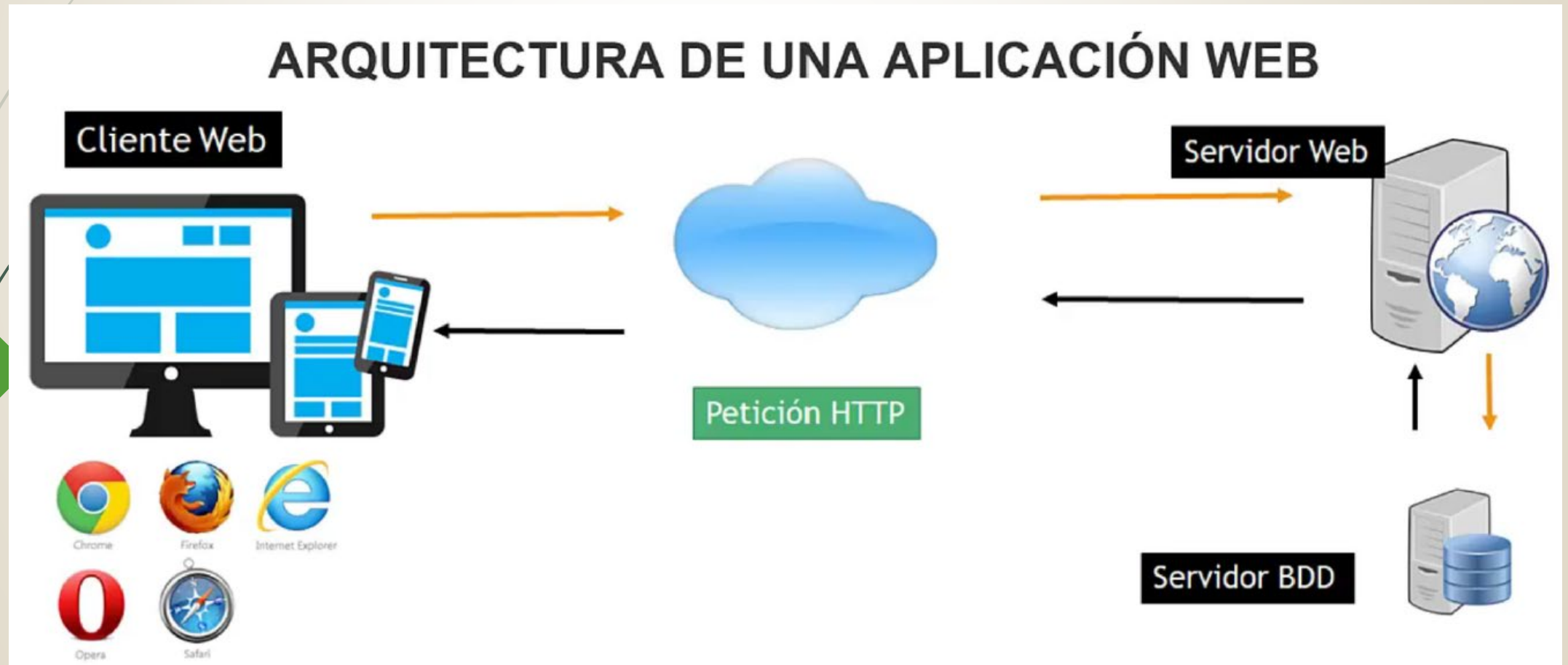


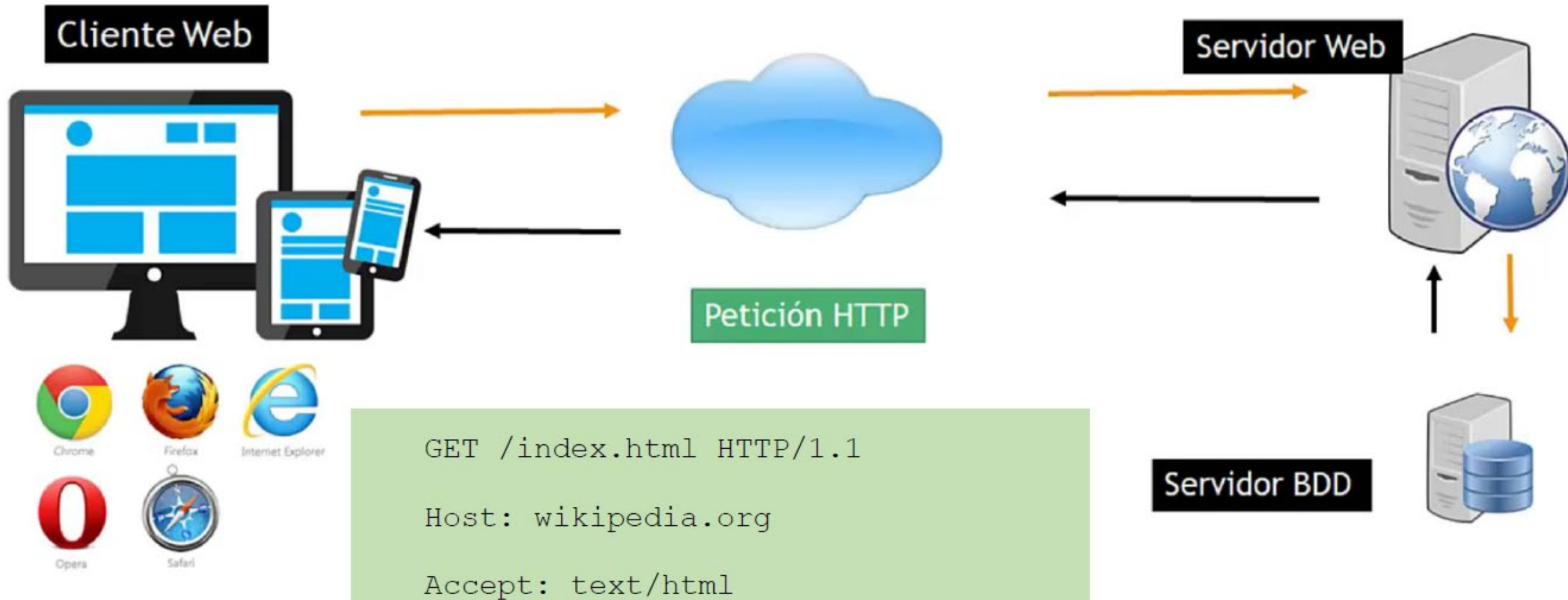
1.2. Lado cliente y lado servidor

Partiendo del esquema del apartado anterior ...



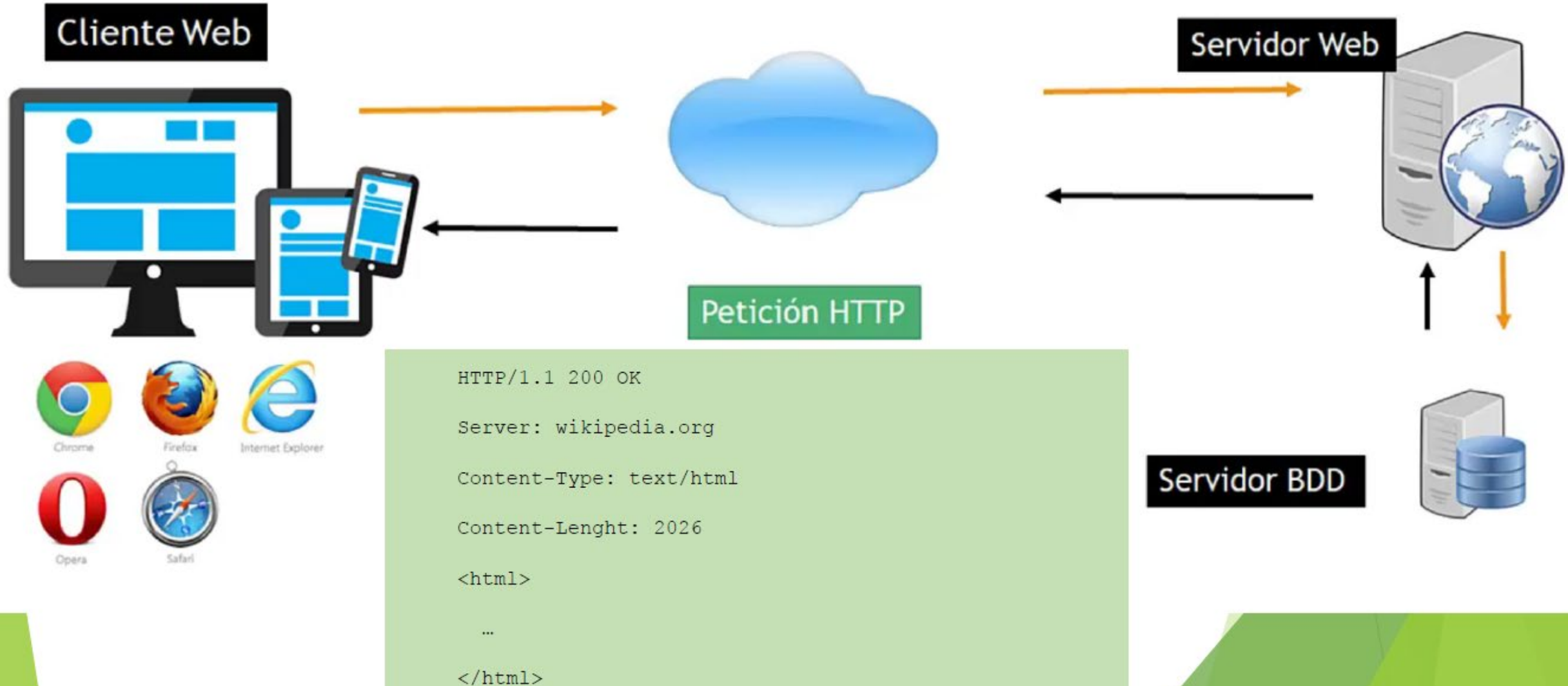
Tendríamos por un lado la petición ...

ARQUITECTURA DE UNA APLICACIÓN WEB

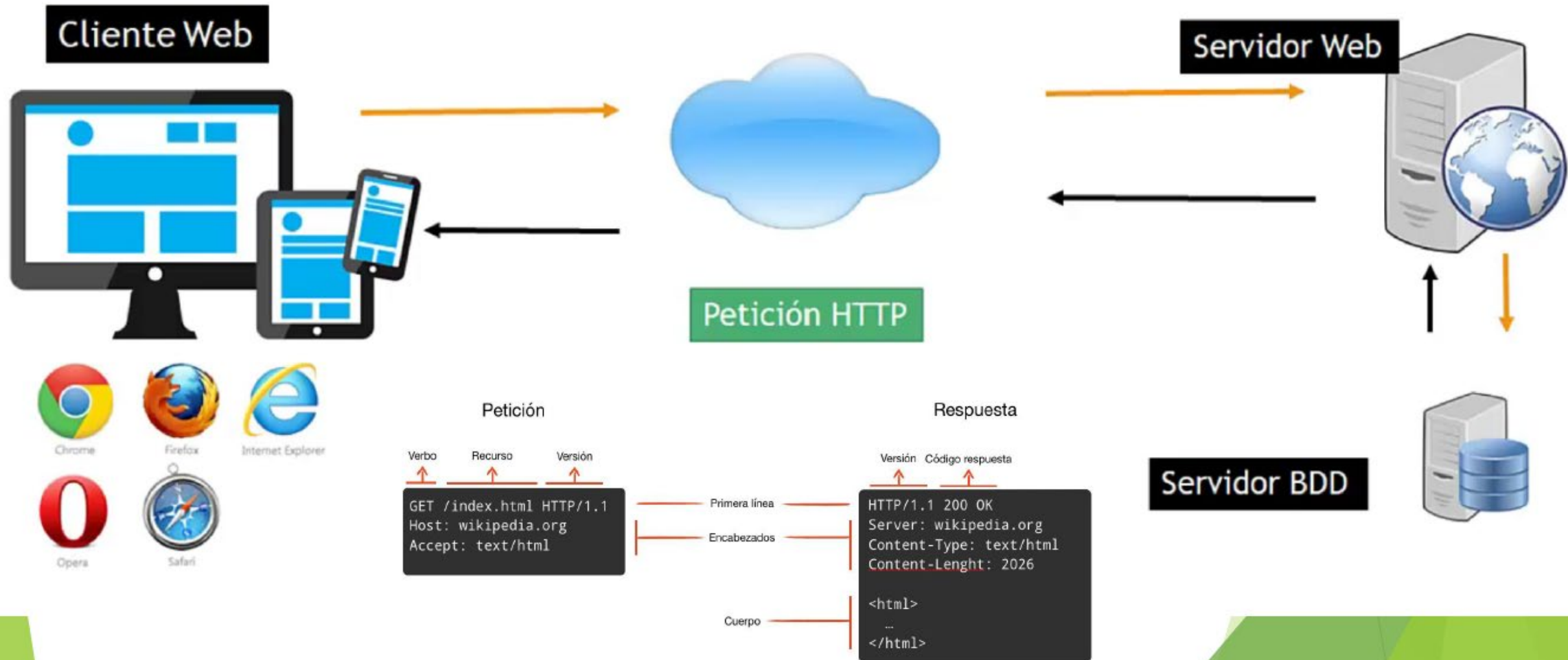


Y por otro, la respuesta ...

ARQUITECTURA DE UNA APLICACIÓN WEB



ARQUITECTURA DE UNA APLICACIÓN WEB



1.2.1. Arquitectura cliente-servidor

La arquitectura cliente-servidor se basa en el modelo de comunicación **petición-respuesta**.

El campo del desarrollo web generalmente se divide en **front-end** (el lado del usuario o lado del cliente) y **back-end** (el lado de la máquina o lado del servidor).

Veamos las características de cada lado:

Lado cliente

Se trata de todo aquello que el usuario puede ver, con lo que puede interactuar y experimentar .

El objetivo de estos desarrolladores es programar las partes del sitio web que son visibles para el usuario y que se ejecutan en el navegador.

Desarrollan los diseños de la interfaz de usuario (UI) y su experiencia (UX), que son los elementos clave para dar vida a la interfaz.



Lado servidor

Las acciones realizadas por el usuario son analizadas, recuperadas y devueltas por el back-end a través de los programas almacenados y ejecutados en el servidor que los aloja.

El trabajo de estos desarrolladores incluye vincular todos los aspectos del front-end entre sí y con las bases de datos desde donde extraen los datos que aparecerán en el front-end.

El lado servidor es muy importante porque es donde está programada la lógica de negocio de la aplicación.

Lógica de negocio → son las reglas, cálculos y procesos que rigen el funcionamiento de la aplicación, dictando cómo se manejan, transforman y presentan los datos.



	Lado cliente	Lado servidor
Definición	Implica la implementación efectiva de los componentes visuales de una aplicación web.	Implica la implementación efectiva de funcionalidades de una aplicación web, donde se incluyen el acceso a datos, la administración de servidores, etcétera.
Habilidades requeridas	HTML, CSS, SASS, JavaScript...	Python, Ruby, Java, PHP...
Independencia	No puede funcionar de forma independiente excepto en el caso de sitios estáticos.	Funciona de forma independiente respecto del <i>front-end</i> .
Objetivo	Garantizar que todos los usuarios puedan acceder a la aplicación y que siga respondiendo en todos los dispositivos.	Garantizar que la aplicación se ejecute en todos los casos, sea escalable y funcione de manera eficiente con baja latencia y sin fallos.
Equipo de desarrollo	Su trabajo es diseñar y desarrollar la apariencia e interactividad de la aplicación en función de la entrada de usuario.	Su trabajo es proporcionar datos al <i>front-end</i> , vincular páginas, brindar seguridad y soporte a los usuarios.
Frameworks utilizados	AngularJS, React, vue.js...	Django, Flask, CakePHP, Laravel, Ruby on Rails...
Habilidades adicionales	Una buena comprensión del diseño de UI y UX.	Razonamiento lógico y resolución de problemas.




Tareas comunes en el lado servidor:

- Acceder y guardar los datos - base de datos
- Enviar correos electrónicos
- Procesar archivos
- Generar páginas web dinámicas
- Validación de formularios (también en el cliente)
- Autenticación y autorización (parcialmente en el cliente)
- Procesar datos y realizar cálculos




Tareas comunes en el lado cliente:

- Interacción con el usuario – interfaz
 - Validación de formularios
 - Autenticación y autorización
 - Procesar datos y realizar cálculos
- 

1.2.2. Arquitectura de tres niveles

Es muy importante conocer la arquitectura de una aplicación web típica (esquema del principio del apartado).

Las aplicaciones web de hoy en día utilizan las tecnologías de ambos lados siguiendo una arquitectura de tres niveles.



Organiza las aplicaciones en tres niveles informáticos lógicos y físicos: el **nivel de presentación** o interfaz de usuario; el **nivel de aplicación**, donde se procesan los datos; y el **nivel de datos** donde se almacenan y gestionan los datos asociados a la aplicación.

El principal beneficio de esta separación es que cada nivel se ejecuta en su propia infraestructura, cada nivel puede ser desarrollado simultáneamente por un equipo de desarrollo independiente y puede actualizarse o escalarse según sea necesario sin afectar a los otros niveles.

A continuación, se verá cada nivel con un poco más de detalle:



Nivel de presentación

Define la interfaz de usuario y la comunicación con la aplicación, donde el usuario final interactúa con la aplicación.

Su objetivo es mostrar información y recopilar información del usuario.

Se ejecuta en el navegador y generalmente se desarrolla utilizando HTML, CSS y JavaScript.

Este nivel estaría representado como el servidor web.

Nivel de aplicación

También conocido como nivel lógico o intermedio, es el corazón de la aplicación.

La información recopilada en el nivel de presentación se procesa, a veces con la colaboración del nivel de datos, utilizando la lógica de negocio, pudiendo también desde aquí añadir, modificar o eliminar datos del nivel de datos.

Se desarrolla utilizando Python, Java, PHP o Ruby, y se comunica con el nivel de datos mediante llamadas a su API.

API (Interfaz de Programación de Aplicaciones) → es un conjunto de reglas y definiciones que permite a dos aplicaciones de software diferentes comunicarse, intercambiar datos y utilizar funcionalidades de forma eficiente.


Este nivel estaría representado como el servidor de aplicaciones



Nivel de datos

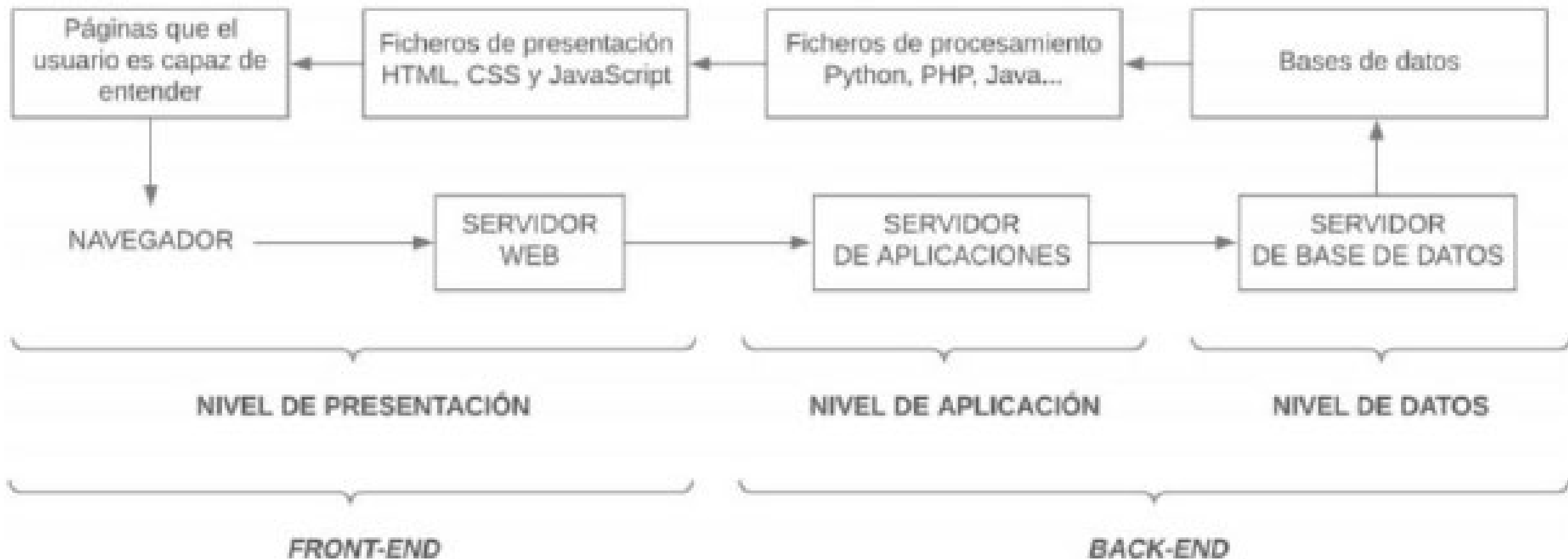
También conocido como nivel de bases de datos o nivel de acceso a datos.


Es donde se almacena y administra la información procesada por la aplicación.



Este puede ser un sistema de gestión de base de datos relacional como MySQL, PostgreSQL, Oracle o Microsoft SQL Server, o un servidor de base de datos NoSQL como MongoDB, CouchDB o Cassandra.

Este nivel estaría representado como el servidor de bases de datos.






Usuario → Acción en el navegador → Desencadena una petición http al servidor web → El servidor web necesita realizar cálculos de procesamiento pertenecientes a la lógica de negocio → Solicita al servidor de aplicaciones que realice esta tarea → Es posible que el servidor de aplicaciones necesite datos que están almacenados, por lo que solicita estos datos al nivel de datos → Una vez completado el flujo de solicitud se comienza a construir la respuesta al usuario → Los datos serán procesados → Se generarán ficheros de presentación que el servidor web mostrará al usuario en el navegador en un formato entendible.

1.2.3. Tecnologías del lado cliente

Para el desarrollo web en entorno cliente hay que dominar al menos estas tres tecnologías:

- HTML
- CSS
- JavaScript


Además, existen otras que complementan, aceleran y optimizan el desarrollo de aplicaciones web del lado cliente. Aunque existe un inmenso catálogo disponible, algunas de las más utilizadas serían:

- 
- **jQuery** es una librería de JavaScript que mejora el procesamiento del código HTML, manejo de eventos y las animaciones. Es muy conciso y reduce drásticamente la cantidad de líneas del código.
 - **vue.js** es un framework JavaScript creado por un ex ingeniero de Google para crear aplicaciones web compactas.
 - **AngularJS** es un framework JavaScript lanzado por Google que proporciona elementos más atractivos a las plantillas HTML y aumenta su rendimiento.

- **React** es otro framework JavaScript, lanzado por Facebook, muy popular, que mejora los componentes de la interfaz de usuario y provee de mucho más dinamismo a las aplicaciones web. Se utiliza principalmente en sitios web de alto tráfico y mucha interactividad . Es el preferido por los desarrolladores seguido de vue.js y luego Angular.

1.2.4. ¿Por qué JavaScript?

- Es muy fácil de implementar: tan solo es necesario colocar el código en un documento HTML y decirle al navegador que es JavaScript.
- Funciona en todos los navegadores que usan los usuarios de la web, incluso cuando están desconectados.
- Permite crear interfaces altamente amigables que mejoran la experiencia del usuario y brindan mucho dinamismo, sin tener que esperar a que el servidor reaccione y muestre otra página.

- 
- Puede cargar contenido de forma asíncrona en el documento si el usuario lo necesita y cuando lo necesite, sin recargar toda la página.
 - Puede ayudar a solucionar problemas de incompatibilidades entre navegadores.
 - En la actualidad alrededor del 95% de todas las aplicaciones web se crean con JavaScript.