# Aplicación Web - Web application

**Definición**

*A distributed application, a kind of client-server computer program, hosted in a web server and that runs in a web browser.*

**Características**

- *Portables:* No están comprometidas a una arquitectura o sistema operativo
- *Usan estándares:* Es decir que son soportados en una variedad de navegadores, algunos ejemplos son: HTML, CSS y ECMAScript

***Algunas aplicaciones móviles son desarrolladas como aplicaciones web (no son nativas) pero, ¿qué opinas de las aplicaciones nativas: iOS, android? ¿presentan características similares a las de una aplicación web actual? ¿en qué son diferentes?***

***Mi opinión de las aplicaciones nativas es que son aplicaciones creadas específicamente para un sistema operativo por lo que lo aprovechan completamente para ser lo mas fluido posibles el problema es que no solo existe un sistema operativo y ahí sus desventajas.***
***Las aplicaciones web actuales hoy en día logran hacer cosas similares como las nativas pero aun existen algunas diferencias. Por ejemplo las nativas ocupan recursos como del sistema como del hardware mientras las web solo el navegador, las nativas puede ser publicadas por tiendas de distribución y por ultimo las nativas la mayoría no necesita estar conectada a Internet para su uso. Pero una de las mayores diferencias es que las nativas no pueden usarse en cualquier dispositivo importa es sistema operativo por lo contrario de las aplicaciones web.***

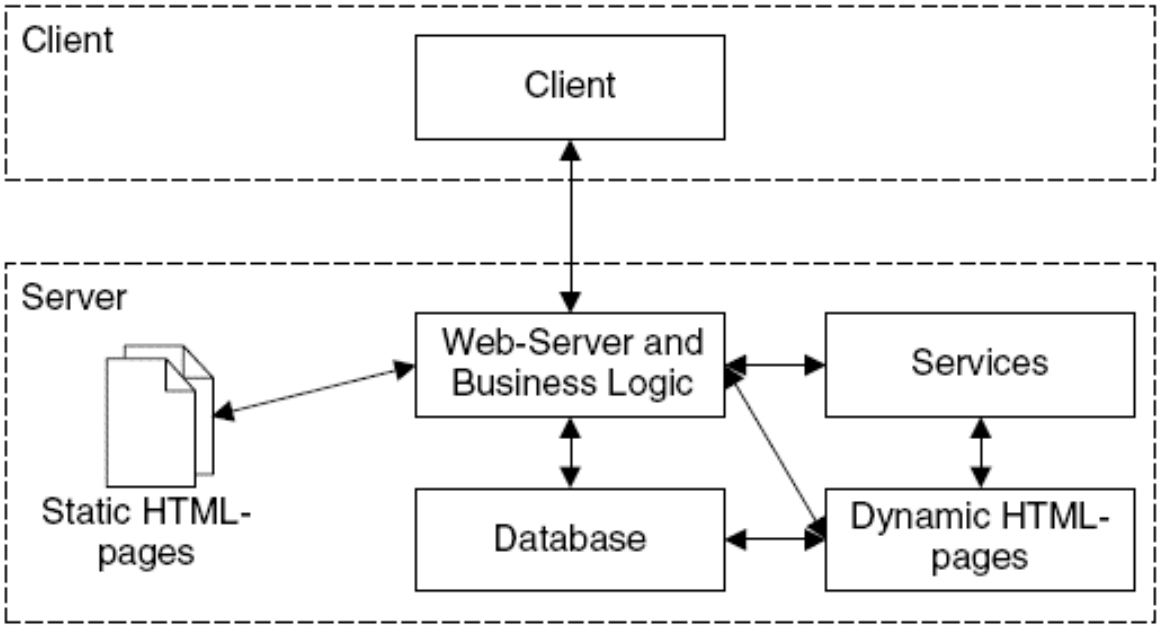
**Historia**

Anteriormente, el contenido era en su mayoría estático y solo era interactivo por contener híper vínculos. Sin embarro cualquier cambio por más insignificante, significaba una ida y vuelta al servidor para refrescar el sitio completo.

JavaScript agregó elementos dinámicos para agregar contenido, sin embargo aún se necesitaba refrescar ciertos segmentos del sitio web.

Empezando en 2001 hasta 2005, se agregaron nuevos paradigmas de interacción con usuarios tal como XMLHttpRequest y AJAX. Estos permitieron enviar o recibir información sin recargar la página. Entonces pasamos de tener sitios web a aplicaciones web.

**Arquitectura**




La mayoría de los recursos en la Web se basan en el modelo cliente servidor y usando lenguajes de marcado para transferir o representar información: XML y HTML. Debajo de esa primer capa, existen varios lenguajes de programación o scripting que procesan, modifican o generan información dinámicamente e incluso proveen una interfaz de usuario.

***El desarrollo de las aplicaciones web es multidisciplinario, ¿por qué crees que se dice eso?***

***Yo creo es considerado multidisciplinario ya que para su desarrollo es necesario conocer y aplicar de diferentes disciplinas. Pueden ser ocupar diferentes lenguajes de programación, frameworks o distintas tecnologías.***

El desarrollo de aplicaciones web se encuentra en un ambiente cambiante y los requerimientos evolucionan a medida que la comunidad crece.

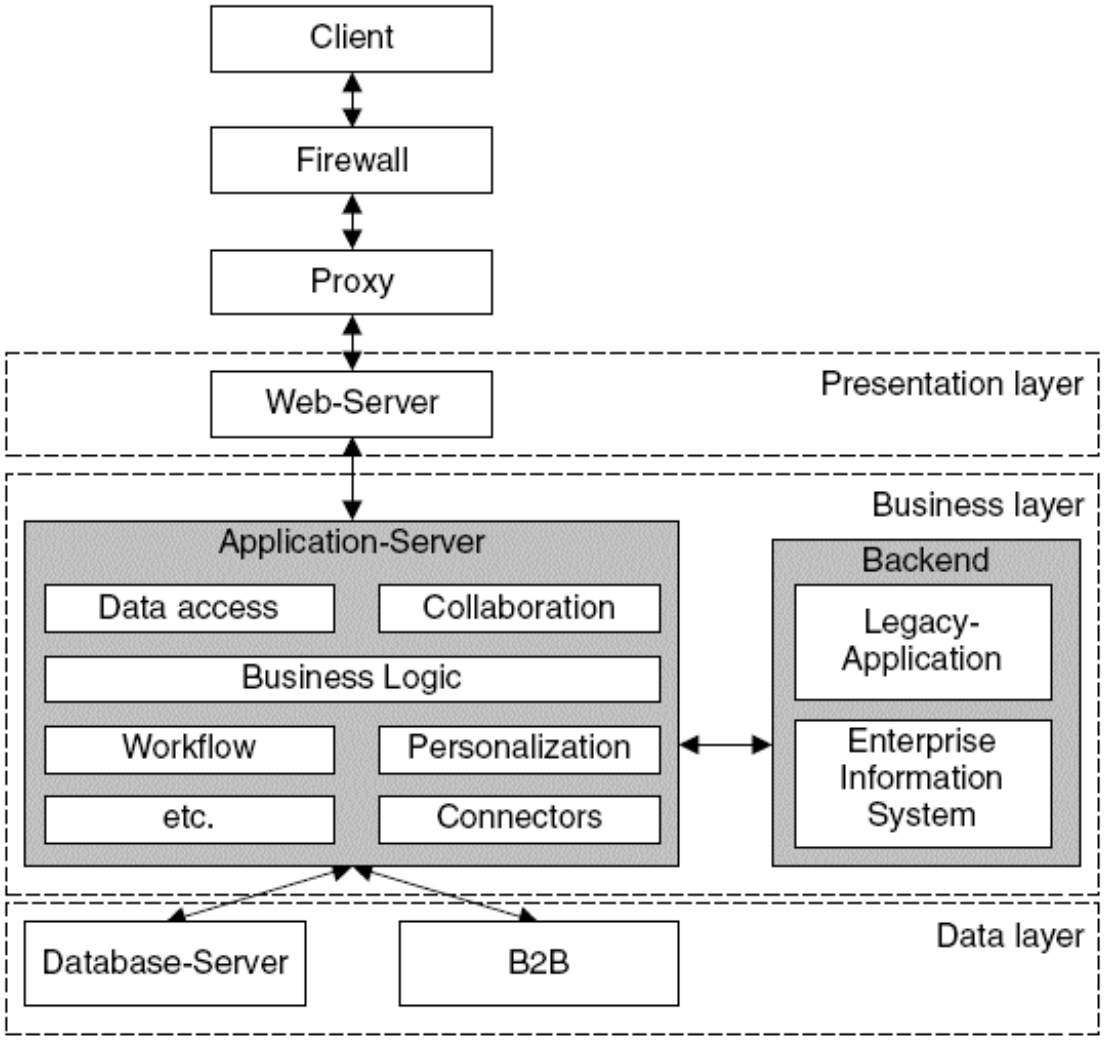
Las aplicaciones web manejan información de varios tipos: texto, gráficos, video, audio; por lo que hay retos al estructurar, procesar, almacenar y presentar la información.

***En tu opinión (y experiencia), ¿cuál crees que es el reto más difícil de resolver: estructurar, procesar, almacenar o presentar información? y ¿porqué?***

***En mi opinión el reto mas difícil es es la estructura ya que si logra estructurar de una buena forma es mucho mas sencillo lo siguiente tanto procesar, almacenar o presentar información si esta bien estructurado.***

*Modelo de capas:*

Arquitectura escalable a cualquier tamaño o propósito de una aplicación web, es el estándar en la mayoría de las aplicaciones web. Apodada 3-tier architecture o n-tier architecture aunque en otros contextos layer se refiere a la distinción lógica entre las partes integrales de una aplicación y tier puede significar las estructuras físicas donde corre una aplicación (red, computadoras, servidores).



La parte más visible y la que interactúa con el cliente: the View layer. La capa intermedia es aquella que sirve como intermediario entre la presentación (View Layer) y la información (Data Layer), contiene reglas, personalización, entre otros: the Business Layer. La última capa se encarga de almacenar la información utilizada en la aplicación web: the Data Layer.

Una arquitectura dividida en capas, hace más sencillo su construcción y mantenimiento, así como rehusar código y rastrear errores (debug).

*The View Layer*

*W3C: "A user wants to see the weather in her trip destination"*
*UI: "see weather in different periods of time or another places, and see pictures of it"*

Conocida como vista, presentación, UI; esta capa se encarga de presentar la información necesaria al usuario y las herramientas para interactuar con esa información.

Las tecnologías utilizadas en esta capa son:

- Markup procesado por el navegador: **HTML**
- Estilo de la página: **CSS**
- Scripts del lado del cliente: **Javascript/JQuery**

Aunque la generación de contenido puede provenir de la capa de lógica del negocio, la capa de presentación se encarga de presentarla y puede modificarla: para implementar paradigmas interesantes de interacción con el usuario, para ser adaptable a cualquier dispositivo, ser accesible a cualquier persona, etc.

Después de presentar información, puede ser que el usuario a su vez responda. En ese caso, la capa de presentación pasa esta información de regreso a la capa de lógica del negocio, no dejando decisiones a la capa de presentación. El traspaso de información (enviar peticiones y recibir respuestas) puede ser asíncrono o síncrono.

### *¿Cuál es la diferencia entre peticiones síncronas y asíncronas?, ¿para qué utilizarías cada tipo de petición?*

***Las peticiones síncronas son cuando el cliente realiza una consulta y el servidor responde esa consulta mientras que las peticiones asíncronas permiten una notificación al cliente cuando un evento ocurre en el servidor . Se pueden utilizar las peticiones asíncronas cuando es necesario la actualización de datos en pantalla ya que la síncrona existiría un lapso de tiempo en el que estarían desactualizados.***

*The Business Logic Layer*

*W3C: "A user wants to see the weather in her trip destination"*
*UI: *takes the request and sends it to the server**
*Business Logic layer: *takes the request and makes calls for the lower layer* … *WAITS* …*
*"The weather in Pachuca is fair currently, with a high temperature of 22º and a low of 7º."*

Se encarga de la lógica del programa, recibe información de la capa de presentación y la transforma de acuerdo a reglas establecidas y recibe información de la capa de datos y la utiliza para formar dichas reglas.

Contiene la parte determinante de la lógica de la aplicación:

- Ejecuta todos los cálculos y validaciones
- Administra el flujo de trabajo:
  - Administración del estado: mantener registro de la ejecución de la aplicación
  - Administración de la sesión: distinguir diferentes instancias de la aplicación
  - Identificación del usuario
  - Acceso del servicio: proveer servicios de la aplicación de manera consistente
- Administra los accesos a información solicitados en la capa de presentación

La capa de la lógica del negocio se encuentra implementada, por lo general, en un servidor de aplicación que automatiza los servicios como transacciones, seguridad, persistencia, *connection pooling*, mensajes, DNS.

Las herramientas utilizadas en esta capa son:

- Server-side scripts: Ruby, Node.js (JS para el servidor) PHP, ASP .NET, CGI scripts like Perl

*The Data Layer*

*W3C: "A user wants to see the weather in her trip destination"*
*UI: *takes the request and sends it to the server**
*Business Logic layer: *takes the request and makes calls for the lower layer**
*Data layer: *searches for the data using the given parameters and returns it**

Se encarga de recuperar información de sus fuentes. En un buen diseño de capas, forma una parte esencial del intercambio y reusabilidad de las tecnologías ya que se podría cambiar la fuente de los datos por otra pero la información genuina debería ser la misma.

Las tecnologías utilizadas en esta capa son:

- Database management systems: base de datos relacionales como MySQL, PostgreSQL, base de datos no relacionales como MongoDB, Apache Cassandra, plain XML o archivos de texto
- Mecanismo para realizar queries y recuperar la información:
  - API para los sistemas de administración
  - Scripts para los archivos

***KAHOOT!***
***Genera dos preguntas acerca del tema con cuatro respuestas posibles:***

***Cual no es una característica de una aplicación nativa:***

- ***Utilizar de forma optima el sistema operativo***
- ***Se publican en tiendas de distribución***
- ***En su mayoría, no necesitan estar conectadas a Internet***
- ***Se puede ocupar en cualquier dispositivo sin importar el SO***

***Capa que se encarga de presentar la información necesaria al usuario y las herramientas para interactuar con esa información***

- ***View Layer***
- ***Data Layer***
- ***Business Layer***
- ***Design Layer***

**Bibliografía** ***( agrega bibliografía extra )***


- W3C (2016) Web design and applications. https://www.w3.org/standards/webdesign/
- Dr. Adamkó, Attila (2014) Layered Architecture of Web Applications. https://gyires.inf.unideb.hu/GyBITT/08/ch04.html
- Pimienta, P. (2014, May 5). Tipos de aplicaciones móviles y sus características. In ZENVA. Retrieved from https://deideaaapp.org/tipos-de-aplicaciones-moviles-y-sus-caracteristicas/
- Madrigal, S. (2014, March 24). Aplicaciones síncronas y asíncronas. In SERGIOMADRIGAL.COM. Retrieved from http://www.sergiomadrigal.com/2014/03/24/
- aplicaciones-sincronas-y-asincronas/