

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?

Una aplicación nativa es siempre mejor que una web para dispositivos móviles. Las aplicaciones nativas tienen mejor rendimiento a comparación de una web, pueden verse muy similares y en algunos casos actuar casi de forma idéntica, pero siempre va a haber casos en los que la aplicación web sea lenta y se necesite de una nativa. Sin embargo para aplicaciones nativas necesitas desarrollar para cada sistema operativo (iOS, Android) y esto puede alargar el tiempo de desarrollo, mientras que en una web se obtiene cierta portabilidad ya que solo se necesita desarrollar una sola aplicación. (Y luego puedes utilizar React Native).

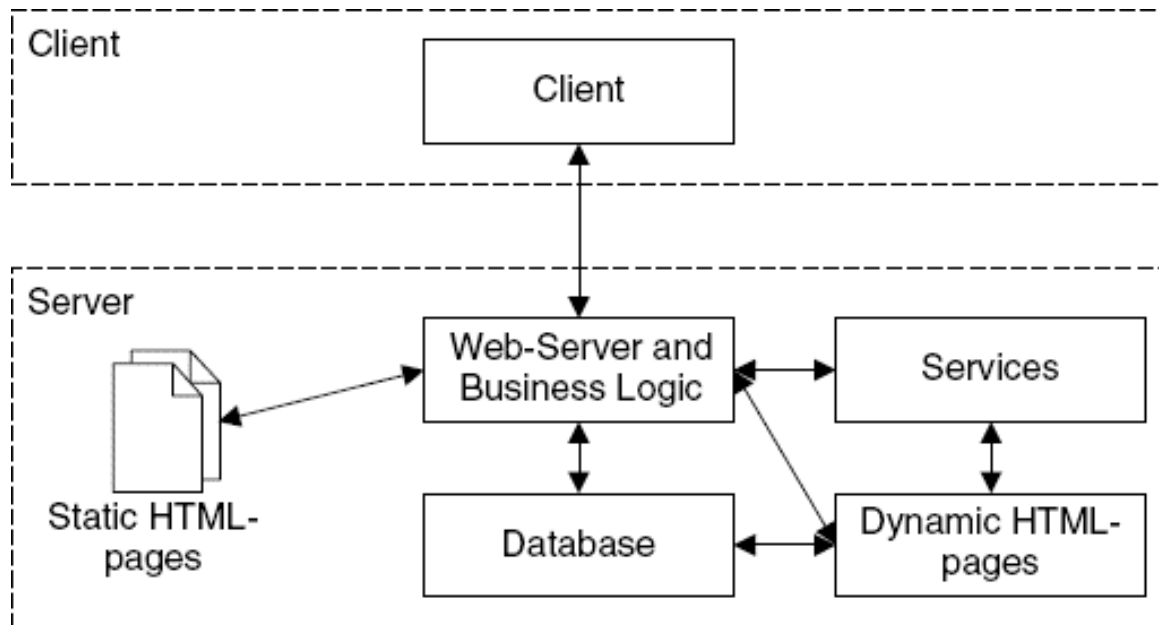
Historia

Anteriormente, el contenido era en su mayoría estático y solo era interactivo por contener hiper vínculos. Sin embargo 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?

El desarrollo de aplicaciones web es multidisciplinario porque necesitas gente que se dedique a tareas en específico. Entre estás está la parte administrativa que se encarga conseguir clientes y hablar con ellos, ventas, marketing, entre otras. Está la parte de diseño que se encarga de cómo se van a ver las interfaces y de qué manera se va a interactuar con ellas. En cuanto a desarrollo está la parte de *front end* y *back end*, y así mismo está la parte de seguro de calidad que se encarga de verificar que la aplicación se comporte como debe. En pocas palabras, el desarrollo de aplicaciones web es multidisciplinario porque se necesita de la participación de varias personas de diferentes áreas.

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

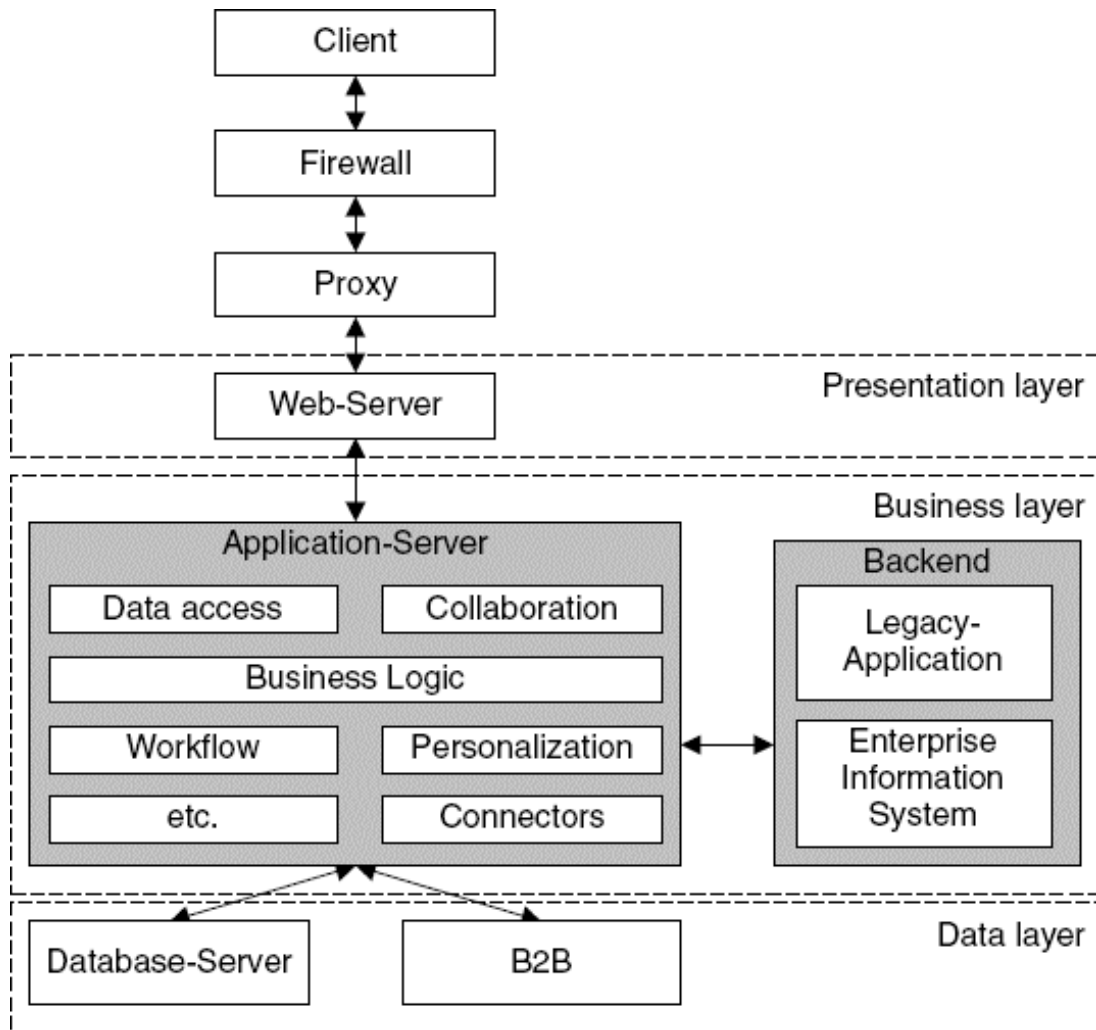
Las aplicaciones web manejan información de varios tipos: texto, gráficos, video, audio; por lo que hay retos al estructurar, procesar, almacenar y presentar la información.

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

Creo que el más difícil es el presentar la información (con la estructuración muy de cerca) ya que al final el usuario es quién va a utilizar la aplicación y si no le agrada la forma en que está presentada no la va a utilizar y tal vez la cambie por otra aplicación similar.

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: JS

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?

Una petición síncrona solo permite que una cosa se realice a la vez, entonces al realizar una petición, el navegador no continúa con ninguna otra acción hasta que se complete. En una petición asíncrona, se pueden correr otros comandos y esta tiene una llamada cuando los datos se han terminado de pedir, de esta forma el navegador trabaja de forma normal. Una petición asíncrona se puede ocupar cuando quieres cargar grandes cantidades de datos sin que se vea afectada la vista, una petición síncrona es raro que se utilice sin embargo podría utilizarse para precargar datos necesarios.

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 22° and a low of 7°."

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:

¿Qué hay en la capa de presentación?

- a) Application-server
- b) Web-server**
- c) Backend
- d) Database-server

¿Qué herramientas se utilizan en *The business logic layer*?

- a) HTML, CSS, JS
- b) MySQL, PostgreSQL
- c) Ember
- d) Ruby, Node js, PHP

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>
- InnovaAge. (2011). Apps Híbridas vs Nativas vs Generadas.
<https://www.innovaportal.com/innovaportal/v/696/1/innova.front/apps-hibridas-vs-nativas-vs-generadas-que-decision-tomar>
- Yescas, J. (2017). Solicitudes síncronas y asíncronas.
https://developer.mozilla.org/es/docs/Web/API/XMLHttpRequest/Synchronous_and_Asynchronous_Requests#Peticiones_as%C3%ADncronas
-
-
-