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

Web app

Is an app developed essentially for one particular mobile device and is installed directly onto the device itself. An example is the Camera+ app for Apple's iOS devices.

Native app

On the other hand native apps are basically internet- enabled apps that are accessible via the mobiles device's web browser. They need not to be downloaded onto de user's mobile device

The most important differences between them are:

- Native apps need a specific language to code. So, the development process is different for each mobile device. Web apps, on other hand, uses languages such as JavaScript, HTML 5, CSS3 or other web application frameworks.
- There are not standardized SDK development tools for web applications. Anyways there are several tools ad frameworks available to web developers.
- Web apps updates itself without the need for user intervention, but native apps need updates to be downloaded by the user and applied to the current app version.
- Native apps are more expensive to develop. However they are faster and more efficient, as they work paired with the mobile device.
- Web apps may result in higher costs of maintenance, also there are not specific authority to control quality standards, compare to native apps which are regulated by app stores and brands.

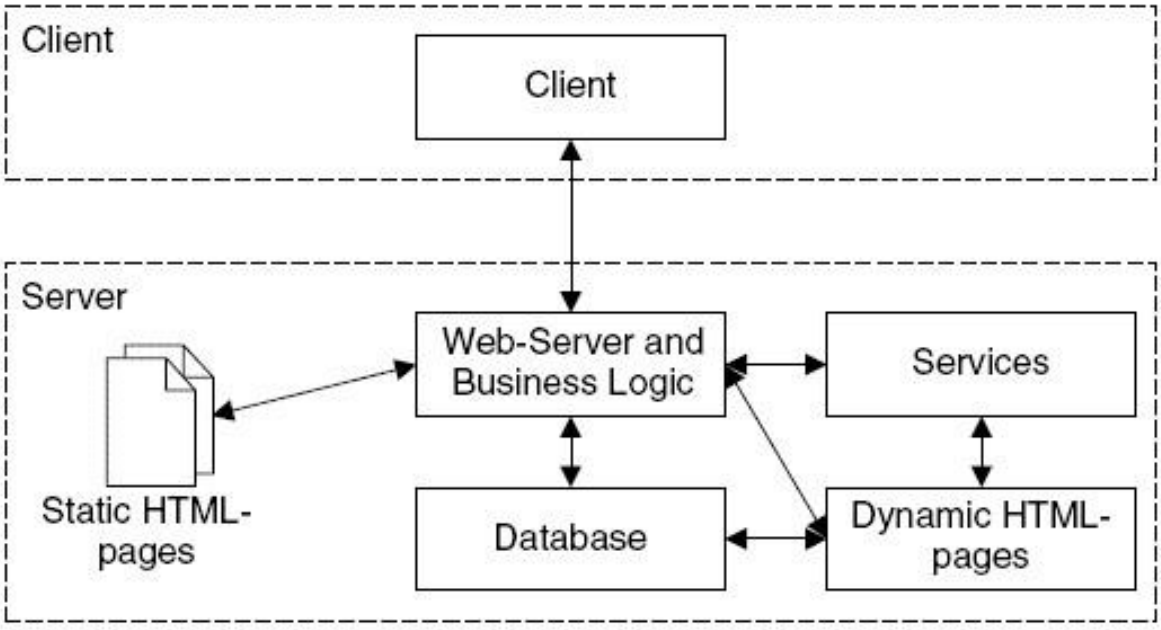
Historia

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 primera capa, existen varios lenguajes de programación o scripting que procesan, modifican o generan información dinámicamente e incluso proveen una interfaz de usuario.

El desarrollo web no es una tarea de solo programadores, siempre es importante tener en cuenta que para desarrollar necesitamos un equipo que domine diferentes disciplinas, además de que, en el caso de ser una aplicación de mucho trabajo, requiere de administración, asignación de recursos, tareas y sobre todo planeación.

Por eso es por lo que el desarrollo se considera como multidisciplinario. Requiere de muchas disciplinas trabajando en equipo para alcanzar el mismo objetivo. Un programador jamás lo lograría sin diseñadores, contadores, administradores de proyecto, diseñadores de interfaz y experiencia de usuario, incluso se necesita acercarse con expertos en el área de interés(desde químicos hasta artistas).

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.

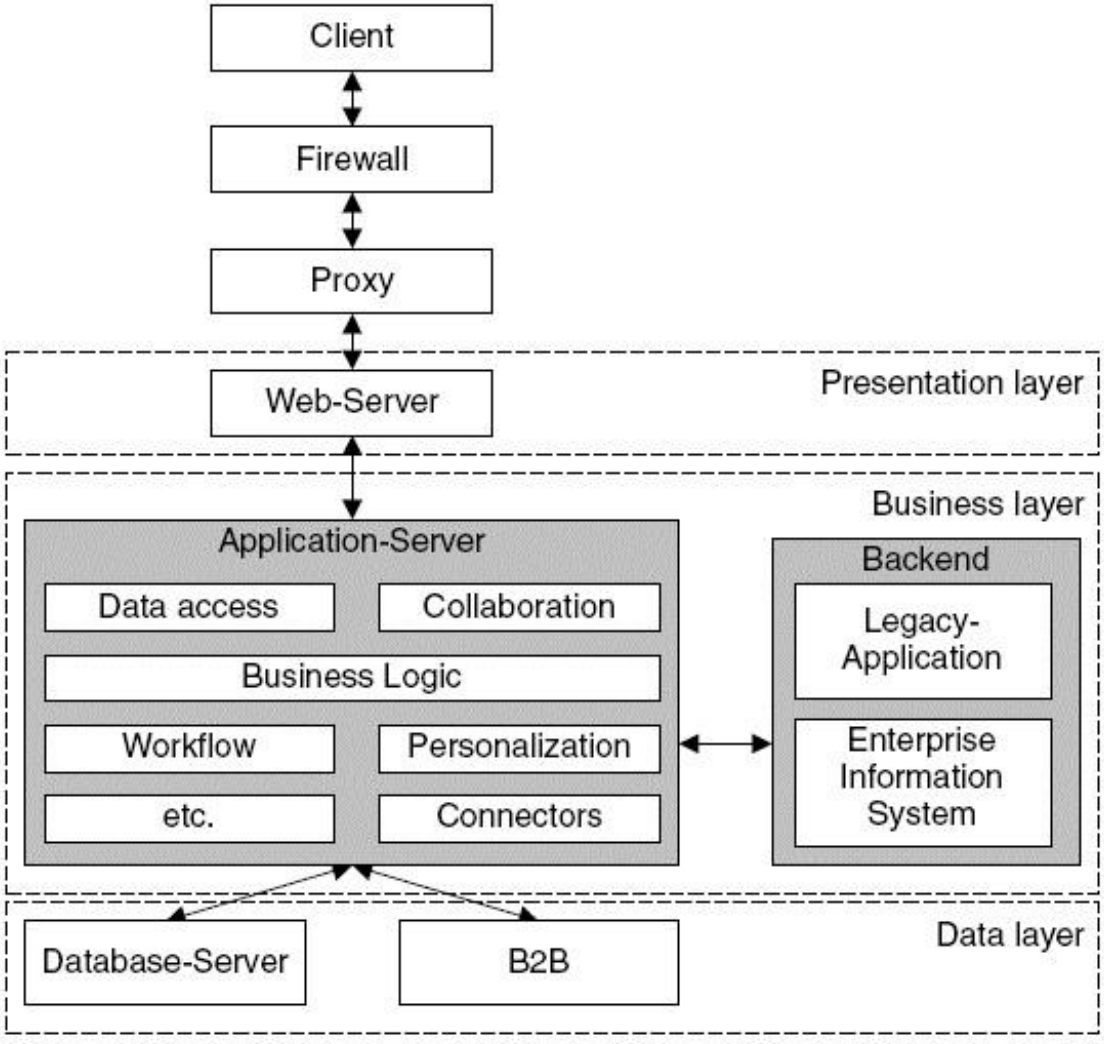
¿Estructurar, procesar, almacenar o presentar información?

Con la poca experiencia que tengo en desarrollo, puedo afirmar que estructurar es la parte más difícil. ¿Por qué? Porque la estructura, siendo a veces la parte más imperceptible en el resultado final, es usualmente la base de la experiencia de usuario y de la creación de las rutas. Por ende, de ella depende el éxito y aceptación por parte del público.

Además, la estructura de una aplicación sienta las bases de lo que será todo nuestro desarrollo, por lo que un error en esta, la capa más básica de la aplicación puede significar más recursos para continuar.

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

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.

Peticiones asíncronas y síncronas

Las peticiones síncronas son las que se realizan cada cierto periodo de tiempo al servidor para poder obtener la versión más actualizada de los datos. En cambio, las asíncronas son peticiones que permiten que la información se mantenga actualizada ya que notifican al cliente cuando un evento ocurre en la parte del servidor. Para permitir un comportamiento asíncrono la comunicación se produce en paralelo con el flujo principal.

Algunas aplicaciones de las peticiones asíncronas son: XML http Request, peticiones POST, carga asíncrona de archivos, JSON, promises, etc. Y un ejemplo serían un sistema de compras, el cual necesita interactuar con múltiples servicios web de suministradores al mismo tiempo, para así buscar el suministrador que le ofrezca el precio más bajo y la entrega más temprana. En cambio una petición síncrona debería ser realizada en una bolsa de valores, en verificaciones de seguridad, en cosas que dependan de actualizaciones permanentes.

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

Preguntas

¿Cuál es la capa que puede ver el usuario final en un explorador de computadora

a) The view layer
b) The business layer
c) The data layer
d) The software layer

¿Cuántas capas tiene por convención en la arquitectura de software una aplicación?

a) 4
b) 3
c) 6
d) 2

Bibliografía

- 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
- -Viswanathan, P. (2019). Should You Develop a Native App or a Web App?. Retrieved from https://www.lifewire.com/native-apps-vs-web-apps-2373133
- Madrigal, S. (2019). Aplicaciones síncronas y asíncronas - Sergio Madrigal Blog. Retrieved from http://www.sergiomadrigal.com/2014/03/24/aplicaciones-sincronas-y-asincronas/
- -Procesos BPEL síncronos y asíncronos. (2019). Retrieved from http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion04-apuntes.html

-

-