Web Application Architecture

A web application is just like a normal computer application except that it works over the Internet. As everyone is on the web these days, most developers are looking to benefit from web apps and attract as many users as possible via opportune offerings.

What is Web Application Architecture?

The web application architecture describes the interactions between applications, databases and middleware systems on the web. It ensures that multiple applications work simultaneously.

Let us understand it with a simple example of opening a webpage.

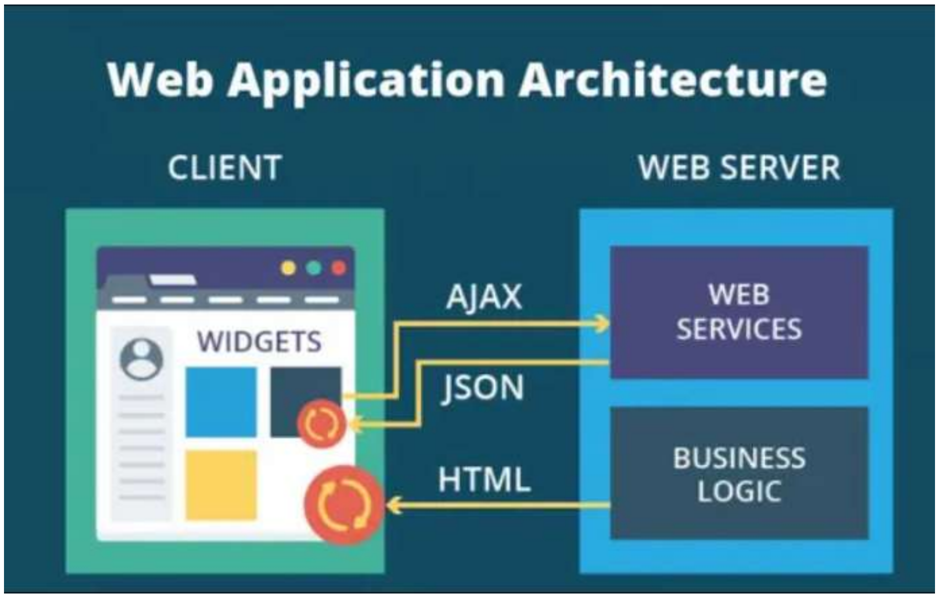
As soon as the user hits the go button after typing a URL in the address bar of a web browser, it requests for that particular web address. The server sends files to the browser as a response to the request made. The browser then executes those files to show the requested page.

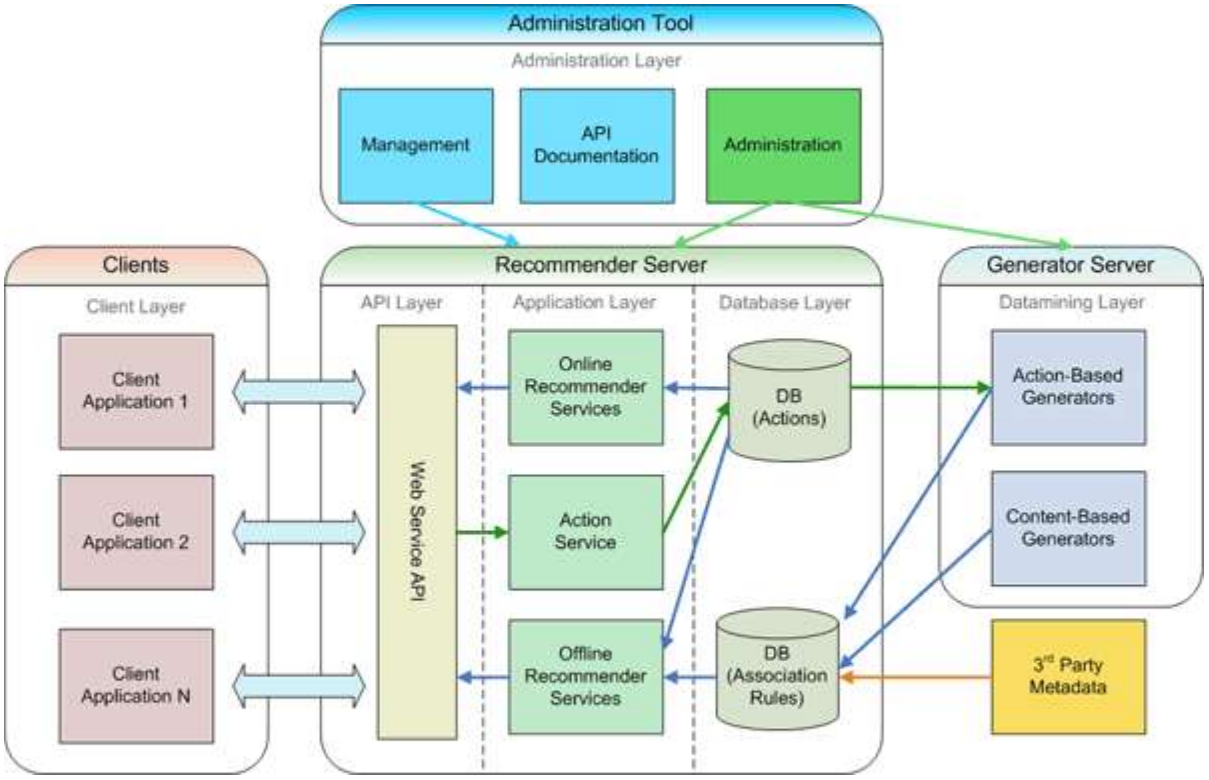
Finally, the user is able to interact with the website. The most important thing to note here is the code parsed by the web browser. A web app works in a similar way.

This code might or might not have specific instructions that tell the browser how to respond with respect to the different types of user inputs.

Hence, a web application architecture has to include all the sub-components as well as the external applications interchanges for the entire software application.

A web application architecture has to not only deal with efficiency, but also with reliability, scalability, security and robustness.





How Does It Work?

With any typical web application, there are two different codes (sub-programs) running side-by-side. These are:

Client-side Code - The code that is in the browser and responds to some user input

Server-side Code - The code that is on the server and responds to the HTTP requests

A web developer (team) developing the web application decides as to what the code on the server will do with respect to the code in the browser. For writing server-side code, C#, Java, JavaScript, Python, PHP, Ruby, etc. are used.

Any code that is able to respond to HTTP requests has the ability to run on a server. The server-side code is responsible for creating the page that the user requested as well as storing different types of data, including user profiles and user input. It is never seen by the end-user.

A combination of CSS, HTML, and JavaScript is used for writing the client-side code. This code is parsed by the web browser. Unlike the server-side code, client-side code can be seen as well as modified by the user. It reacts to user input.

The client-side code communicates only via HTTP requests and is not able to read files off a server directly.

Web Application Components:

When we say web application components, we can mean any of the following two:

UI/UX Web Application Components – This includes activity logs, dashboards, notifications, settings, statistics, etc. These components have nothing to do with the operation of a web application architecture. Instead, they are part of the interface layout plan of a web app.

Structural Components – The two major structural components of a web app are client and server sides.

Client Component - The client component is developed in CSS, HTML, and JS. As it exists within the user's web browser, there is no need for operating system or device-related adjustments. The client component is a representation of a web application's functionality that the end-user interacts with.

Server Component - The server component can be build using one or a combination of several programming languages and frameworks, including Java, .Net, NodeJS, PHP, Python, and Ruby on Rails. The server component has at least two parts; app logic and database. The former is the main control center of the web application while the latter is where all the persistent data is stored.

Web service and API –

Web service:

- A Web service is a collection of open protocols and standards which are widely used for exchanging data between systems or applications.
- Software applications are written using various programming languages and running on multiple platforms. It allows you to use web services to exchange data over computer networks.

API: Application Programming Interface

- API is a software interface that allows two applications to interact with each other without any user involvement.
- A Web API is an application programming interface for the Web. A Browser API can extend the functionality of a web browser.

KEY DIFFERENCE

- Web service is a collection of open source protocols and standards used for exchanging data between systems or applications whereas API is a software interface that allows two applications to interact with each other without any user involvement.
- Web service is used for REST, SOAP and XML-RPC for communication while API is used for any style of communication.
- Web service supports only HTTP protocol whereas API supports HTTP/HTTPS protocol.
- Web service supports XML while API supports XML and JSON.
- All Web services are APIs but all APIs are not web services.

Types of Web Application Architecture

A web application architecture is a pattern of interaction between various web application components. The type of web application architecture depends on how the application logic is distributed among the client and server sides.

There are three primary types of web application architecture. Each one of them is explained as follows:

Single-Page Applications (SPAs) – Instead of loading completely new pages from the server each time for a user action, single page web applications allows for a dynamic interaction by means of providing updated content to the current page.

AJAX, a concise form of Asynchronous JavaScript and XML, is the foundation for enabling page communications and hence, making SPAs a reality. Because single-page applications prevent interruptions in user experience, they, in a way, resemble traditional desktop applications.

SPAs are designed in a way so that they request for most necessary content and information elements. This leads to the procurement of an intuitive as well as interactive user experience.

Microservices – These are small and lightweight services that execute a single functionality. The Microservices Architecture framework has a number of advantages that allows developers to not only enhance productivity but also speed up the entire deployment process.

The components making up an application build using the Microservices Architecture aren't directly dependent on each other. As such, they don't necessitate to be built using the same programming language.

Hence, developers working with the Microservices Architecture are free to pick up a technology stack of choice. It makes developing the application simpler and quicker.

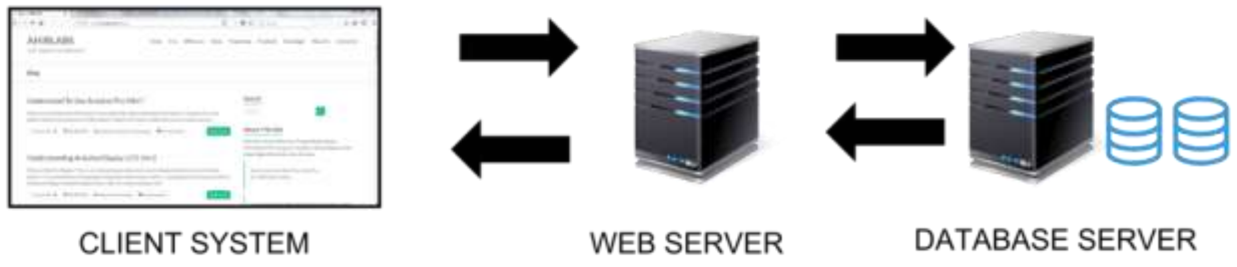
Serverless Architectures – In this type of web application architecture, an application developer consults a third-party cloud infrastructure services provider for outsourcing server as well as infrastructure management.

The benefit of this approach is that it allows applications to execute the code logic without bothering with the infrastructure-related tasks.

The Serverless Architecture is best when the development company doesn't want to manage or support the servers as well as the hardware they have developed the web application for.

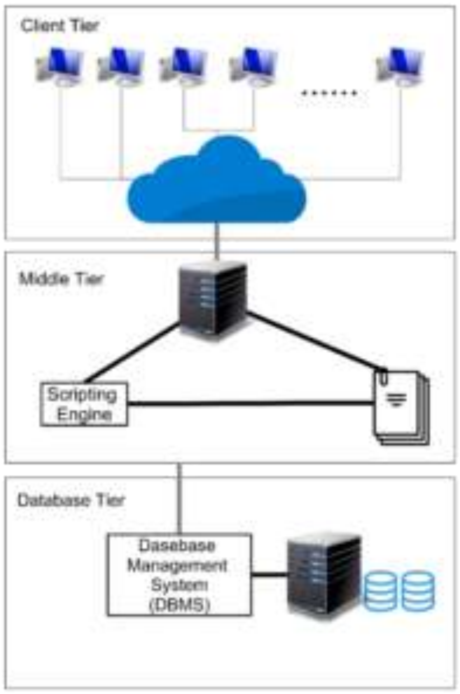
Web database Architecture

- Introduction
- Two tier architecture
- Three tier architecture



A three-tier architecture where a web browser requests a resource and a response is generated from a database

At the base of an application is the database tier, consisting of the database management system that manages the data users create, delete, modify, and query. Built on top of the database tier is the middle tier , which contains most of the application logic that you develop. It also communicates data between the other tiers. On top is the client tier , usually web browser software that interacts with the application.



The three-tier architecture model of a web database application