Web Application Fundamentals

Contents

Web Application Fundamentals

Web applications have developed as an integral element of our daily lives in the fast-paced digital world, affecting the way we communicate, work, and conduct business. Web apps power the modern online world, from simple e-commerce platforms to complex social networking sites and productivity aids. The deep workings of web application development, a dynamic and developing discipline that merges art and science, lie behind flawless user experience and functioning. Before we dive into the fundamentals of a web applications, Let's see what a web application is.

A web application is software that runs in your web browser. Businesses must exchange information and deliver services remotely. They use web applications to connect with customers conveniently and securely. The most common website features like shopping carts, product search and filtering, instant messaging, and social media newsfeeds are web applications in their design. They allow you to access complex functionality without installing or configuring software. These programs are available on a variety of platforms, including desktop computers, laptops, smartphones, and tablets, making them extremely adaptable and widely used.

Components of a Web Application

Web apps can be accessed from all web browsers and across various personal and business devices. Teams in different locations can access shared documents, content management systems, and other business services through subscription-based web applications.

Front End Technologies

Front-end and back-end technologies are often used to build web applications. The front end of a program is the part of it that users interact with directly in their web browsers. It consists of user interface (UI) design, layout, and client-side scripting languages such as HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript. These technologies allow the application to show information in an engaging and interactive way.

Back End Technologies

The back end oversees data processing, application logic, and interactions with databases or third-party services. Back-end technologies that are commonly used include:

Languages for server-side scripting like

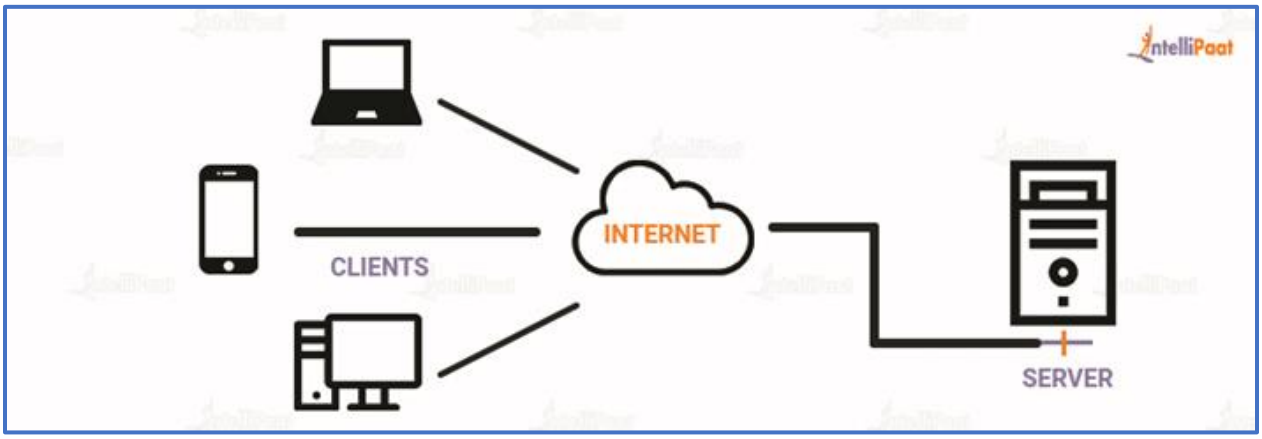
- PHP
- Python
- Ruby
- Java
- Node.js etc…

Databases such as

- MySQL
- OracleDB
- Redis
- SQLite
- Cassandra
- MariaDB
- DynamoDB
- Neo4j
- FirebirdSQL
- PostgreSQL
- MongoDB etc...

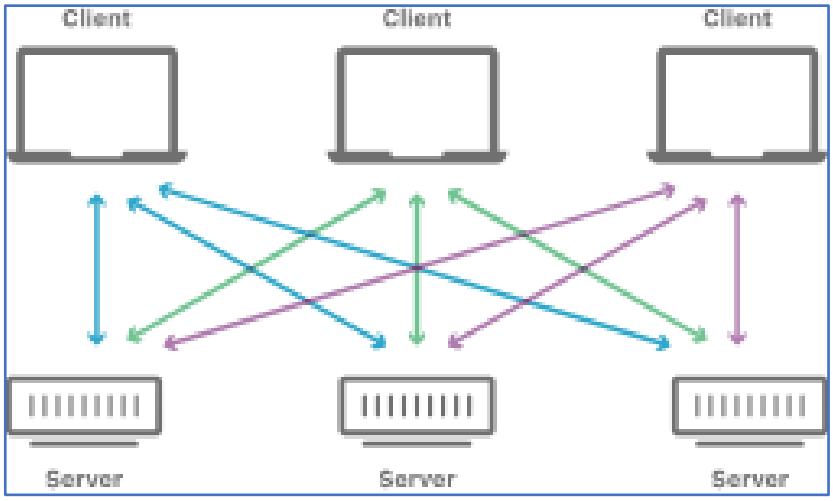
Client - Server Design

Web applications use a client-server design in which the client (the user's browser) sends requests to the web server for webpages email clients that connect to email servers to send and receive emails, and database management systems where clients communicate with database servers to query and update data., And the web server processes those requests and returns results to the client.



The client-server model is used because servers are typically more powerful and more reliable than user devices. They also are constantly maintained and kept in controlled environments to make sure they're always on and available; although individual servers may go down, there are usually other servers backing them up. Meanwhile, users can turn their devices on and off, or lose or break their devices, and it should not impact Internet service for other users.

Servers can serve multiple client devices at once, and each client device sends requests to multiple servers while accessing and browsing the Internet.



HTTP and APIs

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message. The request includes various request headers such as Host, User-Agent, Referrer etc.,

Example of a HTTP request header

```
GET /home.html HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referrer: https://developer.mozilla.org/testpage.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 20 Apr 2024 02:36:04 GMT
If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"
Cache-Control: max-age=0
```

And request methods and versions whereas the responses can be classified into various response codes.

An API, or application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality. APIs simplify and accelerate application by allowing developers to integrate data, services, and capabilities from other applications, instead of developing them from scratch. APIs also provides a simple, secure way to make the application data and functions available to departments with other organizations.

Version Control

Version control is a software development technique that lets multiple developers work on the same set of source code files. Each time they save a file, the version control system automatically saves a copy of the previous version, known as a "checkpoint." This way, if one developer accidentally deletes or overwrites another developer's work, or if someone accidentally erases all the files, the most recent copy can be restored.

Version control systems also let you see who changed what and when; roll back to previous versions; compare different versions (like when you're updating an existing website); and collaborate with other developers on projects that have grown too large for one person to handle alone. Version control also helps in case something goes wrong with your website - for example, if it stops working because of server issues or if there is malware on it. If you have a source control system in place, you can restore the site to its last known good state and then use that as your starting point for fixing things. This is especially important if you build websites for clients - they are going to need an easy way to get their website back up and running if something goes wrong.

Version control is one of the most useful tools that web developers use, especially if they are involved in large projects. It’s an important tool that helps developers manage their projects. It is a system that keeps track of the changes made to files and allows users to revert to previous versions if needed.

With version control, the code base can be managed and tracked easily. This means that any changes made will be recorded and can be traced back to see who made them and when they were made. It also helps to avoid conflicts between team members as each person's changes are kept separate from each other's changes.

Version control systems are also useful for people who need a quick way to find the most recent version of a file. If you've changed a document several times over the course of a week and want to see what it looked like at the beginning, it can be difficult to find out which version is the most current. A version control system makes this process much easier by storing multiple versions of files in one place so they're easy to access and compare. The importance of having version control cannot be overstated as it allows developers to work independently on the same project without fear of messing up each other's work or stepping on each other's toes.

Web Security

Web application security encompasses the processes, technologies, and methods to protect websites, web servers, web applications, and web services from external threats and internet-based attacks. Content management systems, APIs, and SaaS applications are examples of common cyberattack targets.

Web apps are like an online road that users can take to exchange information with your company. Emails, web forms, and any application that clients can use to interact with your company is a web application. With this kind of information avenue, creating and maintaining a high level of security is necessary, otherwise anyone can get into your company and take sensitive information without any roadblocks.

Web-based applications are targeted because they can contain a complex source code that can be hijacked with improper inputs and are externally available. Attackers try to override or overwhelm these systems with different styles of code "mutations". Injection type flaws allow attackers to input malicious code to reconfigure and manipulate the application, creating unexpected crashes or unintended outcomes that can provide a foothold for criminals to access sensitive data.

Cyberattacks against web applications can be easily automated and can be executed against thousands of targets simultaneously. The technique is to barrage them with bombardment of coding, much like a virtual battering ram, and try to crash web applications. Once they knock down the web application gateway with this style of injection attack, sensitive data can be exposed, and the company can be left hobbled from doing business.

The biggest reason to enable web application security is to protect the sensitive data your company has been entrusted with by your customers. If an attacker can destroy, alter, or steal private client data, it would create a wave of backlash for an organization. The loss of sensitive data and the cost of correcting a breach are hard to overcome, especially when the news of an attack reaches the public and business partners. Damaged reputation from a cyberattack can be nearly impossible to overcome. Customers and companies may not want to do business with an organization that's unable to secure their shared data.

Responsive Design

Responsive design can help you solve a lot of problems for your website. It will make your site mobile-friendly, improve the way it looks on devices with both large and small screens, and increase the amount of time that visitors spend on your site. It can also help you improve your rankings in search engines. This is important because Google now uses a mobile-first indexing approach, which means that it prioritizes the mobile version of your website when ranking it in search results.

Testing and Debugging

Testing helps identify and rectify errors in HTML code, ensuring adherence to web standards and consistent rendering across different browsers and devices. Debugging allows developers to pinpoint and fix issues, such as syntax errors or misaligned elements, preventing potential disruptions to the user experience.

Performance Optimization

One of the most obvious and important benefits of optimizing web application performance is enhancing the user experience. Users expect web applications to load fast, respond smoothly, and work reliably, regardless of their device, network, or location. A slow or unstable web application can frustrate users, reduce their engagement, and increase their bounce rate. On the other hand, a fast and reliable web application can delight users, increase their retention, and boost their conversion rate. Optimizing web application performance can help you deliver a better user experience by reducing the load time, improving the interactivity, and ensuring the availability of your web application.

Another benefit of optimizing web application performance is improving your search engine optimization (SEO) ranking. Search engines, such as Google, use web application performance as one of the factors to rank web pages in their search results. A faster and more responsive web application can rank higher than a slower and less responsive one, which can increase your visibility, traffic, and authority. Optimizing web application performance can help you improve your SEO ranking by following the best practices and guidelines recommended by search engines, such as using HTTPS, minimizing, and compressing your resources, and implementing lazy loading and caching.

A third benefit of optimizing web application performance is reducing your operational costs. A poorly performing web application can consume more resources, such as bandwidth, memory, CPU, and disk space, than a well-performing one, which can increase your hosting, maintenance, and support costs. A poorly performing web application can also cause more errors, bugs, and downtime, which can affect your reputation, customer satisfaction, and revenue. Optimizing web application performance can help you lower your operational costs by optimizing your code, architecture, and infrastructure, and by using tools and techniques to monitor, analyze, and debug your web application.

Continuous Integration and Deployment

Continuous Integration and Deployment allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method. In the ever-evolving world of web development, delivering high-quality software rapidly and consistently is paramount. Continuous Integration and Deployment (CI/CD) has emerged as a game-changer, enabling developers to automate and streamline the software development pipeline.

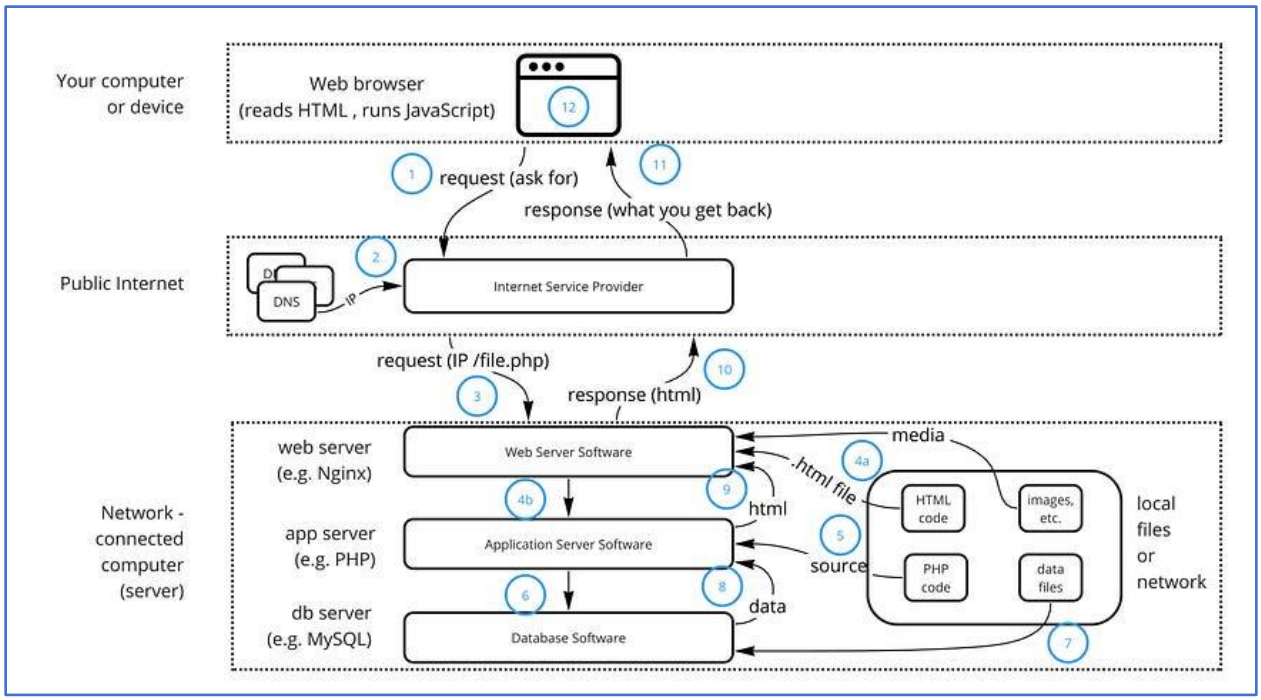
Here's a breakdown of the CI/CD process:

Continuous Integration (CI): Developers frequently commit their code changes to a shared repository. Automated CI tools, such as Jenkins, Travis CI, or CircleCI, continuously build and test the code as new changes are pushed. The goal is to identify and fix integration issues early in the development cycle.

Continuous Delivery (CD): After successful CI, the code is automatically deployed to a staging or pre-production environment. In a CD pipeline, further testing, including performance and user acceptance testing, is performed. If all tests pass, the code is ready for deployment to production.

Continuous Deployment: In this scenario, if all tests pass in the CD stage, the code is automatically deployed to the production environment without manual intervention. This is the goal for many organizations looking to achieve rapid and reliable software releases.

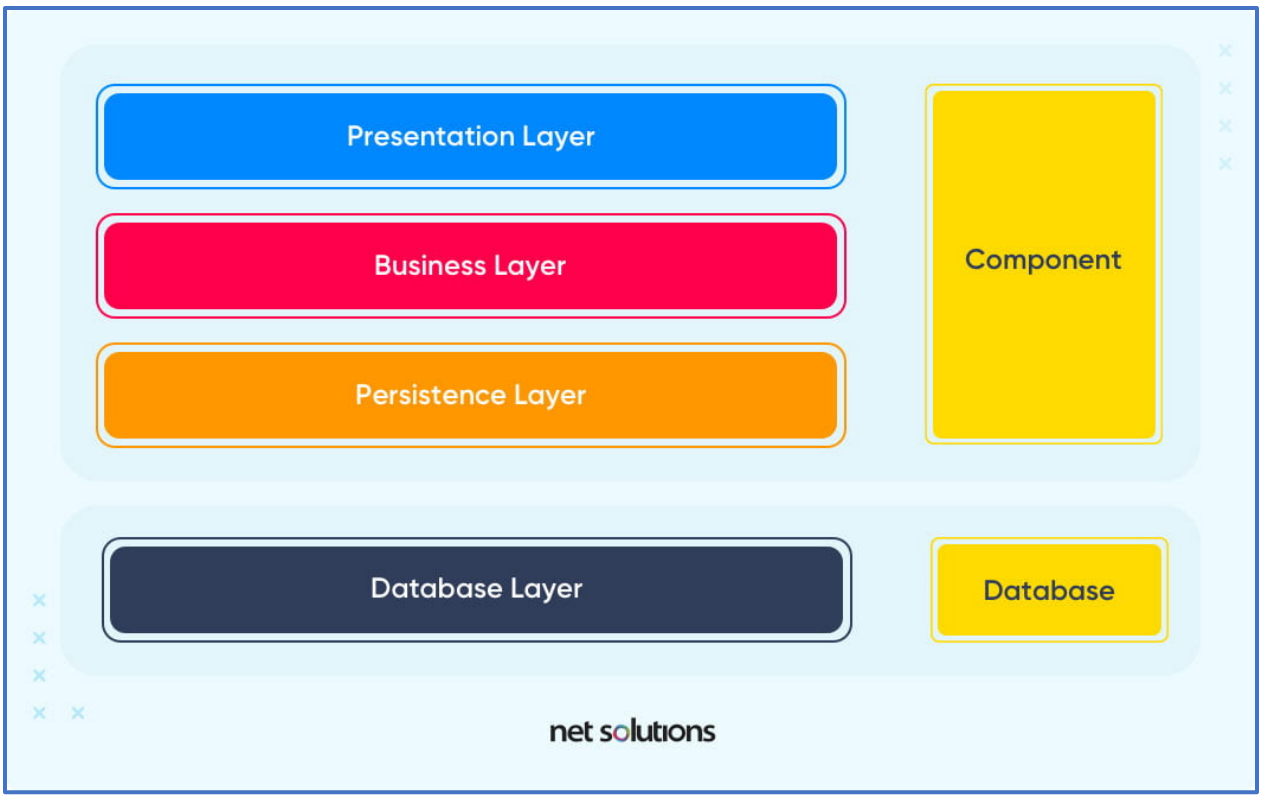
How Web Applications Work



- Your device or computer is connected to the Internet and your browser requests a URL (test.com/about-us.php)
- Your device connects to the public Internet via your ISP (broadband or cellular provider) and tries to find the address for test.com via DNS lookup which is like a phone book for network addresses to computers connected to the Internet.
- It allows a look up by [domain] name and returns an IP address like 187.64.34.56, just like your name might return your phone number in a phone book. This is the network address of the computer that is hosting that domain.
- Your request is transmitted through the Internet to the IP address you found similar to typing that number in your phone and placing a call.

- The computer connected to that IP address would have web server software running on it that will go fetch the requested file and send it back.
- if just a .html file, the server finds it on the computer and sends it back.
- if a .php file, the web server knows to hand off the request to the application server that will process the PHP code.
- The application server receives an HTTP Request from the web server, finds the file with the source code (programming instructions) for that web address (i.e., about-us.php), and executes your PHP program.
- The program references a connection to a database server (just another program run on the computer) and sends it a "query" to request data stored and organized by the database server.
- The database server fetches the files requested from the query.
- The database server sends back the requested data to the application server.
- The application server combines this data with any other data your program instructs it to add and sends it all back to the web server as an HTTP Response. Typically, you just output HTML-formatted text, but your program can inject data into it as it loops, etc.
- The web server sends back the HTML text to the requesting connection.
- The data is transmitted over the Internet back to your computer or device and the connection ends.
- Your device browser reads the HTML code (that was generated) and displays it on the screen.

Architecture of a Web Application



Presentation Layer

The presentation layer manages the app user interface, dealing with UI components and process components such as HTML, CSS, and JavaScript. It also receives user input and sends it to the business layer for processing, interacting through APIs or interfaces. The presentation layer typically includes web components such as controllers, views, and templates.

Business / Application Layer

The business (or application) layer handles the web application's business logic, Error Handling, Record handling etc. It contains controllers, services, and models responsible for performing the necessary actions to fulfill user requests. The business layer interacts with the data access layer to retrieve or manipulate data as needed.

Persistence / Data Access Layer

The persistence (or data access) layer translates application data into a format that can be stored and retrieved from a data store. It contains the components that interact with the database, such as data access objects (DAOs), object-relational mappers (ORMs), and stored procedures.

Database Layer

The database layer includes the database management system (DBMS) and the data stored in the database. This layer stores data in a structured format that can be easily queried and manipulated by the data access layer.

The overall summary of how the layers work together:

- The user interacts with the presentation layer by providing input through the user interface.
- The presentation layer receives the user input and sends it to the business layer.
- The business layer processes the user input, performs the necessary actions, and retrieves or updates data through the data access layer.
- The data access layer retrieves or updates data from the database layer and sends it back to the business layer.
- The business layer processes the retrieved data and generates a response to the presentation layer.
- The presentation layer receives the response from the business layer and updates the user interface accordingly.
- The process repeats as the user provides further input or navigates through the application.

Types of Web Applications

Single-Page Applications

The web application loads only once and then dynamically updates the content as the user interacts. The data is loaded asynchronously through APIs, making it more responsive and reducing the server's load. This architecture is suitable for applications that require a lot of user interaction and real-time data updates.

Examples: Gmail, Trello, Spotify, and Twitter.

Multi-Page Applications

Multi-page applications are web applications that consist of many HTML pages. Each page displays different content that must be refreshed each time a user interacts with it.

MPAs have been around since the early days of the web and are still widely used today. They are especially common in content-heavy websites, such as news sites or e-commerce sites, where each page represents a different piece of content or product.

Examples: Amazon, The New York Times, Wikipedia, and eBay.

Server-Side Rendered Applications

In Server-Side Rendered Application, the server generates HTML pages for each request, and the client only receives the result. This architecture is suitable for applications that require fast loading times and good SEO, but it can be slower than Single Page Applications.

Examples: WordPress, Airbnb, Shopify.

Progressive Web Applications

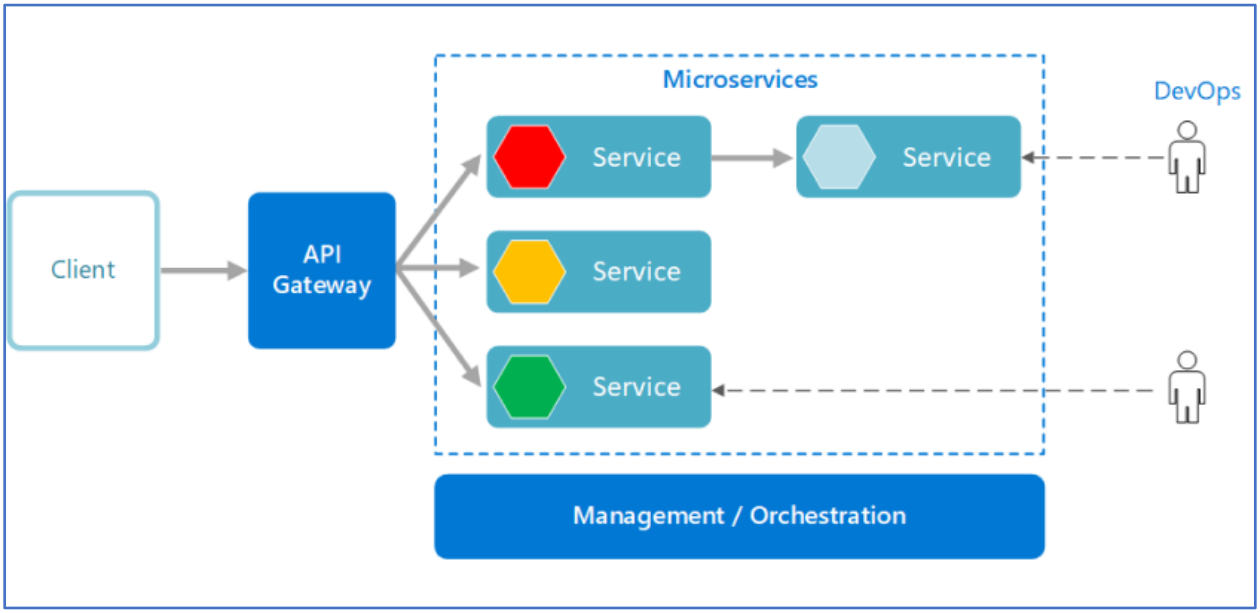
A Progressive web application behaves like a native mobile application, with features like offline access, push notifications, and full-screen mode. This architecture is suitable for solutions that need to be accessible on mobile devices and have the same user experience as native applications.

Examples: Starbucks, Pinterest, Forbes, Uber, and Twitter Lite.

Microservices

In a microservices architecture, the backend is divided into small, independent services that communicate with each other through APIs. Each service is responsible for a specific function: authentication, payments, or messaging. This highly scalable architecture allows for more granular control over individual components but can be complex to manage.

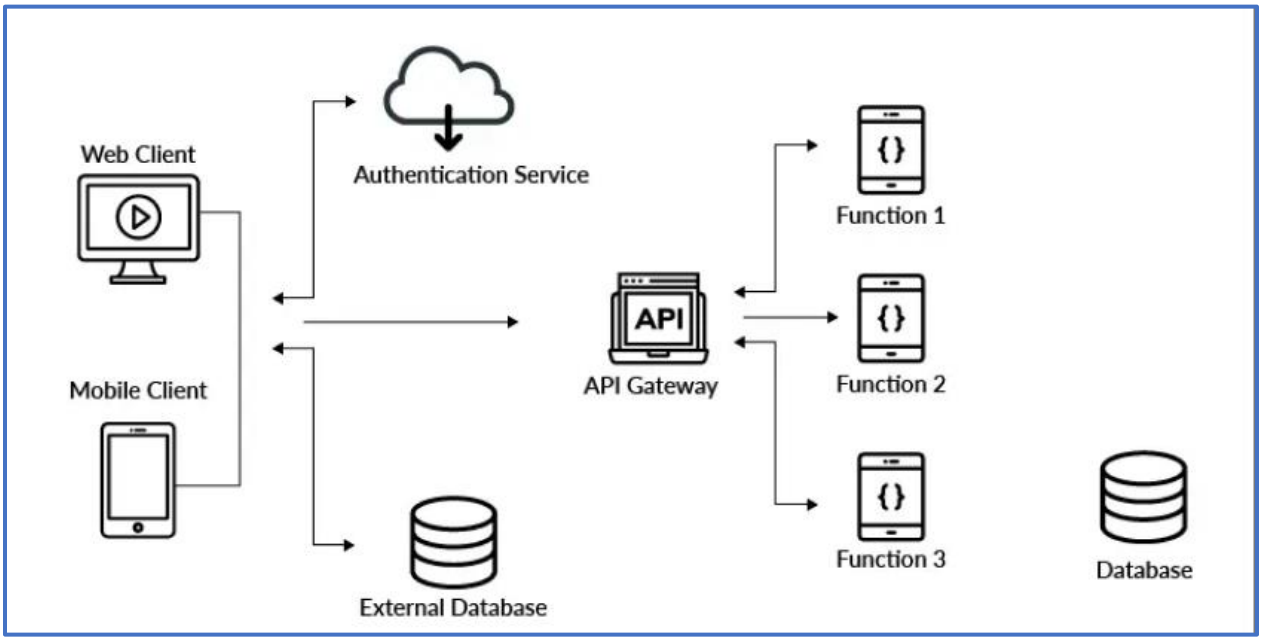
Examples: Netflix, Amazon, Uber, and Airbnb.



Serverless Architecture

In this architecture, the backend is built using cloud-based solutions, such as AWS or Azure. Each function is responsible for tasks like registering users or sending email notifications. It is highly scalable and cost-effective but difficult to manage and debug.

Examples: Coca-Cola, Capital One, The New York Times, and Fender.



Protocols in Web Applications

When two parties are communicating for example two people talking to each other, they need to use the same language and set up grammar rules so that they can understand each other. Similarly in computers, when two applications are communicating they need to use the same set of rules which we call it as protocol.

Let's see some of the common protocols used in a web application.

Hypertext Transfer Protocol [HTTP]

HTTP is a protocol for fetching resources such as HTML documents. It is the foundation of any data exchange on the Web, and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is reconstructed from the different sub-documents fetched, for instance, text, layout description, images, videos, scripts, and more.

Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called requests and the messages sent by the server as an answer are called responses.

File Transfer Protocol [FTP]

The File Transfer Protocol is a standard communication protocol used for the transfer of files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server.

Domain Name System Protocol [DNS]

It translates the domain name into IP addresses, and these translations are included within the DNS. Every host is identified by the IP address but remembering numbers is very difficult for people also the IP addresses are not static therefore a mapping is required to change the domain name to the IP address. So, DNS is used to convert the domain name of the websites to their numerical IP address.

Simple Mail Transfer Protocol [SMTP]

Simple mail transfer protocol (SMTP) is defined as an email protocol that enables the transmission of emails among user accounts over an internet connection. In other words, SMTP is a set of rules that allows different email accounts and clients to streamline information exchange. SMTP is a push protocol and is used to send the mail whereas POP (post office protocol) or IMAP (internet message access protocol) is used to retrieve those emails at the receiver's side.

Hypertext Transfer Protocol Secure [HTTPS]

HTTPS is a secure version of HTTP that ensures data transfer integrity by leveraging SSL and TLS encryption methods. HTTPS uses public and private keys to encrypt and decrypt data, making data transmitted over HTTPS secure from eavesdropping and tampering.

Certificates: HTTPS uses digital certificates to establish the identity of a website, across the web browser, and the server. These certificates are usually issued by certificate authorities (CAs), who perform identity validation before issuing a certificate.

Secure File Transfer Protocol [SFTP]

Secure File Transfer Protocol (SFTP) is a network protocol that enables secure and encrypted file transfers between a client and a server. It is designed to provide a secure alternative to the traditional File Transfer Protocol (FTP) by incorporating Secure Shell (SSH) for authentication and data encryption.

Real-Time Transport Protocol [RTP]

The Real-time Transport Protocol is a network protocol used to deliver streaming audio and video media over the internet, thereby enabling the Voice Over Internet Protocol (VoIP).

HTTP Request Methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data. The purpose of the GET method is to simply retrieve data from the server. The GET method is used to request any of the resources like

- A webpage or HTML file
- An image or video
- A JSON document.
- A CSS or JS file

The GET request method is said to be a safe operation, which means it should not change the state of any resource on the server.

POST

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server. It sends data to the server for processing. The data sent to the server is like

- Input Fields from the online forms.
- XML or JSON data.
- Text data from query parameters.
- Post a message to a bulletin board.
- Save data from the forms to a database.
- Calculate a result based on the data submitted.

A POST operation is not considered a safe operation, as it has the power to update the state of the web resource and cause potential side effects to the server's state when executed.

HEAD

The HEAD method simply returns metadata about a resource on the server. This HTTP request method returns all the headers associated with a resource at a given URL but does not actually return the resource.

It is commonly used to check the following conditions:

- The size of a resource on the server.
- If a resource exists on the server or not.
- The last-modified date of a resource.
- Validity of a cached resource on the server.

Example

```
HTTP/1.1 200 OK
Date: Fri, 20 Apr 2024 12:03:00 GMT
Content-Type: text/html
Content-Length: 1562
Last-Modified: Thu, 19 Apr 2024 15:30:00 GMT
```

PUT

The HTTP PUT method is used to completely replace a resource identified with a given URL. The PUT request method includes two rules:

- A PUT operation always includes a payload that describes a completely new resource definition to be saved by the server.
- The PUT operation uses the exact URL of the target resource.

If a resource exists at the URL provided by a PUT operation, the resource's representation is completely replaced. If a resource does not exist at that URL, a new resource is created.

DELETE

The DELETE method deletes the specified resource from the server.

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource. The connect operation is used to create a connection with a server-side resource. The most common target of the HTTP method CONNECT is a proxy server, which a client must access to tunnel out of the local network.

RESTful API designers rarely interact with the CONNECT HTTP request method.

OPTIONS

The OPTIONS method describes the communication options for the target resource. The server does not have to support every HTTP method for every resource it manages. Some resources support the PUT and POST operations. Other resources only support GET operations. The HTTP OPTIONS method returns a listing of which HTTP methods are supported and allowed.

The following is a sample response to an HTTP OPTIONS method call to a server:

```
OPTIONS /example/resource HTTP/1.1
Host: www.example.com HTTP/1.1 200 OK
Allow: GET, POST, DELETE, HEAD, OPTIONS
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, OPTIONS
Access-Control-Allow-Headers: Authorization, Content-Type
```

TRACE

The TRACE method is used for diagnostics, debugging and troubleshooting. It simply returns a diagnostic trace that logs data from the request-response cycle. The content of a trace is often just an echo back from the server of the various request headers that the client sent.

PATCH

Sometimes object representations get very large. The requirement for a PUT operation to always send a complete resource representation to the server is wasteful if only a small change is needed to a large resource. The PATH method, added to the Hypertext Transfer Protocol independently as part of RFC 5789, allows for updates of existing resources. It is significantly more efficient, for example, to send a small payload rather than a complete resource representation to the server.

References

https://aws.amazon.com/what-is/web-application/

https://www.linkedin.com/pulse/fundamentals-web-application-development-manakanalyticsdevelopment

https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

https://reinvently.com/blog/fundamentals-web-application-architecture/

https://www.freecodecamp.org/news/http-request-methods-explained/

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/HTTP-methods