Web Application Architecture: How the Web Works

When building a web application, there are three main principles to bear in mind. From a customer’s point of view, the application should be simple, aesthetically pleasing, and address most of their problems. From the business aspect, a web application should stay aligned with its [product/market fit](https://www.altexsoft.com/blog/business/product-strategy-how-to-find-product-market-fit/). From a software engineer’s perspective, a web application should be scalable, functional, and able to withstand high traffic loads.

All these issues are addressed in the web application’s architecture. We’ll cover the basic concepts of any modern web application and explain how the architecture patterns may differ depending on the application you’re building.

## **What is Web Application Architecture?**

So, what is a web application and how is it different from a website?

The basic definition of a **web application** is a program that runs on a browser. It’s not a website, but the line between the two is fuzzy. To differentiate a web application from a website, remember these three formal characteristics. A web application:

* addresses a particular problem, even if it’s simply finding some information
* is as interactive as a desktop application
* has a Content Management System

A website is traditionally understood to simply be a combination of static pages. But today, most websites consist of both static and dynamic pages, which makes almost all modern websites – you guessed it! – web applications. In this article, we will use the terms interchangeably.

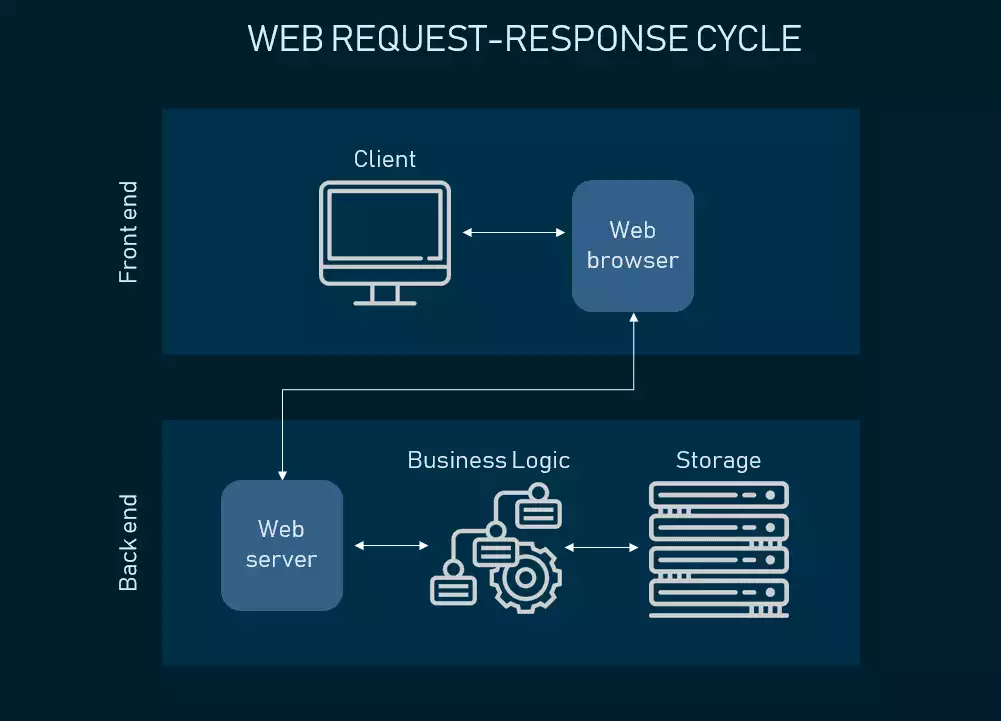
Your computer, or smartphone, or any other device you’re browsing with is called a client. The other half of the web equation is called a server because it serves you the data you request. Their communication is called a client-server model, whose main concern is receiving your request and delivering the response back.

**Web application architecture** is a mechanism that determines how application components communicate with each other. Or, in other words, the way the client and the server are connected is established by web application architecture.

Web applications of different sizes and complexity levels all follow the same architectural principle, but details may differ. We will further explain how a basic request-response process works and what components comprise the architecture.

## **How does the web request work?**

To understand the components of web application architecture, we need to understand how they are used in performing the most basic action – receiving and responding to a web request.

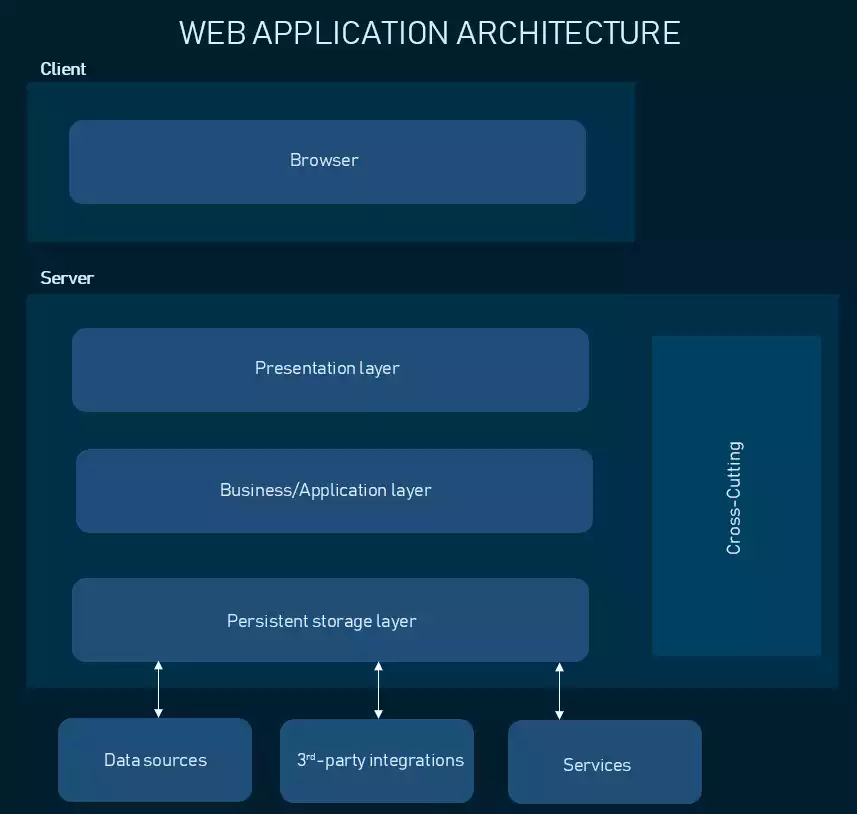


**First, you visit amazon.com**. You type in the URL and as you hit Enter, your browser prepares to recognize this URL, because it needs to know the address of the server where the page is located. So it sends your request to the Domain Name Center (DNS), a repository of domain names and their IP addresses. If you’ve already visited Amazon from the same browser, it will pull the address from the cache. Then, a browser sends the request to the found IP address using the HTTPS protocol.

**Second, the web server processes the request**. The web server where Amazon.com is located catches the request and sends it to the storage area to locate the page and all data that follows with it. But its route is held via Business Logic (also called Domain Logic and Application Logic). BL manages how each piece of data is accessed and determines this workflow specifically for each application . As BL processes the request, it sends it to storage to locate the looked-for data.

**Third, you receive your data**. Your response travels back to you and you see the content of the web page on your display. The graphical interface you see when scrolling Amazon’s or any other website is called the front end of an application – it depicts all UX and UI components so that a user can access the information they came looking for.

## **Web application architecture components and Three-Tier Architecture**

Most web applications are developed by separating its main functions into layers, or tiers. This allows you to easily replace and upgrade each layer independently. This architectural pattern is called Multi- or Three-Tier Architecture.

### **Presentation layer**

The presentation layer is accessible to users via a browser and consists of user interface components and UI process components that support interaction with the system. It’s developed using three core technologies: HTML, CSS, and JavaScript. While HTML is the code that determines what your website will contain, CSS controls how it will look. JavaScript and its frameworks make your website interactive – responsive to a user’s actions. Developers use JavaScript frameworks such as [Angular](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/) and [React](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/) to make the content on the page dynamic.

### **Business layer**

This layer, also called Business Logic or Domain Logic or Application Layer, accepts user requests from the browser, processes them, and determines the routes through which the data will be accessed. The workflows by which the data and requests travel through the back end are encoded in a business layer. For example, if your application is a hotel booking website, business logic will be responsible for the sequence of events a traveler will go through when booking a room.

Although business rules can be a manifestation of the business logic, they are not the same. Sometimes business rules are extracted and managed separately, using a Business Rules Management System, as we discussed in our article on [back office systems](https://www.altexsoft.com/blog/travel/managing-an-online-travel-agency-how-back-office-system-of-an-ota-works/).

### **Persistence layer**

Also called the storage or data access layer, the persistance layer is a centralized location that receives all data calls and provides access to the persistent storage of an application. The persistence layer is closely connected to the business layer, so the logic knows which database to talk to and the data retrieving process is more optimized.

The data storage infrastructure includes a server and a [Database Management System](https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/), software to communicate with the database itself, applications, and user interfaces to obtain data and parse it. Typically you can store your data either in owned hardware servers or in the cloud – meaning, that you purchase data center management and maintenance services while accessing your storage virtually. Using the services of [cloud technology providers](https://www.altexsoft.com/blog/cloud/comparing-serverless-architecture-providers-aws-azure-google-ibm-and-other-faas-vendors/) such as Amazon, Google, or Microsoft, you can utilize Infrastructure-as-a-Service, Platform-as-a-Service, or serverless approaches to cloud management.

There are also components that usually exist in all web applications but are separated from the main layers:

**Cross-cutting code.**This component handles other application concerns such as communications, operational management, and security. It affects all parts of the system but should never mix with them.

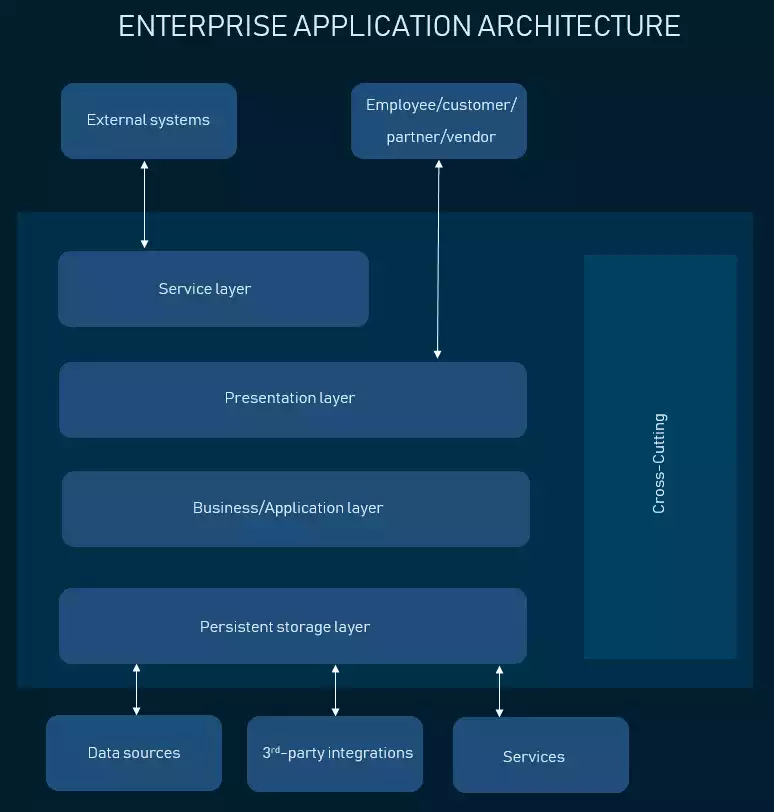
**Third-party integrations**. Payment gateways, social logins, GDSs in travel websites are all integrations connected to the application’s back end via pieces of code called APIs. They allow your software to source data from other software and widen your functionality without coding it from scratch. Read [how APIs work](https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/) in our dedicated article.

## **Enterprise applications**

Enterprise application is a highly customizable software that’s developed specifically for the needs of a particular organization. It usually has several business-oriented tools integrated under a single interface. Enterprise systems are also directly wired into the company’s existing workflow. They are robust, give access to a lot of users simultaneously, and share interfaces with various other enterprise tools. If you want to know more about the process of [developing an enterprise architecture](https://www.altexsoft.com/blog/business/how-enterprise-architects-close-the-gap-between-technology-and-business/), check out article.

The two main distinctions enterprise application architecture has from a regular web application is the addition of another layer to the classic pattern – the service layer.

The **service layer** is another abstraction between Presentation and Business Logic. It’s an integration gateway that allows other software to access your business logic and resources without interacting with those resources directly. It works by passing messages through a separate interface and works like an API.



Apart from an extra layer, enterprise applications have access to data sources from other applications in an organization, making it a network of software solutions, connected by APIs. Besides, there are more groups of users who have access to different functional components – they can be business partners, clients, admins, and several groups of staff. Sometimes the presentation tiers are separate for all of them, so you can deploy the application as intranet or extranet.