**What is Backend development?**

Backend development refers to the server-side aspect of web development, focusing on creating and managing the server logic, databases, and APIs. It involves handling user authentication, authorization, and processing user requests, typically using [backend development languages](https://roadmap.sh/backend/languages) such as Python, Java, Ruby, PHP, JavaScript (Node.js), and .NET.

**What does a Backend Developer do?**

A backend developer focuses on creating and maintaining the server-side components of web applications. They are primarily tasked with developing server-side APIs, handling database operations, and ensuring that the backend can manage high traffic volumes efficiently. Key responsibilities include integrating external services such as payment gateways and cloud services, and enhancing the performance and scalability of systems. This role is crucial for processing and securing data, serving as the backbone that supports frontend developers in delivering a seamless user experience.

**Internet**

**How does the internet work?**

The internet is a global network of interconnected computers that communicate using standardized protocols, primarily TCP/IP. When you request a webpage, your device sends a data packet through your internet service provider (ISP) to a DNS server, which translates the website's domain name into an IP address. The packet is then routed across various networks (using routers and switches) to the destination server, which processes the request and sends back the response. This back-and-forth exchange enables the transfer of data like web pages, emails, and files, making the internet a dynamic, decentralized system for global communication.

**What is HTTP?**

HTTP (Hypertext Transfer Protocol) is a protocol used for transmitting hypertext via the World Wide Web. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands. HTTP operates on a request-response model: a client (usually a web browser) sends an HTTP request to a server for resources, such as web pages or files, and the server responds with the requested content and an HTTP status code indicating the result of the request. HTTP is stateless, meaning each request from a client to a server is independent and does not retain information about previous interactions. It forms the foundation of data communication on the web and is typically used with secure HTTP (HTTPS) for encrypted communication.

**What is Domain Name?**

A domain name is a human-readable address used to identify a specific location on the internet, making it easier to access websites and online services. It translates to an IP address, which is a numerical identifier used by computers to locate and connect to servers. A domain name consists of two main parts: the **second-level domain** (e.g., "example" in "example.com") and the **top-level domain** (e.g., ".com"). Domain names are managed by domain name registrars and are essential for establishing a web presence, providing a user-friendly way to navigate to websites instead of using numeric IP addresses.

**What’s Hosting?**

Hosting refers to the service of providing server space and resources for storing and delivering website files and applications to users over the internet. Hosting providers offer the infrastructure, such as servers, storage, and network connectivity, required to make websites and applications accessible online. There are various types of hosting, including shared hosting (where multiple websites share a single server), virtual private servers (VPS), dedicated hosting (where a single server is dedicated to one user), and cloud hosting (which uses a network of servers to provide scalable resources). Hosting services often include domain registration, security features, and technical support to ensure websites are reliably available and perform well.

**DNS and how it works**

DNS (Domain Name System) is a hierarchical, decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It translates human-readable domain names (like www.example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other. DNS servers distributed worldwide work together to resolve these queries, forming a global directory service. The system uses a tree-like structure with root servers at the top, followed by top-level domain servers (.com, .org, etc.), authoritative name servers for specific domains, and local DNS servers. DNS is crucial for the functioning of the Internet, enabling users to access websites and services using memorable names instead of numerical IP addresses. It also supports email routing, service discovery, and other network protocols.

**Browsers and how they work**

Web browsers are software applications that enable users to access, retrieve, and navigate information on the World Wide Web. They interpret and display HTML, CSS, and JavaScript to render web pages. Modern browsers like Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge offer features such as tabbed browsing, bookmarks, extensions, and synchronization across devices. They incorporate rendering engines (e.g., Blink, Gecko, WebKit) to process web content, and JavaScript engines for executing code. Browsers also manage security through features like sandboxing, HTTPS enforcement, and pop-up blocking. They support various web standards and technologies, including HTML5, CSS3, and Web APIs, enabling rich, interactive web experiences. With the increasing complexity of web applications, browsers have evolved to become powerful platforms, balancing performance, security, and user experience in the ever-changing landscape of the internet.

**Backend languages:**

**JavaScript**

JavaScript is a versatile, high-level programming language primarily used for adding interactivity and dynamic features to websites. It runs in the browser, allowing for client-side scripting that can manipulate HTML and CSS, respond to user events, and interact with web APIs. JavaScript is also used on the server side with environments like Node.js, enabling full-stack development. It supports event-driven, functional, and imperative programming styles, and has a rich ecosystem of libraries and frameworks (like React, Angular, and Vue) that enhance its capabilities and streamline development.

**Java**

Java is a high-level, object-oriented programming language known for its portability, robustness, and scalability. Developed by Sun Microsystems (now Oracle), Java follows the "write once, run anywhere" principle, allowing code to run on any device with a Java Virtual Machine (JVM). It's widely used for building large-scale enterprise applications, Android mobile apps, and web services. Java features automatic memory management (garbage collection), a vast standard library, and strong security features, making it a popular choice for backend systems, distributed applications, and cloud-based solutions.

**Python**

Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. Its design emphasizes code readability and a clear, straightforward syntax, making it accessible for both beginners and experienced developers. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It has a rich ecosystem of libraries and frameworks, such as Django and Flask for web development, Pandas and NumPy for data analysis, and TensorFlow and PyTorch for machine learning. Python is widely used in web development, data science, automation, and scripting, and it benefits from a strong community and extensive documentation.

**C#**

C# (pronounced C-sharp) is a modern, object-oriented programming language developed by Microsoft as part of its .NET framework. It combines the power and efficiency of C++ with the simplicity of Visual Basic, featuring strong typing, lexical scoping, and support for functional, generic, and component-oriented programming paradigms. C# is widely used for developing Windows desktop applications, web applications with ASP.NET, games with Unity, and cross-platform mobile apps using Xamarin. It offers features like garbage collection, type safety, and extensive library support. C# continues to evolve, with regular updates introducing new capabilities such as asynchronous programming, nullable reference types, and pattern matching. Its integration with the .NET ecosystem and Microsoft's development tools makes it a popular choice for enterprise software development and large-scale applications.

**Version Control Systems**

Version Control Systems (VCS) are tools that manage and track changes to code or documents over time, allowing multiple users to collaborate on a project efficiently. They record every change made to files, enabling developers to revert to previous versions, compare changes, and maintain a history of modifications. VCS can be centralized, where the repository is hosted on a central server (e.g., Subversion), or distributed, where each user has a complete copy of the repository (e.g., Git, Mercurial). Version control facilitates collaboration, enhances code integrity, and supports continuous integration by enabling smooth management of concurrent changes and resolving conflicts.

**Git**

Git is a distributed version control system designed to handle projects of any size with speed and efficiency. Created by Linus Torvalds in 2005, it tracks changes in source code during software development, allowing multiple developers to work together on non-linear development. Git maintains a complete history of all changes, enabling easy rollbacks and comparisons between versions. Its distributed nature means each developer has a full copy of the repository, allowing for offline work and backup. Git's key features include branching and merging capabilities, staging area for commits, and support for collaborative workflows like pull requests. Its speed, flexibility, and robust branching and merging capabilities have made it the most widely used version control system in software development, particularly for open-source projects and team collaborations.

**Repo Hosting Services**

Repo hosting services are platforms that provide storage, management, and collaboration tools for version-controlled code repositories. These services support version control systems like Git, Mercurial, or Subversion, allowing developers to manage and track changes to their codebases, collaborate with others, and automate workflows. Key features often include branching and merging, pull requests, issue tracking, code review, and integration with continuous integration/continuous deployment (CI/CD) pipelines. Popular repo hosting services include GitHub, GitLab, and Bitbucket, each offering various levels of free and paid features tailored to different team sizes and project requirements.

# **GitHub**

GitHub is a web-based platform for version control and collaboration using Git. Owned by Microsoft, it provides hosting for software development and offers features beyond basic Git functionality. GitHub includes tools for project management, code review, and social coding. Key features include repositories for storing code, pull requests for proposing and reviewing changes, issues for tracking bugs and tasks, and actions for automating workflows. It supports both public and private repositories, making it popular for open-source projects and private development. GitHub's collaborative features, like forking repositories and inline code comments, facilitate team development and community contributions. With its extensive integrations and large user base, GitHub has become a central hub for developers, serving as a portfolio, collaboration platform, and deployment tool for software projects of all sizes.

**What is a Database**

A database is a collection of useful data of one or more related organizations structured in a way to make data an asset to the organization. A database management system is a software designed to assist in maintaining and extracting large collections of data in a timely fashion.

**Relational** **Databases**

Relational databases are a type of database management system (DBMS) that organizes data into structured tables with rows and columns, using a schema to define data relationships and constraints. They employ Structured Query Language (SQL) for querying and managing data, supporting operations such as data retrieval, insertion, updating, and deletion. Relational databases enforce data integrity through keys (primary and foreign) and constraints (such as unique and not-null), and they are designed to handle complex queries, transactions, and data relationships efficiently. Examples of relational databases include MySQL, PostgreSQL, and Oracle Database. They are commonly used for applications requiring structured data storage, strong consistency, and complex querying capabilities.

**MySQL**

MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It uses SQL (Structured Query Language) for database interactions and supports a range of features for data management, including transactions, indexing, and stored procedures. MySQL is widely used for web applications, data warehousing, and various other applications due to its scalability and flexibility. It integrates well with many programming languages and platforms, and is often employed in conjunction with web servers and frameworks in popular software stacks like LAMP (Linux, Apache, MySQL, PHP/Python/Perl). MySQL is maintained by Oracle Corporation and has a large community and ecosystem supporting its development and use.

**MS SQL**

Microsoft SQL Server (MS SQL) is a relational database management system developed by Microsoft for managing and storing structured data. It supports a wide range of data operations, including querying, transaction management, and data warehousing. SQL Server provides tools and features for database design, performance optimization, and security, including support for complex queries through T-SQL (Transact-SQL), data integration with SQL Server Integration Services (SSIS), and business intelligence with SQL Server Analysis Services (SSAS) and SQL Server Reporting Services (SSRS). It is commonly used in enterprise environments for applications requiring reliable data storage, transaction processing, and reporting.

**Oracle**

Oracle Database is a highly robust, enterprise-grade relational database management system (RDBMS) developed by Oracle Corporation. Known for its scalability, reliability, and comprehensive features, Oracle Database supports complex data management tasks and mission-critical applications. It provides advanced functionalities like SQL querying, transaction management, high availability through clustering, and data warehousing. Oracle's database solutions include support for various data models, such as relational, spatial, and graph, and offer tools for security, performance optimization, and data integration. It is widely used in industries requiring large-scale, secure, and high-performance data processing.

**SQLite**

SQLite is a lightweight, serverless, self-contained SQL database engine that is designed for simplicity and efficiency. It is widely used in embedded systems and applications where a full-featured database server is not required, such as mobile apps, desktop applications, and small to medium-sized websites. SQLite stores data in a single file, which makes it easy to deploy and manage. It supports standard SQL queries and provides ACID (Atomicity, Consistency, Isolation, Durability) compliance to ensure data integrity. SQLite’s small footprint, minimal configuration, and ease of use make it a popular choice for applications needing a compact, high-performance database solution.

**APIs**

An API (Application Programming Interface) is a set of defined rules and protocols that allow different software applications to communicate and interact with each other. It provides a standardized way for developers to access and manipulate the functionalities or data of a service, application, or platform without needing to understand its internal workings. APIs can be public or private and are commonly used to integrate disparate systems, facilitate third-party development, and enable interoperability between applications. They typically include endpoints, request methods (like GET, POST, PUT), and data formats (like JSON or XML) to interact with.