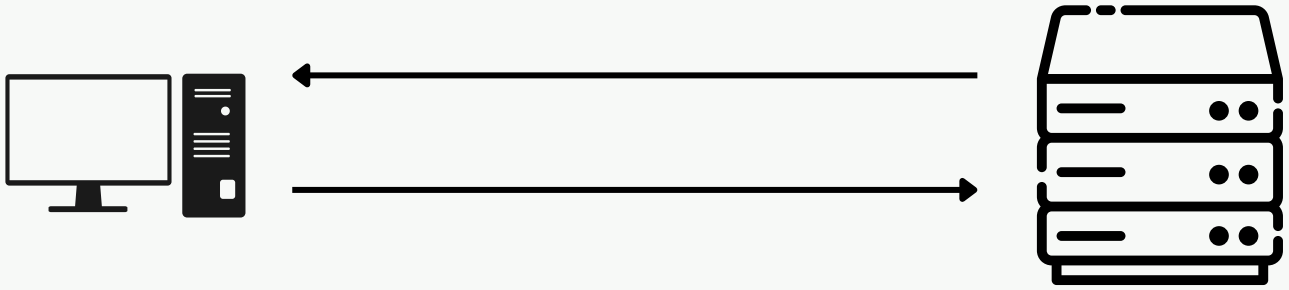
by Mohsin Ansari

2023

Introduction to

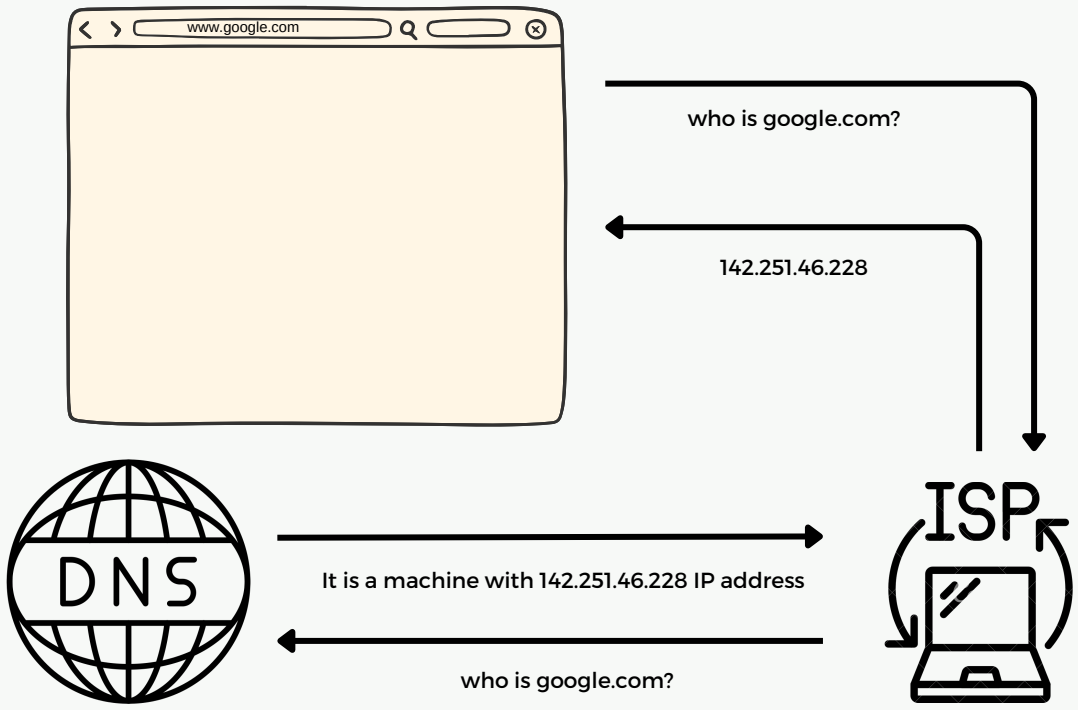
Backend Development

How Web works?



The web, short for the World Wide Web, is a system of interconnected web pages and resources that are accessible over the internet. It allows users to access and share information through various technologies and protocols.

At the core of the web is the client-server model. The web is divided into two main components: clients and servers. Clients are devices, such as computers, smartphones, or tablets, that users use to access the web. Servers are powerful computers that store and serve web content and respond to client requests.



When you enter a domain name into your web browser's address bar, the browser first needs to find the corresponding IP address of the server hosting that domain. It sends a DNS query to the DNS resolver.

Your ISP provides a DNS resolver, which is a specialized server responsible for handling DNS queries. The resolver looks up the requested domain name in its local cache; if the information is found, it returns the corresponding IP address directly.

If the DNS resolver doesn't have the IP address in its cache, it performs a recursive DNS query. It contacts other DNS servers on the internet to resolve the domain name step by step until it finds the correct IP address.

Once the DNS resolver obtains the IP address, it returns it to your browser, allowing the browser to initiate the HTTP request to the server hosting the website.

By working together, web browsers, ISPs, and DNS form the essential infrastructure that enables users to access websites and resources on the internet seamlessly.

URL (Uniform Resource Locator):

A URL is the address used to locate a specific resource on the internet. It's like a set of directions that helps your web browser find the right webpage, image, video, or any other online content you want to access. URLs are used to identify and access resources hosted on web servers.
for example : "https://www.example.com/page.html"

- "https://" is the protocol (HTTP Secure) used for secure communication.
- "www.example.com" is the domain name, which tells the browser where to find the website.
- "/page.html" is the path, indicating the specific resource (a webpage named "page.html") you want to access on that website.

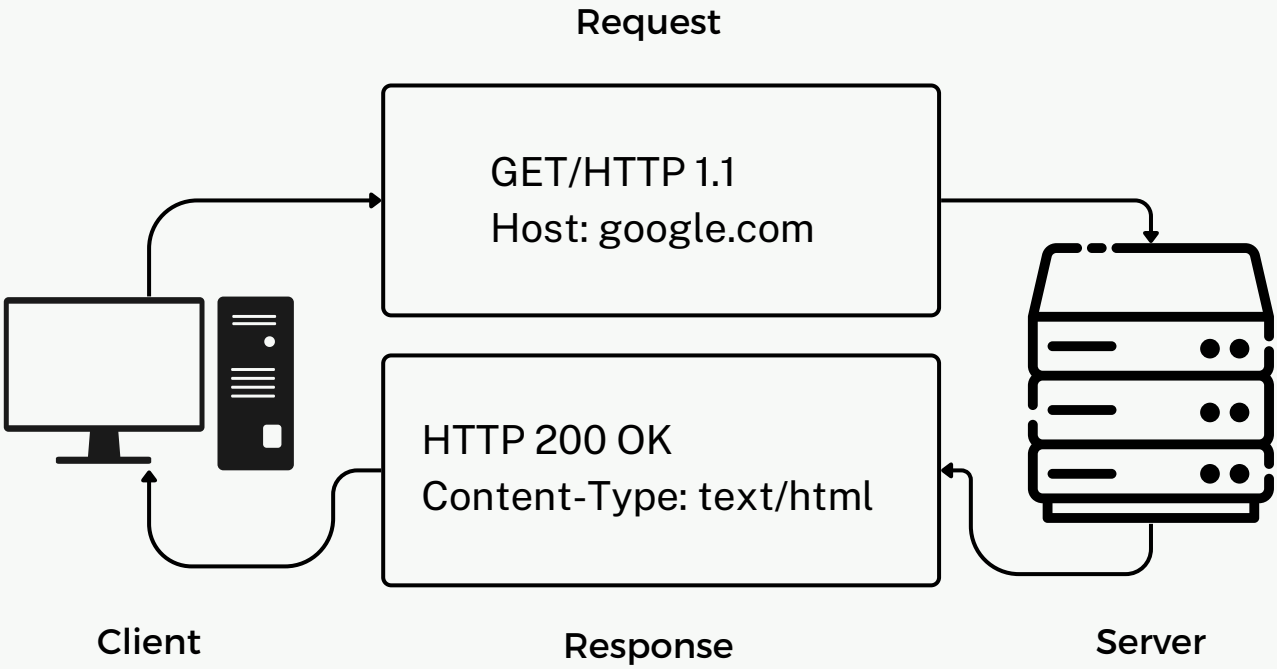
URL Query Parameters:

URL query parameters are a specific type of URL parameters used to customize and filter the content of a web page or to request specific information from a server. They are commonly used in search queries and filtering options on websites.
Query parameters are also added to the end of the URL after a question mark (?), and multiple query parameters are separated with ampersands (&).

example:
"https://www.example.com/search?query=shoes&size=10&color=blue"

- "query=shoes" is a query parameter indicating that the search query is for the term "shoes."
- "size=10" is another query parameter indicating the size of the shoes the user is looking for is 10.
- "color=blue" is a third query parameter indicating the desired color is blue.

HTTP Protocol



HTTP, which stands for Hypertext Transfer Protocol, is the foundation of communication on the World Wide Web. It's the language that web browsers and web servers use to talk to each other. Think of it as a set of rules that allows your computer to request and receive web pages, images, videos, and other online content from the internet.

Client Requests: When you type a web address (URL) into your web browser or click on a link, your browser wants to see a specific web page or resource. So, it sends a request to the web server that hosts that webpage or resource.

Server Responds: The web server receives the request and processes it. It then sends back a response to your browser. This response includes the web page or resource you wanted to access, along with additional information like the status of the request and some instructions for the browser.

HTTP methods: also known as HTTP methods are a set of actions or verbs that define the type of operation a client wants to perform on a resource hosted on a web server. These methods are an essential part of the HTTP protocol and determine how the server should handle the incoming request. Each HTTP method has a specific purpose and is used for different types of interactions between clients and servers.
Here are some commonly used HTTP methods:

GET: The GET method is used to request data from the server. When a client sends a GET request, it asks the server to retrieve and return a specific resource, such as a web page or an image. The data is sent in the URL as part of the request, making it visible in the browser's address bar. GET requests are considered "safe" and "idempotent," meaning they should not cause any side effects on the server and can be repeated without changing the server's state.

POST: The POST method is used to submit data to be processed by the server. When a client sends a POST request, it typically includes data in the request body (not in the URL) that the server needs to create or update a resource on the server. For example, when you fill out a form on a website and click the "Submit" button, the data you entered is sent to the server using a POST request.

PUT: The PUT method is used to update or replace an existing resource on the server. It sends the entire updated representation of the resource to the server. PUT requests are idempotent, meaning making the same PUT request multiple times should have the same result as making it once.

DELETE: The DELETE method is used to remove a resource from the server. When a client sends a DELETE request, it instructs the server to delete the specified resource. Like PUT, DELETE requests are also idempotent.

Learn more about HTTP methods here.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

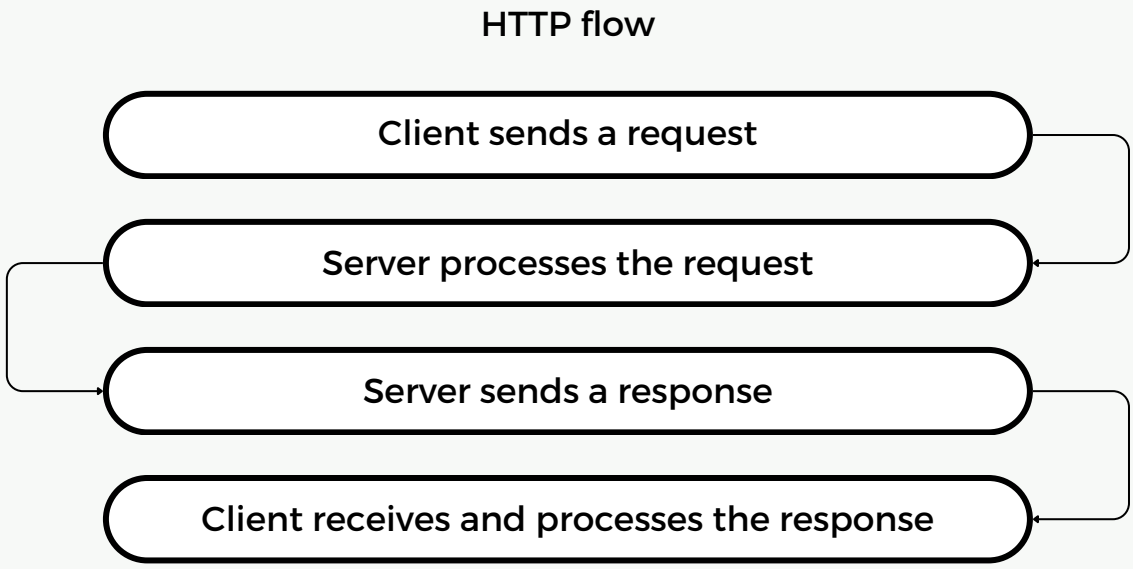
Status Codes: The server's response includes a three-digit number called a "status code." These codes tell your browser whether the request was successful, encountered an error, or needs further action. For example, "200 OK" means the request was successful, while "404 Not Found" means the server couldn't find the requested resource.

Responses are grouped in five classes:

1. Informational responses (100 – 199)
2. Successful responses (200 – 299)
3. Redirection messages (300 – 399)
4. Client error responses (400 – 499)
5. Server error responses (500 – 599)

Learn more about http status codes here.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Request and Response Headers: Both the client (your browser) and the server include some extra information in their requests and responses, called "headers." Headers provide essential details about the request and response, such as the type of content being sent, the date and time of the request, and caching instructions



REST API

REST API (Representational State Transfer Application Programming Interface): REST API is a specific type of API that follows the principles of Representational State Transfer (REST). REST is an architectural style that defines a set of constraints for designing networked applications.

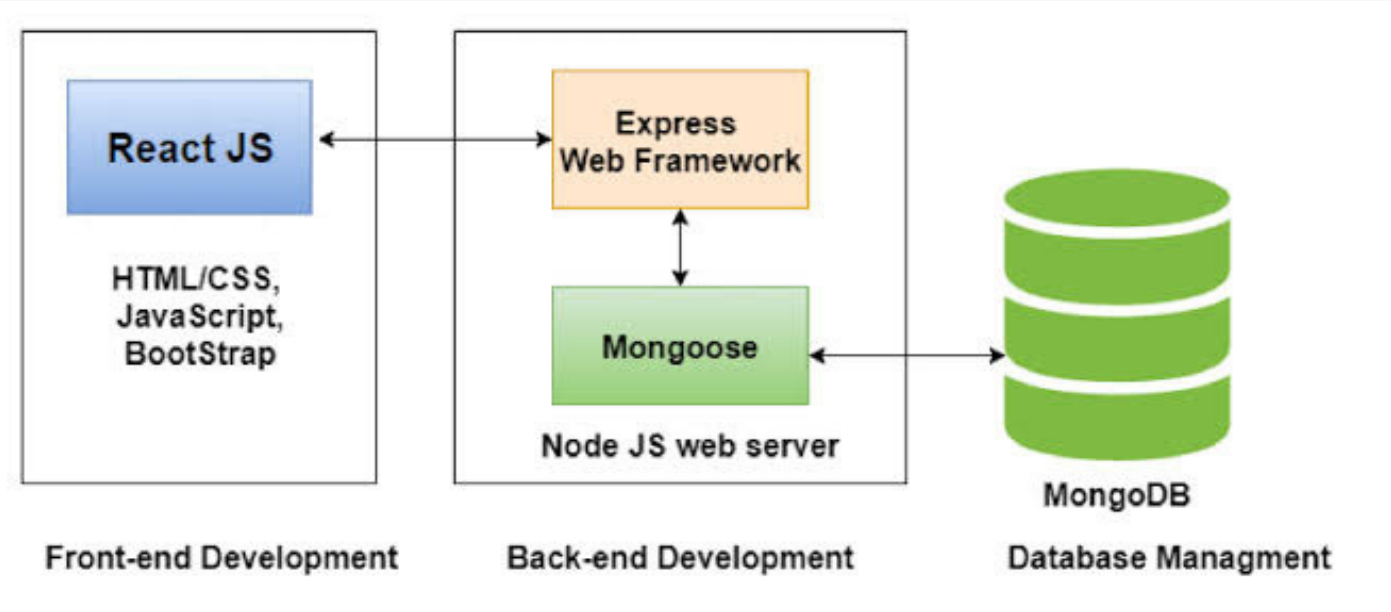
Stateless: Each request from the client to the server must contain all the information needed to understand and process the request. The server does not store any information about the client's state between requests.

Resource-Based: REST APIs are centered around resources, which are identified by URLs. Each resource can be accessed or manipulated using the appropriate HTTP methods.

CRUD Operations: REST APIs often map the four basic CRUD (Create, Read, Update, Delete) operations to the corresponding HTTP methods (POST, GET, PUT, DELETE).

Resource	Method	Description
`/teams`	GET	Gets a list of all teams
`/teams/{id}`	GET	Gets details for a single team
`/teams/{id}/members`	GET	Gets members of a team
`/teams`	POST	Creates a new team
`/teams/{id}/members`	POST	Adds a member to a team
`/teams/{id}`	PUT	Updates team properties
`/teams/{id}/members/{memberId}`	PUT	Updates member properties
`/teams/{id}/members/{memberId}`	DELETE	Removes a member from the team
`/teams/{id}`	DELETE	Deletes an entire team

MERN Stack



The MERN stack is a popular web development stack that includes four key technologies: MongoDB, Express.js, React.js, and Node.js. It is used to build full-stack web applications, with each technology serving a specific purpose in the development process.

React.js: React.js is a JavaScript library for building user interfaces. It enables developers to create dynamic and interactive UI components, making it easier to develop complex web applications. React.js follows a "virtual DOM" approach to optimize rendering and improve performance.

Axios: Axios is a popular JavaScript library used for making HTTP requests from the browser. It provides a simple and efficient way to interact with APIs and fetch data from servers. Axios supports Promises, making it easy to handle asynchronous operations.

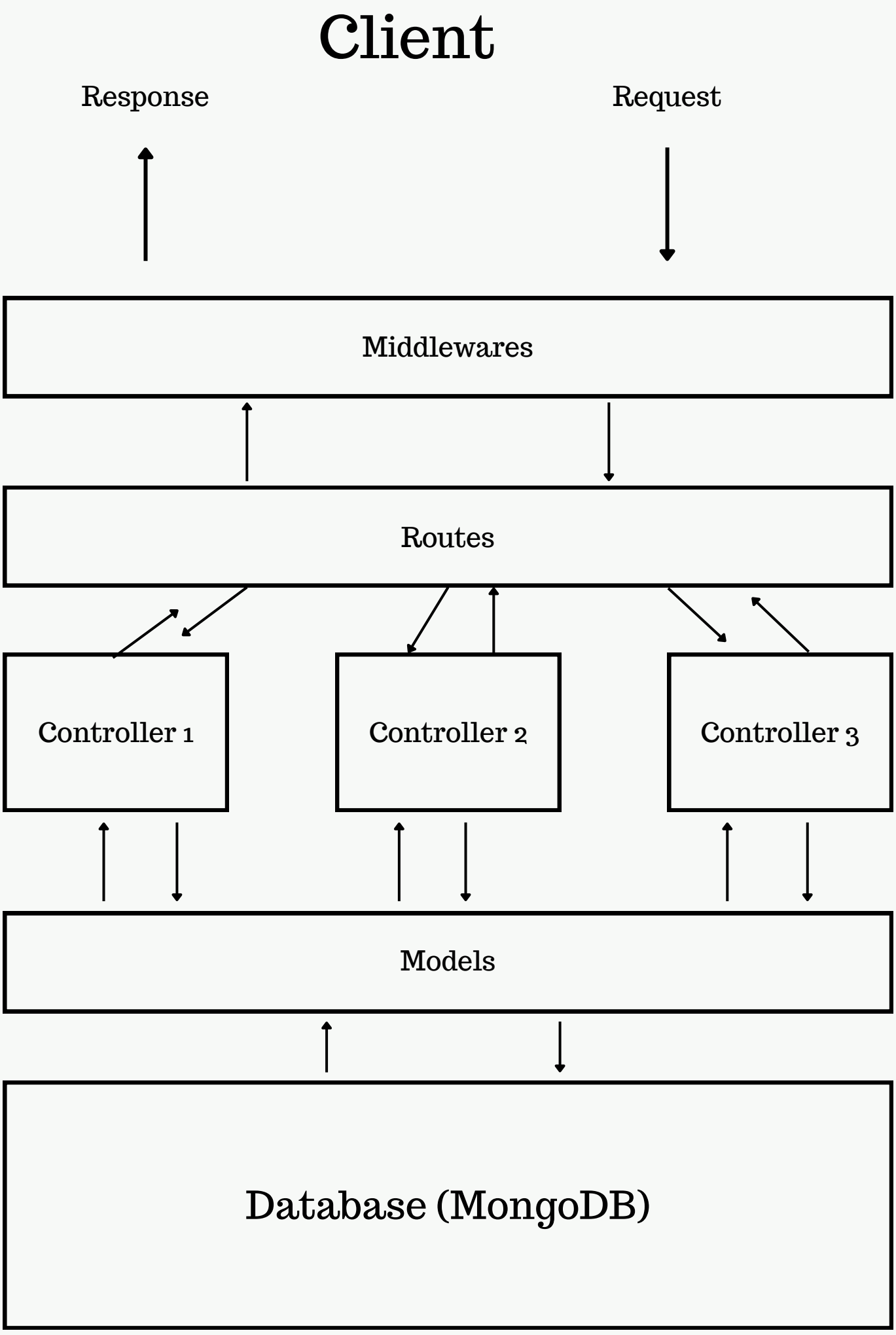
Node.js: Node.js is a server-side JavaScript runtime environment that enables developers to build scalable and efficient server applications. It uses an event-driven, non-blocking I/O model, which makes it ideal for handling a large number of concurrent connections.

Express.js: Express.js is a web application framework for Node.js. It simplifies the process of building server-side applications and APIs. Express.js provides a robust set of features for handling HTTP requests, routing, and middleware integration.

MongoDB: MongoDB is a NoSQL database that stores data in a flexible, document-oriented format. It allows developers to work with data in a more natural and dynamic way compared to traditional SQL databases. MongoDB is suitable for handling large volumes of unstructured or semi-structured data.

Mongoose: Mongoose is an Object-Document Mapping (ODM) library for MongoDB. It provides a higher-level abstraction for working with MongoDB and simplifies tasks like data validation, schema definition, and querying.

In conclusion, the MERN stack is a powerful and popular choice for developing modern web applications. It combines the flexibility of MongoDB, the server-side capabilities of Express.js and Node.js, and the interactive user interfaces created with React.js to provide a full-stack development environment that is efficient, scalable, and well-suited for building a wide range of web applications.


Client
Response
Request
Middlewares
Routes
Controller 1
Controller 2
Controller 3
Models
Database (MongoDB)