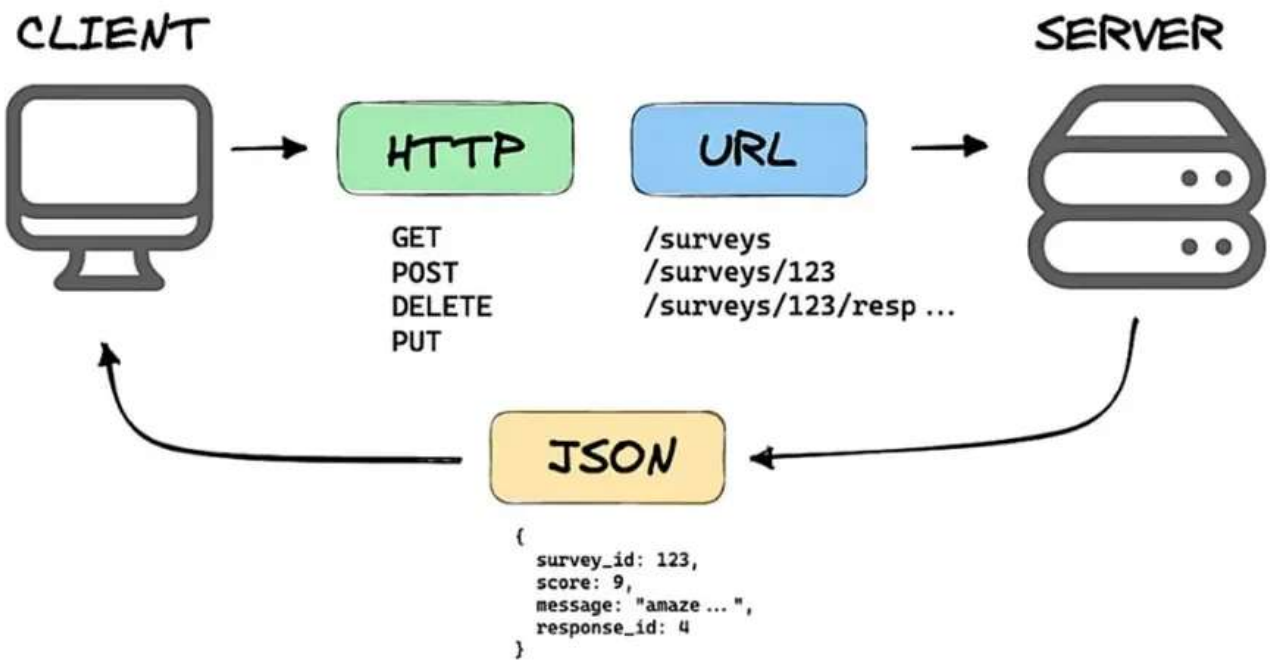
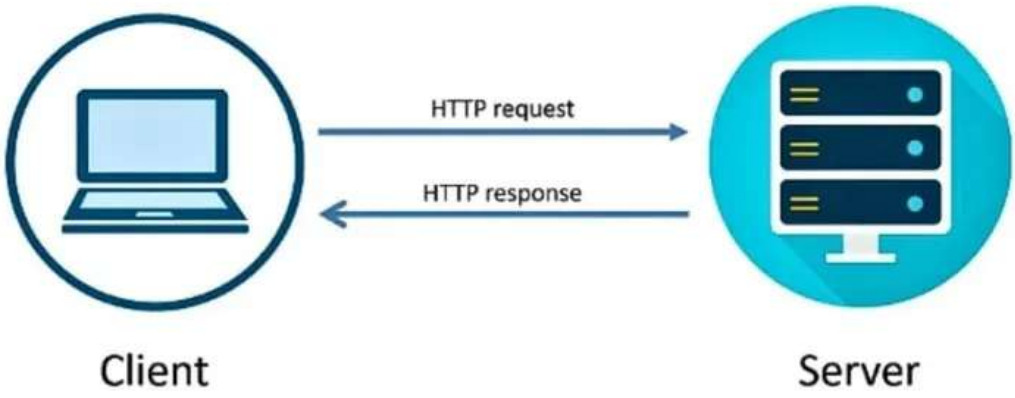
REST API Explained



Swipe>>>

A REST API (Representational State Transfer Application Programming Interface) is a **set of rules that allows applications to communicate with each other over HTTP.** It is based on the principles of REST, an architectural style defined by Roy Fielding in his doctoral dissertation.



(Coding Notes Bundle in bio)

Key Principles of REST

Stateless: Each request from a client to a server must contain all the information needed to understand and process the request. The server does not store any client context.

Client-Server Architecture: The client and server operate independently, promoting separation of concerns.

Cacheable: Responses must define themselves as cacheable or not, to improve performance.

Uniform Interface: REST APIs use standard HTTP methods (GET, POST, PUT, DELETE, etc.) and follow standard conventions for interacting with resources.

Layered System: The architecture can have layers, improving scalability and encapsulating logic.

Code on Demand (Optional): Servers can extend the functionality of clients by sending code (e.g., JavaScript) for execution.

REST API Operations

HTTP Method	CRUD Operation	Description
GET	Read	Retrieve data from the server.
POST	Create	Send data to the server to create a resource.
PUT	Update	Update an existing resource.
DELETE	Delete	Remove a resource.
PATCH	Update (Partial)	Partially update a resource.

REST API Components

Resources: The entities or data your API interacts with. Resources are represented as URLs.

- **Example:** /users, /products, /orders

Endpoints: The specific URLs where the API can be accessed.

- **Example:** https://api.example.com/v1/users

HTTP Status Codes:

- **200 OK:** Request succeeded.
- **201 Created:** Resource created successfully.
- **400 Bad Request:** Invalid input or request.
- **401 Unauthorized:** Authentication required.
- **404 Not Found:** Resource not found.
- **500 Internal Server Error:** Server-side error.