. Overview

09 May 2025 23:36

→ What is REST API ?

→ Why / Benefits .

→ Building Blocks
- URL
- Methods
- Headers
- Request
- Response
- Status code

→ Build App

→ Postman

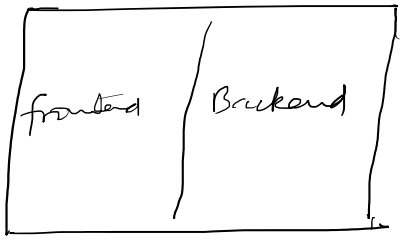
→ HTTP 1/2/3

→ Best practices

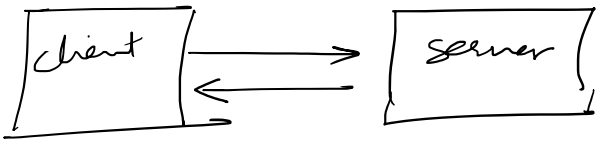
→ Advance

2. Architecture

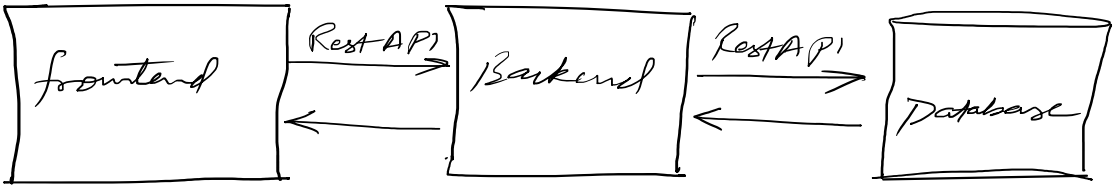
10 May 2025 01:07



1-tier Arc.



2-tier Arc.



3-tier Arc.

Restfull apis

10 May 2025 01:25

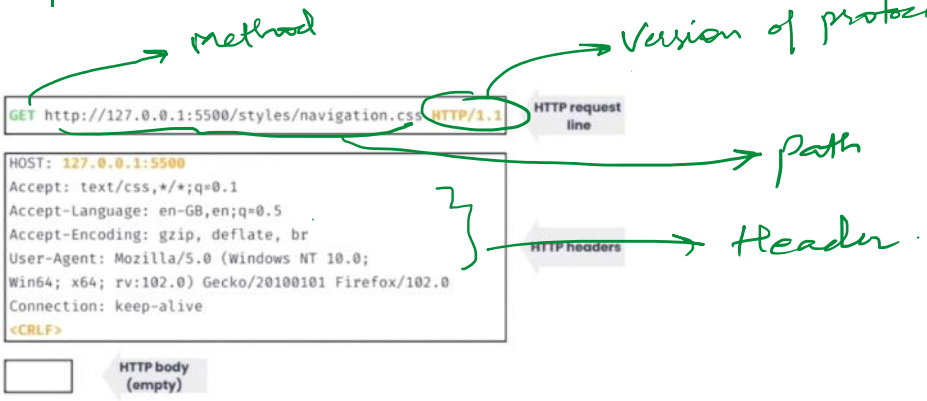
API → Application programming Interface.
It is a set of rules and protocols that allows one piece of software/system to communicate with another.

REST → Representational state Transfer
API that follows the rules of REST is RESTful API.

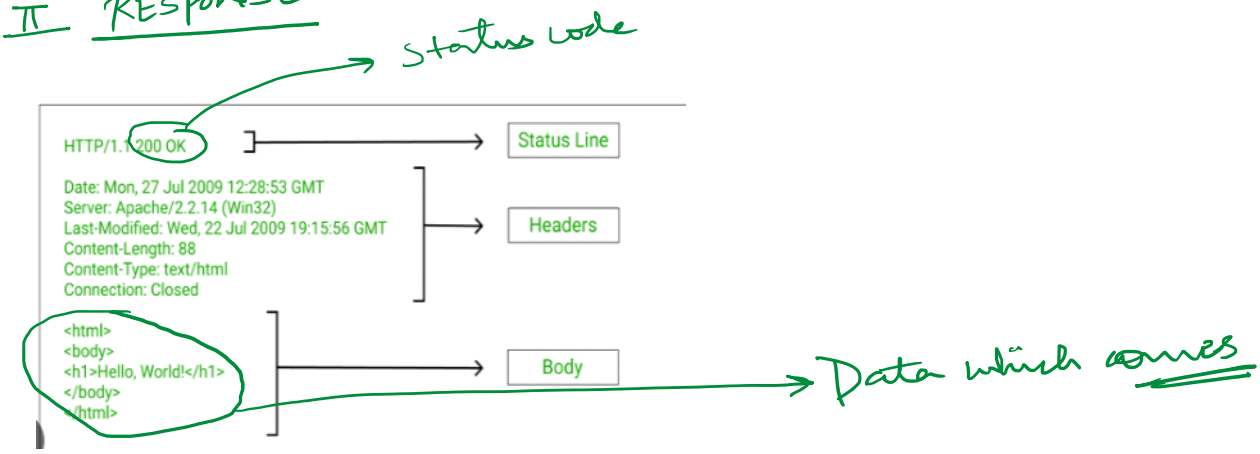
Built on top of HTTP

Benefits of Rest
→ Ease of use.
→ Stateless (Maintain no state)
→ Scalability
→ Flexibility with Data
→ Ease of Testing
→ Security
→ uniform Interface
→ caching.
→ Seperation of concern
frontend → JS & nodejs
Backend → any
→ Interoperability → language agnostic

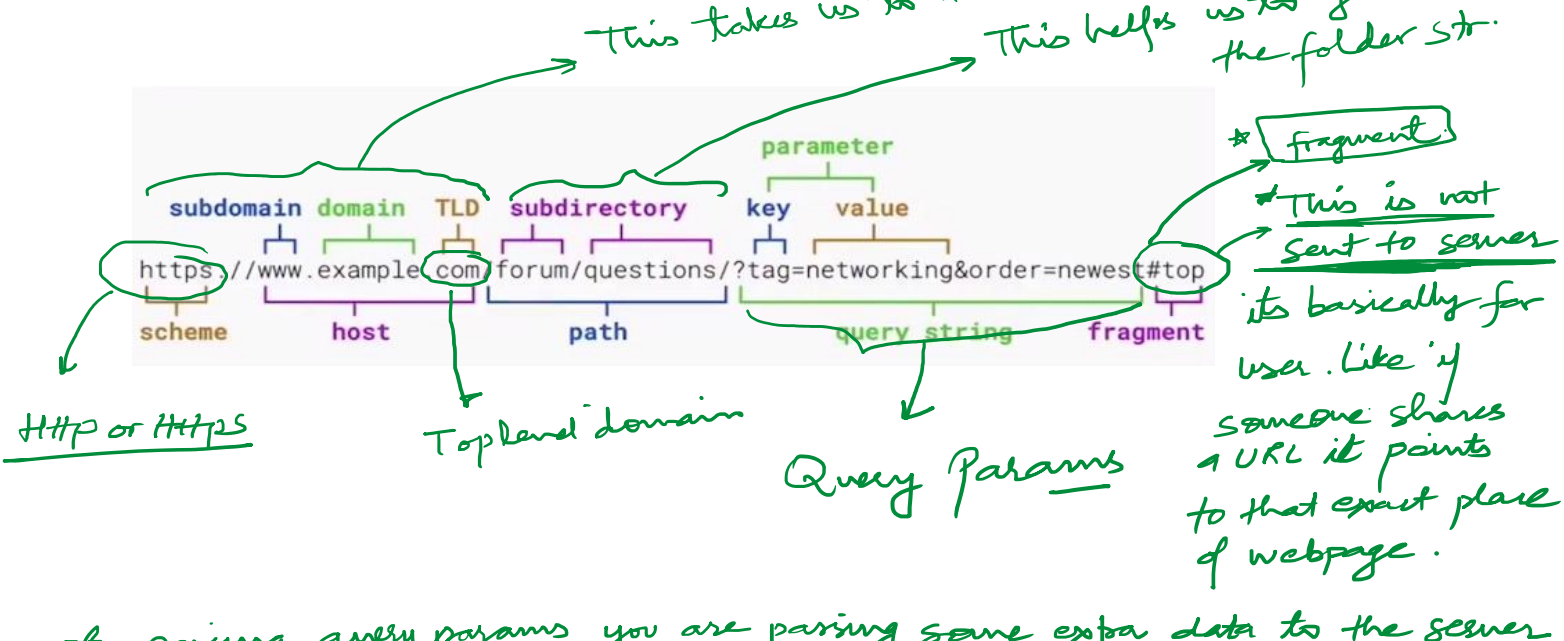
I. Request



II RESPONSE



III URL



→ By passing query params you are passing some extra data to the server. So when the programs execute it will use that data and gives accordingly.

So when the programs execute it will use that data and gives Response accordingly.

Methods

10 May 2025 15:41

Let say our Todo Application has given functionality

→ Create Todo → Create
→ Get Todo → Read
→ Update Todo → Update
→ Delete Todo → Delete

} CRUD operations

We can utilize these by using Methods

C POST
R GET
U PUT/PATCH
D. DELETE

Additionally we also have

→ HEAD → checking if a resource exist w/o fetching the full content.
→ OPTIONS → check supported Methods.
→ CONNECT
→ TRACE → Tracing the path of a request for debugging.

———— X ————

By Browser we can only send GET Request, for patch, delete, Post we can use fetch and all, but that is a tedious Job. So for this we use a Tool called POSTMAN.

What is bodyparser?

→ It is a middleware
→ Reads the raw body of incoming Request
→ parses it
→ Add it as a JS-object to req.body.

for ex:- I send { 'name' : 'abhilash' } in req body

↓

It will get parsed by body-parser.

It will get parsed by body-parser.

So that inside app.J can use req.body.name directly.

	GET	PUT	POST	DELETE
URL	/todos /todos/:id	/todos/:id	/todos	/todos/:id
Req body	✗	✓	✓	✗

Headers (REQUEST)

10 May 2025 21:29

	USECASE	EXAMPLES
HOST	Specifies the **domain name** of the server you're requesting.	Host: api.example.com
ORIGIN	Specifies the **origin (domain)** of the request.	Origin: https://myfrontend.com
REFERER	Indicates the **URL of the page** making the request.	Referer: https://example.com/page1
USER-AGENT	Identifies the **client software** (e.g., browser, mobile app).	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) User-Agent: PostmanRuntime/7.35.0
ACCEPT	Tells the server **what content type** the client can accept in the response.	Accept: application/json Accept: text/html
ACCEPT-LANGUAGE	Tells the server **which language(s)** the client prefers for the response.	Accept-Language: en-US,en;q=0.9,fr;q=0.8
ACCEPT-ENCODING	Tells the server **what compression algorithms** the client can handle to save bandwidth.	Accept-Encoding: gzip, deflate, br • gzip: Popular compression format • deflate: Another algorithm (slightly older) • br: Brotli compression (modern and efficient)
CONNECTION	controls **whether the network connection** stays open **after the current HTTP request/response** cycle.	Connection: keep-alive
AUTHORIZATION	Sends **credentials** or **tokens** for accessing protected resources.	Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6... Authorization: Basic YWRtaW46cGFzc3dvcmQ=
COOKIE	Sends previously stored **key-value pairs** (cookies) from the client (browser) to the server.	Cookie: sessionId=abc123; theme=dark
IF-MODIFIED-SINCE	Used for **caching optimization** — asks the server to send the file **only if it was modified** after a certain date/time.	If-Modified-Since: Sat, 10 May 2025 08:00:00 GMT
CACHE-CONTROL	Controls how **caching should be handled** by browsers, proxies, etc.	Cache-Control: no-cache Cache-Control: no-store Cache-Control: max-age=3600

Headers(Response)

11 May 2025 00:55

HEADERS	USECASE	EXAMPLES
DATE	Timestamp of when the response was generated	Date: Sat, 10 May 2025 10:30:00 GMT
SERVER	Identifies the server software	Server: Apache/2.4.1 (Unix)
CONTENT-TYPE	Tells client the **type of content**	Content-Type: application/json
CONTENT-LENGTH	Length of the response body (in bytes)	Content-Length: 3495
SET-COOKIE	Tells browser to **store a cookie**	Set-Cookie: sessionId=abc123; HttpOnly
CONTENT-ENCODING	Informs how the content is **compressed**	Content-Encoding: gzip
CACHE-CONTROL	Directs how response should be **cached**	Cache-Control: max-age=3600
LAST-MODIFIED	Timestamp when the resource was last changed	Last-Modified: Fri, 09 May 2025 18:00:00 GMT
ETAG	A unique version identifier (like a hash)	ETag: "a7f8c2e"
EXPIRES	Exact date/time when the cached content **should be considered stale**	Expires: Sun, 11 May 2025 10:30:00 GMT

STATUS CODES

11 May 2025 01:24

1XX → Information

100 → Continue → Go ahead & send the req body

101 → Switch Ex- (Http → Socket)

2XX → Success

200 → OK → standard success → GET Pending

201 → Created → Resource created → POST

202 → Accepted → Accepted (pending) → POST

204 → No content → Success (No body) → Delete

206 → partial content → partial Respone → Download

3XX → Redirection

301 → Permanently Moved

302 → Temporary Moved

307 → 302 = Method Retained

308 → 301 = Method Retained

4XX → Client Error

400 → BAD Request (enter pan)

401 → Unauthosized (not logged in)

403 → Auth Error (can't go to Admin)

405 → Method not allowed

404 → Content Not available

429 → Concurrent Req

5XX → Server Error

500 → Internal Server Error

502 → Bad Gateway

503 → Service is Unavailable

504 → Gateway timeout

507 → Insufficient Storage.