HTTP Versions :- Protocol versions that define how data is transferred.

1.1 — Sequential, slower — TCP —

2 — faster, multiplexed — TCP — parallel requests

3 — faster, secure and efficient — QUIC

TCP — Transmission control Protocol.

will transfer packets in a correct order, if one packet is delayed everything will have to wait, Security is optional TLS/SSL.

" Will send everything in order or die trying".

Banking - example

Guarented delivery, order

multiple handshakes lead to overhead which slows down.

UDP — User Datagram Protocol

Speed over performance

" Will throw packets into wind and hope they land"

No delivery guarentee, no order guarentee.

example - video calls, livestreams

QUIC — built on top of UDP

faster than TCP, only one single handshake at the begining

built in encryption

HTTP3

benefits cloud comp — low latency, end to end encryption.

→ HTTP Methods.
indicate the purpose of the request and what is expected if the req is successful.

GET – retrive data

POST – Submits entry to specified source

PUT – replaces the current data with request content

PATCH – partial modifications to a resource

DELETE – delete the data.

→ HTTP Status – codes returned from the server indicating request result

1xx – hold on

2xx – looks, here you go

3xx – you don't have permission to view this

4xx – doesn't work

5xx – Server issues

404 – not found
200 – ok
500 – server error.

→ HTTP Headers – metadata sent with request/response

Content-Type

Authorization.

→ Cookies — small data stored by the browser for maintaining stateful sessions

Stateful vs stateless — don't retain info from past interactions, treating each interaction as independant.

/

retain data from past interactions

traditional webapps

Banking systems

- improve performance by accessing previous data
- complex to develop
- data synchronization issues

example - RESTful API's

easier to scale and distribute

simple to develop and debug.

→ Caching — temporarily storing data to speed up repeated requests. avoids redundant work.

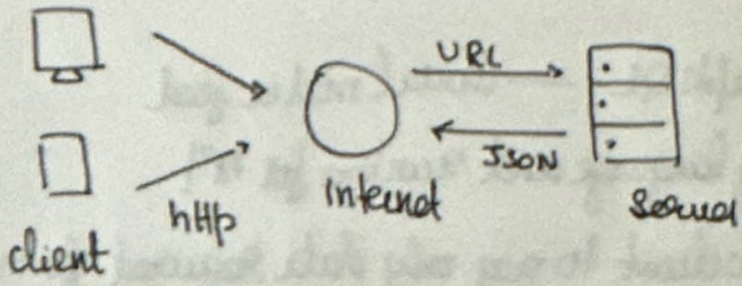
API Protocol —

Set of rules and standards that define how Software application communicate over a network

Factors
* Performance needs
* Security consideration
* Ease of implementation
* Scalability

① REST — Representational State Transfer — Ecommerce platform

* networked applications
* web based applications
* most common used
* HTTP
* Stateless
* resource based URLs

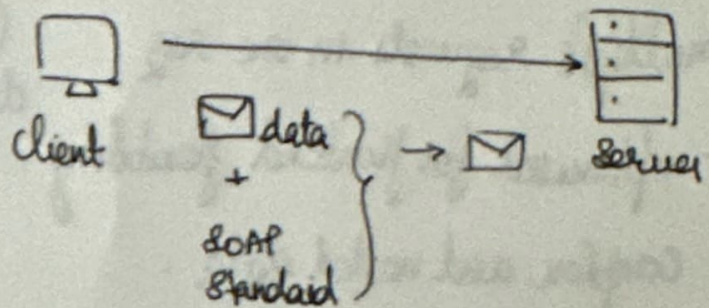


② SOAP — Simple Object Access Protocol

* XML based protocol for structured data
* define message structure using XML

Banking and Financial transactions.

Security, reliability, formal standards — Good for legacy enterprise system



③ gRPC

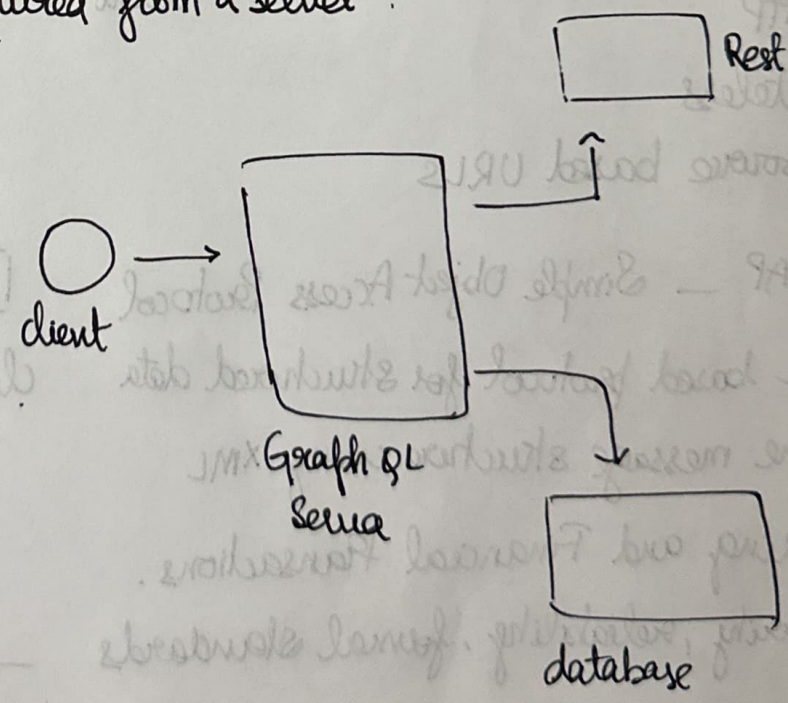
google

- high performance RPC framework using HTTP/2 protocol buffers
  Remote Procedure Call → allows one program on one device to ask another to do a task.
- backend to backend commn, microservice arch
- open source developed by google.

protocol buffer — open source cross platform data format used to serialize structured data.

④ Graph QL — Social media feed

- Query language and runtime for API
  allow client to req only data required from a server.
- facebook developed
- flexible way to interact with API
  multiple requests in one req
- optimised for frontend flexibility.
  complex and nested data.



⑤ WebSockets.

- two way communication b/w client and server
- realtime
- long lived connection.
- data sent and recived without overhead of repeated establishing new connections
- real time chat application
  chats, games, live dashboard.

API Authentication

minimizing vulnerabilities in API interactions.

→ Basic authorig - username / password in http headers
- quick prototyping
- Simple internal tools
- easy to implement

→ Token based authentication.

user logins with credential → server returns a token (random string) → client stores the token (localstorage / headers)

token is sent on each request

mobile apps, Single page applications.

Stateless on server.

token theft is a risk if not HTTPS

JWT JSON Web Tokens

Compact, used in API authen and authorization to securely transmit info b/w clients and services

JWT
- Header + Payload + Signature
  - Header — token type and hashing algorithm
  - Payload — claims
  - Signature — verify the tokens integrity.

Self contained, no need to hide it in DB

good for distributed system

OAuth 2.0 — Open Authorization

delegation protocol, allows a third party to acces you account without sharing pwd.

example — sign in with google - granting app access via google

third party integrations.

Cloud API's, Social logins

— complex to implement.

→ Hash-Based Message Authentication Code (HMAC)

cryptographic hash func + Secret key ↳ verify the integrity and authenticity of message

data not tampered.

→ CBOR Object Signing and Encryption COSE

Concise Binary Operation Representation.

keeps message size minimal.

various cryptographic algorithms — digital signatures

→ X.509 Certs

API to implement TLS for encryption communications.

Public key + identity of cert holder] — validated by a certificate authority.

→ (SAML) Security Assertion Markup Language.

open standard for exchanging authentication and authorization data b/w an identity provider (IdP) and a service provider

* Single Sign on (SSO)

→ System for Cross Domain Identity Management (SCIM)

multiple domains

OpenID Connect (OIDC)

→ built on top of OAuth 2.0

provide identity tokes that include user info, identity and authentication details

CORS – Cross Origin Resource Sharing

allows web apps to make request to a domain other than the one from which the resource originated

API uses CORS to specify which domains have permission to access resources prevents unauthorised domains from making cross origin req that can lead to data lekage.

API documentation.

contains info, instructions about how to effectively use and integrate with an API

reference manual for an API
Improved user adoption
increased awareness

understand your API
Know your audience
provide detailed instructions for the most common use cases
review, test and verify the documentation.
continously update your documant

API Endpoints
Methods
Parameters
Authentication
Headers
Request / Response Formats
Examples
Status Codes, Error Handling
Best Practices
API Lifecycle Information

Swagger UI
- Interactive API docs
- you define API in YAML or JSON _ Swagger UI reads this specs and creates an interactive document page
- helps non developers explore API
- Internal/external developers want to test API endpoints in a browser

Postman
- API testing, monitoring and documentation
- visual and user friendly
- autogenerates doc from collection
- powerful testing + collaboration features
- ex- frontend devs test endpoints before backend is ready

OpenAPI Specification (OAS)
- Standard format for describing REST API in YAML or JSON
- tools like Swagger, Redoc, Dapper Dox use it to generate doc and code
- Supported by many tools, code generation and automation

Redoc
- Static site generator for clean, customizable API docs from openAPI spec
- Single page HTML doc
- very polished UI, responsive design, easy to embed or host

Dapperdox
- document generator with support for markdown guides, diagrams
- enterprise with complex API's and multiple versions
- Supports guides and diagrams
- multiversion API's, easy to integrate into CI/CD.

API Features

→ Pagination — splitting results into multiple pages

→ URL, Query, Path Parameters
  ↳ passing input to API's

→ Idempotency — Safe repetable API calls.

→ API Versioning — managing changes using version numbers.

→ Content negotiation — client and server agree on data format.

→ HATEOS → Hypermedia as the Engine of Application state
hyperlinks in the API response
State of application changes as the client follows these links allowing for dynamic navigation and interaction

API Performance

Caching

Rate limiting — restricting the number of requests from a client
1000/hr - example

Load balancing — distributing API traffic across multiple servers for scalability.

Pagination — improve performance by returning limited data

Indexing — optimize database query to return results faster

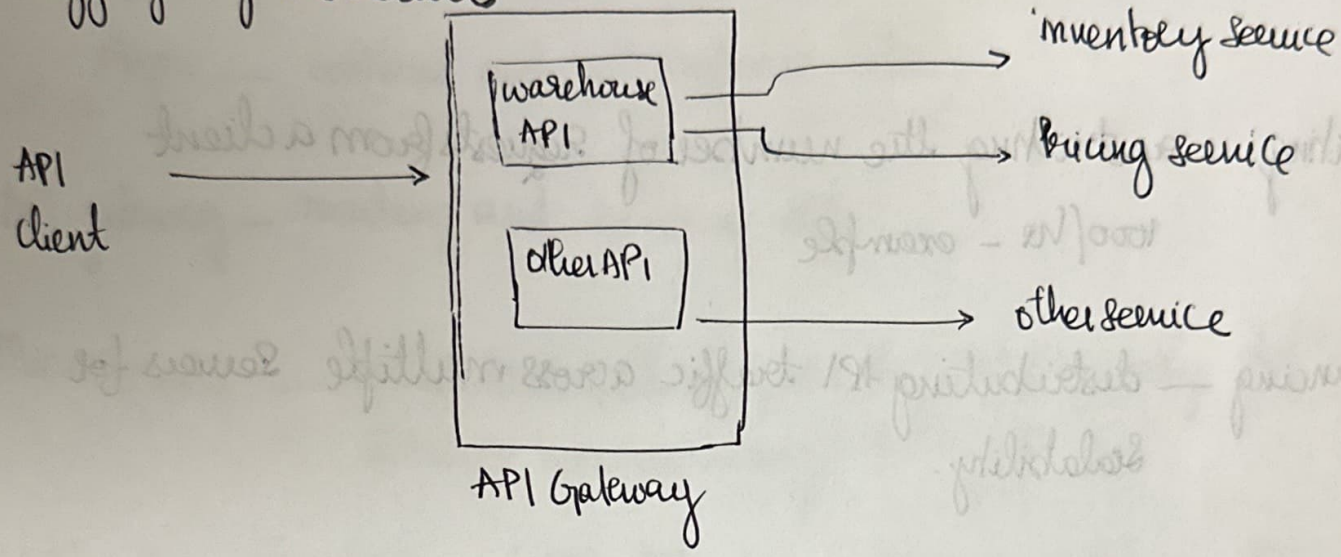
Scaling — horizontal and vertical

Performance testing — tools and techniques to measure API speed and scalability.

API Gateways

accepts API requests from a client, processes them based on defined policies, directs them to appropriate services and combines the response for a simplified user experience.

Typically API gateway handels requests by invoking multiple microservices and aggregating the results



capabilities — security policy

routing policy — routing, rate limiting, request/response manipulation, circuit breaker, blue-green and canary deployments, A/B testing load balancing, healthchecks custom error handling

Observability policy — real time metrics, historical metrics logging and tracing

API Gateway vs Gateway API

↳ Kubernetes Gateway API

opensource project managed by Kubernetes community to improve and standardize service networking in Kubernetes.

NGINX Kub Gateway

API Integration Patterns

1. Sync — wait for response

   Async — continue and get response later

2. API gateway — routing and securing API

3. Microservice — architecture pattern of small, independently deployable services communicating via API's

4. Webhooks — event driven HTTP callbacks — sends a real time HTTP Post req to a URL provided
   - ↳ Stripe
   - (Service) → example — payment succeeded.
   - * Payment confirmation (Stripe, Paypal)
   - * New github push event (CI/CD Triggers)
   - * SMS/Email delivery updates (Twilio, Sendgrid)
   - * Subscription or user status changes

5. Polling — client repeatedly asking server for updates

6. Batch Processing

7. Message Queue

Frameworks for Implementation

Provide a structured approach to build and manage API's offering tools, libraries and conventions that simplify development, testing and maintance

- language specificity
- features and functionality
- scalability and performance
- community support, developer experience
- testing and automation.

Springboot — extension of spring framework, spring based applications

Django REST — a full featured Python web framework is Django - restful API

Flask — lightweight and flexiable python microframework, simple to complex

FastAPI — high performing python framework for building APIs, leveraging type hints and offering built in Open API support

Lumen — PHP