17/12/22

Week 10_day 1

API Concepts

Krishna

System designing understanding helps you to avoid raising invalid bugs.

What before API?
How comm'n happended b4 API?

purpose is to ensure 2 sys to talk to each other.

b4 API, CORBA or RPC are used for this purpose

- common Object Req broker architecture
- Remote Procedure calls

Any utility serving some propose & enabling to communicate, is called API

Ex:- jar docm'th

String api

ex:- pdf document to word

What is API

↓

Appln programing interface.

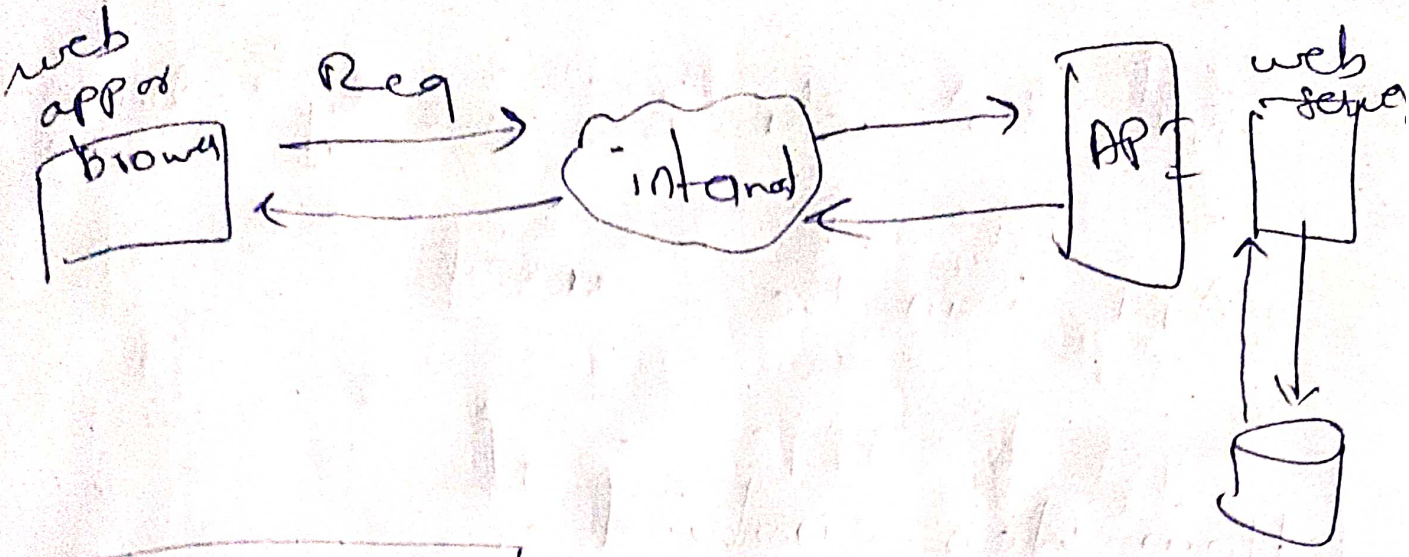
- Apis are little pieces of codes that makes it possible for digital devices, s/w appln, and data servers to talk with each other.

- Api is an intermediate.

How API works?

- acts like a middleman.

- There must be set of definations & protocols in place that allow s/w components to talk each other.



Why API ?

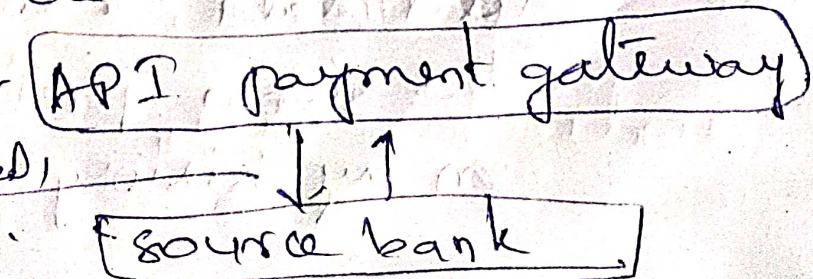
- technology agnostic
  i.e., platform independent.

- unmarshall (deserializing)
  marshall (serialize)

- At api level, as contracts are maintained, even if diff services designed on diff tech stack, marshalling & unmarshalling is maintained unique.

- Acts as abstraction layer of security

only when this entire connection established, payment is done.



- API also acts as quality gateway
- API can be reused

How Web evolved?

web 1.0 → static access for resources, read-only

2.0 → data gathering for users, advertisement, no data privacy.

3.0 → owning the resources, more privacy ex:- crypto, NFTs

ex:- wikipedia

ex:- apps become intelligent, & analysed user by collected user data, or whenever I browse to show preferences based on my history search.

→ we as a user, becomes more responsible & owning the content.
if someone else, wants to access content to share or edit will be difficult.

Who uses APIs?

- nowadays, everything is integrated with APIs.

ex:- used in, web-based sys.
- operating sys.
- DB
- computer h/w.

ex:- social media
- content streaming platforms
- cloud platform - AWS, Azure
- Selenium uses API to process the selenese cmds.

Selenium is lang. agnostic, it is designed based APIs.

Types of APIs.

- Request driven / sync APIs
- Event driven / Async APIs

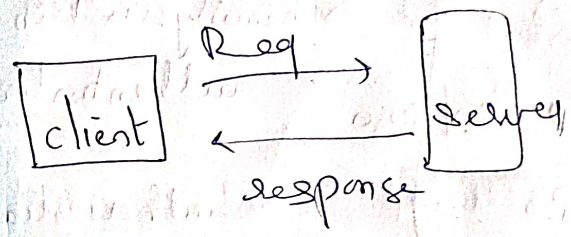
Request Driven

- widely used bcz most of apps used client-server model.

whenever, a client wants to access server,

client sends req by gathering all the required params in a form of serialized data.

then on recieving the serialized data, server deserializes the data & responses accordingly.



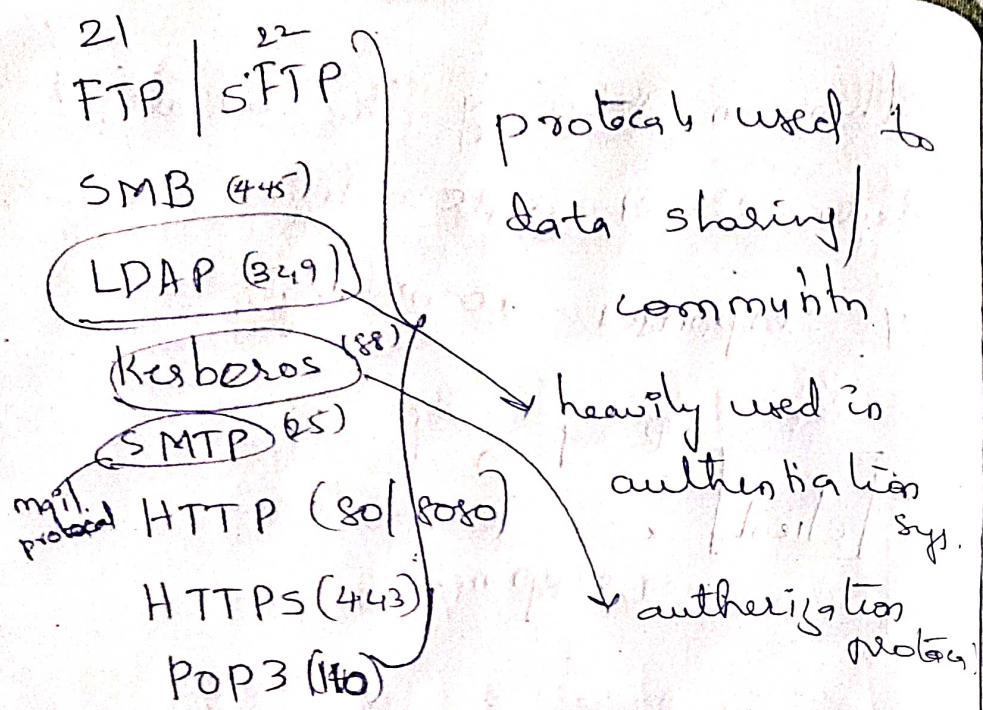
Ex:- googlePay has bcme flakely.

Protocals in N/wing

OST → open source interconnection

- physical layer
- data
- transport
- N/w
- session
- presentation
- Application

} proces involves in



Every protocol works with specific portNo:

bcz of diff ports, data collision doesn't happen

~~tlw~~ TelNet is something to tell you if servers are open or not.

ex:- ping to a server IP. to ensure if it is available.

TCP/IP → mostly used for data transfer.

UDP → alternate to TCP/IP bcz TCP/IP is connection oriented where as UDP is connection less

How client & server established a conn?

① handshake b/w client & server
② client sends req.
③ server responds
④ client acknowledges

TCP/IP

where as in UDP, anonymous calls

Why TCP/IP?

- bcz regenerates data packets instead of connection loss.

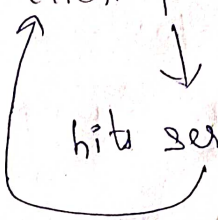
whereas UDP, losses packets when connection lost.

- in TCP/IP, connection can be kept alive for certained period.

- If time expires, connection also becomes stale.

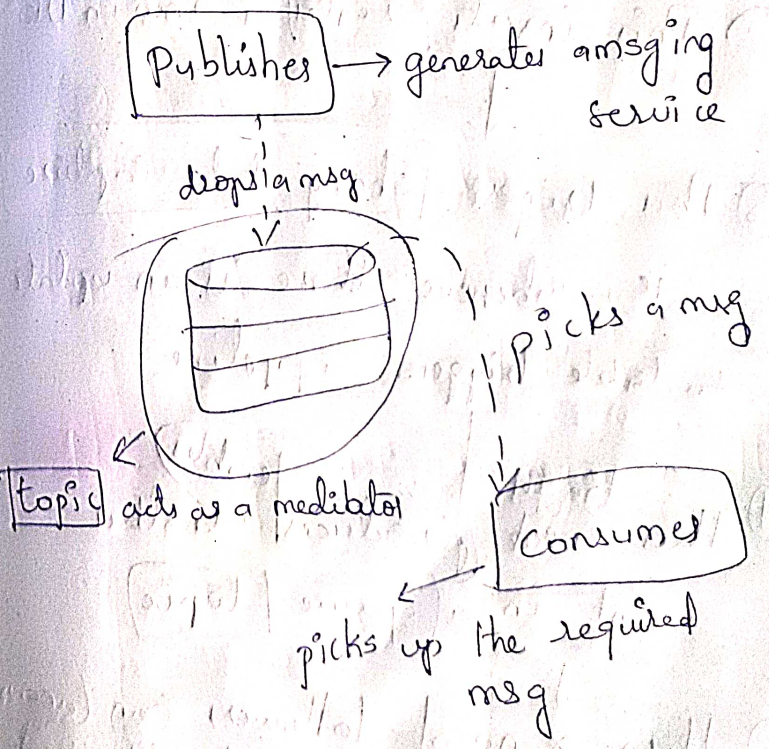
UDP is mostly used in login oriented designs.

Sync / Request Driven APIs.



hits back with response.

Async / Event driven APIs



Async API explanation with Tweet

① Posting a tweet

```
Http : POST
{
} // some data
endpoint / tweet
```

↓

② This (inserts / creates) a row in DB

↓

③ This triger (chang data capture)
ie, whenever there is an update to table triggers a pipeline

↓

④ this (trigger / ~~event~~) (produces / enables) another event in a (queue / topic)

↓

⑤ then x no. of followers can (access) the tweet from (topic)

What is Topic

- schema (JSON / xml) (Avro) std match with producer & consumer
- Retention period
  ie, info stays on the queue ~~for x hours x days~~
- Partition
  ie, if 6 parts available, it will be

Msg in queue / topic evicts / removed whenever consumer reads & sends acknowledgment
(or)
whever the retention period is met.

Throttling

limit the access to data.

~~keep ping etc~~

ie ① consider ping only once in any given minute

② consider the event when there is a state change (offline/online) ie, location change or action change.

When things can go wrong?

- pointing to wrong topics
- schema mismatches

  headers : correlation-id
- identity (handshake) incorrect certificates
- msg retries
  - ~~dead~~ technical errors needs to be retried

  compared to functional retries

quest

if I have dB connection vth singlet pattern,

will this restricts only 1 lkl dB connections?

(or)

will this also impacts my testcase lkl execution?

Type of request Driven APIs

- SOAP (simple obj Access Protocol)
- REST (Representational
- GraphQL
- GRPC

SOAP

- msging protocol vch exchanges structured info in the implementation of web services.
- works only vth XML data.
- supports only POST mthd
- works purely based on WSDL web service descriptive lang.
- payload is always heavies (req + wsdl)
- As it is stateful, cant be cached.
  server keeps remembering the state of a req.
- never have multiple end-points
- difficult to implement

REST

- architectural style to create webservices, using end-points
- built over HTTP protocol
- supports JSON, XML, HTML, Multipart
- Lesser payload.
- mthds supported
  - post
  - get
  - put
  - patch
  - delete
- Stateless & can be cached
  - ↓
  - server will not remember previous calls is coming to pool. but instead of saved data is cache.

REST Limitation

- N+1 service calls
- Data over fetching
  - or
  - underfetching
- schema is defined by server.

GraphQL

- recently developed using query lang. on top of webservices.
- It mainly tries to solve data fetching issue we see in REST.
- Unlike REST, GQL works with
  - type
  - fields.

- type refers to table
- field refers to column.

using this, schema can be defined at client layer itself unlike RSET, where schema is defined by server layer.

Tables (types)

Ex:-

Book table	Author Table
bookid	bookname
bookname	auth_name
auth_name	auth_id
publisher	

Fields.

Fields

Now, we can implement below endpoints:-

→ graphQL.

```
{ https://mybooks.com/books?query{
  books: {"bookname"},
  author: {"auth_name"} }
```

```
{ http://mybooks.com/books{bookid}
  " " " " ? book_info
          ="publish"
```

RST urls.