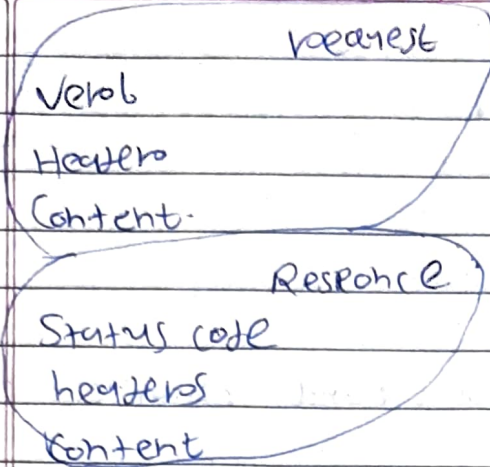


* API *



REST :- REpresentation, State transfer.

Rest API parts

req verb URI (every string)
 Headers
 request body

res status code
 headers
 response body.

- => URI Point to specific resources
- use unique identifiers
 - do not use primary keys (programmer can create trouble with this insted identify API programmer in readability)

- => every string.
- use non resource properties
- /sites? sort=name
 /sites? page=1
 /sites? format=json

⇒ Verbs

Get - Retrieve a resource	select
Post - Add a new resource	insert
Put - update an existing "	update
Patch - " resource with change	"
Delete - remove resource	delete

⇒ imp. idempotency if Put five 2's or 3's then that update ~~just~~ happen if no update then doesn't give error 404 but 200 OK
(this would not verify)

⇒ Design result

- multiple name
- Shouldn't expose server details.
 - java, node
 - Prefer camel casing.

⇒ Decide format during design

Accept : app / json, text / xml

between : application / json

Common formats :

JSON : application / json

XML : text / xml

JSONP : application / javascript*

RSS : application / xml + rss

Atom : application / xml + atom

⇒ Hypermedia

- Allow result to be self-describing
 - Allow programmatic navigation
 - Adds complexity.
- this can give some result to user
into program to use

=> Paging

HTTP://api/notes?Page=2

Page allow you to limit the output

=> API and Security

Do you need to secure your API?

Secure it?

1. using private and personalized data

yes

2. Sending sensitive data across the 'wire'?

yes

3. using credentials of any kind?

yes

4. trying to protect against abuse of your

yes

Server?