Agenda

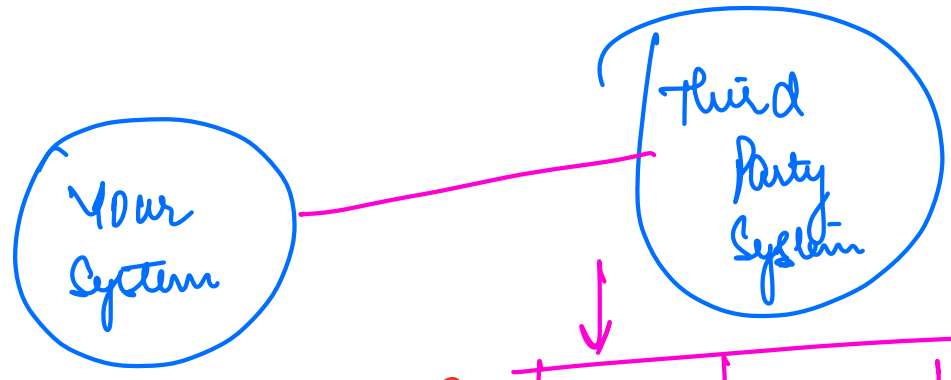
1. What is an API
2. What is REST
3. Good REST API Design Practices
4. LLD of Product Service
5. MVC Pattern
6. How Request is served in Spring Boot
7. Integrating Fake Store API

What is an API

API → Application Programming Interface.

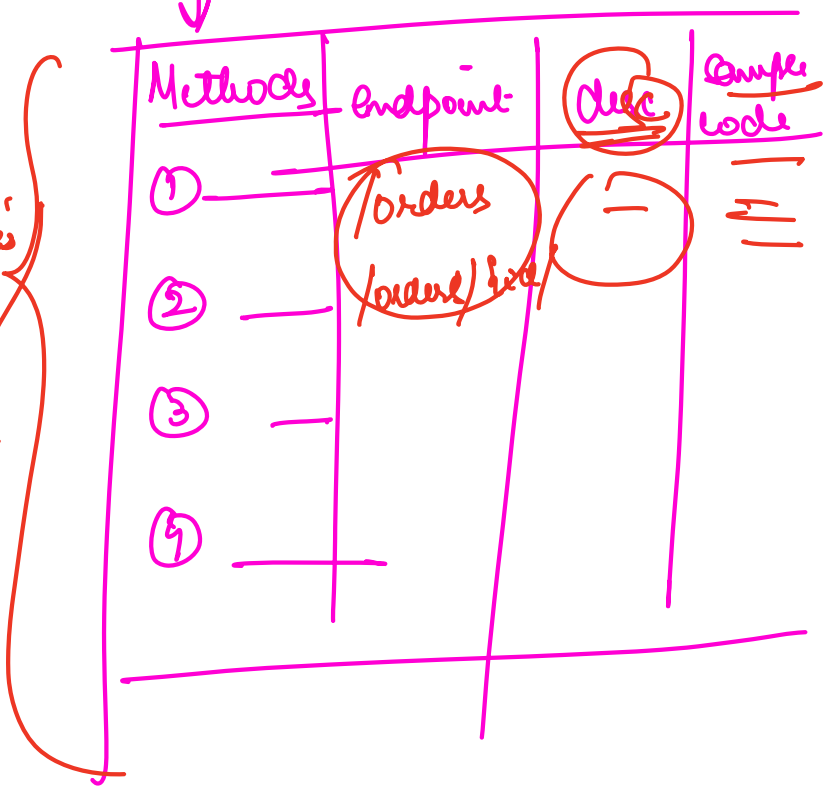
Contract

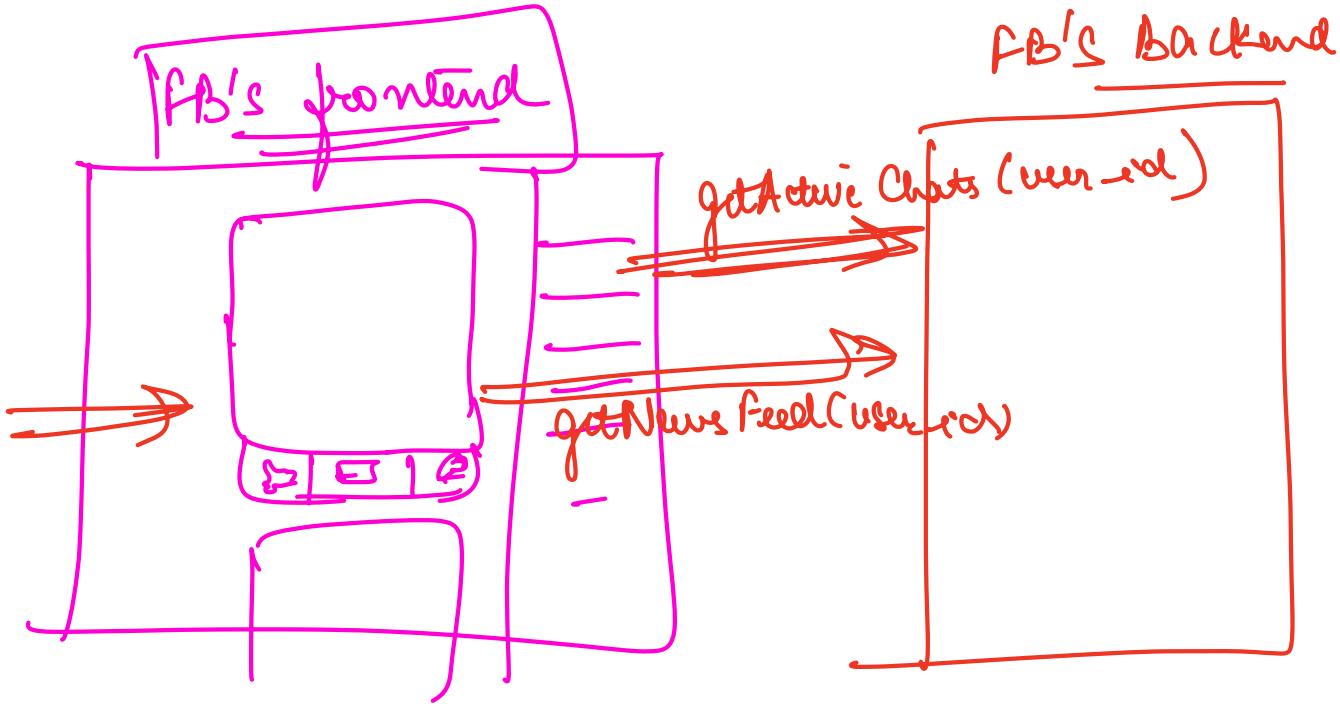
→ a contract that is followed by 2 appⁿ to be able to talk to each other.

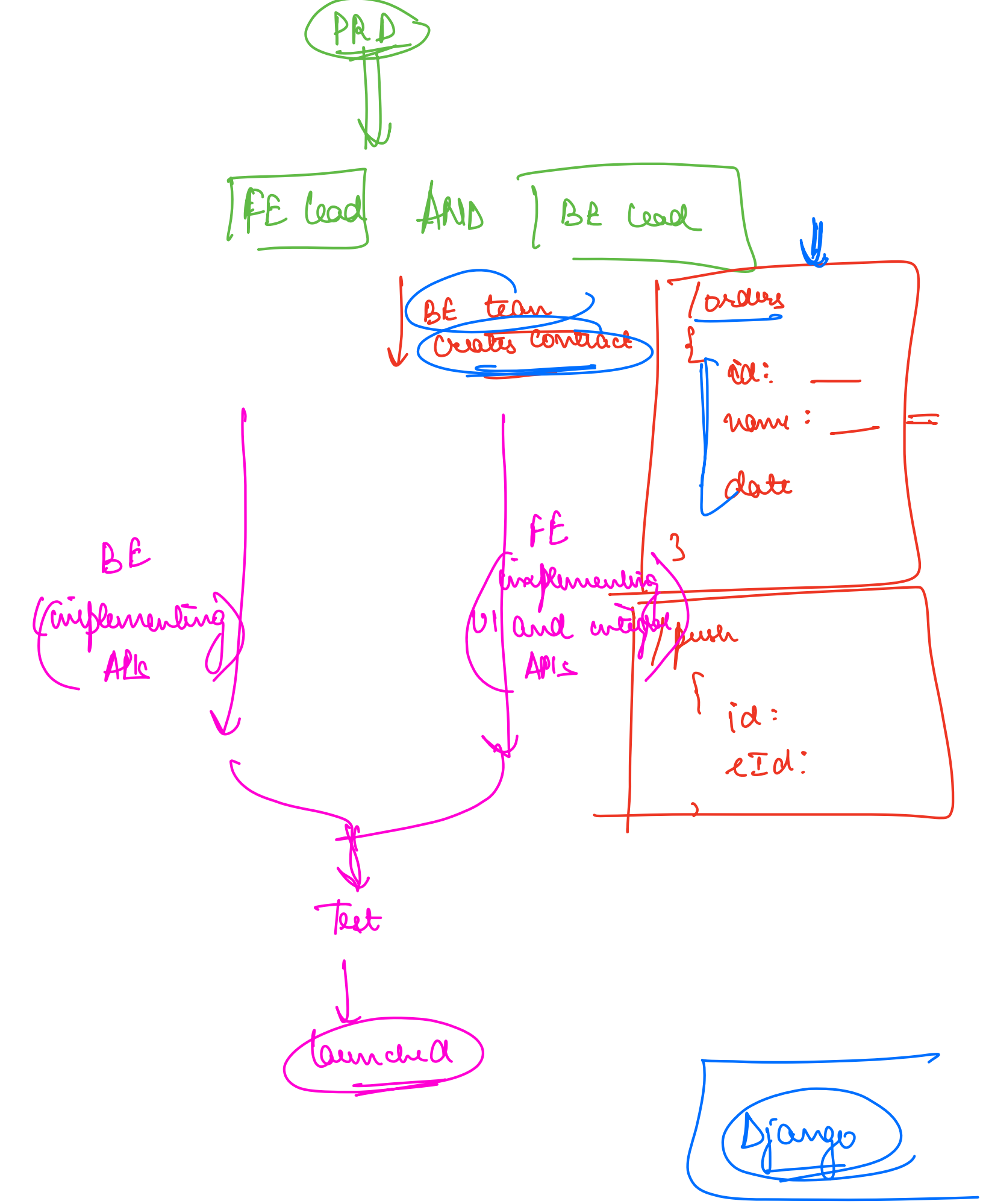


API: set of methods / endpoints / functionalities / features that are provided by a third party

: a contract b/w you and a provider.






PRD
FE lead
AND
BE lead
BE team
creates contract
/orders
{
id: ___
name: ___
data
BE
(implementing
APIs
FE
implementing
UI and integrating
APIs
3
push
{
id:
eId:
Test
Launched
Django

scaler.com/get-all-users

[{ }]

✗ Not a good way to name APIs

scaler.com/upload_video

REST APIs

→ Set of good practices on how your APIs should look like

→ how you should structure your APIs

RE → Representational
S → State
T → Transfer

⇒ creating / fetching / updating / entities

→ every request happens on a resource / set of resource.

① REST APIs should be designed around resources.

(endpoints should be name of the resource that is being affected)

NOUN

Scaler.com/users

Scaler.com/users/123

delete-user

get-user

updata-user

REST APIs also provide a way to specify what kind of action do you want to do on a resource.

HTTP Method ↓

Create → POST /users

GET /users

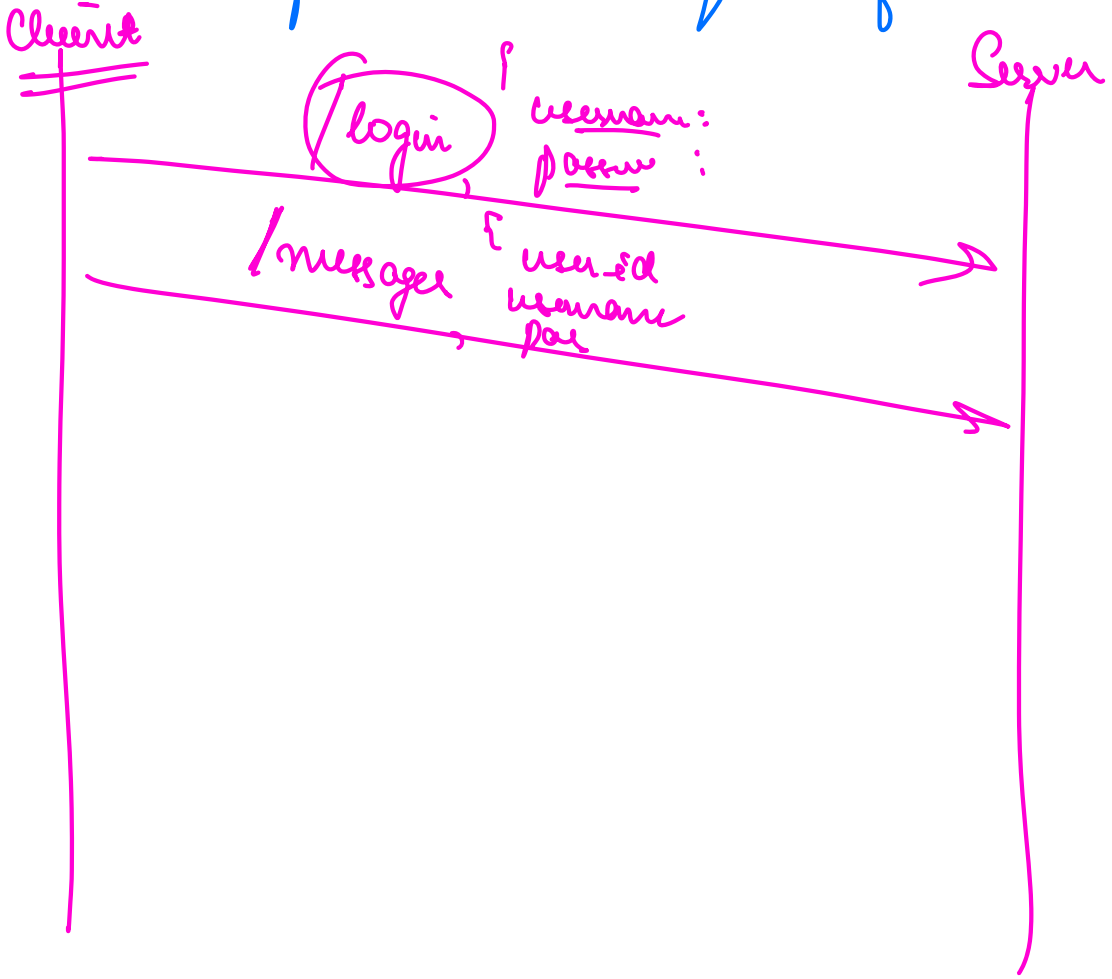
GET /users/1

PATCH /users/1

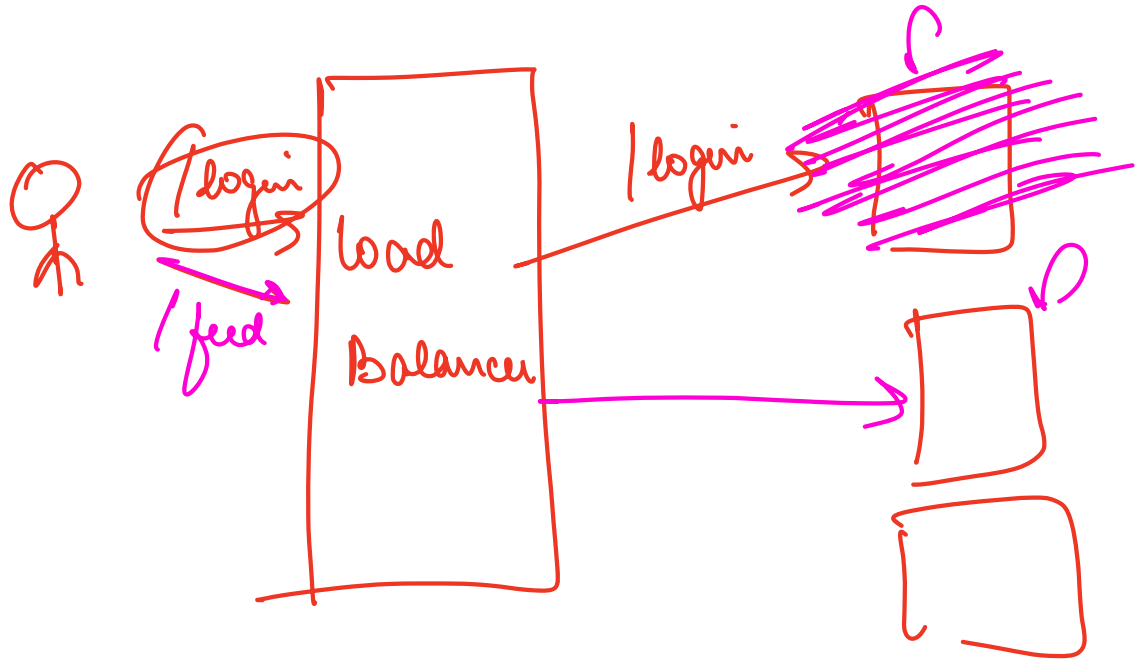
DELETE /users/1

(2) REST APIs should be stateless.

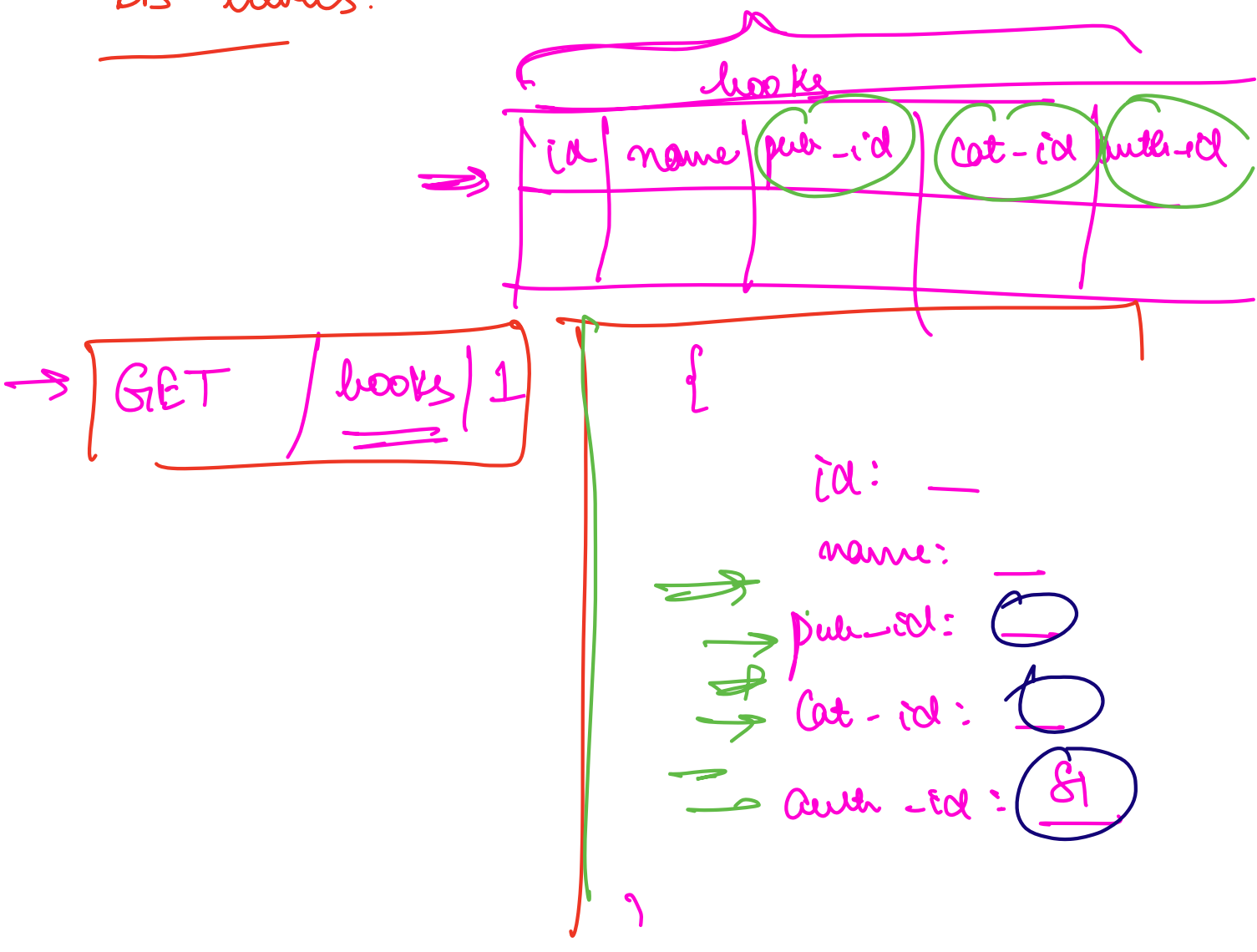
→ Every request should be independent and self-sufficient (every detail required to serve that request must be present in req itself.

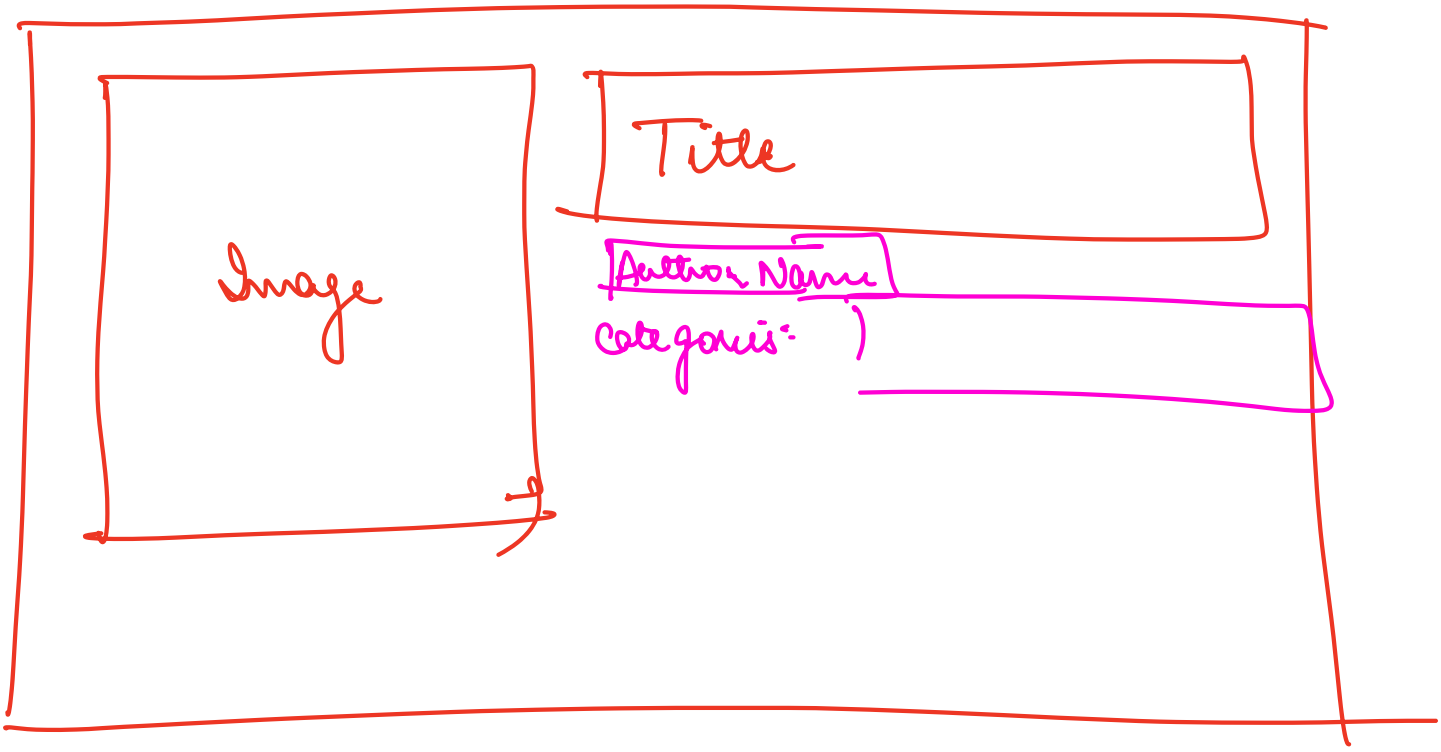


→ it allows systems to scale.



(3) Dont always have 1:1 mapping with DB tables.



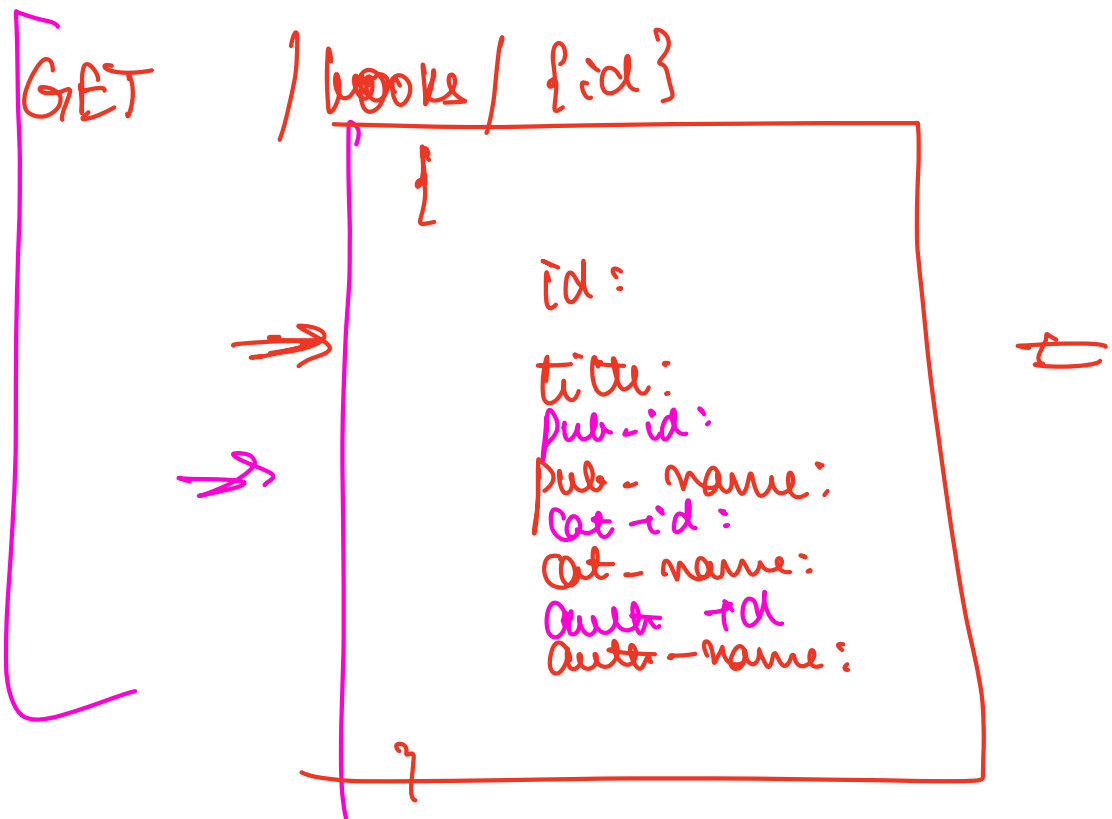


GET /authors/ &

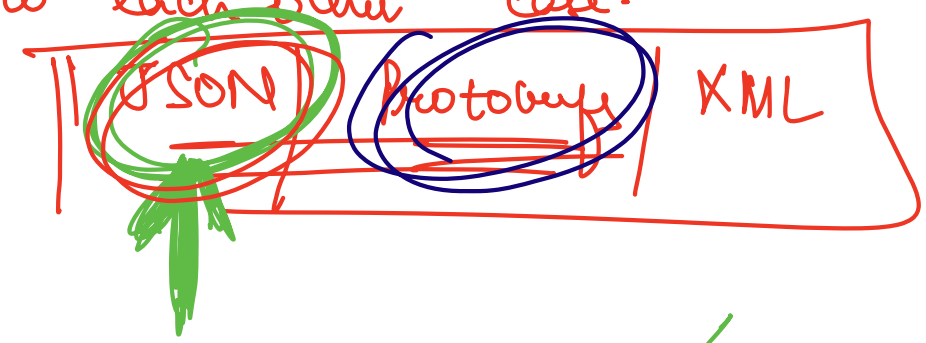
→ Chatty APIs ⇒ BAD Practices

→ Too many req to backend.

GET /books/{:id}



(4) REST APIs will use an exchange format to talk to each other. Ex:



→ easier to read

→ smaller in size

```
{
    id: ___
    name: ___
    desc: ___
}
```

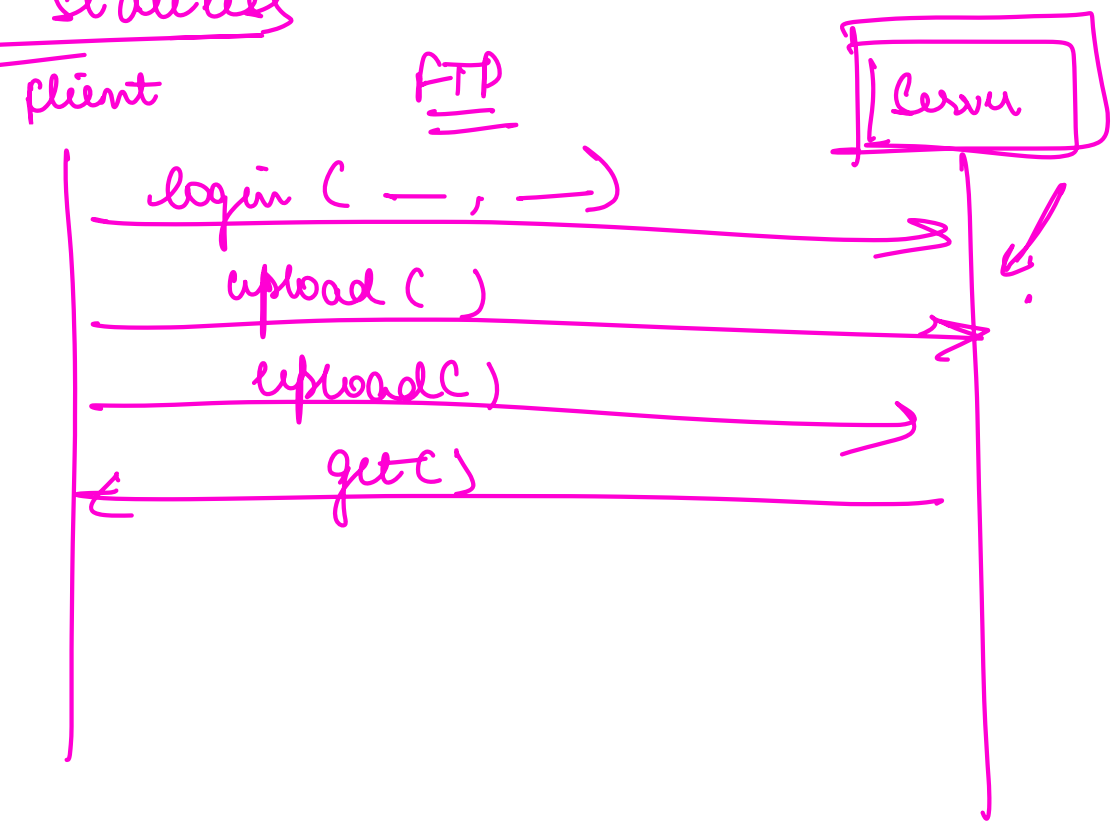
```
<user>
    <id> 1 </id>
    <name> Naman </name>
    <desc> ___ </desc>
</user>
```

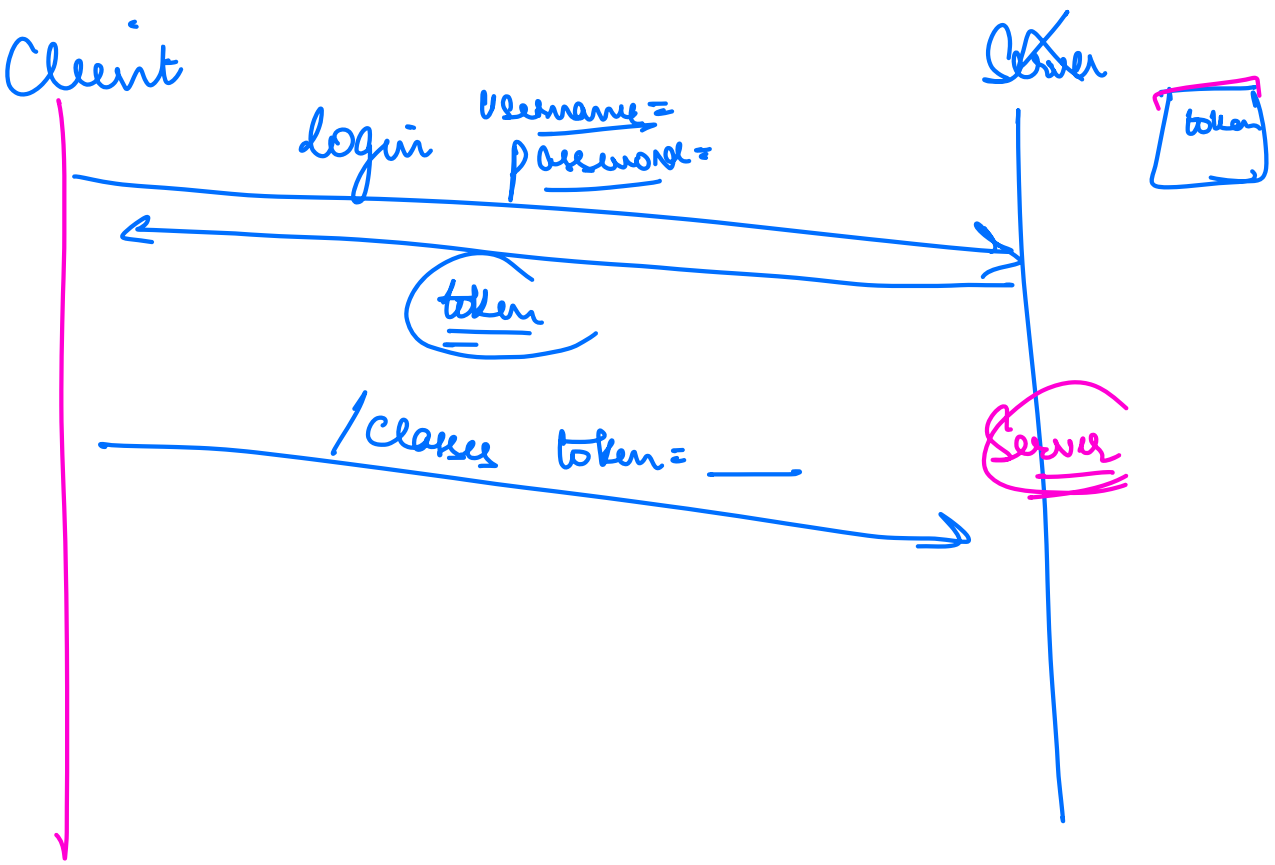
(5) Endpoint should be plurals

/users/1 ✓ Good

/user/1 ✗ Bad

HTTP is Stateless





Product Service

→ Product

→ Category

Product
- id
- name
- String details
- price
- quantity
- category

Category
- id
- name

Product
- id
- title
- desc
- image
- category
- price

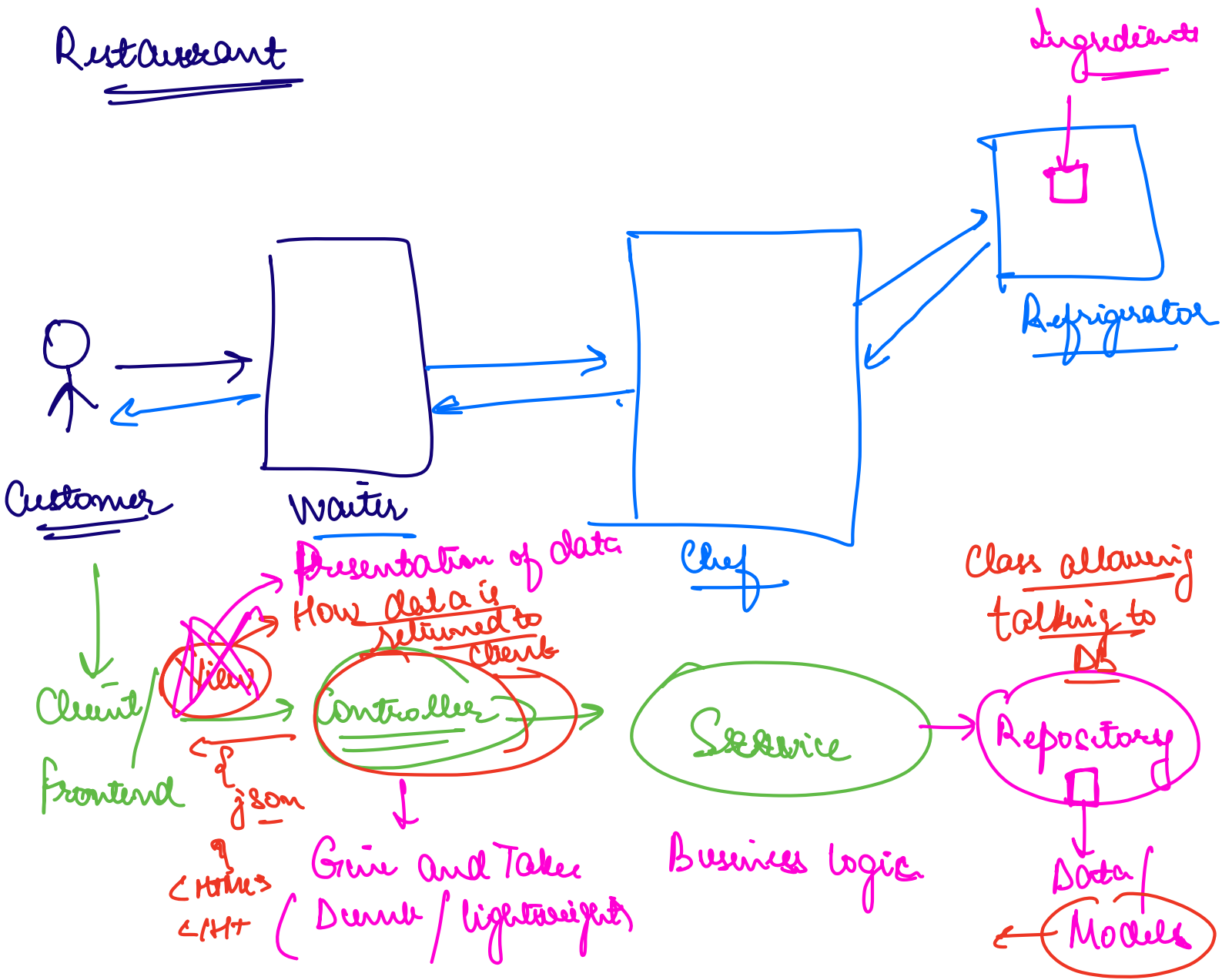
Category
- id
- name

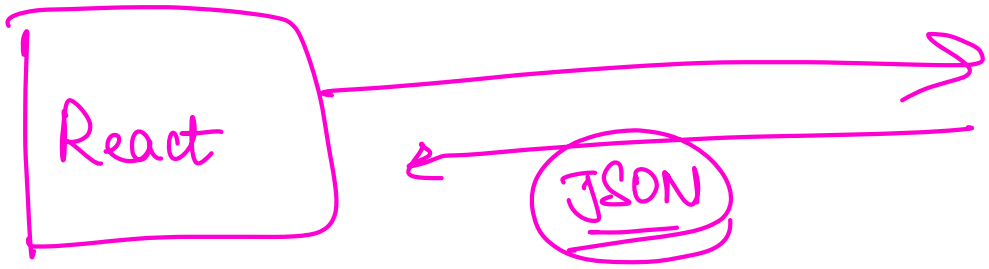
MVC

M → Models
V → Views
C → Controllers

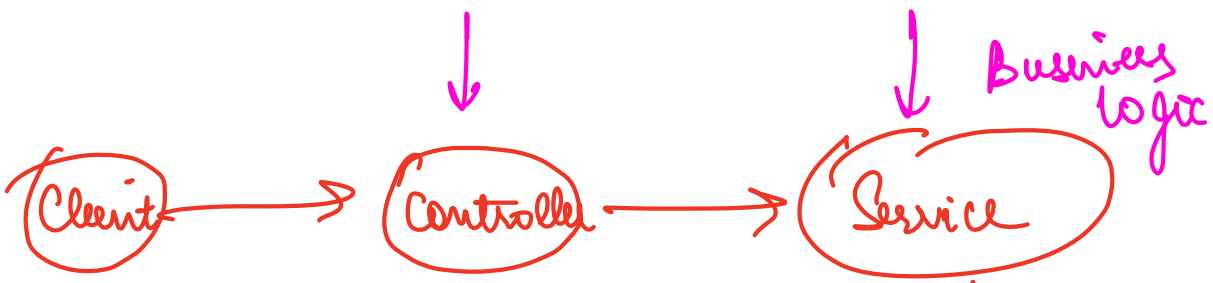
⇒ way to divide the different resp of doing different things when serving a request to diff classes.

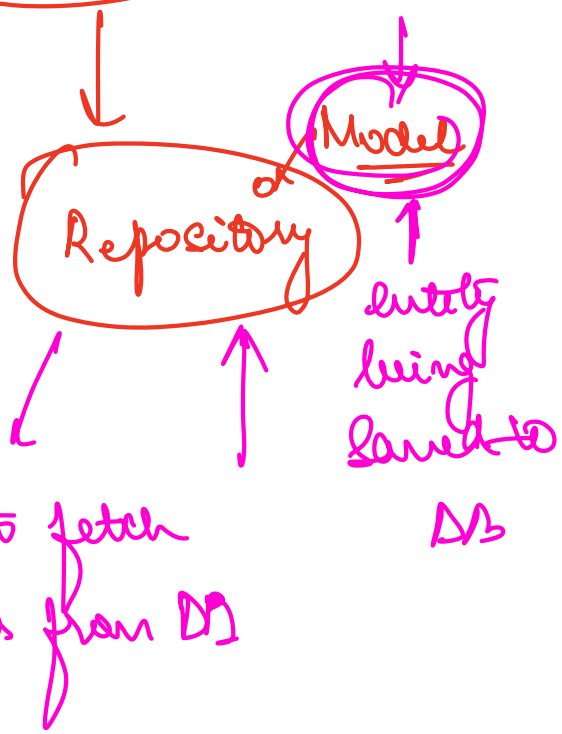
Restaurant





These days how data will be displayed is a part of frontend code.





Dispatcher

- get request from DC
- → figure out which controller should serve
- → call that controller
- → send data controller returns over N/W.

Sat → 11AM → 12:30 PM → Integrating REST APIs

↳ Complete Product Service