HTML. css. Js
Libraries (.Jar file)
↑ web.xml
contain everything needed to run an app
↑
Web application Archieve
↑
work as a war file
↗ run .war file

Tomcat (Server Container)
i.e. node.js runtime

Servelet
↳ handle request & response.
↳ i.e Express.js

Servelet vs. Tomcat

Java Spring

IoC
inversion of control
↑
developer vs. computer
allow computer to control create/delete of obj.
↑
JVM → IOC container will manage these → make use of dependency injection principle. → Springboot manage every obj created is called bean
↳ Spring Application.run ↗ start IOC container
↳ Application Context
↳ to access Container

dependency injection principle.
↓
Avoid tightly Coupled system
↓
new Service().
↓
If Name of Service is changed Need to update everywhere.

project manager i.e. Npm in Node — Maven. Gradle. Ivy — has lifecycle of build test. etc.

Maven Archetype [Ex. prototype] — templating tools → All dependencies are Jar file → Java Archive file
↳ Jar similar to zip
{ 1) compress into 1 file
2) managed by PM

GAV → Group ID. Artifact, version.
↳ Need this to create unique ID for Jar file → not conflict with other dependen.

transitive dependency
↘ App doesn't depend it. but Lib depends on
↳ transitive → carry over / pass them
e.g. You trust A, A trust B, you trust B

POM
↓
project object Modal (similar to package.json)
↳ developer edit POM, Maven use actual effective POM

.M2
↓
local dependency cache
↓
insert into <table> values (1, "Navin, 34);

prepared statement
↙
prevent SQL injection
↓
convert all ~~stri~~ user input into string
↓
it will not be executed as sql.

Java Spring

diff http services

client not interested in many API just call 1 Api

API

Hide complexity behind the scene.

Api gateway

AOP → Aspected oriented programming → separation of control → insert extra function → but not rewrite the code

Service registry (SR) → who is calling whom → when Microservice start, register Name. IP/ port (can change)
↓
a MS wants to call to another, it will call (SR) for latest location of its service [dynamic]

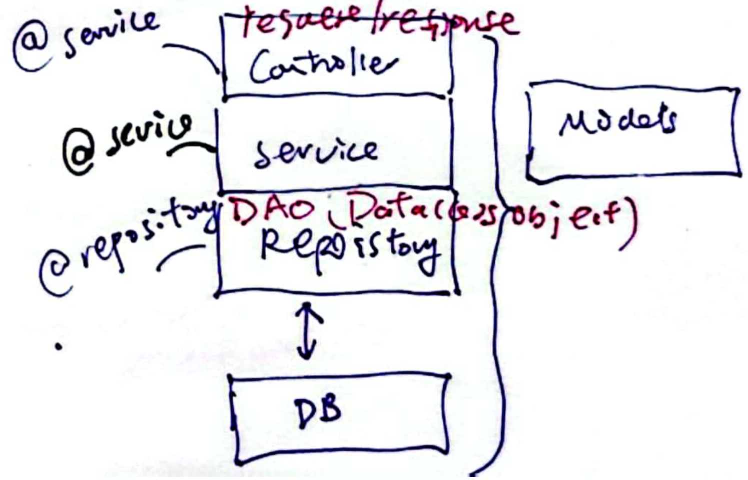
Lombok → reduce no of boilerplate Java code

what is @autowire → used in sub class of a @component class ↳ tell IOC container to create a bean

JDBC → Java Database connectivity (DB)
↳ connect app to DB.
↳ use raw JDBC is quite difficult
↓
now use ORM hibernate

3 ways to manage bean
① Java
② XML
③ Annotation Spring boot

different layer

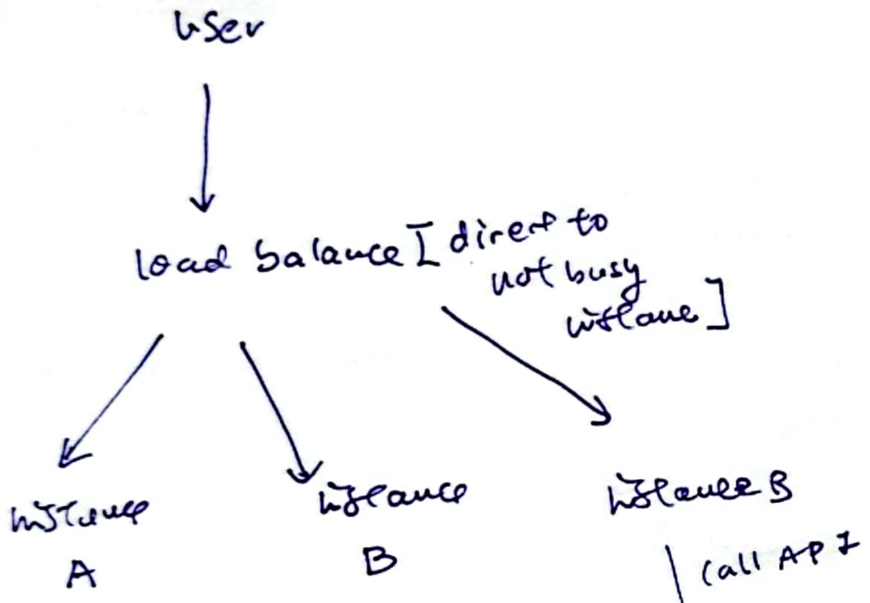


Map a class to a table
↓
JPA
Java persistence API
↓
Java obj □ → Map DB □

create sql schema
~~rest~~
~~application property~~
↓
schema.sql
↓
can be executed by JDBC

application.properties
↓
config the file to connect to DB
↓
url
UserName
pw.
driver

Micro service

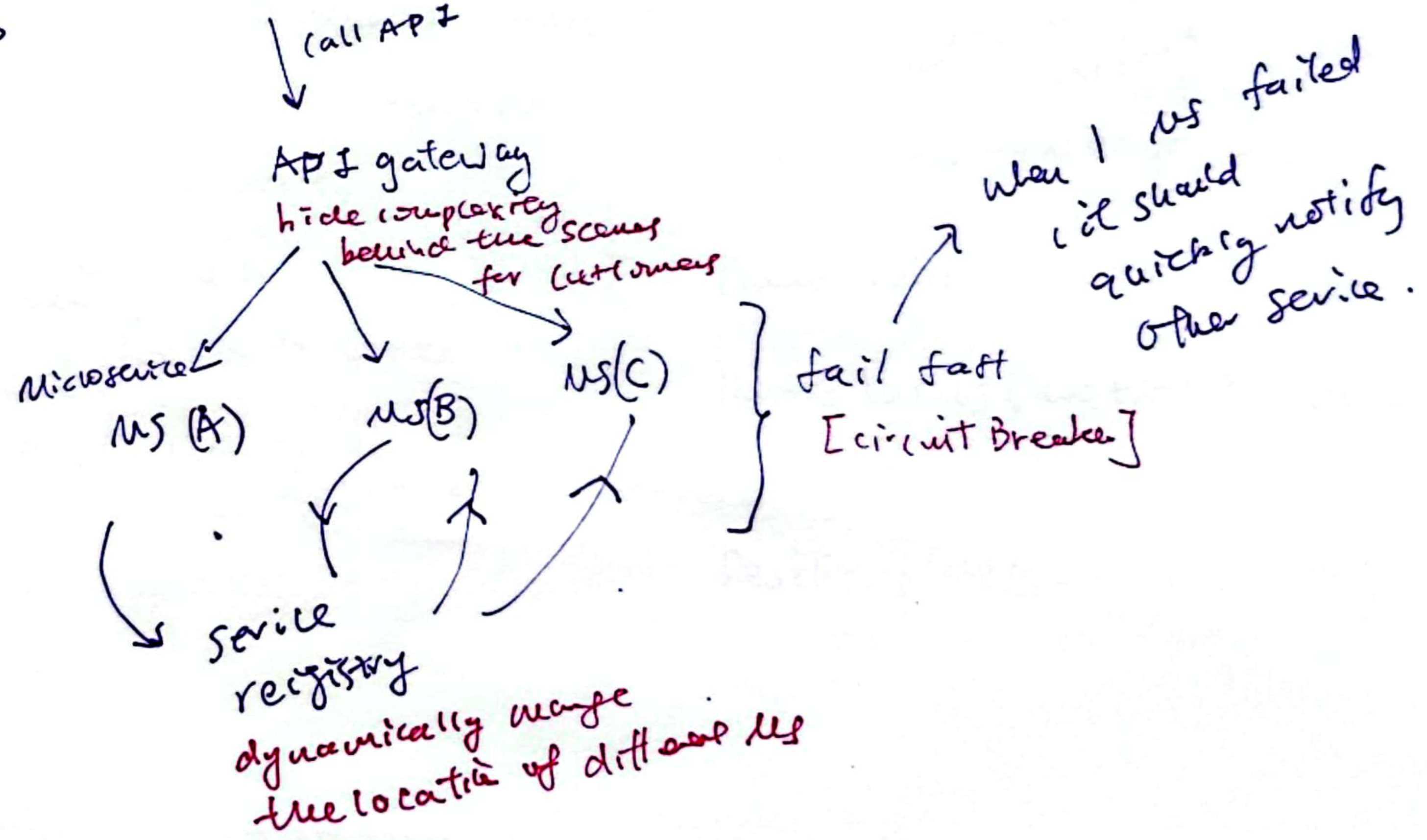


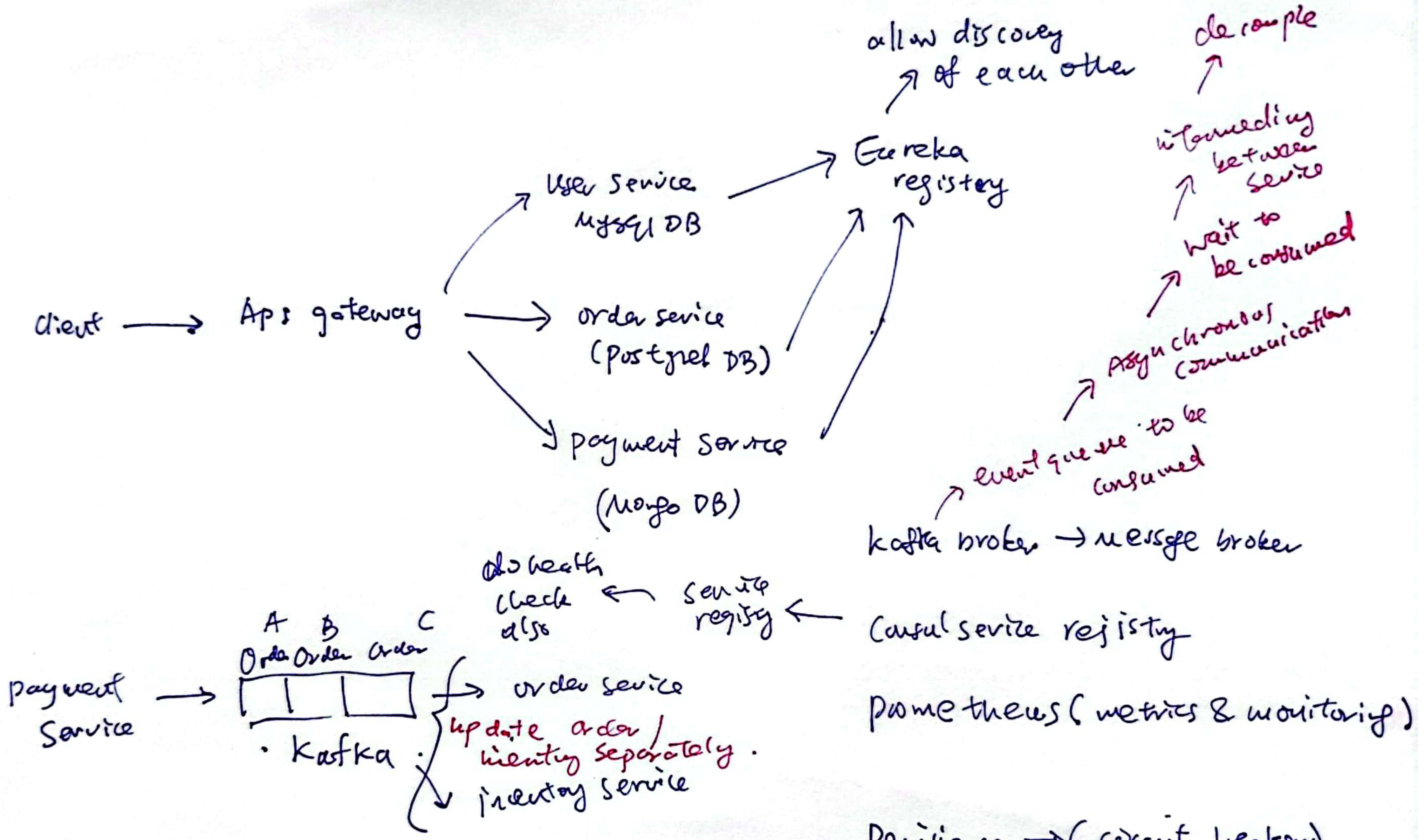
Forward Proxy → like VPN

$$\begin{tikzcd} \text{client} \arrow[r] & \text{~~prox~~ forward proxy} \arrow[r] & \text{Server} \end{tikzcd}$$

Reverse proxy → receptionist to handle request.

$$\begin{tikzcd} \text{client} \arrow[r] & \text{reverse proxy} \arrow[r] & \text{server} \end{tikzcd}$$


call API
API gateway
hide complexity behind the scenes for customers
Microservice
MS (A)
MS (B)
MS (C)
service registry
dynamically manage the location of different MS
fail fast [circuit Breaker]
when 1 MS failed it should quickly notify other service.


allow discovey of each other
decouple
intermediary between service
wait to be consumed
Asynchronous communication
event queue to be consumed
client
Api gateway
User service MySQL DB
order service (postgres DB)
payment service (Mongo DB)
Eureka registery
kafka broker → message broker
do health check also
service registry
Consul service rejisty
A
B
C
Order
Order
Order
payment service
Kafka
order service
update order / inventory separately.
inventory service

Prometheus (metrics & monitoring)

Resilience → (circuit breaker fail fast)