

Spring framework -- boot

Managing the dependencies

starter projects

Spring Initializr

>mvn <options>

>mvnw <options>

JSP-jstl view template

ViewMarker

Tiles

Velocity

Thymeleaf

Mustache

starter-parent

Spring boot config

1. through dependency in pom.xml

certain default config is auto activated

2. activates the key-value (properties) based config

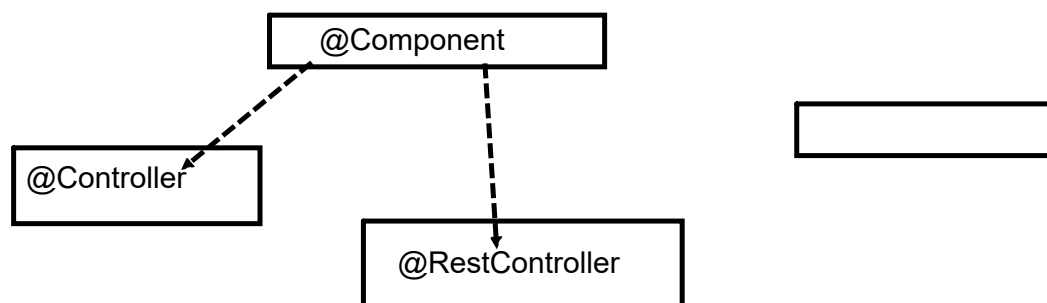
application.properties

application.yml

3. std folder structure

4. Exposes specially curated Annotations

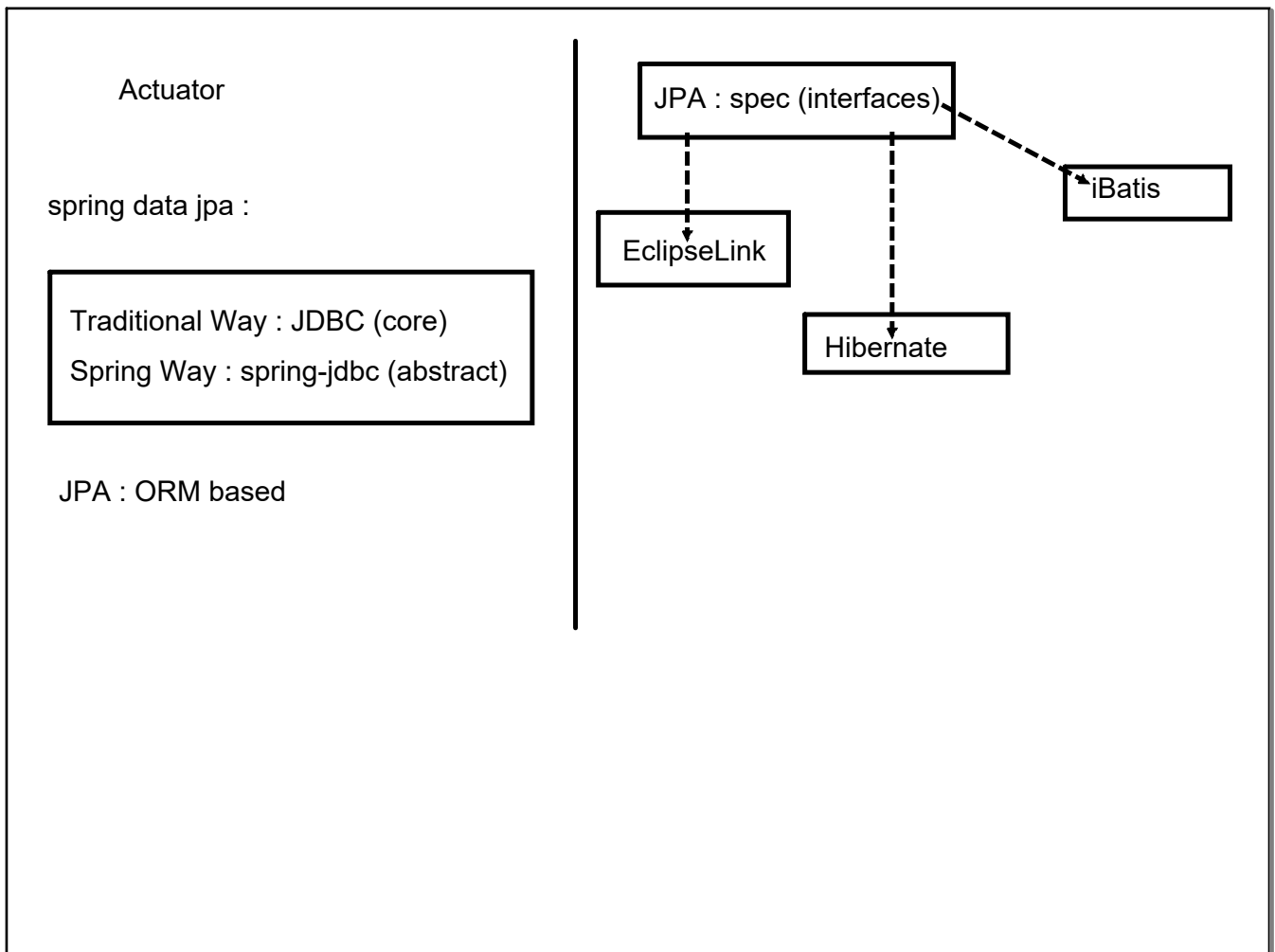
@component

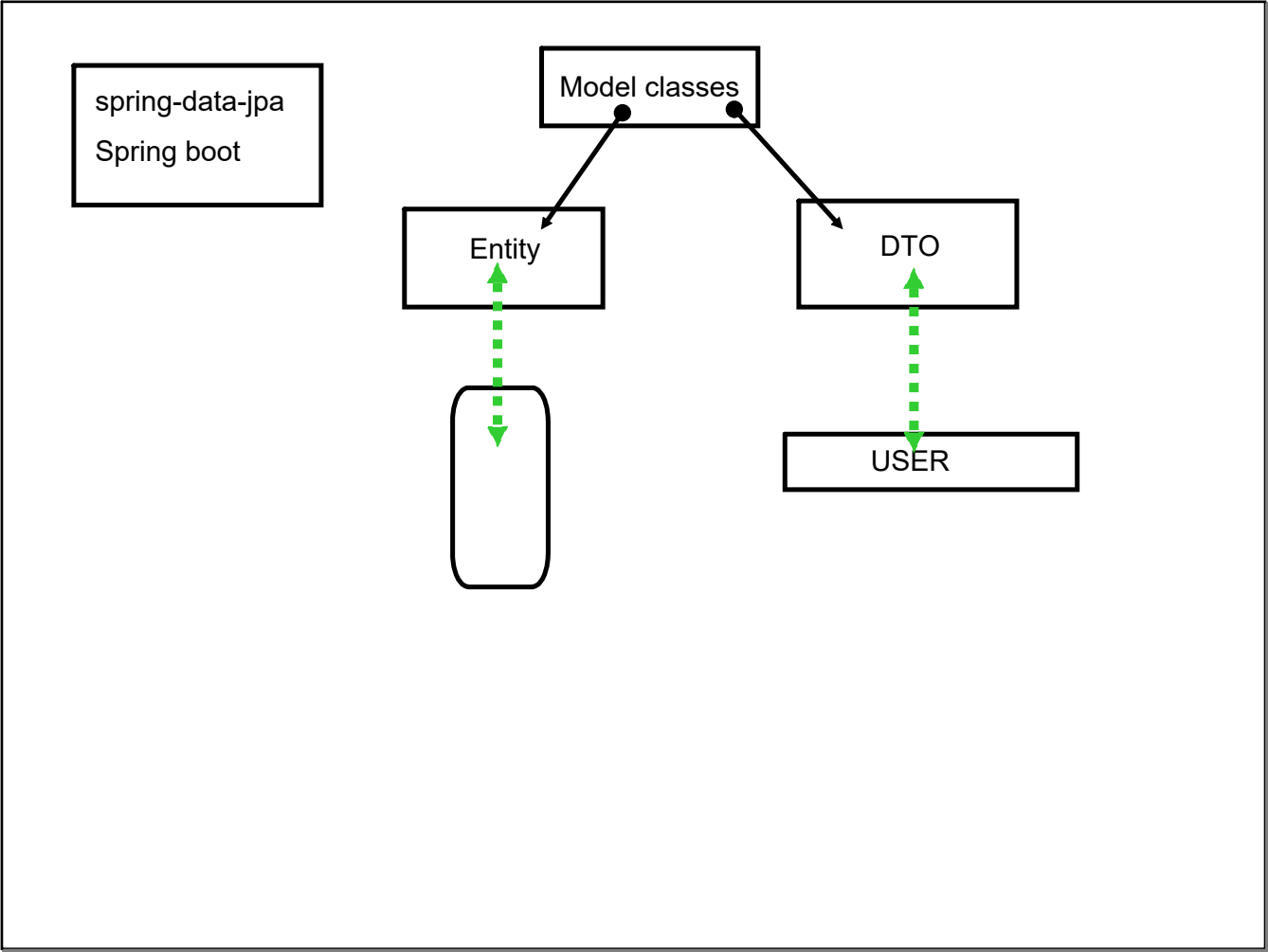


Response is treated as data (Response Data)

supports auto conversion of POJO & JSON (jackson-databind project)

devtools





Any DB interaction needs JDBC config

db name

location

user name

password

driver (auto detect the driver based on URI)

JPA

custom config for JPA

Hibernate

custom config for Hibernate

SQL : dialect

DataSource

(interface)

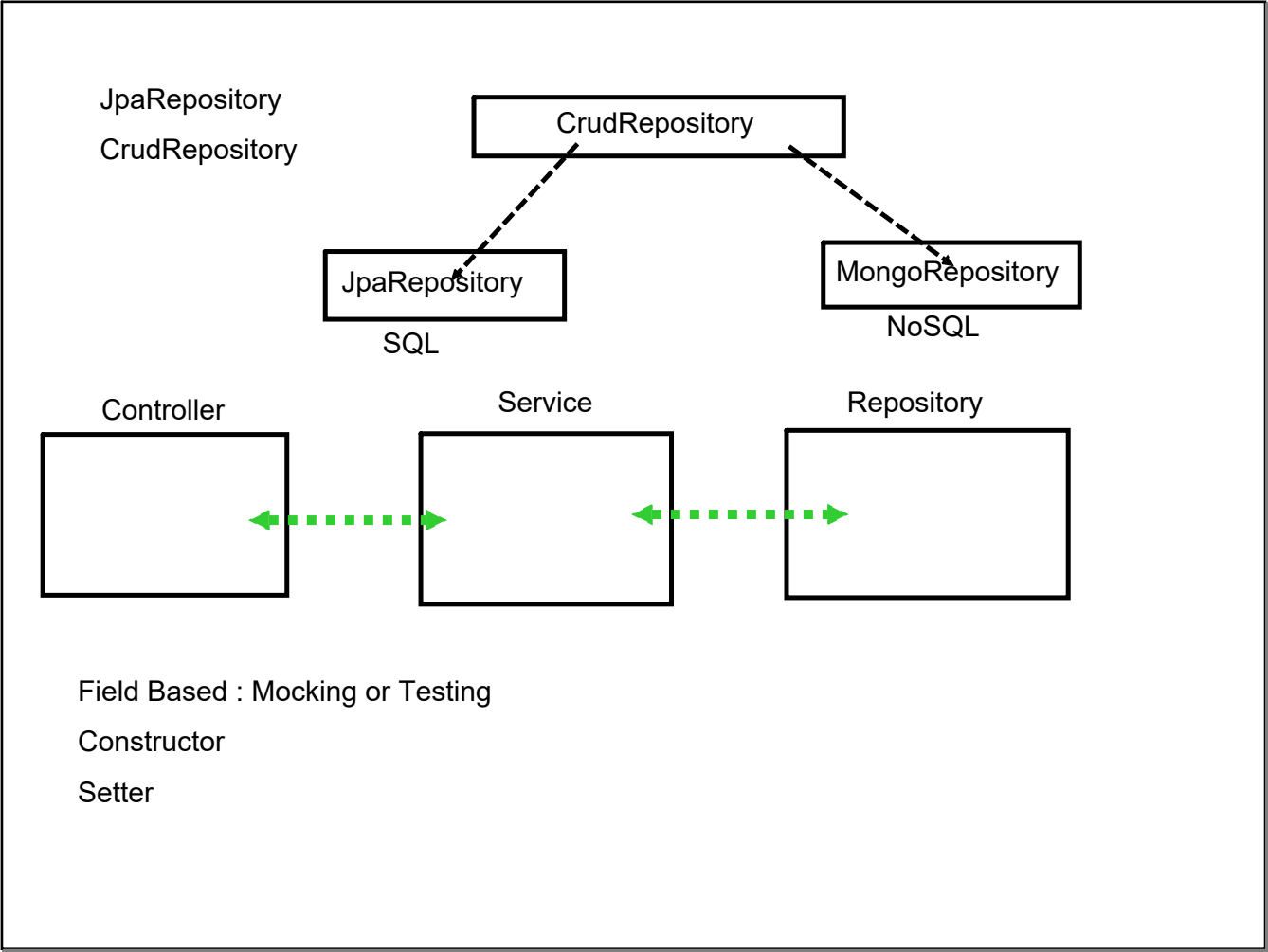
JPA--> Repository : Entity

Interfaces containing basic
CRUD functionalities pre-coded

Custom Interface and inherit
Repository interface

#associate with entity

platform for custom
implementation



Product Controller

CRUD functionality

/products : GET : fetch all records

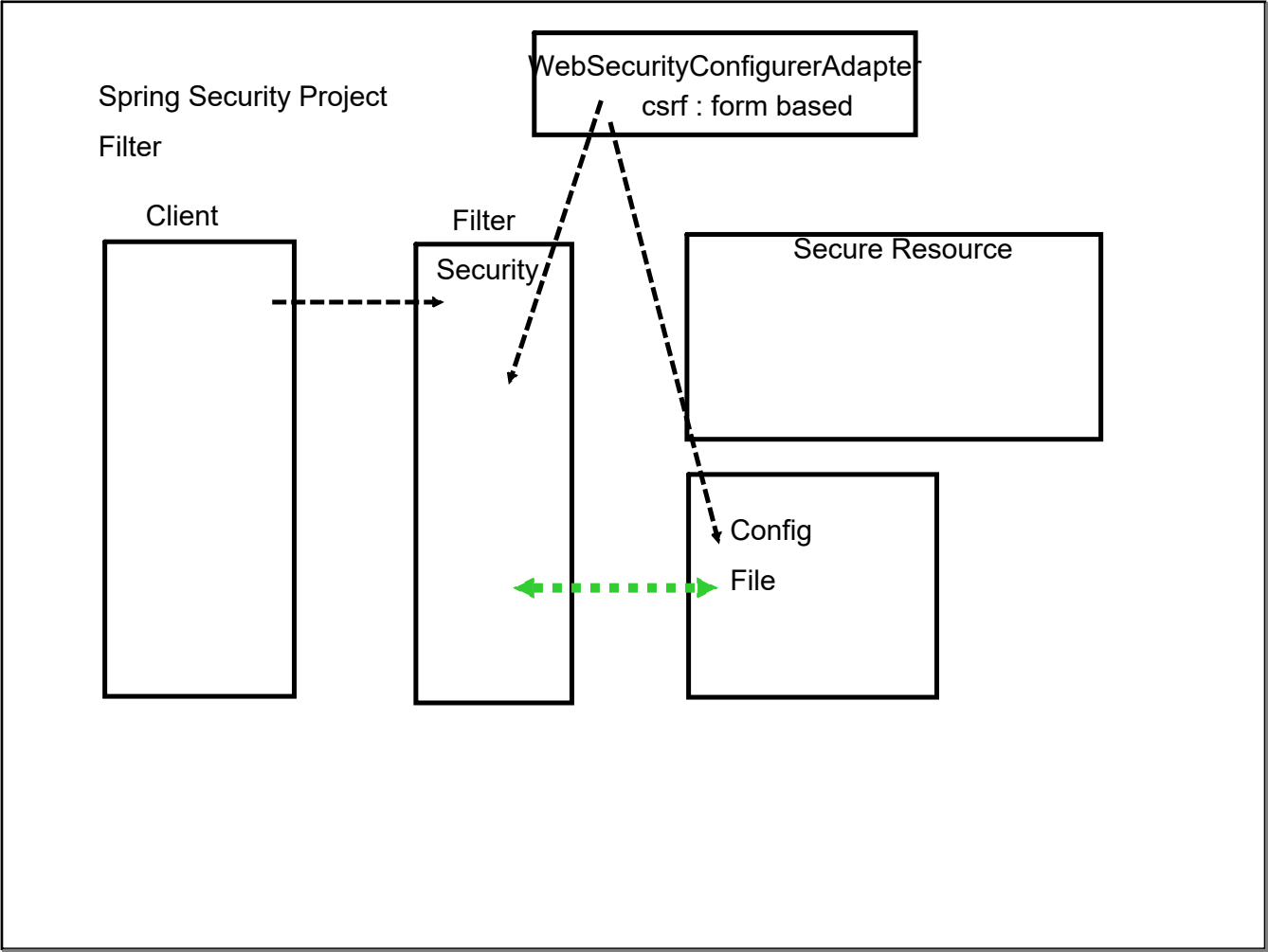
/products/id : GET

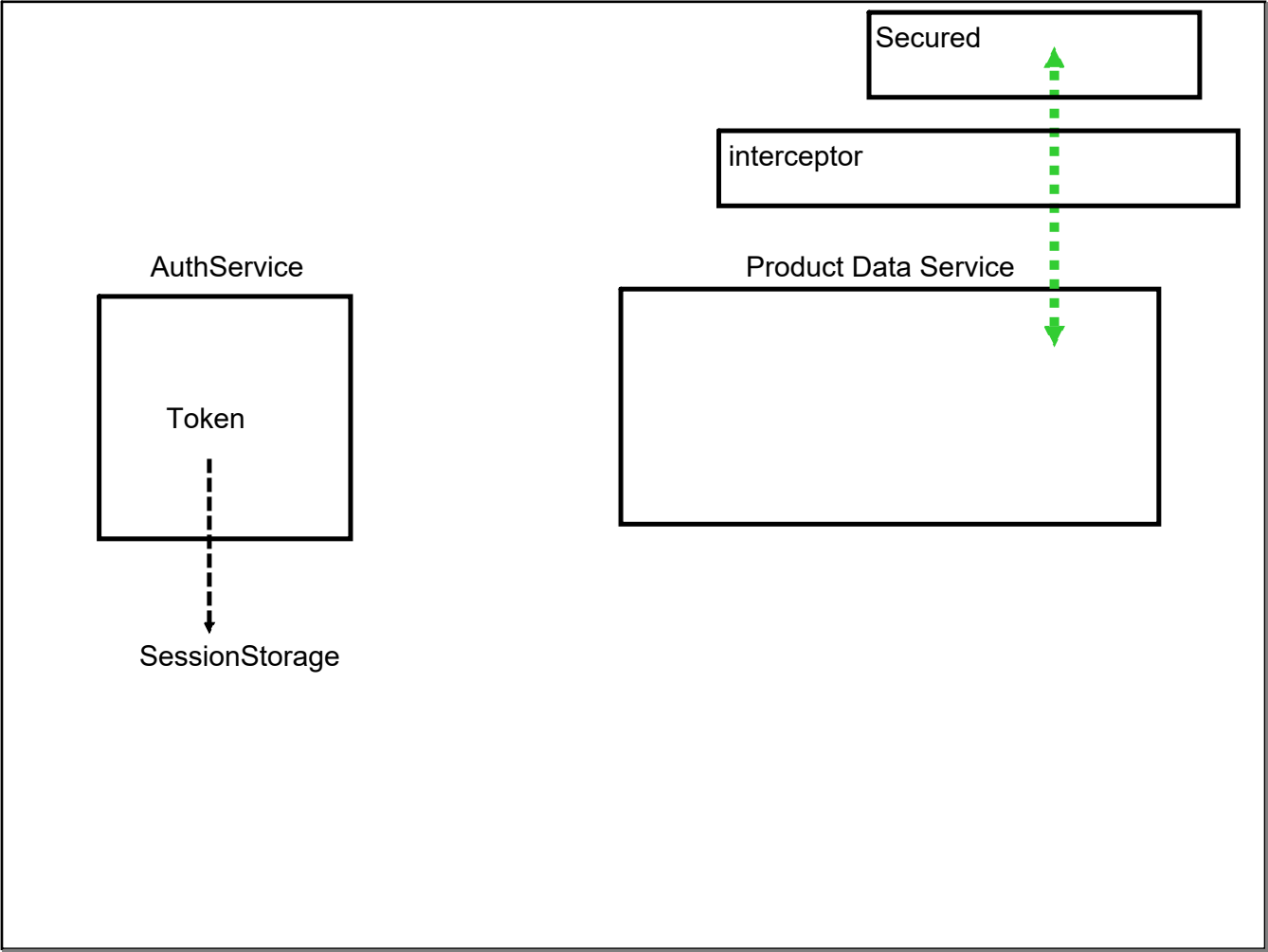
/products : POST

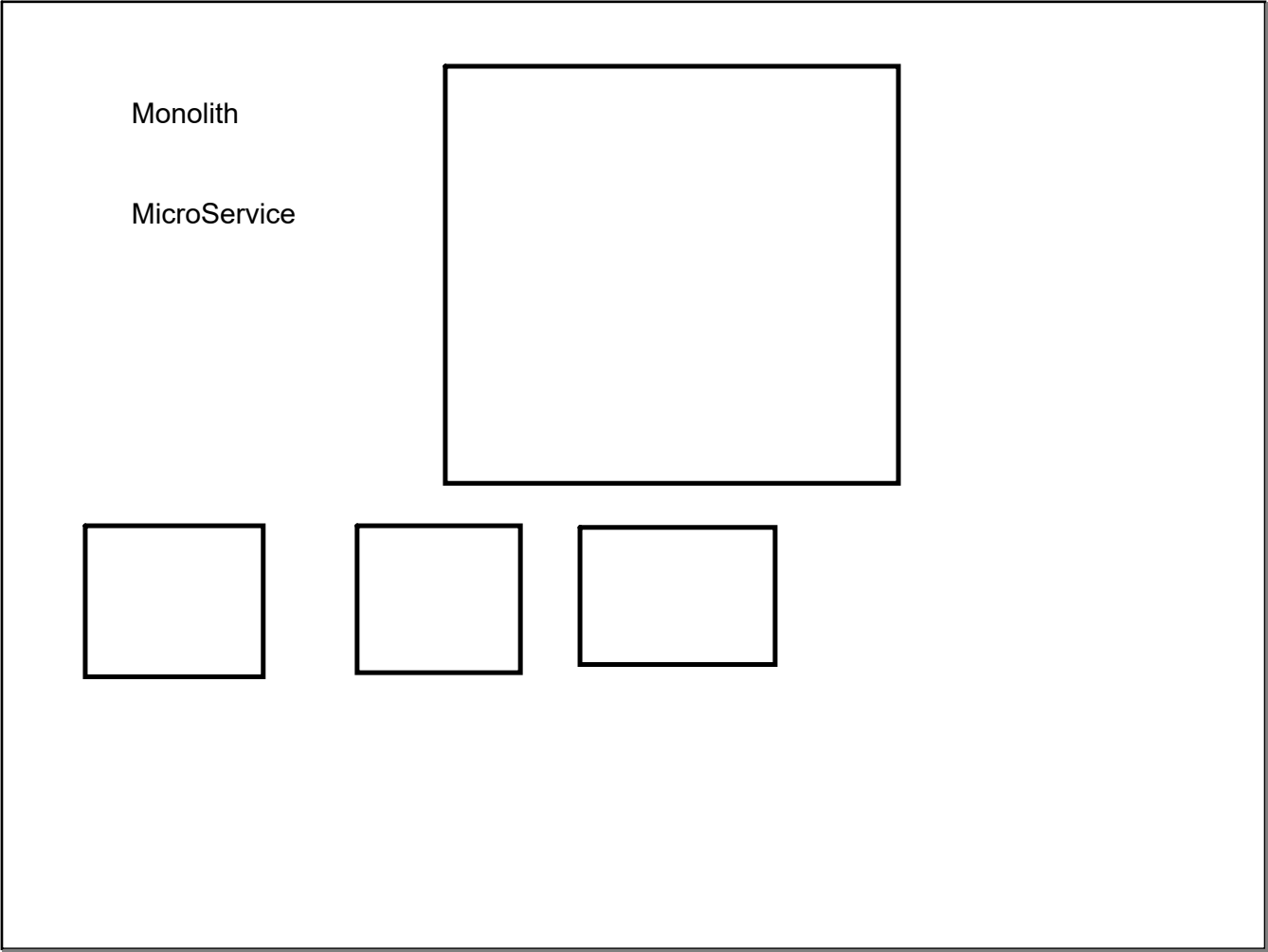
/products : PUT

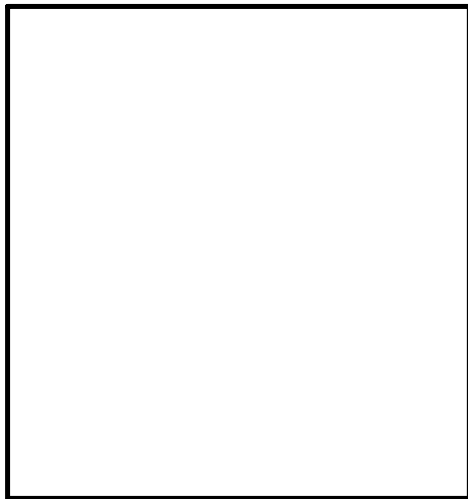
/products/id : DELETE

Status Code
Headers
Response Body









High possibility of bugs

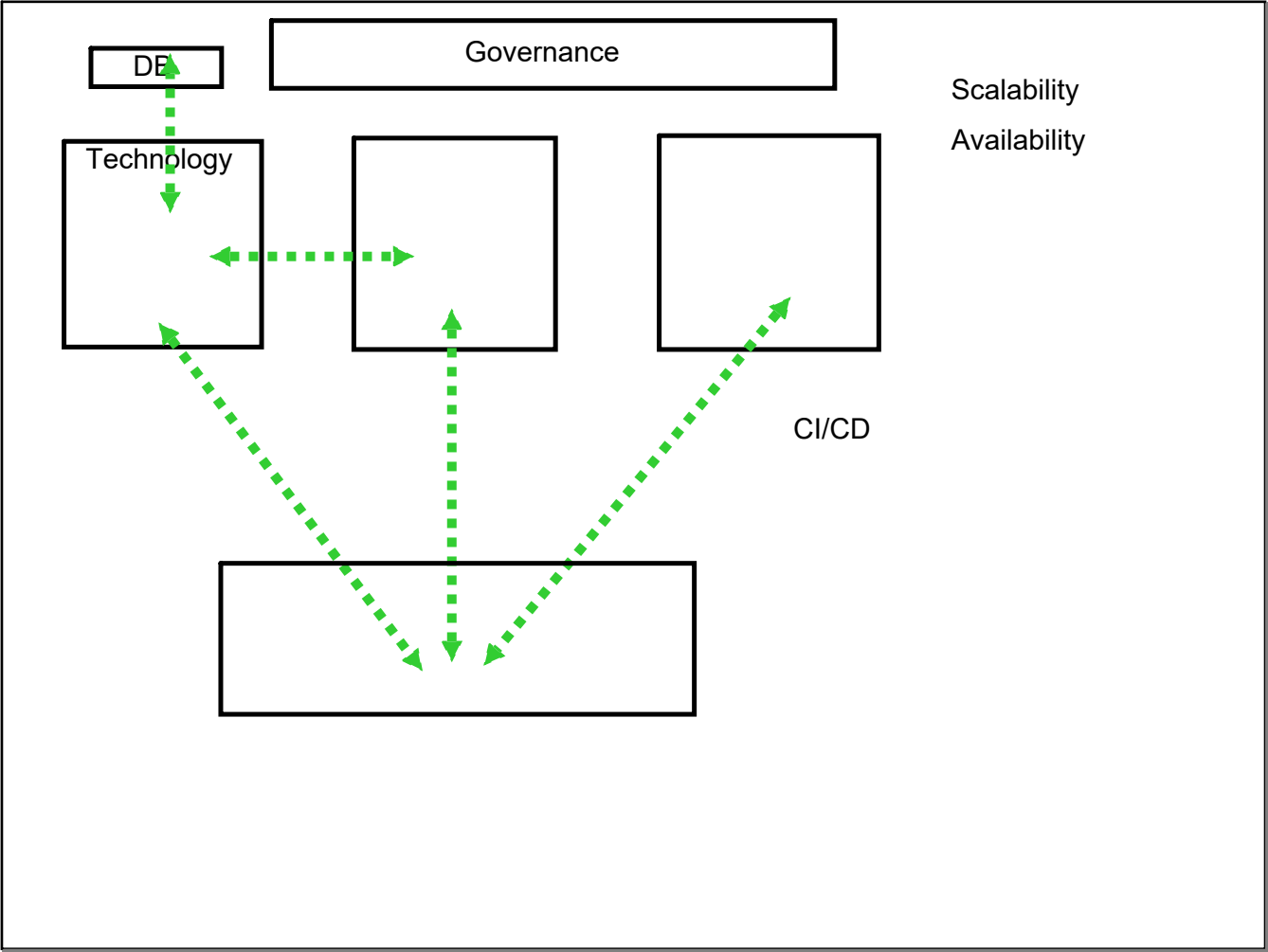
Tight coupling

Huge in size

May result into complete crash

Scaling

Technology bound



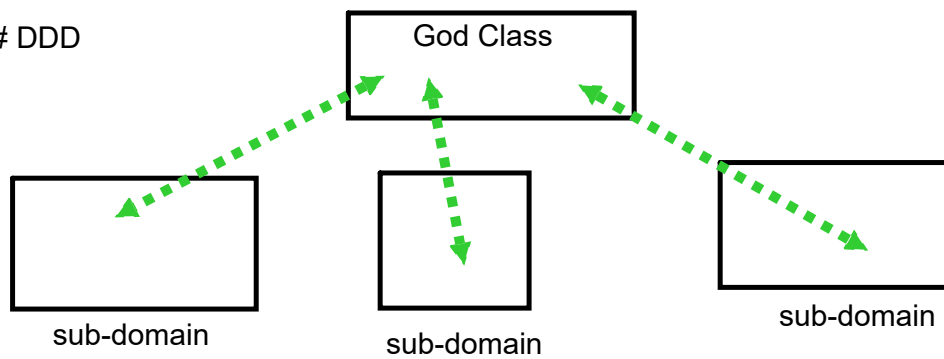
Decomposition

based business capabilities : business oriented rather than technical

"God Classes" :

Order class

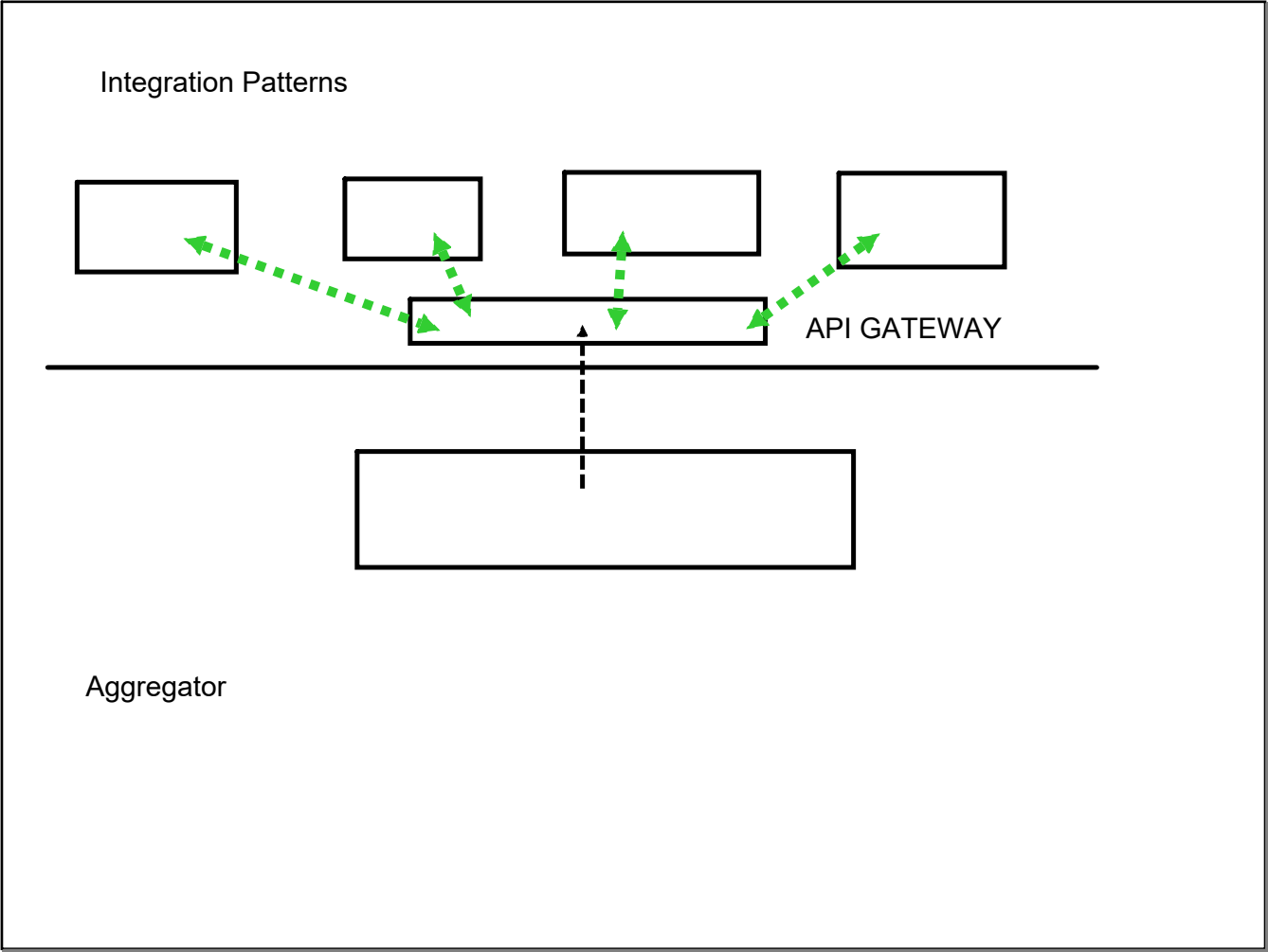
DDD

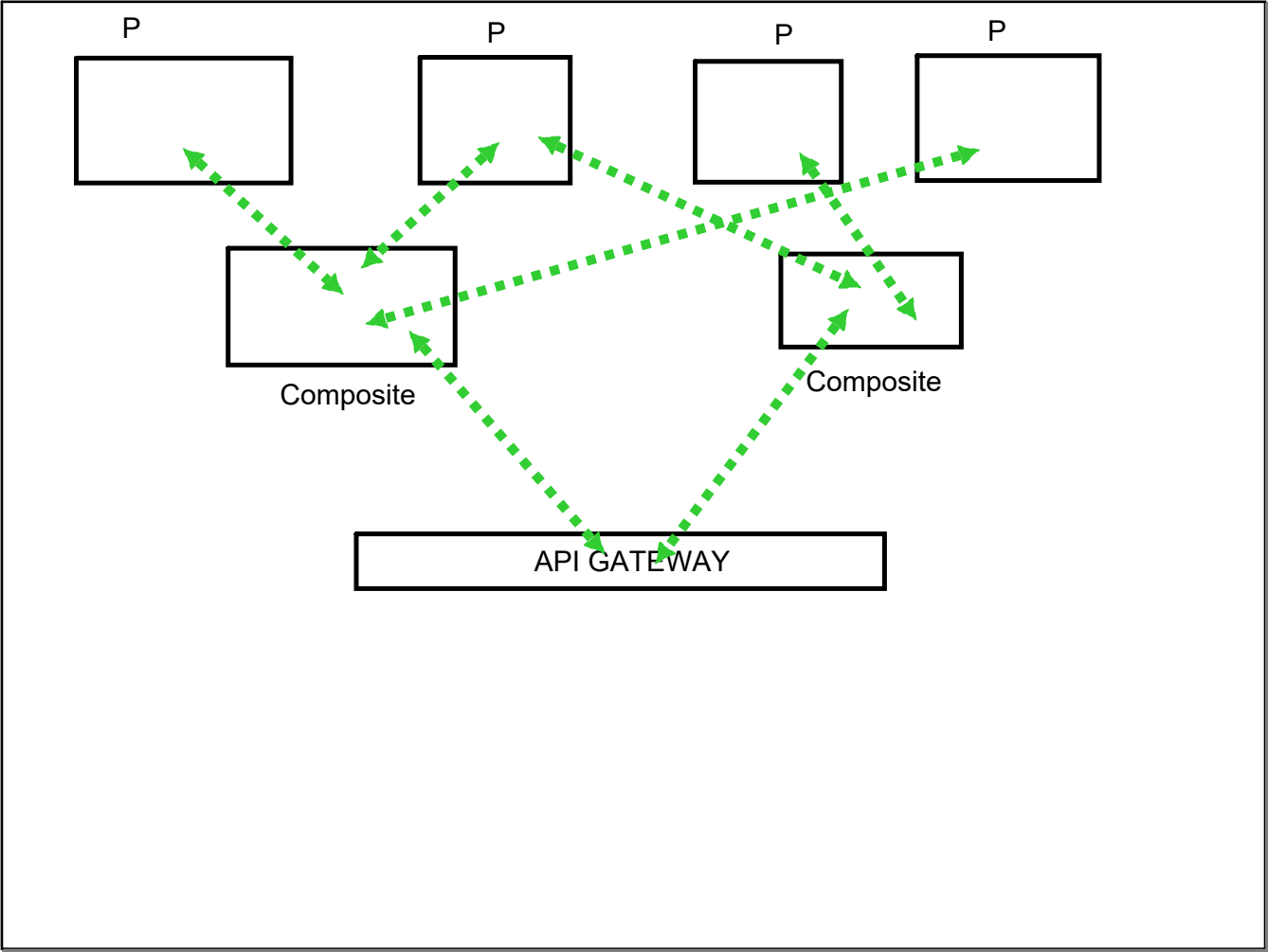


greenfield

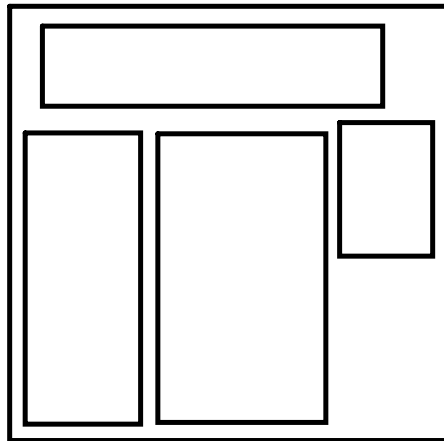
brownfield application

#Strangler pattern





Client-Side UI Composition Pattern



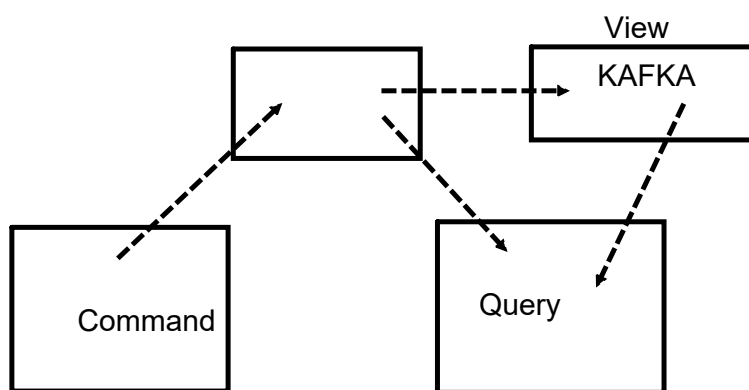
SPA : Angular/React

Database Pattern

- a) One DB per service
- b) Shared DB per service
2-3 microservice

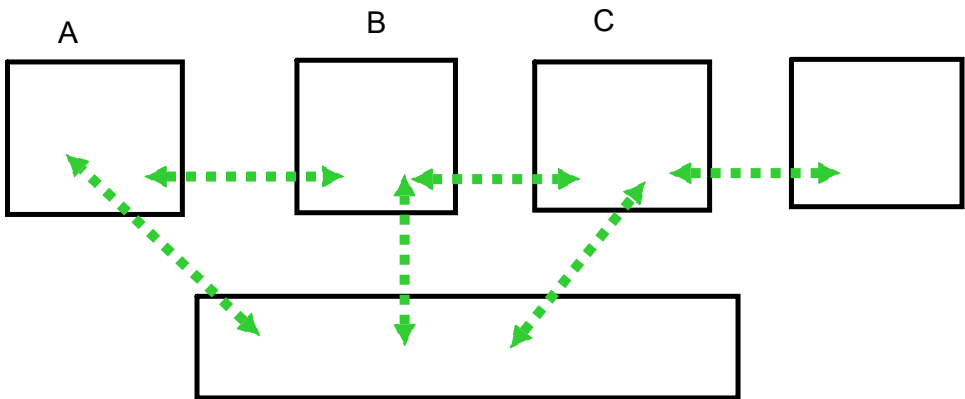
c) CQRS (Command Query Responsibility Segregation)

Event Sourcing



d) SAGA Pattern

1. Choreography

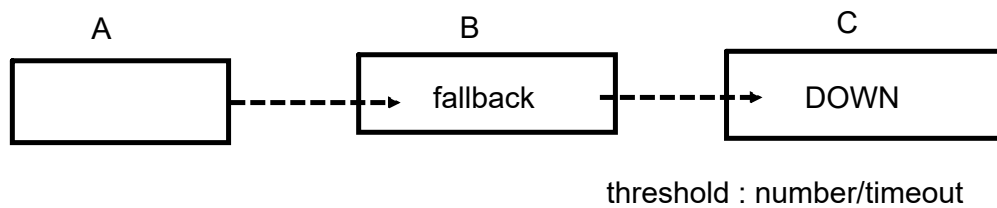


2. Orchestration

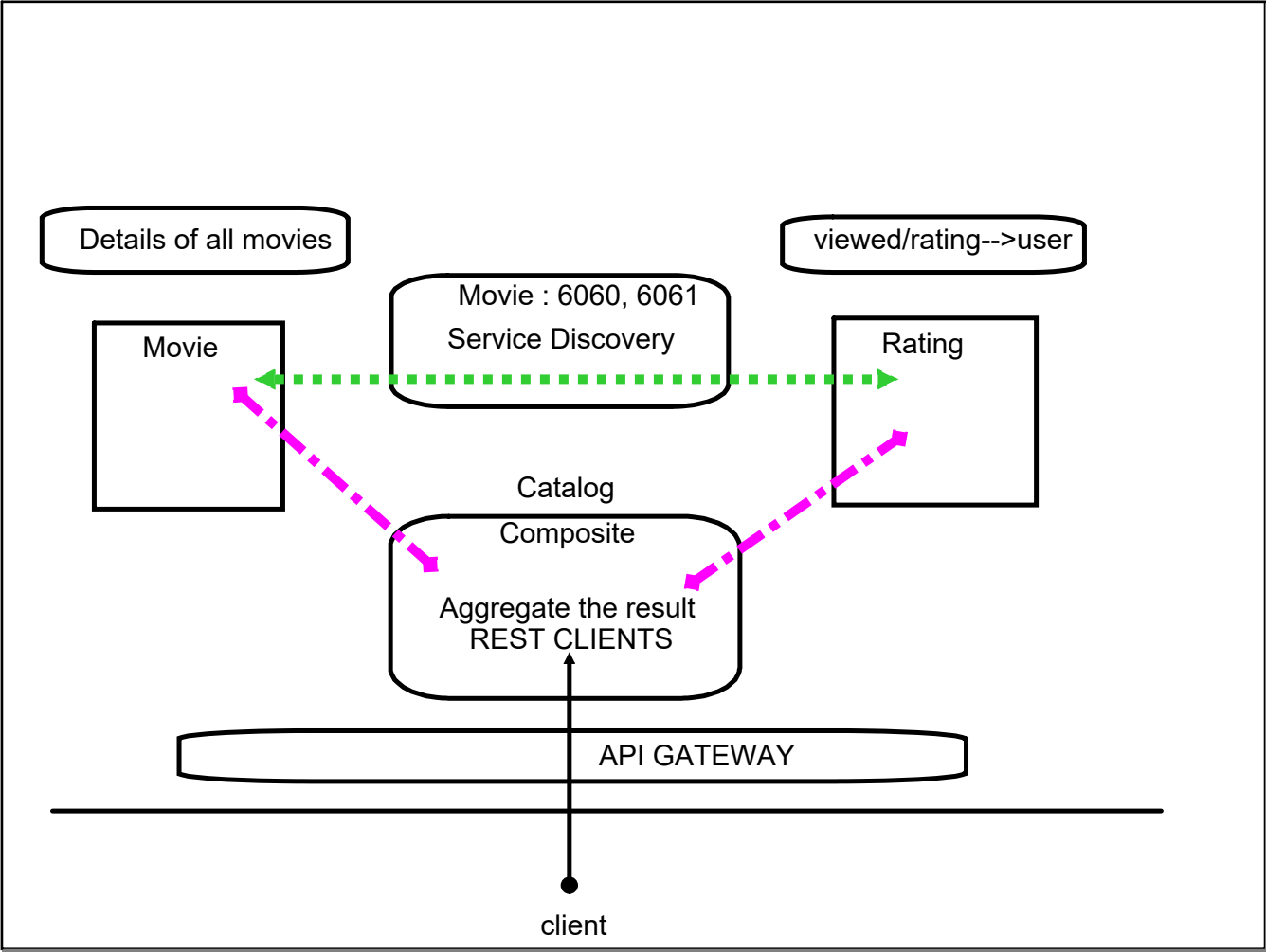
Observability Patterns

- a) Log Aggregation
- b) Performance Metrics
- c) Distributed Tracing
- d) Health Check

Cross-Cutting



1. External Configuration
2. Service Discovery Pattern
3. Circuit Breaker Pattern
4. Blue-Green Deployment



/movie-service/api/movies/{id} => Details/Record of movie of that id

/rating-service/api/ratings/{userId} => List of Movies(Id) and rating of that userId

/catalog-service/api/catalog/{userId} => List of Movies(Details) and ratings of that userd

