Spring framework -- boot

Managing the dependencies

starter projects

Spring Initializr

>mvn <options>

>mvnw <options>

JSP-jstl view template

ViewMArker

Tiles

Velocity

Thymeleaf

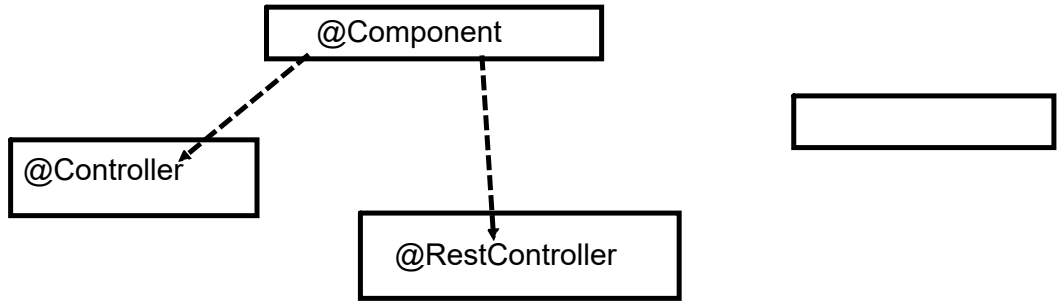
Mustache

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

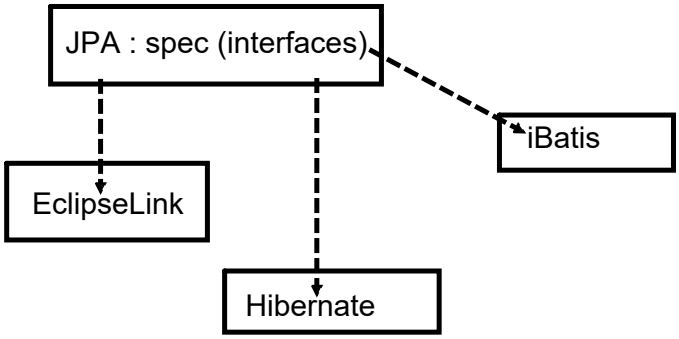
Actuator

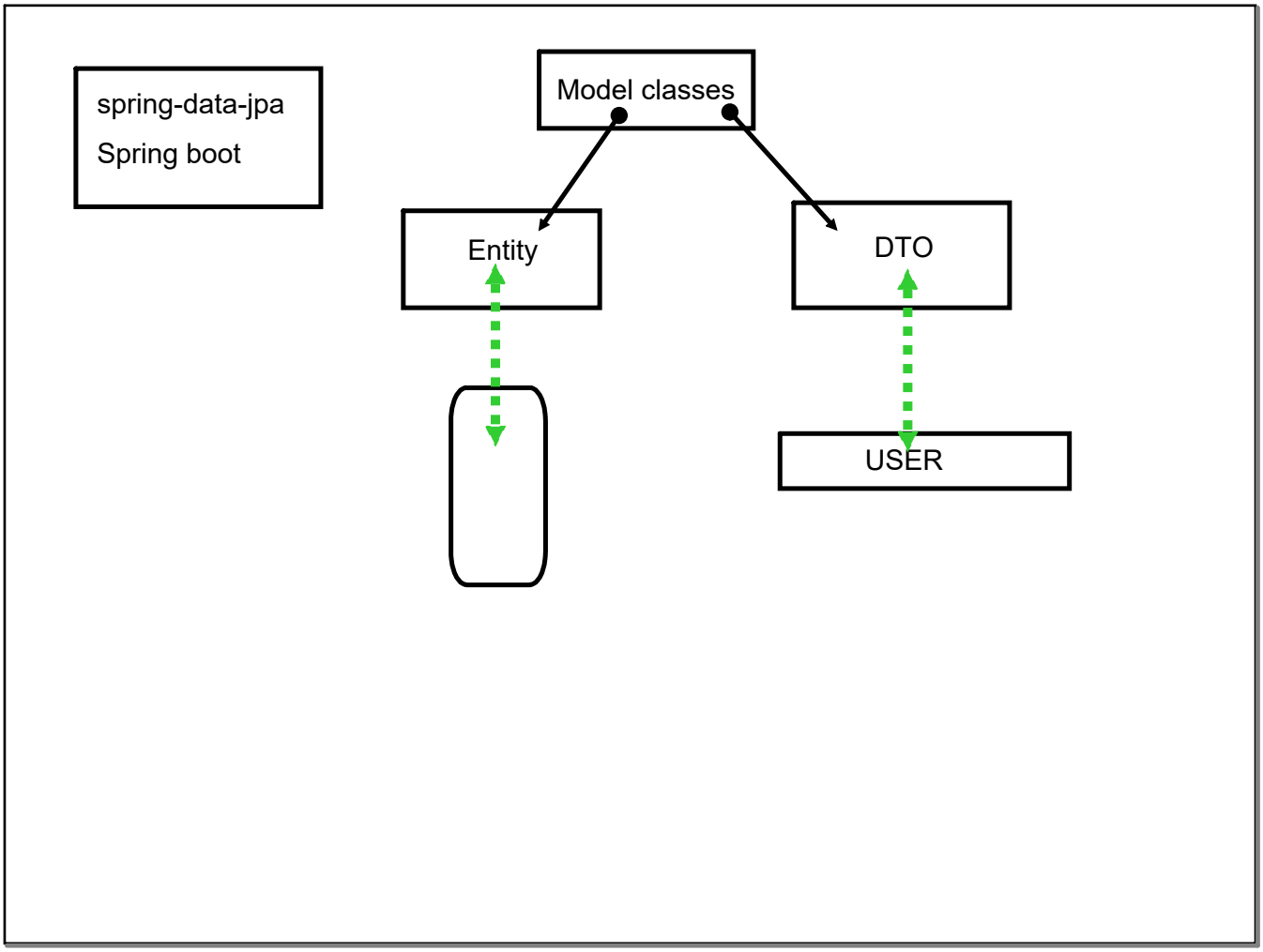
spring data jpa :

Traditional Way : JDBC (core)

Spring Way : spring-jdbc (abstract)

JPA : ORM based




spring-data-jpa
Spring boot
Model classes
Entity
DTO
USER

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

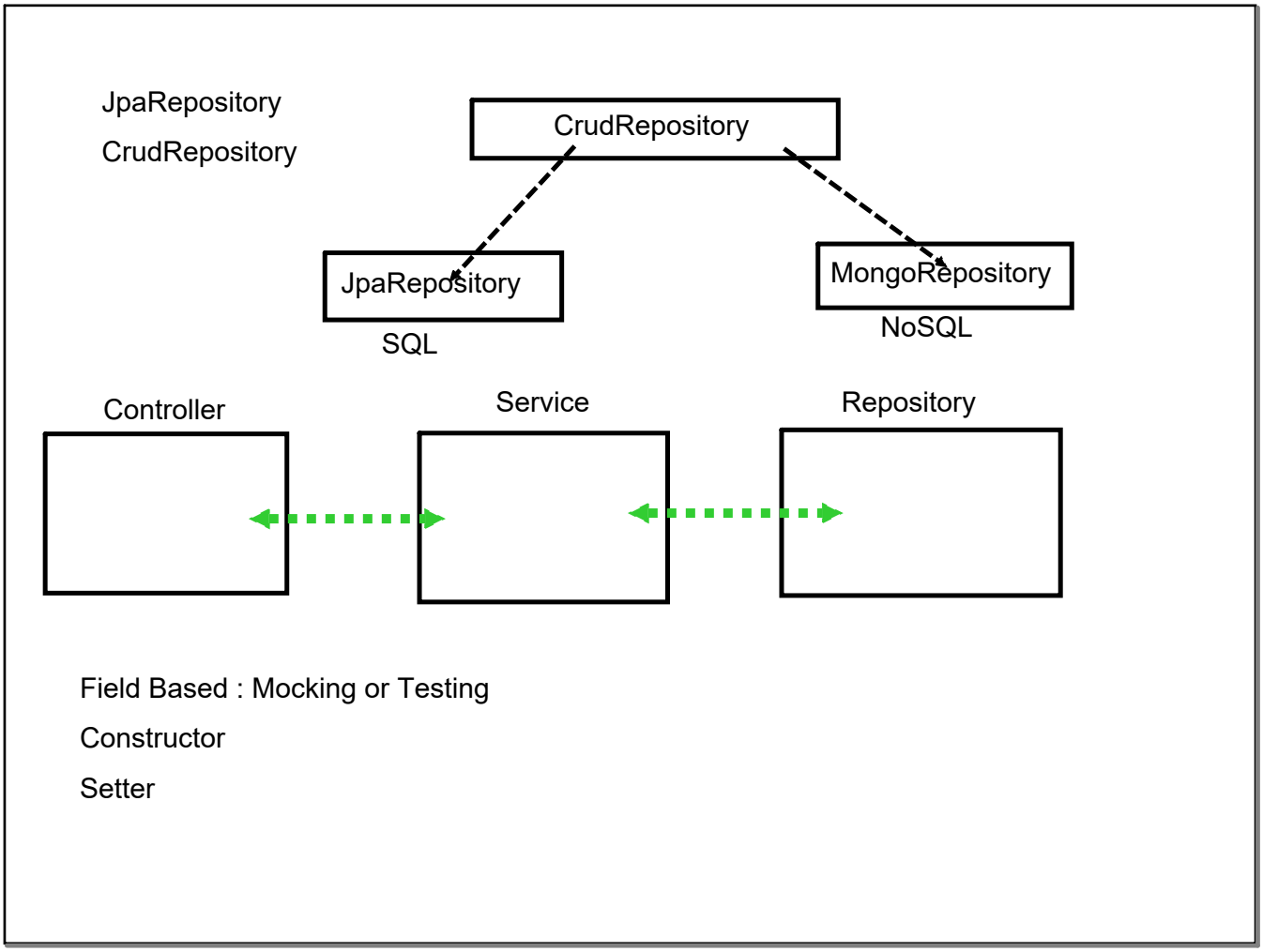
DataSource
(interface)

JPA--> Repository : Entity

Interfaces containing basic CRUD functionalities pre-coded

Custom Interface and inherit Repository interface

- #associate with entity
- # platform for custom implementation



Field Based : Mocking or Testing

Constructor

Setter

Product Controller

CRUD functionality

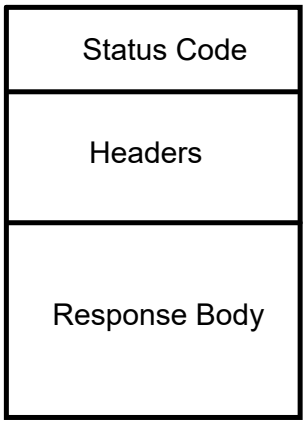
/products : GET : fetch all records

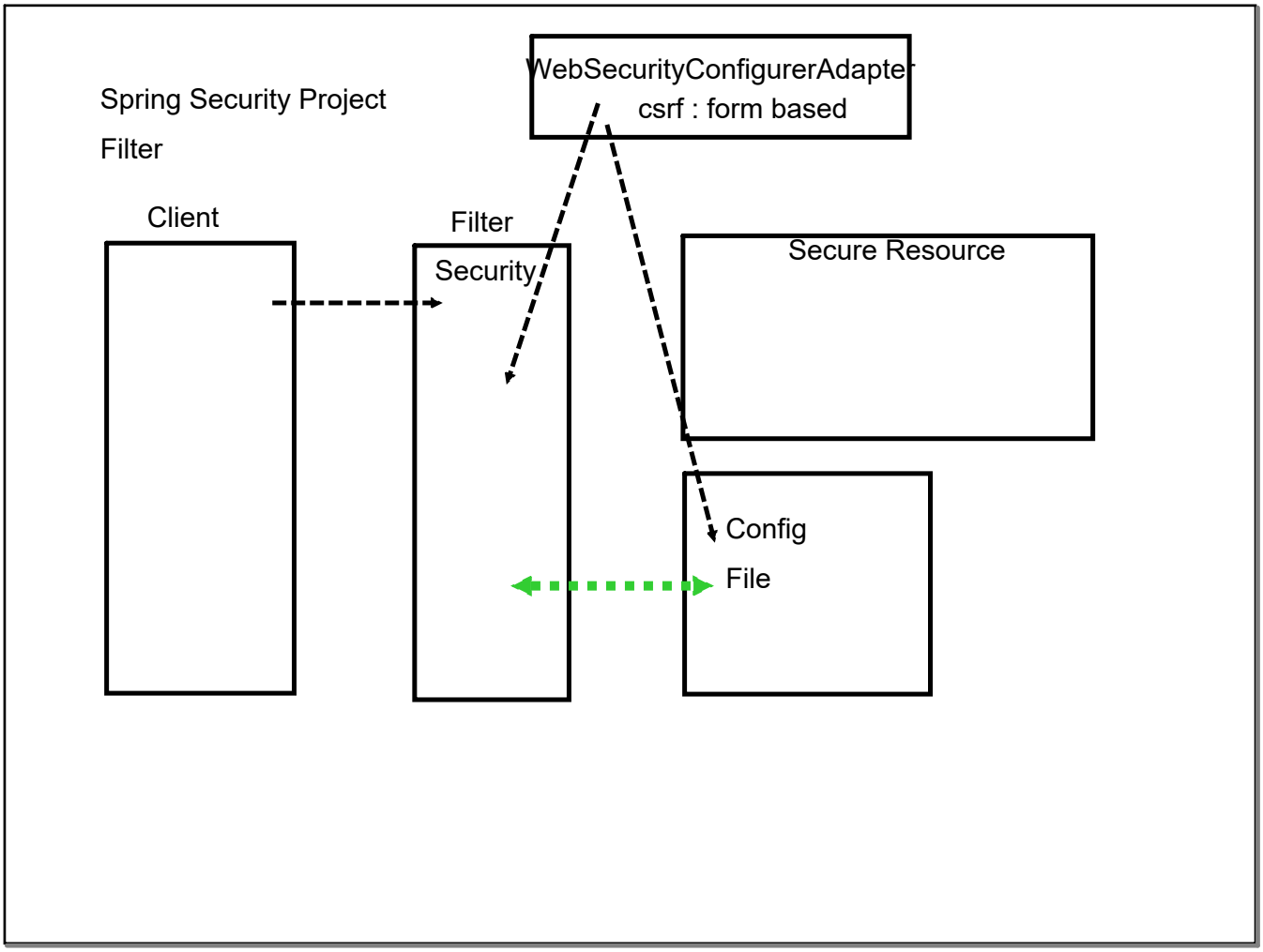
/products/id : GET

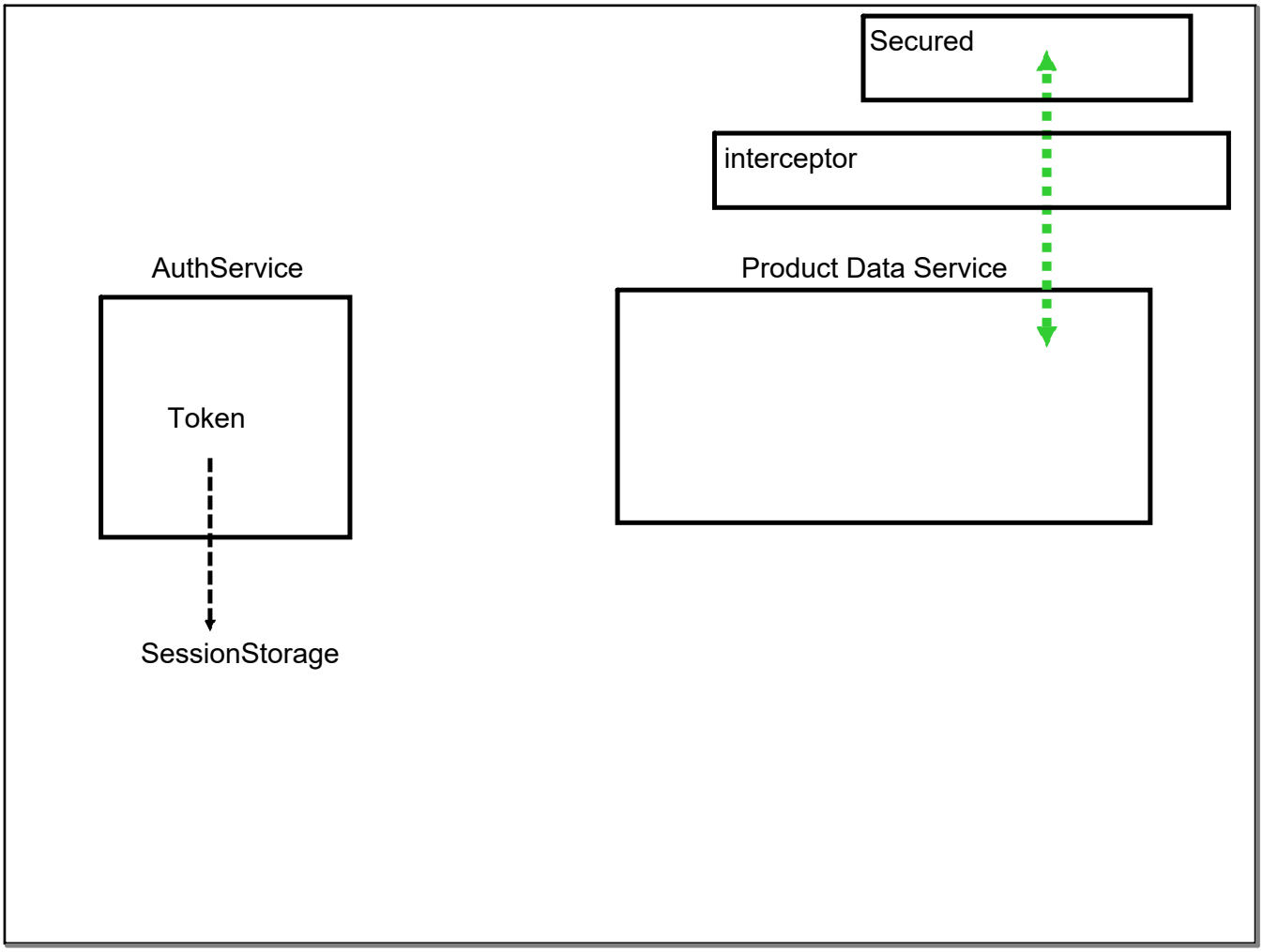
/products : POST

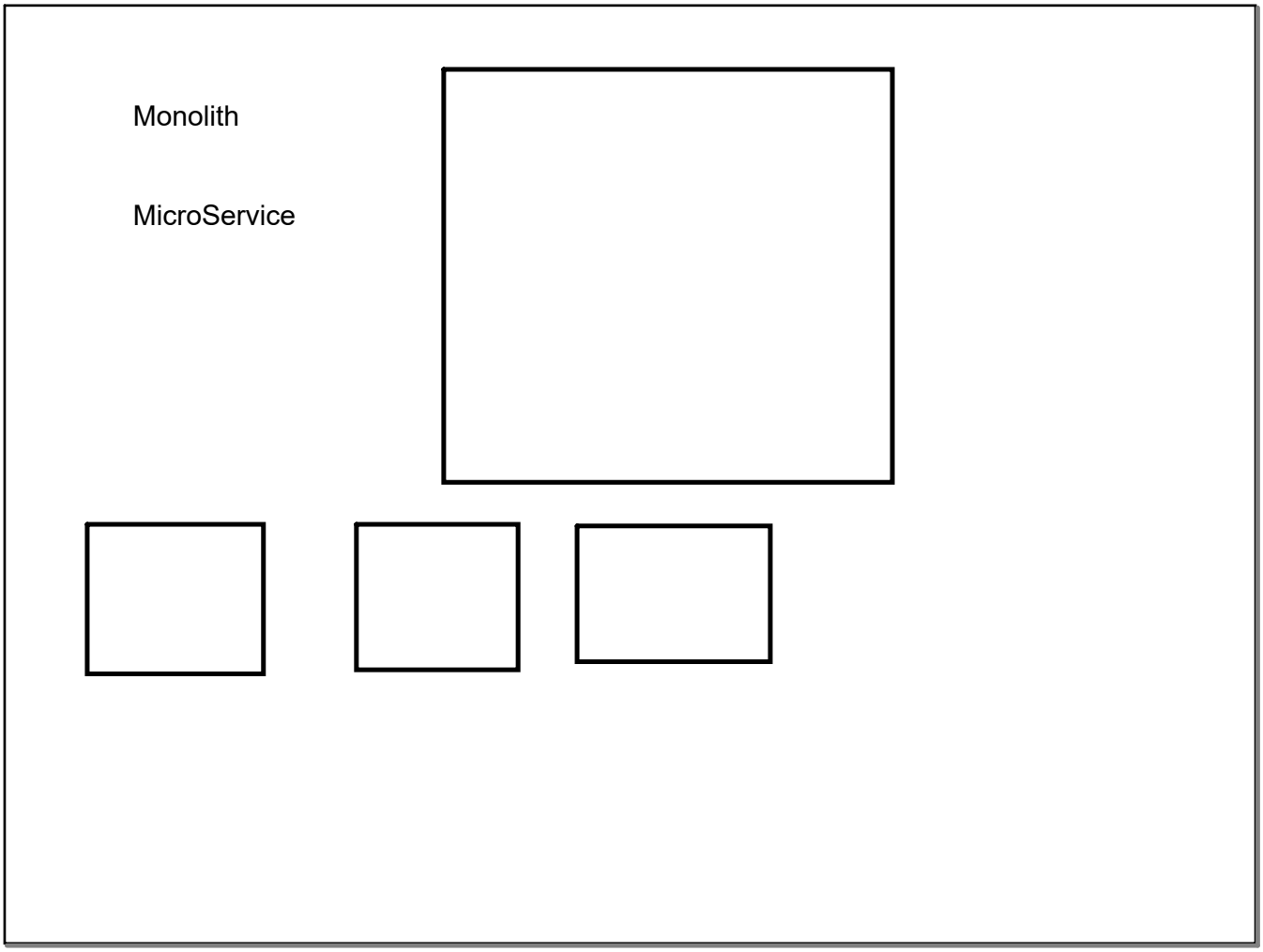
/products : PUT

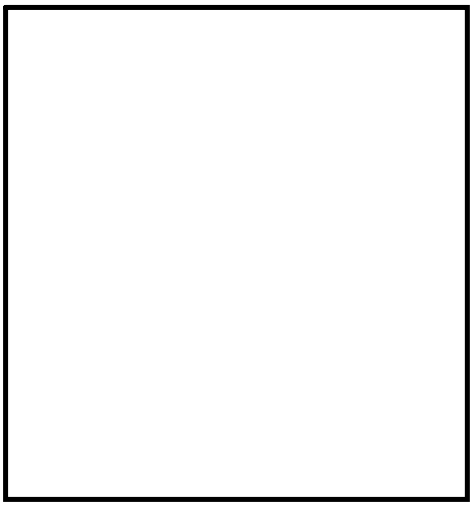
/products/id : DELETE




Spring Security Project
Filter
WebSecurityConfigurerAdapter
csrf : form based
Client
Filter
Security
Secure Resource
Config
File


Secured
interceptor
AuthService
Product Data Service
Token
SessionStorage


Monolith
MicroService



High possibility of bugs

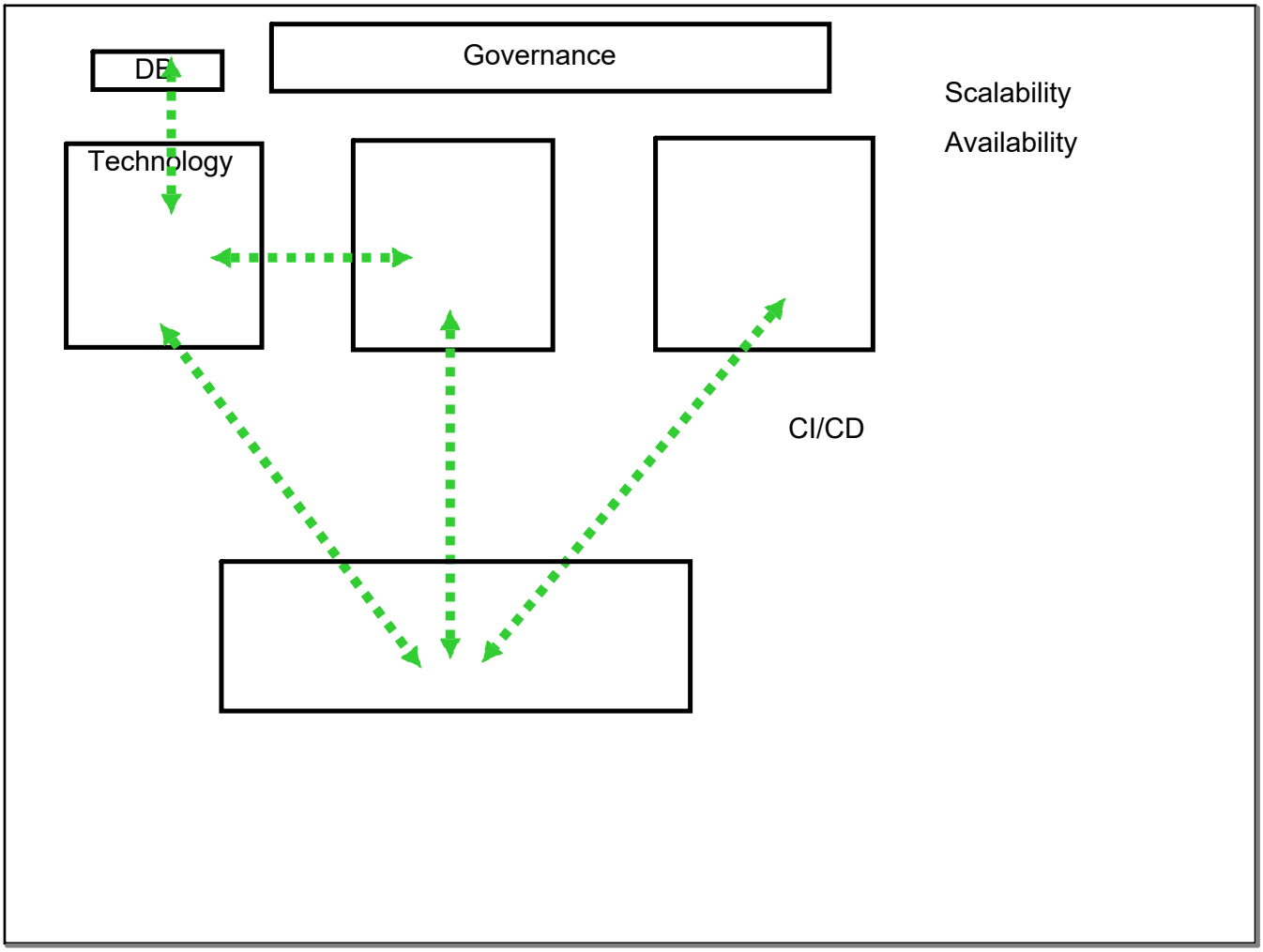
Tight coupling

Huge in size

May result into complete crash

Scaling

Technology bound


DB
Governance
Technology
Scalability
Availability
CI/CD

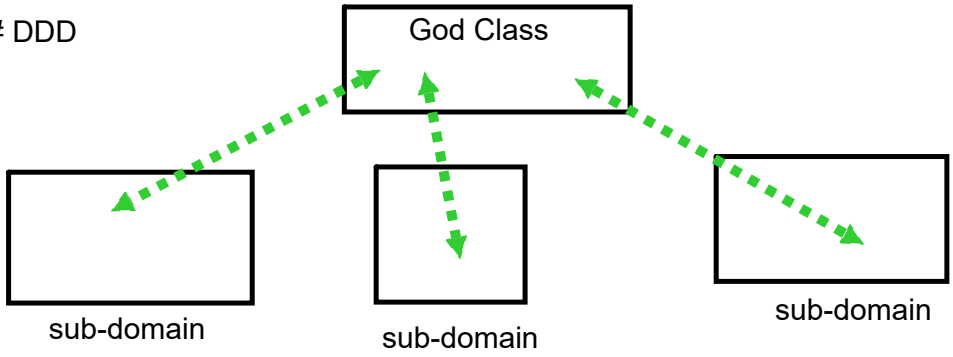
Decomposition

based business capabilities : business oriented rather than technical

"God Classes" :

Order class

DDD

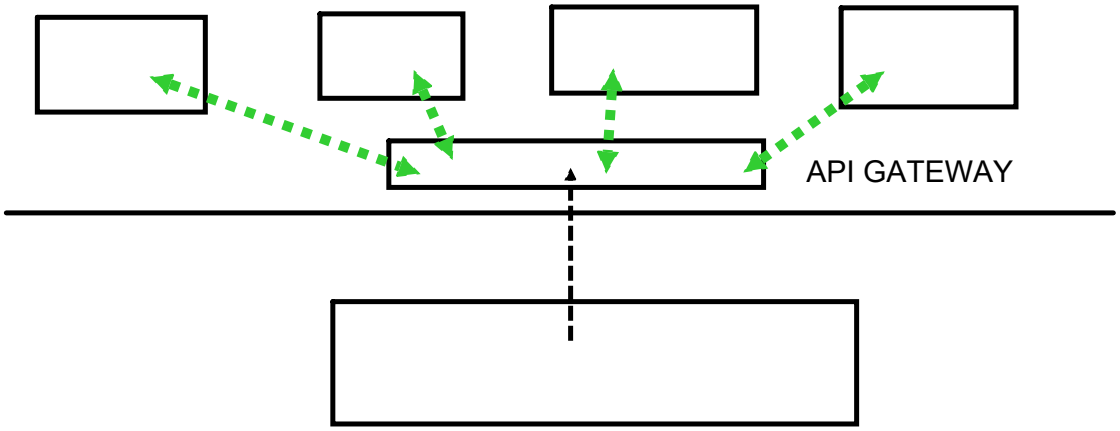


greenfield

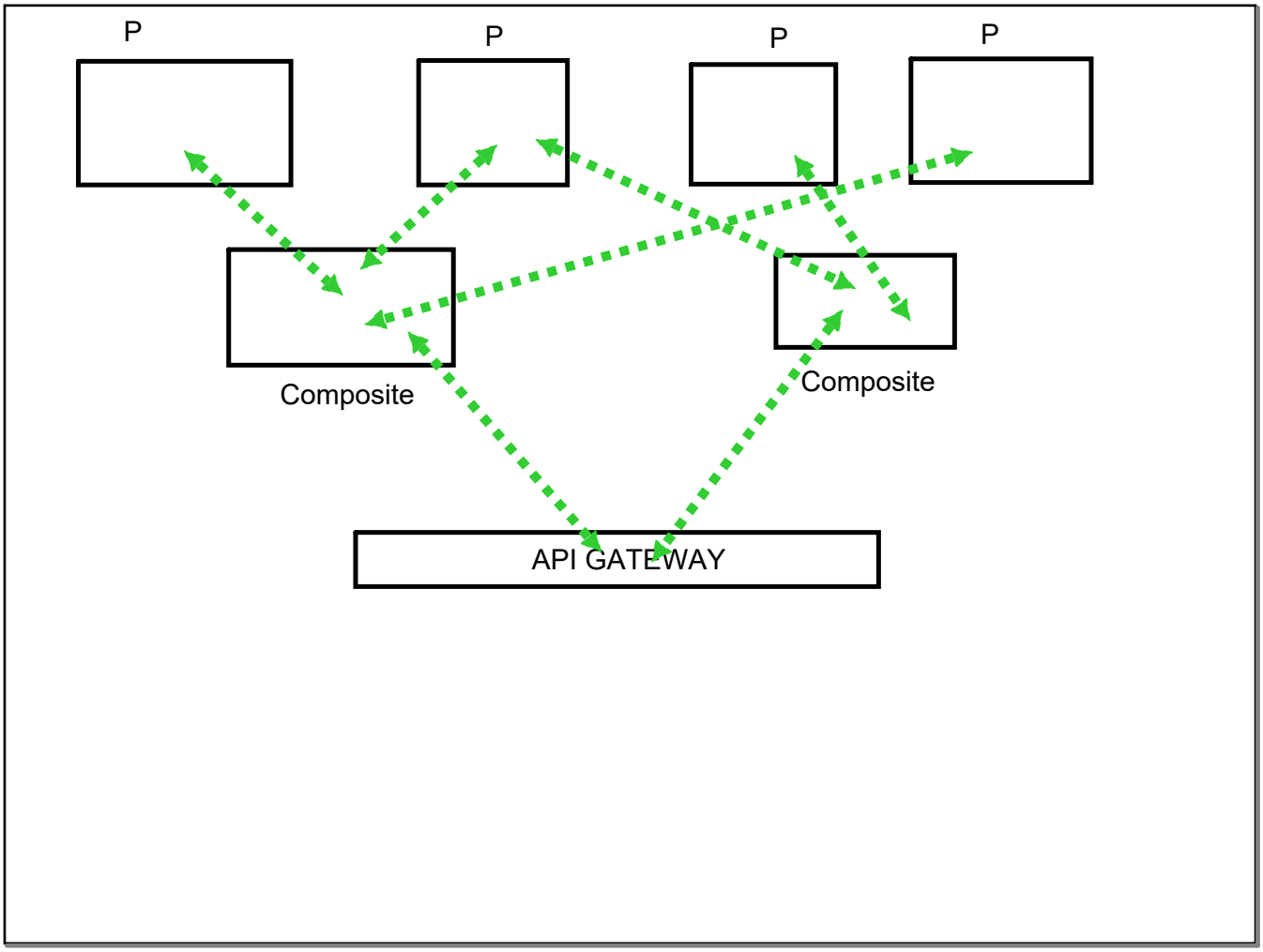
brownfield application

#Strangler pattern

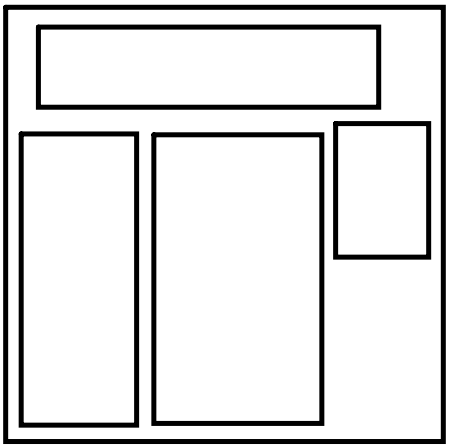
Integration Patterns



Aggregator


P
P
P
P
Composite
Composite
API GATEWAY

Client-Side UI Composition Pattern



SPA : Angular/React

Database Pattern

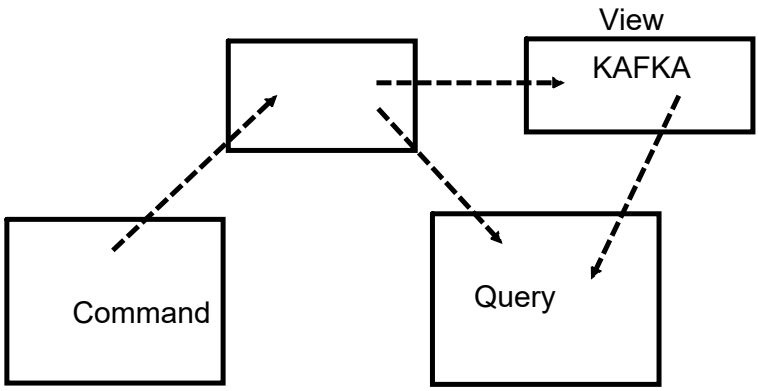
a) One DB per service

b) Shared DB per service

2-3 microservice

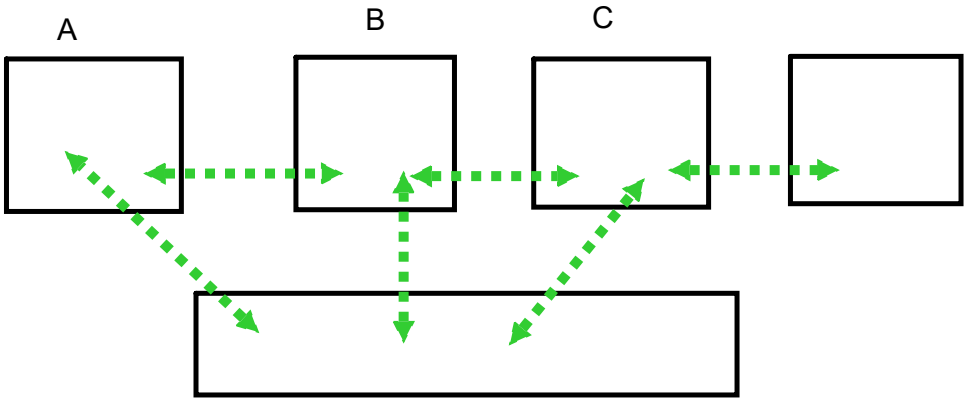
c) CQRS (Command Query Responsibility Segregation)

Event Sourcing



d) SAGA Pattern

1. Choreography



2. Orchestration

Observability Patterns

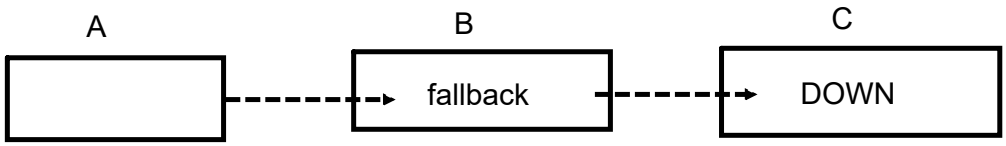
a) Log Aggregation

b) Performance Metrics

c) Distributed Tracing

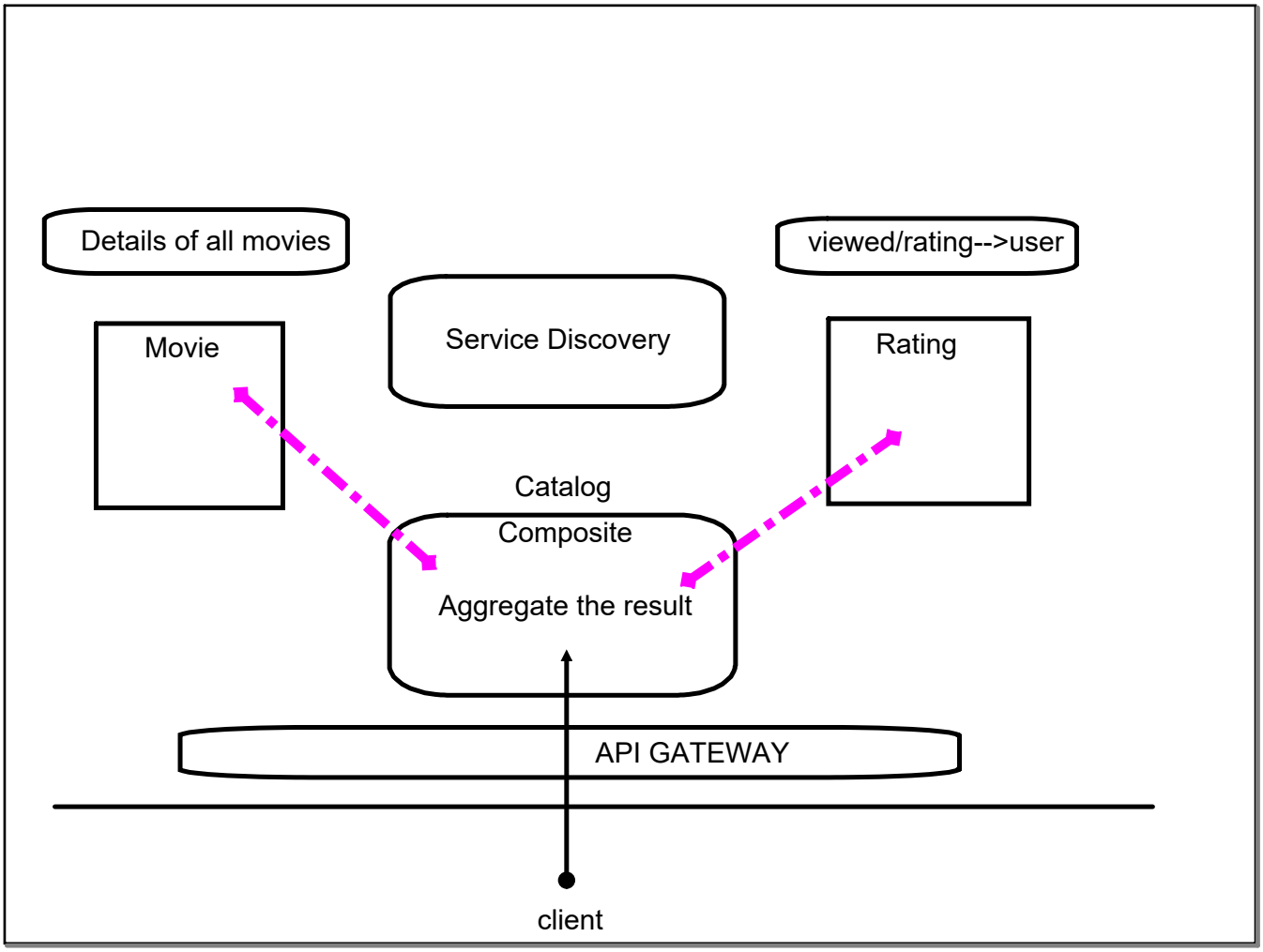
d) Health Check

Cross-Cutting



threshold : number/timeout

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


Details of all movies
viewed/rating-->user
Service Discovery
Movie
Rating
Catalog
Composite
Aggregate the result
API GATEWAY
client

/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