Micorservices

Microservice

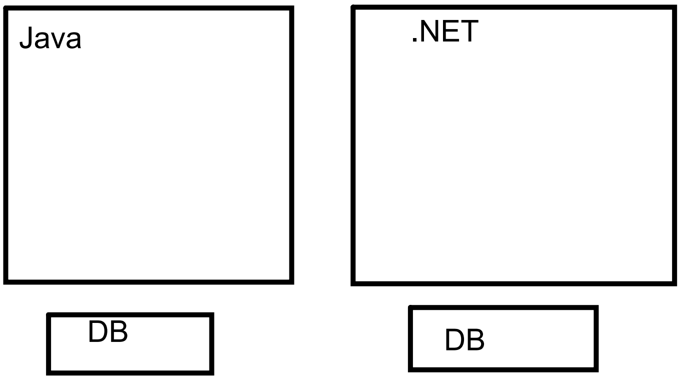
# Modularity

# Use technology

# Single MS drop

# Dynamic Scale up/Scale Down

Independent/Autonomous



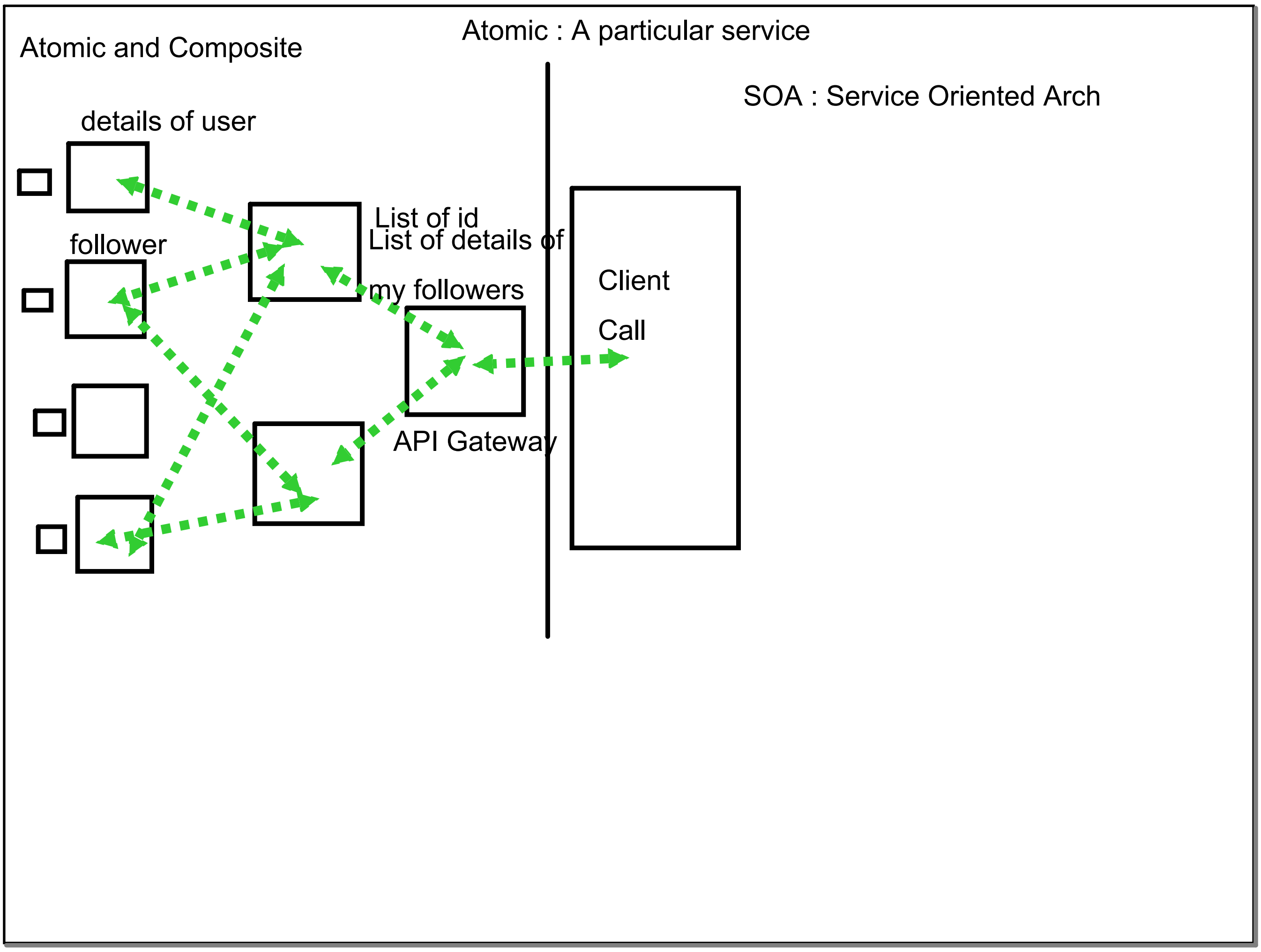
Java

.NET

DB

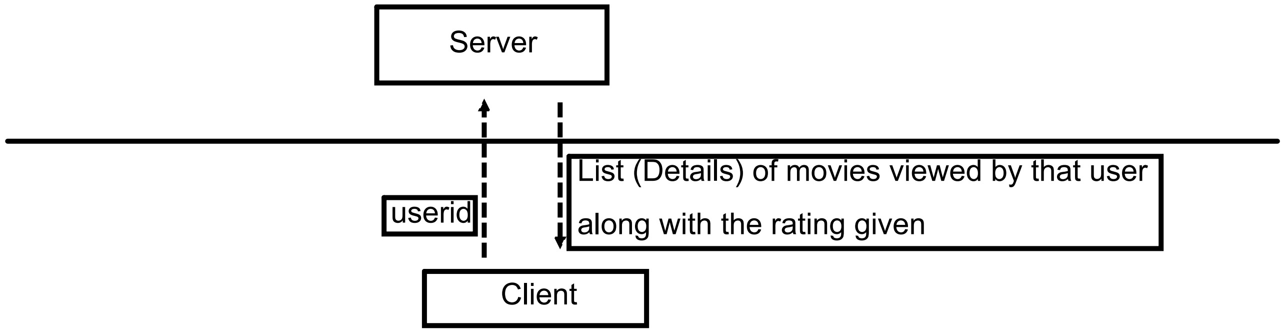
DB

Micorservices



Use Case:

# Movie Streaming portal



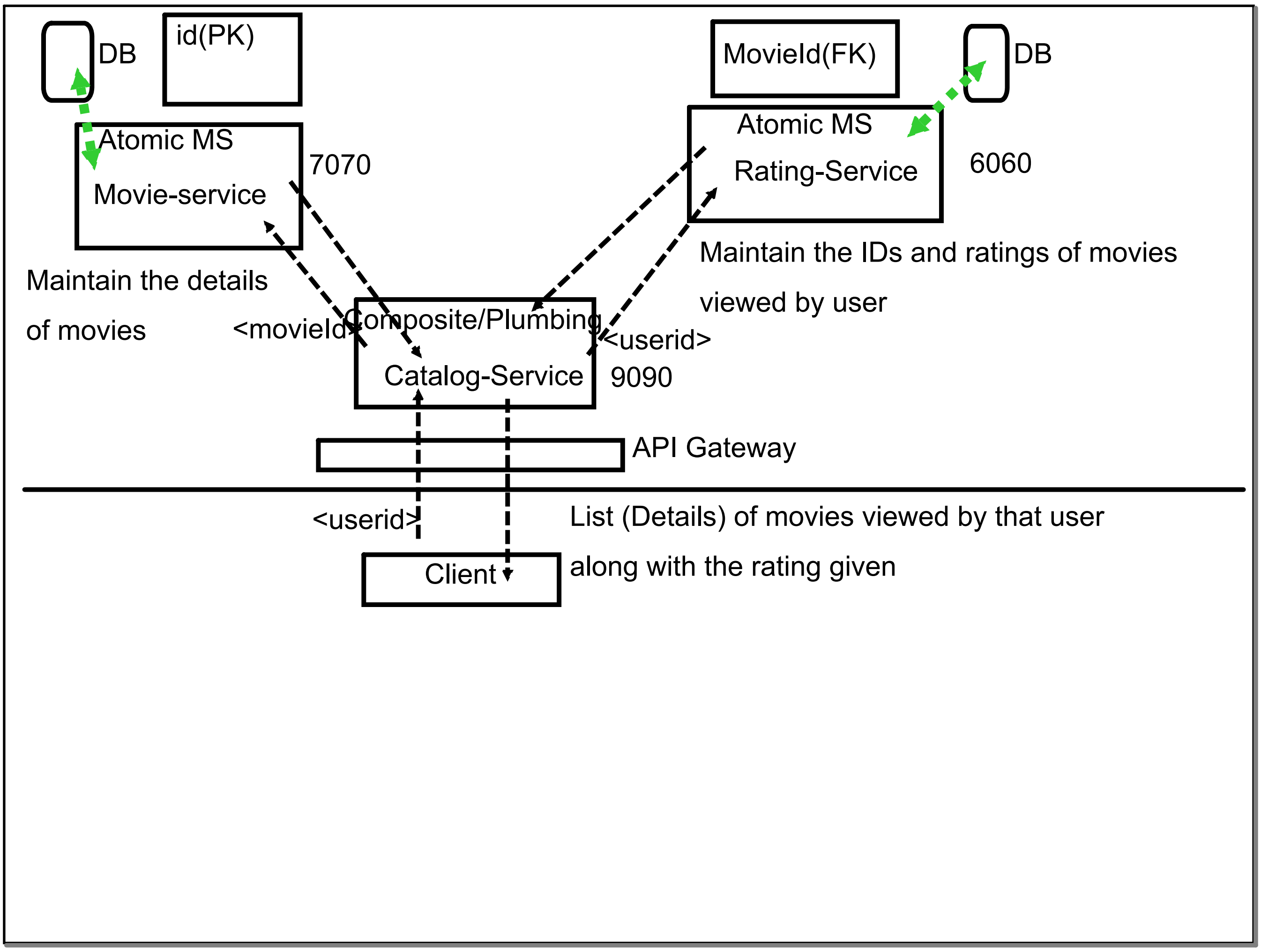
Server

userid

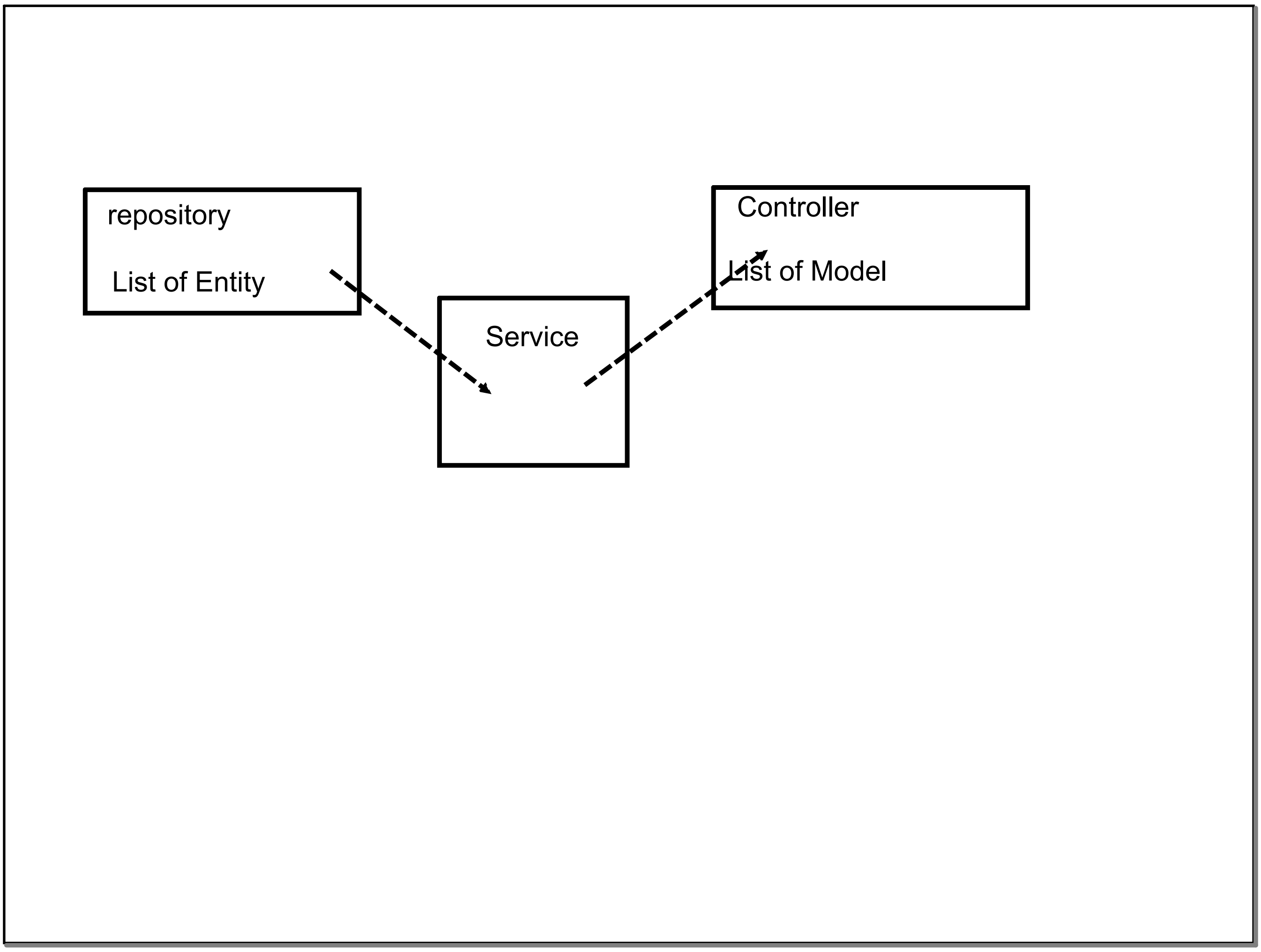
List (Details) of movies viewed by that user along with the rating given

Client

Micorservices



Micorservices



repository

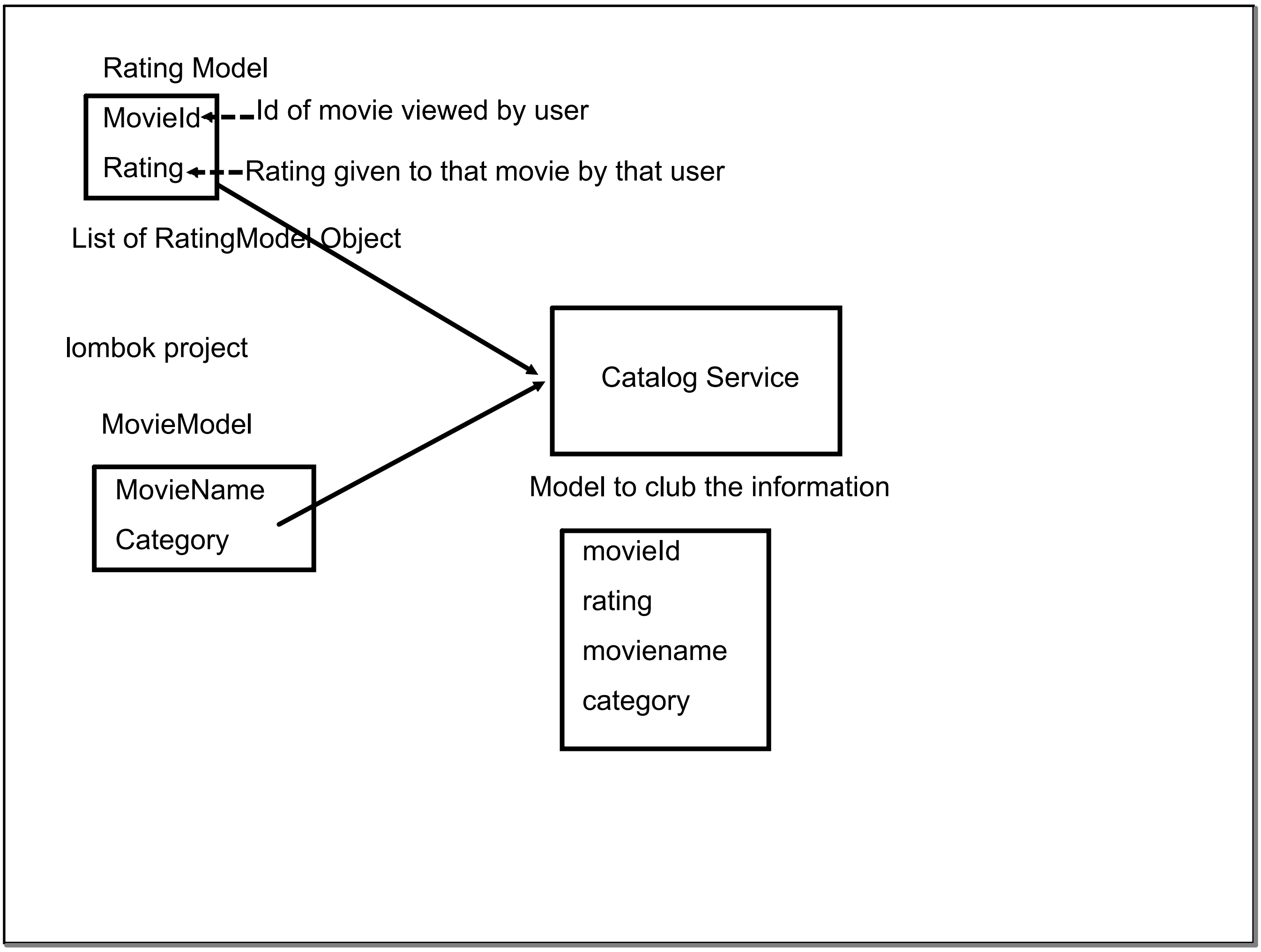
List of Entity

Service

Controller

List of Model

Micorservices



Rating Model

MovieId

Rating

Id of movie viewed by user

Rating given to that movie by that user

List of RatingModel Object

lombok project

MovieModel

MovieName

Category

Catalog Service

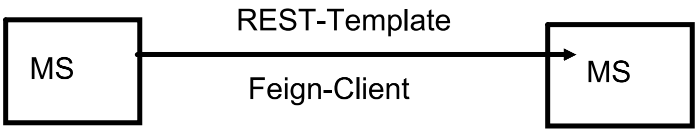
Model to club the information

movieId

rating

moviename

category

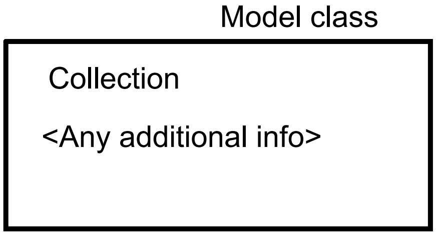


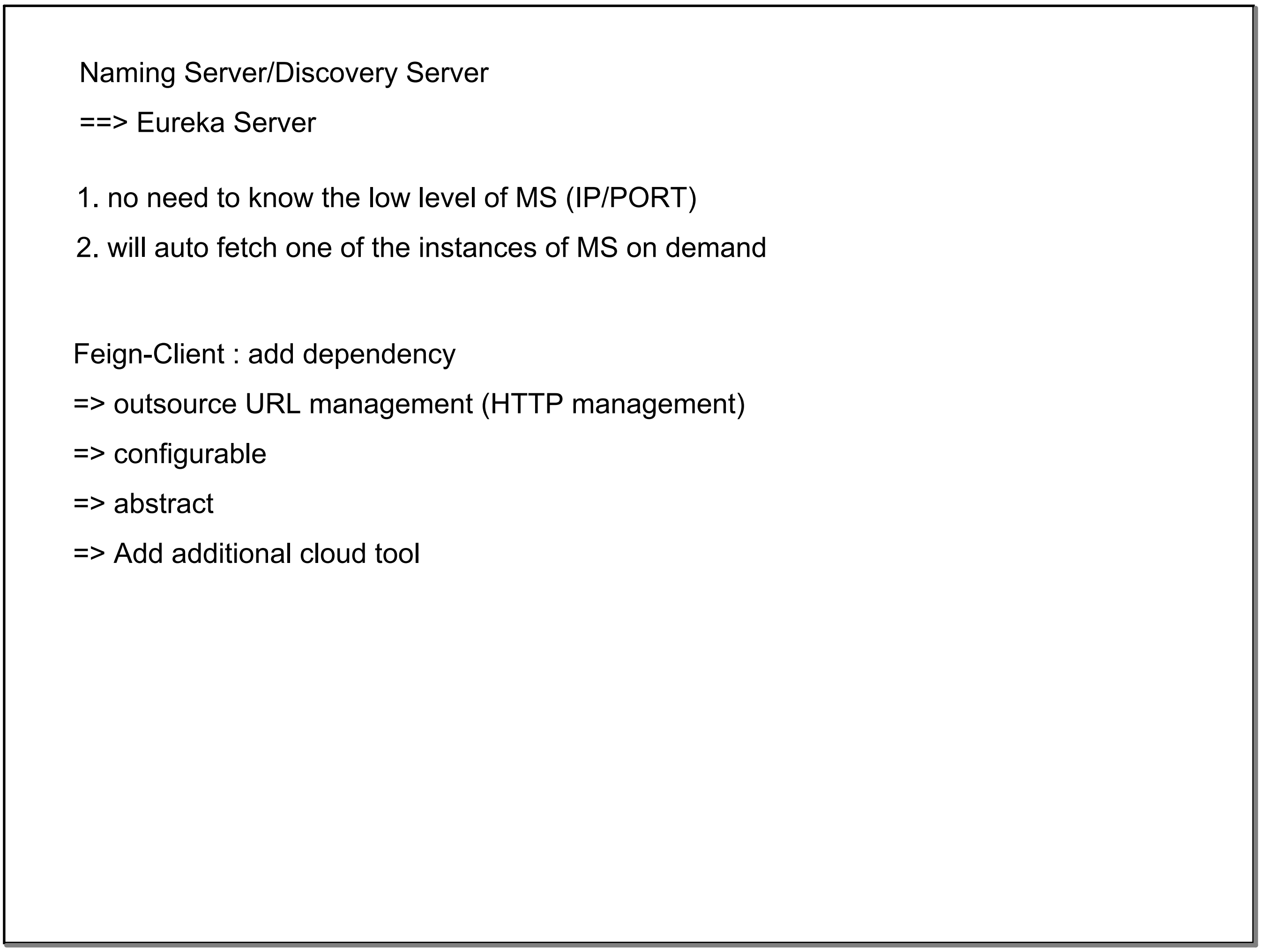
Best Practice:

1. Should never return a list/collection directly

   # if list is huge, it creates problem in streaming back to calling service

   # it will stop us to update/add any additional info to associate in future





Naming Server/Discovery Server

==> Eureka Server

1. no need to know the low level of MS (IP/PORT)
2. will auto fetch one of the instances of MS on demand

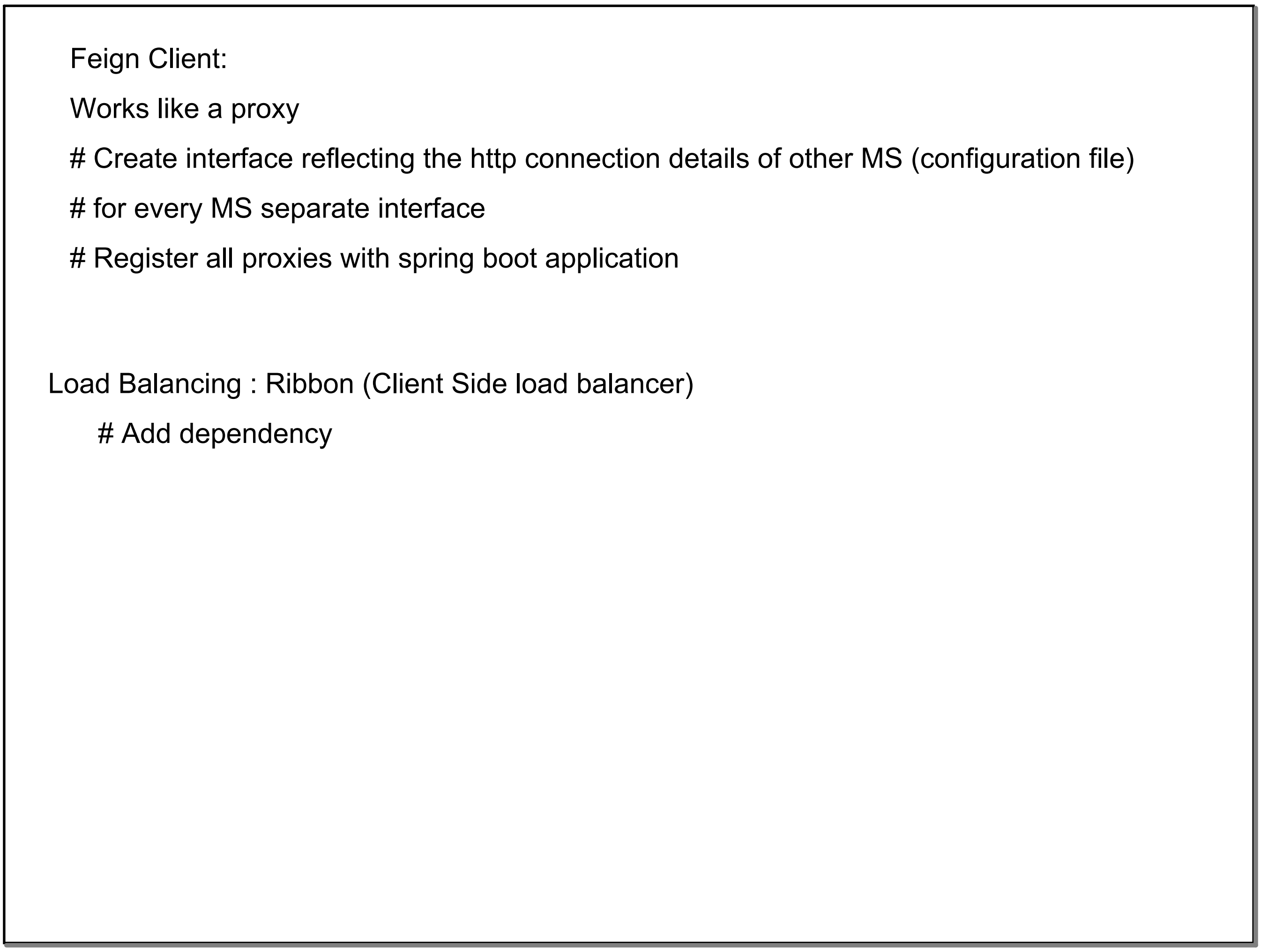
Feign-Client : add dependency

=> outsource URL management (HTTP management)

=> configurable

=> abstract

=> Add additional cloud tool



Feign Client:

Works like a proxy

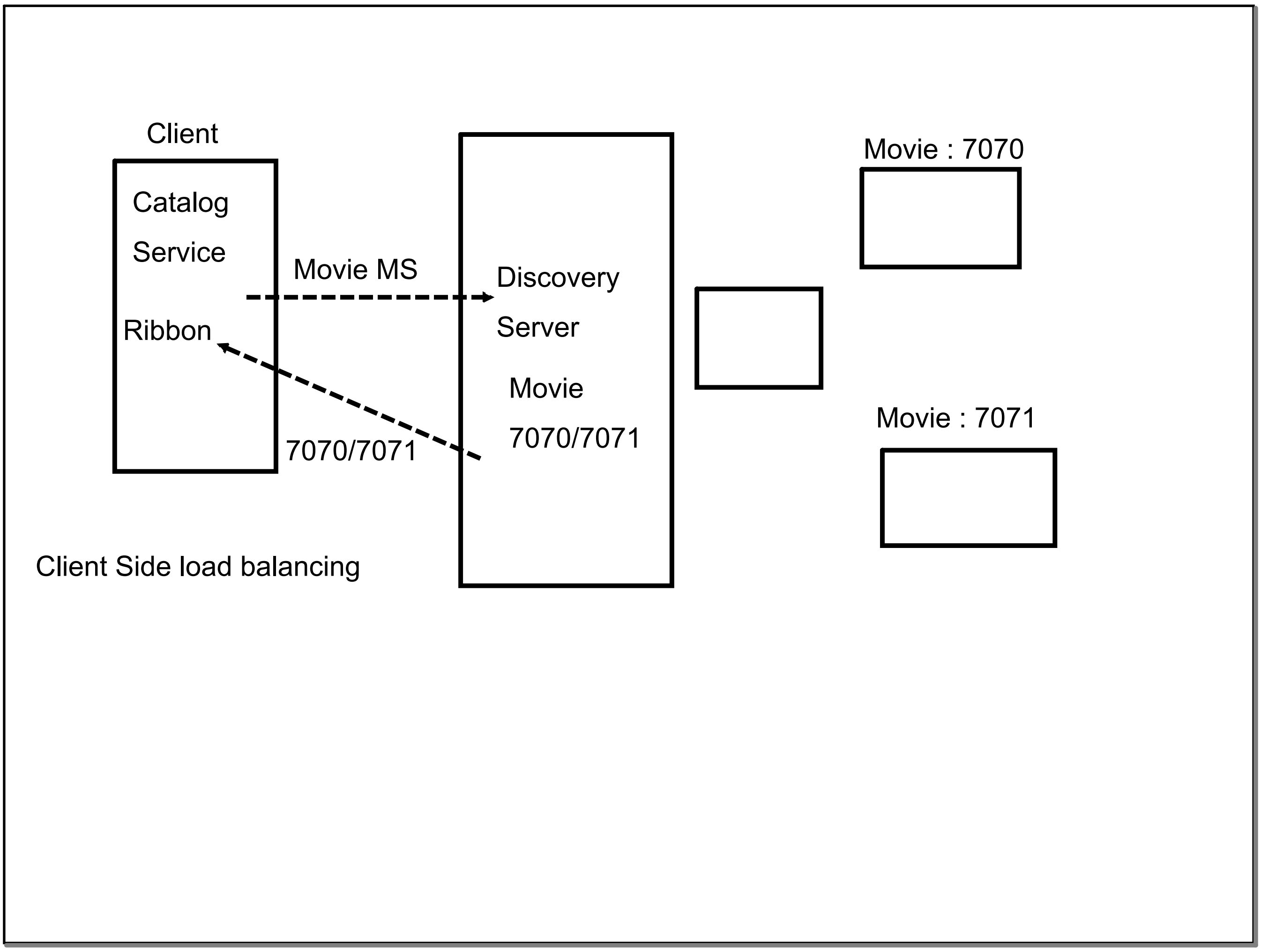
# Create interface reflecting the http connection details of other MS (configuration file)

# for every MS separate interface

# Register all proxies with spring boot application

Load Balancing : Ribbon (Client Side load balancer)

# Add dependency



Client

Catalog Service

Ribbon

Movie MS

7070/7071

Discovery Server

Movie 7070/7071

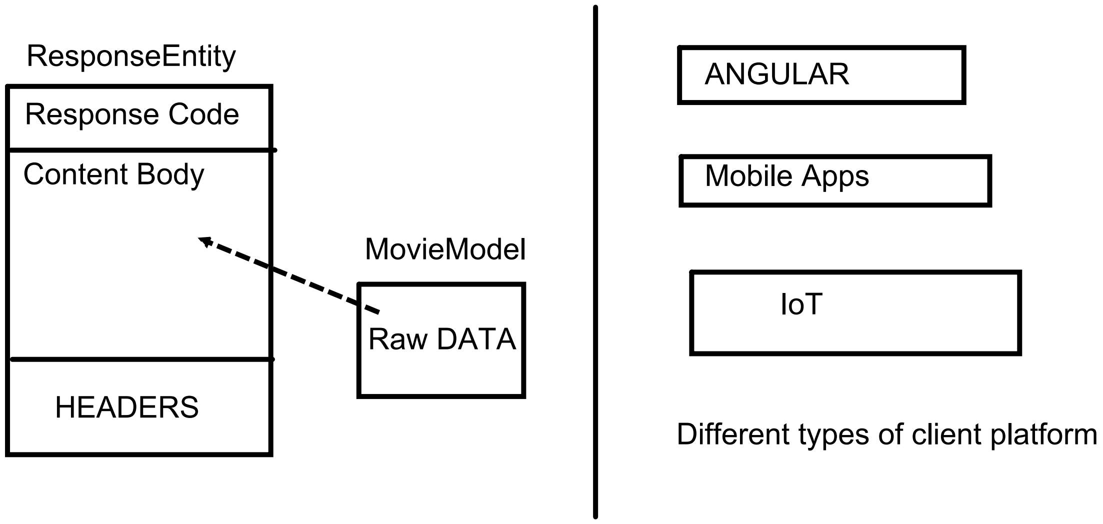
Movie : 7070

Movie : 7071

Client Side load balancing

Best Practices

: instead of returning raw data : wrap it in ResponseEntity



ResponseEntity

Response Code

Content Body

HEADERS

MovieModel

Raw DATA

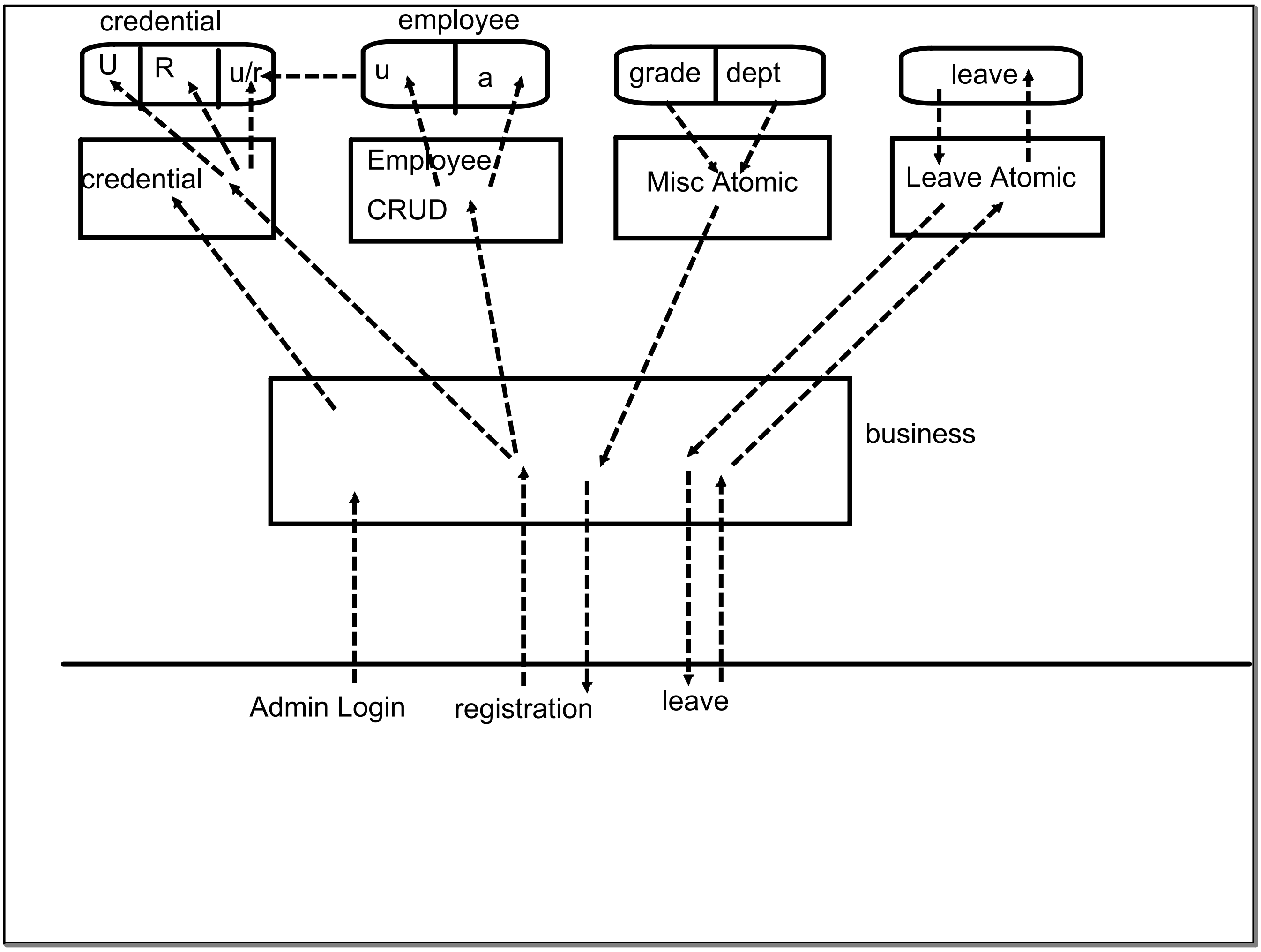
ANGULAR

Mobile Apps

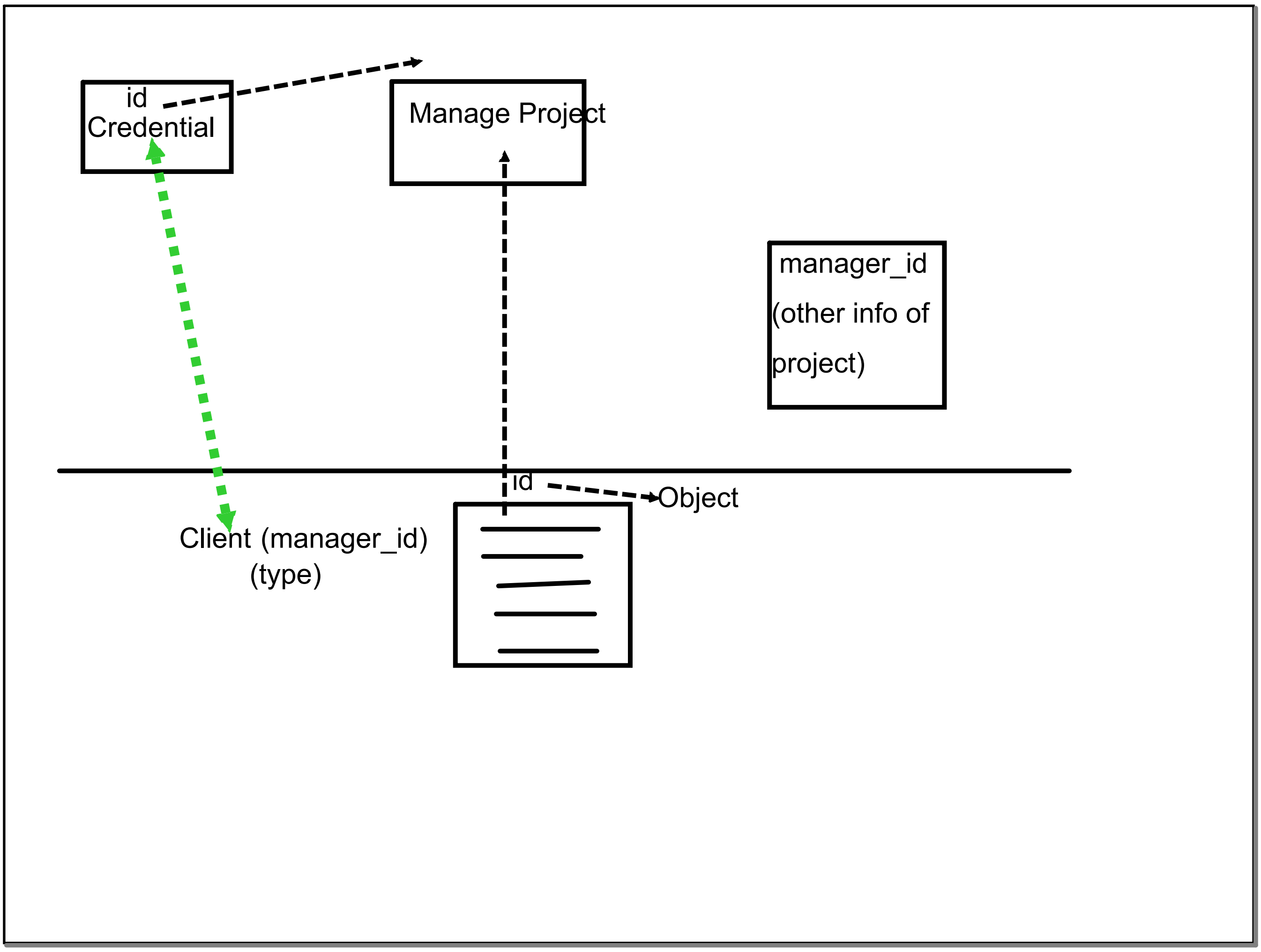
IoT

Different types of client platform

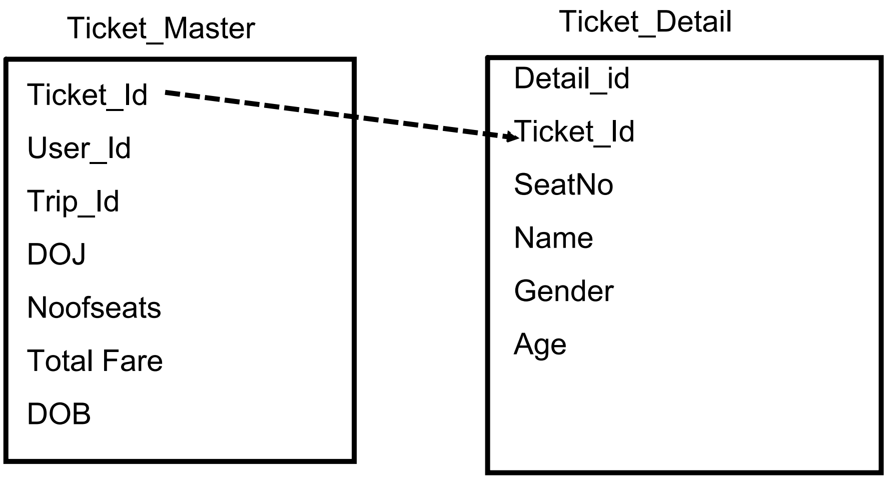
Micorservices



Micorservices



Micorservices

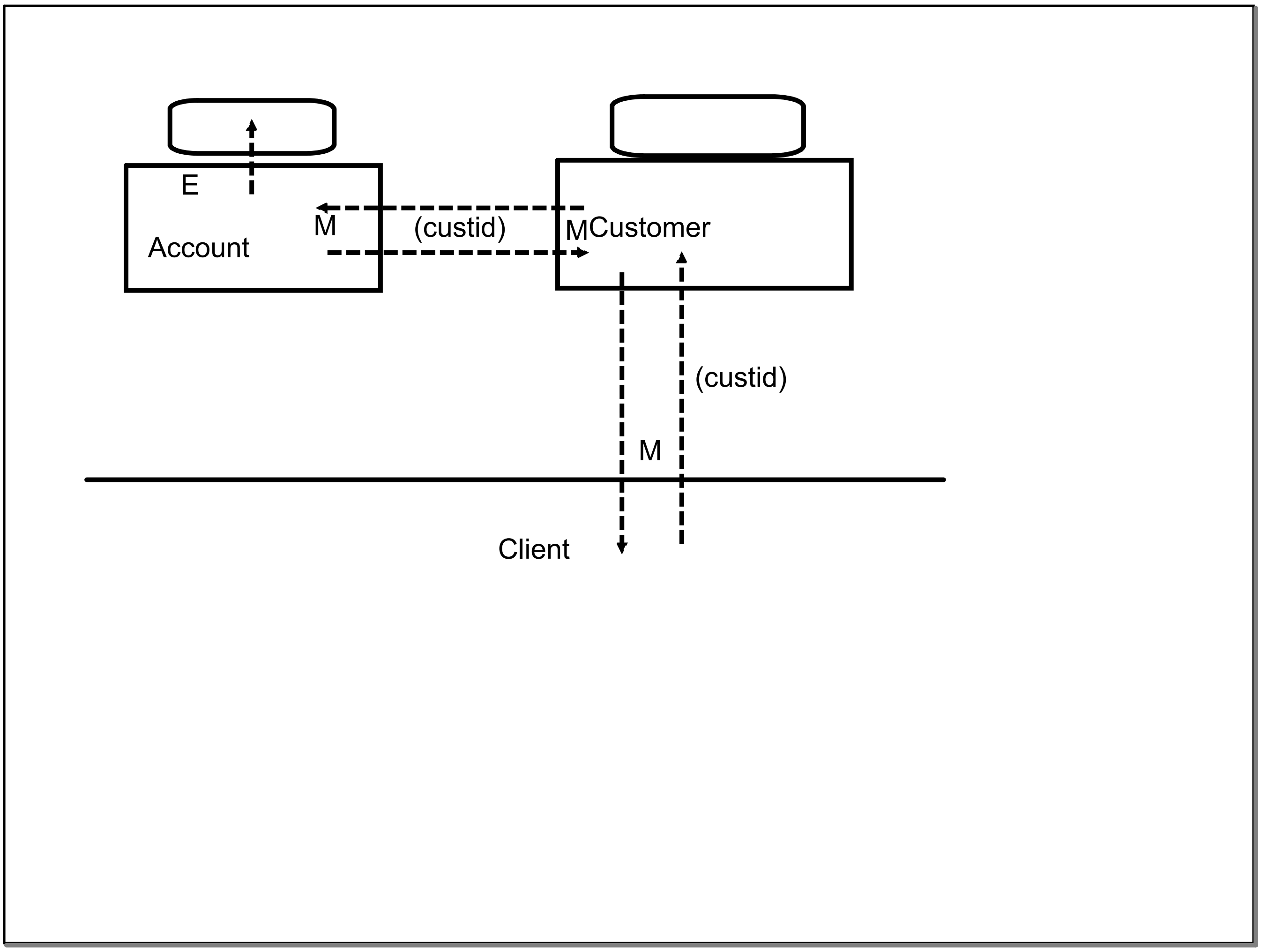


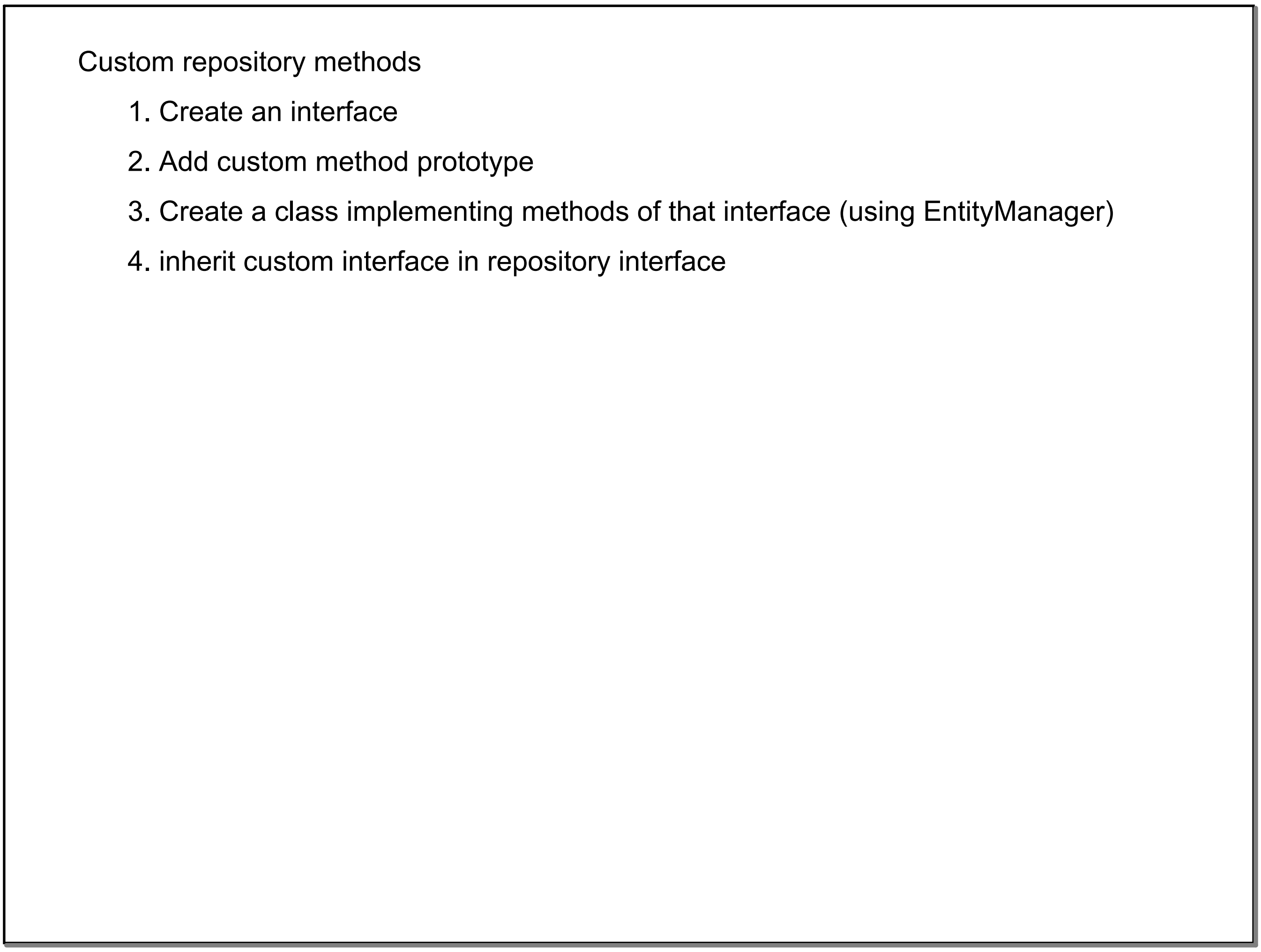
Ticket_Master

Ticket_Id
User_Id
Trip_Id
DOJ
Noofseats
Total Fare
DOB

Ticket_Detail

Detail_id
Ticket_Id
SeatNo
Name
Gender
Age

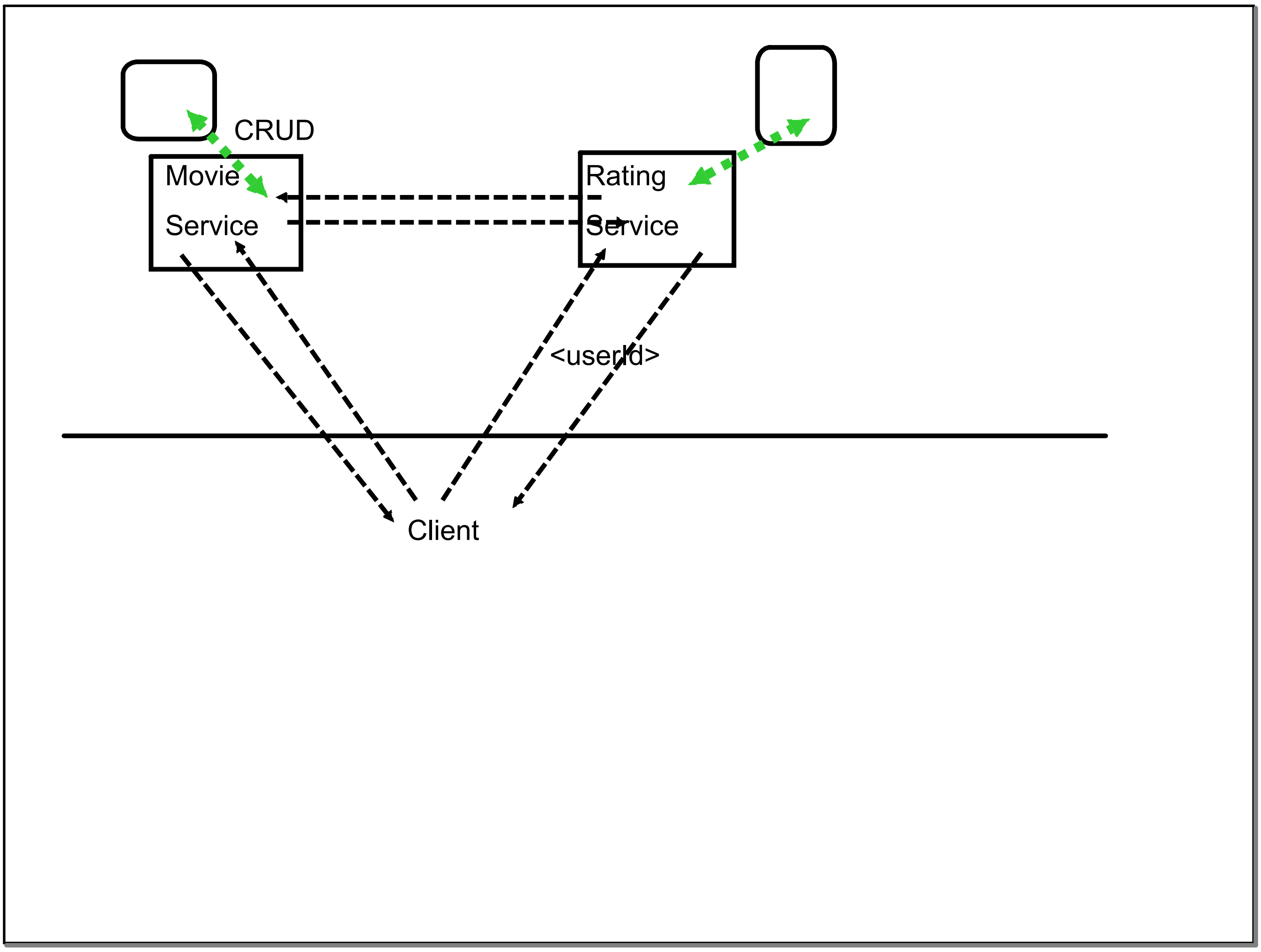

E
M
Account
(custid)
M
Customer
(custid)
M
Client

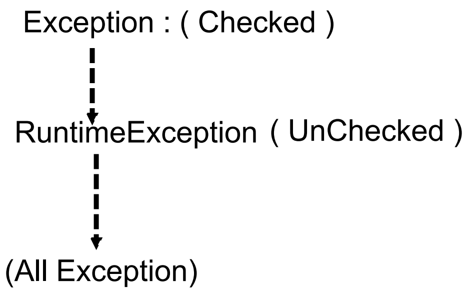


Custom repository methods

1. Create an interface
2. Add custom method prototype
3. Create a class implementing methods of that interface (using EntityManager)
4. inherit custom interface in repository interface

Micorservices

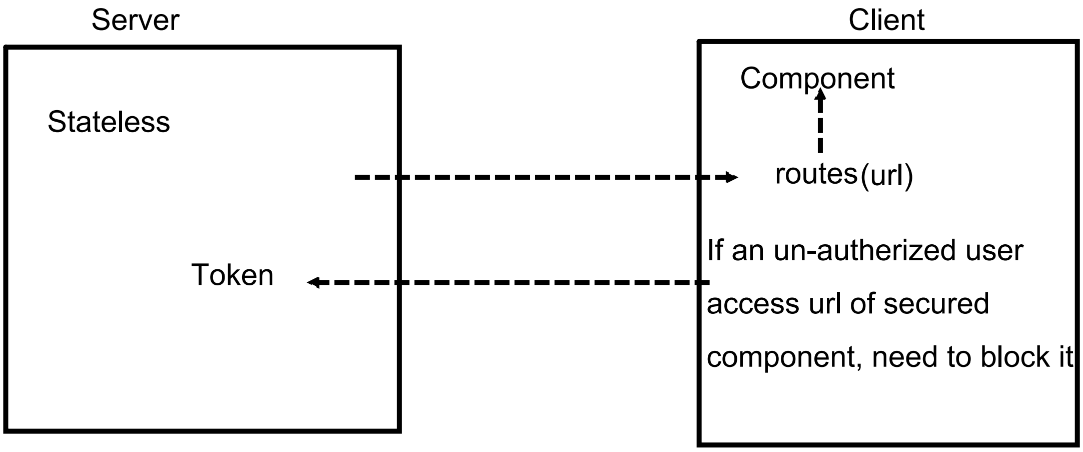




Exception : (Checked)

RuntimeException (UnChecked)

(All Exception)



Server

Stateless

Token

Client

Component

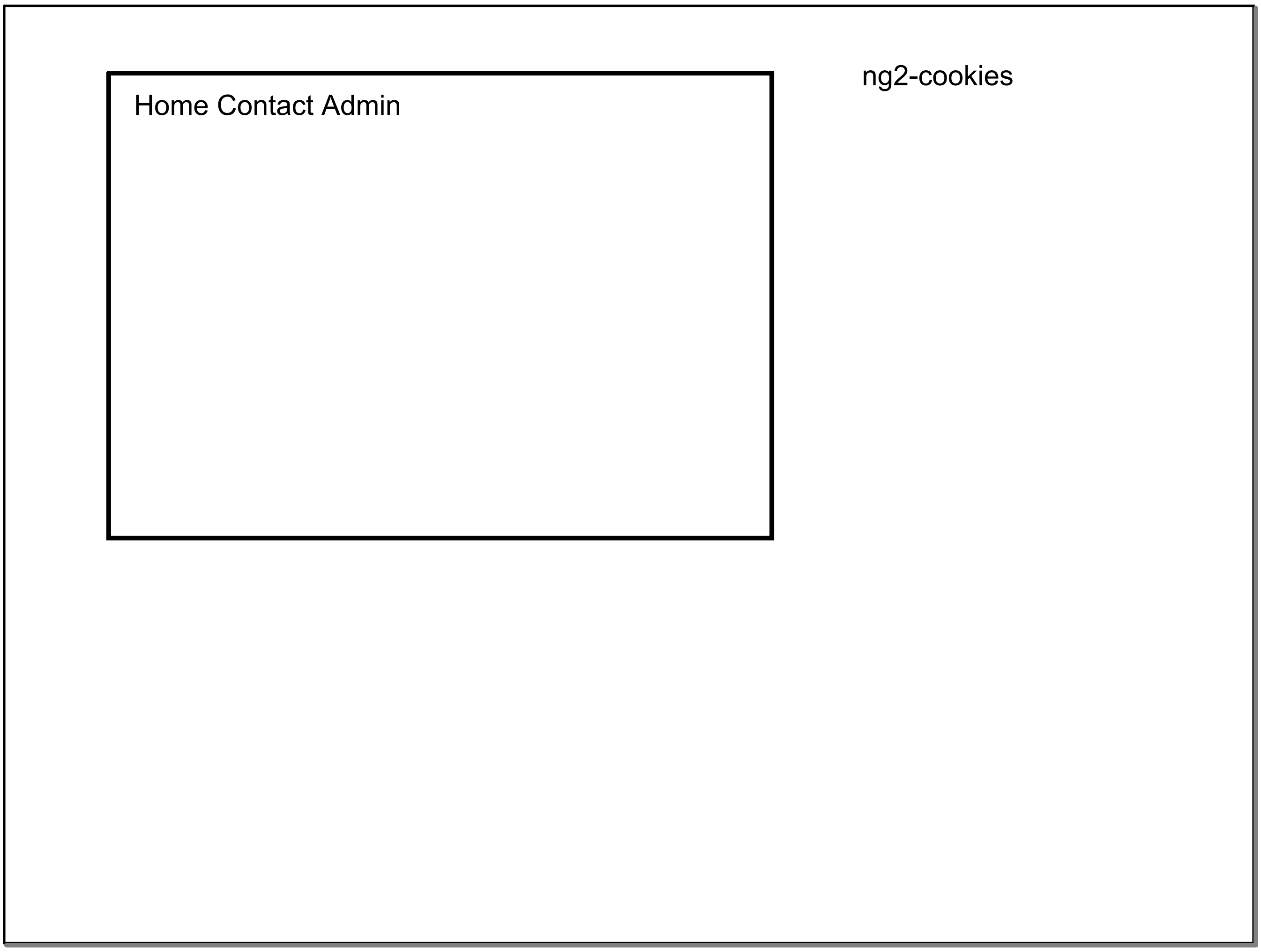
routes(url)

If an un-autherized user access url of secured component, need to block it

Spring Security : Basic Auth (Token : no expiration time)

JWT
OAuth | Time bounded token

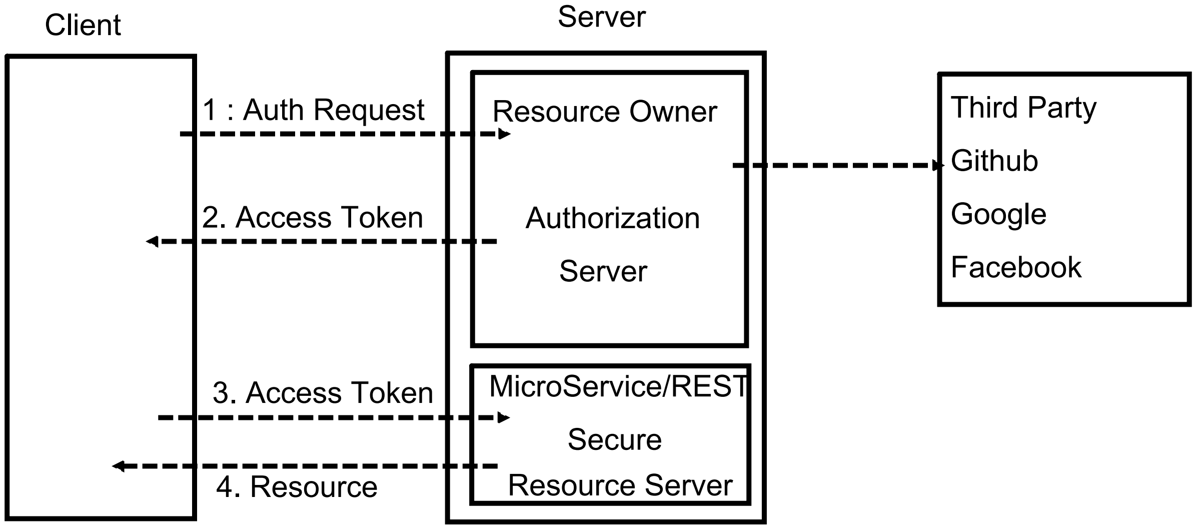
Micorservices

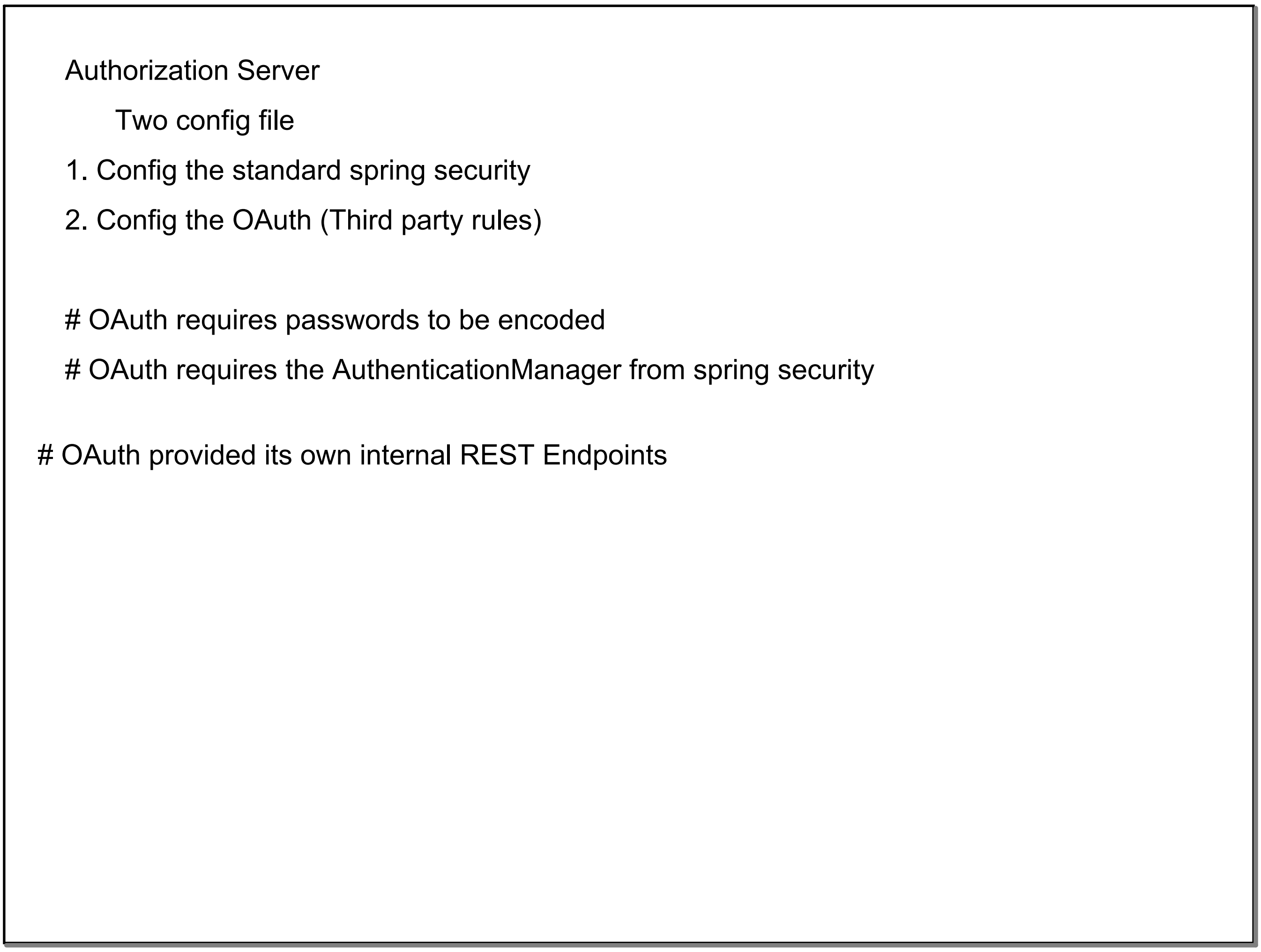


Home Contact Admin

ng2-cookies

OAuth Model of security





Authorization Server

Two config file

1. Config the standard spring security
2. Config the OAuth (Third party rules)

# OAuth requires passwords to be encoded

# OAuth requires the AuthenticationManager from spring security

# OAuth provided its own internal REST Endpoints

OAuth Credential (Basic Authentication)

Application Credentials (form data)

Resource Server (Microservice/REST Service)

1. Resource Server Configuration