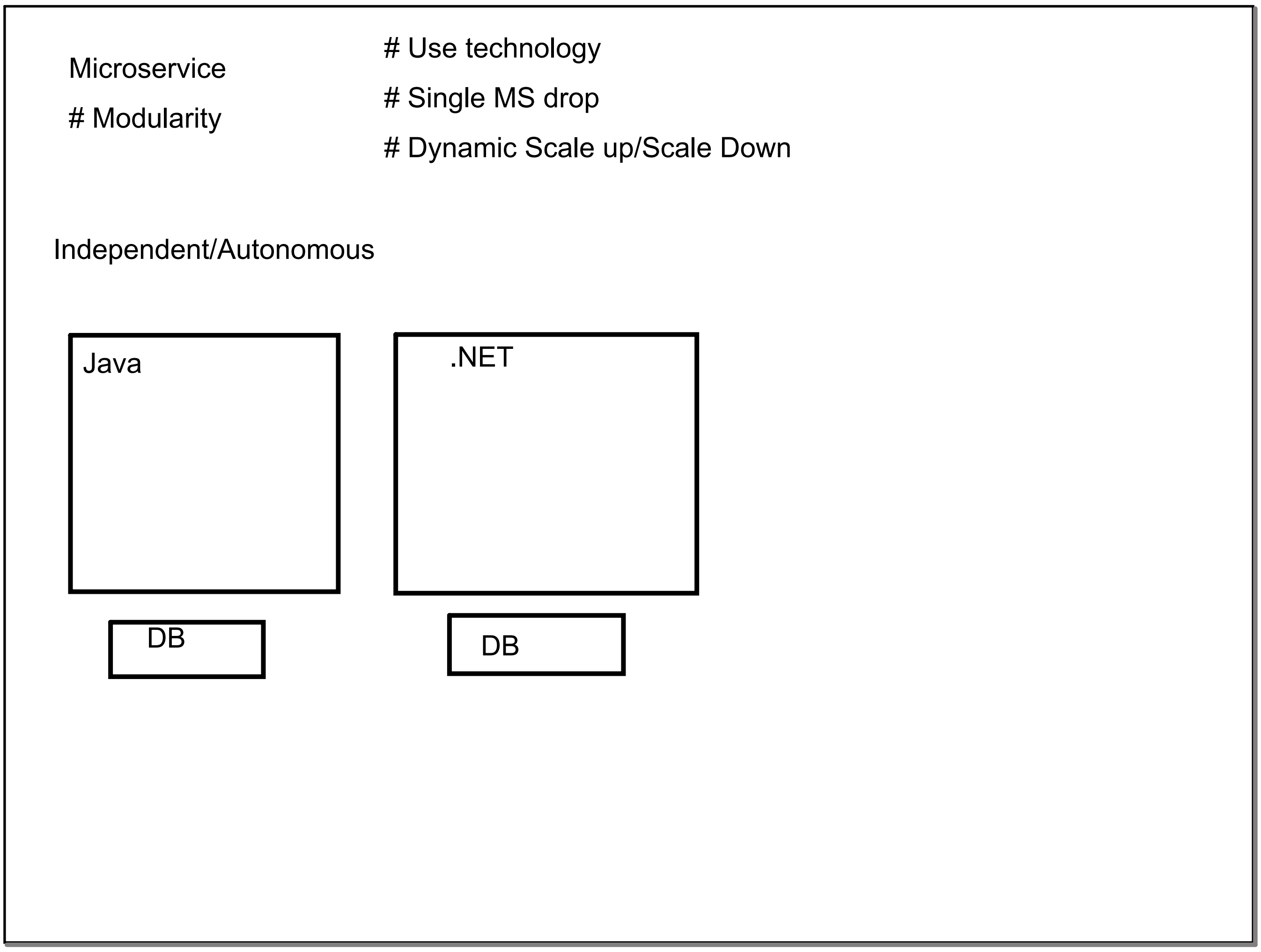
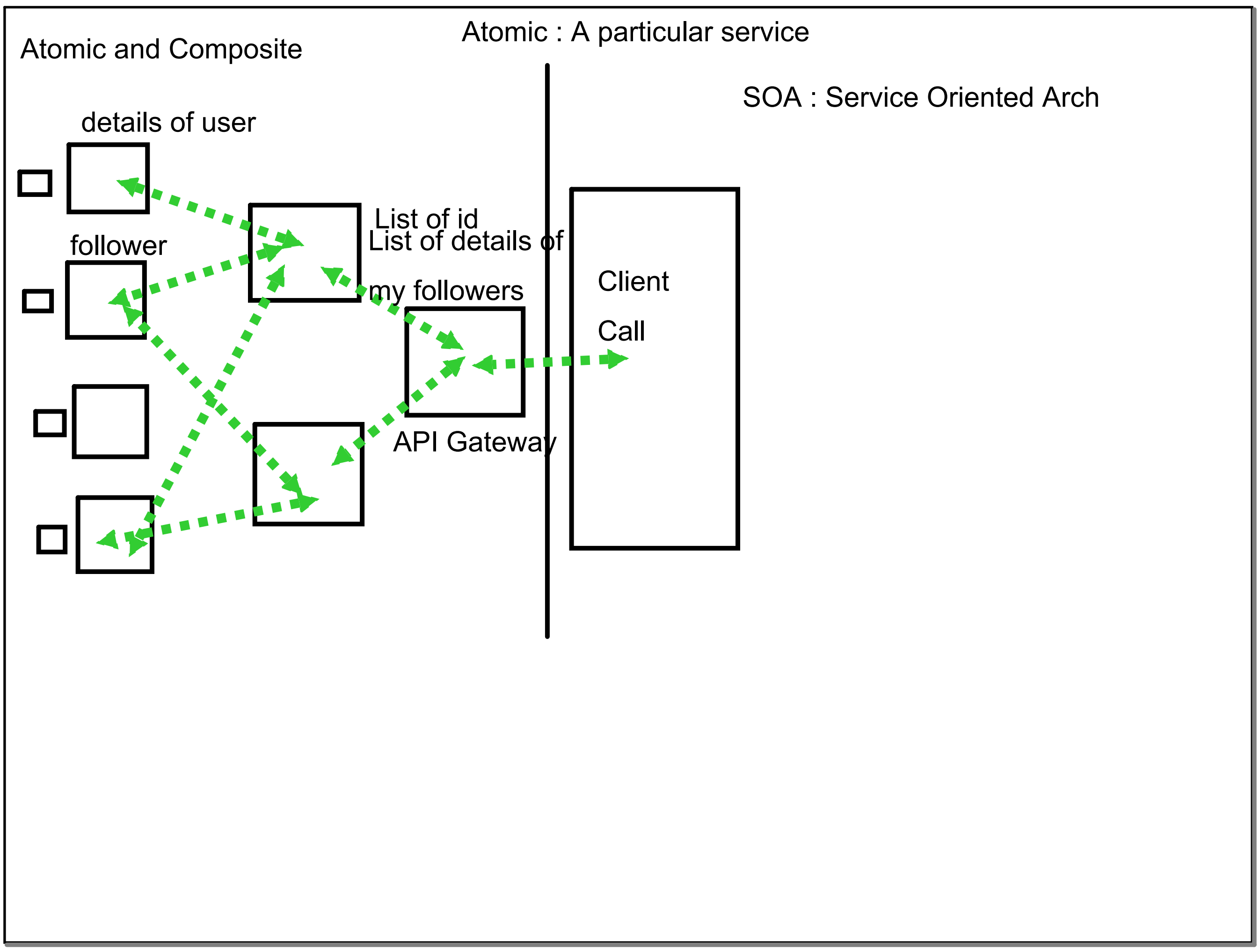
Micorservices

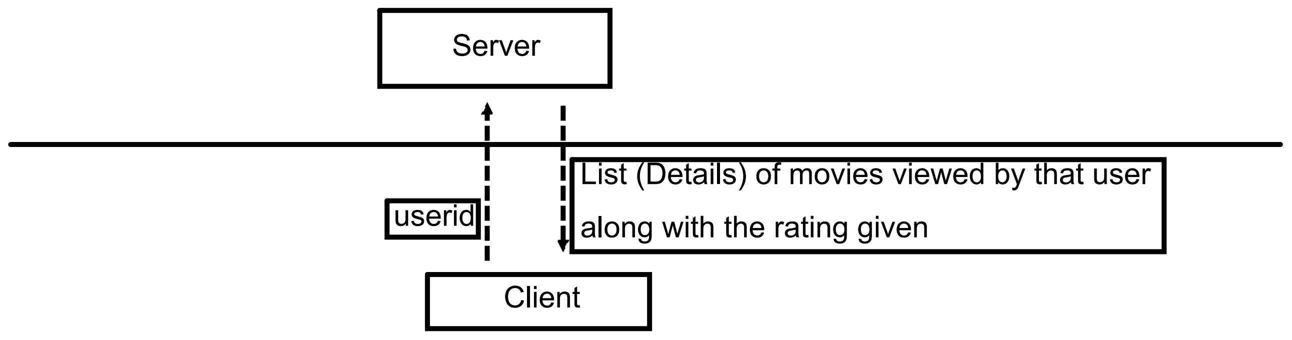


Micorservices

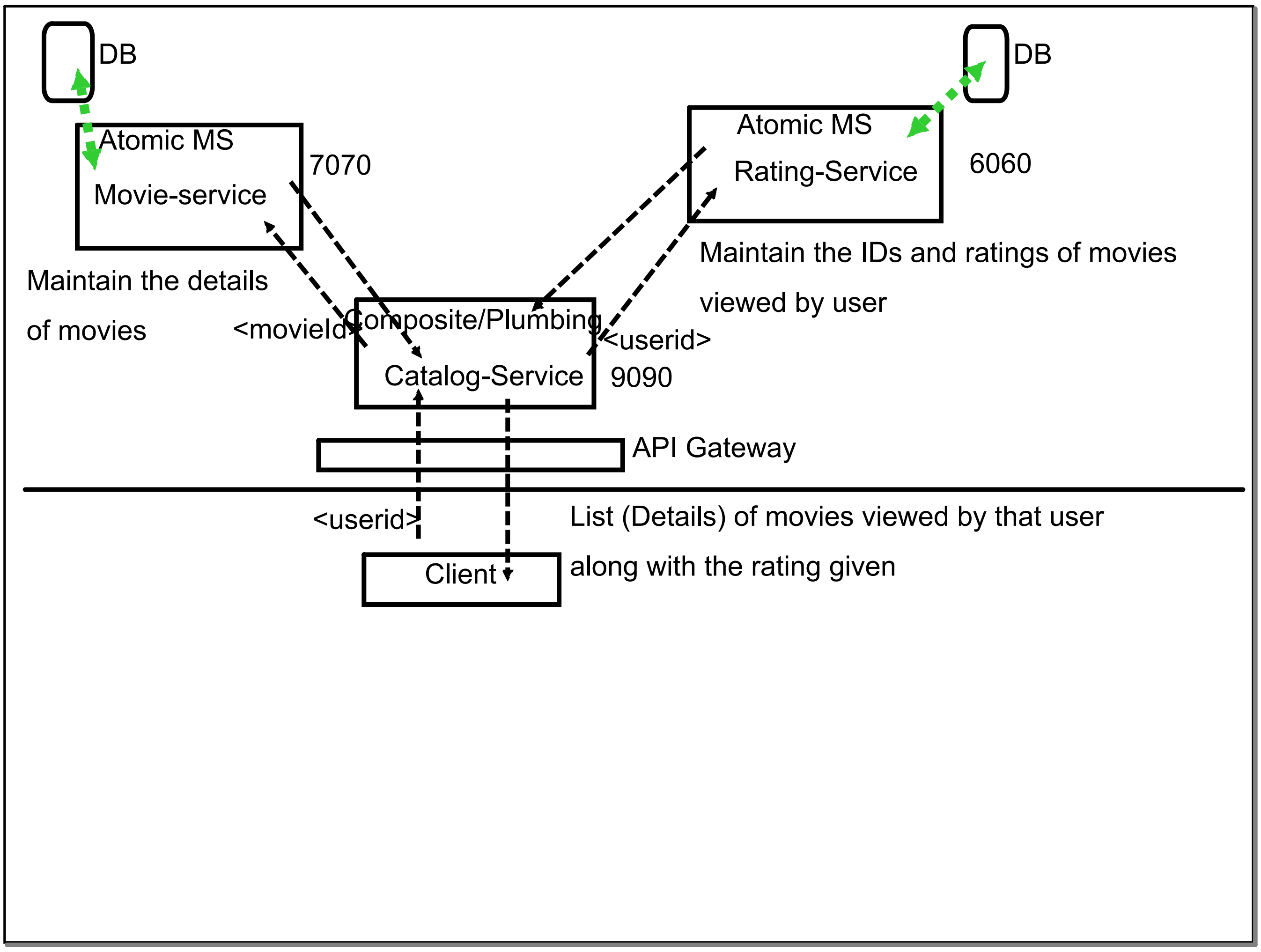


Use Case:

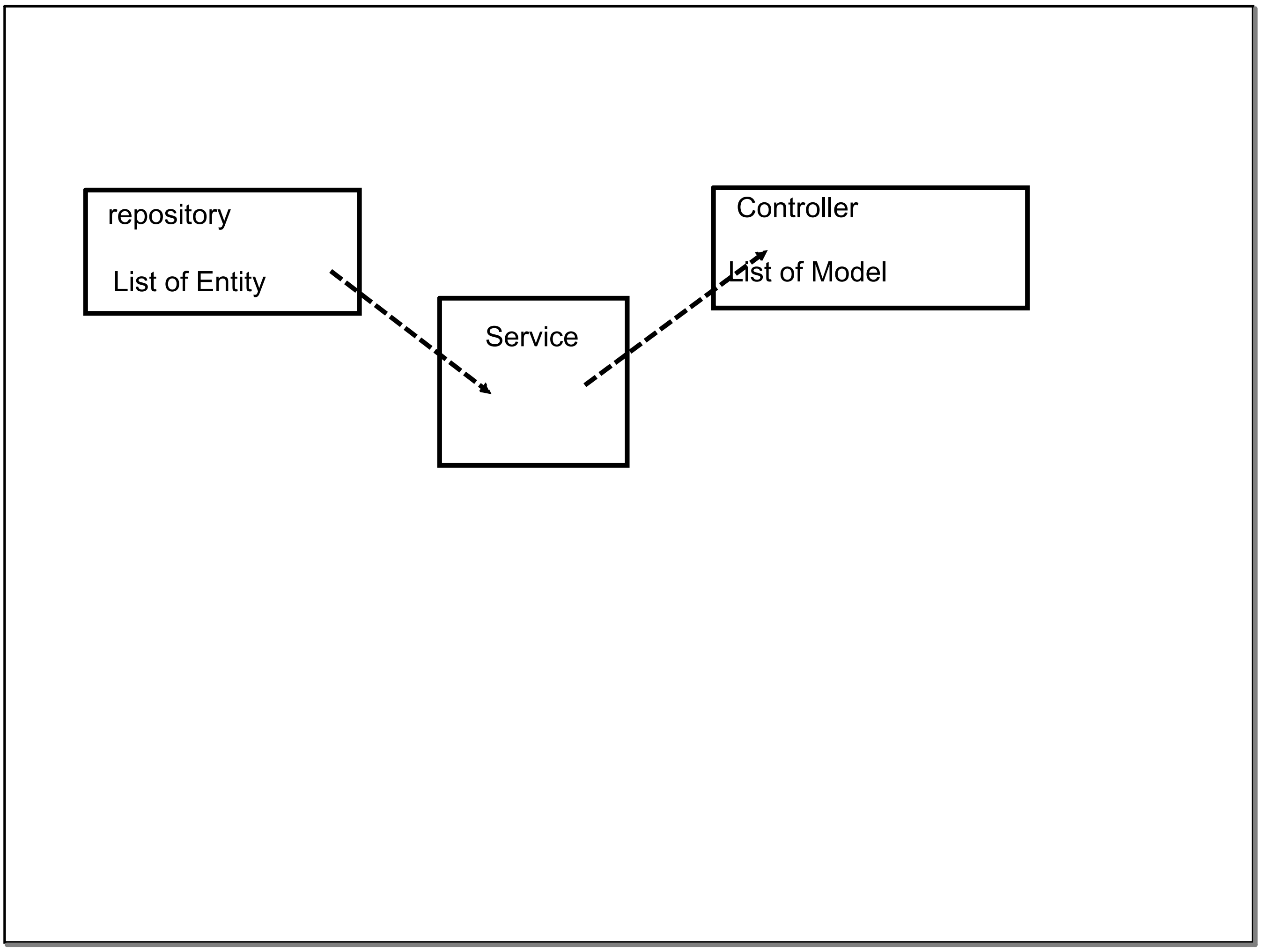
# Movie Streaming portal

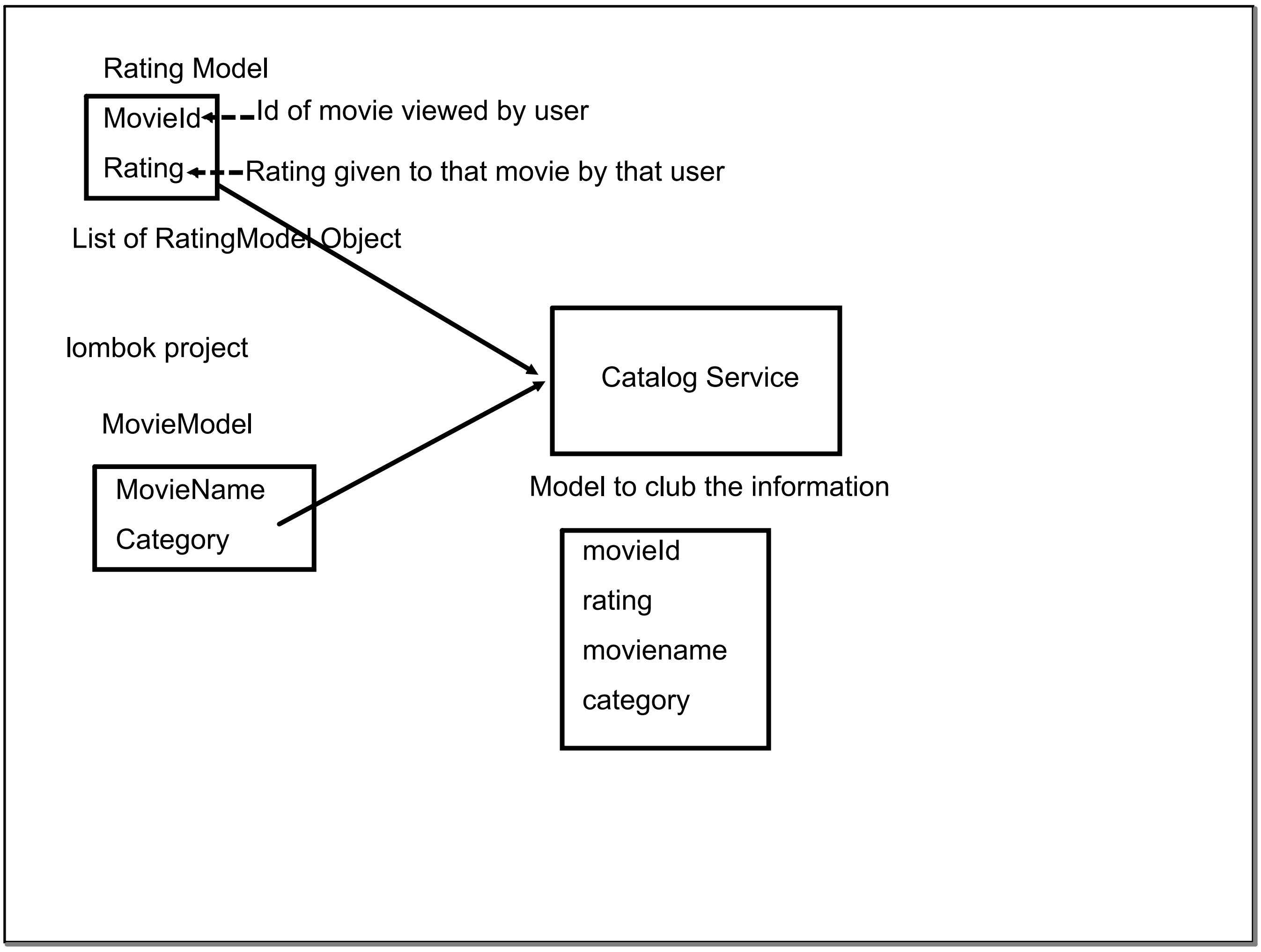

Server
userid
List (Details) of movies viewed by that user along with the rating given
Client

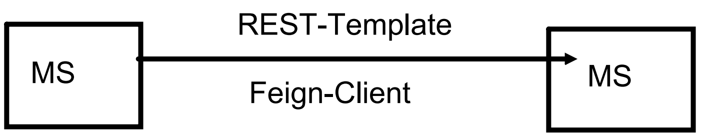
Micorservices



Micorservices

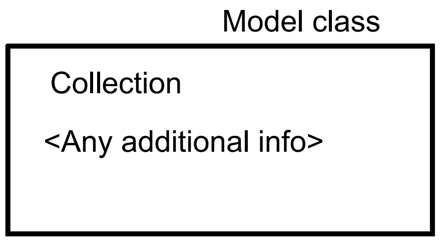



Rating Model
MovieId
Rating
Id of movie viewed by user
Rating given to that movie by that user
List of RatingModel Object
lombok project
MovieModel
MovieName
Category
Catalog Service
Model to club the information
movieId
rating
moviename
category



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

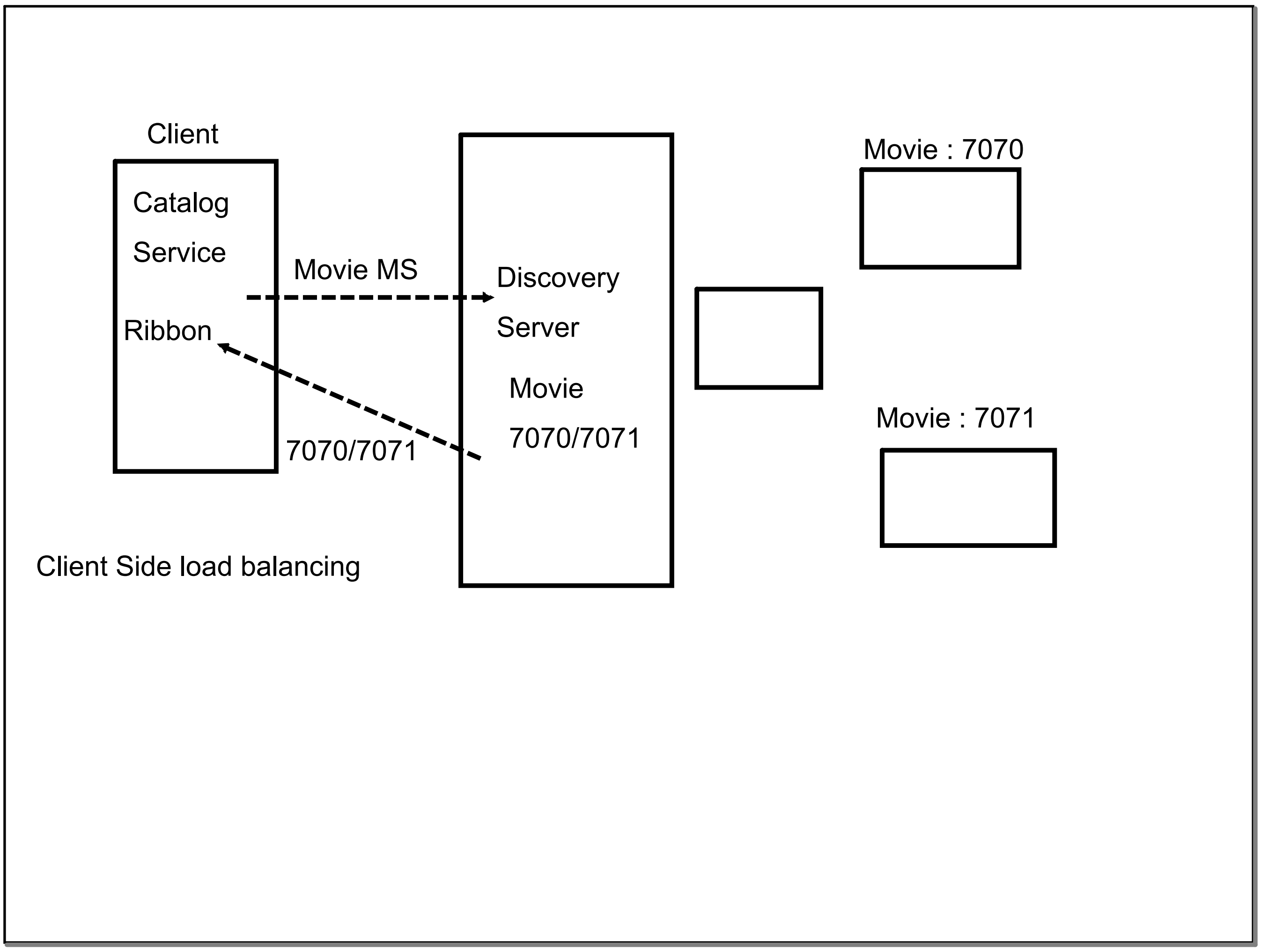
# Create interface reflecting the http connection details of other MS (configuration file)

# for every MS separate interface

# Register all proxies with spring boot application

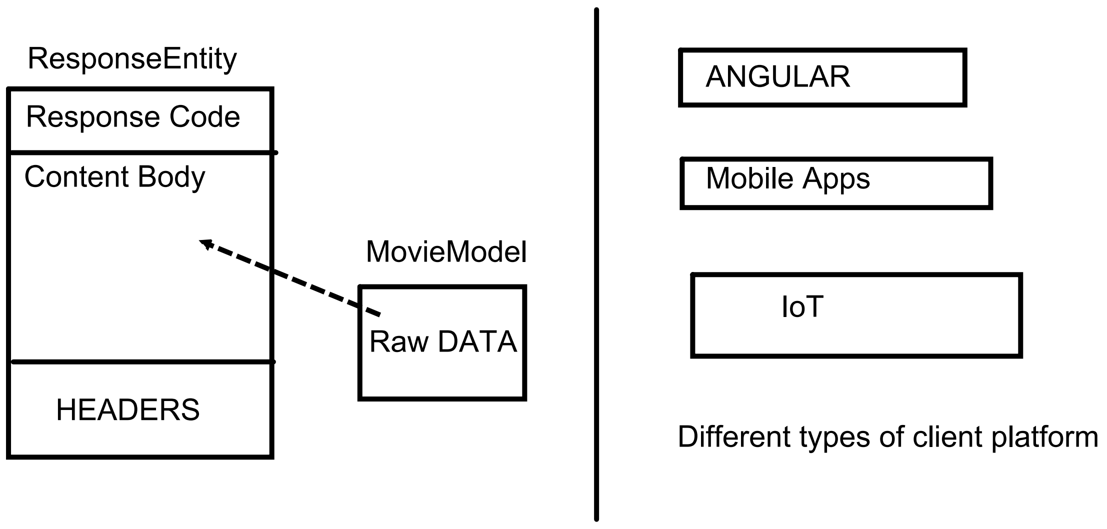
Load Balancing : Ribbon (Client Side load balancer)

# Add dependency

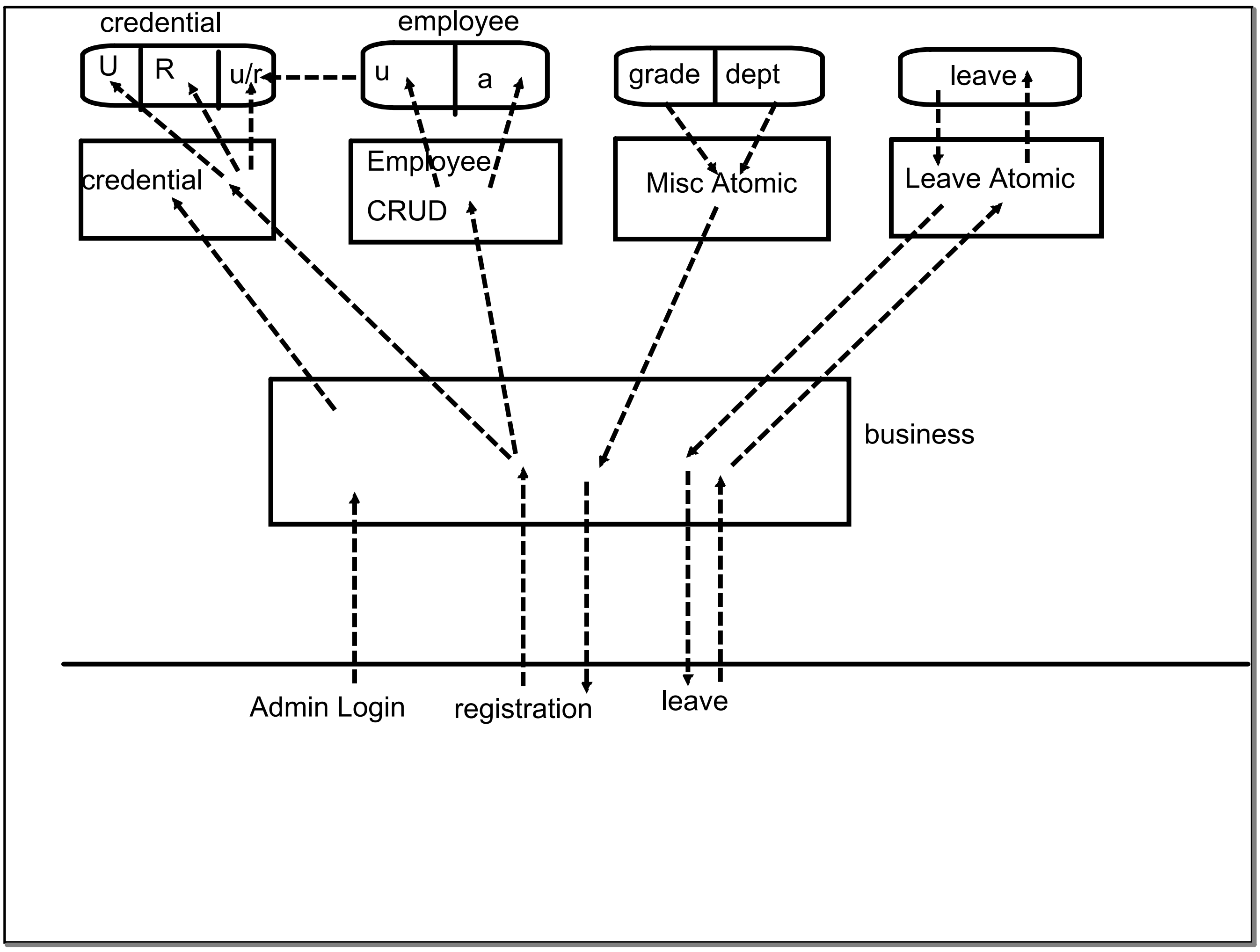

Client
Catalog
Service
Ribbon
Movie MS
7070/7071
Discovery
Server
Movie
7070/7071
Movie : 7070
Movie : 7071
Client Side load balancing

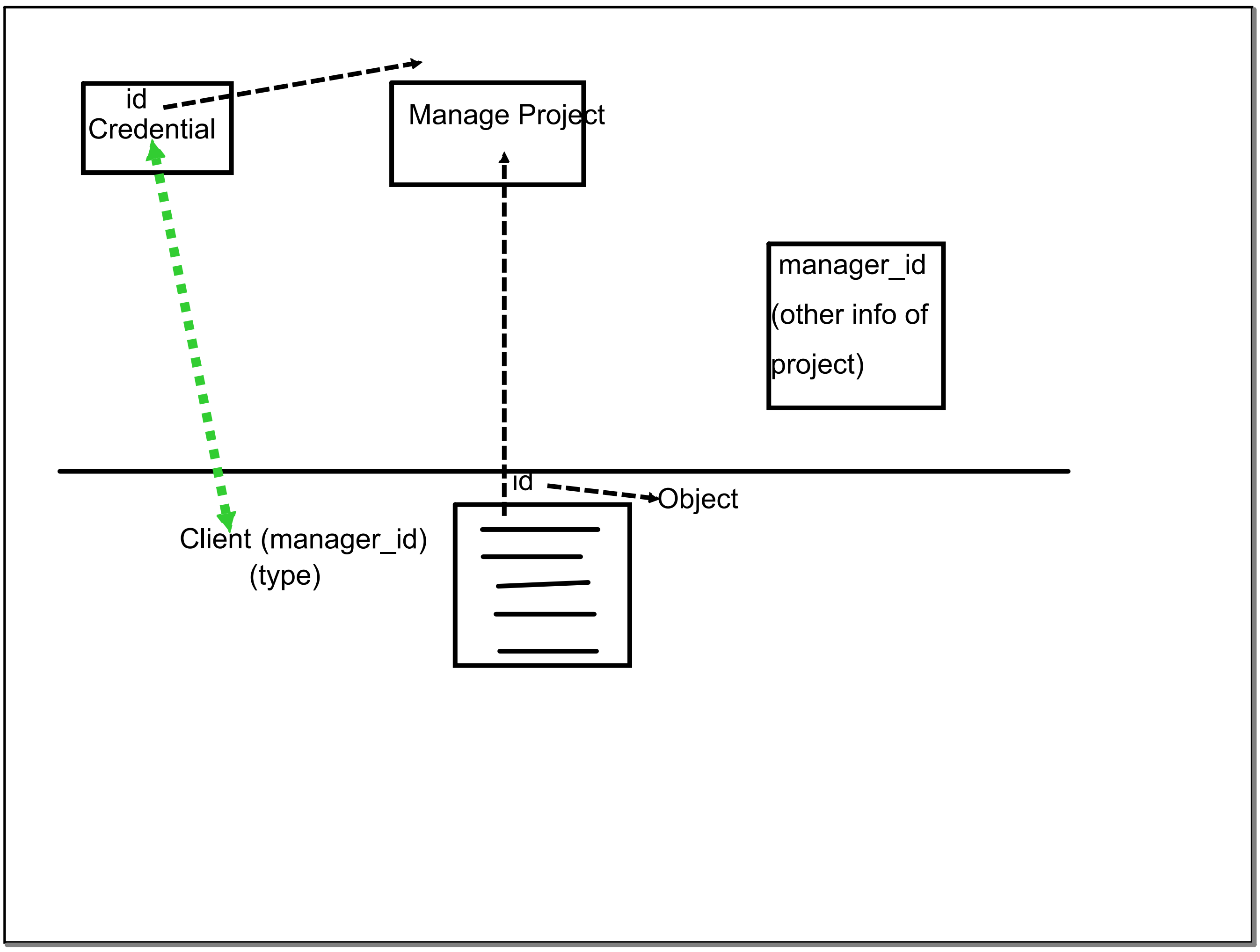
Best Practices

: instead of returning raw data : wrap it in ResponseEntity

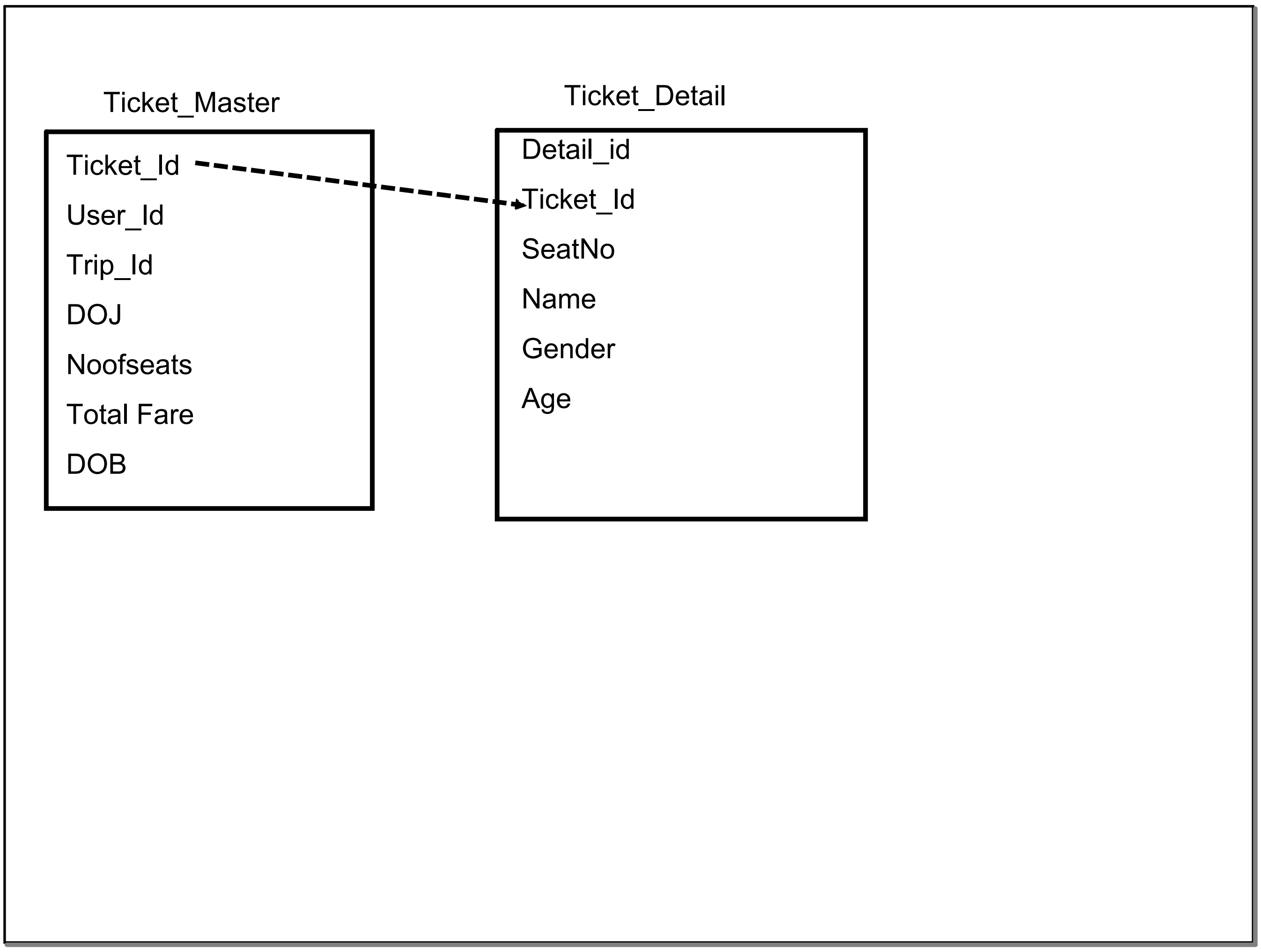


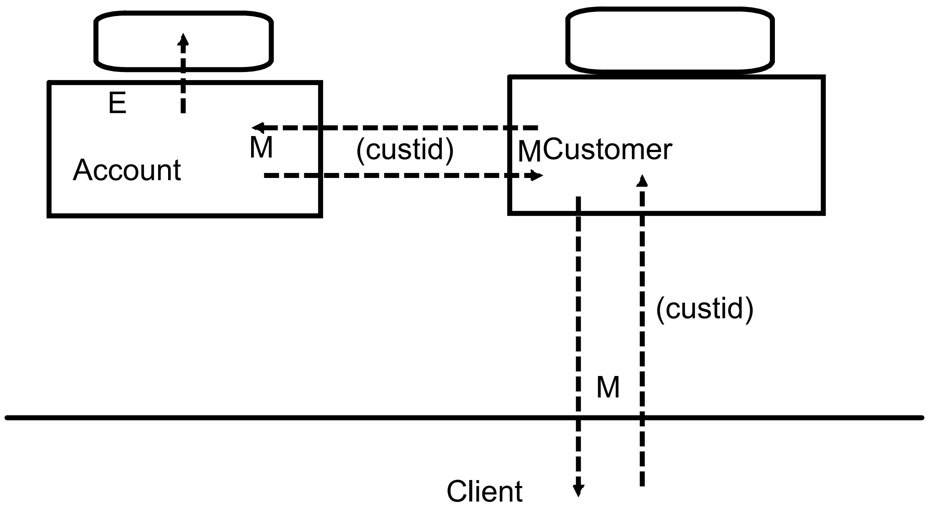
Micorservices




id
Credential
Manage Project
manager_id
(other info of
project)
id
Object
Client (manager_id)
(type)

Micorservices




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