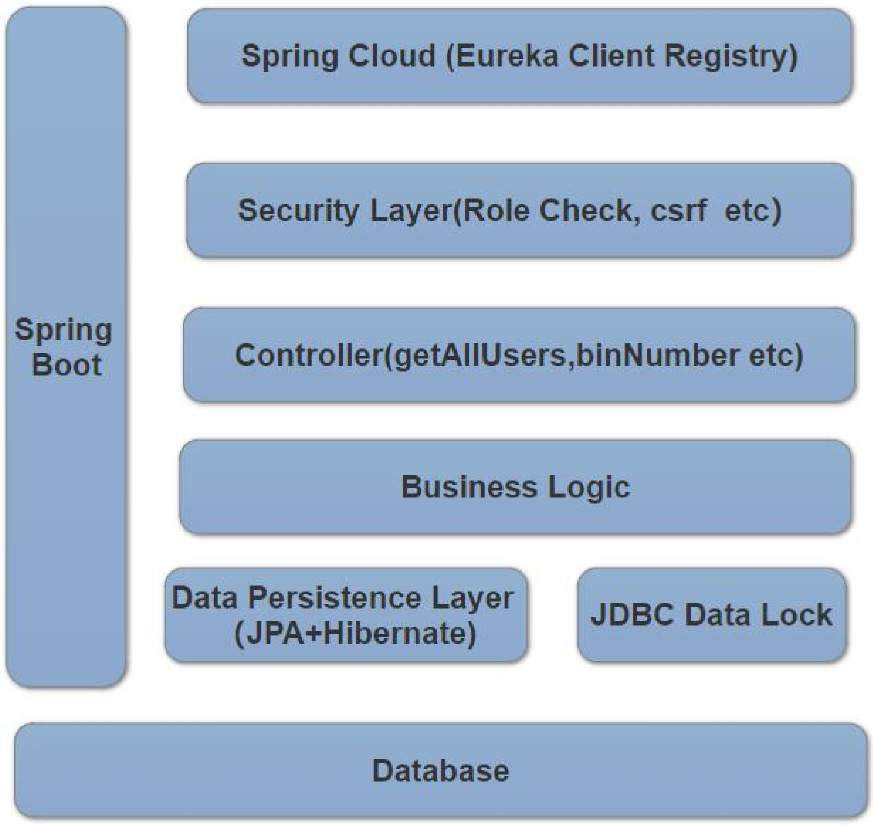
Design Specification for Mobile Number Management

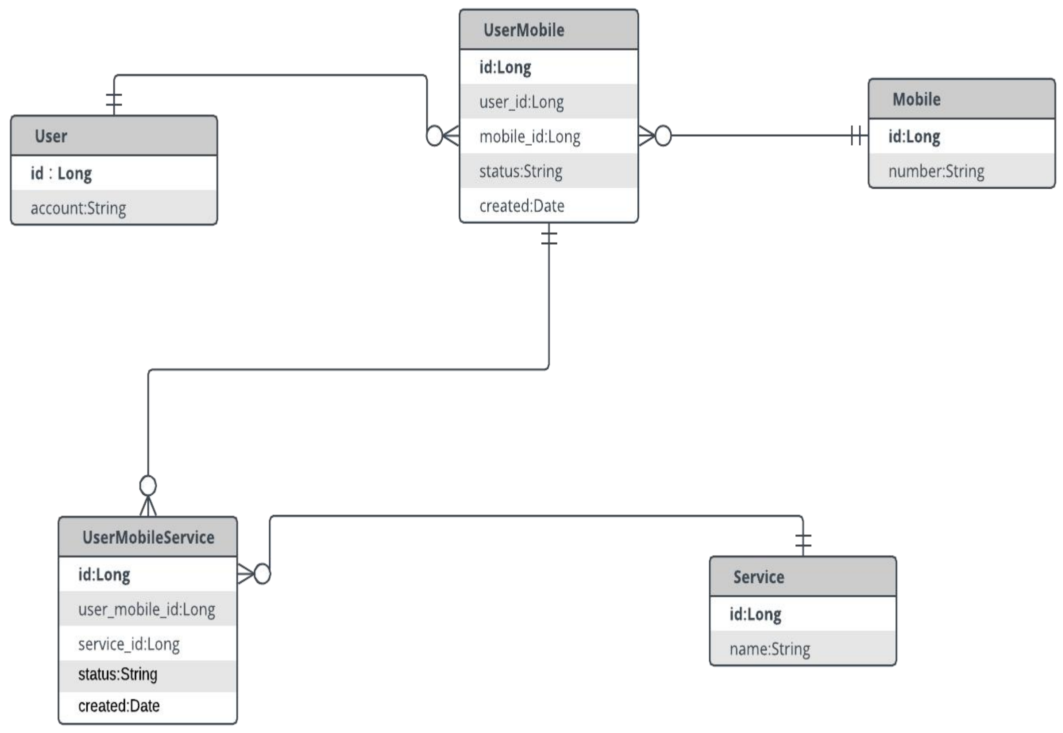
Author	Time	Version	Reviewer	Comment
Yang Yongjie	2018/11/26	1.0		Initial Release

System Design Architecture


Spring Boot
Spring Cloud (Eureka Client Registry)
Security Layer(Role Check, csrf etc)
Controller(getAllUsers,binNumber etc)
Business Logic
Data Persistence Layer (JPA+Hibernate)
JDBC Data Lock
Database

The system is implemented by Spring Boot+JPA+Hibernate, Spring Eureka Server, and a jdbc distributed lock library.

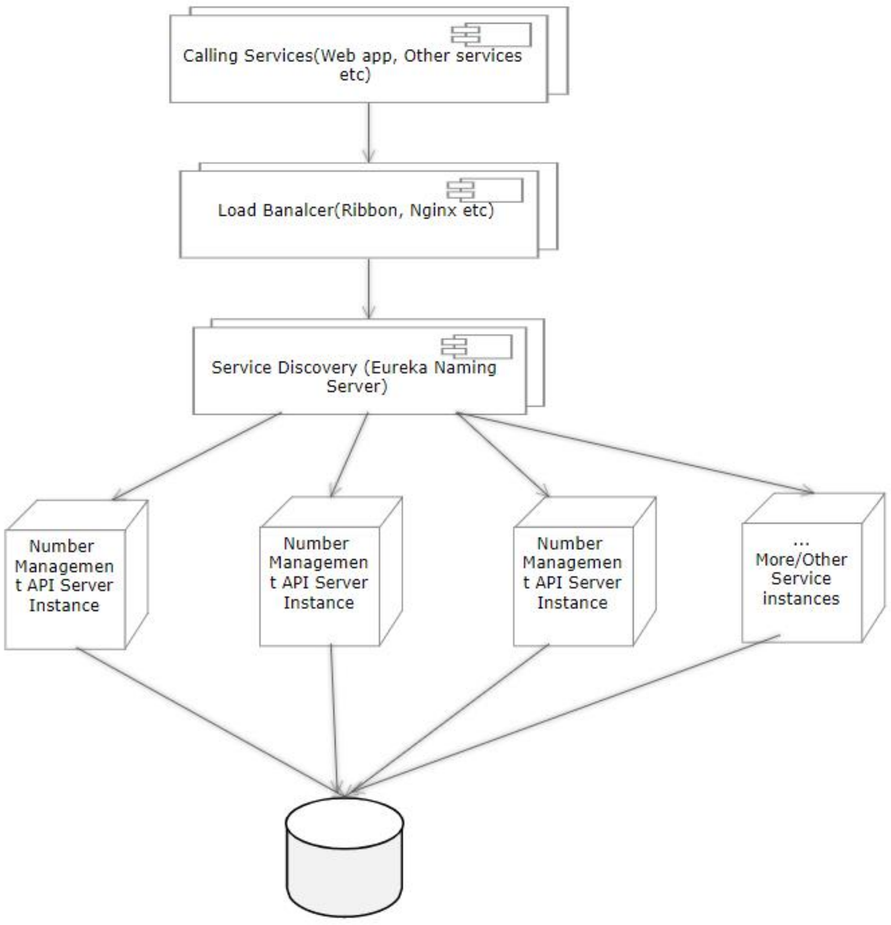
Database Entity Relationship Diagram



Tables:

User : all the user information
Mobile : all the mobile information and a status to track whether it is occupied
Service: all the services provided by service provider
UserMobile: all the user bind/unbind mobile information
UserMobileService: all the user bind/unbind services with a mobile information

Deployment Diagram


Calling Services(Web app, Other services etc)
Load Banalcer(Ribbon, Nginx etc)
Service Discovery (Eureka Naming Server)
Number Managemen t API Server Instance
Number Managemen t API Server Instance
Number Managemen t API Server Instance
... More/Other Service instances

Using Load balancer, Naming Server for multiple instances of service, in case horizontal scale is needed.

Discussions

● API exposed to consumer application

Assume there is a public application (Web,mobile) for users to view/bind/unbind numbers and services themselves, and there is also an admin portal for admin usage. Then we have the following exposure table:

API	Admin	Public
getAllUsers	Y	N
getAllMobiles	Y	N
getAllServices	Y	Y
getAvailableMobiles	Y	Y
getCurrentStatus	Y	N
getMobileHistory	Y	N
bindNumber	Y	Y
unBindNumber	Y	Y
bindService	Y	Y
unBindService	Y	Y
getUserFullPicture	Y	Y

This is illustrated in the class SecurityConfig.java

● Concurrency Handling

In this design, there can be two situations when currency causes issue.

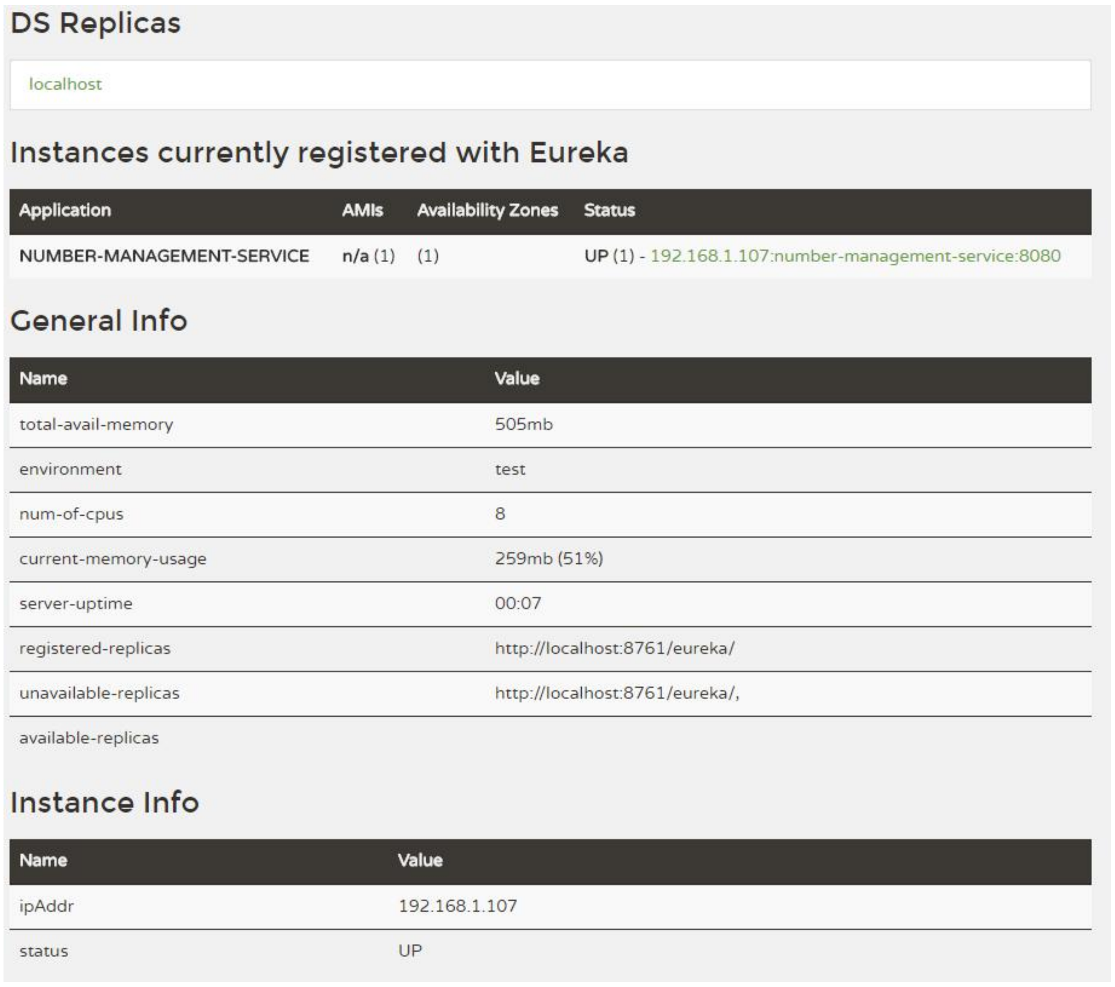
1. Two or more users binding for the same mobile number. This is solved by hibernate optimistic locking. Add a version tag in mobile table, whenever a binding happens, firstly this table need to be updated, using hibernate version support can eliminate this issue.

2. User bind/unbind a service, while at the same time, admin bind/unbind the same service for the same user of the same mobile number, this will cause duplicate insertion into the user_mobile_service table. Solved by adding a distributed lock: add a lock table, make the user account, mobile number, service name as a composite key, and insert into the lock table. While this key exists in the lock table, if other thread wants to do the same thing, and it finds the same key in the table, then will not proceed.

● Extending the microservice horizontally

The system should be able to scale horizontally, i.e. add more server instances to handle increasing load. This achieved by using spring cloud, add eureka client into the service, then we can user Eureka Naming server to handing multiple instances.

The following screenshot is an example of a instance registered to naming server.



DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
NUMBER-MANAGEMENT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:number-management-service:8080

General Info

Name	Value
total-avail-memory	505mb
environment	test
num-of-cpus	8
current-memory-usage	259mb (51%)
server-uptime	00:07
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,
available-replicas	

Instance Info

Name	Value
ipAddr	192.168.1.107
status	UP

- ## Others

For Spring boot+JPA+Hibernate 5, it is advised to use JPA native persistence manager, rather than the hibernate session factory, hibernate dao support. But for illustration of hibernate usage, this project still uses hibernate session factory.

Since Spring Eureka client introduced into the project, and embedded DB used, the server start up time is a bit long, maven test is disabled in default to faster the process. If there is a need to run the test, run this command: mvn test -Dmaven.test.skip=false