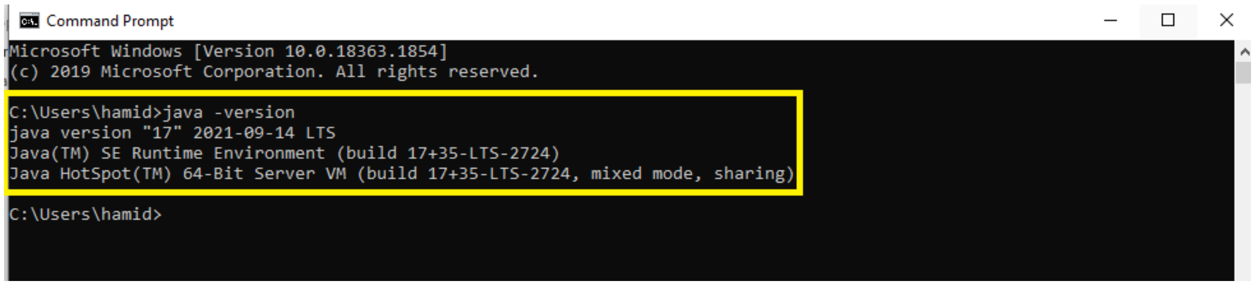
Deployment Document

Requirements:

Java development Kit (JDK): version-> 17



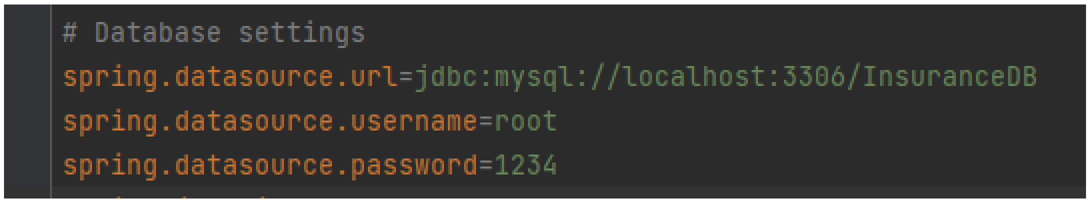
```
Microsoft Windows [Version 10.0.18363.1854]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hamid>java -version
java version "17" 2021-09-14 LTS
Java(TM) SE Runtime Environment (build 17+35-LTS-2724)
Java HotSpot(TM) 64-Bit Server VM (build 17+35-LTS-2724, mixed mode, sharing)

C:\Users\hamid>
```

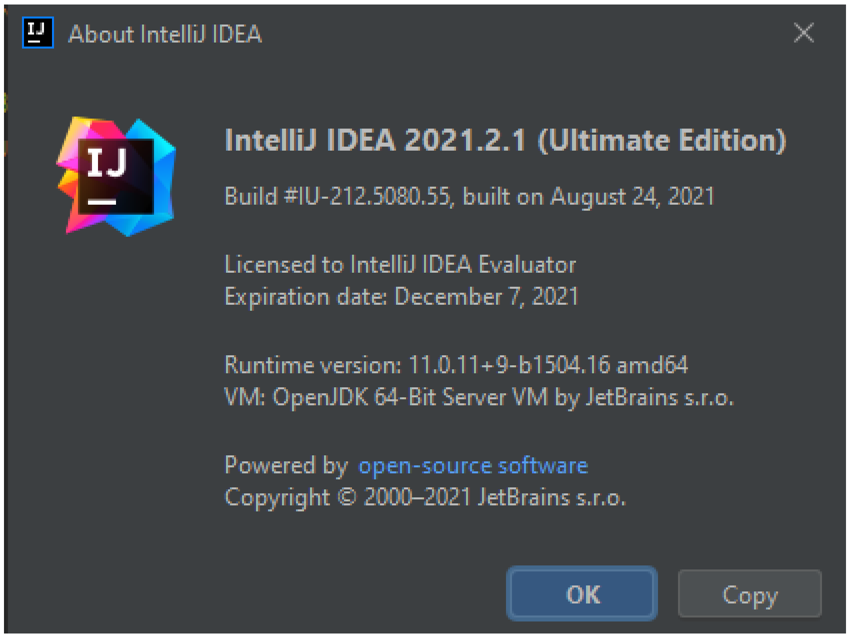
MySQL Data Base: version -> MySQL Server 5.7

You can find database name, user name and password in this picture:



```
# Database settings
spring.datasource.url=jdbc:mysql://localhost:3306/InsuranceDB
spring.datasource.username=root
spring.datasource.password=1234
```

Intellij IDEA: version -> 2021.2.1(Ultimate Edition)



About IntelliJ IDEA

IntelliJ IDEA 2021.2.1 (Ultimate Edition)

Build #IU-212.5080.55, built on August 24, 2021

Licensed to IntelliJ IDEA Evaluator
Expiration date: December 7, 2021

Runtime version: 11.0.11+9-b1504.16 amd64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

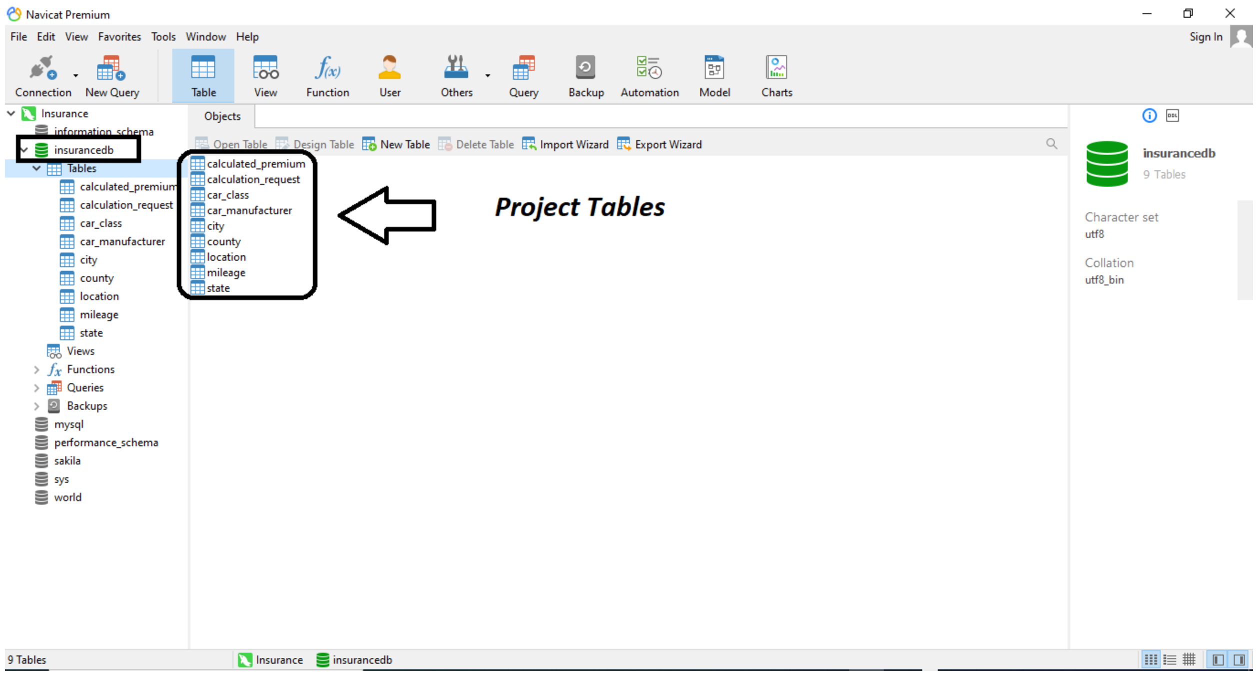
Powered by open-source software
Copyright © 2000–2021 JetBrains s.r.o.

Modules Run Priority

1- Discover Service
2- MileagePolicy – RegionPolicy – CarPolicy
3- Calculation
4- Getway

DATA BASE

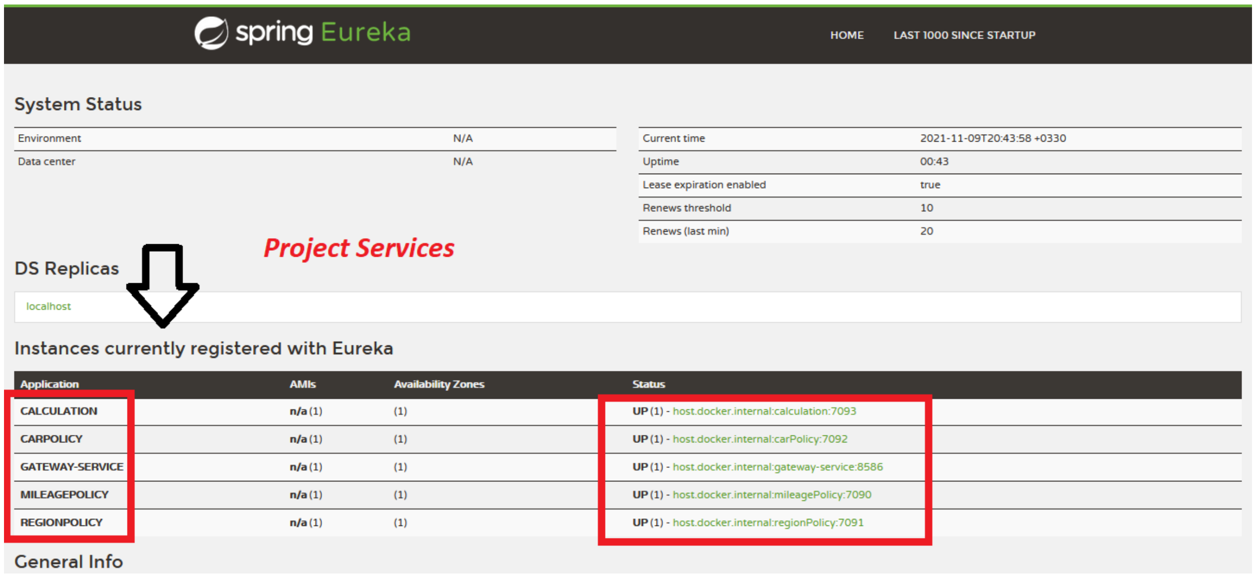
After running projects you can see created tables in MYSQL Database.



Netflix Eureka Panel

You can access service status in this URL:

Eureka Panel URL: http://localhost:8761



spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	N/A
Data center	N/A

Current time	2021-11-09T20:43:58 +0330
Uptime	00:43
Lease expiration enabled	true
Renews threshold	10
Renews (last min)	20

Project Services

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CALCULATION	n/a (1)	(1)	UP (1) - host.docker.internal:calculation:7093
CARPOLICY	n/a (1)	(1)	UP (1) - host.docker.internal:carPolicy:7092
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:gateway-service:8586
MILEAGEPOLICY	n/a (1)	(1)	UP (1) - host.docker.internal:mileagePolicy:7090
REGIONPOLICY	n/a (1)	(1)	UP (1) - host.docker.internal:regionPolicy:7091

General Info

After running Project, you can test with Postman

According Database initialization you can use this data for test this API:

Postal Code: 555 or 666 or 777 or 888

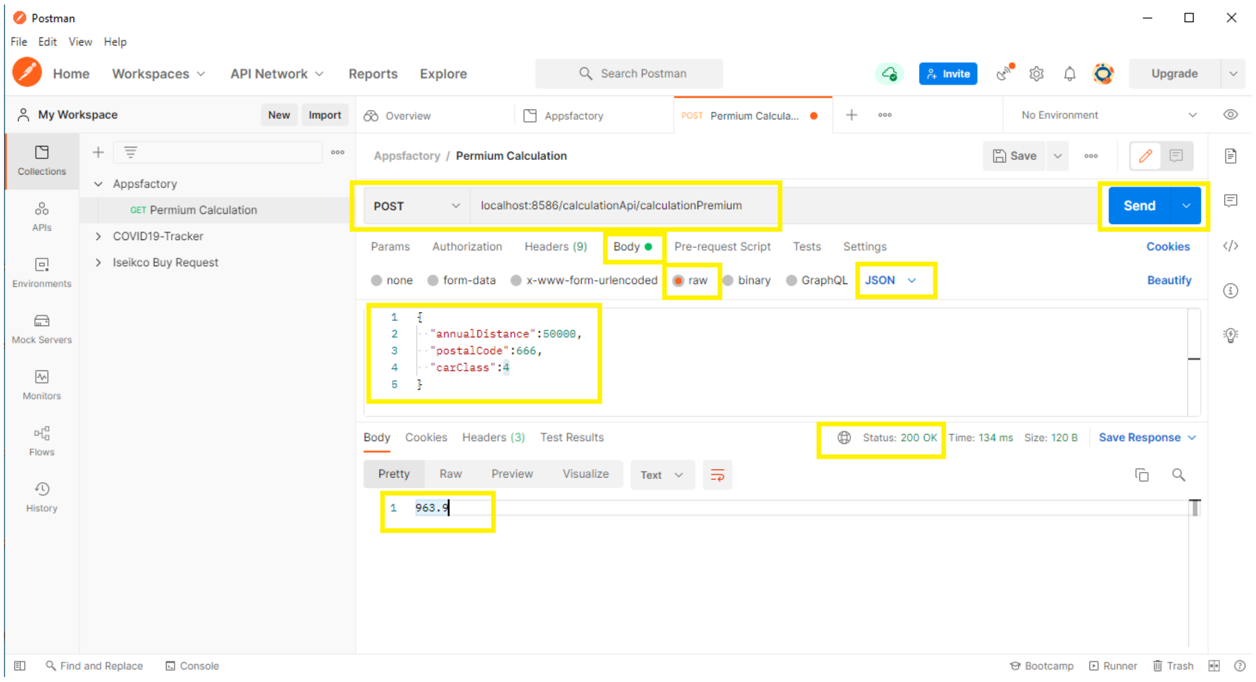
Mileage Distance Between: 0 - 1000000 (KM)

Car Class: between: 1 - 6

{ localhost:8586/calculationApi/calculationPremium
localhost:8586/carApi/getAllCarManufacturer } ***for Third Party***

{ localhost:8586/calculation/calculationPremium
localhost:8586/carPolicy/getAllCarManufacturer } ***for Local Usage***

Calculate Premium



Appsfactory / Permium Calculation

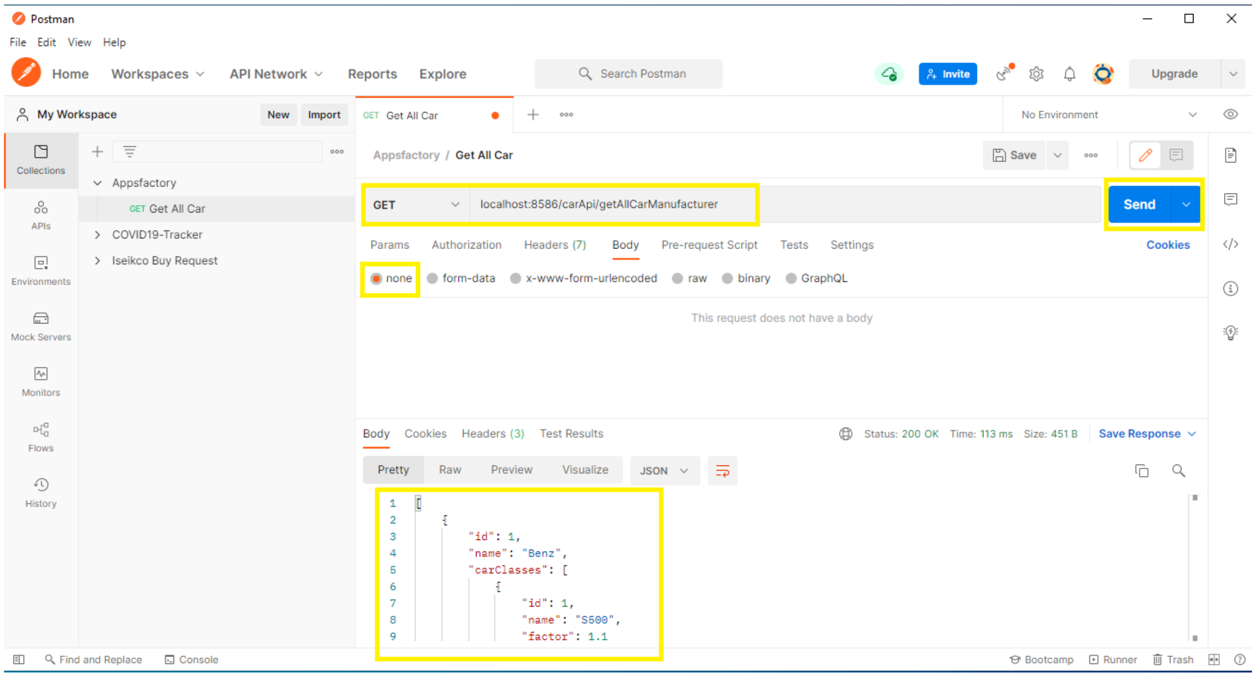
POST localhost:8586/calculationApi/calculationPremium

```
{
  "annualDistance":50000,
  "postalCode":666,
  "carClass":4
}
```

Status: 200 OK Time: 134 ms Size: 120 B

```
963.9
```

Get All Car Data



Appsfactory / Get All Car

GET localhost:8586/carApi/getAllCarManufacturer

Body: none

This request does not have a body

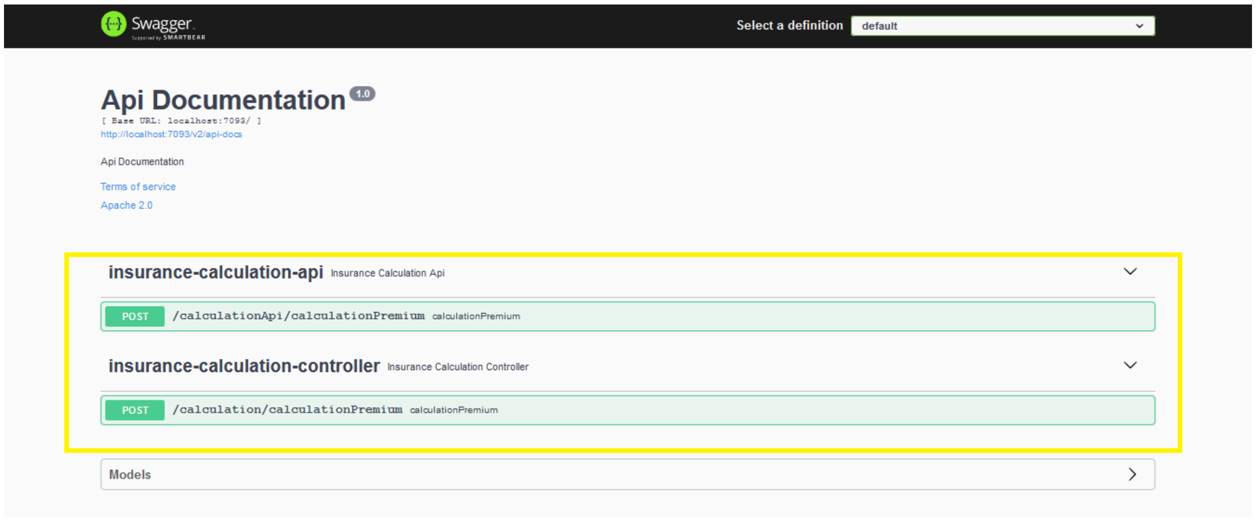
Status: 200 OK Time: 113 ms Size: 451 B

```
[
    {
        "id": 1,
        "name": "Benz",
        "carClasses": [
            {
                "id": 1,
                "name": "S500",
                "factor": 1.1
```

Swagger UI

You can Access Swagger UI panel in this URL:

http://localhost:7093/swagger-ui/index.html



Api Documentation 1.0
[Base URL: localhost:7093/]
http://localhost:7093/v2/api-docs

Api Documentation

Terms of service
Apache 2.0

insurance-calculation-api Insurance Calculation Api

POST /calculationApi/calculationPremium calculationPremium

insurance-calculation-controller Insurance Calculation Controller

POST /calculation/calculationPremium calculationPremium

Models

How to improve this Project quality?

I recommend:

- **Using OAuth 2.0 and JWT for Third Party Authentication**
- **Using Aspect Oriented Programming(AOP) and Cross-Cutting Concern thinking for Logging any API, Controller, Service calling**
- **Using advance Filter on Gate Way for checking Request and improving Security**
- **Using Spring Cloud Circuit Breaker for handling Microservice failing**
- **Using Docker File for creating Image and container and running Project on Docker platform and use Kubernetes for Microservices management**