Application Initialisation :

Database ‘bms’ is created for the application with no table at initial stage

A black screen with white text

Description automatically generated

Tables will be created by the applications on startup as we mentioned following hibernate property

spring.jpa.hibernate.ddl-auto=create-drop

After a successful start of application , table got created as shown.

A screenshot of a computer

Description automatically generated

Service Registry Console after registering with clients (property, banking, and payment) and api\_gateway and config server also as below:

A screenshot of a computer

Description automatically generated

Requests : all the requests are using api\_gateway (port No : 9002)

1. Post request for creating a property in database

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. Get all properties from database :

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. Retrieving a property from repository can be done by using location, price, type, title and even property id :

I. Using id :

A screenshot of a computer

Description automatically generated

II. using type:

A screenshot of a computer

Description automatically generated

1. Booking a property : (post request from booking microservice.)

A screenshot of a computer

Description automatically generated

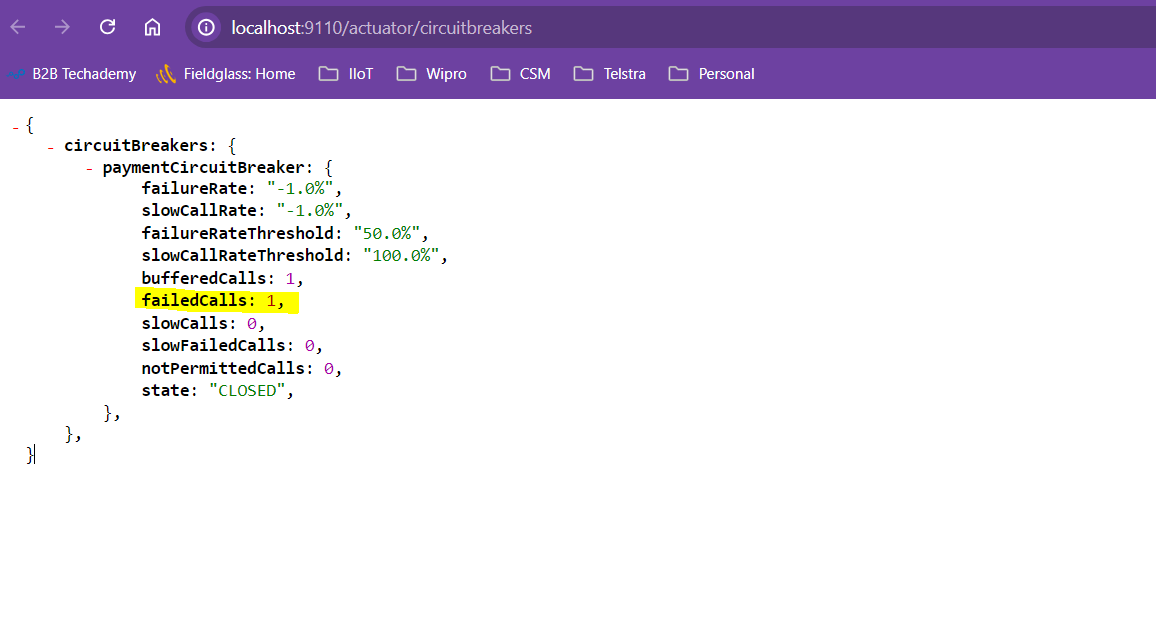
Failure Case (covered in circuit breaker fallback scenario) :

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



Post request on booking microservice will in turn call the property microservice for the availability of property with given id and checks for the number of rooms required is less than the available ones. Then payment microservice will be called for the payment for booking that property, if it is successful then booking status will be Booked and available rooms will be decremented by the number of rooms booked. else it will be failed. As the booking is successful for property id 1 , the available rooms got reduced from 5 to 3 as shown below.

A screenshot of a computer

Description automatically generated

Note:

All the microservices are clients for the Eureka server which is running on port 9000. Config server microservice (running on port 9001) holds the property files for the other microservices (booking ,payment, property ). All the crud operations made on the property (port no. 9109) and booking(port no. 9110) and payment(port no.9111) microservices are through the api\_gateway which was running on port no. 9002.

* Booking microservice is having circuit breaker (resilience4j) implemented as it has the internal calls for property and payment microservice. So, if any microservice is down or having some issue , as per the configuration the circuit will be either open, half-open or closed and from open to half-open , automatic transition will take place after 5s (given in config).

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Distributed tracing using Zipkin :

As shown below, for the post request of booking a property, these are the internal calls that are being made to other microservices and we can trace them and their respective timings using zipkin.

A screenshot of a computer

Description automatically generated

* Monitoring using Prometheus:

Number of calls made per microservice, and their response status etc can be monitored by Prometheus which can be configured using prometheus.yml as show below.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

* Documentation using Swagger3 :

All the api’s exposed and its features can be viewed in swagger documentation and even we can try out testing the rest api from here.

A screenshot of a computer

Description automatically generated

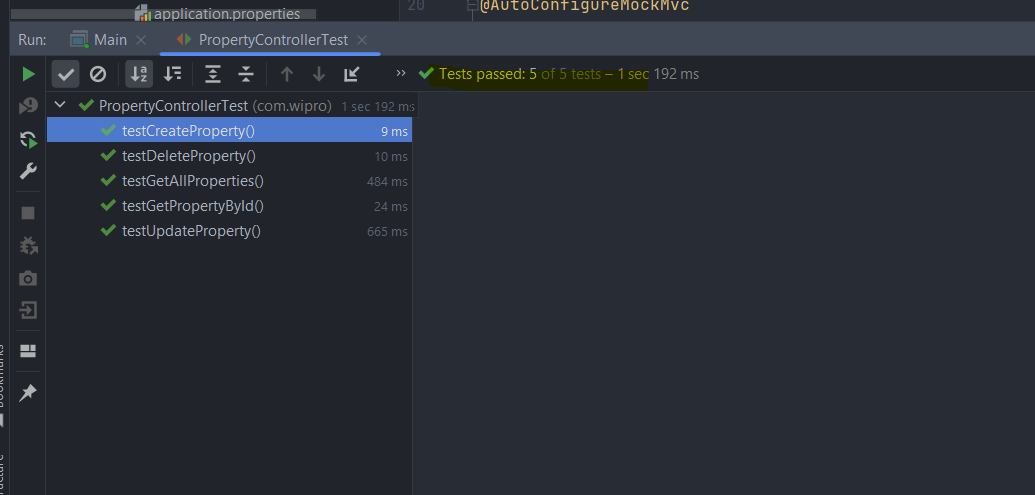
A screenshot of a computer

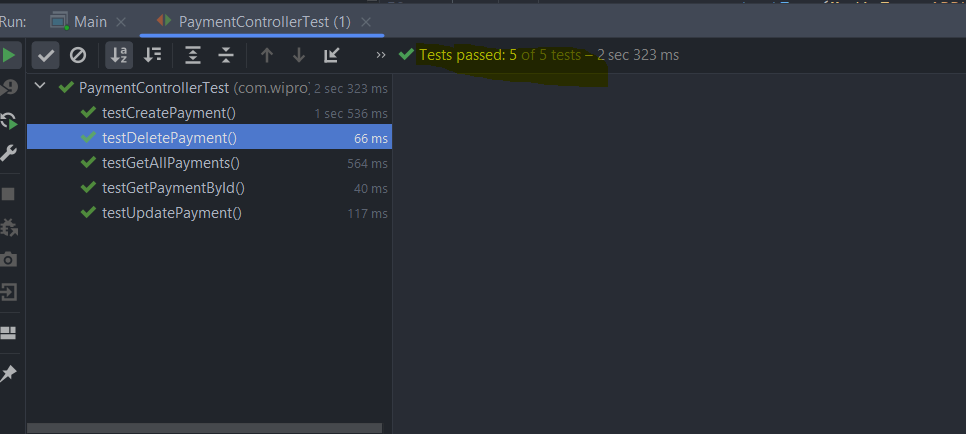
Description automatically generated

A screenshot of a computer

Description automatically generated

* All the microservices ( property, booking and payment) have the rest endpoints for the CRUD operations. So, we can use either create, update, select and delete functionality as show in the documentation also.
* Test cases are also included for these microservices at the controller level as shown below:





A screenshot of a computer program

Description automatically generated