**POC – Parallel Microservice Call**

**UseCase**: Create microservices to calculate the price for selected items. The microservice that pulls the price from database is taking only one item at a time. Write the program to calculate the price in parallel mode in order to improve the performance.

**Design:**

1. Develop and build the **dbItemPrice- Services** microservice
   1. Set the item in database
   2. Fetch the price for given Item ID
   3. Fetch the all item from the database
   4. Delete the item for given itemid
2. Develop and Build the **ItemPrice-Services** microservice
   1. Get the price for selected Item ids
   2. All the logic for parallel processing of price calculation
3. Develop and build the **Eureka-Service** microservice and configure the Zuul API gateway and Service registry.
4. Register the dbItemPrice-Services and ItemPrice-Service into Eureka Server/Geteway.

Browser/Postman

dbItemPrice-Services

Service Registry

API Gateway

H2 Database

ItemPrice-Services

**Technology Stack**

1. Java 1.8
2. Springboot 2.0.4
3. JPA
4. Spring Cloud ( Zuul Proxy, Eureka, Ribbon- client side load balancing)
5. H2 Database

**Validation:**

1. Start the all services
2. Use API gateway URL to add the item in Database (<http://localhost:8761/api/dbItemPrice-Services/rest/db/add>

JSON payload example :

[

{

"itemPrice": "7.49",

"itemName": "Pen"

},

{

"itemPrice":"18.19",

"itemName": "SunGlasses"

},

{

"itemPrice": "29.99",

"itemName": "Socks"

},

{

"itemPrice": "40.99",

"itemName": "Shirt"

}

]

1. Use the API gateway URL to validate the Items added in database (<http://localhost:8761/api/dbItemPrice-Services/rest/db/items>
2. Use API gateway URL to calculate the price for given itemList (<http://localhost:8761/api/ItemPrice-Services/rest/price/calcualtePrice?productIds=1,2,3>
3. You can use the API gateway URI to delete the item from database ( <http://localhost:8761/api/dbItemPrice-Services/rest/db/3>