*ONLINE HOTEL RESERVATION APPLICATION*

continut: SERVICE DISCOVERY, CENTRALIZED CONFIGURATION, DISTRIBUTED TRACING, EVENT DRIVEN ARCHITECTURE, CENTRALIZED LOGGING, CIRCUIT BREAKER, SECURE MICROSERVICE USING KEYCLOCK

PROJECT OVERVIEW

Modului principal care ne ajuta să scriem microservicii folosind **Spring Framework** este **Spring Cloud**. Principalele componente ale proiectului nostru sunt:

1. Spring Cloud Config Server

2. Spring Cloud Bus

3. Spring Cloud Netflix Eureka

4. Spring Cloud Circuit Breaker

5. Spring Cloud Sleuth

6. Spring Cloud Gateway

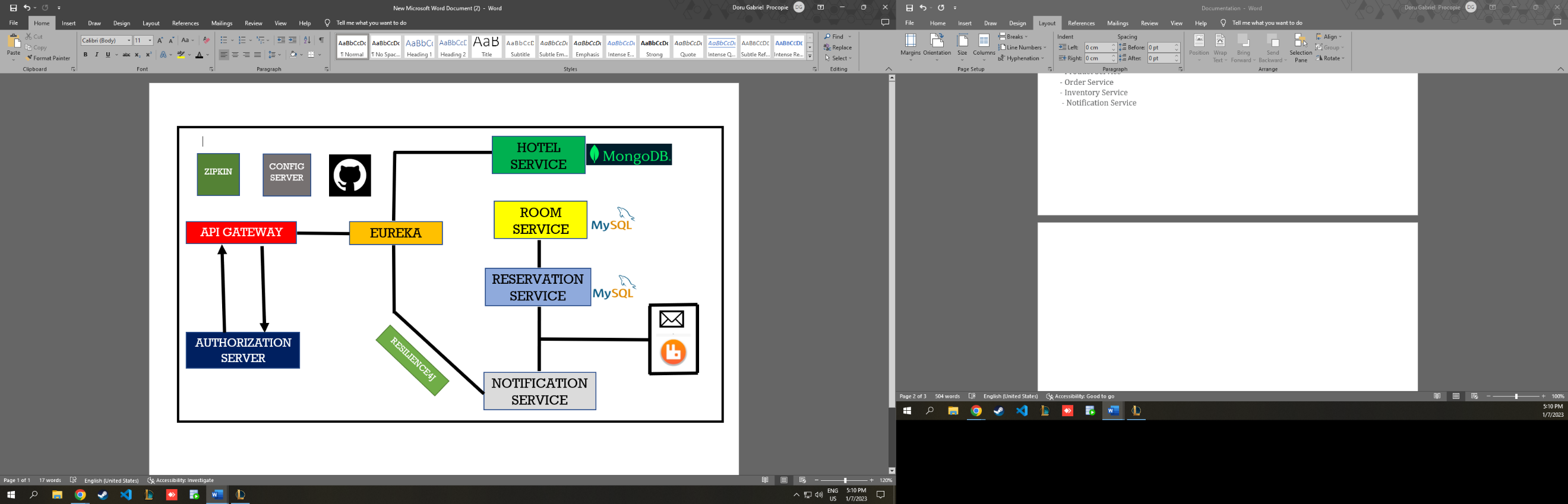
7. Spring Cloud Stream

Vom implementa 4 microservicii și anume:

- Hotel Service

- Room Service

- Reservation Service

 - Notification Service

**1. Spring Cloud Config Server**

Acest modul este folosit pentru a externaliza configuratia microserviciilor noastre într-un loc centralizat. Dacă există modificări în configurație, aplicațiile noastre ar trebui să fie actualizate fără a fi nevoie să le repornim.

Există multe opțiuni pentru a implementa această configurație centralizată, cum ar fi folosirea unui repository Git. De asemenea, vom folosi HashiCorp Vault ca unul dintre backend-uri pentru a menține secretele aplicației noastre.

**2. Spring Cloud Bus**

Acest modul conține o implementare ușoară a brokerului de mesaje, care este folosită în principal pentru a difuza unele mesaje către celelalte servicii.

În proiectul nostru, vom folosi acest modul pentru a difuza modificările de configurare în serverul nostru de configurare.

**3. Spring Cloud Netflix Eureka**

Acest modul este folosit pentru a implementa Service Reistry și Discovery Pattern în arhitectura nostru cu microservicii. Deoarece pot exista multe microservicii independente, avem nevoie de o modalitate fiabilă de a scala serviciile și de a oferi comunicații între servicii, în loc să codificăm informațiile despre serviciu.

Modulul Spring Cloud Netflix Eureka folosește biblioteca Netflix Eureka pentru a oferi această funcționalitate.

**4. Spring Cloud Circuit Breaker ( RESILIENCE4J )**

Comunicarea între servicii este o caracteristică importanta a acestui tip de arhitectura pecare dorim să o efectuăm mai ales dacă unul dintre servicii este nefuncțional, iar celălalt serviciu care comunică cu acesta, ar trebui să poată gestiona această problemă.

Spring Cloud Circuit Breaker oferă o abstractizare asupra diferitelor implementări de Circuit Breaker, cum ar fi Reslience4J, Hystrix. În proiectul nostru, vom folosi Spring Cloud Circuit Breaker cu Resilience4J.

**5. Spring Cloud Sleuth**

În cadrul arhitecturii pe microservicii, dacă există vreo eroare, este greu să o depanăm și să urmărim aparitia unei defecțiuni. Spring Cloud Sleuth ne oferă funcționalitatea de a urmări apelurile dintre servicii.

**6. Spring Cloud Gateway**

Această bibliotecă ne ajută să implementăm modelul API Gateway, ascunzând complexitatea microserviciilor noastre de clienții externi**.** Dacă un client dorește să se conecteze la unul dintre serviciile noastre, tot traficul va trece prin API Gateway, iar gateway-ul va direcționa cererea către serviciul corespunzător.

Această bibliotecă se ocupă, de asemenea, de preocupările transversale precum securitatea, monitorizarea, limitarea ratei și rezistența. Ne vom securiza microserviciile folosind Keycloak ca server de autorizare împreună cu Spring Cloud Gateway.

**7. Spring Cloud Steam ( RABBITMQ )**

Acest modul ne permite în principal să implementăm comunicarea asincronă între microserviciile noastre folosind arhitectura bazată pe evenimente.

Vom folosi RabbitMQ ca broker de mesaje pentru a implementa unele microservicii bazate pe evenimente.

