Online Salon Service Application

Using Spring Boot

Batch-8-J2EE
Name : Shivani Budankayala
E-Mail : shivani132003@gmail.com
Date : Sep-01-2024

Project Flow Chart

STEPS

1. Server Registery and Discovery
2. API Gateway
3. Authentication Service
4. Microservices Management
5. Admin Module and User Module

CONTENT

- Introduction
- Project Explorer
- Technologies Used
- Microservices Architecture
- Admin and User Modules
- Testing and Refinement
- Conclusion

INTRODUCTION

In an era where convenience and personalization are paramount, our online salon service application is designed to meet the evolving needs of beauty and wellness enthusiasts. We aim to elevate the salon experience by leveraging technology to offer a more accessible, efficient, and tailored service.

Our Goal: To transform the way people experience salon services by combining cutting-edge technology with personalized care, making beauty and wellness more accessible and enjoyable.

PROJECT EXPLORER

TECHNOLOGIES USED

- **Java**
- **Spring Boot**
- **Spring Cloud**
- **Spring Data JPA**
- **MySQL/MariaDB**
- **RESTful APIs**

MICROSERVICES ARCHITECTURE

Service Registry & Discovery
Eureka Server:

```
-01T23:08:42.301+05:30  WARN 11104 --- [GateWay] [           main] trationDelegate$BeanPostProcessorChecker : Bean 'deferringLoadBalancerInterceptor' of type [o
-01T23:08:42.303+05:30  WARN 11104 --- [GateWay] [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalance
-01T23:08:42.308+05:30  WARN 11104 --- [GateWay] [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.cloud.client.loadbalance
-01T23:08:42.311+05:30  WARN 11104 --- [GateWay] [           main] trationDelegate$BeanPostProcessorChecker : Bean 'reactorDeferringLoadBalancerExchangeFilterFu
-01T23:08:43.588+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [After]
-01T23:08:43.590+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Before]
-01T23:08:43.591+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Between]
-01T23:08:43.591+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Cookie]
-01T23:08:43.592+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Header]
-01T23:08:43.592+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Host]
-01T23:08:43.592+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Method]
-01T23:08:43.592+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Path]
-01T23:08:43.594+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Query]
-01T23:08:43.596+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [ReadBody]
-01T23:08:43.596+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [RemoteAddr]
-01T23:08:43.596+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [XForwardedRemoteAddr
-01T23:08:43.596+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [Weight]
-01T23:08:43.596+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.g.r.RouteDefinitionRouteLocator    : Loaded RoutePredicateFactory [CloudFoundryRouteSer
-01T23:08:44.310+05:30  INFO 11104 --- [GateWay] [           main] DiscoveryClientOptionalArgsConfiguration : Eureka HTTP Client uses RestTemplate.
-01T23:08:44.376+05:30  WARN 11104 --- [GateWay] [           main] iguration$LoadBalancerCaffeineWarnLogger : Spring Cloud LoadBalancer is currently working wit
-01T23:08:44.484+05:30  INFO 11104 --- [GateWay] [           main] o.s.c.n.eureka.InstanceInfoFactory       : Setting initial instance status as: STARTING
-01T23:08:44.567+05:30  INFO 11104 --- [GateWay] [           main] com.netflix.discovery.DiscoveryClient    : Initializing Eureka in region us-east-1
-01T23:08:44.582+05:30  INFO 11104 --- [GateWay] [           main] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
-01T23:08:44.615+05:30  INFO 11104 --- [GateWay] [           main] com.netflix.discovery.DiscoveryClient    : Disable delta property : false
-01T23:08:44.616+05:30  INFO 11104 --- [GateWay] [           main] com.netflix.discovery.DiscoveryClient    : Single vip registry refresh property : null
```

```
Updating port to 8761
Started ServiceRegistryEurekaServerApplication in 5.606 seconds (process running for 11.166)
```

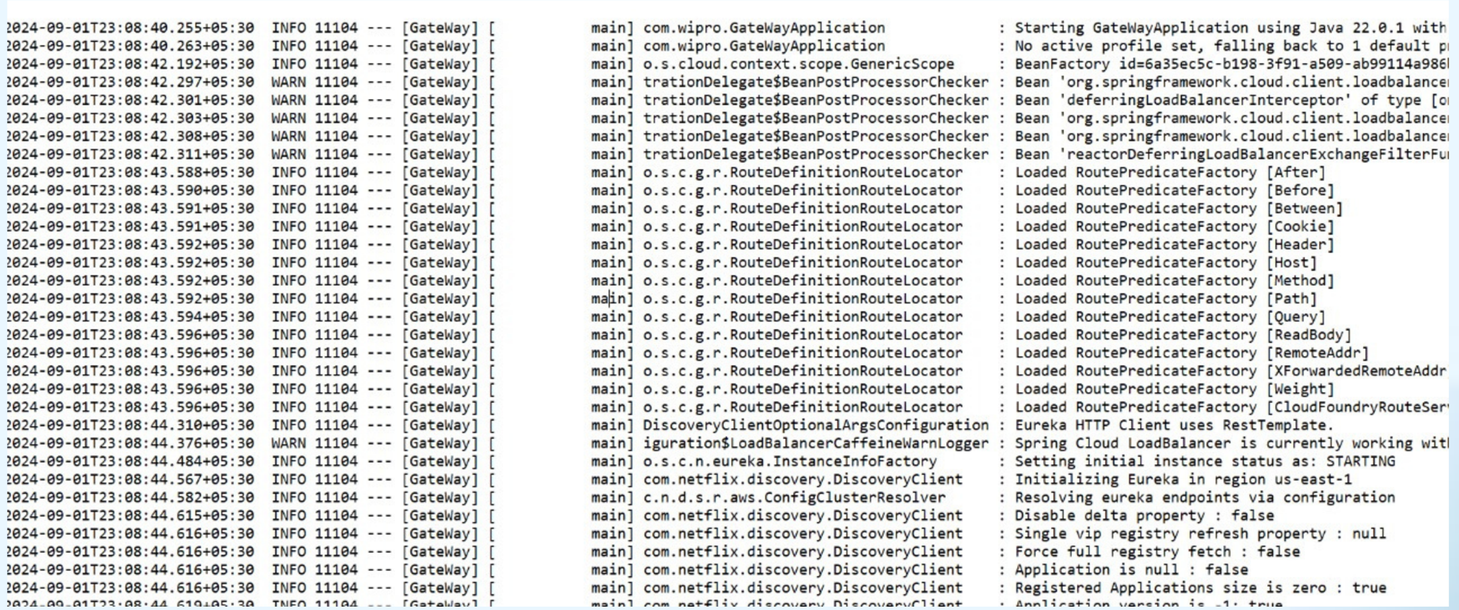
Eureka dashboard

Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:Auth-Service:9876
BOOKING-MICROSERVICE	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:Booking-Microservice:1010
CUSTOMER-MICROSERVICE	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:Customer-Microservice:2020
GATEWAY	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:GateWay:5050
SERVICE-MICROSERVICE	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:Service-Microservice:3030
STAFF-MICROSERVICE	n/a (1)	(1)	UP (1) - DESKTOP-VHAD0PQ:Staff-Microservice:4040

API GATEWAY

Spring Cloud Gateway:

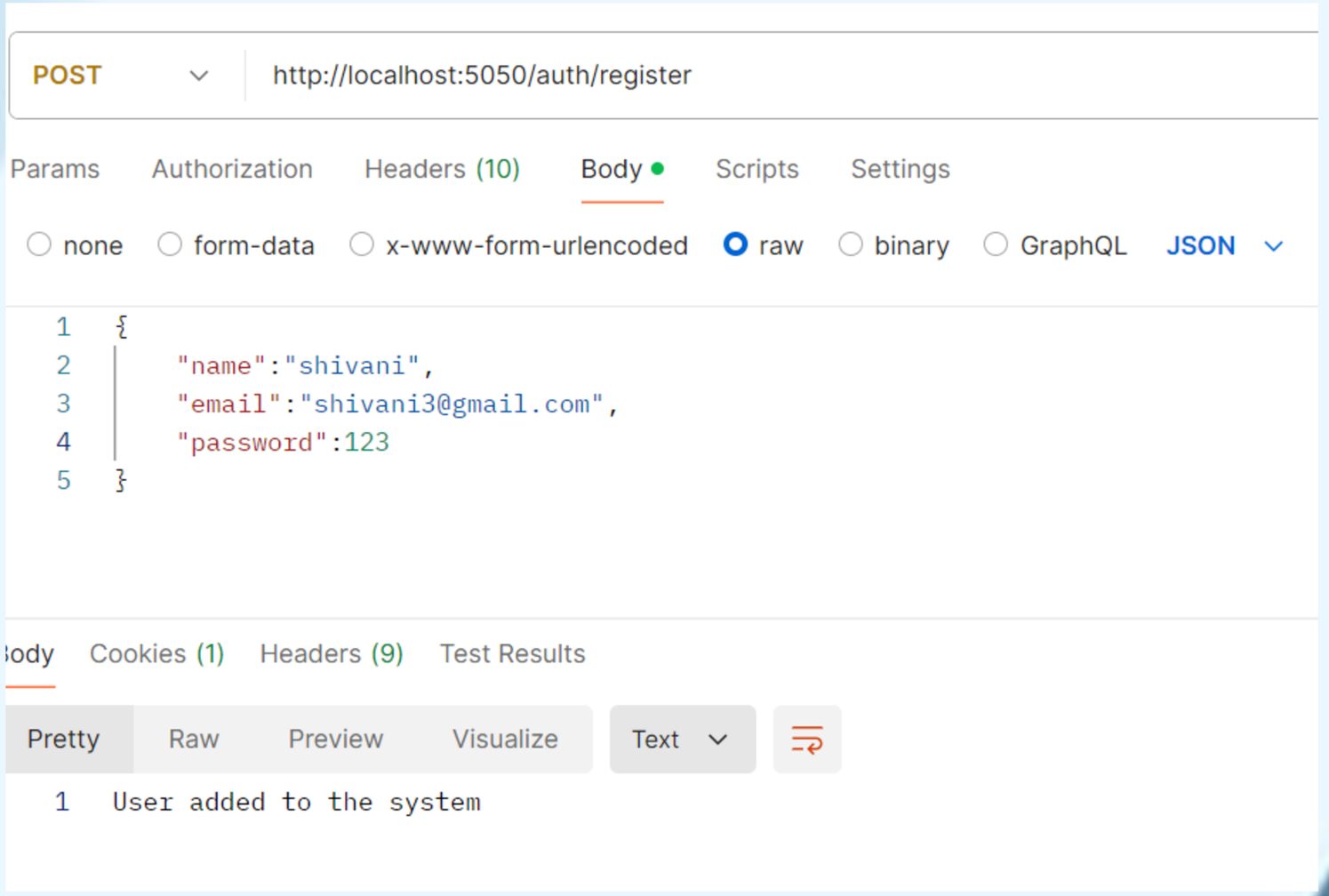
Spring Cloud Gateway is a powerful and flexible API Gateway built on top of the Spring Framework, designed to handle routing and management of microservices requests in a scalable and efficient manner. Its seamless integration with the Spring ecosystem, including Spring Boot and Spring Security, allows for simplified configuration and enhanced control over the routing and management of API calls, making it an ideal choice for managing complex microservices architectures.



Authentication Service

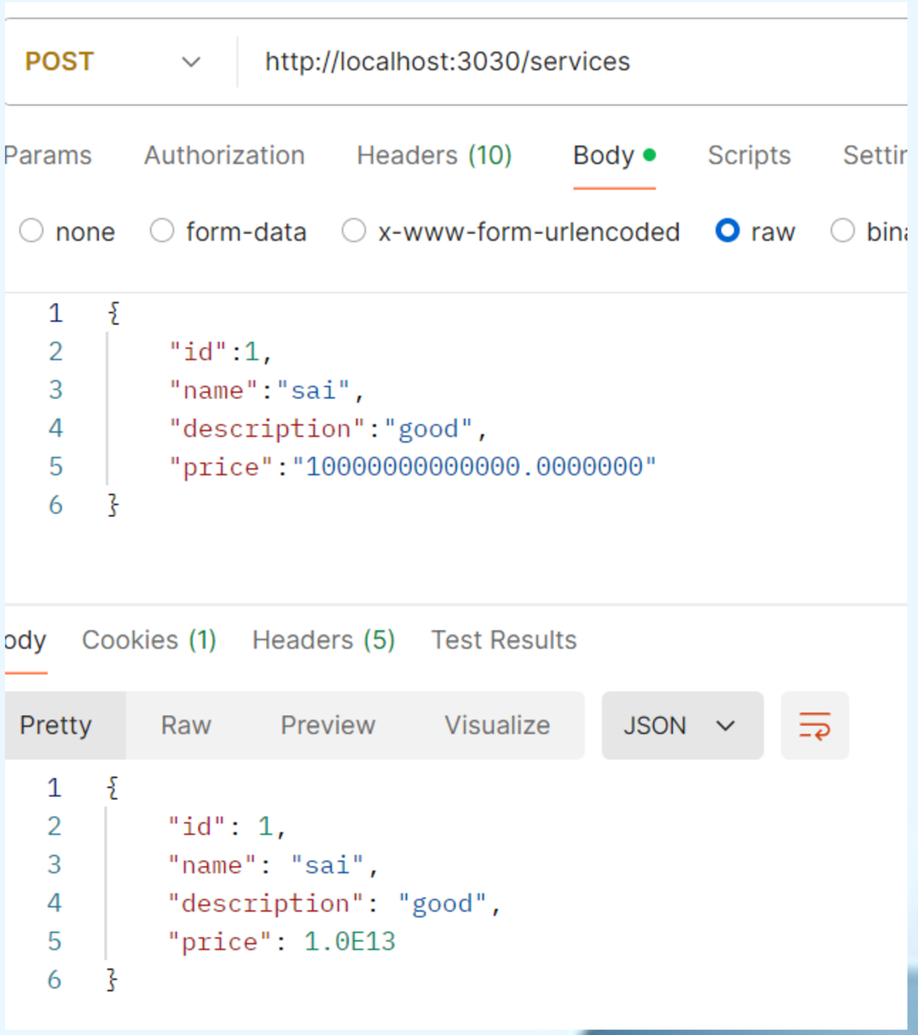
An Authentication Service is a critical component in securing modern applications and managing user access. It is responsible for verifying the identity of users or systems before granting access to resources or services. This ensures that only authorized users can access sensitive information or perform actions within an application, thereby protecting both data and system integrity.

Authentication Service OutPut



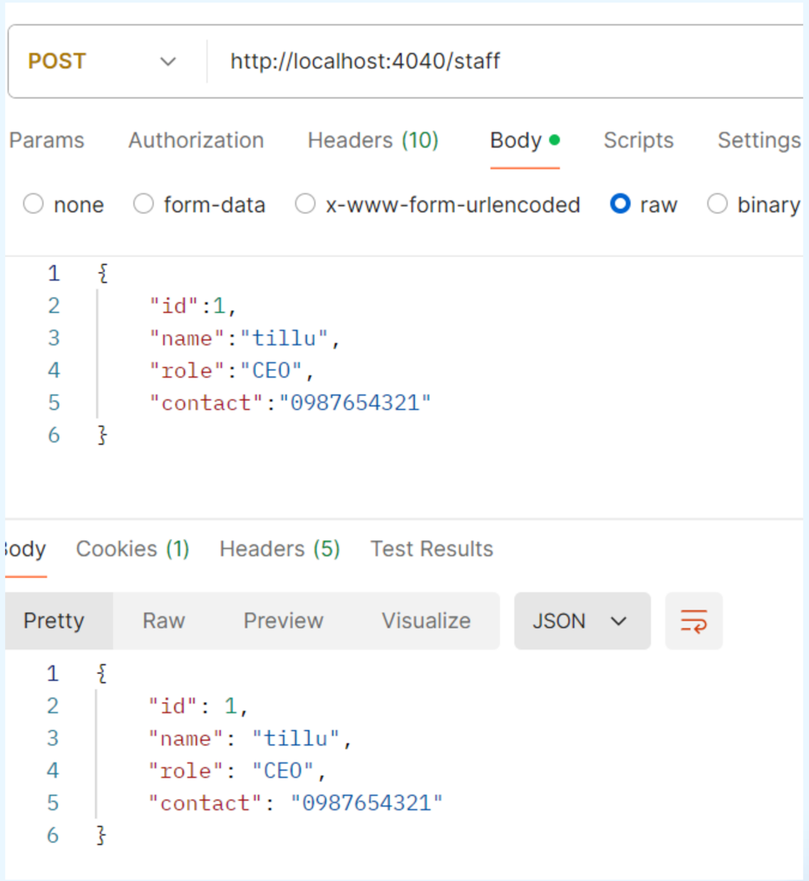
Service Management Microservice

It is a crucial element in a microservices architecture, responsible for overseeing and maintaining the lifecycle of other microservices. It manages service registration and discovery, allowing for dynamic and efficient location of services within the system. Service Management Microservice handles health monitoring and maintenance tasks, such as tracking service performance, detecting failures, and automating recovery or scaling operations.



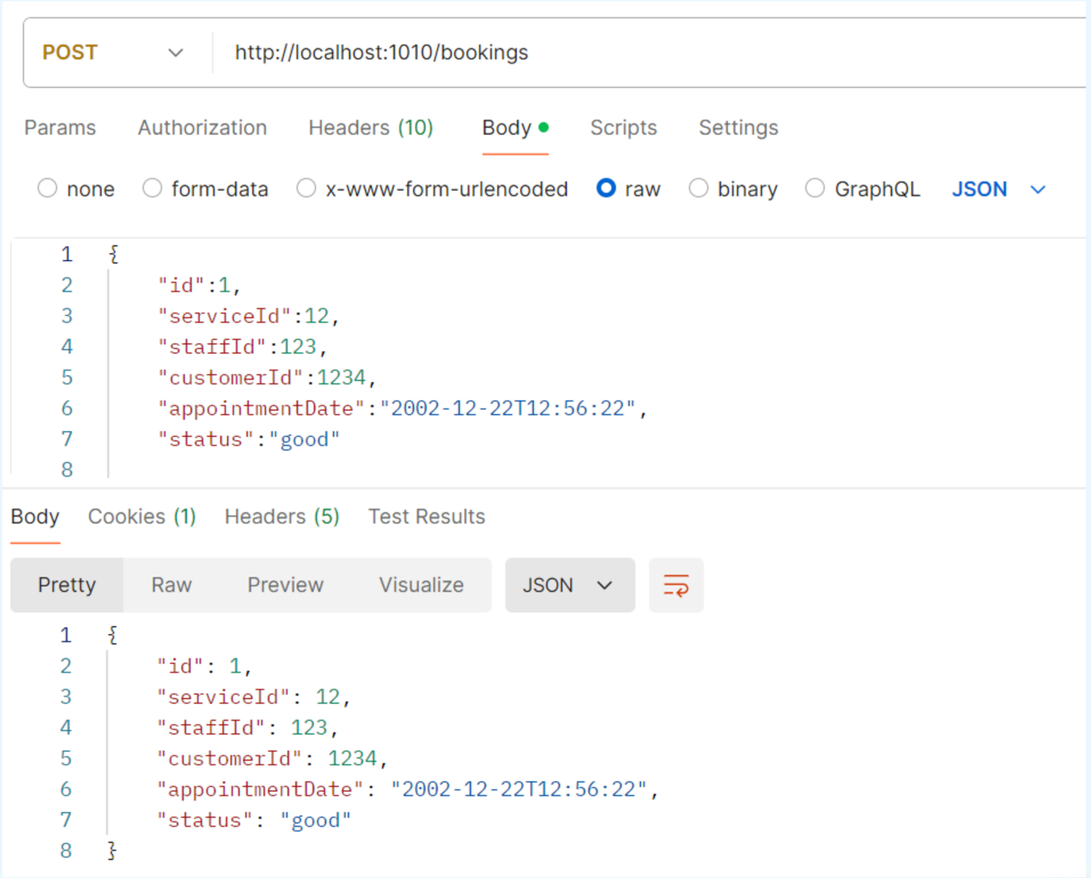
Staff Management Microservice

It is a specialized component in a microservices architecture designed to handle the administration of staff-related functions. It manages tasks such as staff onboarding, role assignments, scheduling, and performance tracking. This microservice ensures efficient management of employee data, facilitates seamless integration with other systems for resource planning, and supports operational workflows by providing up-to-date information on staff availability and assignments. By centralizing these functions, the Staff Management Microservice streamlines administrative processes and enhances overall organizational efficiency.



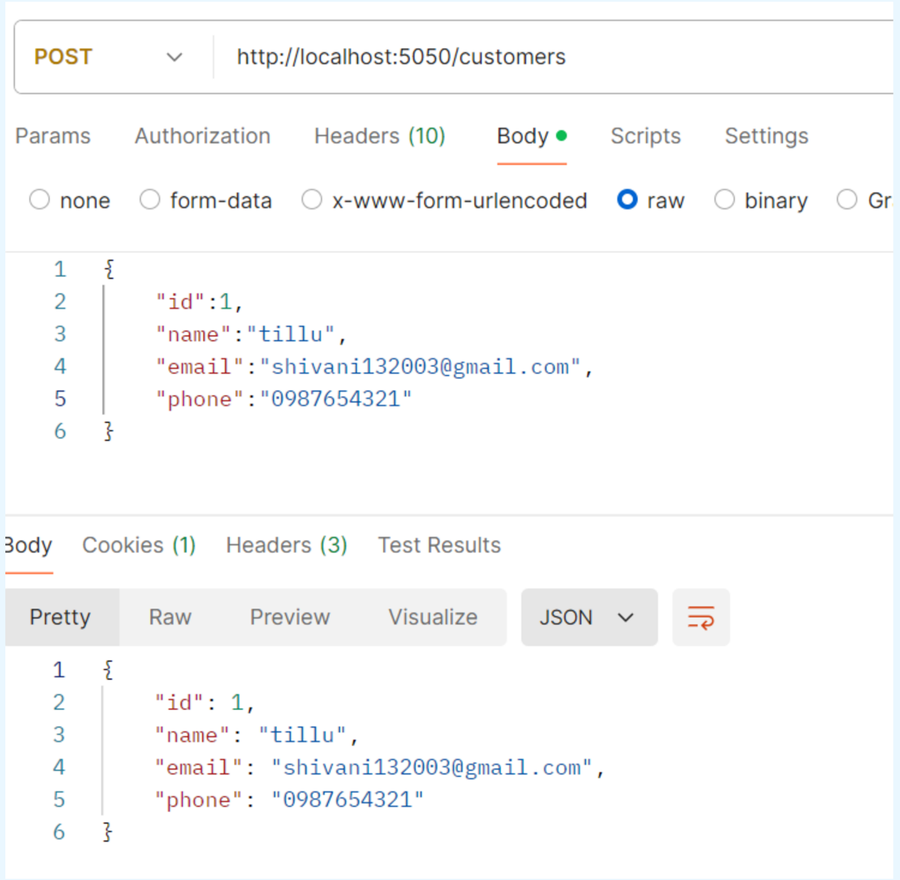
Booking Management Microservice

output :



Customer Management Microservice

The Customer Management Microservice is a key component in a microservices architecture focused on handling all aspects of customer data and interactions. It manages customer profiles, including personal information, preferences, and transaction history. This microservice facilitates functions such as registration, updates, and retrieval of customer details, enabling personalized service and communication.



Admin Module

The Admin Module is a crucial part of an application's backend, providing a comprehensive interface for administrators to manage and oversee various system functionalities. It typically includes tools for user management, system configuration, data analytics, and access control, allowing administrators to perform tasks such as monitoring system performance, managing user permissions, and configuring system settings. By centralizing these administrative functions, the Admin Module enhances operational efficiency, ensures proper oversight, and facilitates effective management of the application's overall infrastructure and user interactions.

User Module

The User Module is a fundamental component of an application designed to manage and enhance user interactions and experiences. It handles functions such as user registration, login, profile management, and personalized settings. This module ensures that users can create and update their accounts, access their personalized information, and interact with the application according to their preferences. By centralizing these user-centric features, the User Module improves usability, streamlines user management, and provides a seamless and tailored experience for each individual user.

Testing and Refinement

Testing and Refinement is a crucial phase in the development lifecycle focused on ensuring the quality and functionality of an application. It involves systematically evaluating the software through various testing methods, such as unit tests, integration tests, and user acceptance tests, to identify and address defects or issues. Refinement follows testing, where developers make necessary adjustments and optimizations based on feedback and test results. This iterative process helps improve the application's performance, usability, and reliability, ensuring it meets the required standards and delivers a seamless user experience.

conclusion

Developed a distributed online salon service application using microservice

THANK YOU

Shivani Budankayala
shivani132003@gmail.com