**API Gateway Pattern in Microservices 🡪” SINGLE ENTRY POINT FOR ALL CLIENT REQUESTS”**

* In a **microservices architecture**, managing multiple services can be complex, especially when it comes to **routing, authentication, load balancing, and security**.
* **The API Gateway Pattern solves this problem by acting as a single entry point for all client requests, directing them to the appropriate microservices.  
    
    
    
  What is the API Gateway Pattern?**
* The **API Gateway Pattern** is a **design pattern** used in microservices to provide a single entry point for all client requests. It acts as an **intermediary** between clients and microservices, handling **authentication, logging, request routing, load balancing, and rate limiting**.
* Instead of calling multiple microservices directly, clients communicate through the **API Gateway**, which then routes requests to the appropriate services.

**Key Concept:**

* A client sends a request to the **API Gateway**.
* The API Gateway forwards the request to the correct **backend service**.
* It handles authentication, rate limiting, request transformations, and security measures before forwarding the response to the client.

**How Does an API Gateway Work?**

**✅ Workflow**

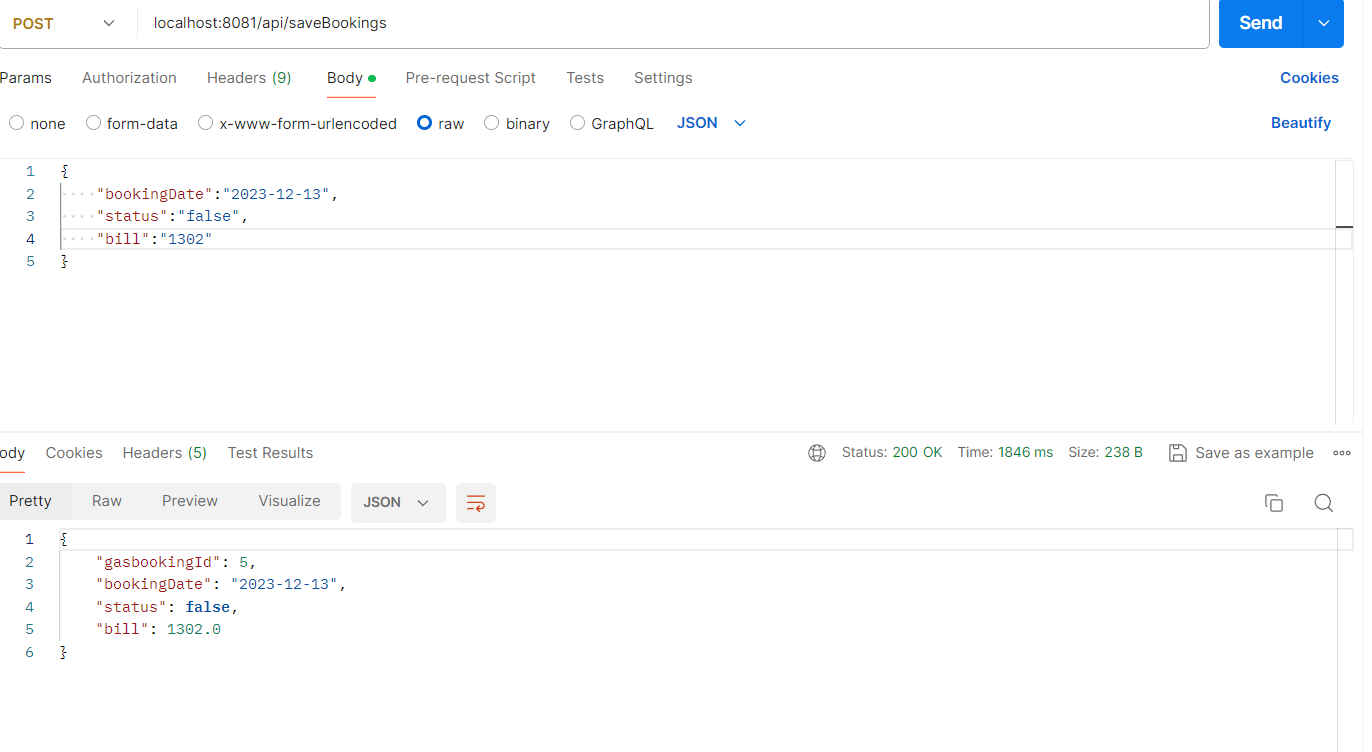
1. **Client sends a request** → The client application makes an HTTP request to the API Gateway.
2. **API Gateway processes the request** → It authenticates, logs, and applies rate limits if needed.
3. **Routing to backend services** → The gateway forwards the request to the correct microservice.
4. **Microservice processes the request** → The service retrieves data and sends a response.
5. **Response transformation** → The API Gateway modifies the response if required.
6. **Client receives the response** → The final response is sent back to the client.

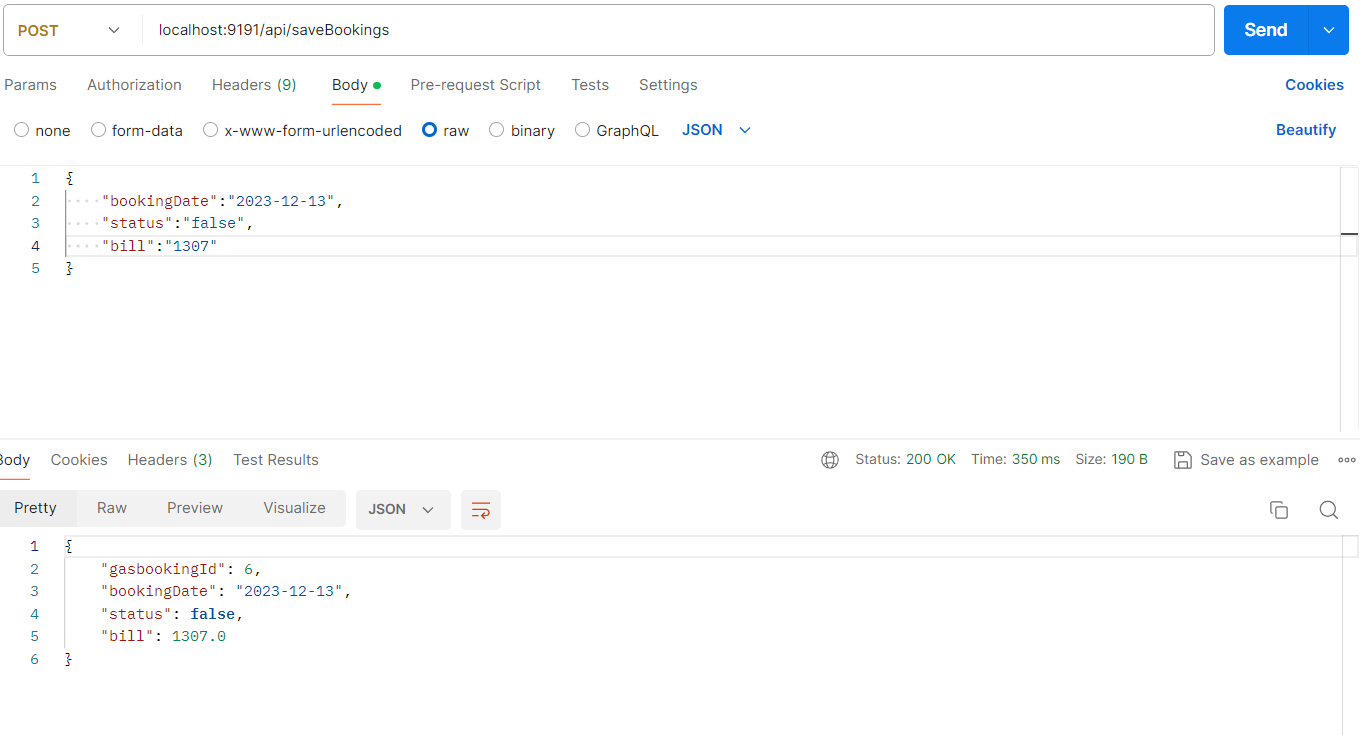
SAMPLE PROJECT   
1. Create Eureka server (mandatory to run all the microservices in this server) dependencies required   
 1. <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
 </dependency>  
  
@EnableEurekaServer annotation is used in EurekaServer   
  
application.properties  
   
 server.port=8761  
 spring.application.name=eureka-server  
 eureka.client.register-with-eureka=false  
 eureka.client.fetch-registry=false

Eureka server :local host :8761 --🡪default



8082

Example requests  
with port 8081 – for GASBOOKING SERVICE  


Now with API GATEWAY port , which is 9191  


So instead of using the original port ,wrt particular microservice the request will goes from apigateway microservice which is port 9191  
  
  
GASBOOKING-MICROSERVICE -8081  
CUSTOMER-MICROSERVICE -8080

API-GATEWAY – 9191  
EUREKA-SERVER -8761