Week 5: Microservices

|  |  |
| --- | --- |
| **Complexity** | Medium |
| **Technology** | Java |
| **Framework** | Spring Boot |
| **Repository support** | Spring Data JPA |
| **Database** | MYSQL|Oracle |
| **API development Approach** | Code First approach |

Task estimation

|  |  |
| --- | --- |
| Task | Day |
| Task 1 | Day 1(week5) |
| Task 2 | Day 2(week5) |
| Task 3 | Day 3(week5) |
| Task 4 | Day 4(week5) |
| Task 5 | Day 5(week5) |

Use case:

Consider a microservices called as Product Service, that would do following tasks

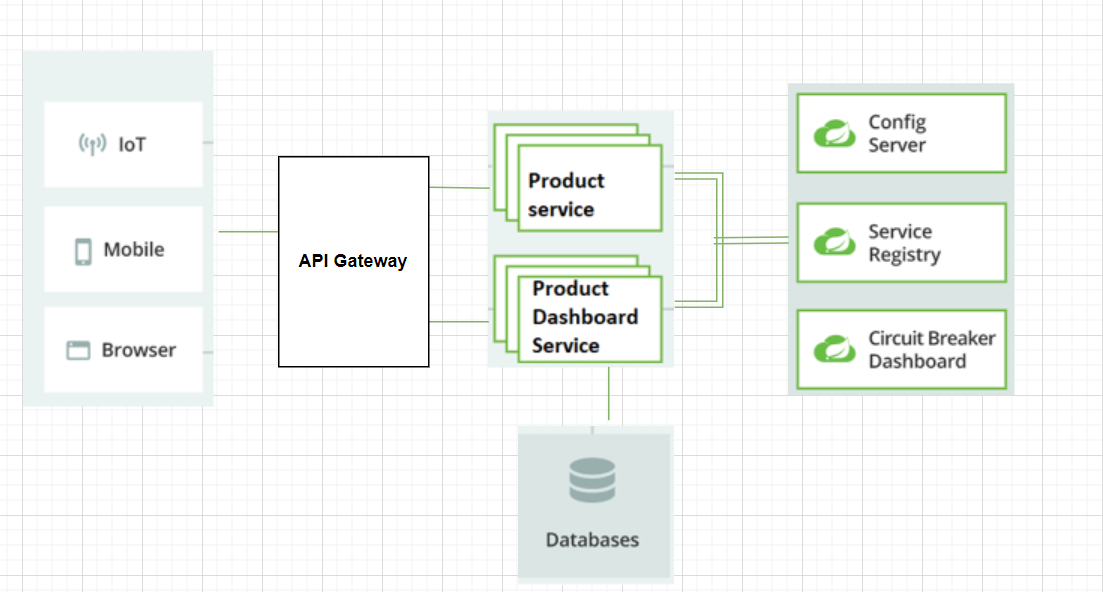
1. create a product
2. list all the product
3. list a product by id.

Consider the below entity for the use case. Add additional attributes or entities if required.

|  |  |
| --- | --- |
| Product | Datatype |
| id | int(pk) |
| productName | string |
| quantity | int |
| price | double |
| description | string |

Create an another microservice called as Product dashboard that would call the Product service via Eureka Server. The communication between the 2 microservice would be synchronous.

Consider the below diagram for architecture ref.



Task 1:

1. Create Product service with following endpoints
2. Ensure to test the service and handle exception whenever required.
3. Add sample data using the POST request.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rest endpoint | HTTP Verb/Method | Request Format | Response Format | Http Status code |
| <http://host/basePath/v1/products> | **POST** | application/json | application/json | **201** |
| <http://host/basePath/v1/products> | **GET** |  | application/json | **200** |
| <http://host/basePath/v1/products>/{prodId} | **GET** |  | application/json | **200** |

Sample POST request

{

"productName": "Book",

"price":95.5,

"quantity":2,

"description":"Camel Coloring Book"

}

Sample GET request

http://host/basePath/v1/products

{

"id": 1,

"productName": "Book",

"price": 95.5,

"quantity": 2,

"description": "Camel Coloring Book"

}

Sample GET request

http://host/basePath/v1/products/1

{

"id": 1,

"productName": "Book",

"price": 95.5,

"quantity": 2,

"description": "Camel Coloring Book"

}

Task 2:

1. Create a **Netflix Eureka service registry** and then Product Service registers itself with the registry and uses it to resolve its own host.
2. Create a **Config Server** to external the configuration of Product Service.

Task 3:

1. Create Product Dashboard Service that would call the <http://host/basePath/v1/products> GET request and list out all the products available.
2. Consider the below endpoint available in Product Service which the Product Dashboard has to consume GET : <http://host/basePath/v1/products>
3. Points to consider:

Use **Feign Client** to consume the service instead of RestTemplate.

Register the Product Dashboard Service with the registry as well.

1. Externalize the configuration and let the microservice use the configuration from config server

Task 4:

1. In case if the Product Service goes down it should not lead to failure of Product Dashboard service. Configure the **circuit breaker** to achieve the same.
2. Create a fallback method called “reliable in Product Dashboard service. If the remote API does not respond in time, the method "reliable" will be called and that will serve the request. In the fallback method, you can return either a default output or even call some other remote or local API to serve the request.
3. Test the scenario by stopping the Product Service and see if the request is routed to the fallback method.

Task 5:

1. Create an edge service that would act as the entry point for all request.
2. Use **Zuul API Gateway** for the same.
3. Configure the routes so that the request is routed properly
4. Test the functionality of the application.