

Assignment 4

Developing a toy microservices system

目录

1 完成情况	2
2 系统整体设计	3
2.1 应用场景.....	3
2.2 接口设计.....	3
网关端口：8088	3
2.2.1 oauth-service 身份验证服务	4
2.2.2 warehouse-service 仓储服务	4
2.2.3 logistics-service 物流服务	7
2.2.4 payment-service 支付服务	9
2.2.5 crm-service CRM	11
2.3 数据库设计.....	12
3 测试截图	14
4 功能实现	17
4.1 服务发现.....	17
4.2 断路器实现.....	17
4.3 Oauth2 授权.....	19
4.5 通过网关向外部用户公开 API	20
4.6 使用 Spring cloud 进行集中配置和跟踪	21
4.6.1 集中配置.....	21
4.6.2 跟踪.....	23

1 完成情况

Basic Requirements:

- ✓ Develop a toy microservices system using Spring cloud infrastructures.
- ✓ Service discovery with Eureka/Alibaba Nacos is necessary

Credit Implementations:

- ✓ Circuit breaker implementation with Resilience4j or Hystrix.
- ✓ OAuth2 authorization server integrated.
- ✓ Expose API to external users with Gateway
- ✓ Centralized configuration and tracking with Spring cloud config server and sleuth.

本次项目旨在开发一个包含多个微服务的系统，包括 CRM 服务、物流服务、自动化仓库服务和支付服务等。例如，当客户下单要求存储一批货物到仓库时，CRM 服务将调用相应的物流服务、仓库服务和支付服务来完成交易。为了开发该系统，我们将添加包括服务发现、网关和 OAuth 服务器在内的微服务基础设施。

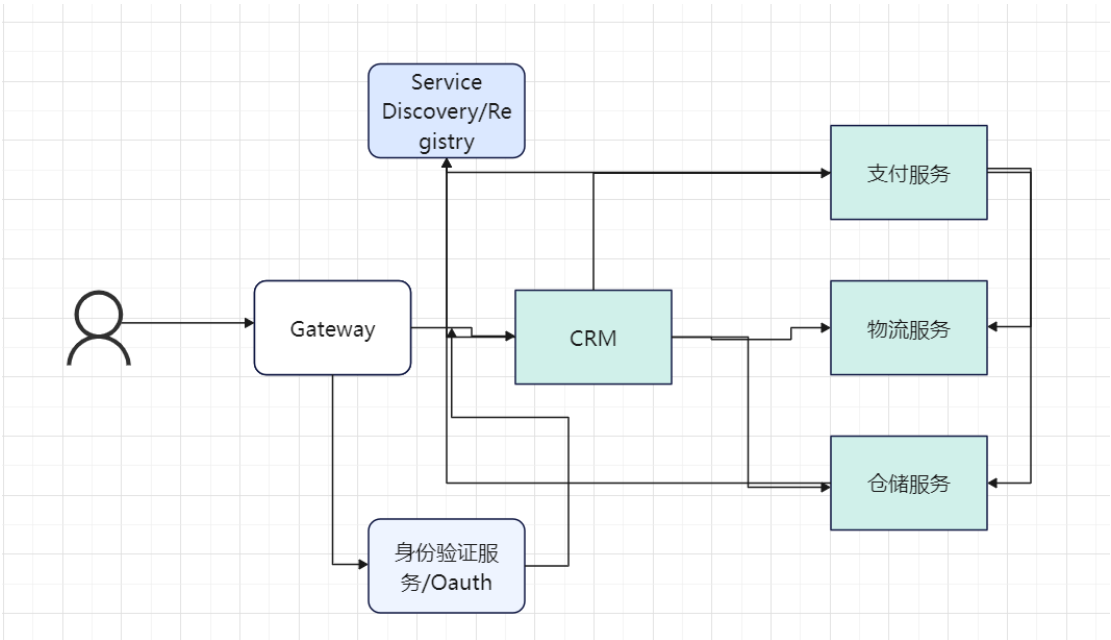
我们使用 Java 语言结合 Spring Cloud 框架来完成微服务系统的开发，通过采用 Alibaba Nacos 实现服务的自动发现，确保了服务间的高效通信和动态管理。系统中还集成了 Spring Cloud Gateway 来管理外部 API 的暴露，保证了服务的统一入口和安全控制。此外，我们选择了 OAuth 2.0 作为授权服务器，来强化系统的安全性，确保每个请求都经过身份验证和授权。

为了提高系统的可用性和容错性，我们实现了 Resilience4j 的断路器模式，防止服务间调用时可能出现的级联故障。同时，为了方便服务配置的管理和变更跟踪，我们引入了 Spring Cloud Config Server 进行集中配置管理，以及使用 Spring Cloud Sleuth 进行调用链跟踪，增强了系统的可监控性和维护性。

在开发过程中，我们使用 Spring Boot 的测试框架进行单元测试和集成测试，确保每个服务的稳定性和可靠性。通过这些措施，我们不仅提升了系统的性能和用户体验，还加深了我们对微服务架构和 Spring Cloud 技术栈的理解和应用。

2 系统整体设计

2.1 应用场景



本玩偶微服务系统展示了典型的微服务分布式处理流程，其中每个服务都是松耦合且独立部署的，通过服务发现机制互相协作完成复杂的业务需求。

用户通过网关发送请求，请求首先经过 OAuth 服务进行用户认证和授权。认证通过后，请求根据需要被路由到 CRM 服务。CRM 服务接收到用户的服务订单请求后，首先调用仓库服务确认商品库存。接下来，根据订单需求调用物流服务安排商品配送。最后，调用支付服务处理订单支付。

每个服务处理完毕后，结果返回给 CRM 服务，CRM 服务再将处理结果汇总后通过网关返回给用户。

2.2 接口设计

网关端口：8088

参数名	字段值
baseUrl	http://localhost:8088

2.2.1 oauth-service 身份验证服务

- 1. 端口：8084
- 2. 用户登录

POST /login

请求:

方法	路径	参数	描述
POST	/login	LoginRequest 对象	用户登录，验证凭证

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
access_token	String		FALSE	为用户生成的 JWT，用户凭此访问受保护的资源
token_type	String	Bearer	FALSE	令牌的类型，此处为 Bearer
expires_in	String	3600	FALSE	令牌的有效期，此处以秒为单位，通常设置为 3600 秒（1 小时）

2.2.2 warehouse-service 仓储服务

- 1. 端口：8080
- 2. 获取所有商品

GET /items

请求:

方法	URL	参数	描述
GET	/items	无	获取所有商品的列表

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

3. 创建新商品

POST /items

请求:

方法	URL	参数	描述
POST	/items	Item 对象	创建新的商品记录

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

4. 删除商品

DELETE /items/{id}

请求:

方法	URL	参数	描述
DELETE	/items/{id}	id (路径变量)	删除指定 ID 的商品记录

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

5. 更新商品信息

PUT /items/{id}

请求:

方法	URL	参数	描述
PUT	/items/{id}	id (路径变量), Item 对象	更新指定 ID 的商品信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

6. 根据 ID 获取商品

GET /items/{id}

请求:

方法	URL	参数	描述
GET	/items/{id}	id (路径变量)	获取指定 ID 的商品详细信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

7. 根据订单 ID 获取商品信息

GET /items/order/{orderId}

请求:

方法	URL	参数	描述
GET	/items/order/{orderId}	id (路径变量)	获取指定订单 ID 的商品详细信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

2.2.3 logistics-service 物流服务

- 1. 端口：8081
- 2. 获取所有物流信息

GET /logistics

请求:

方法	URL	参数	描述
GET	/logistics	无	获取所有物流信息

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

- 3. 创建新物流信息

POST /logistics

请求:

方法	URL	参数	描述
POST	/logistics	Logistics 对象	创建新的物流记录

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

4. 删除商品

DELETE /logistics/{id}

请求:

方法	URL	参数	描述
DELETE	/logistics/{id}	id (路径变量)	删除指定 ID 的物流记录

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

5. 更新商品信息

PUT /logistics/{id}

请求:

方法	URL	参数	描述
PUT	/logistics/{id}	id (路径变量), Logistics 对象	更新指定 ID 的物流信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

6. 根据 ID 获取商品

GET /logistics/{id}

请求:

方法	URL	参数	描述
GET	/logistics/{id}	id (路径变量)	获取指定 ID 的物流详细信息

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

7. 根据订单 ID 获取物流信息

GET /logistics/order/{orderId}

请求:

方法	URL	参数	描述
GET	/logistics/order/{orderId}	id (路径变量)	获取指定订单 ID 的物流详细信息

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

2.2.4 payment-service 支付服务

1. 端口：8082

2. 创建新支付信息

POST /payments

请求:

方法	URL	参数	描述
POST	/payments	Payment 对象	创建新的支付记录

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	

flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

3. 根据订单 ID 获取支付信息

GET /payments/order/{orderId}

请求:

方法	URL	参数	描述
GET	/payments/order/{orderId}	id (路径变量)	获取指定订单 ID 的支付详细信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

4. 更新支付信息

PUT /payments/{id}

请求:

方法	URL	参数	描述
PUT	/payments/{id}	id (路径变量), Payment 对象	更新指定 ID 的支付信息

响应体

200 响应数据格式: JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

2.2.5 crm-service CRM

1. 端口：8085

2. 创建新订单信息

POST /crm/order

请求:

方法	URL	参数	描述
POST	/crm/order	order 对象	创建新的订单记录

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

3. 根据订单 ID 获取支付信息

GET /crm/order/{orderId}

请求:

方法	URL	参数	描述
GET	/crm/order/{orderId}	id (路径变量)	获取指定订单 ID 的支付详细信息

响应体

200 响应数据格式：JSON

参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、404 and so on

4. 更新支付信息

PUT /order/{orderId}/item

请求:

方法	URL	参数	描述
PUT	//order/{orderId}/item	id (路径变量), Order 对象	更新指定 orderId 的状态信息

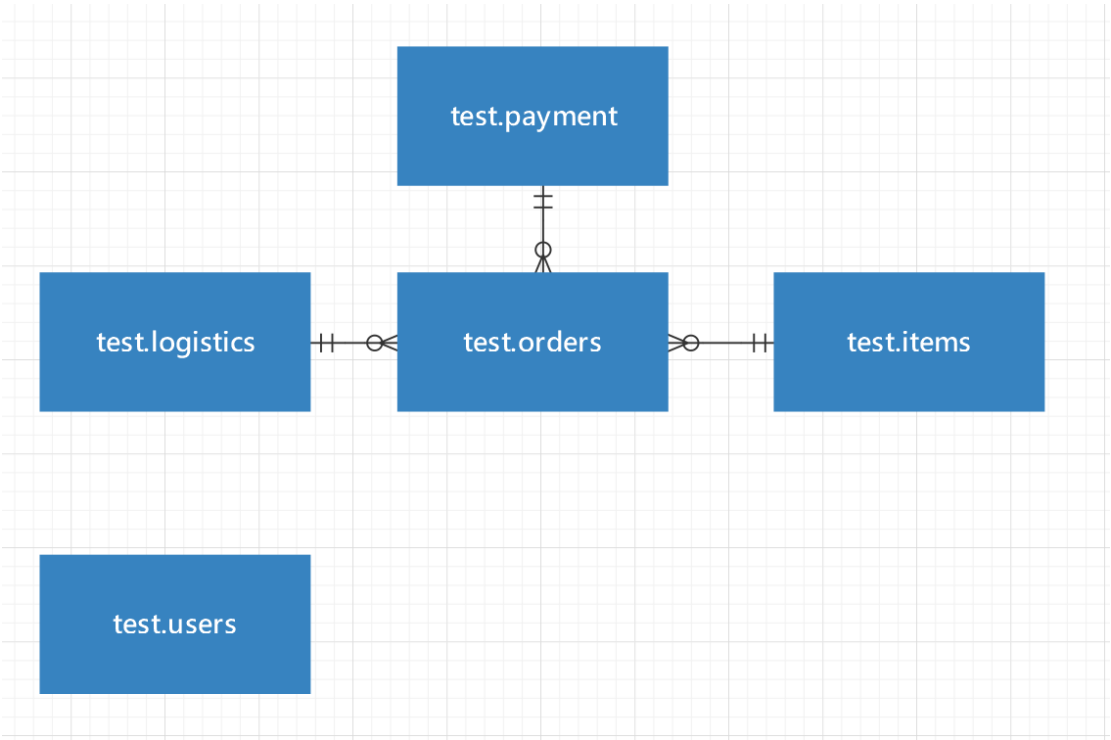
响应体

200 响应数据格式: JSON

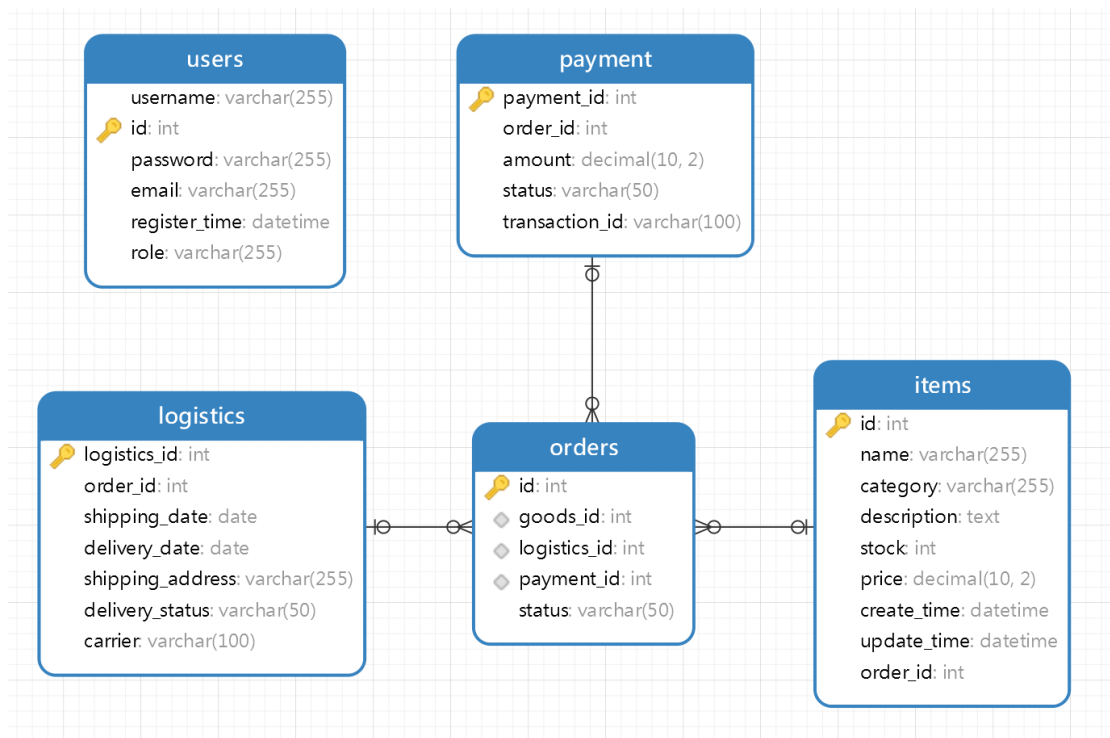
参数名称	类型	默认值	不为空	描述
data	object		FALSE	
flag	boolean		FALSE	
msg	string		FALSE	
statusCode	int32	200	FALSE	the http code 200、 404 and so on

2.3 数据库设计

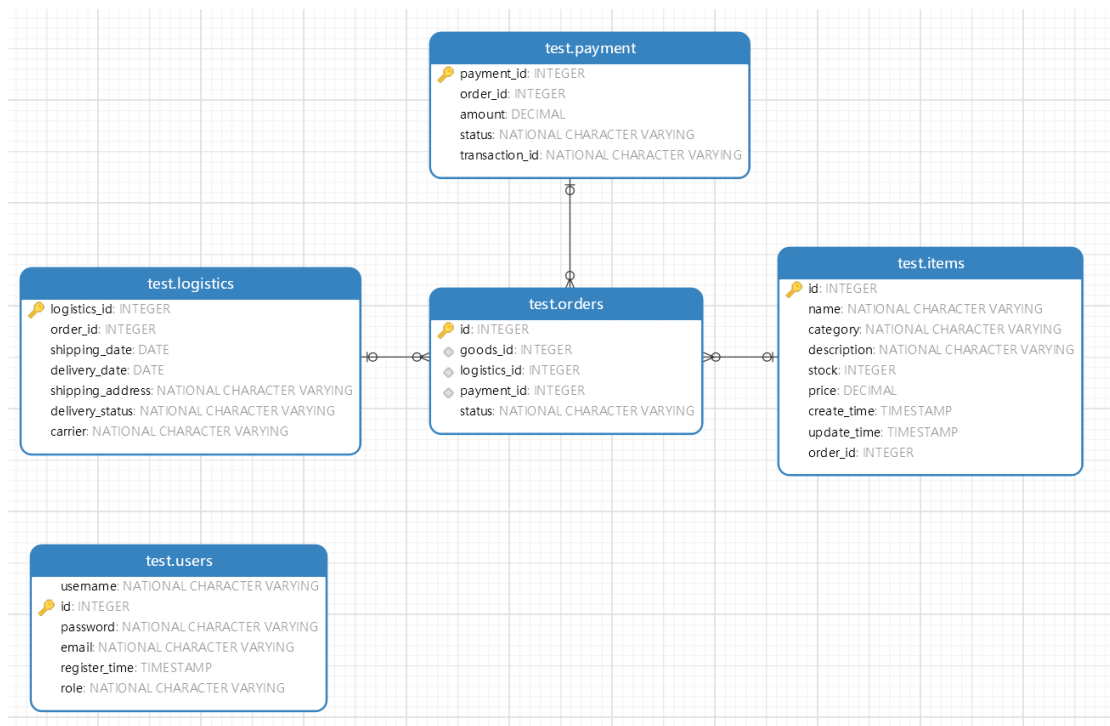
(1) 概念模型:



(2) 物理模型:



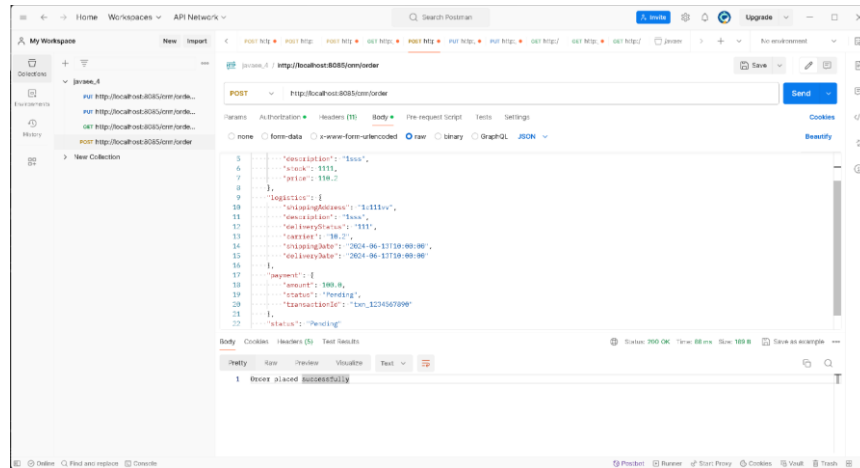
(3) 逻辑模型:



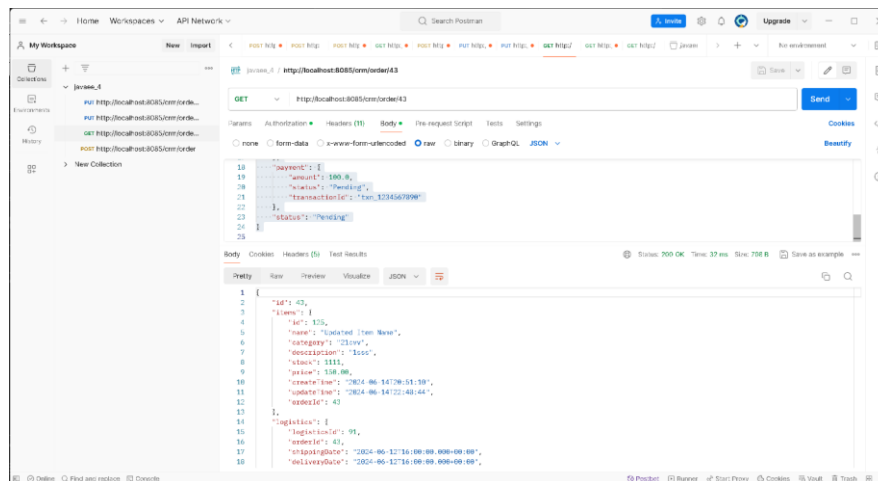
3 测试截图

依次测试增加订单，查看订单信息，修改订单状态，修改仓储/物流/支付状态：

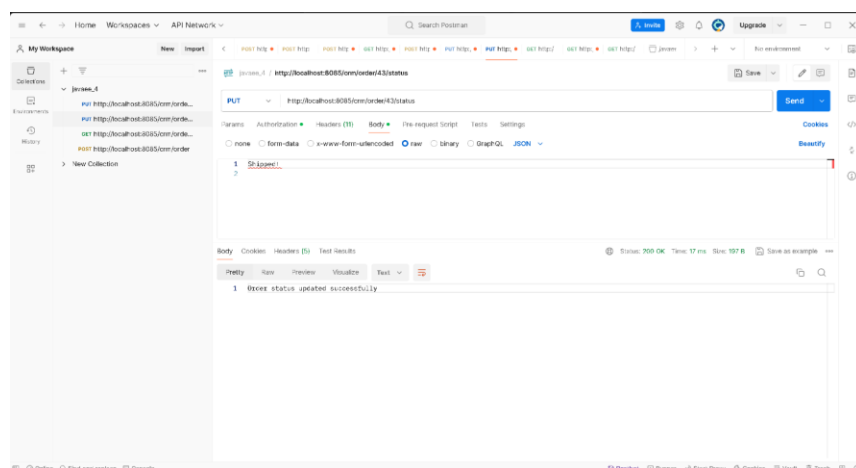
(1) 增加订单：



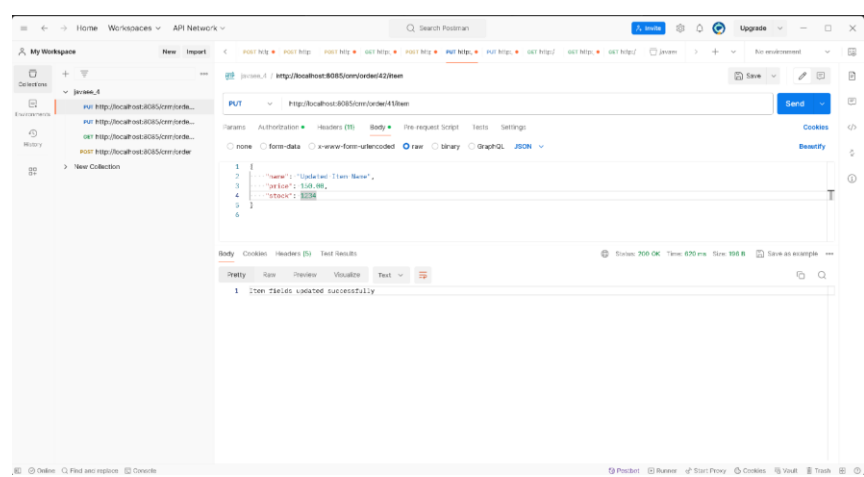
(2) 查看订单和其余三个表的具体信息：



(3) 更新订单状态：



(4) 通过订单 id 更改 item 表的信息:



(5) 增加数据后的 logistics 表

logistics_id	order_id	shipping_date	delivery_date	shipping_address	delivery_status	carrier
72	2	2024-06-13	2024-06-13	1cw	111	10.2
73	2	2024-06-13	2024-06-13	1cw	111	10.2
74	2	2024-06-13	2024-06-13	1c111w	111	10.2
75	2	2024-06-13	2024-06-13	1c111w	111	10.2
76	2	2024-06-13	2024-06-13	1c111w	111	10.2
77	2	2024-06-13	2024-06-13	1c111w	111	10.2
78	2	2024-06-13	2024-06-13	1c111w	111	10.2
79	2	2024-06-13	2024-06-13	1c111w	111	10.2
80	2	2024-06-13	2024-06-13	1c111w	111	10.2
81	2	2024-06-13	2024-06-13	1c111w	111	10.2
82	2	2024-06-13	2024-06-13	1c111w	111	10.2
83	0	2024-06-13	2024-06-13	1c111w	111	10.2
84	2	2024-06-13	2024-06-13	1c111w	111	10.2
87	2	2024-06-13	2024-06-13	1c111w	111	10.2
88	40	2024-06-13	2024-06-13	1c111w	111	10.2
89	41	2024-06-13	2024-06-13	1c111w	111	10.2
90	42	2024-06-13	2024-06-13	1c111w	111	10.2
91	43	2024-06-13	2024-06-13	1c111w	111	10.2
92	2	2024-06-13	2024-06-13	1c111w	111	10.2
93	2	2024-06-13	2024-06-13	1c111w	111	10.2
94	0	2024-06-13	2024-06-13	1c111w	111	10.2
95	0	2024-06-13	2024-06-13	1c111w	111	10.2
96	0	2024-06-13	2024-06-13	1c111w	111	10.2
97	49	2024-06-13	2024-06-13	1c111w	111	10.2
98	50	2024-06-13	2024-06-13	1c111w	111	10.2
99	51	2024-06-13	2024-06-13	1c111w	111	10.2

(6) 增加数据后的 orders 表

id	goods_id	logistics_id	payment_id	status
24	106	72	43	Pending
25	107	73	44	Pending
26	108	74	45	Pending
27	109	75	46	Pending
28	110	76	47	Pending
29	111	77	48	Pending
30	112	78	49	Pending
31	113	79	50	Pending
32	114	80	51	Pending
33	115	81	52	Pending
34	116	82	53	Pending
35	117	83	54	Pending
36	118	84	55	Pending
39	121	87	58	Pending
40	122	88	59	Pending
41	123	89	60	Pending
42	124	90	61	Pending
43	125	91	62	Shipped?
44	126	92	63	Pending
45	127	93	64	Pending
46	128	94	65	Pending
47	129	95	66	Pending
48	130	96	67	Pending
49	131	97	68	Pending
50	132	98	69	Pending
51	133	99	70	Pending

(7) 增加数据后的 items 表

id	name	category	description	stock	price	create_time	update_time	order_id
106	213ee	21cvw	1sss	1111	110.20	2024-06-14 18:11:30	2024-06-14 18:11:30	2
107	213ee	21cvw	1sss	1111	110.20	2024-06-14 18:12:27	2024-06-14 18:12:27	2
108	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:14:00	2024-06-14 18:14:00	2
109	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:16:00	2024-06-14 18:16:00	2
110	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:18:26	2024-06-14 18:18:26	2
111	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:18:40	2024-06-14 18:18:40	2
112	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:20:12	2024-06-14 18:20:12	2
113	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:26:13	2024-06-14 18:26:13	2
114	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:31:05	2024-06-14 18:31:05	2
115	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:37:18	2024-06-14 18:37:18	2
116	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:38:43	2024-06-14 18:38:43	2
117	21311ee	21cvw	1sss	1111	110.20	2024-06-14 18:41:11	2024-06-14 18:41:11	0
118	21311ee	21cvw	1sss	1111	110.20	2024-06-14 20:05:45	2024-06-14 20:05:45	2
121	21311ee	21cvw	1sss	1111	110.20	2024-06-14 20:26:02	2024-06-14 20:26:02	2
122	21311ee	21cvw	1sss	1111	110.20	2024-06-14 20:26:56	2024-06-14 20:26:56	40
123	Updated II	21cvw	1sss	1234	150.00	2024-06-14 20:31:17	2024-06-15 23:42:39	41
124	Updated II	21cvw	1sss	1111	150.00	2024-06-14 20:31:55	2024-06-14 22:51:10	42
125	Updated II	21cvw	1sss	1111	150.00	2024-06-14 20:51:10	2024-06-14 22:48:44	43
126	21311ee	21cvw	1sss	1111	110.20	2024-06-15 20:39:52	2024-06-15 20:39:52	0
127	21311ee	21cvw	1sss	1111	110.20	2024-06-15 20:41:23	2024-06-15 20:41:23	0
128	21311ee	21cvw	1sss	1111	110.20	2024-06-15 20:43:43	2024-06-15 20:43:43	0
129	21311ee	21cvw	1sss	1111	110.20	2024-06-15 20:46:43	2024-06-15 20:46:43	0
130	321311ee	21cvw	1sss	1111	110.20	2024-06-15 20:48:24	2024-06-15 20:48:24	0
131	321311ee	21cvw	1sss	1111	110.20	2024-06-15 20:50:55	2024-06-15 20:50:55	49
132	321311ee	21cvw	1sss	1111	110.20	2024-06-15 23:41:56	2024-06-15 23:41:56	50
133	321311ee	21cvw	1sss	1111	110.20	2024-06-16 00:28:40	2024-06-16 00:28:40	51

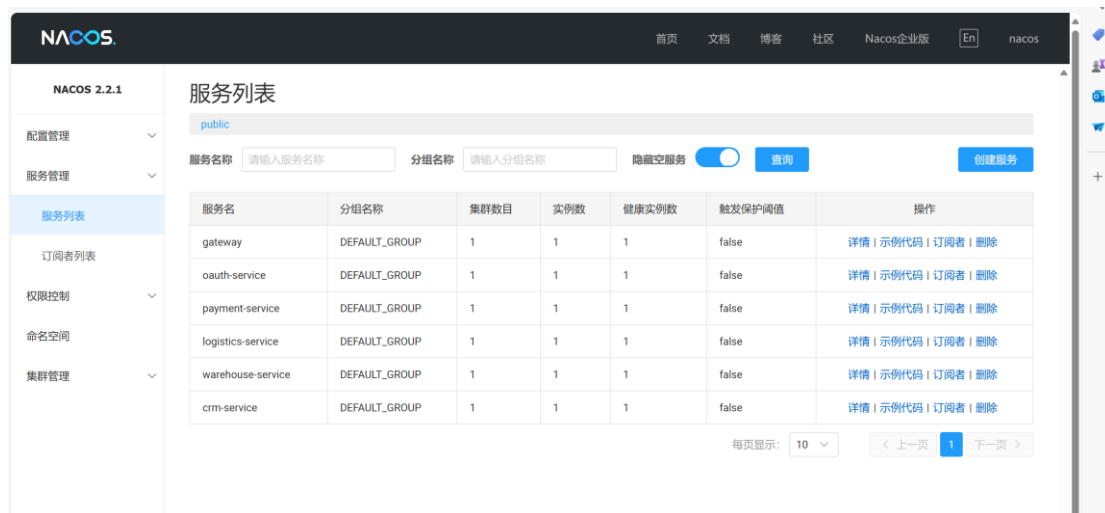
(8) 增加数据后的 payment 表

payment_id	order_id	amount	status	transaction_id
43	1	100.00	Pending	txn_1234567890
44	1	100.00	Pending	txn_1234567890
45	1	100.00	Pending	txn_1234567890
46	1	100.00	Pending	txn_1234567890
47	1	100.00	Pending	txn_1234567890
48	1	100.00	Pending	txn_1234567890
49	1	100.00	Pending	txn_1234567890
50	1	100.00	Pending	txn_1234567890
51	1	100.00	Pending	txn_1234567890
52	1	100.00	Pending	txn_1234567890
53	1	100.00	Pending	txn_1234567890
54	(Null)	100.00	Pending	txn_1234567890
55	1	100.00	Pending	txn_1234567890
58	1	100.00	Pending	txn_1234567890
59	40	100.00	Pending	txn_1234567890
60	41	100.00	Pending	txn_1234567890
61	42	100.00	Pending	txn_1234567890
62	43	100.00	Pending	txn_1234567890
63	(Null)	100.00	Pending	txn_1234567890
64	(Null)	100.00	Pending	txn_1234567890
65	(Null)	100.00	Pending	txn_1234567890
66	(Null)	100.00	Pending	txn_1234567890
67	(Null)	100.00	Pending	txn_1234567890
68	49	100.00	Pending	txn_1234567890
69	50	100.00	Pending	txn_1234567890
70	51	100.00	Pending	txn_1234567890

4 功能实现

4.1 服务发现

使用 Alibaba Nacos 实现服务发现。允许服务实例动态注册自己并被其他服务发现。



4.2 断路器实现

使用 Resilience4j 实现熔断器，提高系统的容错能力。

```
resilience4j:
  circuitbreaker:
    configs:
      default:
        slidingWindowSize: 10
        permittedNumberOfCallsInHalfOpenState: 10
        waitDurationInOpenState: 60s
        failureRateThreshold: 10
        eventConsumerBufferSize: 10
        recordExceptions:
          - java.io.IOException
          - java.util.concurrent.TimeoutException
          - java.net.ConnectException # 添加服务连接异常
    instances:
      itemService:
        baseConfig: default
      logisticsService:
        baseConfig: default
      paymentService:
        baseConfig: default
```

```

@PostMapping("/order")
@CircuitBreaker(name = "paymentService", fallbackMethod = "handlePaymentServiceFallback")
public ResponseEntity<String> createOrder(@RequestBody Order order) {

private ResponseEntity<String> handlePaymentServiceFallback(Order order, Throwable t) {
    return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body("Failed to process payment: " + t.getMessage());
}

```

人为设定触发熔断的条件，用于模拟异常情况。通过检查传入的 `updates` 字典中是否包含 `triggerException` 键，并且对应值为 `true`，来决定是否抛出一个 `IOException`。

```

@PutMapping("/order/{orderId}/item")
@CircuitBreaker(name = "itemService", fallbackMethod = "handleItemServiceFallback")
public ResponseEntity<String> updateItemFields(@PathVariable Integer orderId, @RequestBody Map<String, Object> updates) throws IOException {

no usages
private ResponseEntity<String> handleItemServiceFallback(Integer orderId, Map<String, Object> updates, Throwable t) {
    logger.error("Item service is unavailable, fallback triggered", t);
    return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body("Item service is temporarily unavailable. Please try again later.");
}

```

当传入的 `updates` 字典中包含 `triggerException` 键，并且对应值为 `true` 时，熔断器打开。

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8088/crm/order`
- Method:** `PUT`
- Path:** `http://localhost:8088/crm/order/50/item`
- Body (JSON):**

```

{
  "triggerException": true,
  "name": "Updated Item Name",
  "price": 150.00,
  "stock": 1234
}

```
- Status:** `503 Service Unavailable`
- Time:** `39 ms`
- Size:** `397 B`
- Response Body:**

```

1 Item service is temporarily unavailable. Please try again later.

```

```

    "circuitBreakers": {
      "itemService": {
        "failureRate": "100.0%",
        "slowCallRate": "0.0%",
        "failureRateThreshold": "10.0%",
        "slowCallRateThreshold": "100.0%",
        "bufferedCalls": 10,
        "failedCalls": 10,
        "slowCalls": 0,
        "slowFailedCalls": 0,
        "notPermittedCalls": 0,
        "state": "OPEN"
      }
    }
  }
}

```

4.3 OAuth2 授权

在微服务架构中集成一个 OAuth 2.0 授权服务器，允许第三方应用获取有限的访问权限到 HTTP 服务，为不同的服务提供安全的用户认证和授权机制。

```

security:
  oauth2:
    resourceserver:
      jwt:
        public-key-location: classpath:public_key.pem

```

首先利用 OAuth2 生成 token，用户输入正确的用户名密码获得 token：

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8088/login`
- Method:** `POST`
- Body (JSON):**

```

{
  "username": "user7",
  "password": "pass7181"
}

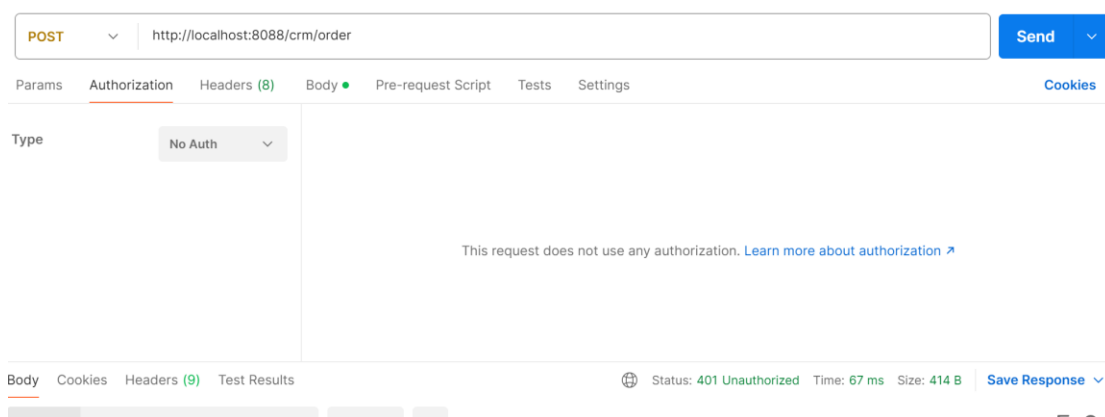
```
- Status:** `200 OK`, Time: `2.13 s`, Size: `941 B`
- Response Body (JSON):**

```

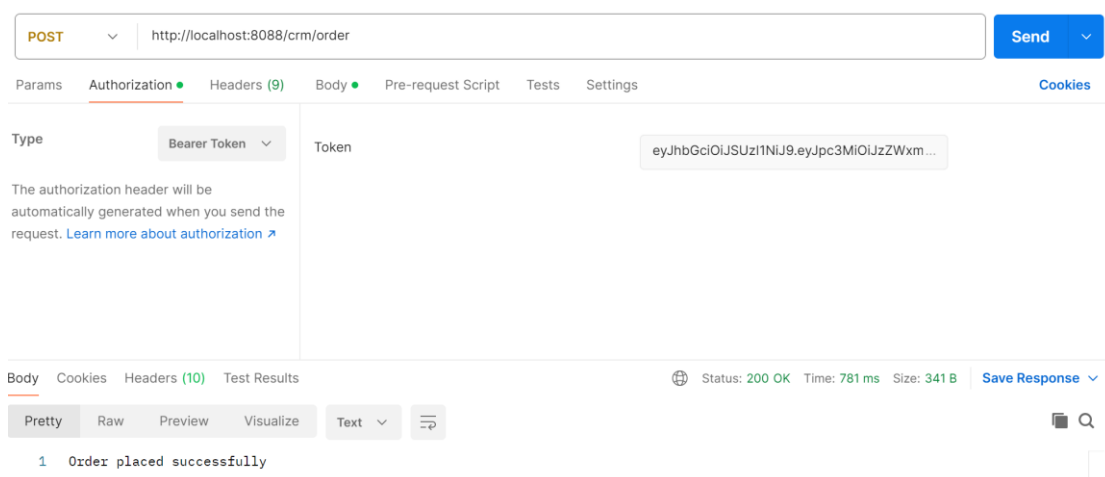
{
  "access_token": "eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoidXNlcjciLCJleHAiOjE3MTg0Njg2NDUsImh0bCI6MTcxODQ2NTA0NSwic2Nvc",
  "token_type": "Bearer",
  "expires_in": "3600"
}

```

没有 token 时无法访问：



有 token 后可以正确访问：



4.5 通过网关向外部用户公开 API

(1) 路由规则：定义了两条路由规则。

crm-service-route：将访问/crm/**的请求路由到 crm-service 服务。

oauth-service-route：将访问/login/**的请求路由到 oauth-service 服务。

(2) 服务调用：使用 lb://表示负载均衡，网关会通过服务发现获取服务实例，并进行负载均衡。

(3) 过滤器：移除请求中的 Cookie 头部，这通常用于增强安全性，避免敏感信息在微服务间传播。

```

routes:
- id: crm-service-route
  uri: lb://crm-service
  predicates:
    - Path=/crm/**
  filters:
    - RemoveRequestHeader=Cookie
- id: oauth-service-route
  uri: lb://oauth-service
  predicates:
    - Path=/login/**
  filters:
    - RemoveRequestHeader=Cookie

```

4.6 使用 Spring cloud 进行集中配置和跟踪

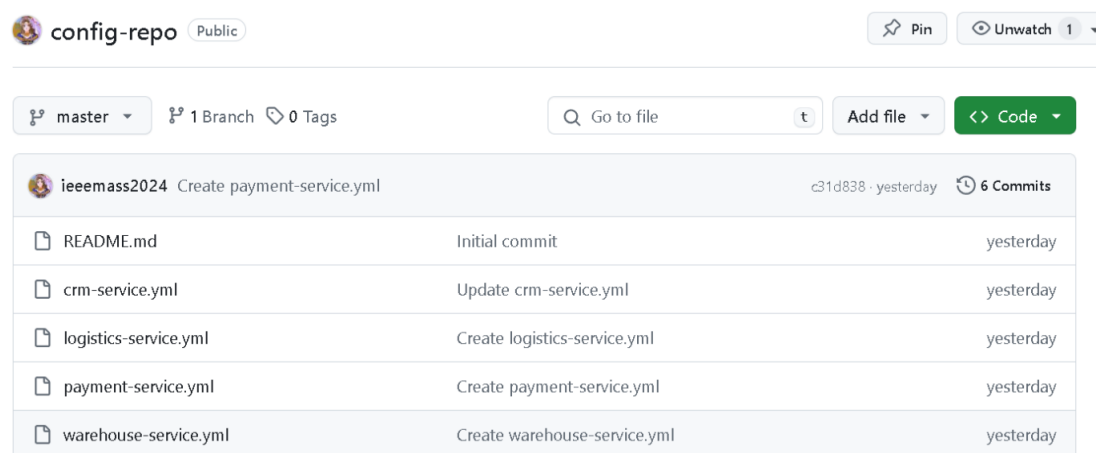
4.6.1 集中配置

配置 config server，并将其与 github 地址绑定。

如果更改项目的配置文件可以在 github 仓库里实施，config server 一直保持启动，server 会定期从仓库拉取配置，将其他微服务和 config server 建立连接，实现动态刷新配而无需重新启动微服务。

配置文件项目链接：

[ieeemass2024/config-repo \(github.com\)](https://github.com/ieeemass2024/config-repo)



config-repo Public

master 1 Branch 0 Tags

Go to file Add file <> Code

File	Commit Message	Date
README.md	Initial commit	yesterday
crm-service.yml	Update crm-service.yml	yesterday
logistics-service.yml	Create logistics-service.yml	yesterday
payment-service.yml	Create payment-service.yml	yesterday
warehouse-service.yml	Create warehouse-service.yml	yesterday

检查服务配置成功：

```
← localhost:8888/logistics-service/default
1 {
2   "name": "logistics-service",
3   "profiles": [
4     "default"
5   ],
6   "label": null,
7   "version": "c31d53838228e4ea91aa461522b023f3b99c936",
8   "state": null,
9   "propertySources": [
10     {
11       "name": "https://github.com/leemans2024/config-repo/logistics-service.yml",
12       "source": {
13         "server.port": 9001,
14         "spring.application.name": "logistics-service",
15         "spring.cloud.nacos.discovery.server-addr": "127.0.0.1:8848",
16         "spring.cloud.nacos.discovery.username": "nacos",
17         "spring.cloud.nacos.discovery.password": "nacos",
18         "spring.datasource.url": "jdbc:mysql://localhost:3306/test?useSSL=false",
19         "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
20         "spring.datasource.username": "root",
21         "spring.datasource.password": "123456",
22         "spring.datasource.initial-size": 10,
23         "spring.datasource.max-active": 20,
24         "spring.datasource.max-idle": 8,
25         "spring.datasource.min-idle": 8,
26         "mybatis.mapper-locations": "classpath:mybatis/*.xml",
27         "mybatis.configuration.map-underscore-to-camel-case": true
28       }
29     }
30   ]
31 }
```

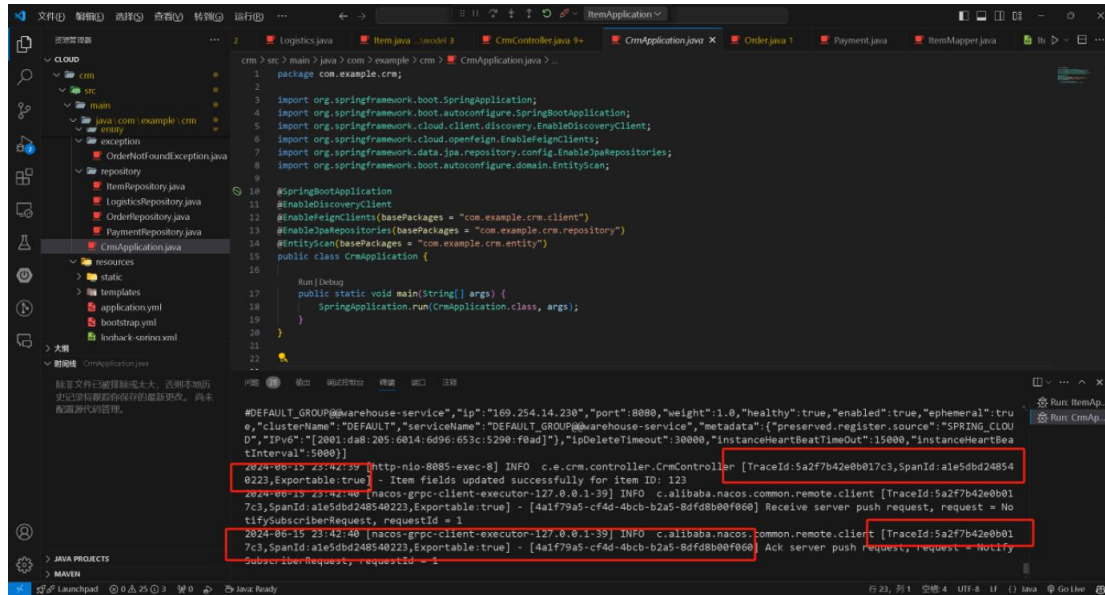
```
← localhost:8888/warehouse-service/default
1 {
2   "name": "warehouse-service",
3   "profiles": [
4     "default"
5   ],
6   "label": null,
7   "version": "c31d53838228e4ea91aa461522b023f3b99c936",
8   "state": null,
9   "propertySources": [
10     {
11       "name": "https://github.com/leemans2024/config-repo/warehouse-service.yml",
12       "source": {
13         "server.port": 8080,
14         "spring.application.name": "warehouse-service",
15         "spring.cloud.nacos.discovery.server-addr": "127.0.0.1:8848",
16         "spring.cloud.nacos.discovery.username": "nacos",
17         "spring.cloud.nacos.discovery.password": "nacos",
18         "spring.datasource.url": "jdbc:mysql://localhost:3306/test?useSSL=false",
19         "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
20         "spring.datasource.username": "root",
21         "spring.datasource.password": "123456",
22         "spring.datasource.initial-size": 10,
23         "spring.datasource.max-active": 20,
24         "spring.datasource.max-idle": 8,
25         "spring.datasource.min-idle": 8,
26         "mybatis.mapper-locations": "classpath:mybatis/*.xml",
27         "mybatis.configuration.map-underscore-to-camel-case": true
28       }
29     }
30   ]
31 }
```

```
← localhost:8888/payment-service/default
1 {
2   "name": "payment-service",
3   "profiles": [
4     "default"
5   ],
6   "label": null,
7   "version": "c31d53838228e4ea91aa461522b023f3b99c936",
8   "state": null,
9   "propertySources": [
10     {
11       "name": "https://github.com/leemans2024/config-repo/payment-service.yml",
12       "source": {
13         "server.port": 9002,
14         "spring.application.name": "payment-service",
15         "spring.cloud.nacos.discovery.server-addr": "127.0.0.1:8848",
16         "spring.cloud.nacos.discovery.username": "nacos",
17         "spring.cloud.nacos.discovery.password": "nacos",
18         "spring.datasource.url": "jdbc:mysql://localhost:3306/test?useSSL=false",
19         "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
20         "spring.datasource.username": "root",
21         "spring.datasource.password": "123456",
22         "spring.datasource.initial-size": 10,
23         "spring.datasource.max-active": 20,
24         "spring.datasource.max-idle": 8,
25         "spring.datasource.min-idle": 8,
26         "mybatis.mapper-locations": "classpath:mybatis/*.xml",
27         "mybatis.configuration.map-underscore-to-camel-case": true
28       }
29     }
30   ]
31 }
```

```
← localhost:8888/crm-service/default
1 {
2   "name": "crm-service",
3   "profiles": [
4     "default"
5   ],
6   "label": null,
7   "version": "c31d53838228e4ea91aa461522b023f3b99c936",
8   "state": null,
9   "propertySources": [
10     {
11       "name": "https://github.com/leemans2024/config-repo/crm-service.yml",
12       "source": {
13         "server.port": 8085,
14         "spring.datasource.url": "jdbc:mysql://localhost:3306/test?useSSL=false",
15         "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
16         "spring.datasource.username": "root",
17         "spring.datasource.password": "123456",
18         "spring.datasource.initial-size": 10,
19         "spring.datasource.max-active": 20,
20         "spring.datasource.max-idle": 8,
21         "spring.datasource.min-idle": 8,
22         "mybatis.mapper-locations": "classpath:mybatis/*.xml",
23         "mybatis.configuration.map-underscore-to-camel-case": true,
24         "spring.cloud.nacos.discovery.server-addr": "127.0.0.1:8848",
25         "spring.cloud.nacos.discovery.username": "nacos",
26         "spring.cloud.nacos.discovery.password": "nacos"
27       }
28     }
29   ]
30 }
```

4.6.2 跟踪

集成 Spring Cloud Sleuth 进行链路追踪，输出 TraceId,SpanId,Exportable 三项



```
cm > src > main > java > com > example > cm > CrmApplication.java > ...
1 package com.example.crm;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.cloud.openfeign.EnableFeignClients;
7 import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
8 import org.springframework.boot.autoconfigure.domain.EntityScan;
9
10 @SpringBootApplication
11 @EnableDiscoveryClient
12 @EnableFeignClients(basePackages = "com.example.crm.client")
13 @EnableJpaRepositories(basePackages = "com.example.crm.repository")
14 @EntityScan(basePackages = "com.example.crm.entity")
15 public class CrmApplication {
16
17     @Run[Debug]
18     public static void main(String[] args) {
19         SpringApplication.run(CrmApplication.class, args);
20     }
21
22 }
```

```
#DEFAULT_GROUP@warehouse-service", "ip": "169.254.14.238", "port": 8080, "weight": 1.0, "healthy": true, "enabled": true, "ephemeral": true, "clusterName": "DEFAULT", "serviceName": "DEFAULT_GROUP@warehouse-service", "metadata": {"preserved.register.source": "SPRING_CLOUD", "IPV6": "[2001:da8:205:6014:6d96:653c:5290:f8ad]"; "ipDeleteTimeout": 30000, "instanceHeartBeatTimeout": 15000, "instanceHeartBeatInterval": 5000}]
2024-06-15 23:42:39 [http-nio-8085-exec-8] INFO c.e.crm.controller.CrmController [TraceId:5a2f7b42e0b017c3,SpanId:a1e5dbd248540223,Exportable:true] - Item fields updated successfully for item ID: 123
2024-06-15 23:42:40 [ncos-grpc-client-executor-127.0.0.1-39] INFO c.alibaba.nacos.common.remote.client [TraceId:5a2f7b42e0b017c3,SpanId:a1e5dbd248540223,Exportable:true] - [4a1f79a5-cf4d-4bcb-b2a5-8dfd8b00f060] Receive server push request, request = NotifySubscriberRequest, requestId = 1
2024-06-15 23:42:40 [ncos-grpc-client-executor-127.0.0.1-39] INFO c.alibaba.nacos.common.remote.client [TraceId:5a2f7b42e0b017c3,SpanId:a1e5dbd248540223,Exportable:true] - [4a1f79a5-cf4d-4bcb-b2a5-8dfd8b00f060] Ack server push request, request = NotifySubscriberRequest, requestId = 1
```