



# 220120

🕒 생성일	@2022년 1월 20일 오전 9:33
🏷 태그	
📁 속성	

## 결과

Web

aa

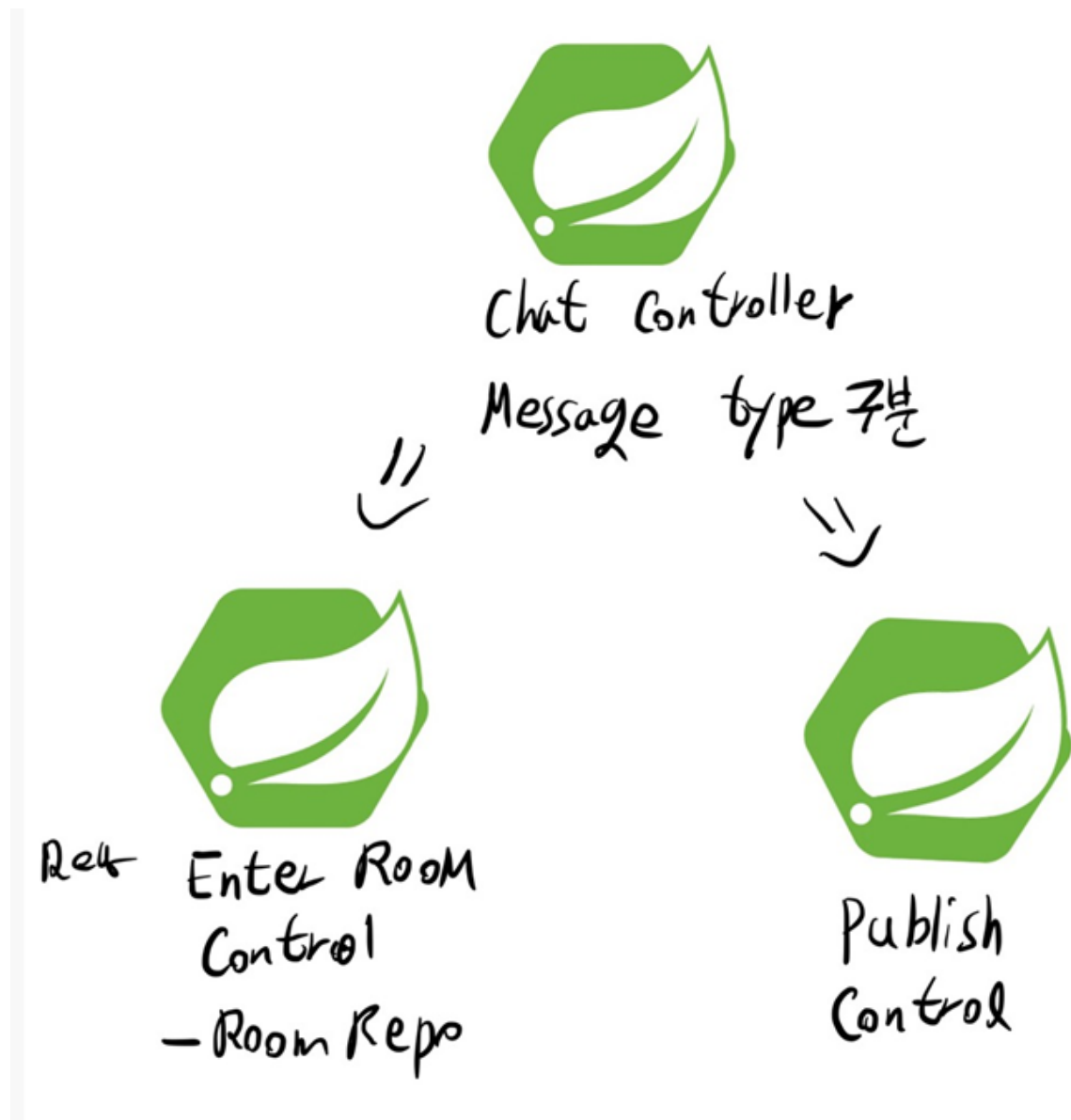
내용		보내기
song - hi		
gyu won - Hi		

CMD(RedisServer)

```
명령 프롬프트 - redis-cli
C:\Users\82104>redis-cli
127.0.0.1:6379> publish $ publish "0e59a319-1c2b-4740-ba80-1fb13b1c617a" "{"type":"TALK",
Invalid argument(s)
127.0.0.1:6379> publish "0e59a319-1c2b-4740-ba80-1fb13b1c617a" "{"type":"TALK",roomId":"3f0f893a-5849-4028-9755-8c6c8ab1846b",
Invalid argument(s)
127.0.0.1:6379> publish "0e59a319-1c2b-4740-ba80-1fb13b1c617a" "{"type":"TALK",sender":"gyu won",
Invalid argument(s)
127.0.0.1:6379> publish "0e59a319-1c2b-4740-ba80-1fb13b1c617a" "{"type":"TALK",message":"Hi"}"
(error) ERR unknown command 'publish'
127.0.0.1:6379> publish "0e59a319-1c2b-4740-ba80-1fb13b1c617a" "{"type":"TALK",roomId":"0e59a319-1c2b-4740-ba80-1fb13b1c617a",sender":"gyu won",message":"Hi"}"
(integer) 1
127.0.0.1:6379> subscribe "0e59a319-1c2b-4740-ba80-1fb13b1c617a"
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "0e59a319-1c2b-4740-ba80-1fb13b1c617a"
3) (integer) 1
1) "message"
2) "0e59a319-1c2b-4740-ba80-1fb13b1c617a"
3) "{"type":"TALK",roomId":"0e59a319-1c2b-4740-ba80-1fb13b1c617a",sender":"song",message":"hi"}"
```

chatController 분리

아래 두 기능 모두 Chatrepository에 요청을 보낸다



chatRoomcontroller 분리

위 ChatController와 다르게 Req이 앞에 붙어 있으면 필요 없는 기능이다



ChatRoom Control



Req Create Room  
- Room Repo



main view



Req find Room



room view

ChatRoomRepository 기능



Chat Room repo



Create Room



find Room

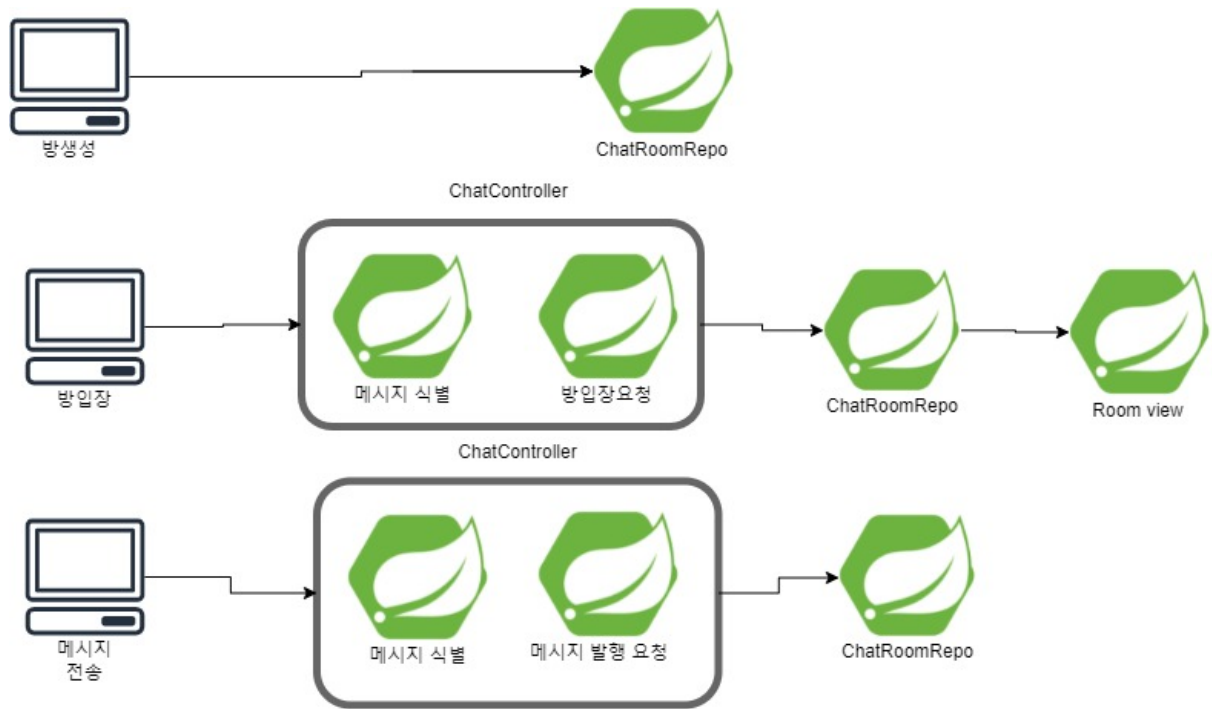


enter Room

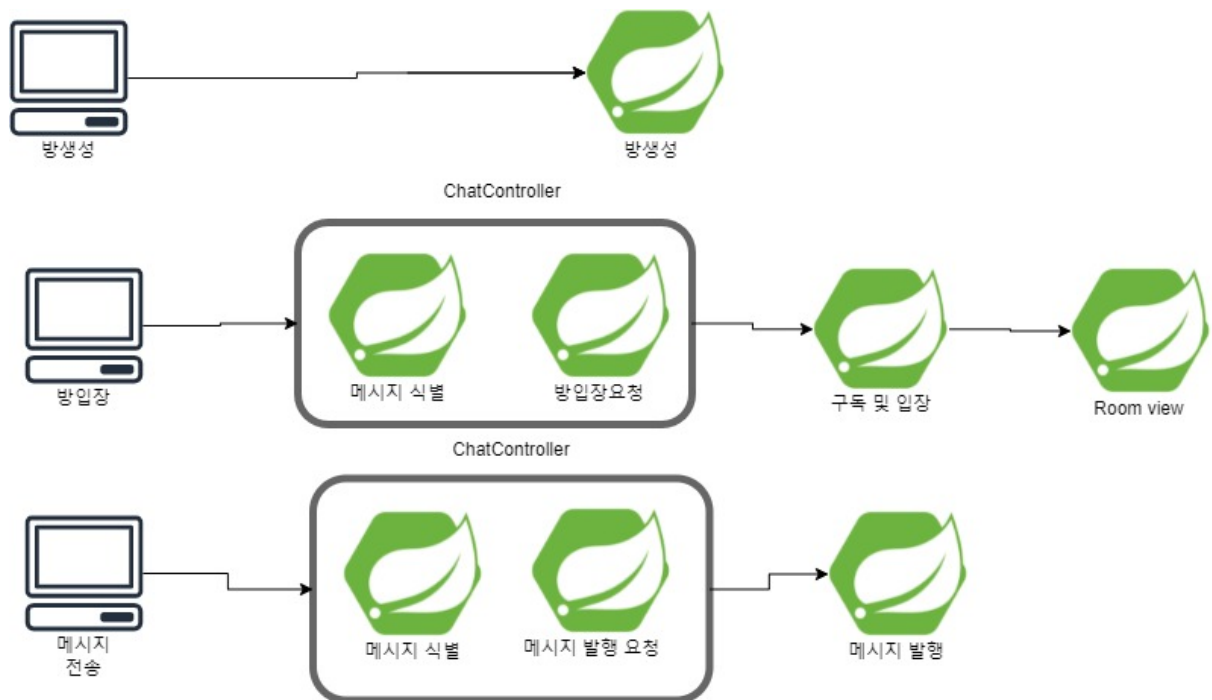


get topic

구성도



## 지향한 구성도



## Eureka

## DS Replicas

localhost

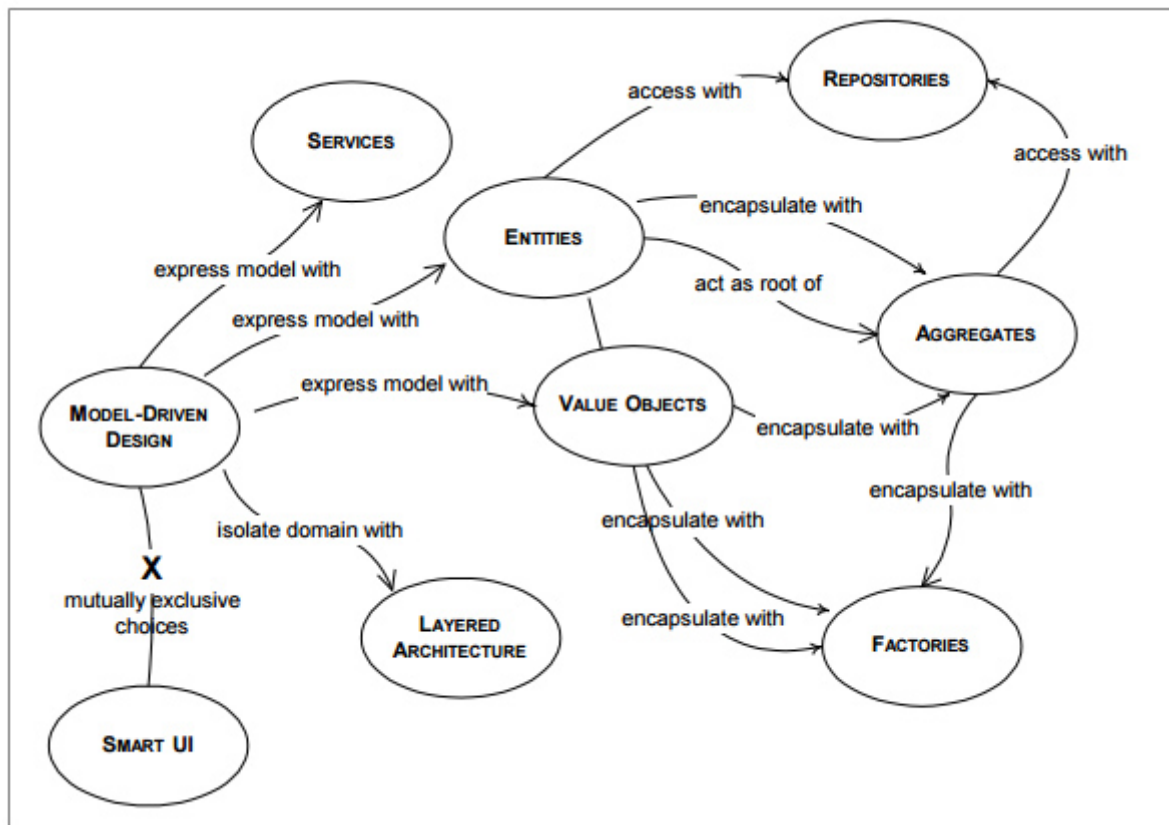
## Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CHATCONTROLLER	n/a (1)	(1)	UP (1) - chatcontroller-microservice-instance
CHATROOMCONTROLLER	n/a (1)	(1)	UP (1) - chatRoomcontroller-microservice-instance
CHATROOMREPO	n/a (1)	(1)	UP (1) - chatRoomrepo-microservice-instance
CLIENTBOUNDARY	n/a (1)	(1)	UP (1) - ClientBoundary-microservice-instance
MAINBOUNDARY	n/a (1)	(1)	UP (1) - MainBoundary-microservice-instance
ROOMVIEW	n/a (1)	(1)	UP (1) - roomview-microservice-instance

## 문제점

ChatRepository에서 구독, 메시지 발행, 방 생성, 입장을 모두 처리하는 형태

아래 그림과 같이 다시 명칭 수정 및 재구성의 필요성이 느껴짐



## 이후 할일

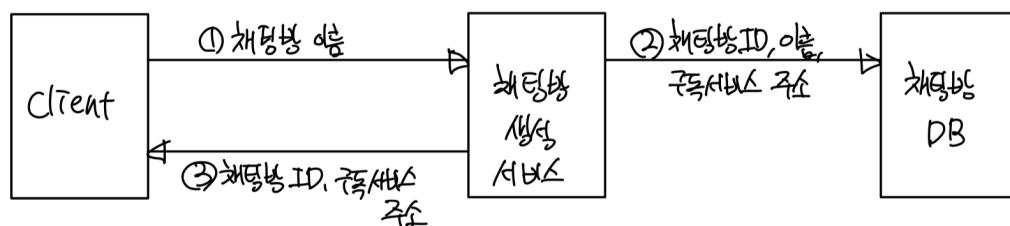
1. Spring view 문제 해결
2. War or jar로 EC2에 배포
3. 마이크로 서비스 추가 분리(코드 수정 이후 repository 분리)

# 진호

## 기능별 구조

### 1. 채팅방 생성

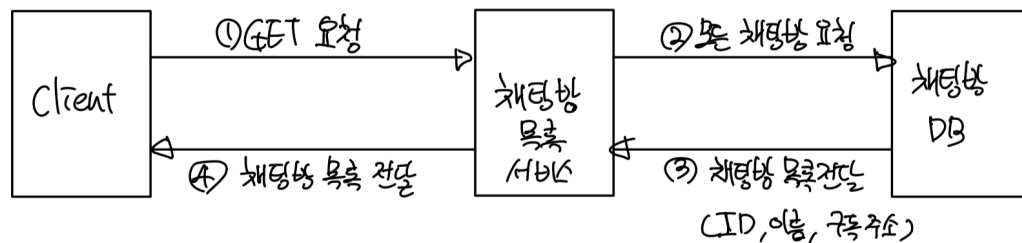
#### ① 채팅방 생성



- 클라이언트 → 채팅방 생성 서비스
  - 채팅방 이름 전송
- 채팅방 생성 서비스 → 채팅방 DB
  - 채팅방 ID, 이름, 구독 서비스 주소 저장
- 채팅방 생성 서비스 → 클라이언트
  - 채팅방 ID, 구독 서비스 주소 전달

### 2. 채팅방 목록

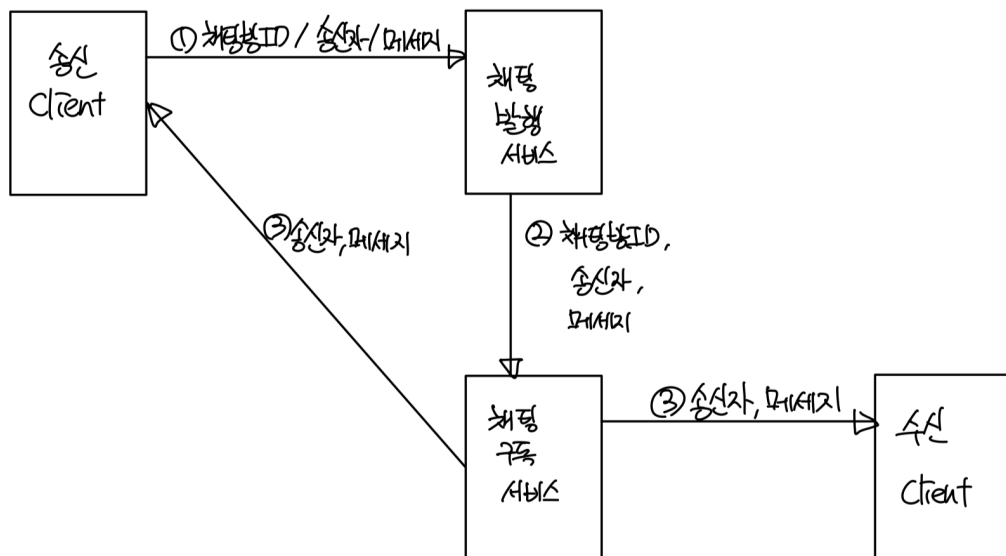
#### ② 채팅방 가져오기



- 클라이언트 → 채팅방 목록 서비스
  - 채팅방 목록 GET 요청
- 채팅방 목록 서비스 ↔ 채팅방 DB

- 모든 채팅방 데이터 요청/전달
3. 채팅방 목록 서비스 → 클라이언트
- 채팅방(ID, 이름, 주소) 리스트 전달
3. 채팅 발행/구독

### ② 채팅 발행/구독



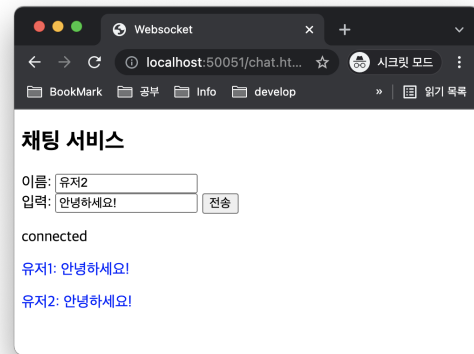
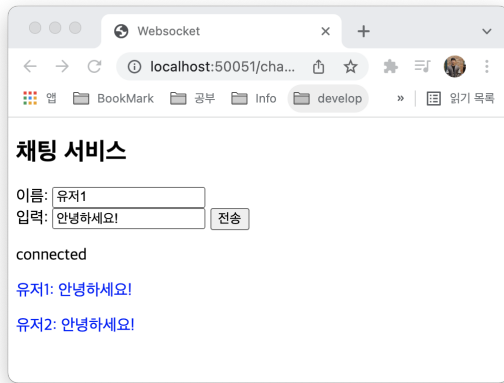
→ 채팅방 생성/접속을 통해서 채팅 구독 서비스와 WebSocketSession이 생성된 상태

1. 송신 클라이언트 → 채팅 발행 서비스
  - 채팅방 ID, 송신자 이름, 메시지 전송
2. 채팅 발행 서비스 → 채팅 구독 서비스
  - 채팅방 ID, 송신자 이름, 메시지 전달
  - 채팅 구독 서비스가 여러 곳일 경우 모두 전달
3. 채팅 구독 서비스 → 수신 클라이언트(송신 클라이언트 포함)
  - 해당 ID의 채팅방에 접속중인 모든 클라이언트에 송신자 이름, 메시지 전송

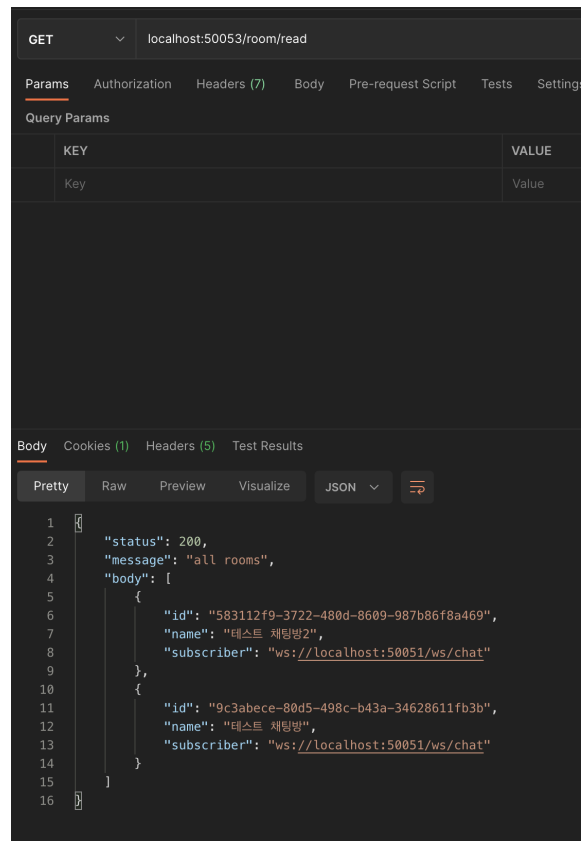
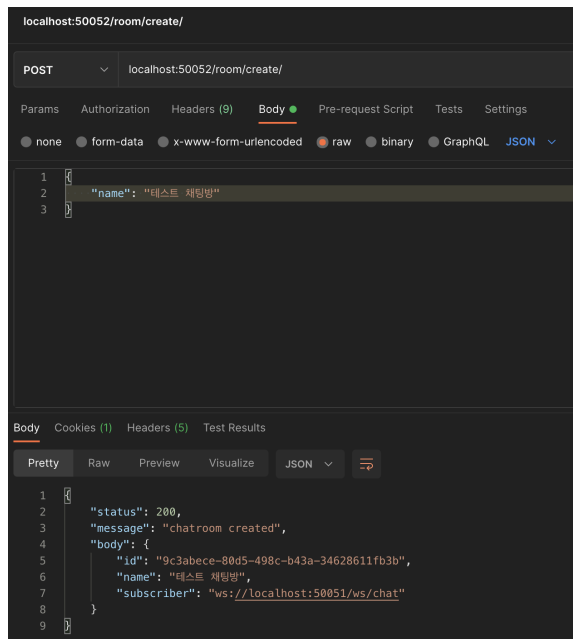
## 결과

1. 채팅 발행/구독 서비스를 통한 채팅 송수신





## 2. 채팅방 생성/목록 api



## 3. 추가 구현사항

- i. 클라이언트에서 채팅방 생성 UI
- ii. 클라이언트에서 채팅방 접속 UI