💬

[신버전] 채팅 서버 기능 명세, 아키텍처

생성일	@2024년 1월 24일 오후 3:18
태그	채팅 서버
사람	수아 강 성균관대학교정소연

아키텍처

- 채팅 서버 - 상태 관리 서버
- 채팅 서버
  - 스케일 아웃에 용이한 구조
- 상태 관리 서버
  - 채팅 on/off 상태 관리
  - 어느 유저가 어느 서버에 위치하는지 ??
- 참고 자료. 가상 면접 사례로 배우는 대규모 시스템 설계 기초

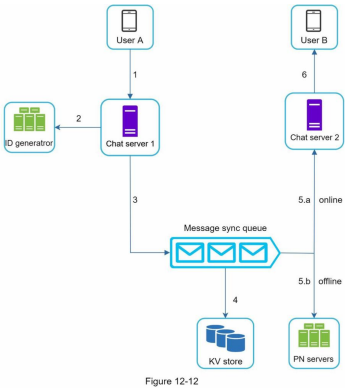
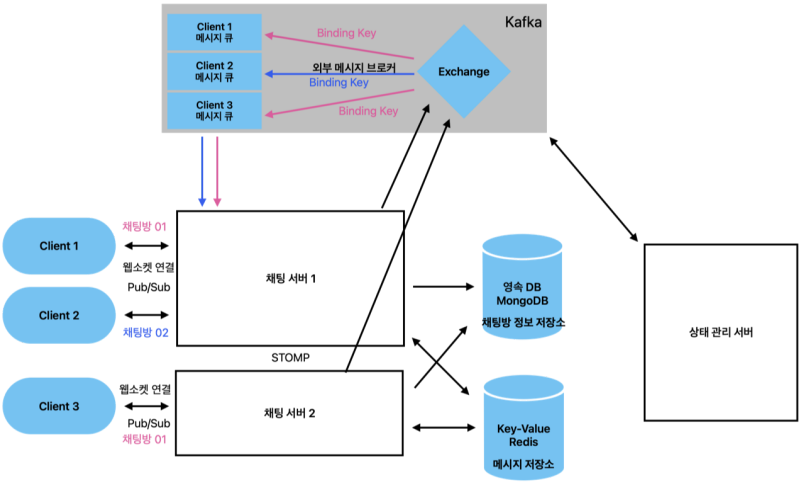
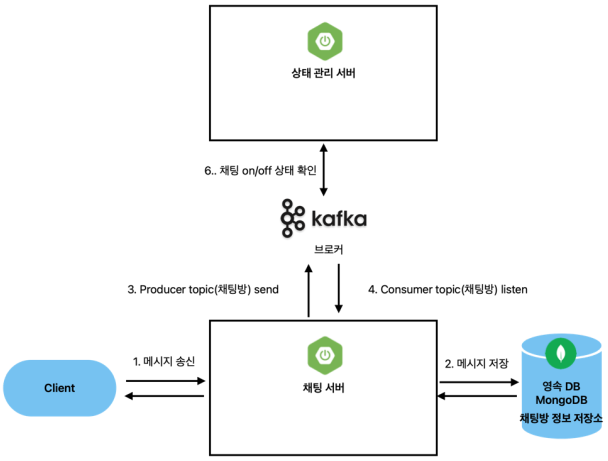


Figure 12-12

아키텍처 - 개괄



아키텍처 - 기능



시퀀스 다이어그램 - 기능

채팅방 생성 로직

```
sequenceDiagram
    participant User as 유저
    participant PlaylistServer as 플레이리스트서버
    participant ChatServer as 채팅서버
    participant SearchServer as 검색서버

    User ->> PlaylistServer: 채팅방 생성 요청
    PlaylistServer ->> ChatServer: 채팅방 생성 요청
    ChatServer ->> ChatServer: 채팅방 생성 로직 수행(채팅방 정보 저장)
    ChatServer ->> SearchServer: 인덱싱 요청

    SearchServer -->> ChatServer: 인덱싱 수행 완료 응답
    ChatServer -->> PlaylistServer: 채팅방 생성 완료 응답
    PlaylistServer -->> User: 채팅방 생성 완료 메시지
```

시퀀스 다이어그램 - 통신

주요 기능

1. **채팅방 입장:**
   - 참여자가 채팅방에 입장하면, 채팅 서버에게 "입장 요청"을 합니다.
   - 채팅 서버는 상태 서버에 "상태 요청"을 보내 현재 참여자의 채팅 가능 여부를 확인합니다.
   - 상태 서버는 "상태 응답"을 통해 채팅 가능 여부를 알려주고, 채팅 서버는 참여자에게 채팅 가능 여부를 전달합니다.
2. **채팅 상태 변경:**
   - 참여자는 언제든지 채팅 상태를 변경할 수 있습니다.

- 상태 변경 요청을 하면, 채팅 서버가 상태 서버에게 "상태 변경 요청"을 전송합니다.
- 상태 서버는 "상태 변경 응답"을 통해 상태 변경이 완료되었음을 알려주고, 채팅 서버는 참여자에게 변경된 채팅 상태를 전달합니다.

3. **채팅 메시지 전송:**
   - 참여자가 채팅을 시작하려면, 채팅 서버에게 "채팅 메시지 전송 요청"을 합니다.
   - 채팅 서버는 참여자의 채팅 상태를 확인하고, "on" 상태인 경우에만 브로커를 통해 채팅 메시지를 전송합니다.
   - 브로커는 채팅 메시지를 다른 참여자에게 전달하고, 채팅 서버는 메시지를 수신하여 해당 참여자에게 전달합니다.

기술 스택

- Springboot 3.2.2
  - Java : Jdk 17
  - build : Gradle
  - config : application.yml
  - Dependencies
    - Lombok
    - Spring web mvc
    - WebSocket
    - Redis (key: 채팅방 식별자, value: 메시지, 메시지 캐시DB-채팅방 종속적 정보)
    - MongoDB (채팅방 정보 영속DB)
    - Kafka
    - Spring Cloud
      - OpenFeign
      - netflix eureka client
    - Tracing
      - micrometer-tracing-bridge-brave
      - zipkin
      - logstash
- MongoDB
  - 채팅방 정보 저장용
    - 채팅방 식별자 id (채팅 시작시 UUID 부여해서 사용)
    - 원본 플레이리스트 정보
      - 플레이리스트 id
      - 플레이리스트 곡
      - 플레이리스트 소유자
    - 채팅방 명 (== 원본 플레이리스트 명)
    - 채팅장 명 (== 원본 플레이리스트 소유자)

```
room_id : 채팅방 식별자 id
```

```
playlist_id : 원본 플레이리스트 식별자 id
```

- MongoDB (채팅방 식별자 : 채팅 메시지)
  - 저장 데이터
    - 채팅방 정보
      - 채팅방 식별자 id (채팅 시작시 UUID 부여해서 사용)
      - 원본 플레이리스트 정보
        - 플레이리스트 id
        - 플레이리스트 곡
      - 채팅방 명 == 원본 플레이리스트 명
      - 채팅장 식별자 id

```
room_id : {
    // uid : messages
    //  ...
}
```

- 웹소켓
  - 클라이언트와 서버 사이의 실시간 통신
  - 비동기 메시징
- Kafka → pub/sub (현재는 STOMP pub/sub 하지만 외부 메시지 브로커 사용 예정)
  - 토픽(채팅방)을 기준으로 메시징
  - producer, consumer 가 pub/sub 역할
  - STOMP
    - 소켓 통신을 STOMP 프로토콜을 이용해 pub/sub 구조로 구현
    - Spring 에서 지원하는 STOMP는 In-Memory 브로커를 이용하므로 다수 서버일 경우 다른 서버 사용자와 채팅 불가 ⇒ 외부 메시지 브로커를 사용
      - pub/sub, 헤더값 세팅 - 인증처리
      - 채팅방 생성 : Topic 생성
      - 채팅방 입장 : Topic 구독
      - 메시지 : Topic으로 메시지 송신 (pub), 메시지 수신 (sub)

기능 명세

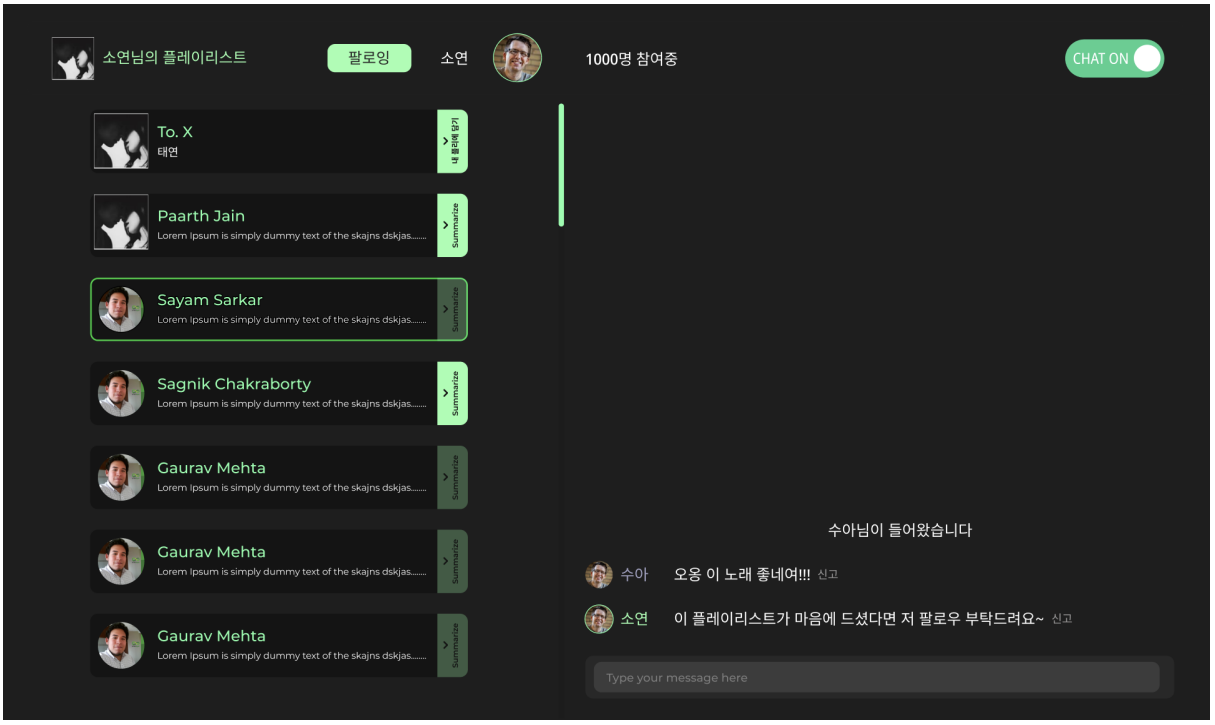
주요 기능

- 채팅 서비스
  - 카톡 느낌
    - 참여자 목록 → 상세보기 가능 ?
    - 참여자 수는 표기됩니다.
  - 채팅방 관리
  - 메시지 송수신

- 접속상태 서버
    - 참여자의 채팅 online/offline

대분류	소분류	detail	기술 명세
메시지 송수신	- 채팅방에 참여한 유저는 채팅 메시지 송수신이 가능합니다.	- 메시지는 유저 닉네임, 유저 프로필 사진, 본문 (텍스트 only)로 구성됩니다. - 한 페이지 최대 페이지네이션 12개	나아아아중에 시간 나면 이미지
채팅방 시작 (생성)	- 플레이리스트 소유자는 플레이리스트 화면에서 채팅 시작 버튼을 통해 채팅방을 생성할 수 있습니다.		- 버튼 클릭시 바로 시작 (딜레이 없이)
		- 채팅이 시작된 이후에 채팅장 (원본 플레이리스트 소유자)은 플레이리스트 수정이 불가능합니다.	
채팅방 입장	- 로그인 한 유저는 채팅페이지를 통해 활성화된 채팅방에 참여할 수 있습니다.	- 채팅방 내에서 유저는 채팅 화면, 플레이리스트 화면을 확인할 수 있습니다. - 유저가 채팅방에 입장시 동채팅방 참여자들은 OO님이 입장하였습니다. 문구를 수신합니다.	들어오는 사람에 대한 환영 메시지에 대한 ENTER과 방에 있는 사람들이 채팅을 칠 때 사용하는 TALK 두 가지로 메시지 타입 구분
채팅방 참여	- 채팅방 입장시 online 상태입니다. (화면상 따로 표기되진 않음)		- 상태 서버
	- 채팅방 참여자간 프로필 조회와 이를 통한 팔로잉이 가능합니다.		
채팅방 리스트 화면	- 메인 페이지 > 채팅 페이지에서 클라이언트는 채팅방 썸네일, 채팅방명, 참여자 수를 확인할 수 있습니다.		
채팅방 나가기 (퇴장)	- 브라우저 닫기, 뒤로가기 통해 채팅 화면 나가기		
채팅방 종료 (영구 퇴장)	- 완전히 나가기 버튼을 통해 영구 나가기 가능 - 현재 참여중인 채팅방이 4개인 상태에서 추가로 새로운 채팅방에 참여한다면 참여 일시가 가장 오래된 채팅방은 영구적으로 나가짐 - 채팅장이 완전히 나가기해도 남은 참여자들 영향 없음	- 내가 참여중인 채팅방 목록서 확인 불가능	
사용자 접속상태 목록	- 사용자의 online/offline 정보를 확인할 수 있습니다.	- 상태관리 서버에서 구현 - 채팅화면에서 확인가능(프론트엔드 우선순위 아주 낮음)	

화면 설계



API 명세

~~Presence-Server API~~

Presence-Server API Specification

Aa Feature	HTTP Method	URI	Response Code	서버 반영 여부	담당자	비고
상태관리 서버 Kafka 설정				Not started	성균관대학교정소연	
접속상태 on인 유저 목록 조회	GET	/api/v1/presence/{roomId}/on		Not started	성균관대학교정소연	라이브 채팅 페이지에서 구현 예정(우선순위는 낮음) 접속 상태 on인 유저 id 리스트 조회 && 접속 상태 off인 유저 id 리스트 조회 (/on-user + /off-user) 원래 하나의 API에 두개의 서비스로 가려했는데 method 요청 방식도 달라서 분리 결정.
접속상태 off인 유저 목록 조회		/api/v1/presence/{roomId}/off		Not started	성균관대학교정소연	
접속상태 on으로 변경 및 상태 전달	PATCH	/api/v1/presence/on		Not started	성균관대학교정소연	

Aa Feature	HTTP Method	URI	Response Code	서버 반영 여부	담당자	비고
접속상태 off로 변경 및 상태 전달	PATCH	/api/v1/presence/off		Not started	성균관대학교정소연	
제목 없음				Not started		

Chatting-Server API

구독 URL **/chatting/topic**

Chatting-Server API Specification

Aa Feature	HTTP Method	URI	Response Code	서버 반영 여부	담당자	비고
웹소켓 연결	WEBSOCKET	ws://localhost:18000/ws-chat		Done	수아 강	
채팅 메시지 송신	STOMP SEND	웹소켓 연결 후 /chat/pub/send		Done	수아 강	
채팅 메시지 수신	STOMP SUBSCRIBE	웹소켓 연결 후 /chat/topic/room/{roomId}		Done	수아 강	
새로 입장	STOMP JOIN	웹소켓 연결 후 /chat/pub/join		Done	수아 강	현재 플레이 해야하는 음악 id 응답에 포함하기
참여중인 채팅방 재입장, 새 메시지 조회	GET	/v1/api/chat/api/v1/rooms/joined/{roomId}		Done	수아 강	현재 플레이 해야하는 음악 id 응답에 포함하기
특정 채팅방 전체 히스토리 조회	GET	/v1/api/chat/api/v1/history/{roomId}		Done	수아 강	
채팅 메시지 페이지네이션	GET	/v1/api/chat/history		Done	수아 강	MAX 20개
메시지 전송 - 디버깅용 -	POST	/v1/api/chat/api/v1/message		Done	수아 강	

Aa Feature	HTTP Method	URI	Response Code	서버 반영 여부	담당자	비고
특정 채팅방 정보 조회	GET	/v1/api/rooms/{roomId}		Done	수아 강	
채팅방 영구 퇴장	POST	/v1/api/rooms/exit/{roomId}		Done	수아 강	
채팅방 잠시 나가기 + 마지막 읽은 메시지 id 저장	PUT	/v1/api/rooms/leave		Done	수아 강	
참여중인 채팅방 리스트 조회	GET	/v1/api/rooms/joined		Done	수아 강	
참여 가능한 채팅방 리스트 조회	GET	/v1/api/rooms/unjoined		Done	수아 강	
채팅방 생성	POST	/v1/api/rooms/create		Done	수아 강	playlist server에서 요청 받아서 처리하는 로직
전체 채팅방 리스트 정보 조회	GET	/v1/api/rooms/		Done	수아 강	for debugging
채팅방 생성시 검색 서버에 등록	POST	localhost:20000/search/index/send		Done		HTTP ver.
채팅방 생성시 검색 서버에 등록	STOMP SEND	/chat/topic/search/room/index		Done		카프카 ver.

구현 마일스톤

- rough
  - 채팅서버
    1. ~~웹소켓 설정~~
    2. config 설정

        a. Producer "KafkaProducerConfig"
        b. Consumer "KafkaConsumerConfig"
        c. Topic "KafkaTopicConfig"
      3. Producers, Consumers 구현
    - 상태서버
      1. 웹소켓 메시지 전달을 위한 kafka 설정
      2. 웹소켓 메시지 브로커 설정
      3. 채팅 on/off 상태 update
    - 통합
    - 테스트
- 순서 ?
  0. 웹소켓 연결 (완)
  1. 채팅 참여 & 송수신 (with kafka pub/sub)
     a. kafka config
     b. 채팅방 crud
     c. 서버간 통합? - kafka하면서 붙이기
     d. 상태관리 서버: 채팅 on/off
  2. 스케일아웃 (어느 유저가 어느 서버에 붙었는지)

채팅

1. 웹소켓 설정
2. config 설정
   a. Producer "KafkaProducerConfig"
   b. Consumer "KafkaConsumerConfig"
   c. Topic "KafkaTopicConfig"
3. Producers, Consumers 구현
4. 플레이리스트 공유 구현
5. 채팅방 내 메시지 페이지네이션

상태

1. 카프카. 웹소켓 설정
2. 채팅서버랑 연결

도메인별 역할 분담

- 채팅방 (소연)
  - **상태관리 서버** kafka 설정
  - 채팅방 입장 퇴장 online/offline 설정
    - 채팅방 목록 API

        - 내가 참여중인 채팅방
        - 내가 참여할 수 있는 채팅방(실시간 라이브 중)
        - 일단은 최신순 정렬
    - 사용자 접속 상태 목록 API
        - 특정 채팅방에 웹소켓으로 연결되어 있는 사용자들 online/offline
- 채팅(수아)
    - ~~채팅 서버 kafka 설정 0 오늘 월 29~~
    - 채팅방 생성 1 오늘 내일 월화 29-30
        - 채팅서버-플레이리스트 서버 간 API
        - 채팅방 생성 - 접속 시간 계산해 기준 싱크?
    - 채팅방 입퇴장 & 송수신 2 모레 수 31
    - 채팅 내역 페이지네이션 3 글피 목 2/1

.env

```
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=1234
MONGO_INITDB_DATABASE=chattingdb
MONGO_PORT=18001
MONGO_URI=mongodb://root:1234@mongodb:18001/chattingdb

# Kafka
KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT:PLAINTEXT
KAFKA_LISTENERS=PLAINTEXT://0.0.0.0:9092
KAFKA_INTER_BROKER_LISTENER_NAME=PLAINTEXT
```

데이터 구조

채팅방 - 영속 db에 저장할 구조

```
@Id
private String id;
private String roomName;
private Playlist playlist;

@Indexed
private HashMap<User> users;
```

```
// User
private Long uid;
private String nickName;
private String lastReadMessageId;
private LocalDateTime enteredAt;
```

```
// Playlist
private String id;
private String name;
private List<Music> musics;

// Music
private String title;
private String artist;
private String playtime;
private String thumbnail;  // image url
```

채팅방 생성 요청 데이터 구조 - roomCreateRequestDto

```
private String uid;
private String userName;
private Playlist playlist;
```

채팅방 응답 데이터 구조 - roomResponseDto

```
private String id;
private String roomName;
private long userCount;
private List<User> users;
private Playlist playlist;
private Music thumbnail;
```

Swagger