12. 채팅 시스템 설계

1단계. 문제 이해 및 설계 범위 확정

- 주요 요구사항
  - 1:1 채팅, 그룹채팅 모두 지원해야. 모바일 앱, 웹 둘다
  - DAU 5천만
  - 그룹채팅 최대 100명 제한
  - 메세지 길이 제한 100,000자
  - 채팅 이력은 영원히 보관
- 구현할 기능
  - 응답 지연이 낮은 1:1 채팅
  - 최대 100명까지 참여 가능한 그룹채팅
  - 사용자 접속상태 표시
  - 하나의 계정으로 여러 단말 동시 접속 지원
  - 푸시 알림

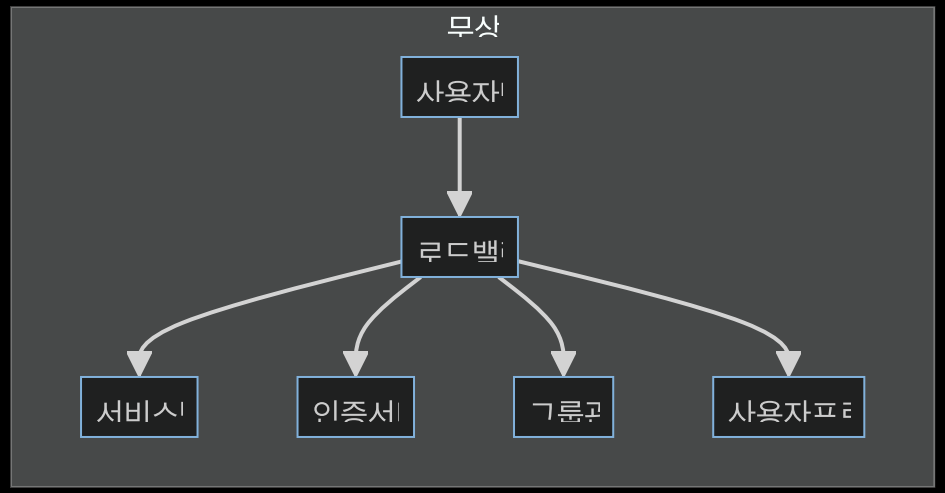
2단계: 개략적인 설계안 제시 및 동의 구하기

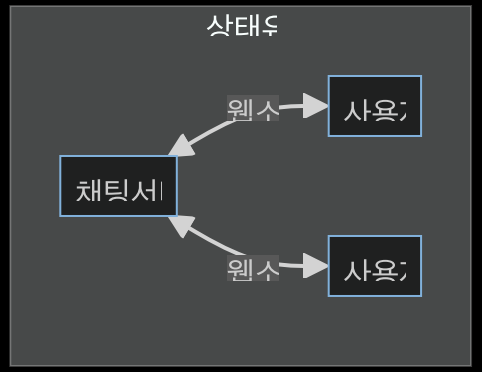
- 채팅 서비스의 역할
  - 클라이언트들로부터 메세지 수신
  - 메세지 수신자 결정 및 전달
  - 수신자가 접속 상태가 아닌 경우, 접속할 때까지 메세지 보관
- 어떤 프로토콜글 사용할 지 면접관과 상의해야
  - HTTP
    - 오랜 세월 검증된 프로토콜. 클라이언트가 연결을 만드는 데에는 적합
    - 서버에서 클라이언트로 임의 시점에 메시지 전송은 어려움
      - 이를 위해 폴링, 롱폴링, 웹소켓 등이 사용

폴링, 롱폴링, 웹소켓

- 폴링
  - 클라이언트가 주기적으로 서버에게 질의
  - 단점: 답해줄 메세지가 없는 경우 서버 자원 낭비
- 롱 폴링
  - 클라이언트는 새 메세지 반환 혹은 타임아웃때까지 연결을 유지.
  - 약점
    - 송신하는 클라이언트와 수신하는 클라이언트가 같은 채팅 서버에 접속하지 않을 수도
    - 서버 입장에서는 클라이언트가 연결을 해제했는지 알기 어려움
- 웹소켓
  - 클라이언트가 연결 생성. 연결은 항구적, 양방향
  - 연결 생성 후 서버는 클라이언트에게 비동기적으로 메세지 전송 가능
  - 연결이 항구적으로 유지되려면, 서버 측에서 연결 관리를 효율적으로 해야.

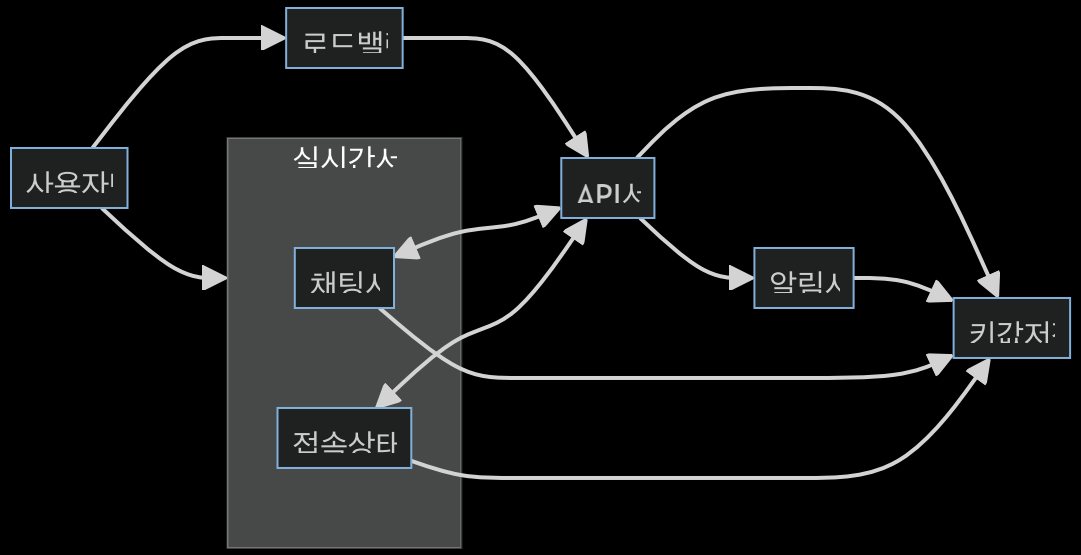
개략적 설계안





규모 확장성

- 고민해야 할 것
  - 서버 한 대로 얼마나 많은 접속을 동시에 허용할 수 있는가
  - 접속당 10K 메모리가 필요하다고 가정시
    - 1M 접속자를 처리하려면: 10GB 메모리 필요
    - 하지만, 한 서버에서 모두 처리하면, SPOF 가 될 수 있음



- 채팅서버: 클라이언트 사이에 메세지 중계 - 접속상태서버: 사용자 접속여부 관리 - API 서버: 로그인, 회원가입, 프로파일 변경 등 그외 나머지 전부를 처리 - 알림서버: 푸시알림 전송 - 키값저장소: 채팅이력 보관

저장소

- 어떤 DB를 쓰느냐는 중요
- 따져야 할 것
  - 데이터의 유형
    - 일반적 데이터: 안정성을 보장하는 관계형 DB에 보관
    - 채팅이력: 채팅 시스템에 고유한 데이터
      - 데이터가 많음. 빈번하게 사용되는 것은 최근에 주고받은 메세지
  - 읽기/쓰기 연산 패턴: 1:1 채팅앱은 읽기:쓰기 비율이 대략 1:1
- 키-값 저장소를 추천
  - 수평적 규모확장 용이
  - 낮은 데이터 접근 지연시간
  - 많은 채팅 시스템이 키-값 저장소를 채택

데이터 모델

- 1:1 채팅을 위한 메세지 테이블

message	
message_id	bigint
message_from	bigint
message_to	bigint
content	text
created_at	timestamp

- 그룹 채팅을 위한 메세지 테이블

group messages	
channel_id	bigint
message_id	bigint
message_to	bigint
content	text
created_at	timestamp

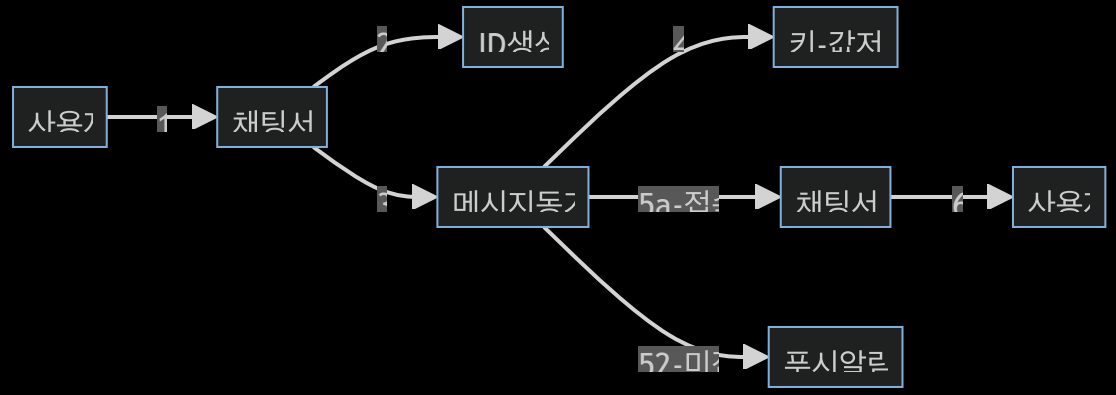
3단계: 상세 설계

서비스 탐색

- 클라이언트에게 적합한 채팅 ㅓ버를 추천
  - 기준: 클라이언트의 위치, 서버의 용량 등
  - 주요 솔루션: apache zookeeper

메세지 흐름

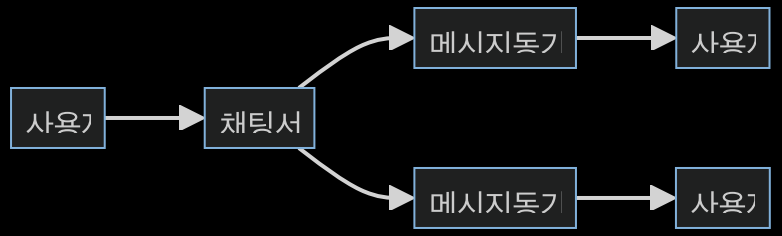
1:1 채팅 메시지 처리 흐름

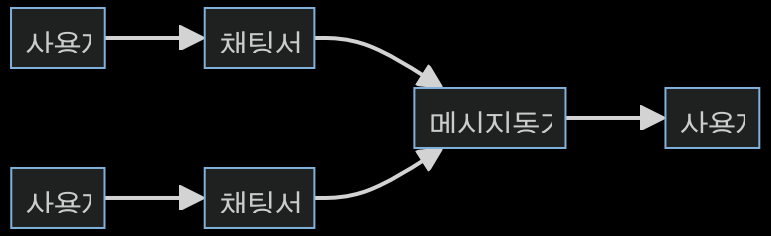


여러 단말 사이의 메시지 동기화

- 각 값 단말은 cur_max_message_id라는 변수를 유지
  - 관측된 가장 최신 메시지 ID를 추적

소규모 그룹 채팅에서의 메시지 흐름





- 새로운 메시지 확인을 위해 자기 큐만 보면 됨. 동기화가 단순
- 그룹이 크지 않으면, 메시지를 수신자별로 복사해서 큐에 넣는 것이 문제가 되지 않음.

접속 상태 표시

- 사용자의 상태가 바뀌는 시나리오
  - 로그인
    - 웹소켓 연결 시, 접속상태서버는 사용자의 last_active_at을 가짐
  - 로그아웃
    - 로그아웃시 접속상태서버에 전파됨
  - 접속 장애
    - 인터넷 연결이 끊어지면, 웹소켓도 끊어짐.
    - 인터넷이 끊어졌다 복구되는 것은 흔한 일. 박동검사를 통해 문제 해결
- 상태 정보의 전송
  - pub-sub model을 사용. 각각의 친구관계마다 채널을 하나씩
  - 크기가 커지면 비용, 시간이 많이 들게 됨.
    - 사용자가 그룹채팅 입장시에만 읽게 하거나, 갱신을 수동으로 하도록 유도

4단계: 마무리

- 추가 논의
  - 사진/비디오 등의 미디어 지원할 방법: 압축방식, 클라우드 저장소, 섬네일 생성 등
  - 종단간 암호화
  - 캐시: 클라이언트에 이미 읽은 메시지를 캐시
  - 로딩속도 개선: 사용자 데이터, 채널 등을 지역적으로 분산
  - 오류 처리
    - 채팅서버 오류: 채티엇버가 죽으면 서비스탐색기능이 동작하여, 클라이언트에게 새 서버를 배정
    - 메시지 재전송: 재시도/큐