채팅 아키텍처

태그	
작성자	조성찬

가상 면접 사례로 배우는 대규모 시스템 설계 기초 에서 배우는 채팅 시스템 설계

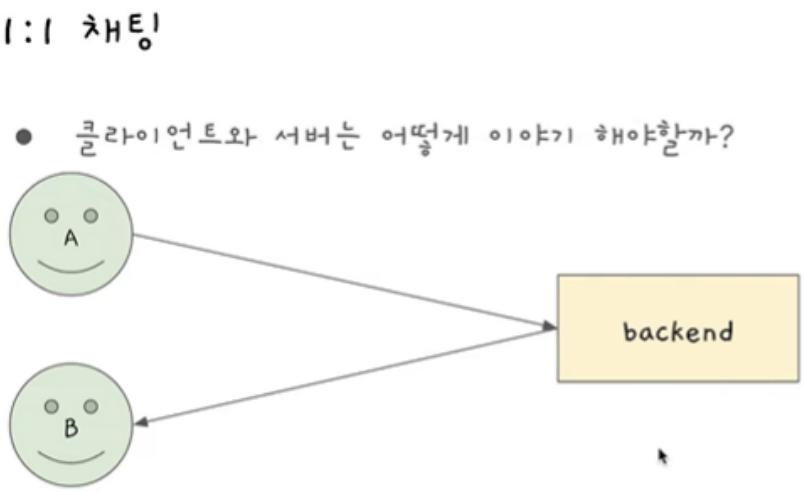
채팅의 트래픽 특성

엄청난 양의 트래픽 그룹 채팅 - 하루에 600억 메시지

오래된 채팅 잘 안 봄

Read / Write 비율 1:1

어떤 데이터베이스를 사용할 지 결정하는 데에 이런 트래픽 특성이 중요하게 작용한다.

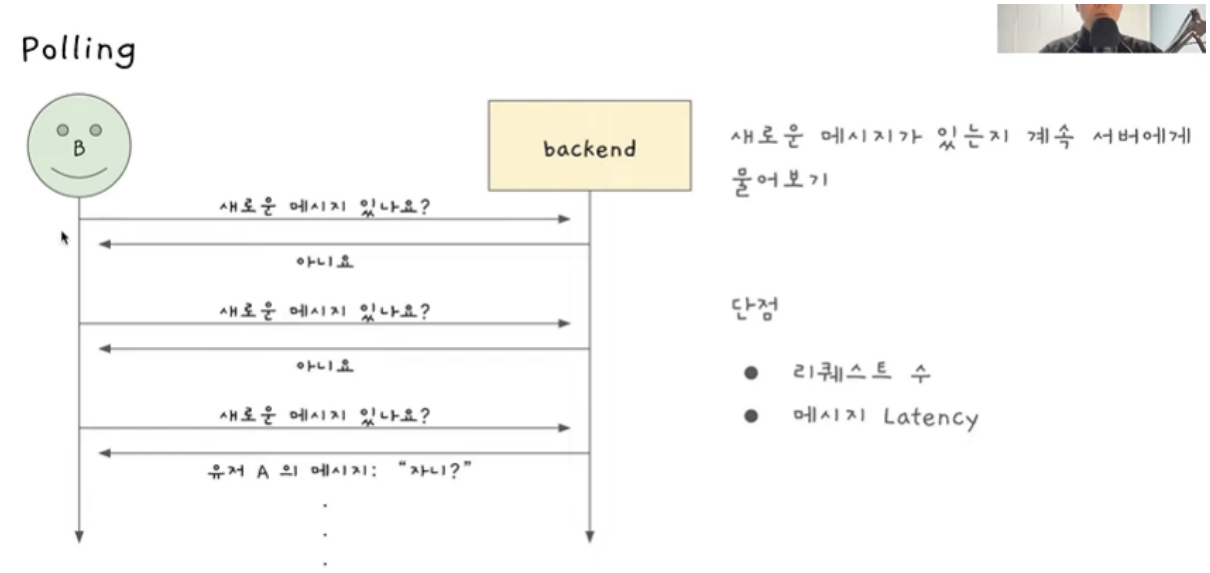


HTTP 커넥션의 문제

클라이언트가 요청을 보내는 구조라는 것

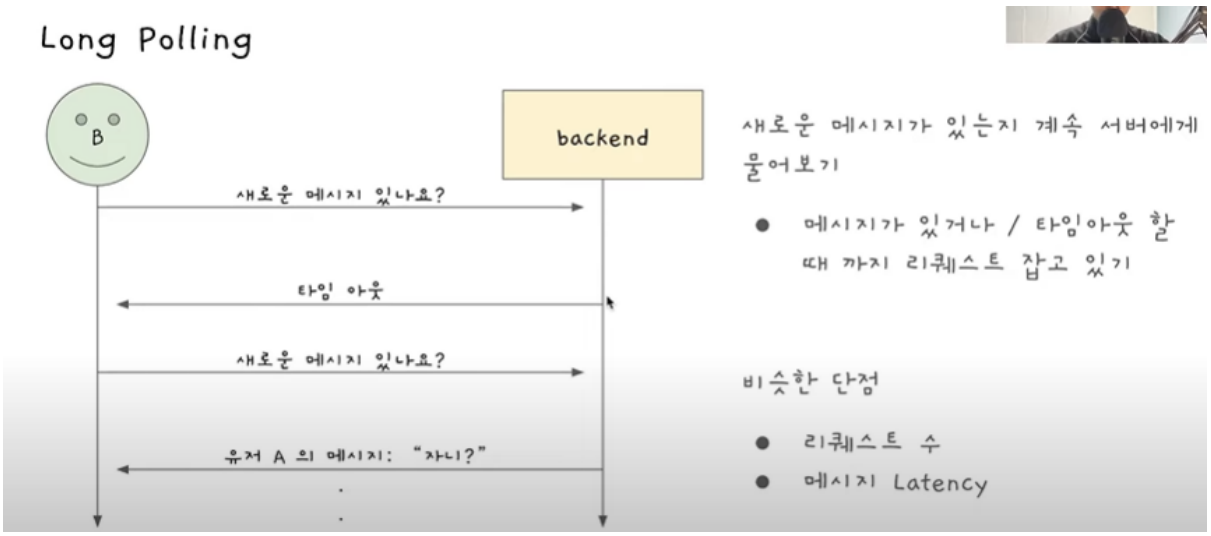
이 문제를 해결하는 몇 가지 옵션

- 폴링
- 롱 폴링
- 웹 소켓



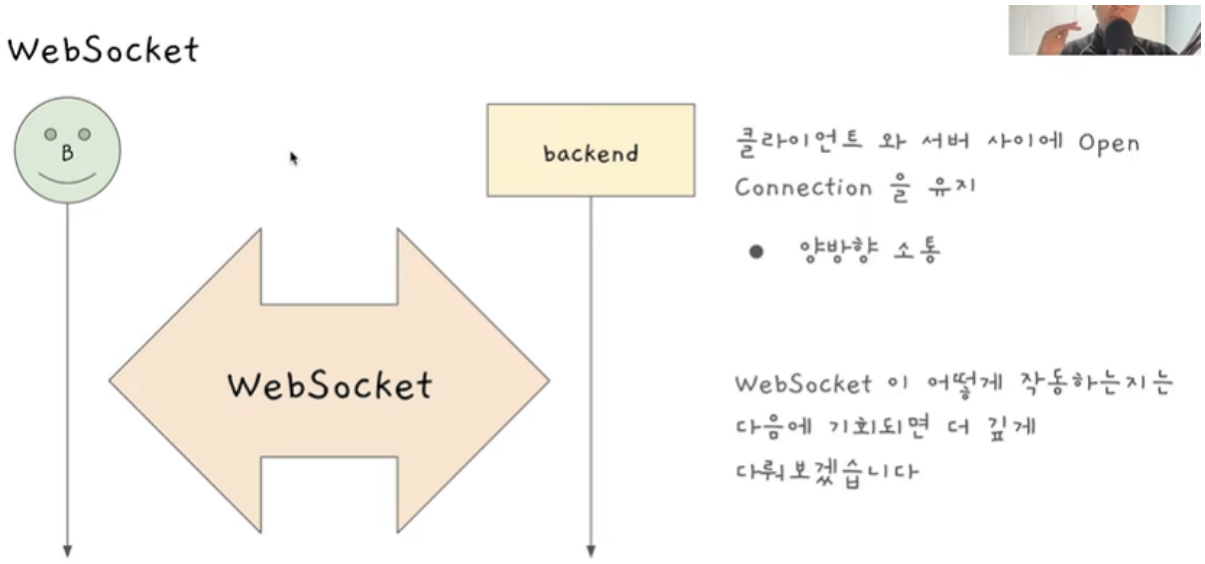
Polling : 새로운 메시지가 있는지 계속 서버에게 물어보기

- 대부분은 메시지가 없는 경우일 텐데 리퀘스트 수가 너무 많아진다. 1초 주기로 메시지를 확인한다면 레이턴시가 최소 1초인 것이다.



Long Polling : 메시지가 있거나 타임아웃할 때까지 리퀘스트 잡고 있기

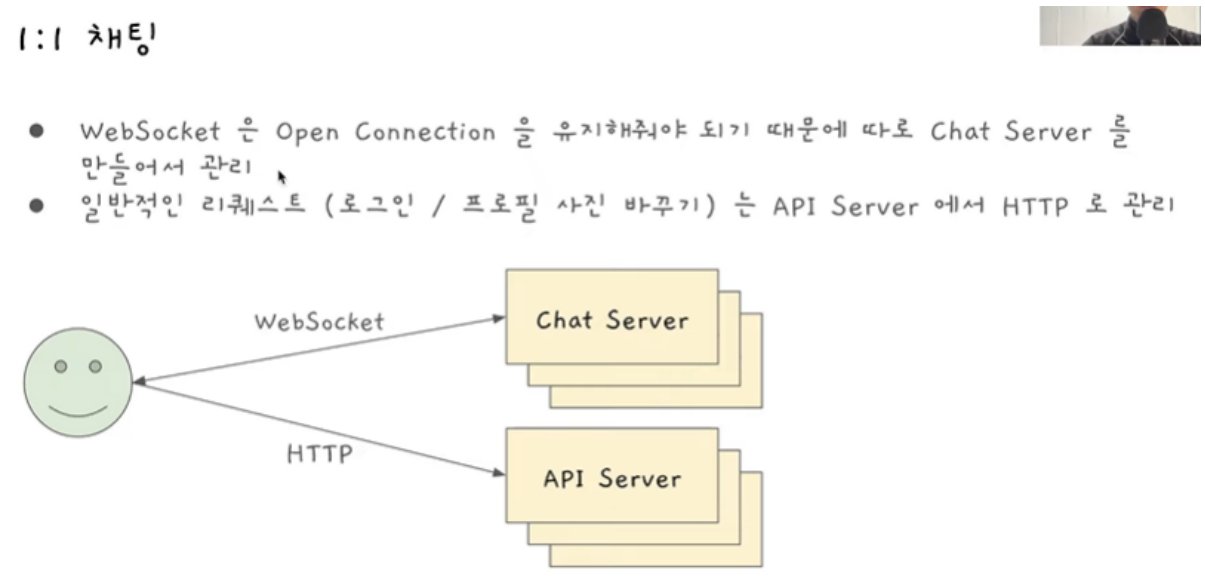
- Polling 보다는 리퀘스트 숫자가 줄어든다. 하지만 Polling과 비슷한 테크닉이다.



WebSocket : 클라이언트와 서버 사이에 Open Connection을 유지

- 양방향 소통

시스템 디자인에서 자주 사용되는 테크닉이기 때문에 한 번 개념을 잡고 간다.

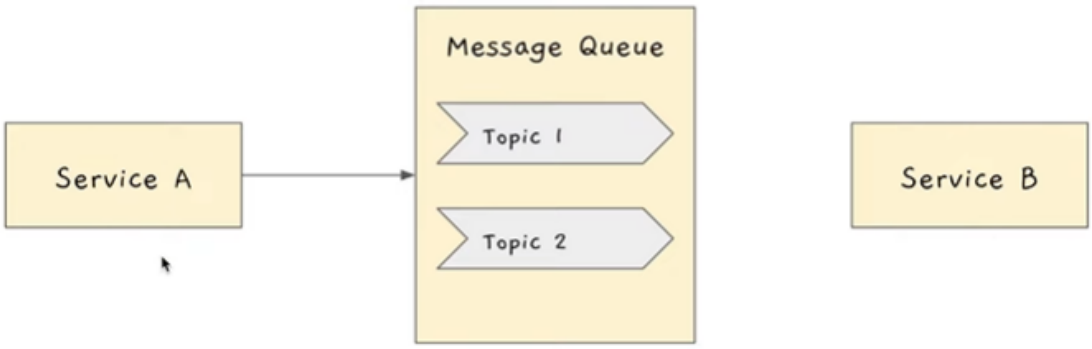


API Server에서 도메인 별로 마이크로 서버로 나눌 수도 있다. 하지만 여기서는 다루지 않겠다.

Message Queue

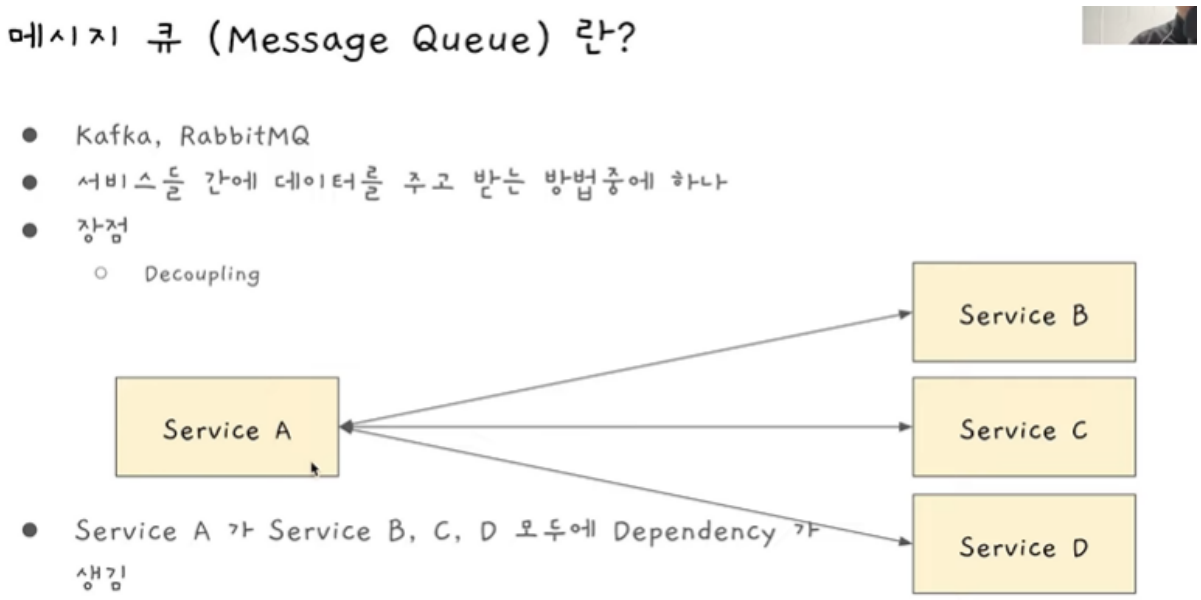
- Kafka, RabbitMQ
- 서비스들 간에 데이터를 주고 받는 방법 중에 하나
- RestAPI / RPc를 사용하면 Synchronous하게 데이터를 주고 받음

- 메시지 큐를 사용하면 Asynchronous하게 데이터를 처리할 수 있다.

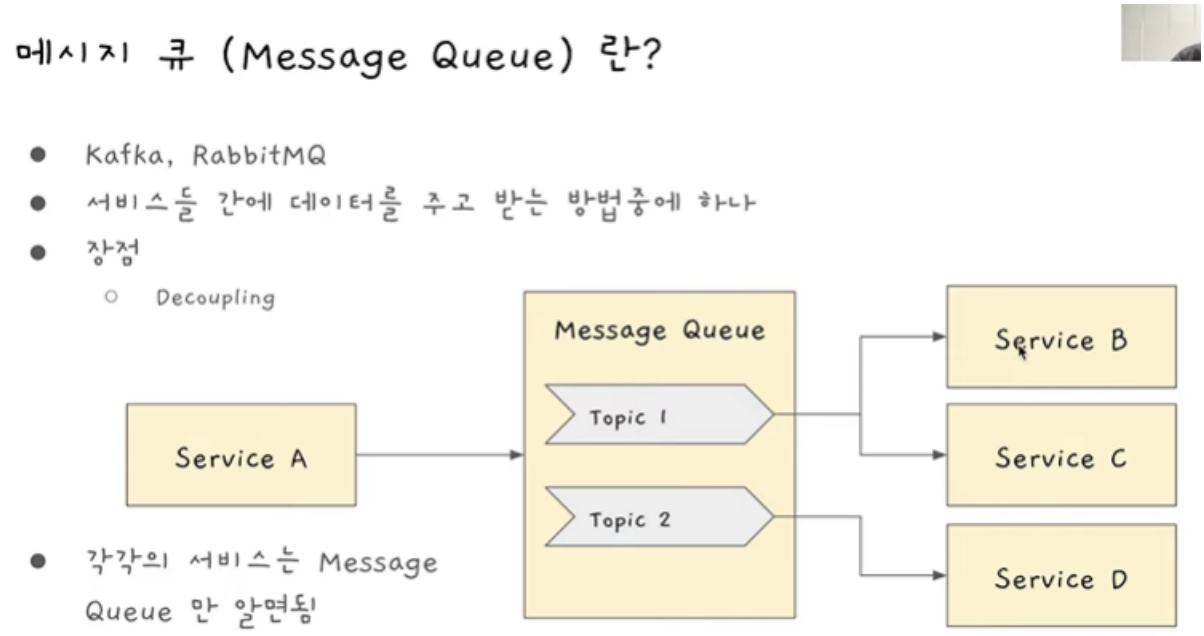


Service A = Publisher . Service B = Subscirber

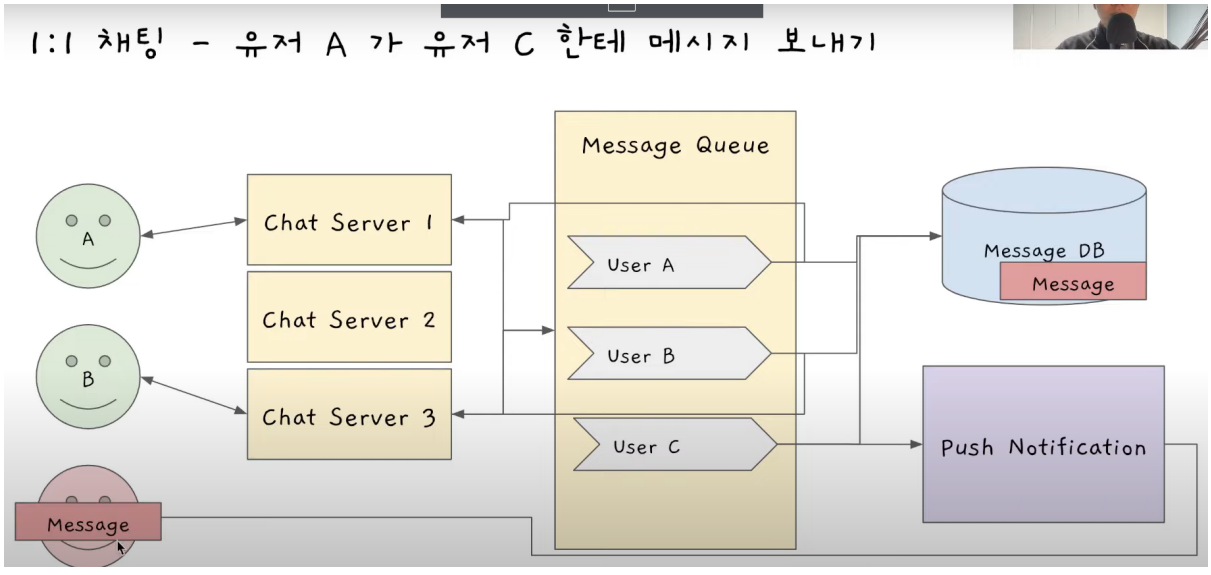
Service A는 메시지를 Message Queue에 보내고, Message Queue는 메시지를 Service B와 같은 Subscriber에게 메시지를 보낸다.



Rest API나 RPC를 사용해서 Synchronous하게 데이터를 주고 받게 되면 Service A에 관련 코드를 많이 넣어야 한다. 그런데 메시지 큐를 쓰면 Service A는 Service B, C, D와 같은 수신자에 대해 알고 있지 않아도 된다. Dependency가 줄어든다.



디커플링할 때 좋다. 메시지 큐는 따로 다룰 정도로 토픽이 크다.



DB 선택

DB는 어떤 종류의 DB를 써야 할까? 트래픽 특성이 중요하다.

트래픽 특성

- 엄청난 양의 트래픽 그룹 채팅 - 하루에 600억 메시지
- 오래된 채팅 잘 안 봄

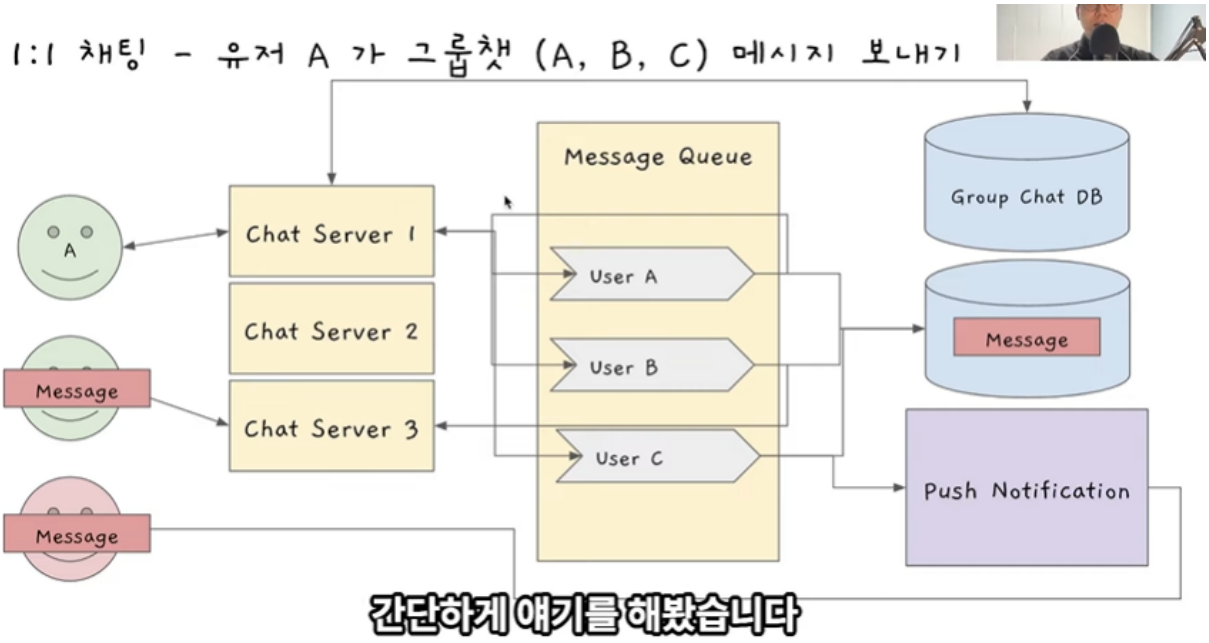
- Read / Write ratio 1:1
- 다른 데이터들과 join할 필요가 거의 없음
- RDB 같은 경우 데이터가 많아지고 index가 많아지면 느려진다.
    - 우리가 만들고자 하는 채팅 어플리케이션은 RDB의 장점을 활용하지 않는다.

Key Value Store

- 스케일 아웃하기가 편함
- Read Latency가 낮음
- Facebook 같은 경우 HBase, 디스코드는 카산드라를 사용함
- Key를 만들 때 Range Scan하기 쉽게 디자인 해야 함
    - 최근에 보낸 메시지일 수록 Key값이 높게 한다.
    - 시간 순서대로 채팅을 읽는 경우가 많기 때문이다. 메시지 아이디가 계속해서 올라가게만 만들면 된다.

그룹 챗 기능 추가

- 그룹 챗 몇 명까지 지원할 거냐에 따라 디자인이 달라질 수 있음
- 그룹 챗 최대 200명까지 지원한다고 가정을 한다면
    - 어느 정도 Fan out은 괜찮다.
        - *Fan out이란, 논리 회로에서 하나의 논리 게이트의 출력이 얼마나 많은 논리 게이트 입력으로 사용되었는지를 나타내는 말. 참고*
    - Chat Server가 Group Chat DB에서 그룹 챗에 참여하고 있는 유저 목록을 가져오고, 해당 유저들의 메시지 큐에 채팅 데이터를 넣는다. 하나의 서버가 여러 입력을 처리하기 때문에 `Fan out` 이라는 용어를 쓴 것으로 보인다.



프로젝트에 적용

Push Notification = 웹 푸시. FCM (Firebase Cloud Messaging) 이용. 어떻게 설계해야 하는가?

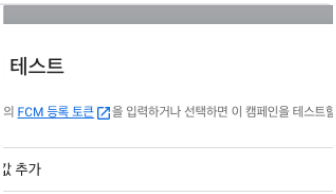
React 클라이언트가 메시지 수신을 대기하고 있다가 FCM 에서 메시지를 보내면 푸시 알림을 띄웁니다. 이때 브라우저의 Service Worker 가 백그라운드 상태에서 돌아가며 사이트를 꺼도 푸시알림을 띄울 수 있게 해줍니다. 그러니 React 클라이언트는 Service Worker 에 대한 설정도 같이 해줘야 합니다.

참고

[FCM] Firebase Cloud Messaging 으로 웹에서 푸시알림 구현하기 (React 클라이언트)

FCM이 뭔지는 일단 나중에... 자세한 것은 👉 Firebase Cloud Messaging Documentation을 따라하면 됩니다. 참고. 단순히 서버 실행 없이 http, js 파일로 실행했더니 아래처럼 뜨더라고요... 아래 오류는 알림 권한 요청을 차단했다는 메시지인데, file:// 도메인으로 시작하면 크롬 알림 설정이 아무리

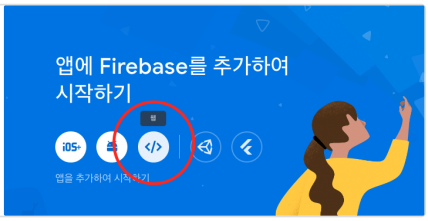
https://anywaydevlog.tistory.com/93



Web Push | React + FCM 구현하기 (feat. pwa, service worker)

웹에서도 네이티브 앱처럼 푸시 알람을 받을 수 있습니다. 웹의 사용성을 네이티브 앱처럼 개선하기 위해서 나온 기술인 PWA(Progressive Web Application)을 활용하면 가능합니다.PWA는 Progressive Web Application으로, 웹이 웹

https://velog.io/@heelieben/FCM-React-Web-Push-구현하기-feat.-pwa-service-worker



가상 면접 사례로 배우는 대규모 시스템 설계 기초 에서 배우는 채팅 시스템 설계

추후에 책 읽고 작성 예정