10장 알림 시스템 설계

알림 시스템은 고객에게 중요한 정보를 비동기적으로 제공하며, 그 종류에는 모바일 푸시 알림, SMS 메시지, 그리고 이메일이 있다.

1단계 문제 이해 및 설계 범위 확정

요구 사항

- 푸시알림, SMS, 이메일 알림 지원.
- 연성 실시간 (soft-real time) 시스템. 즉 최대한 빨리 전달되지만 시스템에 높은 부하가 걸렸을 때 약간의 지연은 무방함.
- iOS 단말, 안드로이드 단말, 랩탑/데스크탑 지원.
- 클라이언트 앱이 만들 수도, 서버 측에서 스케줄링 할 수도 있음.
- 사용자가 알림 받지 않도록 (opt-out) 설정 가능하도록 해야.
- 천만 건의 모바일 푸시 알림, 백만 건의 SMS 메시지, 5백만 건의 이메일 보낼 수 있어야 함.

2단계 개략적 설계안 제시 및 동의 구하기

알림 유형별 지원 전략

iOS 푸시 알림

- 알림 제공자 (provider) : 알림 요청을 만들어 애플 푸시 알림 서비스 (APNS) 로 보내는 주체. 다음의 데이터가 필요하다.
  - 단말 토큰 (device token) : 알림 요청을 보내는 데 필요한 고유 식별자.
  - 페이로드 (payload) : 알림 내용을 담은 JSON 딕셔너리.
- APNS : 애플이 제공하는 원격 서비스. 푸시 알림을 iOS 장치 로 보내는 역할.
- iOS 단말 (iOS device) : 푸시 알림을 수신하는 사용자 단말.

안드로이드 푸시 알림

- APNS 대신 FCM (Firebase Cloud Messaging) 을 사용한다.

SMS 메시지

- 트윌리오 (Twilio), 넥스모 (Nexmo) 같은 상용 서비스를 이용하므로 요금을 지불해야 한다.

이메일

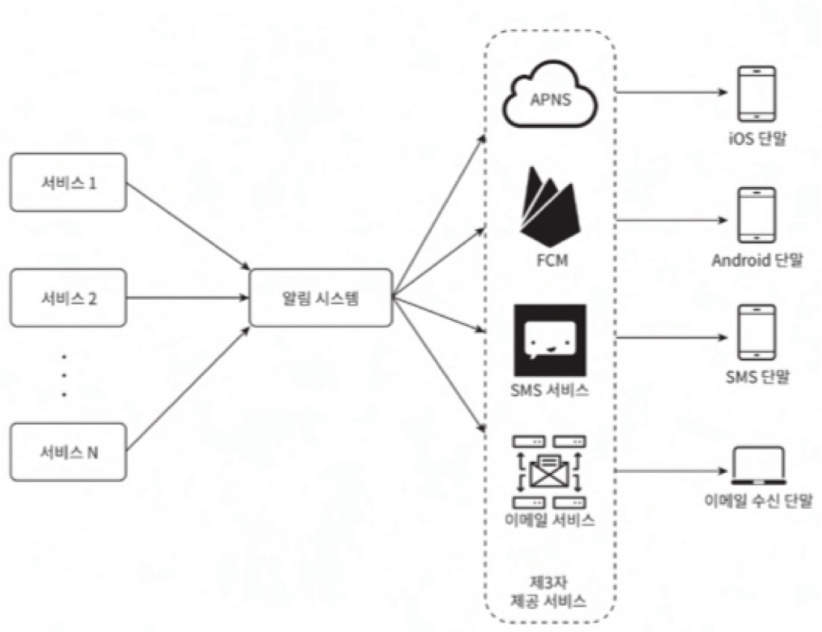
- 대부분 회사가 상용 이메일 서비스 이용. 대표적으로 센드그리드, 메일침프.

연락처 정보 수집 절차

- 알림 보내려면 모바일 단말 토큰, 전화번호, 이메일 주소 등의 정보 필요.
- 사용자가 앱을 설치하거나 계쩡 등록하려면 API 서버는 해당 사용자의 정보 수집하여 db에 저장.
- 이멩ㄹ 주소와 전화번호는 `user` 테이블에, 단말 토큰은 `device` 테이블에 저장.

알림 전송 및 수신 절차

개략적인 설계안



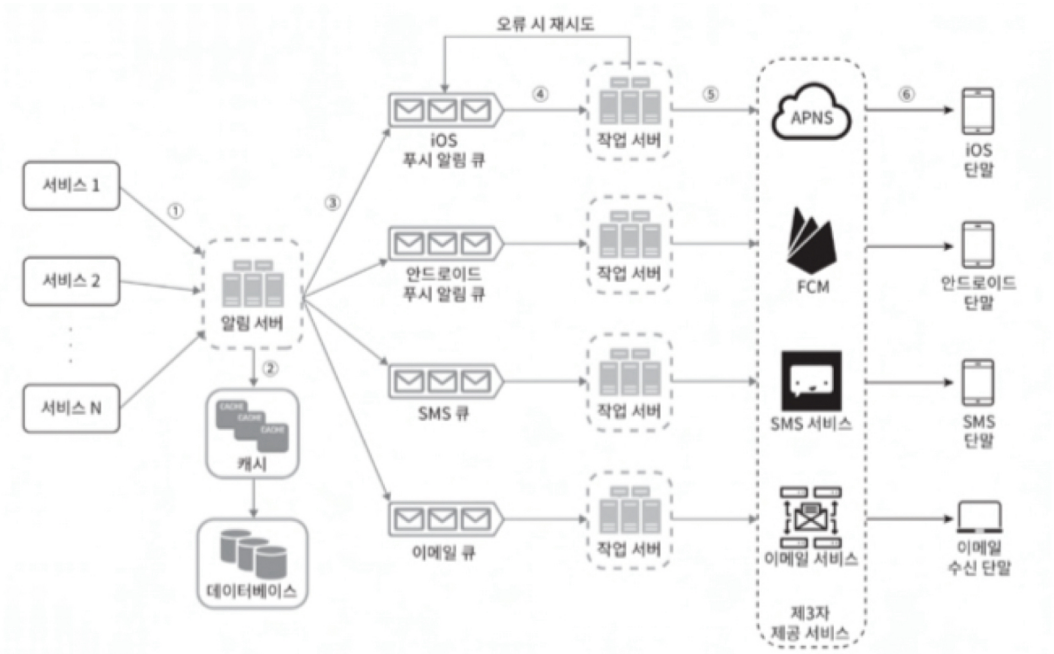
- 1-N 서비스
  - 마이크로 서비스, 크론잡 (cron job : 리눅스에서 작동하는 잡 스케줄러로, 특정한 동작이 미리 정의된 시간에 실행되도록 해준다) 혹은 분산시스템 컴포넌트일 수 있다.

    - 과금 서비스, 배송 알림 등이 그 예시.
- 알림 시스템
    - 1-N 서비스에 알림 전송을 위한 api를 제공하고 제3자 서비스에 전달할 알림 페이로드를 만들어낼 수 있어야 한다.
- 제 3자 서비스
    - 사용자들에게 알림을 실제로 전달하는 역할.
    - 확장성을 고려해야 한다.
    - 어떤 서비스는 다른 시장에서 사용할 수 없다는 점도 고려.

문제점

- SPOF : 서버가 하나밖에 없다.
- 규모 확장성 : 한 대 서비스로 푸시 알림 관련 모든 것을 처리하므로 DB 나 캐시 등 중요 컴포넌트의 규모를 개별적으로 늘릴 방법이 없다.
- 성능 병목 : 알림 처리하고 보내는 것은 자원 많이 필요할 수 있다. 한 대의 서버로 처리하면 트래픽 병목 현상 때문에 과부하 상태에 빠질 수 있다.

개선된 설계안



- DB와 캐시를 알림 시스템의 주 서버로부터 분리.
- 알림 서버 증설하고 수평적 확장 이루어질 수 있도록.

- 메시지 큐 이용해 시스템 컴포넌트 사이의 강한 결합 제거.
- 알림 서버의 기능
  - 알림 전송 API : 스팸 방지 위해 보통 사내 서비스 또는 인증된 클라이언트만 이용 가능.
  - 알림 검증 (validation) : 이메일 주소, 전화번호 등에 대한 기본적 검증.
  - 데이터베이스 또는 캐시 질의 : 알림에 포함시킬 데이터 가져오는 기능.
  - 알림 전송 : 알림 데이터를 메시지 큐에 넣는다. 여기서는 하나 이상의 큐를 사용하므로 알림을 병렬적으로 처리 가능.
- 캐시 : 사용자 정보, 단말 정보, 알림 템플릿 등을 캐시
- DB : 사용자, 알림, 설정 등 정보 저장
- 메시지 큐 : 시스템 컴포넌트 간 의존성 제거. 다량의 알림을 전송할 경우 버퍼 역할.
  - 서비스별로 별도의 큐를 사용하는 경우 해당 서비스에 장애가 발생하여도 다른 서비스는 정상 이용 가능.
- 작업 서버 : 메시지 큐에서 전송할 알림을 꺼내 제3자 서비스로 전달.

알림 전송 프로세스

1. API를 호출하여 알림 서버로 알림을 보낸다.
2. 알림 서버는 사용자 정보, 단말 토큰, 알림 설정 같은 메타데이터를 캐시나 DB에서 가져온다.
3. 알림 서버는 전송할 알림에 맞는 이벤트를 만들어 해당 이벤트를 위한 큐에 넣는다.
4. 작업 서버는 메시지 큐에서 알림 이벤트를 꺼낸다.
5. 작업 서버는 알림을 제 3자 서비스로 보낸다.
6. 제 3자 서비스는 사용자 단말로 알림을 전송한다.

3단계 상세 설계

안정성

데이터 손실 방지

- 알림이 지연되거나 순서가 바뀔 수는 있지만, 소실되면 절대 안된다.
- 알림 데이터를 DB에 보관하고 재시도 메커니즘을 구현해야 한다.

- 알림 로그 DB를 유지하는 것이 하나의 방법.

알림 중복 전송 방지

- 보내야 할 알림이 도착하면 해당 이벤트 ID 검사하여 이전에 본 적 있는 이벤트인지 확인
- 하지만, `You cannot have exactly once delivery`

추가로 필요한 컴포넌트 및 고려사항

알림 템플릿

- 알림의 유사성을 고려하여 인자, 스타일, 추적 링크 만을 조정하여 지정한 형식에 적용.

알림 설정

- 사용자가 알림을 상세이 조정할 수 있도록 해야.
- `channel`, `opt_in` 과 같은 필드들을 설정.

전송률 제한

- 한 사용자가 받을 수 있는 알림의 빈도를 제한하자. 알림을 아예 꺼버릴 수도 있으므로.

재시도 방법

- 제 3자 서비스가 알림 전송 실패시 해당 알림을 재시도 전용 큐에 넣고 같은 문제 반복시 alert.

푸시 알림과 보안

- iOS와 안드로이드의 경우, appKey 와 appSecret 을 사용하여 API 보안 유지.

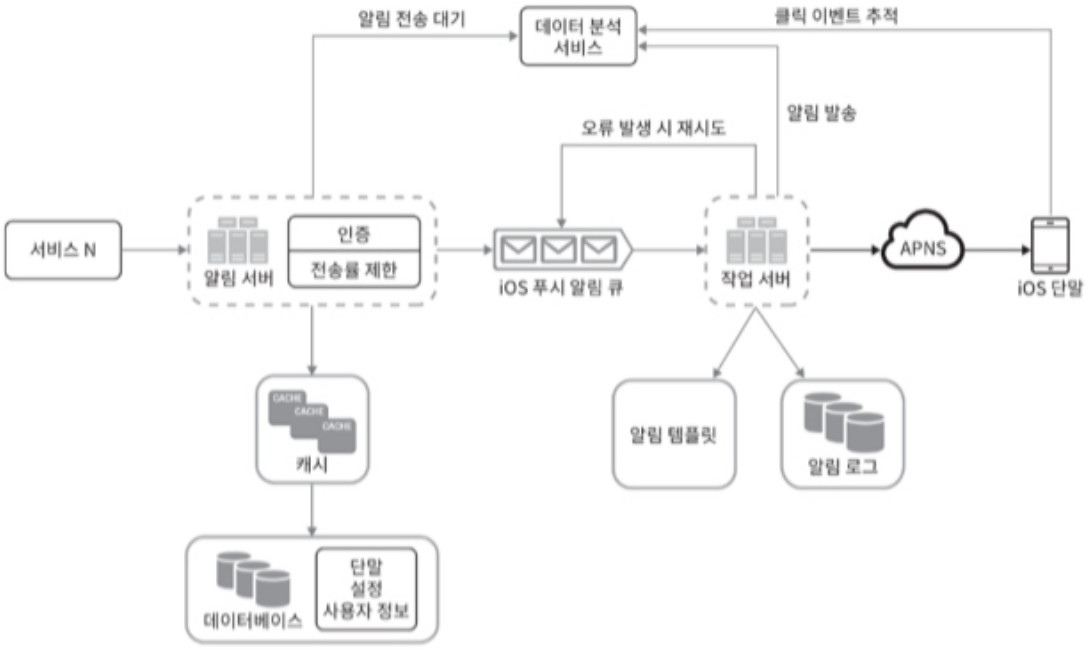
큐 모니터링

- 큐에 쌓인 알림이 매우 많으면 이벤트를 빠르게 처리하고 있지 못하다는 뜻. 서버를 증설해야.

이벤트 추적

- 알림 확인율, 클릭율, 실제 앱 사용으로 이어지는 비율 등을 통해 사용자를 이해할 수 있다.
- 데이터 분석 서비스를 통해 이벤트 추적.

수정된 설계안



- 안정성 : 메시지 전송 실패율을 낮추기 위해 안정적인 재시도 메커니즘을 도입.
- 보안 : 인증된 클라이언트만이 알림을 보낼 수 있도록 appKey, appSecret 등의 메커니즘 이용.
- 이벤트 추적 및 모니터링 : 알림을 만든 후 성공적으로 전송되기까지 과정을 추적하고 시스템 상태를 모니터링하기 위해 알림 전송의 각 단계마다 이벤트를 추적하고 모니터링 할 수 있는 시스템을 통합.
- 사용자 설정 : 사용자가 알림 수신 설정을 조정할 수 있도록 함.