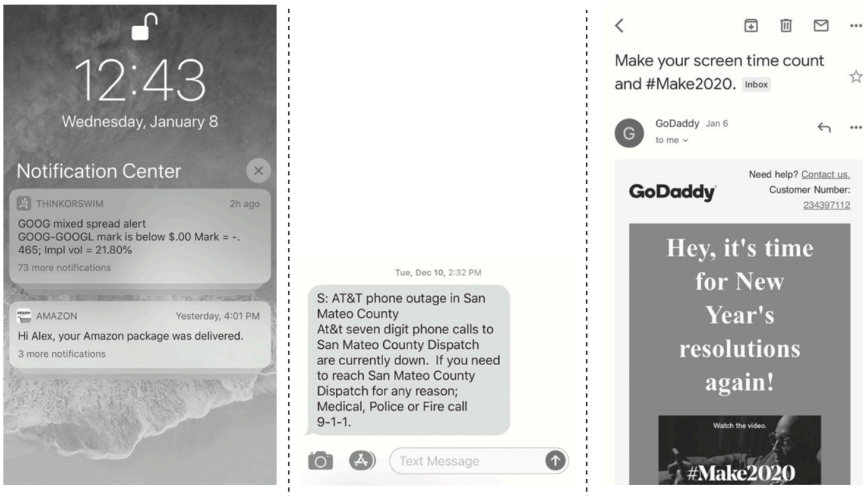
10장 - 알림 시스템 설계

- 알림 시스템은 단순히 모바일 푸시 알림 뿐 아니라, SMS 메시지, 이메일등 세 가지로 분류할 수 있다.



푸시 알림 SMS 이메일

그림 10-1

1단계 : 문제 이해 및 설계 범위 확장

지원자: 이 시스템은 어떤 종류의 알림을 지원해야 하나요?

면접관: 푸시 알림, SMS 메시지, 그리고 이메일입니다.

지원자: 실시간(real-time) 시스템이어야 하나요?

면접관: 연성 실시간(soft real-time) 시스템이라고 가정합니다.
알림은 가능한 한 빨리 전달되어야 하지만 시스템이 높은 부하가 걸렸을 때 지연은 무방합니다.

지원자: 어떤 종류의 단말을 지원해야 하나요?

면접관: IOS 단말, 안드로이드(android) 단말, 랩톱/데스크톱을 지원해야 합니다.

지원자: 사용자에게 보낼 알림은 누가 만들 수 있나요?

면접관: 클라이언트 애플리케이션 프로그램이 만들 수도 있구요.
서버 측에서 스케줄링 할 수도 있습니다.

지원자: 사용자가 알림을 받지 않도록(opt-out) 설정할 수도 있어야 하나요?

면접관: 네. 해당 설정을 마친 사용자는 더 이상 알림을 받지 않습니다.

지원자: 하루에 몇 건의 알림을 보낼 수 있어야 하나요?

면접관: 천만 건의 모바일 푸시 알림, 백만 건의 SMS 메시지, 5백만 건의 이메일을 보낼 수

2단계 : 개략적 설계안 제시 및 동의 구하기

알림 유형별 지원 방안

iOS 푸시 알림

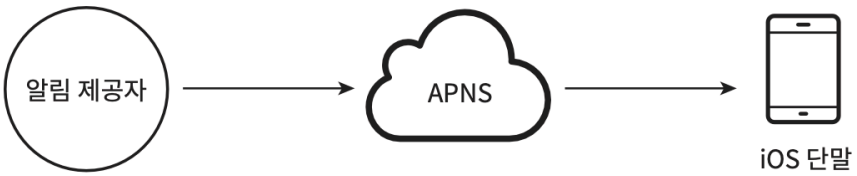


그림 10-2

- **알림 제공자(provider)**
  - **알림 요청**을 만들어 애플 푸시 알림 서비스(APNS: Apple Push Notification Service)로 보내는 주체
  - 알림 요청을 만들기 위해 필요한 데이터
    - **단말 토큰(device token)** : 알림 요청을 보내는 데 필요한 고유 식별자
    - **페이로드(payload)** : 알림 내용을 담은 JSON 딕셔너리(dictionary)

```
ex)
{
   "aps": {
      "alert": {
         "title": "Game Request",
         "body": "Bob wants to play chess",
         "action-loc-key": "PLAY"
      },
      "badge": 5
   }
}
```

- **APNS** : 애플이 제공하는 원격 서비스. 푸시 알림을 iOS 장치로 보내는 역할 담당
- iOS 단말(iOS device) : 푸시 알림을 수신하는 사용자 단말

안드로이드 푸시 알림

그림 10-3

- 안드로이드 푸시 알림도 비슷하다. 단, APNS 대신 FCM(Firebase Cloud Messaging)을 사용한다.

SMS 메시지

그림 10-4

- SMS 메시지를 보낼 때는 보통 트윌리오[1], 넥스모[2] 같은 제 3사업자의 서비스를 많이 사용한다.

이메일

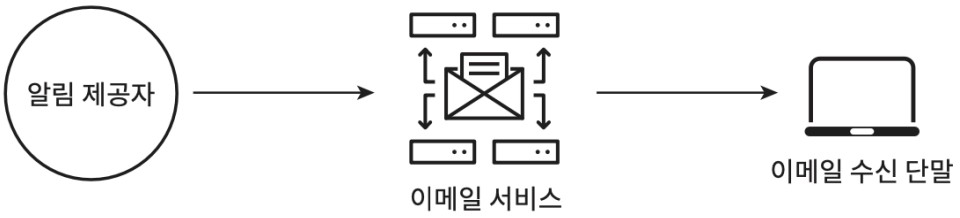


그림 10-5

- 유명한 서비스로 샌드그리드[3], 메일침프[4]가 있다.
  - 자체적으로 구축하기보다 상용 서비스를 사용하면 전송 성공률도 높고, 데이터 분석 서비스도 제공한다.

연락처 정보 수집 절차

- 알림을 보내려면 모바일 단말 토큰, 전화번호, 이메일 주소 등의 정보가 필요하다. 따라서, 다음 그림과 같이 사용자가 우리 앱을 설치하거나 처음으로 계정을 등록하면 API 서버는 해당 사용자의 정보를 수집하여 데이터베이스에 저장한다.

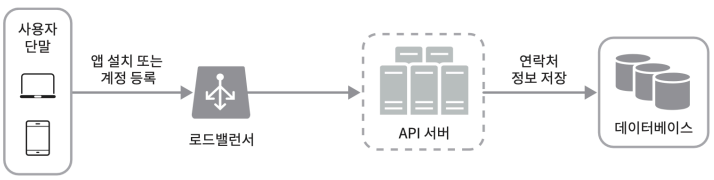


그림 10-7

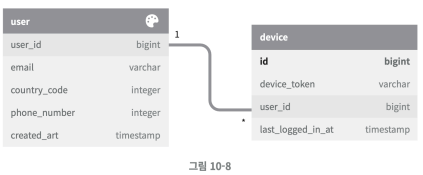


그림 10-8

연락차 정보를 저장할 테이블 구조

알림 전송 및 수신 절차

개략적 설계안 (초안)

- **1부터 N까지의 서비스**
  - 이 서비스 각각은 **마이크로서비스** 일 수도 있고, **크론 잡** 일 수도 있고, **분산 시스템 컴포넌트**일 수도 있다.
  - ex) 납기일을 알리는 과금 서비스(billing service), 배송 알림을 보내는 쇼핑몰 웹 사이트
- **알림 시스템(notification system)**
  - 알림 시스템은 알림 전송/수신 처리의 핵심
  - 1개 서버만 사용하는 시스템이라면 이 시스템은 서비스 1~N에 알림 전송을 위한 API를 제공해야 하고,
    제 3자 서비스에 전달할 알림 페이로드(payload)를 만들어 낼 수 있어야 한다.
- **제 3자 서비스(third party services)**
  - 사용자에게 알림을 실제로 전달하는 역할
  - 3자 서비스와 통합을 진행할 때 확장성을 고려해야 한다.
    - 쉽게 새로운 서비스를 통합하거나 기존 서비스를 제거할 수 있어야 함
    - 또한 어떤 서비스는 다른 시장에서 사용할 수 없다. (ex. FCM은 중국에서 사용할 수 없음)
- **iOS, 안드로이드, SMS, 이메일 단말**
  - 사용자는 자기 단말에서 알림을 수신

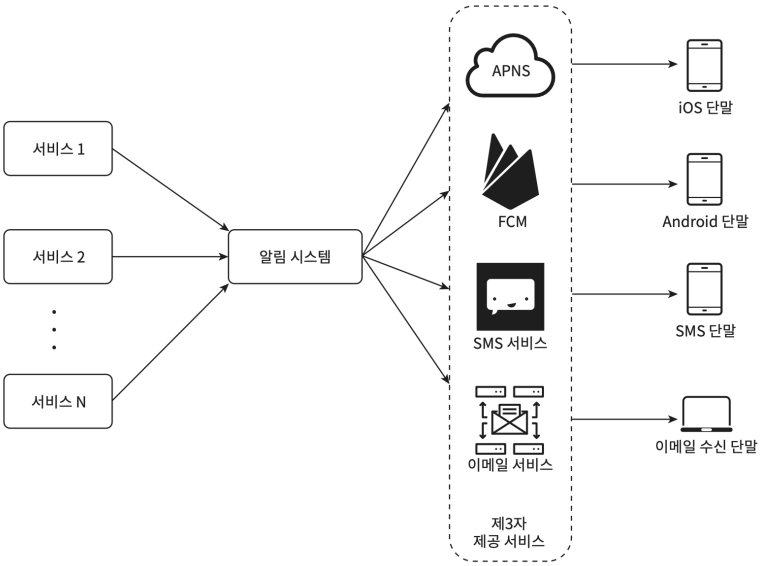


그림 10-9

개략적 설계안(초안)의 문제점

- **SPOF**
  - 알림 서비스에 서버가 하나밖에 없다는 것은 해당 서버에 장애가 생기면 전체 서비스의 장애로 이어짐
- **규모 확장성**
  - 한대 서비스로 푸시 알림을 관계된 모든 것을 처리하므로 데이터베이스나 캐시 등 중요 컴포넌트들의 규모를 개별적으로 늘릴 방법이 없음
- **성능 병목**
  - 알림을 처리하고 보내는 것은 자원을 많이 필요로 하는 작업일 수 있음
  - ex) HTML 페이지를 만들고 제 3자 서비스의 응답을 기다리는 일은 시간이 많이 걸릴 가능성이 있는 작업, 따라서 모든 것을 한 서버로 처리하면 사용자 트래픽이 많이 몰리는 시간에는 시스템이 과부하 상태에 빠질 수 있음

개략적 설계안 (개선된 버전)

- 데이터베이스와 캐시를 알림 시스템의 주 서버에서 분리
- 알림 서버를 증설하고 자동으로 수평적 규모 확장이 이루어질 수 있도록

- 메시지 큐를 이용해 시스템 컴포넌트 사이의 강한 결합을 끊음

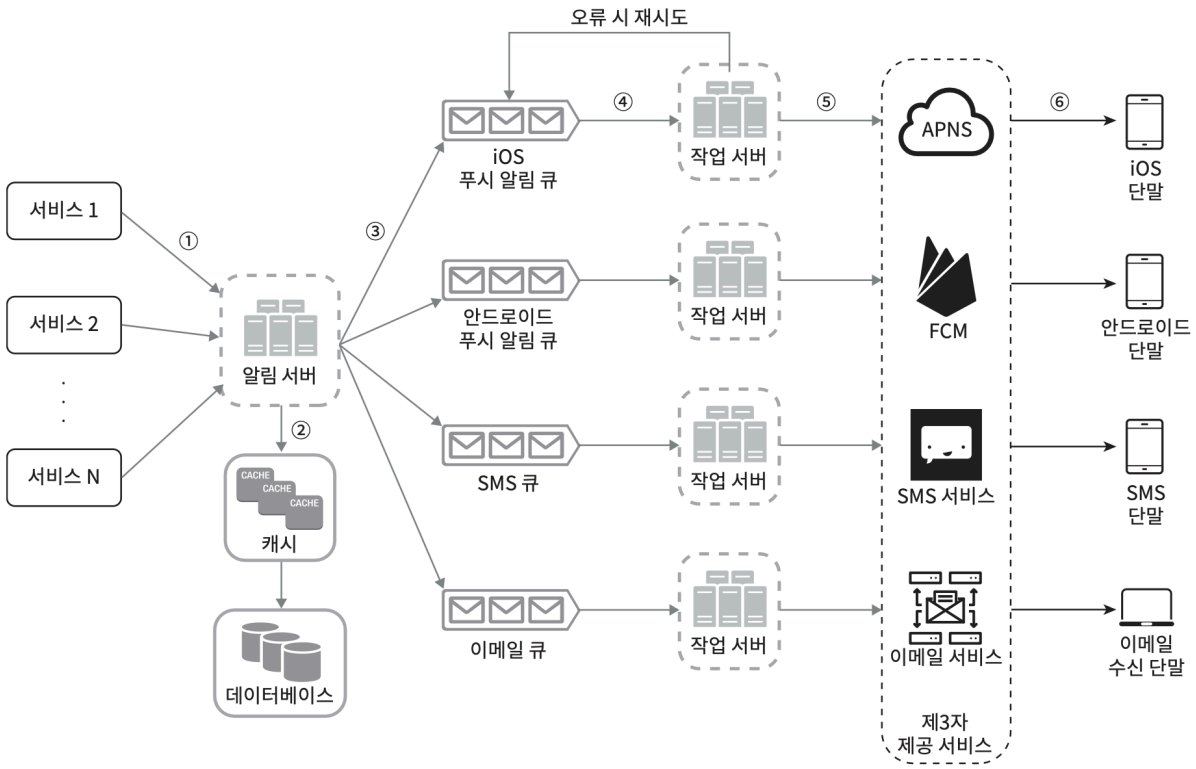


그림 10-10

▼ 1부터 N까지의 서비스

- 알림 시스템 서버의 API를 통해 알림을 보낼 서비스들

▼ 알림 서버 (notification server) 가 담당하는 기능

- **알림 전송 API** : 스팸 방지를 위해 보통 사내 서비스 또는 인증된 클라이언트만 이용 가능
- **알림 검증(validation)** : 이메일 주소, 전화번호 등 기본적 검증 수행
- **데이터베이스 또는 캐시 질의** : 알림에 포삼시킬 데이터를 가져오는 기능
- **알림 전송** : 알림 데이터를 메시지 큐로 전송, **하나 이상의 메시지 큐를 사용하므로 병렬로 처리** 가능

▼ 캐시(cache)

- 사용자 정보, 단말 정보, 알림 템플릿(template) 등을 캐시

▼ 데이터베이스(DB)

- 사용자, 알림, 설정 등 다양한 정보를 저장

▼ 메시지 큐(message queue)

- 시스템 컴포넌트 간 의존성을 제거하기 위해 사용
- 다량의 알림이 전송되어야 하는 경우 대비한 버퍼 역할도 함
- 해당 설계에서는 알림의 종류별로 별도 메시지 큐를 사용
  3자 서비스 가운데 하나에 장애가 발생해도 다른 종류의 알림은 정상 동작

▼ 작업 서버(workers)

- 메시지 큐에서 전송할 알림을 꺼내서 제3자 서비스로 전달하는 역할을 담당하는 서버

▼ 제 3자 서비스(third party services) (초안과 같음)

- 사용자에게 알림을 실제로 전달하는 역할
- 3자 서비스와 통합을 진행할 때 확장성을 고려해야 한다.
  - 쉽게 새로운 서비스를 통합하거나 기존 서비스를 제거할 수 있어야 함
  - 또한 어떤 서비스는 다른 시장에서 사용할 수 없다. (ex. FCM은 중국에서 사용할 수 없음)

▼ iOS, 안드로이드, SMS, 이메일 단말 (초안과 같음)

- 사용자는 자기 단말에서 알림을 수신

실제 동작 예시

1. API를 호출하여 알림 서버로 알림을 보낸다.
2. 알림 서버는 사용자 정보, 단말 토큰, 알림 설정 같은 메타데이터를 캐시나 데이터베이스에서 가져온다.
3. 알림 서버는 전송할 알림에 맞는 이벤트를 만들어서 해당 이벤트를 위한 큐에 넣는다.
   ex) iOS 푸시 알림 이벤트는 iOS 푸시 알림 큐에 넣는다.
4. 작업 서버는 메시지 큐에서 알림 이벤트를 꺼낸다.
5. 작업 서버는 알림을 제3자 서비스로 보낸다.
6. 제3자 서비스는 사용자 단말로 알림을 전송한다.

3단계 : 상세 설계

안정성

데이터 손실 방지

- 어떤 상황에서도 알림이 소실되면 안 됨 (지연되거나 순서가 틀려도 괜찮은데 사라지면 안 됨)
- 이를 만족하려면 **알림 데이터를 데이터베이스에 보관하고 재시도 메커니즘을 구현**해야 한다.

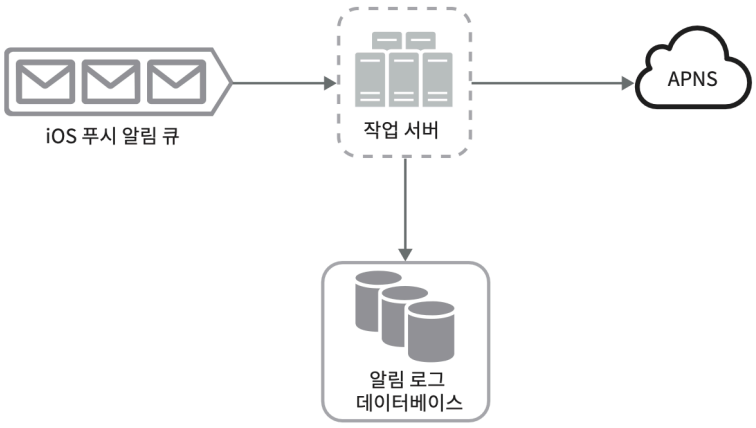


그림 10-11

알림 로그 데이터베이스를 통해 재시도 매커니즘 구현

  - Transactional Outbox Pattern과 비슷해 보이는데?

알림 중복 전송 방지

- 분산 시스템의 특성상 가끔은 같은 알림이 중복되어 전송되기도 한다.
- 이 빈도를 줄이기 위해 중복을 탐지하는 매커니즘을 도입하고 오류를 신중히 처리해야 한다.
- 중복 방지 로직의 사례
  - 보내야 할 알림이 도착하면 그 이벤트 ID를 검사하여 이전에 본 적이 있는 이벤트인지 살핀다.
  - 중복된 이벤트라면 버리고, 그렇지 않으면 알림을 발송한다.
  - 중복 전송을 100% 방지할 수 없다면 [5]를 참고하자.

추가로 필요한 컴포넌트 및 고려사항

알림 템플릿

- 알림 템플릿은 인자나 스타일, 추적 링크를 조정하기만 하면 지정한 형식에 맞춰 알림을 만들어 내는 틀이다.

> 간단한 예제)
>
> 본문:
> 여러분이 꿈꿔운 그 상품을 우리가 준비했습니다. [item_name]이 다시 입고되었습니다!
> [date]까지만 주문 가능합니다!
>
> 타이틀(CTA: Call to Action):
> 지금 [item_name]을 주문 또는 예약하세요!

- 이런 식으로 템플릿을 사용하면 전송될 알림들의 형식을 일관성 있게 유지할 수 있다.

알림 설정

- 알림이 많을 경우 쉽게 피곤함을 느낄 수 있다.
  따라서 웹과 앱에서는 사용자가 알림 설정을 상세히 조정할 수 있도록 하고 있다.
- 이 정보는 알림 설정 테이블에 보관되며, 이 테이블에는 다음과 같은 필드가 필요할 것이다.

```
user_id     bigInt
channel     varchar     # 알림이 전송될 채널. 푸시 알림, 이메일, SMS 등
opt_in      boolean     # 해당 채널로 알림을 받을 것인지의 여부
```

전송률 제한

- 알림을 제한하기 위해 사용자가 받을 수 있는 알림의 빈도를 제한하는 방법이 있다.

재시도 방법

- 제3자 서비스가 알림 전송에 실패하면 해당 알림을 재시도 전용 큐에 넣는다.
- 같은 문제가 계속해서 발생하면 개발자에게 통지

푸시 알림과 보안

- iOS와 안드로이드 앱의 경우, 알림 전송 API는 `appKey`와 `appSecret`을 사용하여 보안을 유지한다.
- 즉, **인증된(authenticated)** 혹은 **승인된(verified)** 클라이언트만 **알림을 보낼 수** 있다.

큐 모니터링

- 알림 시스템을 모니터링할 때 중요한 메트릭 중 하나는 **큐에 쌓인 알림의 개수를 확인**해 보는 것이다.
- 이 수가 너무 크면 작업 서버들이 이벤트를 올바로 처리하지 못한다는 뜻이다. 이럴 땐 서버를 증설해야 함 [7]

이벤트 추적

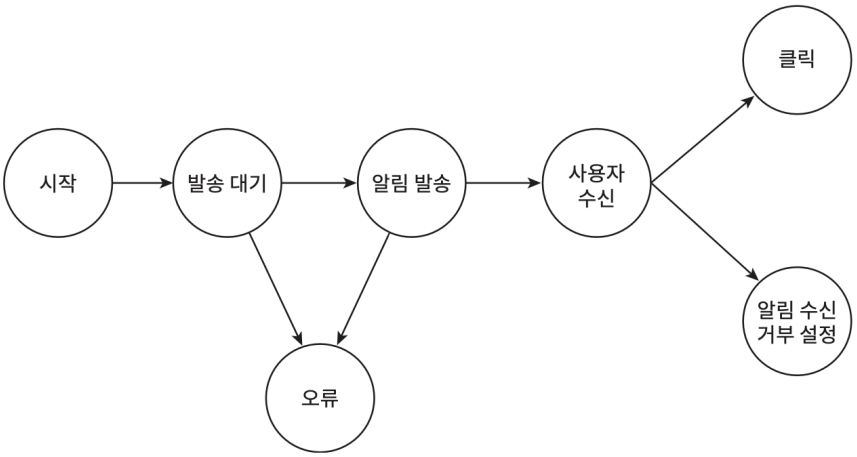


그림 10-13

- 알림 확인율, 클릭율, 실제 앱 사용으로 이어지는 비율 같은 메트릭은 사용자를 이해하는 데 중요하다.
- 데이터 분석 서비스(analytics)는 보통 이벤트 추적 기능도 제공한다.
- 보통 **알림 시스템을 만들면 데이터 분석 서비스와도 통합**해야만 한다.

개선된 설계안

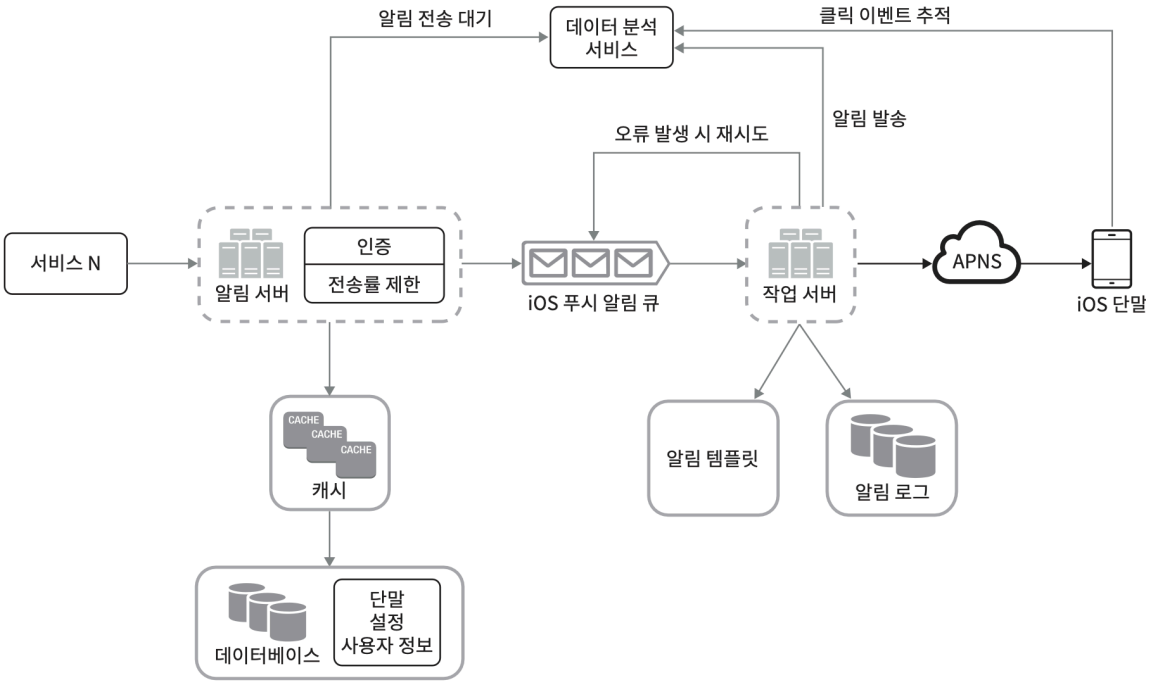


그림 10-14

지금까지 설명한 내용이 모두 반영된 수정된 설계안

- 알림 서버에 인증(authentication)과 전송률 제한(rate-limiting) 기능이 추가되었다.
- 전송 실패에 대응하기 위한 재시도 기능이 추가되었다.
  - 전송에 실패한 알림은 다시 큐에 넣고 지정된 횟수만큼 재시도한다.
- 전송 템플릿을 사용하여 알림 생성 과정을 단순화하고 알림 내용의 일관성을 유지힌다.
- 모니터링과 추적 시스템을 추가하여 시스템 상태를 확인하고 추후 시스템을 개선하기 쉬도록 만들었다.

4단계 : 마무리

- 규모 확장성, 다양한 정보 전달 방식을 지원하는 알림 시스템을 설계했다.
  이때 시스템 컴포넌트 사이의 결합도를 낮추기 위해 메시지 큐를 적극적으로 사용했다.
- 개략적 설계안과 더불어 각 컴포넌트의 구현 방법과 최적화 기법에 대해 살펴보고, 다음의 주제에 집중했다.
  - **안정성(reliability)**
    - 메시지 전송 실패율을 낮추기 위해 안정적인 재시도 매커니즘 도입
  - **보안(security)**
    - 인증된 클라이언트만이 알림을 보낼 수 있도록 appKey, appSecret 등의 매커니즘 이용
  - **이벤트 추적 및 모니터링**
    - 알림이 만들어진 후 성공적으로 전송되기까지의 과정을 추적하고 시스템 상태를 모니터링하기 위해 알림 전송의 각 단계마다 이벤트를 추적하고 모니터링할 수 있는 시스템을 통합
  - **사용자 설정**
    - 사용자가 알림 수신 설정을 조정할 수 있도록 설정
    - 따라서 알림을 보내기 전 반드시 해당 설정을 확인하도록 시스템 설계 변경
  - **전송률 제한**
    - 사용자에게 알림을 보내는 빈도를 제한할 수 있도록

Chapter 10: DESIGN A NOTIFICATION SYSTEM

1. Twilio SMS
2. Nexmo SMS
3. Sendgrid
4. Mailchimp
5. You Cannot Have Exactly-Once Delivery
6. Security in Push Notifications
7. Key metrics for RabbitMQ monitoring