6주차

10장. 알림 시스템 설계

알림 시스템

- 모바일 푸시 알림
- SMS 메시지
- 이메일

1단계 - 문제 이해 및 설계 범위 확정

알림 시스템에 대한 문제가 면접에 출제될 경우, 정해진 정답은 없고 문제 자체가 모호하게 주어짐 → 적절한 질문을 통해 요구 사항이 무엇인지 지원자 스스로 알아내야 함

책에서 구현해야 하는 알림 시스템

- 푸시 알림, SMS 메시지, 이메일
- 연성 실시간 시스템
- ios 단말, 안드로이드 단말, 랩톱/데스크톱 지원
- 클라이언트 어플리케이션 or 서버 측에서 스케줄링하여 알림 생성
- 사용자가 알림을 받지 않도록 설정 가능
- 하루 천만 건의 모바일 푸시 알림, 백만 건의 SMS 메시지, 5백만 건의 이메일 전송

2단계 - 개략적 설계안 제시 및 동의 구하기

- 알림 유형별 지원 방안
- 연락처 정보 수집 절차
- 알림 전송 및 수신 절차

알림 유형별 지원 방안

1) ios 푸시 알림

ios에서 푸시 알림을 보내기 위해서는 세가지 컴포넌트가 필요함

- 알림 제공자: 알림 요청을 만들어 애플 푸시 알림 서비스(APNS)로 보내는 주체, 알림 요청을 만들기 위해서는 아래의 데이터가 필요
    - 단말 토큰: 알림 요청을 보내는 데 필요한 고유 식별자
    - 페이로드: 알림 내용을 담은 JSON 딕셔너리
- APNS: 애플에 제공하는 원격 서비스, 푸시 알림을 ios 장치로 보내는 역할을 담당
- ios 단말: 푸시 알림을 수신하는 사용자 단말

2) 안드로이드 푸시 알림

APNS 대신 FCM을 사용

3) SMS 메시지

트윌리오, 넥스모 등 제 3사업자의 서비스를 많이 사용 → 상용 서비스이므로 이용 요금 지불

4) 이메일

센드그리드, 메일침프 등의 상용 서비스를 사용 → 전공 성공률 ↑ + 데이터 분석 서비스 제공

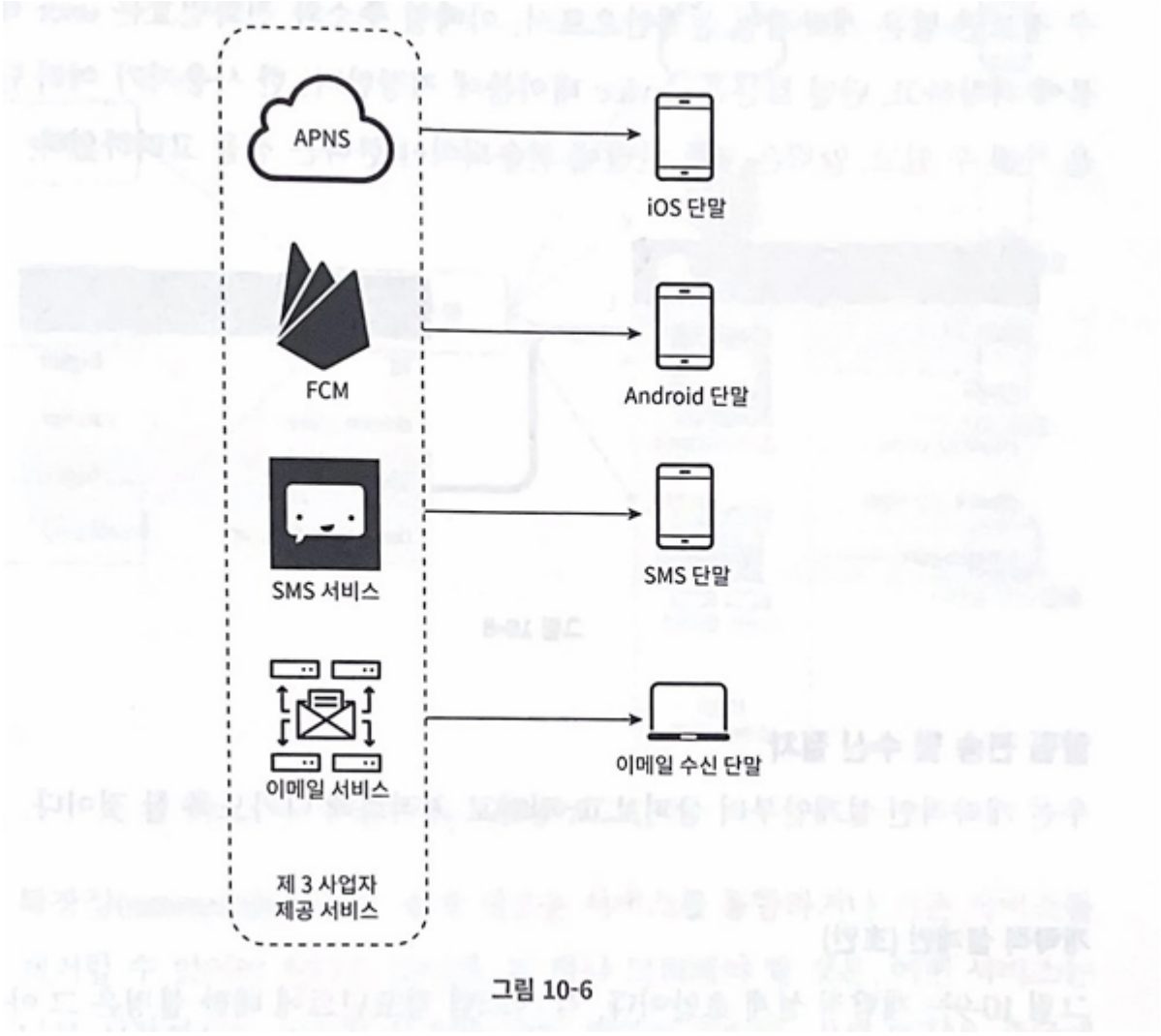

그림 10-6

연락처 정보 수집 절차

알림을 보내기 위해서는 모바일 단말 토큰, 전화번호, 이메일 주소 등의 정보가 필요 → 사용자가 앱을 설치하거나 처음으로 계정을 등록하면 API 서버는 해당 사용자의 정보를 수집, 데이터베이스에 저장

데이터베이스 설계 시, 한 사용자가 여러 단말을 가질 수 있고, 알림이 모든 단말에 전송되야 함을 고려

알림 전송 및 수신 절차

개략적 설계안(초안)

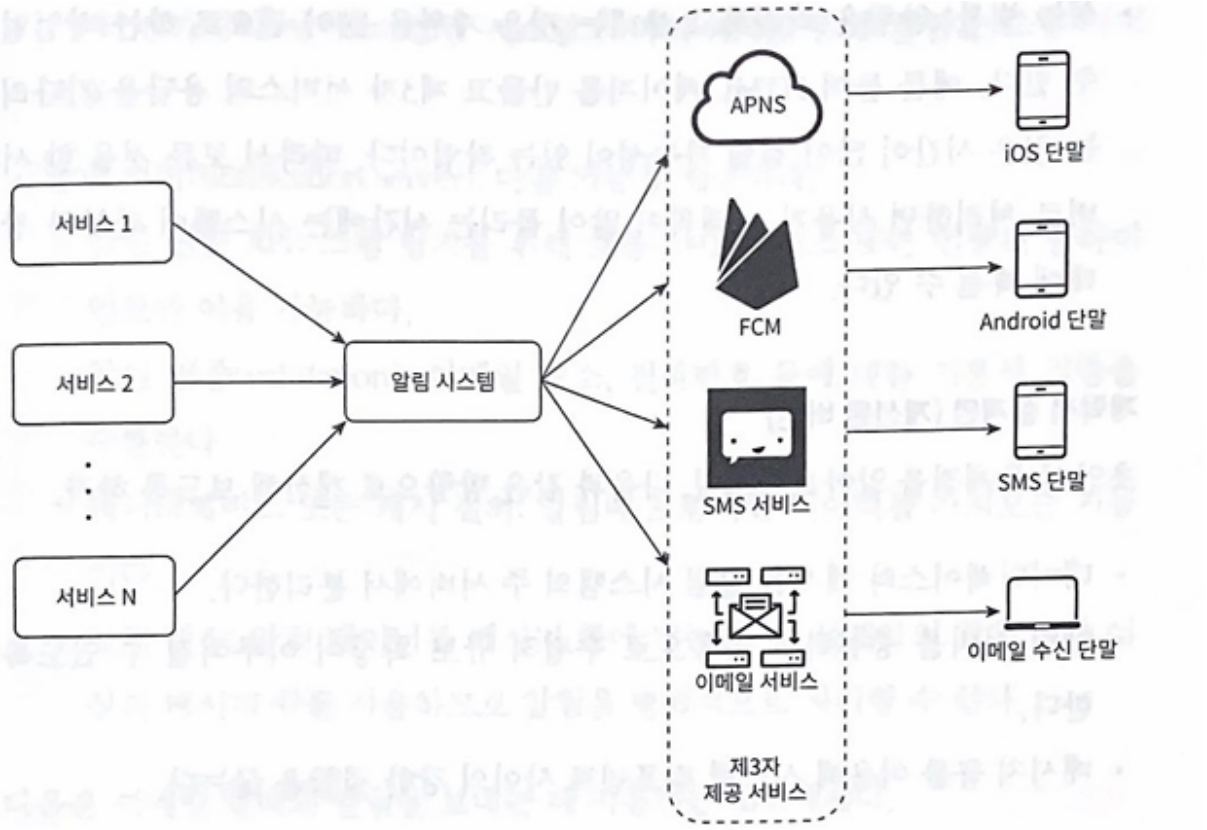

그림 10-9

각 시스템 컴포넌트 설명

- 1부터 N까지의 서비스: 서비스 각각은 마이크로서비스 일 수도, 크론잡일 수도, 분산 시스템 컴포넌트일 수도 있음
    - 사용자에게 납기일을 알리고자 하는 과금 서비스, 배송 알림을 보내려는 쇼핑몰 웹사이트 등이 예시
- 알림 시스템: 알림 전송/수신 처리의 핵심. 시스템은 서비스 1~N에 알림 전송을 위한 API를 제공해야 하고, 제 3자 서비스에 전달할 알림 페이로드를 만들어낼 수 있어야 함
- 제3자 서비스: 사용자에게 알림을 실제로 전달하는 역할. 제3자 서비스와의 통합 진행 시 확장성 + 어떤 서비스는 다른 시장에서 사용할 수 없음을 고려
- ios, 안드로이드, SMS, 이메일 단말: 사용자는 자기 단ㅇ말에서 알림을 수신

문제점

- SPOF: 알림 서비스에 서버가 하나밖에 없음 → 서버 장애 발생 시, 전체 서비스의 장애로 이어짐

- 규모 확장성: 한 대 서비스로 푸시 알림에 관계된 모든 것을 처리 → 데이터베이스 or 캐시 등 중요 컴포넌트의 규모를 개별적으로 늘릴 방법이 없음
- 성능 병목: 알림을 처리하고 보내는 것은 자원을 많이 필요로 하는 작업일 수 있음 → 모든 것을 한 서버로 처리하면 사용자 트래픽이 많이 몰리는 시간에는 시스템이 과부하 상태에 빠질 수 있음

개략적 설계안(개선된 버전)

개선 방안

- 데이터베이스와 캐시를 알림 시스템의 주 서버에서 분리
- 알림 서버를 증설 + 자동으로 수평적 규모 확장이 이루어질 수 있도록 함
- 메시지 큐를 이용, 시스템 컴포넌트 사이의 강한 결합 끊기

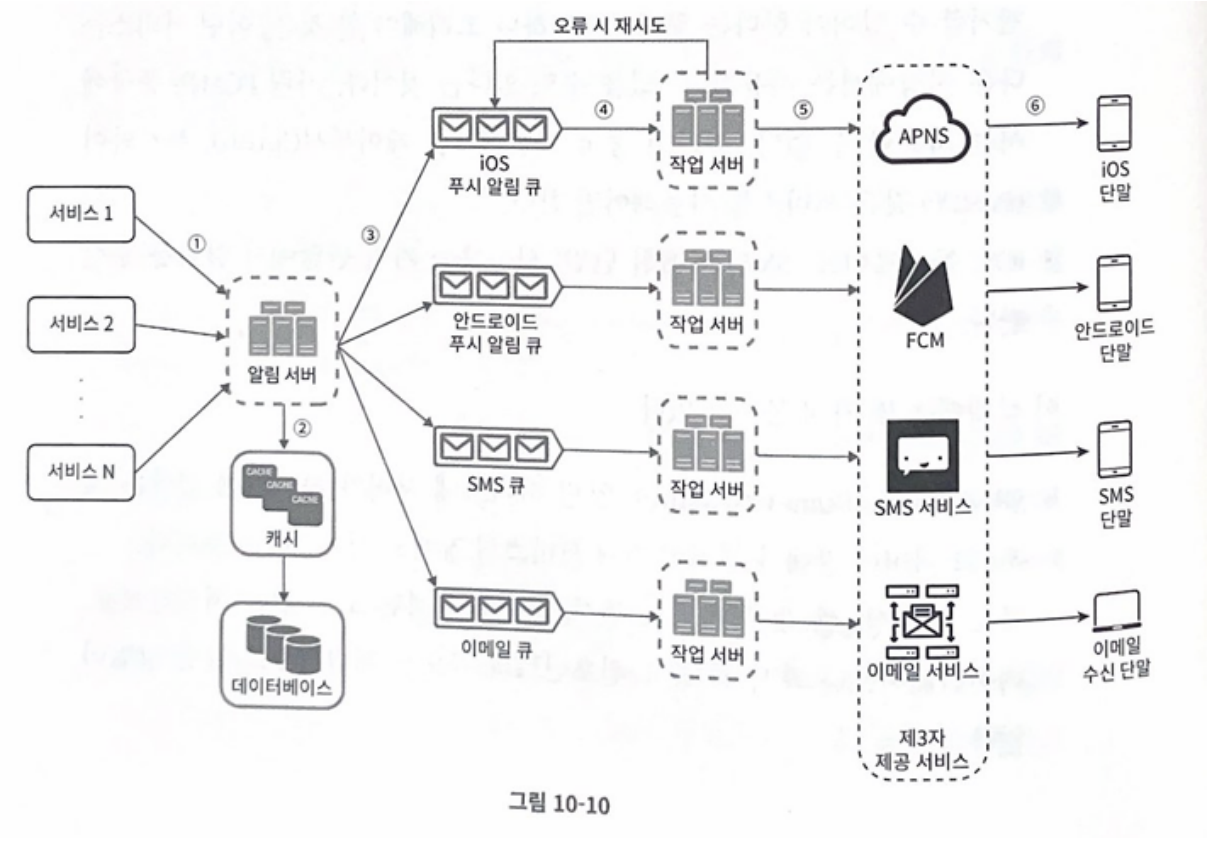

그림 10-10

- 1부터 N까지의 서비스: 알림 시스템 서버의 API를 통해 알림을 보낼 서비스들
- 알림 서버: 다음 기능을 제공

    - 알림 전송 API: 스팸 방지를 위해 보통 사내 서비스 또는 인증된 클라이언트만 이용 가능
    - 알림 검증: 이메일 주소, 전화번호 등에 대한 기본적 검증을 수행
    - 데이터베이스 또는 캐시 질의: 알림에 포함시킬 데이터를 가져오는 기능
    - 알림 전송: 알림 데이터를 메시지 큐에 넣음 → 해당 설계안에서는 하나 이상의 메시지 큐를 사용, 알림을 병렬적으로 처리 O
- 캐시: 사용자 정보, 단말 정보, 알림 템플릿 등을 캐시
- 데이터베이스: 사용자, 알림, 설정 등 다양한 정보를 저장
- 메시지 큐: 시스템 컴포넌 간 의존성 제거를 위해 사용, 다양한 알림ㅇ 전송되어야 하는 경우를 대비한 버퍼 역할도 수행. → 해당 설계안에서는 알림의 종류별로 별도 메시지 큐를 사용, SPOF 발생 X
- 작업 서버: 메시지 큐에서 전송할 알림을 꺼내서 제3자 서비스로 전달하는 역할을 담당
- 제 3자 서비스: 설계 초안을 이야기할 때 이미 설명
- ios, 안드로이드, SMS, 이메일 단말

알림 전송 과정

1. API를 호출하여 알림 서버로 알림을 전송
2. 알림 서버는 사용자 정보, 단말 토큰, 알림 설정 같은 메타 데이터를 캐시나 데이터베이스에서 가져옴
3. 알림 서버는 전송할 알림에 맞는 이벤트를 만들어서 해당 이벤트를 위한 큐에 넣음
4. 작업 서버는 메시지 큐에서 알림 이벤트를 꺼냄
5. 작업 서버는 알림을 제 3자 서비스로 보냄
6. 제 3자 서비스는 사용자 단말로 알림을 전송

3단계 - 상세 설계

아래 3가지를 더 자세히 살펴본다

- 안전성

- 추가로 필요한 컴포넌트 및 고려 사항: 알림 템플릿, 알림 설정, 전송률 제한, 재시도 메커니즘, 보안, 큐에 보관된 알림에 대한 모니터링과 이벤트 추적
- 개선된 설계안

안전성

분산 환경에서 운영될 알림 시스템 설계 시, 안전성 확보를 위한 사항 몇 가지를 반드시 고려

1) 데이터 손실 방지

알림은 어떠한 상황에서도 소실되면 안됨 → 지연되거나 순서가 틀리는 것은 괜찮지만, 사라지면 곤란함

- 알림 시스템은 알림 데이터를 데이터 베이스에 보관하고 재시도 메커니즘을 구현

2) 알림 중복 전송 방지

분산 시스템의 특성상 가끔은 같은 알림이 중복되어 전송되기도 함 → 빈도를 줄이기 위해서는 중복 탐지 메커니즘 도입 + 오류 신중히 처리

ex) 보내야 할 알림이 도착 → ID를 검사하여 이전에 본 적이 있는 이벤트인지 확인 → 중복이면 버리고, 아니면 알림을 발송

추가로 필요한 컴포넌트 및 고려 사항

- 알림 템플릿
- 알림 설정
- 이벤트 추적
- 시스템 모니터링
- 처리율 제한

1) 알림 템플릿

알림 메시지 형식의 유사성을 고려, 알림 메시지의 모든 부분을 처음부터 다시 만들지 않게 하도록 해주는 컴포넌트 → 인자, 스타일, 추적 링크를 조장하면 사전에 지정한 형식에 맞춰

알람을 만들어내는 틀

템플릿을 사용하면 전송될 알림의 형식을 일관성 있게 유지 가능 + 알림 작성에 드는 시간 줄임

2) 알림 설정

사용자가 알림 설정을 상세히 조정할 수 있도록 함 → 설정 정보는 알림 설정 테이블에 보관

테이블에 필요한 필드

- user_id bigint
- channel varchar
- opt_in boolean

이런 설정을 도입한 후에는 특정 종류의 알림을 보내기 전, 반드시 사용자가 해당 알림을 켜 두었는지 확인해야 함

3) 전송률 제한

사용자가 아예 알림 기능을 꺼버리는 상황을 피하기 위해 사용자에게 너무 많은 알림을 보내지 않도록 한 사용자가 받을 수 있는 알림의 빈도를 제한

4) 재시도 방법

제 3자 서비스가 알림 전송에 실패하면 해당 알림을 재시도 전용 큐에 넣음 → 같은 문제가 계속해서 발생하면 개발자에게 통지

5) 푸시 알림과 보안

ios와 안드로이드 앱의 경우 알림 전송 API는 appKey와 appSecret을 사용하여 보안을 유지 → 인증된, 혹은 승인된 클라이언트만 해당 API를 사용하여 알림 보내기 가능

6) 큐 모니터링

큐에 쌓인 알림의 수는 시스템 모니터링 시의 중요한 메트릭이 됨

ex) 큐에 쌓인 알림의 수가 너무 클 경우 작업 서버들이 이벤트를 빠르게 처리하고 있지 못하다는 뜻

7) 이벤트 추적

알림 확인율, 클릭율, 실제 앱 사용으로 이어지는 비율 같은 메트릭은 사용자를 이해할 때 중요 → 데이터 분석 서비스는 이벤트 추적 기능도 제공

알림 시스템을 만들기 위해서는 데이터 분석 서비스와도 통합해야 함

수정된 설계안

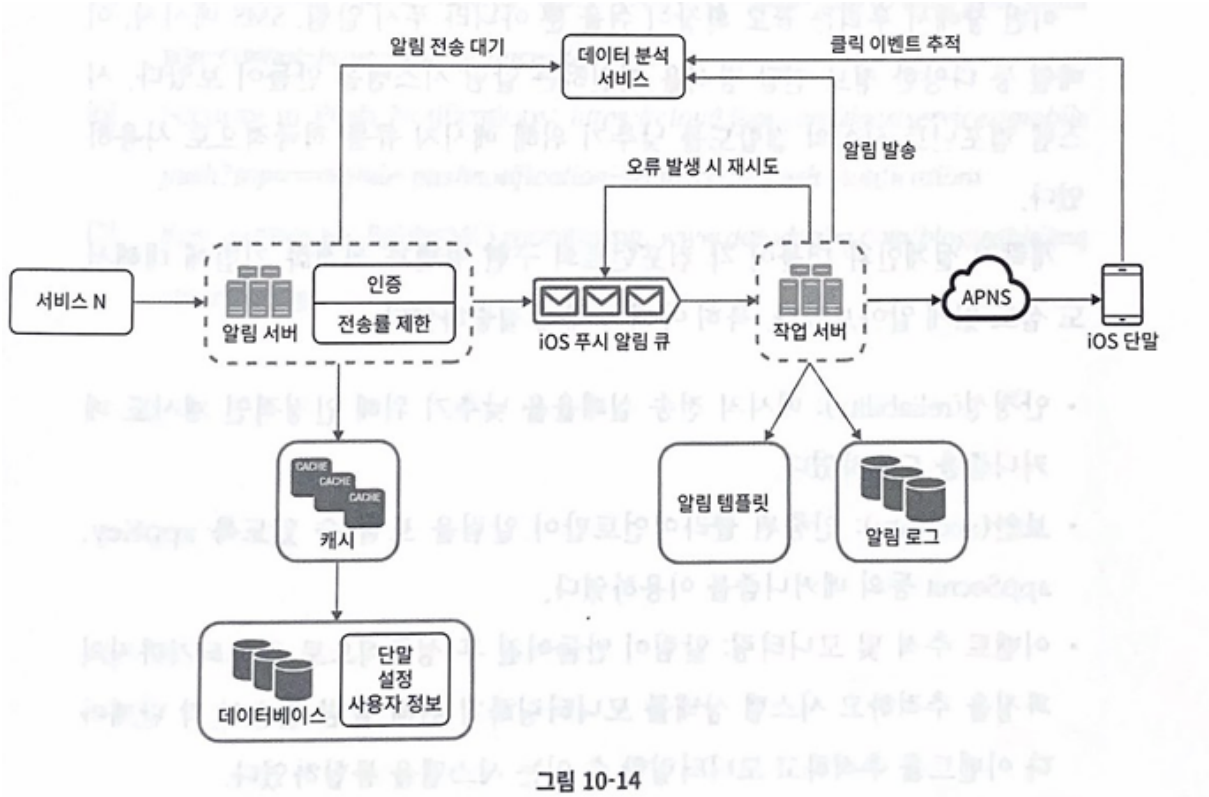

그림 10-14

추가된 컴포넌트

- 알림 서버에 인증과 전송률 제한 기능이 추가
- 전송 실패에 대응하기 위한 재시도 기능이 추가 → 전송에 실패한 알림은 다시 큐에 넣고 지정된 횟수 만큼 재시도

- 전송 템플릿을 사용, 알림 생성 과정을 단순화하고 알림 내용의 일관성을 유지
- 모니터링과 추적 시스템을 추가, 시스템 상태 확인 + 추후 시스템 개선하기 쉽게 함

4단계 - 마무리

알림은 중요 정보를 계속 알려준다는 점에서 필요 불가결한 기능

책에서는 개략적 설계안과 더불어 각 컴포넌트의 구현 방법 + 최적화 기법에 대해 심도 있게 살펴봄 → 특히 집중한 주제

- 안전성: 메시지 전송 실패율을 낮추기 위해 안정적인 재시도 메커니즘 도입
- 보안: 인증된 클라이언트 만이 알림을 보낼 수 있도록 appKey, appSecrete 등의 메커니즘을 이용
- 이벤트 추적 및 모니터링: 알림이 만들어진 후 성공적으로 전송되기까지의 각 과정 추적 + 시스템 상태 모니터링을 위해 알림 전송의 각 단계마다 이벤트 추적 및 모니터링할 수 있는 시스템 통합
- 사용자 설정: 사용자가 알림 수신 설정을 조정할 수 있도록 함 → 알림을 보내기 전 반드시 해당 설정을 확인하도록 시스템 설계를 변경
- 전송률 제한: 사용자에게 알림을 보내는 빈도를 제한할 수 있도록 함

11장. 뉴스 피드 시스템 설계

뉴스피드란?

사용자의 홈 페이지 중앙에 지속적으로 업데이트 되는 스토리들로, 상요자 상태 정보 업데이트, 사진, 비디오, 링크, 앱 활동 및 사용자가 팔로우하는 사람들, 페이지 혹은 그룹으로부터 나오는 좋아요를 포함함.

1단계 - 문제 이해 및 설계 범위 확정

면접관의 의도 - 최소한 어떤 기능을 지원해야 하는지는 반드시-를 파악해야 함

책에서 설계하는 뉴스 피드 시스템

- 모바일 앱과 웹 지원
- 사용자는 뉴스 피드 페이지에 새로운 스토리를 올릴 수 있고, 친구들이 올리는 스토리를 볼 수 있어야 함
- 뉴스 피드는 시간 흐름 역순으로 스토리가 표시
- 최대 5000명의 친구를 가짐
- 매일 천만 명이 방문
- 스토리에는 이미지나 비디오 등의 미디어 파일 포함 가능

2단계 - 개략적 설계안 제시 및 동의 구하기

설계안은 피드 발생과 뉴스 피드 생성의 두 부분으로 나뉨

- 피드 발행: 사용자가 스토리를 포스팅하면 해당 데이터를 캐시와 데이터 베이스에 기록. 새 포스팅은 친구의 뉴스 피드에도 전송됨
- 뉴스 피드 생성: 지면 관계상 뉴스 피드는 모든 친구의 포스팅을 시간 흐름 역순으로 모아서 생성

뉴스 피드 API

클라이언트가 서버와 통신하기 위해 사용하는 수단

HTTP 프로토콜 기반 → 상태 정보 업데이트, 뉴스 피드 가져오기, 친구 추가 등 다양한 작업 수행에 사용

뉴스 피드 API 종류

- 피드 발행 API
- 피드 읽기 API

피드 발행 API

새 스토리를 포스팅하기 위한 API

HTTP POST 형태로 요청을 전송. 아래와 같은 형태를 띔

```
POST /v1/me/feed
```

인자

- 바디: 포스팅 내용에 해당
- Authorization 헤더: API 호출을 인증하기 위해 사용됨

피드 읽기 API

뉴스 피드를 가져오는 API

다음과 같은 형태를 띔

```
GET /v1/me/feed
```

인자

- Authorization 헤더: API 호출을 인증하기 위해 사용

피드 발행

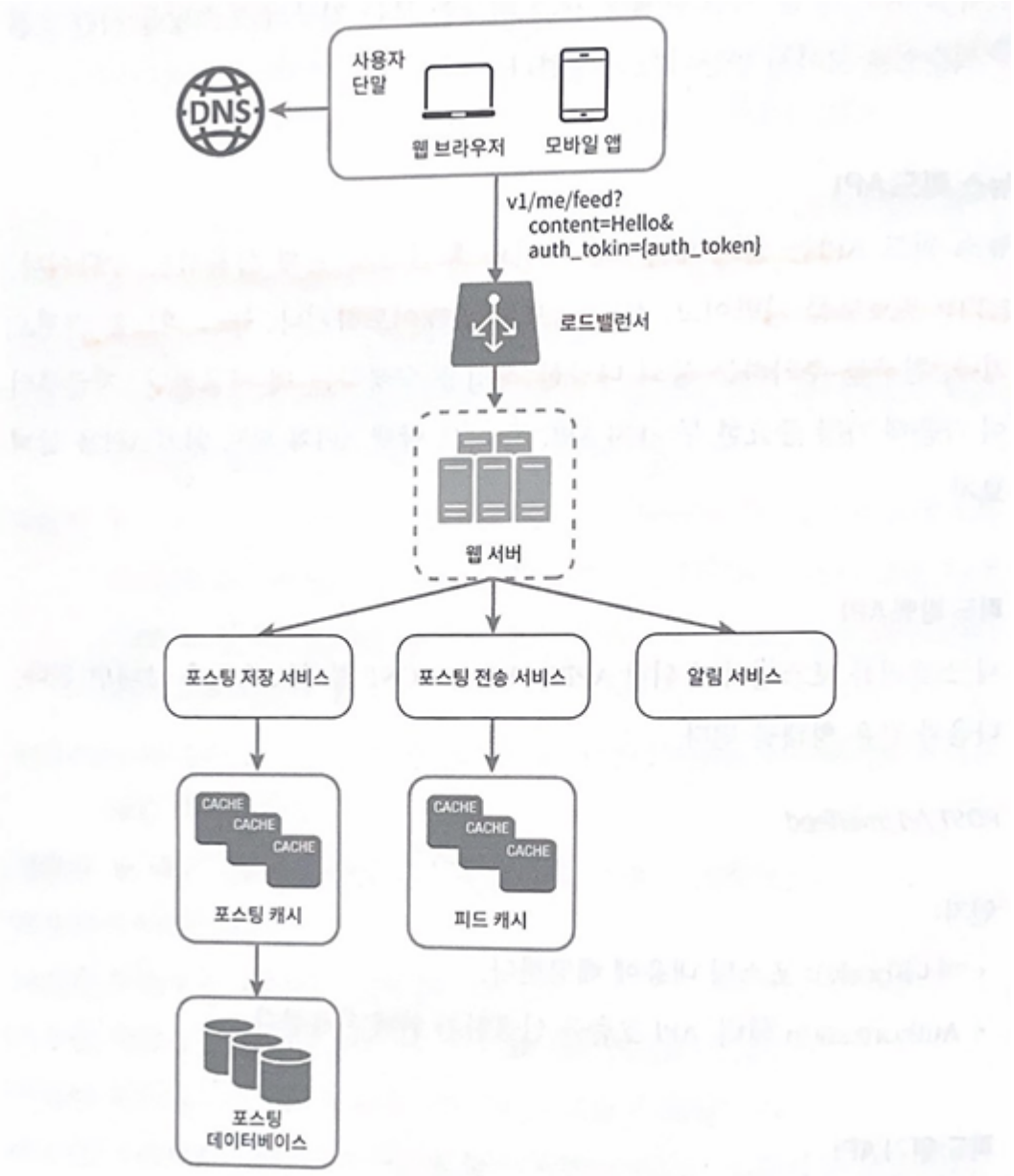

그림 11-2

- 사용자: 모바일 앱이나 브라우저에서 새 포스팅을 올리는 주체. POST /v1/me/feed API를 사용
- 로드밸런서: 트래픽을 웹 서버들로 분산
- 웹 서버: HTTP 요청을 내부 서비스로 중계하는 역할 담당
- 포스팅 저장 서비스: 새 포스팅을 데이터베이스와 캐시에 저장
- 포스팅 전송 서비스: 새 포스팅을 친구의 뉴스 피드에 푸시 → 뉴스 피드 데이터는 캐시에 보관, 빠르게 읽어갈 수 있도록 함

- 알림 서비스: 친구들에게 새 포스팅이 올라왔음을 알리거나, 푸시 알림을 보내는 역할을 담당

뉴스 피드 생성

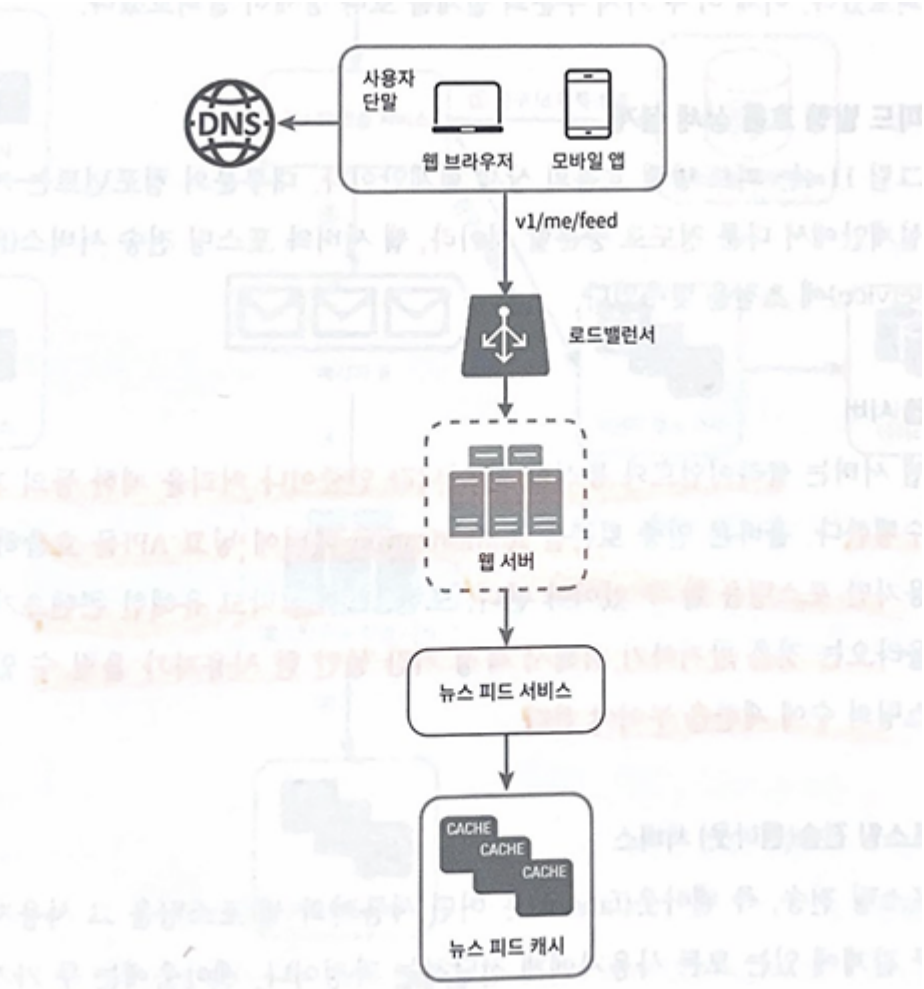

그림 11-3

- 사용자: 뉴스 피드를 읽는 주체. GET/v1/me/feed API를 이용
- 로드 밸런서: 트래픽을 웹 서버들로 분산
- 웹 서버: 트래픽을 뉴스 피드 서비스로 보냄
- 뉴스 피드 서비스: 캐시에서 뉴스 피드를 가져오는 서비스
- 뉴스 피드 캐시: 뉴스 피드를 렌더링할 때 필요한 피드 ID 보관

3단계 - 상세 설계

뉴스 피드 발행과 생성 설계 상세히 살펴보기

피드 발행 흐름 상세 설계

대부분의 컴포넌트는 개략적 설계안에서 다룬 정도로 충분하므로 웹 서버와 포스팅 전송 서비스에 초점을 맞춤

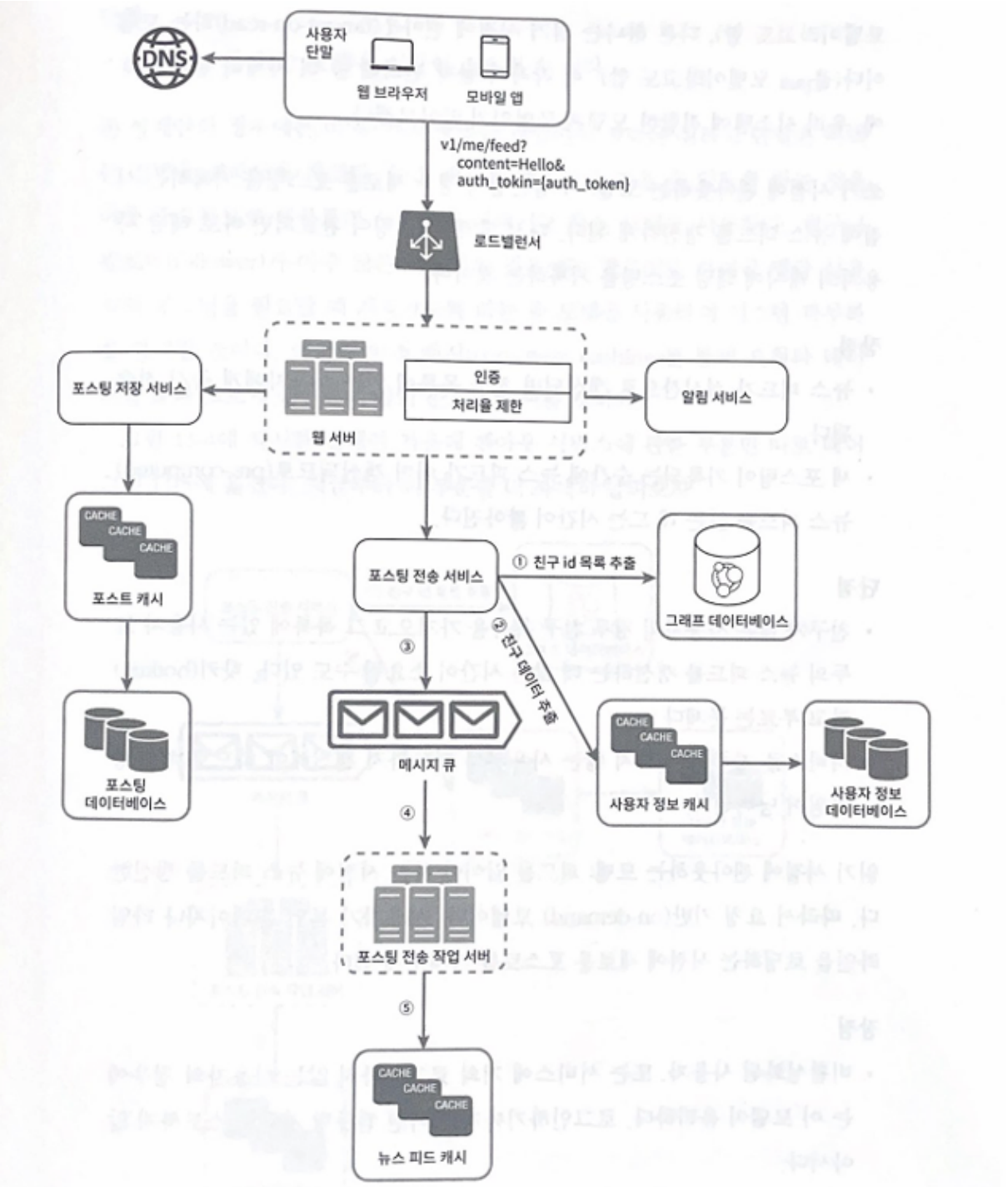

그림 11-4

웹 서버

- 클라이언트와 통신
- 인증, 처리율 제한

- 올바른 인증 토큰을 Authorization 헤더에 넣고 API를 호출하는 사용자만 포스팅할 수 있어야 함
- 스팸 막기 + 유해한 콘텐츠가 자주 올라오는 것 방지 위해 특정 기간 동안 한 사용자가 올릴 수 있는 포스팅의 수에 제한을 둠

포스팅 전송(팬아웃) 서비스

어떤 사용자의 새 포스팅을 그 사용자와 친구 관계에 있는 모든 사용자에게 전달하는 과정

포스팅 전송(팬 아웃) 모델

- 쓰기 시점에서 팬아웃
- 읽기 시점에서 팬아웃

1. 쓰기 시점에 팬아웃하는 모델: 새로운 포스팅을 기록하는 시점에 뉴스 피드를 갱신하게 됨

장점

- 뉴스 피드가 실시간으로 갱신 → 친구 목록에 있는 사용자에게 즉시 전송
- 새 포스팅이 기록되는 순간 뉴스 피드가 이미 갱신 → 뉴스 피드를 읽는 데 드는 시간이 짧아짐

단점

- 친구가 많은 사용자의 경우 친구 목록을 가져오고 목록에 있는 사용자 모두의 뉴스 피드 갱신에 많은 시간 소요 될 수 있음 → 핫키 문제
- 서비스를 자주 이용하지 않는 사용자의 피드백까지 갱신, 컴퓨팅 자원 낭비

2. 읽기 시점에 팬아웃하는 모델: 피드를 읽어야 하는 시점에 뉴스 피드를 갱신 → 요청기반 모델. 사용자가 본인 홈페이지나 타임 라인을 로딩하는 시점에 새로운 포스트를 가져옴

장점

- 비활성화 된 사용자, 또는 서비스에 거의 로그인하지 않는 사용자에 대해 컴퓨팅 자원 낭비 X

단점

- 뉴스 피드 읽을 때 많은 시간이 소요될 수 있음

- 데이터를 친구 각각에 푸시하는 작업이
  필요 없으므로 핫키 문제 발생 X

책의 설계안에서는 두 방법을 적절히 결합, 장점은 가져오고 단점은 버림

- 뉴스 피드를 빠르게 가져오는 것은 중요 → 대부분의 사용자에 대해서는 푸시(쓰는 시점에 팬아웃) 모델 사용
- 친구나 팔로워가 아주 많은 사용자에 대해서는 팔로워로 하여금 해당 사용자의 포스팅을 필요할 때 가져오도록 하는 풀 모델 사용 → 과부하 방지
- 안정 해시를 통해 요청과 데이터를 고르게 분산, 핫키 문제 줄여볼 것

팬아웃 서비스 동작 방식

1. 그래프 데이터베이스에서 친구 ID 목록을 가져옴
2. 사용자 정보 캐시에서 친구들의 정보를 가져온 후, 사용자 설정에 따라 친구 가운데 일부를 걸러냄
3. 친구 목록과 새 스토리의 포스팅 ID를 메시지 큐에 넣음
4. 팬아웃 작업 서버가 메시지 큐에서 데이터를 꺼내어 뉴스 피드 데이터를 뉴스 피드 캐시에 넣음
    - 뉴스 피드 캐시: <포스팅 ID, 사용자 ID>의 순서쌍을 보관하는 매핑 테이블
        - 메모리 요구량이 지나치게 늘어날 수 있으므로 ID만 보관
        - 메모리 크기를 적정 수준으로 유지하기 위해 캐시 크기에 제한을 두고, 해당 값은 조정이 가능하도록 함
        - 대부분의 사용자가 보려하는 것은 최신 스토리이므로 캐시 미스가 일어날 확률은 적음

피드 읽기 흐름 상세 설계

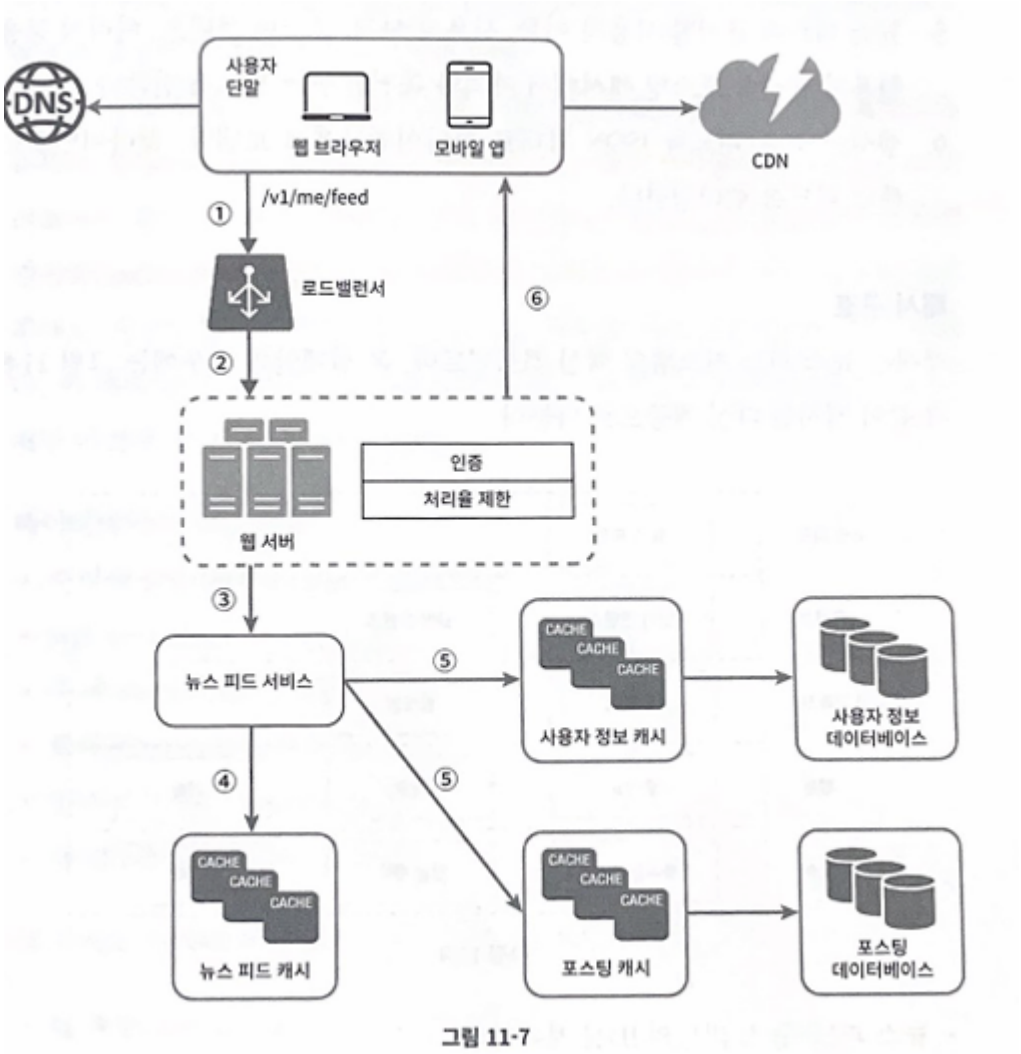

그림 11-7

- 이미지, 비디오 등의 미디어 컨텐츠는 CDN에 저장하여 빨리 읽어갈 수 있도록 함

클라이언트가 뉴스 피드를 읽어가는 과정

1. 사용자가 뉴스 피드를 읽으려는 요청을 보냄 → 요청은 /v1/me/feed로 전송
2. 로드밸런서가 요청을 웹 서버 가운데 하나로 보냄
3. 웹 서버는 피드를 가져오기 위해 뉴스 피드 서비스 호출
4. 뉴스 피드 서비스는 뉴스 피드 캐시에서 포스팅 ID 목록을 가져옴
5. 뉴스 피드에 표시할 사용자 이름, 사용자 사진, 포스팅 콘텐츠, 이미지 등을 사용자 캐시와 포스팅 캐시에서 가져와 완전한 뉴스 피드를 만듦

6. 생성된 뉴스 피드를 JSON 형태로 클라이언트에게 전송 → 클라이언트는 해당 피드를 렌더링

캐시 구조

캐시는 뉴스 피드 시스템의 핵심 컴포넌트

그림 11-8

- 뉴스 피드: 뉴스 피드의 ID를 보관
- 콘텐츠: 포스팅 데이터를 보관, 인기 콘텐츠는 따로 보관
- 소셜 그래프: 사용자 간 관계 정보를 보관
- 행동: 포스팅에 대한 사용자의 행위에 관한 정보를 보관

  ex) 포스팅에 대한 좋아요, 답글 등
- 횟수: 좋아요 횟수, 응답 수, 팔로워 수, 팔로잉 수 등의 정보 보관

4단계 - 마무리

회사마다 독특한 제약이나 요구조건을 가지므로 시스템 설계 시, 그런 점을 고려해야 함

설계를 진행하고 기술을 선택할 때는 배경에 어떤 타협적 결정들이 있었는지 잘 이해하고 설명할 수 있어야 함

! 더 논의해보면 좋을 내용 !

- 데이터베이스 규모 확장
  - 수직적 규모 확장 vs 수평적 규모 확장
  - SQL vs NoSQL
  - 주-부(master-slave) 다중화
  - 복제본에 대한 읽기 연산
  - 일관성 모델
  - 데이터베이스 샤딩
- 웹 계층을 무상태로 운영하기
- 가능한 한 많은 데이터를 캐시할 방법
- 여러 데이터 센터를 지원할 방법
- 메시지 큐를 사용하여 컴포넌트 사이의 결합도 낮추기
- 핵심 메트릭에 대한 모니터링

  ex) 트래픽이 몰리는 시간 대의 QPS, 사용자가 뉴스 피드를 새로 고침할 때의 지연 시간 등