GraphQL

🚀 GraphQL 완벽 가이드

📌 GraphQL이란?

GraphQL은 API를 위한 쿼리 언어이자 런타임으로, 클라이언트가 필요한 데이터만 정확히 요청할 수 있게 해주는 기술이다.

🆚 REST API vs GraphQL 비교

🎯 요청 방식 차이

구분	REST API	GraphQL
엔드포인트	여러 개 (리소스별)	하나의 엔드포인트
요청 구분	HTTP Method + URI	쿼리 내용으로 구분
HTTP Method	GET, POST, PUT, DELETE 등	주로 POST
API 버전 관리	복잡함	용이함

🔄 데이터 전송 문제 해결

❌ REST API의 문제점

1️⃣ Overfetching (데이터 과다 전송)

- RESTful API는 각 리소스의 **모든 데이터를 반환**한다
- 필요 이상의 데이터가 전송되어 **데이터 낭비와 오버헤드** 발생

2️⃣ Underfetching (데이터 부족 전송)

- 한 화면을 위해 **여러 차례 요청**을 보내야 하는 문제
- 하나의 요청으로 충분한 데이터를 얻을 수 없음

✅ GraphQL의 해결책

🎯 필요한 데이터만 요청

```
POST /graphql
```

```
query {
  books {
    title    # 책 제목만 요청
    author   # 책 저자만 요청
  }
}
```

결과: 쿼리에서 요청한 데이터만 응답에 포함되고, 그 외는 포함되지 않는다.

📡 GraphQL의 핵심 기능

🔍 쿼리 (Query)

클라이언트가 필요한 데이터만 선택해서 서버에 요청할 수 있다.

📝 뮤테이션 (Mutation)

데이터를 생성, 수정, 삭제하는 작업을 수행한다.

📲 구독 (Subscription)

실시간 데이터 업데이트를 위한 강력한 기능이다.

🌟 구독 활용 예시

```
subscription {
  reviewAdded(bookId: 1) {
    id
    content
    rating
    createdAt
  }
}
```

시나리오: 책 정보 화면에서 새 리뷰가 달릴 때 실시간으로 감지해서 표시

방식	REST API	GraphQL
실시간 업데이트	수시로 서버에 목록 요청 (Polling)	특정 리소스 업데이트 시 자동 알림
효율성	불필요한 요청 반복	변경 시에만 알림

⚠️ GraphQL의 단점

1️⃣ 캐싱의 어려움

- 요청이 **복잡한 GraphQL 메시지**로 구성됨
- 이를 기준으로 캐싱하기 어려움

2️⃣ 서버 부담 증가

- **복잡한 쿼리를 해석**해서 작업을 실행하기에 서버에 부담을 줄 수 있다
- N+1 쿼리 문제 등 성능 이슈 발생 가능

3️⃣ 학습 곡선

- REST API에 비해 **배우기 어려움**
- 새로운 개념과 문법을 익혀야 함

🎯 GraphQL이 적합한 환경

1️⃣ 방대하고 복잡한 데이터 모델을 가진 서비스

📊 성능 최적화

- **Overfetching, Underfetching**으로 발생하는 오버헤드가 큰 서비스
- GraphQL 사용 시 성능상 유리함

🏢 적용 사례

- 소셜 미디어 플랫폼
- 전자상거래 사이트
- 대규모 엔터프라이즈 애플리케이션

2️⃣ 클라이언트 중심의 데이터 요청

🎮 높은 제어권

- 클라이언트가 **데이터 요청에 많은 제어권**을 가져야 하는 경우
- 다양한 클라이언트(웹, 모바일, 데스크톱)가 서로 다른 데이터 요구사항을 가진 경우

3️⃣ 실시간 반응이 필요한 서비스

⚡ 실시간 업데이트

- **실시간 채팅** 애플리케이션
- **라이브 대시보드**
- **협업 도구** (실시간 문서 편집 등)

📊 선택 가이드

🚀 GraphQL을 선택해야 하는 경우

- ✅ 복잡한 데이터 관계가 많은 경우
- ✅ 다양한 클라이언트가 서로 다른 데이터를 요구하는 경우
- ✅ 실시간 기능이 중요한 경우
- ✅ 데이터 전송량 최적화가 중요한 경우
- ✅ API 버전 관리를 단순화하고 싶은 경우

🔄 REST API를 선택해야 하는 경우

- ✅ 간단한 CRUD 작업이 주된 경우
- ✅ 캐싱이 중요한 경우
- ✅ 빠른 개발과 배포가 필요한 경우
- ✅ 팀의 GraphQL 학습 비용을 피하고 싶은 경우
- ✅ 파일 업로드 등 단순한 작업이 많은 경우

🎭 실제 사용 사례

🌟 GraphQL 성공 사례

- **Facebook**: GraphQL 개발사, 모바일 앱 최적화
- **GitHub**: GitHub API v4에서 GraphQL 도입
- **Shopify**: 전자상거래 플랫폼에서 유연한 데이터 요청
- **Netflix**: 마이크로서비스 환경에서 데이터 통합

🔧 도입 시 고려사항

- 기존 REST API와의 **점진적 전환** 전략
- **캐싱 전략** 수립
- **쿼리 복잡도 제한** 정책
- **보안 및 인증** 체계 구축

SOAP