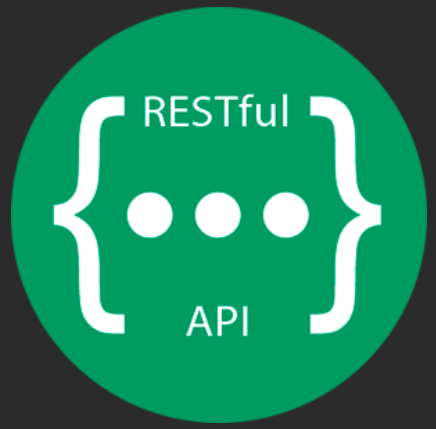
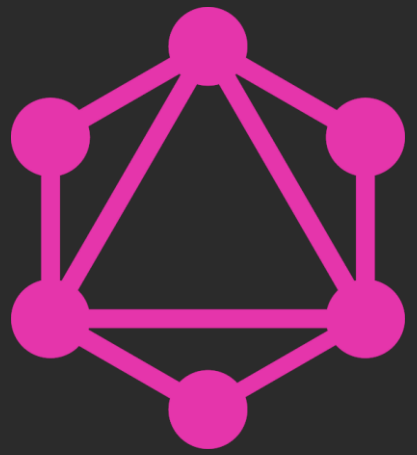


RESTful

vs

GraphQL



2020.08.15
권수연, 김연정

REST란?

REpresentational State Transfer

자원(recource)의 표현(repersentation)에 의한 상태전달

REST란?

REpresentational State Transfer

자원(recource)의 표현(repersentation)에 의한 상태전달

자원을 이름으로 구분하여 (자원의 표현)
해당 자원의 상태(정보)를 주고 받는 모든 것

REST란?

REpresentational State Transfer

자원(recource)의 표현(repersentation)에 의한 상태전달

해당 소프트웨어가 관리하는 모든 것

자원을 이름으로 구분하여 (자원의 표현)

해당 자원의 상태(정보)를 주고 받는 모든 것

REST란?

REpresentational State Transfer

자원(recource)의 표현(repersentation)에 의한 상태전달

자원을 표현하기 위한 이름

자원을 이름으로 구분하여 (자원의 표현)

해당 자원의 상태(정보)를 주고 받는 모든 것

REST란?

REpresentational State Transfer

자원(recource)의 표현(repersentation)에 의한 상태전달

자원을 이름으로 구분하여 (자원의 표현)

해당 자원의 상태(정보)를 주고 받는 모든 것

데이터가 요청되는 시점에서 자원의 상태를 전달

그래서 REST란?

www과 같은 분산 하이퍼미디어 시스템을 위한
소프트웨어 개발 아키텍처의 한 형식

웹의 기존 기술과 HTTP 프로토콜을 그대로 활용

그래서 REST란?

www과 같은 분산 하이퍼미디어 시스템을 위한
소프트웨어 개발 아키텍처의 한 형식

웹의 기존 기술과 HTTP 프로토콜을 그대로 활용

➡ 웹의 장점을 최대한 활용할 수 있는 아키텍처 스타일

그래서 REST란?

www과 같은 분산 하이퍼미디어 시스템을 위한
소프트웨어 개발 아키텍처의 한 형식

웹의 기존 기술과 HTTP 프로토콜을 그대로 활용

네트워크 상에서 Client와 Server 사이의 통신 방식 중 하나

그래서 REST가 뭔데?

HTTP URI를 통해 자원을 명시

HTTP Method(POST, GET, PUT, DELETE)를 통해
자원에 대한 CRUD Operation을 적용하는 아키텍처 스타일

그래서 REST가 뭔데?

HTTP URI를 통해 자원을 명시

HTTP Method(POST, GET, PUT, DELETE)를 통해
자원에 대한 CRUD Operation을 적용하는 아키텍처 스타일

REST 구성 요소

- 자원(Resource) : URI
- 행위(Verb) : HTTP Method
- 표현(Representation of Resource)

REST 구성 요소

- 자원(Resource) : URI

  모든 자원에 고유한 ID가 존재하고, 이 자원은 Server에 존재
  예) /posts/:post_id

- 행위(Verb) : HTTP Method

- 표현(Representation of Resource)

REST 구성 요소

- 자원(Resource) : URI

- 행위(Verb) : HTTP Method

  POST, GET, PUT, DELETE, HEAD, PATCH,
  CONNECT, TRACE, OPTIONS Method 제공

- 표현(Representation of Resource)

REST 구성 요소

- 자원(Resource) : URI

- 행위(Verb) : HTTP Method

- 표현(Representation of Resource)

  REST에서 하나의 자원은 JSON, XML, TEXT, RSS 등 여러 형태의 Representation으로 나타내어 질 수 있음

REST 특징

- Server-Client(서버-클라이언트 구조)
- Stateless(무상태)
- Cacheable(캐시 처리 가능)
- Layered System(계층화)
- Code-On-Demand(optional)
- Uniform Interface(인터페이스 일관성)

REST API란?

RE**S**T

RE**P**resentational **S**tate **T**ransfer

Application **P**rogramming **I**nterface

REST 기반으로 서비스 API를 구현한 것

REST API 설계 예시

GET	/api/posts	모든 게시물 조회
GET	/api/posts/1	1번 게시물 조회
POST	/api/posts	게시물 저장
PUT	/api/posts/1	1번 게시물 전제 수정
PATCH	/api/posts/1	1번 게시물 부분 수정
DELETE	/api/posts/1	1번 게시물 삭제

RESTful이란?

REST 원리를 따르는 시스템을 지칭

REST API 를 제공하는 웹 서비스를 RESTful 하다고 할 수 있음

REST를 REST답게 쓰기 위한 방법,
누군가가 공식적으로 발표한 것이 아님

RESTful의 목적

REST 원리를 따르는 시스템을 지칭

이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것

RESTful한 API를 구현하는 근본적인 목적이 성능향상에 있다??

RESTful의 목적

REST 원리를 따르는 시스템을 지칭

이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것

~~RESTful한 API를 구현하는 근본적인 목적이 성능향상에 있다??~~
일관적인 컨벤션을 통한 API의 이해도 및 호환성을 높이는 것이 주 동기!

성능이 중요한 상황에서는 굳이 RESTful한 API를 구현할 필요는 없음

“RESTful하지 못하다”

CRUD 기능을 모두 POST로만 처리하는 API

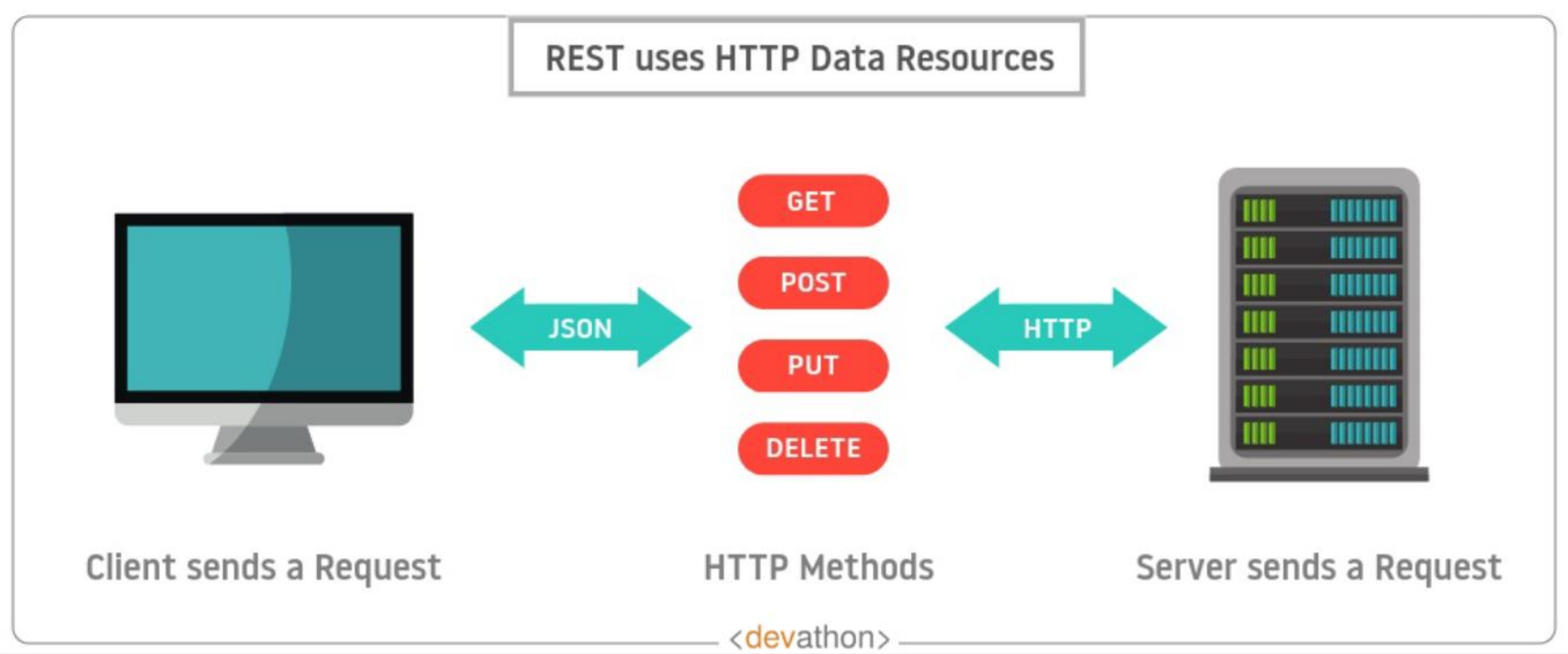
Route URI에 Resource, ID 외의 정보가 들어가는 경우

/posts/updateContents
 -> URI에 행위에 대한 동사 표현이 들어가면 안된다. (CRUD 기능을 나타내는 것은 사용하지 않는다)

/posts/delete/1 -> URI에 HTTP Method가 들어가면 안된다

REST의 장점

HTTP 프로토콜을 그대로 활용



REST의 장점

HTTP 프로토콜을 그대로 활용

👍 REST API 사용을 위한 별도의 인프라를 구축할 필요가 없다

👍 HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용 가능하다

REST의 장점

👍 Hypermedia API의 기본을 충실히 지키면서 범용성을 보장

👍 REST API 메시지가 의도하는 바를 쉽게 파악

👍 서버와 클라이언트의 역할을 명확하게 분리

REST의 단점

👎 표준이 존재하지 않는다

👎 URI에 Endpoint가 복잡한 경우가 많다

👎 사용할 수 있는 HTTP Method 형태가 제한적

👎 구형 브라우저가 지원해주지 못하는 부족함

PUT, DELETE를 사용하지 못하는 점
pushState를 지원하지 않는 점

GraphQL? 왜 등장했나?

페이스북 曰

“ REST API 로는 다양한 기종에서 필요한 정보들을 일일이 구현하는 것이 힘들다. ”

GraphQL? 왜 등장했나?

페이스북 曰

“ REST API 로는 다양한 기종에서 필요한 정보들을 일일이 구현하는 것이 힘들다. ”

== RESTful API는 유연성이 떨어진다.

GraphQL? 왜 등장했나?

페이스북 日

“ REST API 로는 다양한 기종에서 필요한 정보들을 일일이 구현하는 것이 힘들다. ”

== RESTful API는 유연성이 떨어진다.

ex) iOS 와 Android 에서 필요한 정보들이 조금씩 다르고,
그 다른 부분마다 API 를 구현하는 것이 힘들다.

GraphQL? 왜 등장했나?

클라이언트 측에서 **원하는 대로** 정보를 가져올 수 있고,

보다 편하게 정보를 수정할 수 있도록 하는

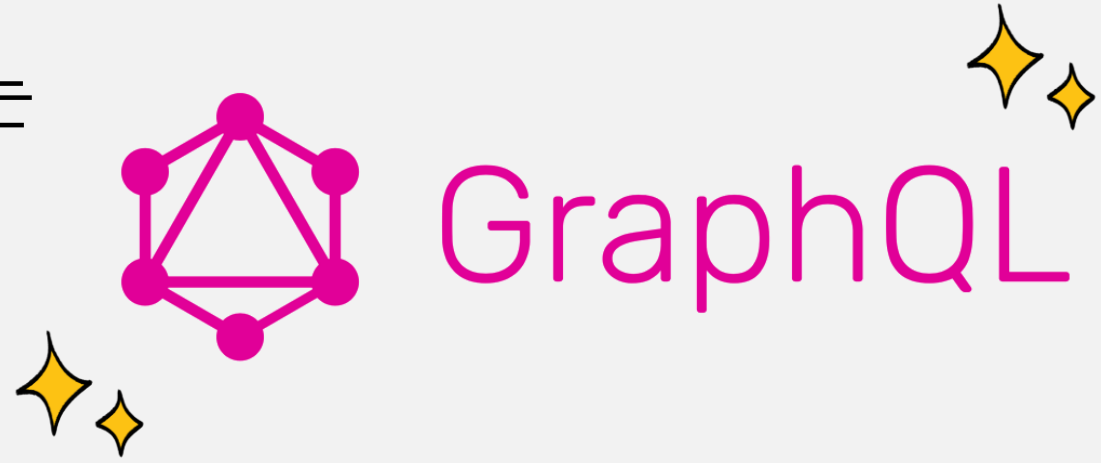
표준화된 **Query language**를 만들자!

GraphQL? 왜 등장했나?

클라이언트 측에서 **원하는 대로** 정보를 가져올 수 있고,

보다 편하게 정보를 수정할 수 있도록 하는

표준화된 **Query language**를 만들자!



GraphQL이란?

Graph Query Language by facebook

GraphQL이란?

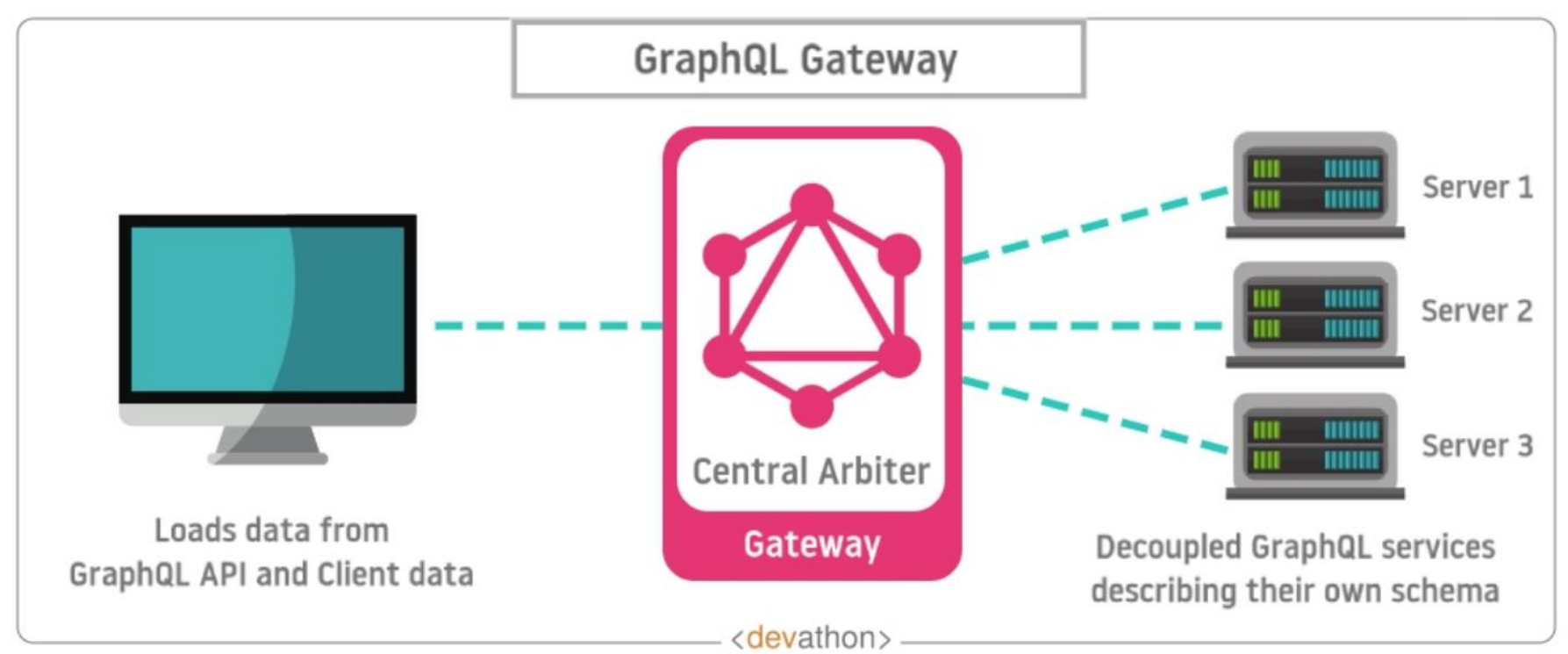
Graph Query Language by facebook

잠깐! Query Language 란 무엇인가요?

정보를 얻기 위해 보내는 질의문(Query)을 만들기 위해 사용되는 언어

GraphQL이란?

단 하나의 Endpoint가 존재!



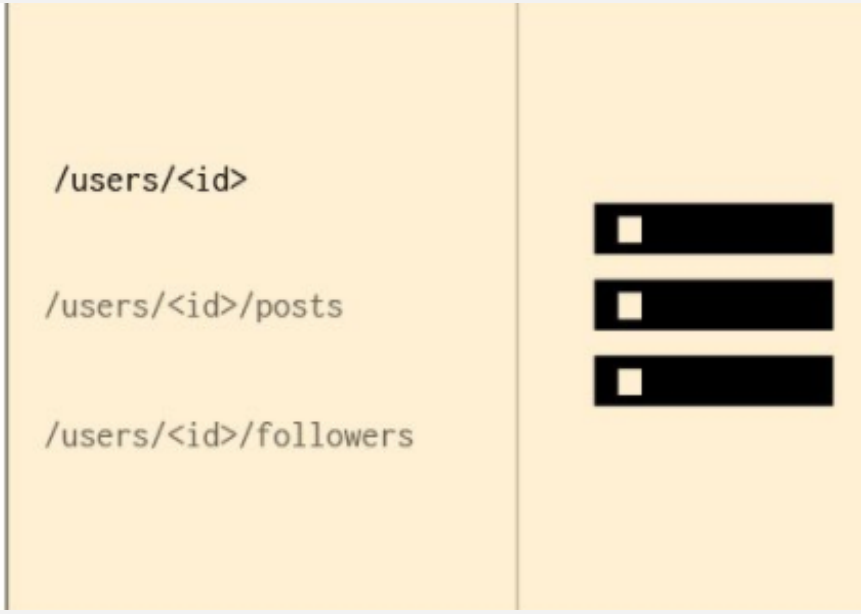
GraphQL 예시
feat. REST와의 차이점

{사용자의 이름, 사용자의 게시물, 사용자의 팔로워를 알고 싶다}

GraphQL 예시
feat. REST와의 차이점

RESTful API 의 경우

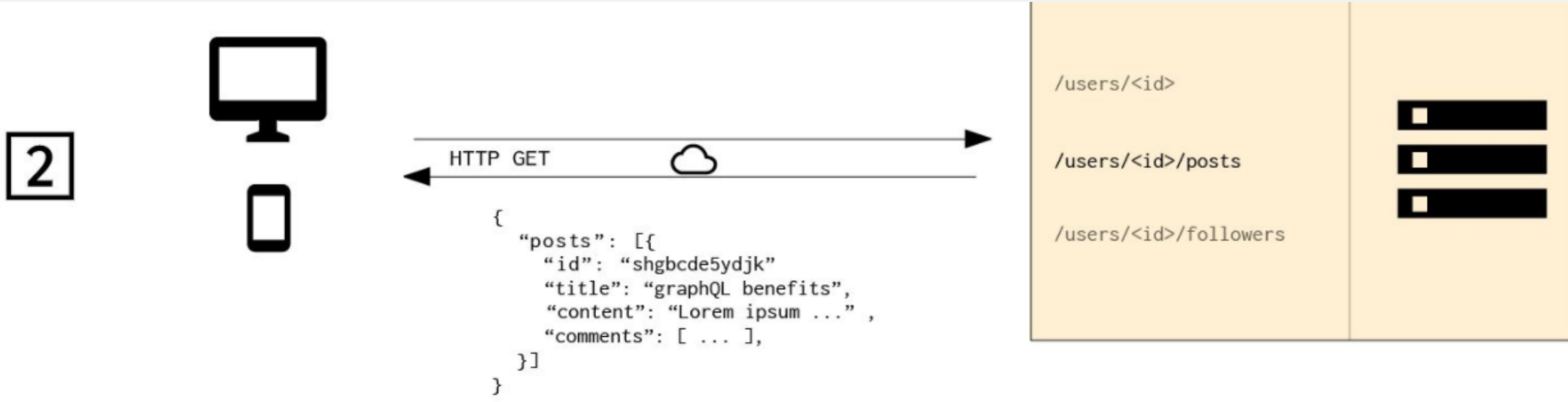
"사용자의 이름, 사용자의 게시물, 사용자의 팔로워를 알고 싶다"



GraphQL 예시
feat. REST와의 차이점

RESTful API 의 경우

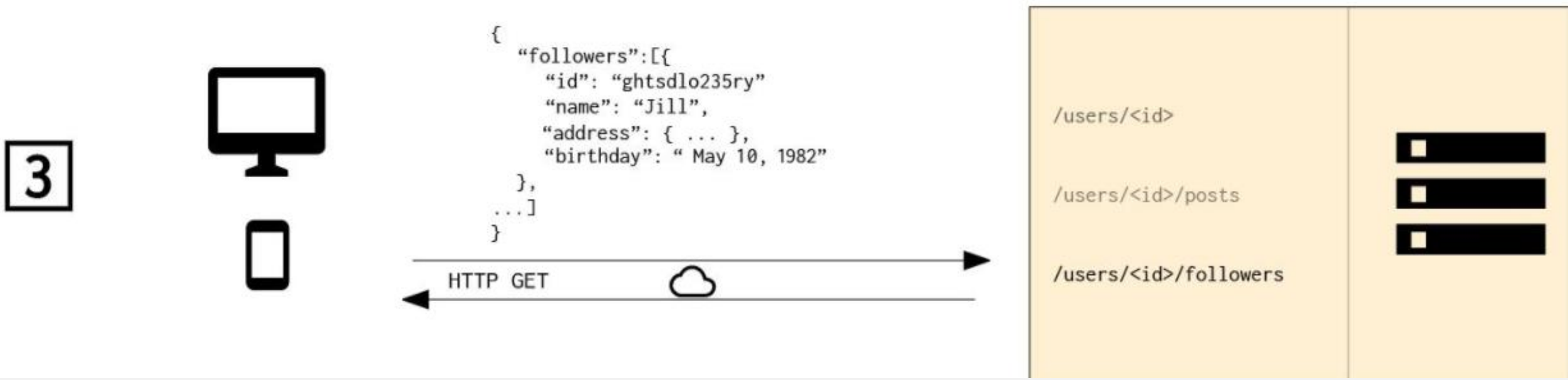
"사용자의 이름, **사용자의 게시물**, 사용자의 팔로워를 알고 싶다"



GraphQL 예시
feat. REST와의 차이점

RESTful API 의 경우

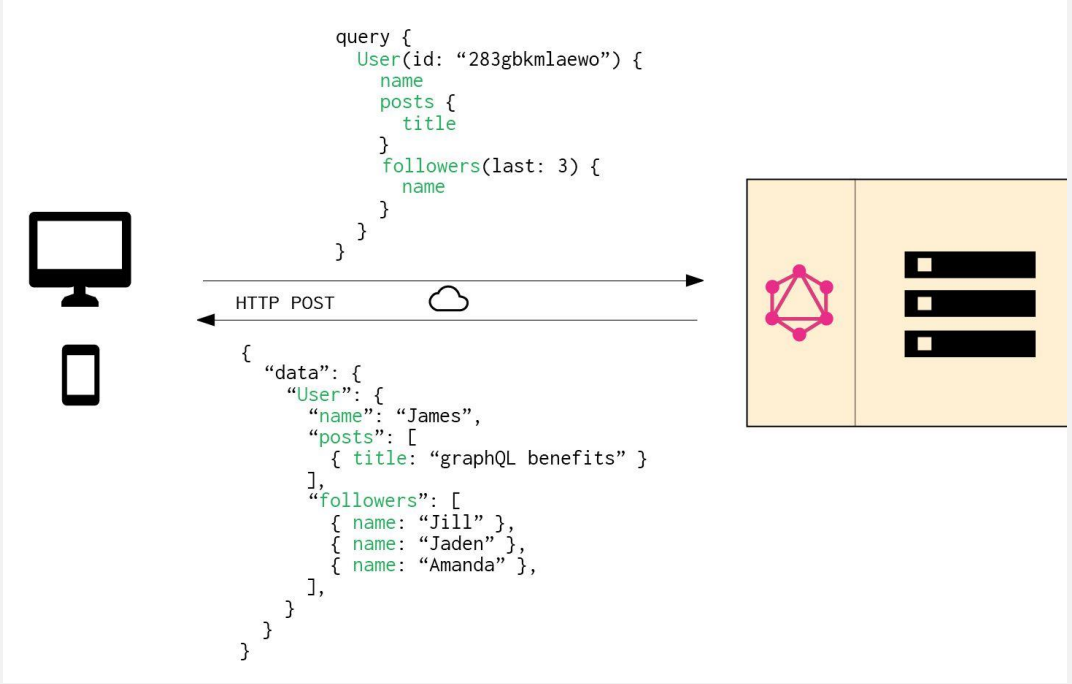
"사용자의 이름, 사용자의 게시물, **사용자의 팔로워**를 알고 싶다"



GraphQL 예시
feat. REST와의 차이점

GraphQL 의 경우

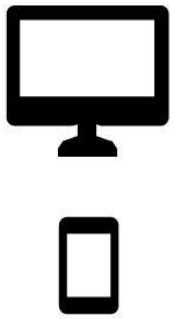
"사용자의 이름, 사용자의 게시물, 사용자의 팔로워를 알고 싶다"



GraphQL 예시

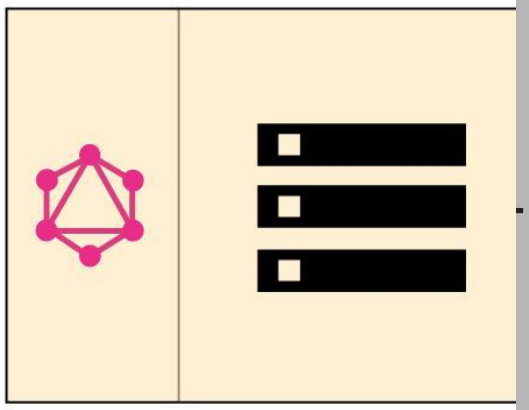
GraphQ

“사용자의 . 싶다”



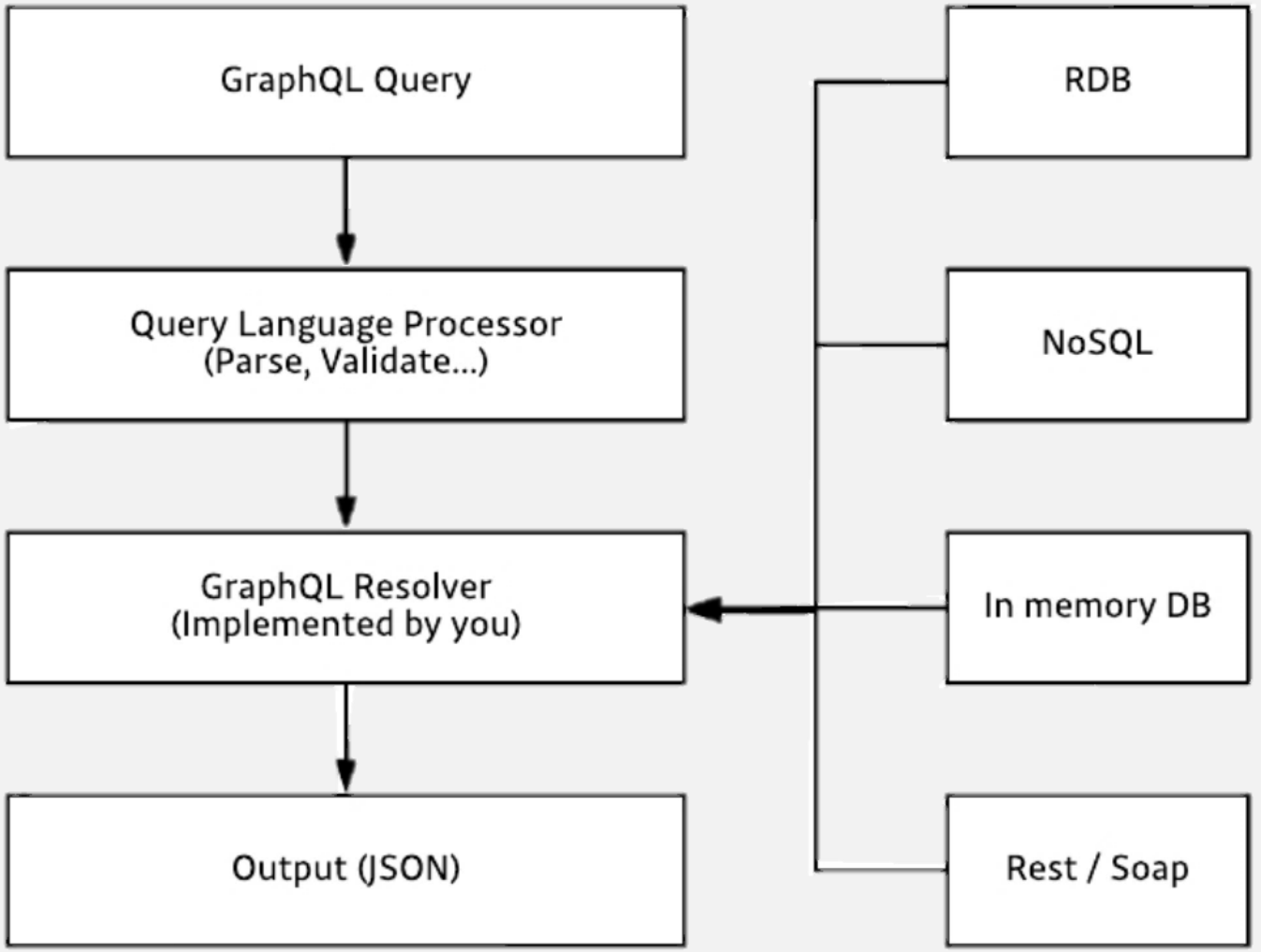
```
query {
  User(id: “283gbkmlaewo”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

HTTP POST



```
{
  “data”: {
    “User”: {
      “name”: “James”,
      “posts”: [
        { title: “graphQL benefits” }
      ],
      “followers”: [
        { name: “Jill” },
        { name: “Jaden” },
        { name: “Amanda” },
      ],
    }
  }
}
```

GraphQL의 Request 파이프라인



GraphQL의 Request

[Query와 Mutation]

Query

쿼리는 데이터를 읽는데(R) 사용,
CRUD 에서 GET 에 해당

Mutation

데이터를 변조(CUD)하는데 사용,
CRUD에서 POST, PUT, DELETE에 해당

GraphQL의 Request

Query : 데이터 가져오기 (fetch)

```
query {
    user(id : "1"){
        id
        username
        age
    }
}
```

```
Response JSON
{
    data: {
        user:{
            id: "1",
            username : "Myname",
            age : 20
        }
    }
}
```

GraphQL의 Request

Mutation : 서버 측 데이터 수정하기

```
mutation {
  createUser(
    username: "sooyeon",
    age: 24) {
      id
      username
      age
  }
}
```

```
Response JSON
{
    data: {
        createUser:{
            id: "1",
            username : "sooyeon",
            age : 24
        }
    }
}
```

GraphQL의 Request

Mutation : 서버 측 데이터 수정하기

```
mutation {
  createUser(
    username: "sooyeon",
    age: 24) {
      id
      username
      age
  }
}
```

```
Response JSON
{
    data: {
        createUser:{
            id: "1",
            username : "sooyeon",
            age : 24
        }
    }
}
```

GraphQL의 장점

👍 쿼리를 통하여 딱 필요한 데이터만 fetching

→ No overfetch & No underfetch

👍 HTTP 요청의 횟수를 줄일 수 있다

👍 HTTP 응답의 사이즈를 줄일 수 있다

GraphQL의 단점

👎 직접 구현해야 하는 파일 업로드

👎 고정된 요청과 응답만 필요할 경우에는 Query 로 인해 요청의 크기가 RESTful API 의 경우보다 더 커짐

👎 HTTP 캐싱 사용 불가능

👎 요청 필터링의 어려움

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

기존 REST API는

GET	/api/posts	모든 게시물 조회
GET	/api/posts/1	1번 게시물 조회
POST	/api/posts	게시물 저장
DELETE	/api/posts/1	1번 게시물 삭제

등등...

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

GraphQL 에서는?

/graphql 이면 끝!!

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

build.gradle 설정

```
dependencies{

    ...

    implementation group: 'com.graphql-java', name: 'graphql-spring-boot-starter'
    implementation group: 'io.leangen.graphql', name: 'graphql-spqr-spring-boot-starter'

    ...
}
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

domain/Post.java

```java
@Data
@NoArgsConstructor
@Entity
@ToString
public class Post {

        @Id
        @GeneratedValue
        @GraphQLQuery(name = "id")
        private Long id;

        @GraphQLQuery(name = "title")
        private String title;

}
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

repository/PostRepository.java

```
@Repository
public interface PostRepository extends JpaRepository<Post, Long>{
}
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

service/PostService.java

```java
@Service
@GraphQLApi //추가
public class PostService {

    @Autowired
    private PostRepository postRepository;

    /*
    query {
        posts{
            id
        title
        }
    }
    */
    @GraphQLQuery(name = "posts")
    public List<Post> getPosts(){
        return postRepository.findAll();
    }
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

service/PostService.java

```java
@Service
@GraphQLApi //추가
public class PostService {
    ...

    /*
    query{
        post(id:1){
            id
            title
        }
    }
    */
    @GraphQLQuery(name = "post")
    public Optional getPostById(Long id){
        return postRepository.findById(id);
    }

    ...
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

service/PostService.java

```java
@Service
@GraphQLApi //추가
public class PostService {

    ...

    /*
    mutation{
        savePost(post:{title:"title"}){
            id
            title
        }
    }
    */
    @GraphQLMutation(name = "savePost")
    public Post savePost(Post post) {
        return postRepository.save(post);
    }

    ...
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

service/PostService.java

```java
@Service
@GraphQLApi //추가
public class PostService {

    ...
    /*
    mutation{
        deletePost(id:1)
    }
    */
    @GraphQLMutation(name = "deletePost")
    public void deletePost(Long id) {
        postRepository.deleteById(id);
    }
}
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

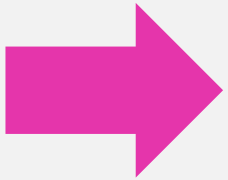
resources/post.graphqls

```
type Post {
    id : Long!
    title : String!
}

type Query {
    post(id : Long) : Post
    posts : [Post]
}

type Mutation {
    savePost(post : Post) : Post
    deletePost(id : Long) : Boolean!
}
```

그럼, Spring Boot에서 GraphQL을 어떻게 적용할까?

/graphql

```
mutation{
    savePost(post:{title:"title"}){
        id
        title
    }
}
```



```
Response JSON
{
    data: {
        savePost:{
            id: "1",
            title : "title"
        }
    }
}
```

RESTful or GraphQL ??

RESTful

- HTTP 와 HTTPs 에 의한 Caching 을 잘 사용하고 싶을 때
- File 전송 등 단순한 Text 로 처리되지 않는 요청들이 있을 때
- 요청의 구조가 정해져 있을 때

GraphQL

- 서로 다른 모양의 다양한 요청들에 대해 응답할 수 있어야 할 때
- 대부분의 요청이 CRUD(Create-Read-Update-Delete) 에 해당할 때

RESTful or GraphQL ??

그러나 더 중요한 것은…

둘 중 하나를 선택할 필요는 없다!!

RESTful and GraphQL !!

둘 중 하나를 선택할 필요는 없다!!

하나의 Endpoint 를 GraphQL 용으로 만들고,

다른 RESTful endpoint 들을 만들어 놓는 것은

API 개발자의 자유다!!

RESTful and GraphQL !!

둘 중 하나를 선택할 필요는 없다!!

주의해야할 것은…

하나의 목표를 위해
두 API structure 를 섞어놓는 것은
API 의 품질을 떨어뜨릴 수 있다!

예: 사용자 정보를 등록하는 것은 RESTful API 로, 사용자 정보를 수정하는 것은 GraphQL API 로 한다면 끔찍할 것이다

THANK YOU