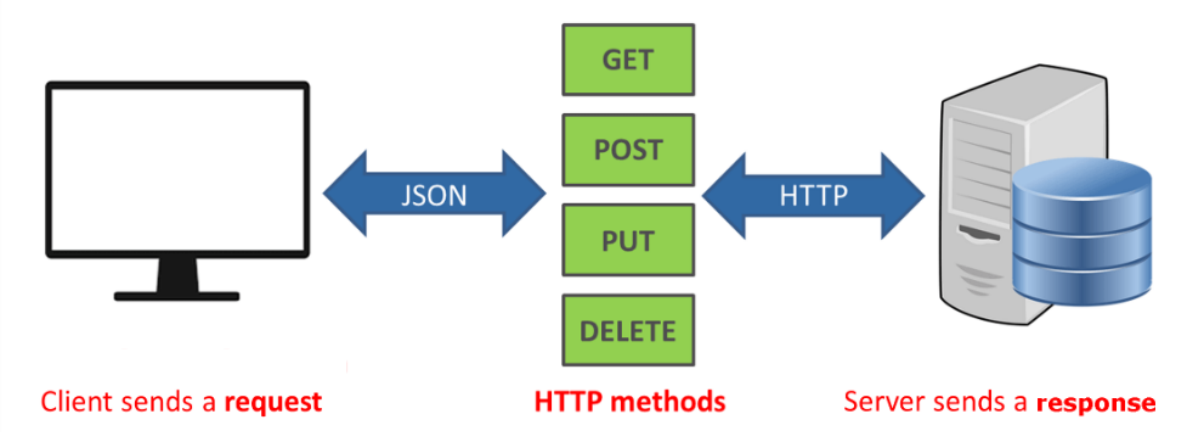
REST API

정보들이 주고 받아지는데 있어서 개발자들 사이에서 널리 지켜지는 일종의 형식, 약속

각 요청이 어떤 동작이나 정보를 위한 것인지를 그 요청의 자체 모습 그대로 추론 가능함



*** 인터페이스란 ?** 기계와 인간의 소통 창구 ex) 컴퓨터 자판, 모니터 전원, 사용자가 명령을 넣는 것 뿐만 아니라 그 결과를 받아오는 것

*** UI (User Interface)**? 소프트웨어와 인간의 소통창구 ex)버튼 스크롤바 슬라이더 브라우저 창

*** API 란?** 기계와 기계 소프트웨어와 소프트웨어 사이에서도 소통 창구가 필요함

Ex) 특정 서버에게 정보를 요청하고 명령을 받을 수 있는 수단

*** REST(Representational State Transtfer)란?** 웹에서 존재하는 모든 자원(이미지, 동영상, DB자원)에 고유한 URI를 부여해 활용하는 것 = 자원을 정의하고 자원에 대한 주소를 지정하는 방법론을 의미

*** URI 란 ?** 자원을 구조와 함께 나타내는 구분자, 동사가 아닌 명사들로 이루어 져야함.

Axios, ajax

서버를 rest api 로 요청을 전송할 때 HTTP(Hyper Text Transfer Protocol) 규약으로 전송

* **CRUD 란 ?** Create 생성 Read 조회 Update 수정 Delete 삭제, -> REST API 메소드

1. Get vs Post

1) Get : read 데이터를 조회하는데 사용

- **GET 요청을 할 때는 Body 부분은 비어있고 헤더에 Body의 콘텐츠 타입을 명시**브라우저 기록에 남고, 북마크에 추가할 수 있으며, 데이터 길이에 제한이 있다.
- Uri에 파라미터로 넘겨서 호출 ex) 게시판
- 장점: 속도가 빠름
- 단점: 데이터 노출됨

2) Post : create 새로운 정보를 추가하는 데 사용

- **POST 요청 시 HTTP 패킷의 헤더에 Body의 콘텐츠 타입을 명시하는 Content-Type 헤더 필드를 포함하고 HTTP 패킷의 Body에는 데이터를 담아서 서버로 전송합니다.**

데이터(Body) 길이에 제한이 없다. 대용량 데이터를 전송 가능하다.

캐시 되지 않고 브라우저 기록에 남지 않고, 북마크에 추가할 수 없다.

- 네트워크 패킷 안에 암호화해서 넘기는 것 ex) 로그인

	GET	POST
캐시	⭕	❌
브라우저 기록	⭕	❌
북마크 추가	⭕	❌
데이터 길이 제한	⭕	❌
HTTP 응답 코드	200(Ok)	201(Created)
언제 주로 사용하는가?	리소스 요청	리소스 생성
리소스 전달 방식	쿼리스트링	HTTP Body
idempotent	⭕	❌

*** idempotent 란?** 동일한 연산을 여러 번 적용하더라도 결과가 달라지지 않는 성질

get으로 서버에게 동일한 요청을 여러 번 전송하더라도 동일한 응답이 돌아와야함.

Post는 non-idempotent 하기 때문에 서버에 동일한 요청을 여러 번 전송해도 응답이 항상 다를 수 있다. 서버의 상태나 데이터를 변경할 때 사용한다.

2. Put vs Fetch

1) Put : update 정보를 수정하는데 사용, 정보를 통째로 갈아 끼울 때 사용, 리소스의 모든 것을 업데이트 한다.

2) Fetch : update 정보를 수정하는데 사용, 정보 중 일부를 특정 방식으로 변경할 때 사용

리소스의 일부를 업데이트 한다.

가령 한 사용자에 대해 여러 정보를 객체로 수집하여 서버로 보내지는 경우, PUT은 보내

지지 않은 정보에 대해서는 null 갑으로 업데이트하지만, patch는 기존 데이터를 유지하는 방식으로 대응한다.

Reference

https://github.com/gyoogle/tech-interview-for-developer/blob/master/Web/%5BWeb%5D%20REST%20API.md