[Network] 쿠키, 세션, 토큰

인증, 인가

서버가 클라이언트 인증을 확인하는 방식은 대표적으로 3가지가 있다. (쿠키, 세션, 토큰)

그 전에, 인증과 인가가 무엇인지 부터 알아보자.

- 인증
    - 사용자가 누구인지 확인
    - ex) 회원가입, 로그인
- 인가
    - 사용자에 대한 권한을 허락하는 것
    - 유저가 요청하는 request를 실행할 수 있는 권한이 있는 유저인가를 확인하는 절차.
    - 'access token'를 쓰면 인가가 수월함. access token을 통해서 해당 유저 정보를 얻을 수 있음으로 해당 유저가 가지고 있는 권한(permission)도 확인 가능.

쿠키와 세션을 사용하는 이유

▼ HTTP 특징

- Connectionless (비연결성)
  - 클라이언트가 서버로 요청을 보낸 후에 서버로부터 응답을 받으면 연결을 끊어버린다.
  - 이로 인해 많은 사람이 웹을 이용하더라도 실제 동시 접속을 최소화하여 더 많은 유저의 요청을 처리할 수 있습니다.
- Stateless (무상태성)
  - Connectionless한 속성을 가지지만, 연결을 끊었기 때문에 클라이언트의 이전 상태(로그인 유무 등)를 알 수가 없다
  - 무상태성으로 인한 문제는 쿠키와 세션을 사용해서 상태 정보를 유지하도록 해결할 수 있다.

지난 시간, HTTP의 특징으로 Connectionless, Stateless 에 대해서 알아보았다.

서버와 클라이언트가 통신을 할 때, 통신이 연속적으로 이어지지 않고 한 번 통신이 되면 끊어지므로 **상태가 유지되지 않는다.**

하지만, 실제로는 상태가 유지되어야 하는 경우가 생긴다!

요청이 끝나면 서버는 우리가 누군지 잊어버리기 때문에, 서버에 데이터를 요청할 때마다 우리가 누군지 알려주어야 한다. 이 때 필요한 것이 바로 쿠키, 세션, 토큰이다.

> 💡 **Stateless**라는 HTTP 프로토콜의 특성이자 약점을 보완하기 위해 필요한 것이 쿠키, 세션, 토큰!

쿠키

쿠키란?

- 클라이언트(로컬) 브라우저에 저장되는, key-value 형태의 작은 데이터 파일
- 클라이언트에 저장되어 필요시 정보를 참조하거나 재사용할 수 있다.

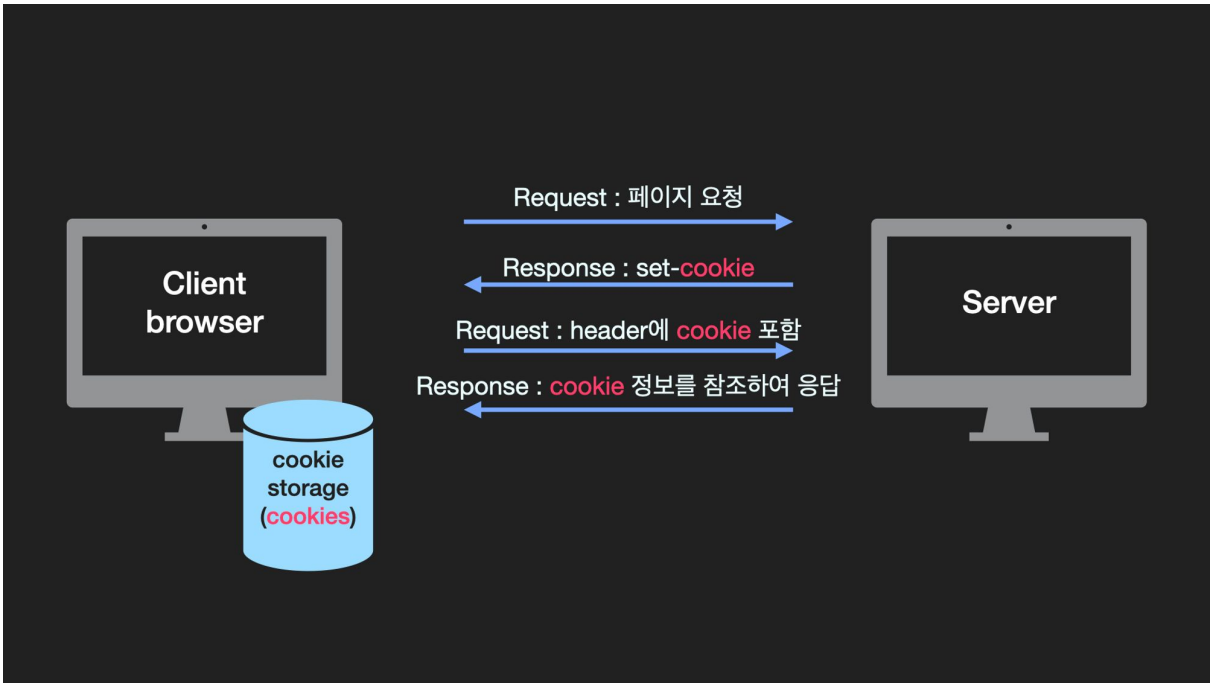
쿠키의 특징

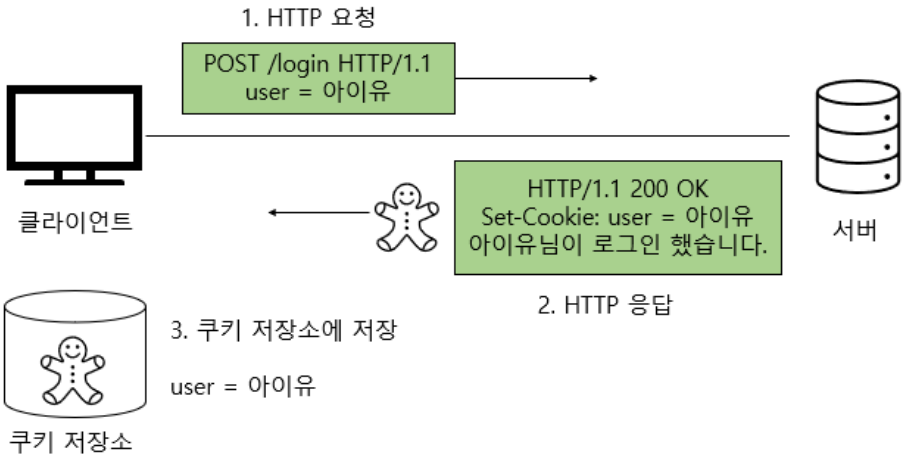
- ‘유효 시간’을 설정할 수 있다.
  - 유효시간: 사용자 인증이 유효한 시간
  - 유효 시간이 정해지면 브라우저가 종료되어도 인증이 유지된다
- Response Header의 Set-Cookie 속성을 사용하여 클라이언트에 쿠키를 만들 수 있다
- 쿠키는 사용자가 따로 요청하지 않아도 브라우저가 Request시에 Request Header를 넣어서 자동으로 서버에 전송한다
- 클라이언트에 300개까지의 쿠키를 저장할 수 있다
- 하나의 도메인당 20개의 쿠키를 가질 수 있다
- 하나의 쿠키는 4KB(=4096바이트)까지 저장할 수 있다

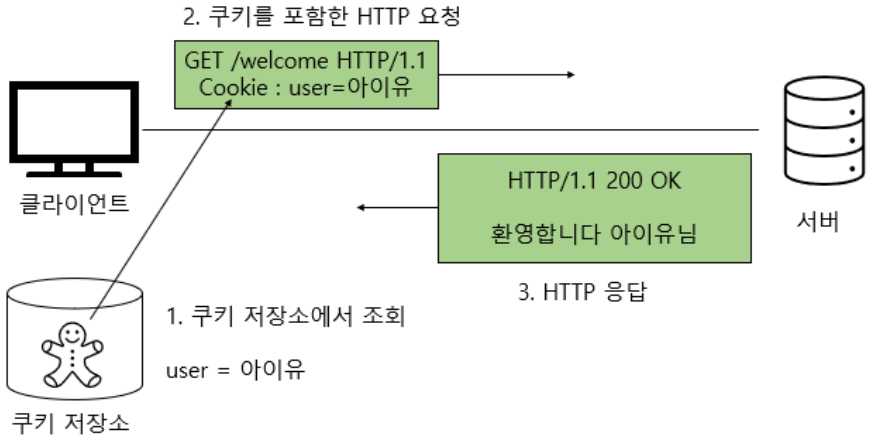
쿠키의 구성 요소

- 이름 : 각각의 쿠키를 구별하는 데 사용되는 이름
- 값 : 쿠키의 이름과 관련된 값
- 유효시간 : 쿠키의 유지시간
- 도메인 : 쿠키를 전송할 도메인
- 경로 : 쿠키를 전송할 요청 경로

쿠키의 동작 방식







1. 클라이언트가 페이지를 요청 (사용자가 웹사이트에 접근)

2. 서버에서 쿠키를 생성하고, HTTP 헤더에 쿠키를 포함 시켜 응답
    - 응답헤더에 `set-cookie:` 를 통해 쿠키를 추가하여 응답
3. 넘겨받은 쿠키는 클라이언트가 가지고 있다가(로컬 PC에 저장) 다시 서버에 요청할 때 요청과 함께 쿠키를 전송한다.
    - 브라우저가 종료되어도 쿠키 유효시간까지 클라이언트에서 보관하고 있다

- 이후에 같은 페이지를 요청을 할 경우(같은 사이트 재방문할 경우), 요청 페이지와 함께 HTTP 헤더에 쿠키를 함께 보낸다.
    - 사용자가 따로 작업을 하지 않아도 자동으로 브라우저가 쿠키를 request header에 담아서 서버에 전송한다.
- 서버에서 쿠키를 읽어 이전 상태 정보를 변경 할 필요가 있을 때 쿠키를 업데이트 하여 변경된 쿠키를 HTTP 헤더에 포함시켜 응답한다.

쿠키 사용 예시

- 방문 사이트에서 로그인 시, "아이디와 비밀번호를 저장하시겠습니까?"
- 쇼핑몰의 장바구니 기능
- 자동로그인, 팝업에서 "오늘 더 이상 이 창을 보지 않음" 체크, 쇼핑몰의 장바구니

세션

세션이란?

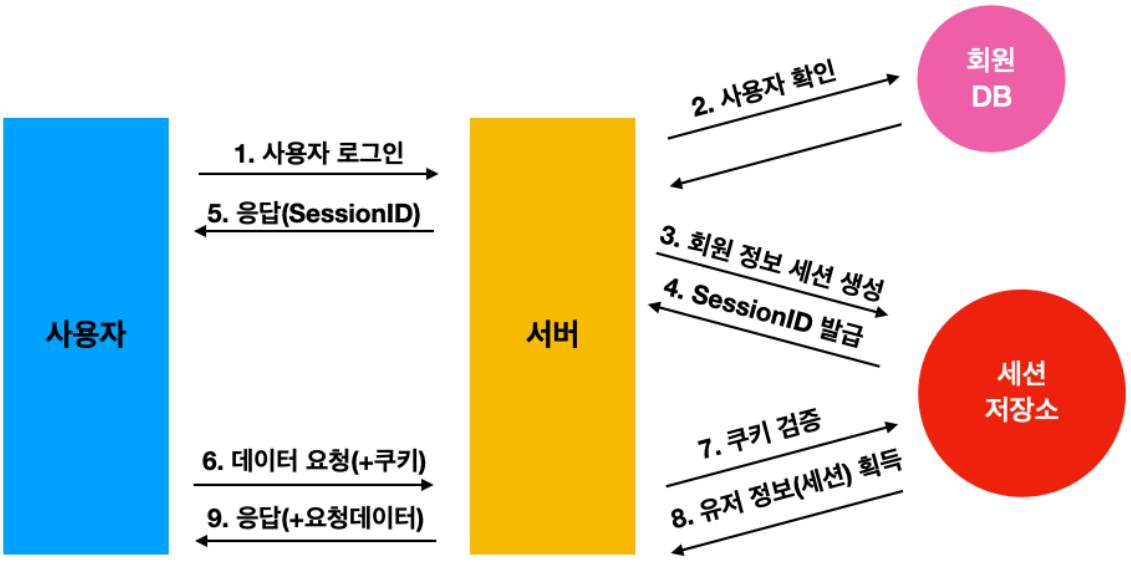
- 방문자가 웹서버에 접속해있는 상태를 하나의 단위로 보고 이를 세션이라 한다

세션 특징

- 웹 서버에 **웹 컨테이너의 상태를 유지하기 위한 정보를 저장**한다.
    - 세션에 대한 정보가 서버에 있어 쿠키에 비해 비교적 속도가 느리다.
- 쿠키보다 상대적으로 보안이 좋다.

    - 쿠키는 클라이언트 쪽에 저장돼서 보안 이슈가 있기 때문에, 민감한 인증 정보를 서버에 저장하고 관리하는 세션을 사용한다.
    - 해커가 세션 ID 자체를 탈취하여 클라이언트인척 위장할 수 있다는 한계가 존재
- 저장 데이터에 제한이 없다.
    - 하지만 데이터가 많아질 경우 서버 메모리를 많이 차지하게 되어 서버에 부하가 발생할 수 있다.
- 각 클라이언트에 고유 Session ID를 부여하고, 클라이언트는 쿠키에 Session ID를 저장해 둔다.
    - (세션은 기본적으로 쿠키를 이용하여 구현된다)
- 세션도 만료시간을 정할 수 있지만 브라우저가 종료되면 만료시간에 상관없이 삭제된다.
    - 다른 브라우저를 사용하게 되면 다른 세션을 사용하게 된다. 크롬에서 다른 탭을 사용하는 경우 같은 브라우저이므로 세션은 공유됨.

세션 동작 방식



1. 사용자가 로그인을 요청한다
2. 서버에서 계정 정보를 읽어 사용자를 확인한다.

3. 사용자의 고유한 SessionId를 부여하여 세션 저장소에 저장한후, 이와 연결된 세션Id를 발급한다.
4. 사용자는 서버에서 해당 세션Id를 받아 쿠키에 저장한 후, 인증이 필요한 요청마다 쿠키를 헤더에 실어 보낸다.
5. 서버는 쿠키를 받아 세션 저장소에서 대조 후 대응되는 정보를 가져온다.
6. 인증이 완료되면 서버는 사용자에 맞는 데이터를 보내준다.

쿠키 vs 세션

	Cookie	**Session**
저장 위치	Client	Server
저장 형식	Text	Object
만료 시점	쿠키 저장 시 설정(설정 없으면 브라우저 종료 시)	정확한 시점 모름
리소스	클라이언트의 리소스	서버의 리소스
용량 제한	한 도메인 당 20개, 한 쿠키 당 4KB	제한 없음
속도	쿠키에 정보가 있어 비교적 빠름	정보가 서버에 있어 비교적 느림

토큰 (JWT)

- JWT 기반 인증은 JWT 토큰(Access Token)을 HTTP 헤더에 실어 서버가 클라이언트를 식별하는 방식이다
- Access Token과 Refresh Token
  - Access Token: 접근에 관여하는 토큰!
    - 클라이언트가 갖고있는 실제로 유저의 정보가 담긴 토큰
    - 클라이언트에서 요청이 오면 서버에서 해당 토큰에 있는 정보를 활용하여 사용자 정보에 맞게 응답을 진행
  - Refresh Token: 재발급에 관여하는 토큰!
    - 새로운 Access Token을 발급해주기 위해 사용하는 토큰
    - 짧은 수명을 가지는 Access Token에게 새로운 토큰을 발급해주기 위해 사용.

JWT 토큰 구성 요소

- JWT는 **.** 을 구분자로 나누어지는 세가지 문자열의 조합이다.
  - **.** 을 기준으로 Header, Payload, Signature을 의미한다.
- Header
  - JWT에서 사용할 타입과 해시 알고리즘의 종류가 담겨있다
- Payload
  - 서버에서 첨부한 사용자 권한 정보와 데이터가 담겨있다
  - Payload는 해독이 가능하기 때문에 중요한 정보는 포함해서는 안된다.
- Signature
  - Header, Payload를 Base64 URL-safe Encode를 한 이후 Header에 명시된 해시 함수를 적용하고, 개인키로 서명한 전자서명이 담겨있다.

JWT 토큰 장단점

장점

- header와 payload 를 가지고 signaure를 생성하므로 데이터 위변조를 막을 수 있다.

- 인증 정보에 대한 별도의 저장소가 필요없다. (서버 부하 ↓)
- JWT는 토큰에 대한 기본 정보과 전달할 정보 및 토큰이 검증 되었다는 서명 등 필요한 모든 정보를 자체적으로 지니고 있다.
- 토큰은 한 번 발급되면 유효기간이 만료될 때까지 계속 사용이 가능하다.

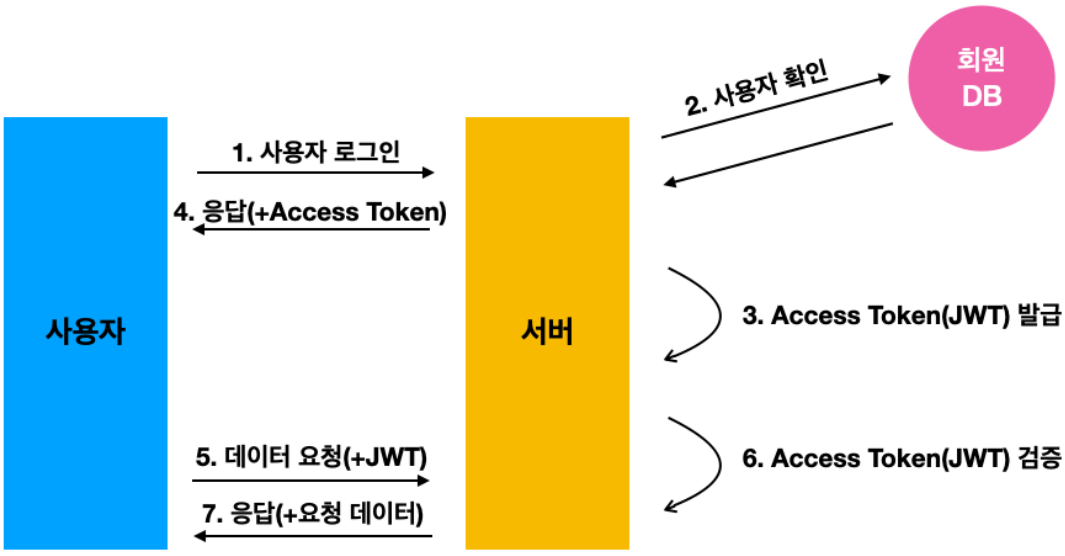
단점

- 쿠키나 세션과 다르게 JWT는 토큰의 길이가 길어 인증 요청이 많아질수록 네트워크 부하가 심해진다.
- payload 자제는 암호화되지 않기 때문에 유저의 중요한 정보를 담으면 안된다.
- 토큰을 탈취당하면 대처하기 어렵다.
- 특정 사용자의 접속을 강제로 만료하기 어렵지만, 쿠키/세션 기반 인증은 서버 쪽에서 쉽게 세션을 삭제 할 수 있다.

쿠키&세션과 비교한 JWT 토큰 기반 인증 방식의 장단점

	Cookie & Session	JWT
장점	- Cookie만 사용하는 방식보다 보안 향상 - 서버쪽에서 Session 통제 가능 - 네트워크 부하	- 인증을 위한 별도의 저장소가 필요 없음 - 별도의 I/O 작업 없는 빠른 인증 처리 - 확장성이 우수함
단점	- 세션 저장소 사용으로 인한 서버 부하	- 토큰의 길이가 늘어날수록 네트워크 부하 - 특정 토큰을 강제로 만료시키기 어려

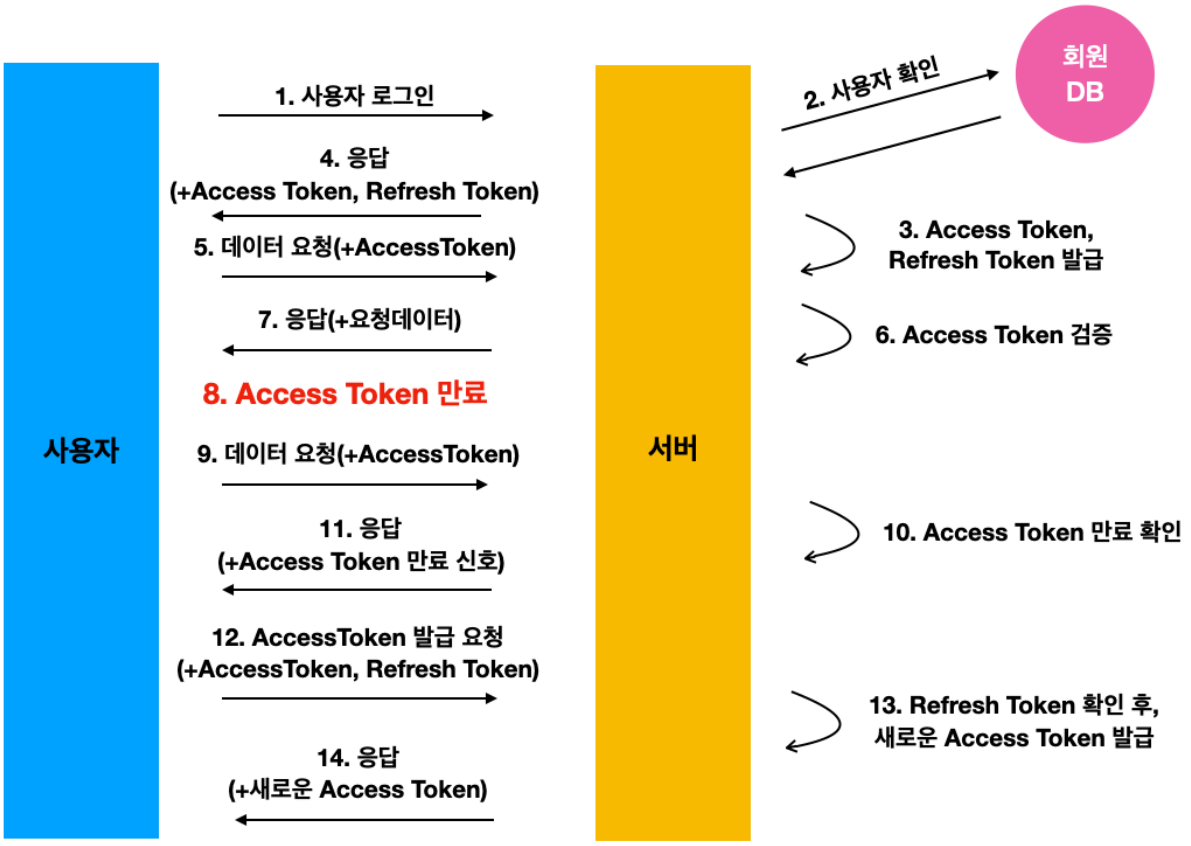
JWT 토큰 인증 방식 (Access Token 사용)



1. 사용자가 아이디와 비밀번호로 로그인을 한다.
2. 서버측에서 계정 정보를 읽어 사용자를 확인 후, 사용자에게 유일한 토큰을 발급한다.
3. 클라이언트는 서버측에서 전달받은 토큰을 쿠키나 스토리지에 저장해두고,

   서버에 데이터 요청을 할 때마다 해당 토큰을 HTTP요청 헤더에 포함시켜 전달한다.
4. 서버는 전달받은 토큰을 검증하고 요청에 응답한다.
   - 토큰에는 요청한 사람의 정보가 담겨있어서 서버는 DB를 조회하지않고 누가 요청했는지 알 수 있다.

JWT 토큰 인증 방식 (Access Token, Refresh Token 사용)



1. 사용자가 ID, PW를 통해 로그인
2. 서버에서는 회원 DB에서 값을 비교한다. (보통 PW는 암호화해서 들어간다.)
3. 사용자 인증이 되면 서버에서 Access Token, Refresh Token을 발급, 보통 회원 DB에 Refresh Token을 저장
4. 서버는 사용자에게 Access Token, Refresh Token을 보낸다.
5. 사용자는 Refresh Token을 안전한 저장소에 저장 후, Access Token을 헤더에 실어 요청을 보낸다.
6. 서버는 Access Token을 검증 후
7. 이에 맞는 데이터를 사용자에게 보내준다.
8. 시간이 흘러 Access Token이 만료
9. 사용자는 만료된 Access Token을 헤더에 실어 요청을 보낸다.
10. 서버는 Access Token이 만료됐음을 확인
11. 서버는 클라이언트에게 만료된 토큰임을 알리고 권한없음을 신호로 보낸다.
    - Access Token이 만료될때 마다 9~11 과정을 거칠 필요는 없다.

Access Token의 Payload를 통해 유효기간을 알 수 있다.

따라서, 프론트엔드 단에서 API 요청전에 토큰이 만료 됐다면 바로 재발급 요청 가능

12. 사용자는 Refresh Token 과 Access Token을 함께 서버로 보낸다.
13. 서버는 받은 Access Token이 조작되지 않았는지 확인하고,

    Refresh Token과 사용자의 DB에 저장되어 있던 Refresh Token을 비교한다.
14. 서버는 Refresh Token이 동일하고 유효기간도 지나지 않았다면 새로운 Access Token을 발급해 사용자에게 보내준다.
15. 새로운 Access Token을 헤더에 실어 API 요청을 한다.