

졸업 작품 보고서



---

---

# JWT 인증과 권한을 이용한 웹기반 채팅

---

---

2023년 11월 24일

서경대학교 컴퓨터공학과

정동기

# 목 차

|                      |    |
|----------------------|----|
| I . 서론               | 2  |
| 1. 제 작 동 기           | 2  |
| 2. 제 작 목 적           | 2  |
| 3. 작 품 설 명           | 3  |
| 4. 작 품 효 과           | 3  |
| II . 개발 내용           | 4  |
| 1. 개발 환경             | 4  |
| 2. J W T             | 5  |
| (1) JWT 인증 방법        | 5  |
| (2) JWT 장 점          | 6  |
| (3) JWT 단점 및 보완      | 6  |
| 3. 권 한               | 8  |
| 4. 시 스템 구 조          | 9  |
| 5. 로 그 인             | 9  |
| (1) 회 원 가 입          | 9  |
| (2) 로 그 인            | 10 |
| 가. 성 공               | 10 |
| 나. 실 패               | 11 |
| (3) 비 밀 번 호 변 경      | 11 |
| 6. J W T 인 증         | 12 |
| 7. 사 용 자 관 리         | 14 |
| 8. 게 시 판 및 댓 글, 파 일  | 16 |
| 9. 채 텅               | 19 |
| (1) WebSocket과 STOMP | 19 |

|                        |    |
|------------------------|----|
| (2) WebSocket 설정 ..... | 19 |
| (3) 채팅과 권한 .....       | 20 |
| Ⅲ. 개발 결과 .....         | 23 |
| Ⅳ. 결론 .....            | 24 |
| 1. 현재 프로젝트 성과 .....    | 24 |
| 2. 아쉬운 점 .....         | 24 |
| 3. 향후 개선 방안 .....      | 24 |
| 참고자료 .....             | 25 |

# I. 서론

## 1. 제작 동기

코로나 기간과 더불어 사람들이 인터넷을 쉽게 사용함에 따라 많은 사람들이 일이나 협업에서 메신저를 사용하고 있다. 이러한 변화는 효율적인 협업과 커뮤니케이션의 필요성을 증가시키게 되었고 이러한 메신저들 중에 보안이 강력한 협업툴이 인기를 끌게 되었다.

대학 동기들과 과제나 스터디를 할 때 카카오톡이나 디스코드를 사용하던 나에게 협업툴과 같은 보안이 강력한 메신저가 매력적으로 다가왔고, 보안을 강화한 메신저를 스스로 구현해보고 싶었다.

## 2. 제작 목적

보안을 강화한 메신저를 구현하는 것이 최종 목표이다.

쿠키, 세션, JWT 인증 방식 중에 JWT로 인증을 구현하려고 한다. JWT의 보안상의 문제점을 보완하고 사용자들 사이의 권한을 지정해 권한이 높은 사용자에게는 많은 기능을 부여하고 사용자를 관리할 수 있게 만들어 보안을 강화하고자 한다.

메신저의 기능으로는 크게 로그인, 사용자 관리, 게시판, 채팅이 있으며, 채팅을 통해 사용자 간의 의견을 주고받을 수 있으며, 게시판을 이용하여 공지나 알리고 싶은 이야기를 올릴 수 있도록 해주하고자 한다.

### 3. 작품 설명

JWT(JSON Web Token) 인증을 통한 게시판과 채팅 기능을 사용할 수 있는 사이트이다. 사용자는 JWT를 가지고 HTTP 통신 시에 서버에서 토큰 인증을 받게 되고 인증에 성공하면 사이트에 접근이 가능하게 된다. 또한 인증 후 자신의 권한에 따라 접근할 수 있는 기능이 달라진다.

권한은 총 5가지로 구성되어 있다. GUEST, SEMIUSER, USER, MANAGER, ADMIN 으로 구성되어 있고 각 권한마다 사용할 수 있는 기능의 범위가 달라진다.

### 4. 작품 효과

JWT를 사용함으로써 데이터베이스와의 통신을 최소화 시켜 부하를 줄일 수 있으며, 사용자 인증 정보의 보안을 강화한다. 인증 정보에 있는 권한에 따라서 사용자의 사이트 접근 범위가 달라지며, 권한이 높은 사용자는 권한이 낮은 사용자를 관리하는 방법을 통하여 보안을 강화한다.

## Ⅱ. 개발 내용

### 1. 개발 환경

#### Front-End

- Jquery
- AJAX
- Socket.js
- Stomp.js

#### Back-End

- SpringBoot
  - JPA
  - security
  - JWT
  - WebSocket
  - Stomp
  - mail
  - QueryDSL
  - Thymeleaf

#### DataBase

- MySQL

## 2. JWT

JWT는 당사자 간에 정보를 JSON 개체로 안전하게 전송하기 위한 토큰이다. 이 정보는 디지털 서명이 되어 있으므로 확인하고 신뢰할 수 있다.

여기서 서명을 설명하기에 앞서 JWT의 구성 요소에 대해서 알아보면, JWT는 Header, Payload, Signature로 구성되어 있다. 여기서 Signature가 서명이다.

Signature를 만드는 방법은 Header, Payload, secretkey를 HMAC SHA256 암호화 알고리즘을 통해 암호화하는 것이다. 여기서 secretkey는 서버가 가지고 있는 값으로 서명을 작성할 때 사용한다.

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret)
```

그림 2 Signature

JWT는 Header, Payload, Signature를 합친 값이다.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.  
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

그림 3 JWT

### (1) JWT 인증 방법

JWT 인증 방법은 사용자가 JWT를 가지고 있다가 사이트에 접근하고자 하면 HTTP 통신 시 JWT를 같이 보내준다. 서버에서는 JWT 토큰에 있는 Header와 Payload를 가지고 secretkey와 함께 Signature를 만든다. 이후 사용자가 보내준 JWT의 Signature와 서버에서 만든 Signature를 비교하여 값이 일치하면 인증이 되고 틀리면 실패하게 된다.

## (2) JWT 장점

1. 인증 정보의 상태를 유지, 갱신할 필요가 없다.

앞서 설명한 것처럼 JWT는 서버에서 서명을 확인하는 방법으로 인증을 한다. 그러므로 인증을 할 때에 DB에 접근을 최소화 하므로, DB의 부하를 막아줄 수 있다.

2. secretkey가 있다면 어디서든 접근이 가능하다.

JWT는 secretkey만 있다면 서버를 추가하더라도 따로 서버 세션이나 DB를 늘릴 필요가 없다. 사용자가 JWT를 가지고 있으면 서버에 secretkey를 이용해 인증을 할 수 있기 때문이다.

3. HTTP Body값이 탈취되어도 Token이 없으면 사용할 수 없다.

값이 있더라도 JWT가 없다면 인증이 불가능하기 때문이다.

4. 서명으로 토큰 변경 여부를 알 수 있다.

Signature는 Header와 Payload로 생성하기 때문에 이 부분이 변경되면 바로 알 수 있게 된다.

## (3) JWT 단점 및 보완

JWT가 탈취되게 되면 다른 사용자가 인증을 할 수 있게 된다. 이러한 단점을 보완하기 위해 실제 프로젝트에 적용한 것들을 소개하겠다.

1. HttpOnly 속성 추가

HttpOnly 속성을 추가하여 JS를 통한 접근을 막아주었다.

2. accessToken의 유효기간을 줄였다.

accessToken은 JWT를 인증에 사용한 것으로 토큰의 유효기간을 줄임으로써 탈취 당하더라도 오래 사용 못하게 해주었다.



3. refreshToken을 이용하여 accessToken을 발급 받는다.

accessToken의 유효기간이 줄어들므로써 토큰이 없을 때마다 로그인을 해야 하는 번거로움이 생길 수 있다. 이를 막기 위해 refreshToken이라는 JWT를 만들어주어 accessToken이 없더라도 refreshToken이 있다면 accessToken을 재발급할 수 있게 해주었다.

#### 4. Rotating Refresh Token

그렇다면 RefreshToken이 탈취당한다면 accessToken을 다른 사용자가 계속해서 받아서 사용할 수 있게 된다는 문제점이 발생하게 된다. 이를 막기 위해 refreshToken을 한 번 사용하면 refreshToken을 재발급하여 한 번 사용한 refreshToken으로 accessToken을 발급받지 못하도록 해주었다.

### 3. 권한

권한은 ADMIN, MANAGER, USER, SEMIUSER, GUEST로 나뉘어져 있다. 권한에 따라서 사용자를 관리하거나 접근 권한을 나누어 권한이 높은 사용자가 낮은 사용자를 관리하는 것으로 보안을 강화한다. 권한에 따른 기능은 추후 시스템 설명시 설명하도록 하겠다.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception{
    http
        .exceptionHandling(exp ->
            exp.accessDeniedPage( accessDeniedUrl: "/error"))
        .formLogin(login -> login.disable())
        .httpBasic(basic -> basic.disable())
        .csrf(csrf -> csrf.disable())
        .sessionManagement(session ->
            session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(auth -> // URL 별 권한 관리 옵션
            auth
                .requestMatchers( ...patterns: "/css/**", "/js/**").permitAll()
                .requestMatchers( ...patterns: "/", "/signup", "/loginForm", "/password", "/favicon.ico", "/error").permitAll()
                .requestMatchers( ...patterns: "/login", "/signup").permitAll()
                .requestMatchers( ...patterns: "/files/**").hasAnyRole( ...roles: "USER", "ADMIN", "MANAGER")
                .requestMatchers( ...patterns: "/board/**", "/chat/**").hasAnyRole( ...roles: "SEMIUSER", "USER", "ADMIN", "MANAGER")
                .requestMatchers( ...patterns: "/updateauthorization").hasAnyRole( ...roles: "ADMIN", "MANAGER")
                .requestMatchers( ...patterns: "/user/updateAuth").hasAnyRole( ...roles: "ADMIN", "MANAGER")
                .anyRequest().authenticated() // 인증된 사용자만 접근 가능
            )
        .addFilterAfter(customUsernamePasswordAuthenticationFilter(), LogoutFilter.class)
        .addFilterBefore(jwtAuthenticationProcessingFilter(), CustomUsernamePasswordAuthenticationFilter.class); // 로그인 후에는 모든 요청이 JWT 필터를 거치게 된다.

    return http.build();
}
```

그림 4 SecurityConfig http

위의 코드는 권한을 나누어주는 코드이다. 코드에 대해 간단히 설명하자면, permitAll()은 모든 접근을 허용하는 것이고, hasRole과 hasAnyRole은 지정한 권한이 있는 사용자만 접근을 허용한다는 코드이다.

## 4. 시스템 구조

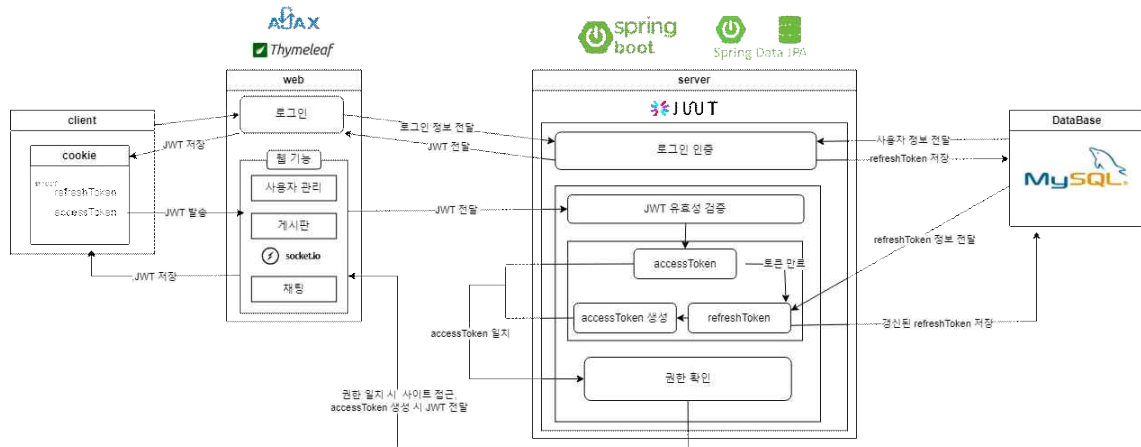


그림 6 프로젝트 구조도

프로젝트 구조도이다. 구조도에서 주요 기능을 뽑아보면 로그인, JWT 인증, 사용자 관리, 게시판, 댓글, 파일, 채팅이 있다.

## 5. 로그인

로그인을 세분화하면 회원가입, 로그인, 비밀번호 찾기로 나눌 수 있다.

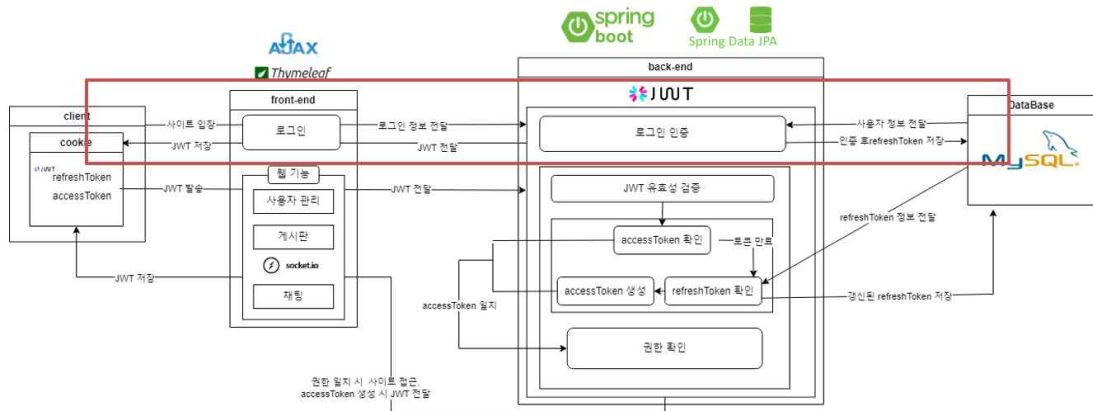


그림 7 프로젝트 구조도 - 로그인

### (1) 회원가입

회원가입을 위해 필요한 값들이 있다. 바로 이름, 이메일, 비밀번호이다. 이 값들을 정규식을 이용하여 받아주었다. 이름은 영어 또는 한글로, 이메일은 이메일 형식

으로, 비밀번호는 대문자, 소문자, 숫자, 특수 문자 한 개 이상의 8자리 이상의 값으로 받아주었다. 이렇게 들어온 값을 DB에 저장을 하는데, 이때 비밀번호는 암호화 처리를 해서 저장을 한다. 이때 사용한 암호화 방식은 bcrypt이다.

bcrypt란 SHA-256 해시 알고리즘 기반으로 salt를 추가하여 같은 비밀번호라도 다른 결과를 생성하는 암호화 방식이다. 이 암호화 방식의 특징으로는 복호화가 불가능하다는 점이다. 복호화가 불가능하기 때문에 비밀번호를 잃어버리게 된다면 새로 발급받아야 한다.

회원가입을 하게 되면 기본 권한이 GUEST로 지정된다. GUEST는 내 정보나 상대방 정보를 보는 것 말고는 아무런 접근이 불가능하다.

회원 가입을 하게 되면 DB가 아래와 같이 저장된다.

|              |                   |      |      |               |      |       |                |
|--------------|-------------------|------|------|---------------|------|-------|----------------|
| 2023-11-2... | 2023-11-23 15:... | 6    | 예비   | {bcrypt}\$... |      | GUEST | test5@test.com |
| HULL         | HULL              | HULL | HULL | HULL          | HULL | HULL  | HULL           |

그림 8 userDB

## (2) 로그인

로그인은 이메일과 패스워드를 정규식에 맞게 작성하면 이 값을 post 형식으로 “/login”으로 보내게 된다.

이 POST 형식의 값을 받아주기 위해 customFilter를 이용하였다. 이때 사용하는 Filter가 CustomUsernamePasswordAuthenticationFilter이다. 이 필터는 AbstractAuthenticationProcessingFilter라는 스프링 시큐리티 기본 인증 필터를 확장하여 구현하였다.

이 코드에서 이메일과 비밀번호를 가지고 UsernamePasswordAuthenticationToken을 생성한다. 이 토큰은 스프링 시큐리티 인증 기본 객체이다. 이제 이 값을 이용해 DaoAuthenticationProvider()에 토큰을 전달하여 사용자 정보를 조회한다. 이후 비밀번호를 검증하고 맞다면 성공(loginSuccessHandler), 틀리면 실패(loginFailureHandler)로 가게 된다.

### 가. 성공

loginSuccessHandler로 가게 된다. 이 코드에서는 accessToken과 refreshToken을 발급받아 사용자의 쿠키에 저장을 할 수 있도록 해준다. refreshToken은 DB에 저

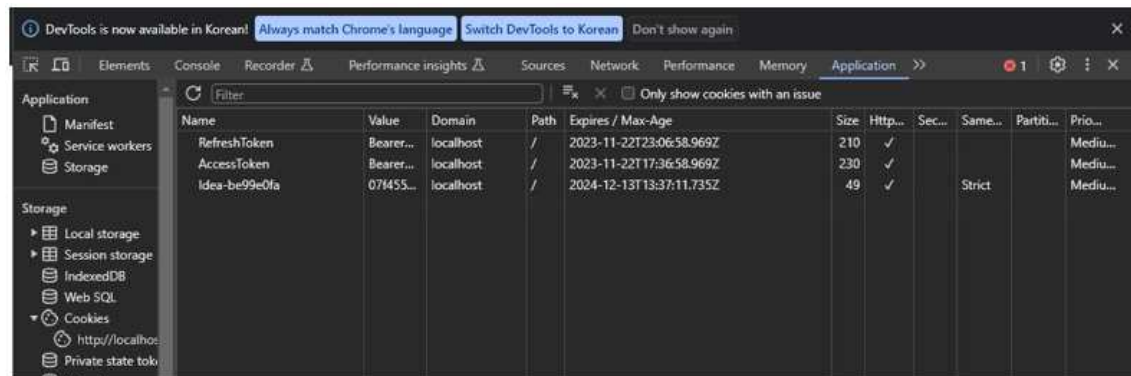
장해준다.

DB에 저장된 값이다.

| createdDate  | lastModifiedDate  | user_id | nickname | password     | refreshToken     | role     | username            |
|--------------|-------------------|---------|----------|--------------|------------------|----------|---------------------|
| 2023-11-1... | 2023-11-23 01:... | 1       | 정동기      | (bcrpt)\$... | Bearer eyJhbG... | ADMIN    | dirhank45@gmail.com |
| 2023-11-1... | 2023-11-23 02:... | 2       | 서활민      | (bcrpt)\$... | Bearer eyJhbG... | MANAGER  | test1@test.com      |
| 2023-11-1... | 2023-11-22 11:... | 3       | 여영       | (bcrpt)\$... | Bearer eyJhbG... | USER     | test13@test.com     |
| 2023-11-1... | 2023-11-22 18:... | 4       | as       | (bcrpt)\$... | Bearer eyJhbG... | SEMIUSER | test4@test.com      |
| 2023-11-2... | 2023-11-22 11:... | 5       | 안녕       | (bcrpt)\$... | Bearer eyJhbG... | GUEST    | test@test.com       |

그림 9 userDB

Cookie에 저장된 값이다.



| Name          | Value     | Domain    | Path | Expires / Max-Age        | Size | Http... | Sec... | Same... | Partiti... | Prio...  |
|---------------|-----------|-----------|------|--------------------------|------|---------|--------|---------|------------|----------|
| RefreshToken  | Bearer... | localhost | /    | 2023-11-22T23:06:58.969Z | 210  | ✓       |        |         |            | Mediu... |
| AccessToken   | Bearer... | localhost | /    | 2023-11-22T17:36:58.969Z | 230  | ✓       |        |         |            | Mediu... |
| Idea-be99e0fa | 076455... | localhost | /    | 2024-12-13T13:37:11.735Z | 49   | ✓       |        | Strict  |            | Mediu... |

그림 10 Cookie

## 나. 실패

BAD\_REQUEST를 보낸다.

### (3) 비밀번호 변경

비밀번호는 암호화 방식이 복호화를 할 수 없기 때문에 값을 받아서 변경할 수가 없다. 비밀번호를 변경하기 위해서는 재발급을 받은 후 비밀번호를 변경하는 방법을 이용해야 한다.

이를 구현하기 위해 자신이 사용하는 이메일 값을 넣으면 그 이메일에 비밀번호 정규식에 맞는 랜덤한 값을 출력해서 작성한 이메일로 새로운 비밀번호를 보내준다.

새로운 비밀번호를 받은 사용자는 로그인을 한 후 내 정보 페이지에서 비밀번호

를 변경하여 사용할 수 있다.

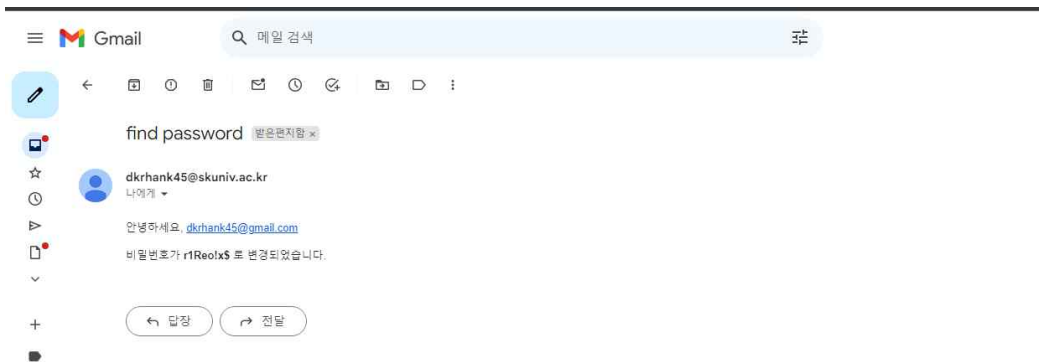


그림 11 비밀번호 메일

## 6. JWT 인증

JWT 발급 및 인증은 아래와 같은 구조도로 진행이 된다.

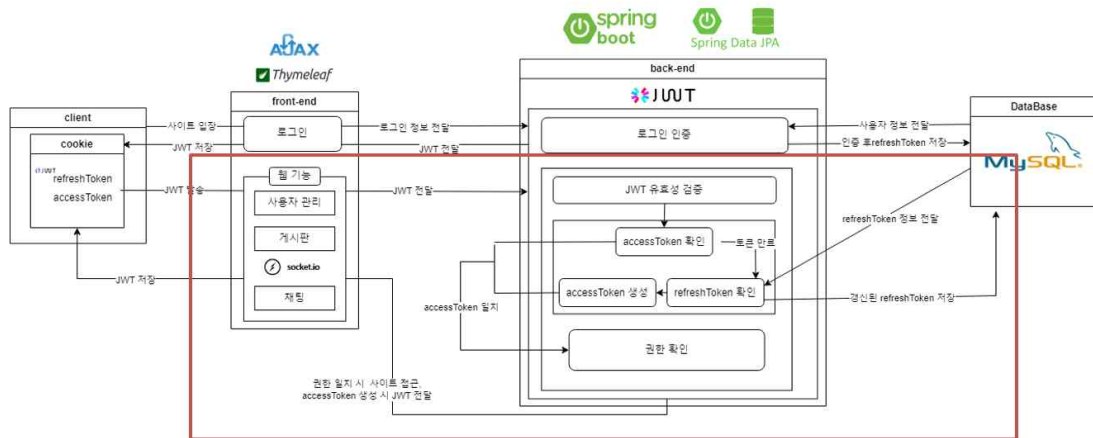


그림 12 인증 구조도

### 1. 인증이 없어도 접근 가능

인증이 없어도 접근 가능한 URL과 정적 리소스를 지정해주고 JWTFilter에서 인증을 진행하지 않게 한다.

접근 가능한 URL은 “/”, “/login”, “/signup”, “/password”, “/favicon.ico”, “/error” 이다. 이 URL은 앞선 로그인 기능에서 사용했던 회원가입, 로그인, 비밀번호 찾기

와 기본 페이지, 에러 페이지를 의미한다.

정적 리소스는 “/css/\*\*” , “/js/\*\*”를 무시해주었다.

## 2. 인증

인증이 필요한 URL로 들어오게 되면 JWT 인증을 시작하게 된다.

### 1) accessToken과 refreshToken이 둘 다 있을 경우

accessToken이 있을 경우에는 accessToken의 사용 가능 여부에 따라 인증을 해준다.

### 2) accessToken만 있는 경우

앞선 내용과 마찬가지로 accessToken의 사용 가능 여부에 따라 인증을 해준다.

### 3) refreshToken만 있는 경우

refreshToken의 사용 가능 여부에 따라 사용이 가능하다면 accessToken과 refreshToken을 생성 후 리프레시 토큰은 DB에 저장을 한다. 이후 accessToken과 refreshToken을 쿠키로 보내주고, 인증을 한다.

### 4) 둘 다 없는 경우

인증을 거부한다.

## 7. 사용자 관리

프로젝트의 설명에서 권한이 높은 사용자가 권한이 낮은 사용자를 관리하여 보안을 강화한다고 하였다. 사용자 관리는 세부적으로 ADMIN, MANAGER, 내 정보로 나뉜다.

### 1. 사용자 관리 - ADMIN

사용자 관리 - ADMIN

| 이메일                 | 이름  | 권한 설정    | 상세 조회 | 회원 삭제 |
|---------------------|-----|----------|-------|-------|
| dkrhank45@gmail.com | 정동기 | ADMIN    | 변경    | 삭제    |
| test2@test.com      | 사람인 | SEMIUSER | 변경    | 삭제    |
| test3@test.com      | 예잉  | USER     | 변경    | 삭제    |
| test4@test.com      | as  | SEMIUSER | 변경    | 삭제    |
| test@test.com       | 안녕  | GUEST    | 변경    | 삭제    |
| test5@test.com      | 예비  | GUEST    | 변경    | 삭제    |

그림 13 사용자관리- ADMIN

ADMIN이 사용자를 관리하는 방법으로는 권한 설정과 회원 삭제가 있다.

처음 회원가입을 한 사용자는 기본적으로 권한이 GUEST로 지정되어 있다. ADMIN은 권한을 지정해주어 사용자에게 따라 접근할 수 있는 기능을 나누어 준다. 이를 통해 다른 사용자들을 관리하여 보안을 강화할 수 있게 된다.

회원 삭제를 통해 프로그램에 맞지 않는 사람을 추방할 수도 있다.

### 2. 사용자 관리 - MANAGER

사용자 관리 - MANAGER

| 이메일            | 이름  | 권한 설정    | 상세 조회 |
|----------------|-----|----------|-------|
| test2@test.com | 사람인 | MANAGER  | 변경    |
| test3@test.com | 예잉  | USER     | 변경    |
| test4@test.com | as  | SEMIUSER | 변경    |
| test@test.com  | 안녕  | GUEST    | 변경    |
| test5@test.com | 예비  | GUEST    | 변경    |

그림 14 사용자 관리 -MANAGER



MANAGER가 사용자를 관리하는 방법으로는 권한 설정이 있다.

MANAGER가 ADMIN보다 권한이 낮기 때문에 ADMIN의 권한을 지정할 수 없다. 다른 권한을 가진 사용자들은 MANAGER가 권한 지정을 해주어 다른 사용자들을 관리하여 보안을 강화할 수 있게 된다.

### 3. 내 정보

#### 내 정보

| 이메일            | 이름  | 권한      |
|----------------|-----|---------|
| test2@test.com | 사람인 | MANAGER |

비밀번호 변경

이름 변경

회원탈퇴

어드민 키

어드민 키를 입력하세요

어드민 변경

그림 15 내 정보

내 정보에서는 기본적인 정보와 비밀번호 변경, 이름 변경, 회원 탈퇴, 어드민 변경이 있다.

#### 1. 비밀번호 변경

현재 비밀번호와 변경하고 싶은 비밀번호 값을 받아 현재 비밀번호가 일치하는지 확인하고 비밀번호를 변경한다.

#### 2. 이름 변경

변경하고 싶은 이름을 받아 변경한다.

#### 3. 회원 탈퇴

현재 비밀번호를 받아 비밀번호가 일치하게 되면 사용자를 DB에서 삭제한다.

#### 4. 어드민 변경

환경 변수에 어드민 키를 만들어두고 똑같은 값을 어드민 키에 넣고 변경 버튼을 누르게 되면 권한이 ADMIN으로 변경된다.

## 8. 게시판 및 댓글, 파일

게시판은 게시글을 모아놓은 공간이다. 권한에 따라서 게시판의 기능을 이용할 수 있는 범위를 나누어주었다.

게시판을 세부적으로 나누게 되면 게시글 저장, 게시글 수정 및 삭제, 게시글 읽기, 게시글 페이징, 게시글 서칭으로 나뉘게 된다.

GUEST 권한의 사용자는 게시판 기능을 이용할 수 없다.

### 1. 게시글 저장

#### ADMIN, MANAGER

게시글 작성

☐ 내 권한 아래 접근 금지

|    |   |
|----|---|
| 제목 | <input type="text" value="Enter title"/>    |
| 내용 | <div><div>Enter contents</div></div>        |
| 파일 | <div>파일 선택   선택된 파일 없음</div> <div>글작성</div> |

그림 16 저장 - ADMIN, MANAGER

#### USER, SEMIUSER

게시글 작성

|    |   |
|----|---|
| 제목 | <input type="text" value="Enter title"/>    |
| 내용 | <div><div>Enter contents</div></div>        |
| 파일 | <div>파일 선택   선택된 파일 없음</div> <div>글작성</div> |

그림 17 저장 - USER, SEMIUSER

게시글 작성에서도 권한에 따라서 사용할 수 있는 기능이 다르다. 앞선 사진을 보면 ADMIN과 MANAGER에게는 내 권한 아래 접근 금지라는 기능이 있다. 이 기능은 글쓴이의 권한보다 아래인 사용자들은 게시글에 접근이 불가능하도록 해주는 기능이다.

게시글을 저장할 때 필요한 값들은 제목, 내용, 파일이다. 여기서 제목과 내용은

무조건 있어야 하는 값으로 파일값은 없어도 저장이 된다.

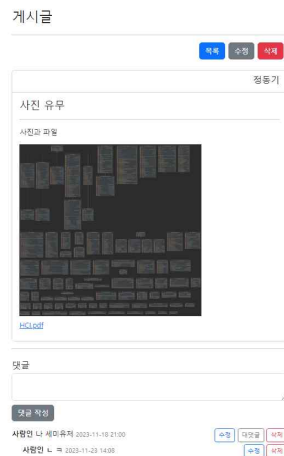
파일을 저장할 때에는 파일이 실행파일, 스크립트 및 자동화 파일, 매크로가 포함된 오피스 문서 파일, 웹 파일 및 스크립트일 경우에는 저장을 할 수 없도록 하였다.

## 2. 게시물 수정 및 삭제

게시글을 사용할 수 있는 사용자들은 모두 글을 쓴 사용자여야 한다.

## 3. 게시물 읽기

### ADMIN, MANAGER, USER



### 그림 18 읽기 - ADMIN MANAGER, USER

### SEMIUSER



### 그림 19 읽기 - SEMIUSER

작성된 게시글을 보는 상황에서도 권한에 따라 사용할 수 있는 기능이 달라진다. 사진을 보게 되면 SEMIUSER는 사진이나 파일에 접근할 수 없다.

댓글이나 대댓글의 경우에는 게시글을 사용할 수 있는 사용자는 모두 사용할 수 있게 해주었고 수정이나 삭제는 댓글을 쓴 본인일 경우에만 삭제할 수 있도록 해주었다.

#### 4. 게시글 페이징

##### 게시판

게시글 저장

제목

검색

| 제목           | 글쓴이 | 날짜               | 권한지정 |
|--------------|-----|------------------|------|
| 메니저 권한       | 사람인 | 2023-11-21 17:44 | ●    |
| 피어쓰기 있는 파일   | 정동기 | 2023-11-18 21:17 |      |
| 권한 있고 사진도 있어 | 정동기 | 2023-11-18 20:35 | ●    |
| 사진 유료        | 정동기 | 2023-11-17 22:03 |      |

1
2
3
4
5

그림 20 게시글 페이징

Pageable을 사용하여 페이지를 지정해주었다.

#### 5. 게시글 서칭

##### 찾으시는 게시물

제목

검색

| 제목           | 글쓴이 | 날짜               | 권한 지정 |
|--------------|-----|------------------|-------|
| 메니저 권한       | 사람인 | 2023-11-21 17:44 | ●     |
| 권한 있고 사진도 있어 | 정동기 | 2023-11-18 20:35 | ●     |

1
2
3
4
5

등록으로

그림 21 게시글 서칭 - 제목: 권한

게시글 서칭은 제목, 내용, 제목+내용으로 찾을 수 있도록 해주었다. 결과에 맞는 게시글을 찾으면 페이징 처리를 해주어 좀 더 편하게 볼 수 있도록 해주었다.

이는 QueryDSL를 사용하여 구현하였다. QueryDSL은 Java코드로 쿼리를 작성할 수 있도록 해주는 라이브러리이다.

## 9. 채팅

채팅은 WebSocket과 STOMP를 기반으로 작성하였고 채팅에서도 권한에 따라 사용할 수 있는 기능이 달라지게 된다.

### (1) WebSocket과 STOMP

#### 1. WebSocket

웹에서 실시간, 양방향, 전이중 통신을 가능하게 하는 기술이다.

쉽게 이야기해서 처음 HTTP 통신을 할 때 웹소켓을 연결하고 데이터를 교환을 하는 방식이다.

#### 2. STOMP

Simple Text Oriented Messaging Protocol의 약자로 간단한 텍스트 기반 프로토콜이다. 구독과 발행을 통해 메시지를 주고 받을 수 있도록 해준다.

#### 3. WebSocket과 STOMP

STOMP를 웹소켓 위에서 사용함으로써, 메시징 관련 명령어(ex: CONNECT, SUBSCRIBE, SEND)를 이용해서 쉽게 메시지를 보낼 수 있게 된다.

### (2) WebSocket 설정

```
@Configuration
@EnableWebSocketMessageBroker
@RequiredArgsConstructor
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    1 usage
    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry){
        registry.enableSimpleBroker( ...destinationPrefixes: "queue", "/sub"); // 받는 사람 Queue; Sub: 서버가 클라이언트에게
        registry.setApplicationDestinationPrefixes("/pub"); // 보내는 사람 Pub: 클라이언트가 서버에게
    }

    1 usage
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry){
        registry.addEndpoint( ...paths: "/ws") Ws: 경로를 통해 웹소켓 서버에 연결
        .setAllowedOrigins("http://localhost:8080") http://localhost:8080: 접근 가능한 클라이언트 출처
        .withSockJS(); withSockJS: 웹소켓을 지원하지 않는 브라우저에서도 유사한 기능
        사용 가능
    }

    // STOMP에서 64KB 이상의 데이터 전송을 못하는 문제 해결
    1 usage
    @Override
    public void configureWebSocketTransport(WebSocketTransportRegistration registry) {
        registry.setMessageSizeLimit(160 * 64 * 1024);
        registry.setSendTimeLimit(100 * 10000); 메시지 최대 크기 지정
        registry.setSendBufferSizeLimit(3 * 512 * 1024);
    }
}
```

그림 22 WebSocketConfig

STOMP를 이용하여 설정을 해주는 코드이다. 웹소켓을 연결 할 때 /ws 경로를 통해 연결을 한다. 연결 후에 queue, sub, pub를 이용해서 메시지를 전송한다. queue는 개인적인 메시지를 보낼 때 사용하고 sub는 구독자에게 메시지를 보낼 때 사용한다.

### (3) 채팅과 권한

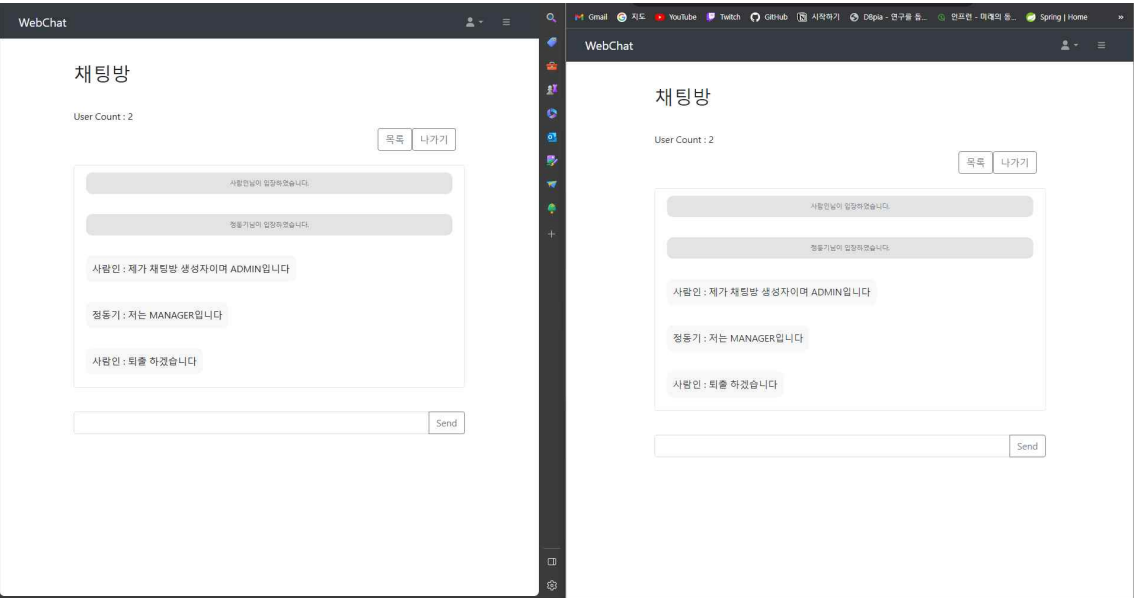


그림 23 채팅방

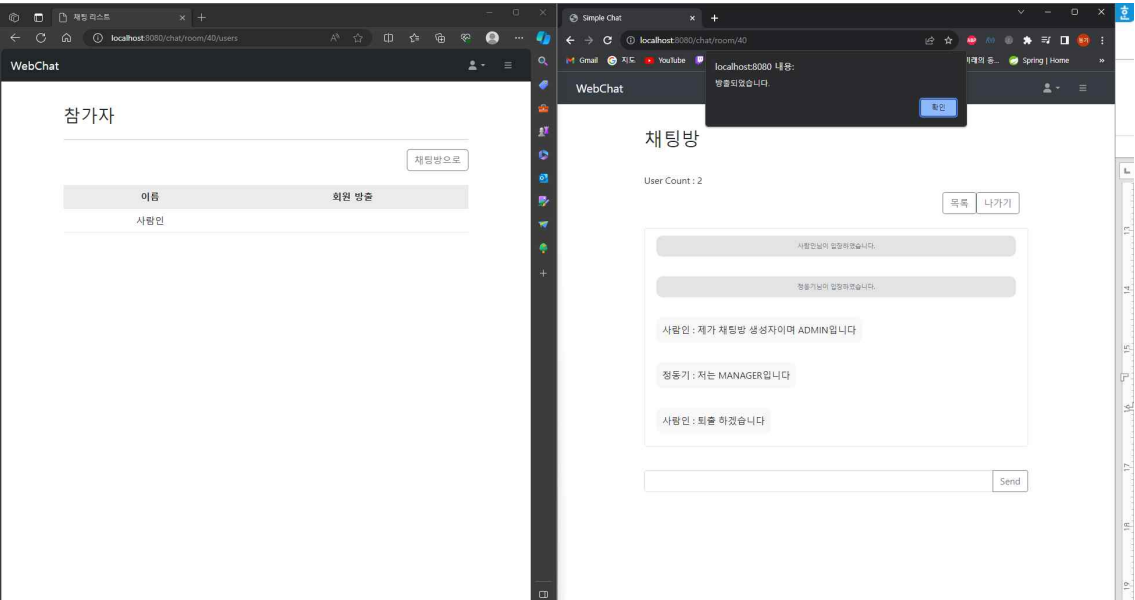


그림 24 채팅방 추방

위의 사진은 채팅방 내부의 사진이다. GUEST는 채팅 기능을 사용하지 못한다.

채팅은 공개 채팅과 비공개 채팅으로 나뉜다. 공개 채팅은 채팅을 사용하는 모든 사용자가 들어올 수 있는 채팅이다. 비공개 채팅은 채팅 생성자가 방을 생성하기 전에 초대할 사용자를 지정할 수 있다. 비공개 채팅방에 초대되지 않은 사용자는 들어올 수 없다.

채팅에서 권한에 따른 기능으로는 방출 기능이 있다. 방출 기능은 ADMIN과 MANAGER, 채팅방 생성자가 방출이 가능하게 해주었다. 이때 채팅방에서 방출되게 되면 개인 메시지(/queue)를 보내서 방출된 사용자에게 알림을 주도록 해주었다. 공개 채팅방에서 퇴출이 되면 다시 방에 들어올 수 있지만, 개인 채팅방에서 퇴출되게 되면 다시 들어올 수 없고 채팅방 리스트에서도 확인할 수 없게 된다.

## 1. 공개 채팅

### 공개 채팅 목록



그림 25 채팅 목록 - 공개

위의 사진은 공개 채팅 목록으로 채팅을 사용할 수 있는 모든 사용자가 생성할 수 있고, 참여할 수 있다. 채팅에 참여하게 되면 어떤 사용자가 들어가고 나갔는지 알려준다.

## 2. 비공개 채팅

### 개인 채팅 목록



그림 26 채팅 목록 - 비공개

## 프라이빗 채팅 생성

### User

☒ 정동기

☒ 예잉

☐ as

Private Room name:

비공개 채팅입니다.

방 생성

사진과 같이 비공개 채팅은 자신이 속한 채팅방만 표시가 되고, 방을 생성할 때 먼저 초대할 사용자를 선택하고 방을 생성하게 된다. 아무도 초대하지 않게 되면 자신만 들어와 있는 채팅이 생성된다.



### III. 개발 결과

결과적으로 앞서 작품을 구현함으로써 JWT 토큰을 이용하여 인증이 가능하게 되었고, Rotating Refresh Token 방식이 잘 적용되는 모습을 볼 수 있었다.



그림 28 refreshToken만 있는 경우



그림 29 refreshToken을 재발급 받고 accessToken 발급

사진에서 볼 수 있듯이 refreshToken만 있는 경우에 새로고침을 하게 되면 refreshToken과 accessToken을 발급받는다. refreshToken을 자세히 보면 유효기간이 늘어난 것을 볼 수 있다.

사용자의 권한에 따라 접근이나 관리를 할 수 있게 해주어서 ADMIN이나 MANAGER 권한이 있는 사용자들이 쉽게 사용자들을 관리할 수 있게 되었다.

프로젝트의 주된 목적이었던 JWT를 좀 더 안전하게 사용하고 권한을 통해 프로그램 사용자들을 관리함으로써 토큰이 탈취되어 모르는 사용자가 인증을 하더라도 계속해서 인증을 못하도록 하고 사용자가 프로젝트 내부의 파일이나 사진, 중요한 메시지들을 외부로 가지고 나가지 못하도록 권한이 높은 사용자가 사용자들을 관리하여 프로그램의 보안을 강화하는 목적을 잘 이행하였다고 생각한다.

## IV. 결론

### 1. 현재 프로젝트 성과

프로젝트를 진행하면서 현재까지 JWT를 이용한 인증과 accessToken과 refreshToken가 명확하게 구분되어 accessToken은 인증에 refreshToken은 accessToken 재발급에 사용되고 Rotating Refresh Token 방식을 잘 수행하고 있다.

사용자의 권한에 따른 기능 접근이 잘 수행되고 있다. 또한 ADMIN과 MANAGER 권한을 가지고 있는 사용자가 자신과 동등하거나 그 이하의 권한을 가진 사용자들을 관리할 수 있고 ADMIN과 MANAGER도 권한에 따라 사용자 관리 기능을 사용할 수 있는 범위가 달라지도록 구현하였다.

### 2. 아쉬운 점

1. 프로젝트를 진행하면서 Rotating Refresh Token 방식에서 좀 더 나아가 토큰 방식의 보안 문제점을 보완하는 방법을 스스로 구현하지 못하였다.
2. 기본적으로 내가 구현한 권한에 따른 기능 제한에서 더 나아가 ADMIN이나 MANAGER가 권한에 따른 기능 제한을 하지 못하고 있다.
3. 프로젝트를 배포를 하지 못하여 서버 확장 시에 JWT의 장점을 실제로 느껴보지 못하였다.
4. 배포를 하지 못하여 화면 공유나 화상통화를 구현하지 못하였다. 이는 채팅 관련 기능이 부실해지는 원인이 되기도 하였다.
5. JWT를 헤더에 넣는 방식으로 처음에 구현하였으나 로컬에서 돌리고 있기 때문에 Cookie에서 받아오는 방식으로 변경해서 구현하였다. 이를 다시 헤더에 넣는 방식으로 변경해주지 못하였다.

### 3. 향후 개선 방안

1. 프로젝트를 좀 더 진행하게 된다면 토큰 방식을 보완하는 기존의 방식보다 좀 더 보안적으로 우수한 방식을 구현하고자 한다.
2. ADMIN과 MANAGER가 주도적으로 권한에 따른 기능 범위를 변경할 수 있도록 해주고자 한다.
3. 프로젝트를 배포하며 채팅의 기능을 늘리고자 한다.
4. 토큰을 HTTP Header에 넣는 방식으로 다시 바꾸어준다.

## 참고 자료

### 1. JWT

[https://www.youtube.com/watch?v=GEv\\_hw0VOxE&list=PL93mKxaRDIdERCyMaoBSLkvSPzYtIk0Ah](https://www.youtube.com/watch?v=GEv_hw0VOxE&list=PL93mKxaRDIdERCyMaoBSLkvSPzYtIk0Ah)

: 메타코딩 - Springboot-시큐리티 특강

<https://ttl-blog.tistory.com/272> , <https://ttl-blog.tistory.com/273>

: JWT 로그인

<https://blogeon.tistory.com/entry/JWT%EC%9D%98-Refresh-Token%EA%B3%BC-Access-Token%EC%9D%80-%EC%96%B4%EB%94%94%EC%97%90-%EC%A0%80%EC%9E%A5%ED%95%B4%EC%95%BC-%ED%95%A0%EA%B9%8C>

: JWT의 RefreshToken과 AccessToken은 어디에 저장해야 할까?

### 2. 게시판, 댓글

[https://www.youtube.com/watch?v=YshcPPHCIR4&list=PLV9zd3otBRt7jmXvwCkmvJ8dH5tR\\_20c0](https://www.youtube.com/watch?v=YshcPPHCIR4&list=PLV9zd3otBRt7jmXvwCkmvJ8dH5tR_20c0)

: 코딩레시피 - 스프링 부트 게시판 프로젝트

<https://ttl-blog.tistory.com/372>

: 게시글 조회& 검색 서비스 구현

<https://velog.io/@juhyeon1114/Spring-QueryDsl-gradle-%EC%84%A4%EC%A0%95-Spring-boot-3.0-%EC%9D%B4%EC%83%81>

:QueryDsl gradle 설정(Spring boot 3.0 이상)

<https://ttl-blog.tistory.com/278> , <https://ttl-blog.tistory.com/334> , <https://ttl-blog.tistory.com/383>

: 댓글 및 대댓글 구현

### 3. 채팅

<https://terianp.tistory.com/146>

: Spring Boot Web Chatting: 스프링 부트로 실시간 채팅 만들기

### 4. 메일

<https://dev-aiden.com/spring/Spring-%EC%9D%B4%EB%A9%94%EC%9D%BC-%EB%B0%9C%EC%86%A1/>

: [Spring] SMTP 서버를 이용한 이메일 발송