

5월 5주차 회의록

팀	MAC
참석자	한결 민 상연 안 최필환
장소	원흥관 3층 아리수
시간	23.06.04 13:00~15:00

팀 주제

동국대학교 E-class 내 파일 스토리지 기능 구현 → 협업 능력 향상

주요 안건

- 진행도 파악
- 진행 사항 파악
- 보고서 작성 계획 수립

1. 진행도 파악

분류	내용	담당	4월4주	5주	5월1주	2주	3주	4주	5월5주	6월1주	6월2주
웰컴 페이지	로그인 기능 구현	민한결									
	프론트 템플릿 수정	안상연									
파일 스토리지	AWS 설정/파일 업로드 구현	최필환									
팀 활동 페이지	팀 구성 기능 구현	민한결									
웰컴 페이지	로그인 요청 및 응답 처리	안상연									
파일 스토리지	파일 관련 기능 구현	최필환									
팀 활동 페이지	개별팀 조회 및 초대장 요청 및 응답 처리	민한결									
	팀 목록 조회 요청 및 응답 처리	안상연									
	공지사항 기능 구현, 파일 페이지 처리	최필환									
	공지사항 요청 및 응답 처리	민한결									
	리프레시 토큰 구현, DB 암호화	민한결									
	파일 스토리지 요청/응답	안상연									
	통합 테스트, 배포	최필환									
오류 해결, 보고서, 발표자료 작성		모 두									
최종 수정 및 발표		모 두									

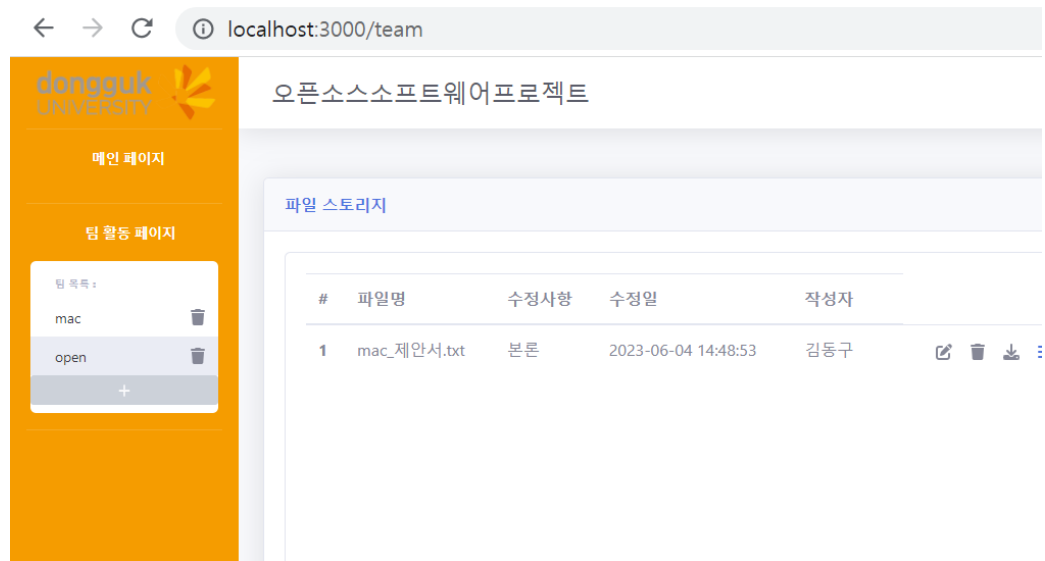
2. 진행 사항 파악

1. 프론트엔드

a. 안상연

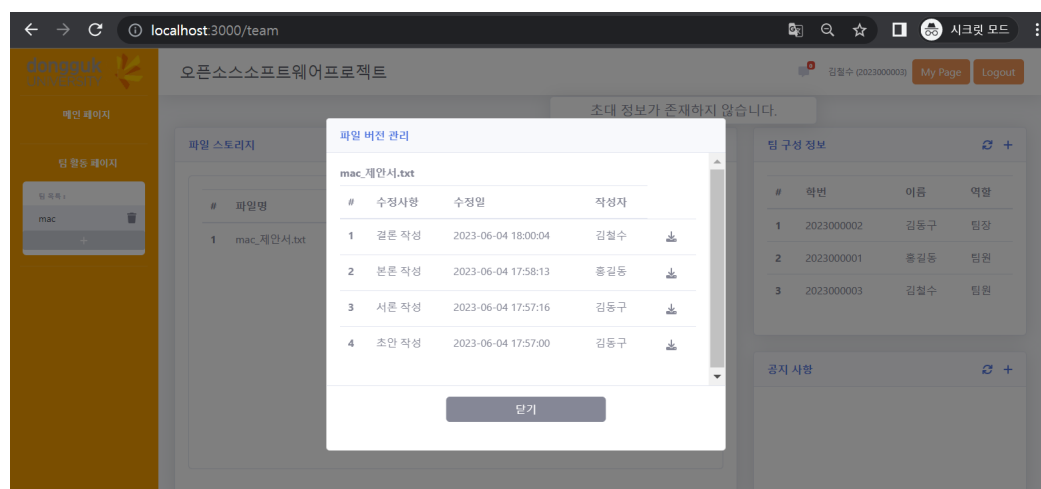
- UI 디자인 개선

현재 어떤 팀의 정보를 보고 있는지 알 수 있도록 팀 버튼 클릭 시 강조 처리



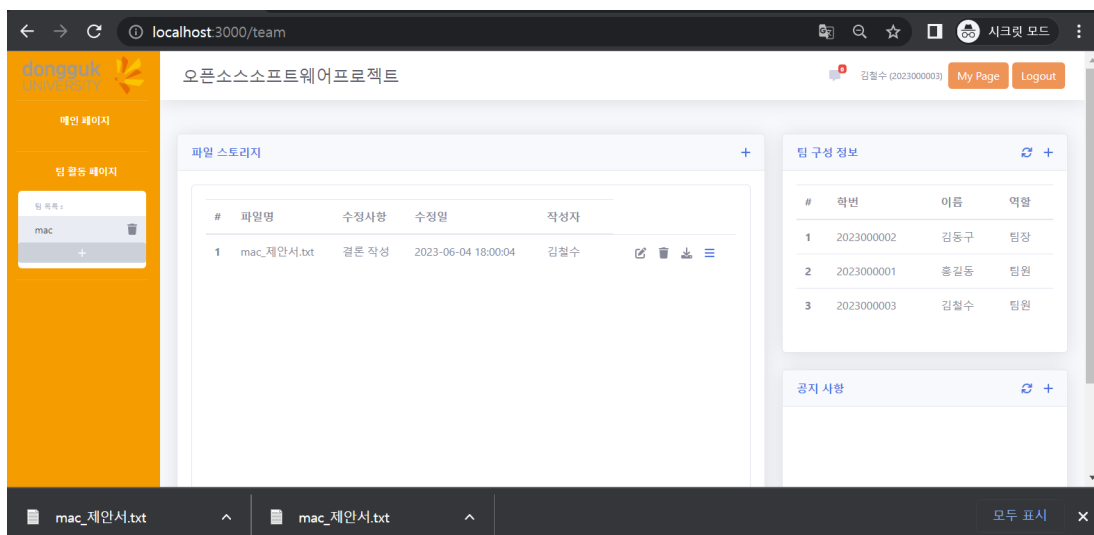
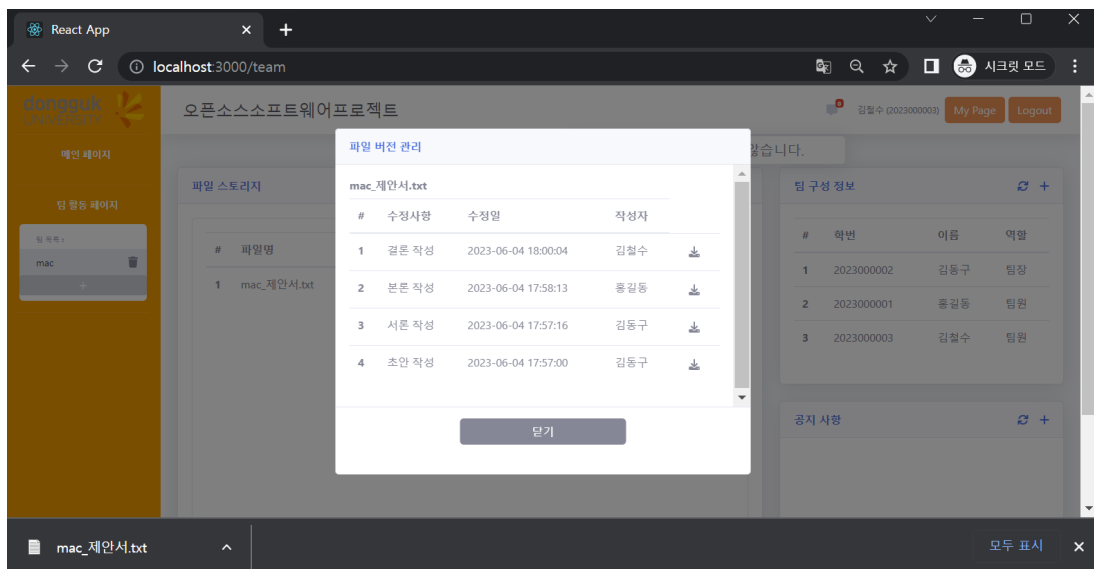
파일 이력 조회 모달 디자인 개선

- 이전 파일 내역이 많아지면 스크롤 처리
- 해당 버전의 제목(파일명)을 상단에 표시

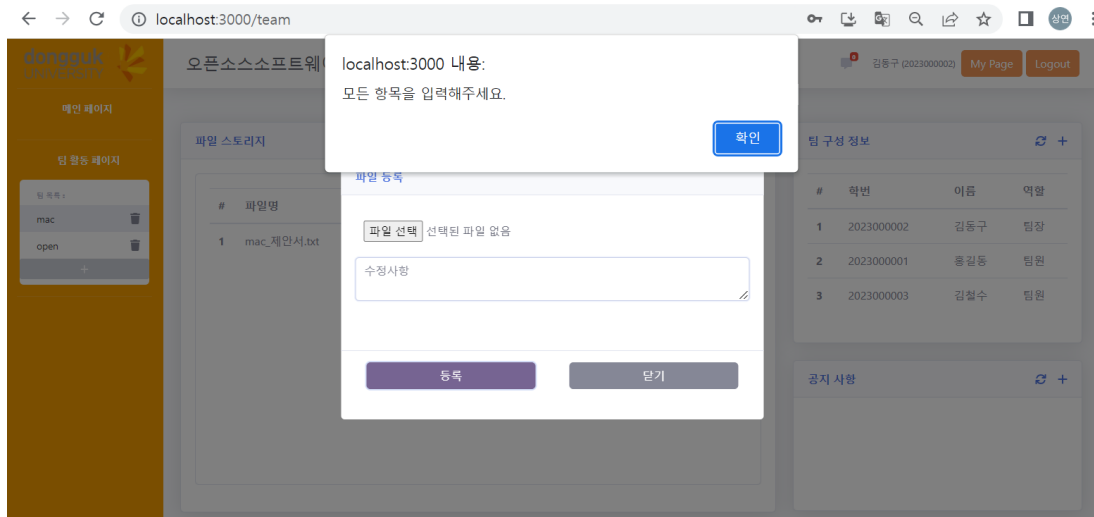


- 파일 다운로드 기능 구현

파일 스토리지와 이력 조회 모달 모두 다운로드 버튼을 클릭했을 때 다음과 같이 사용자 컴퓨터에 저장되도록 구현.



- 에러 처리
 - 모든 에러 발생에 대하여 alert창으로 에러 발생 이유를 표시.



b. 민한결

- 에러 처리 HTTP 코드별 세분화

```
const fetchFileList = () => {
  const accessToken = sessionStorage.getItem("accessToken");
  if (accessToken === null) {
    alert("로그인이 필요합니다.");
    navigate("/");
  }
  fetch(`http://localhost:8080/team/${teamId.id}/file/${fileId}`)
    .then((response) => {
      headers: {
        Authorization: `Bearer ${accessToken}`
      }
    })
    .then((response) => {
      if (response.status === 200) {
        return response.json();
      }
      if (response.status === 400) {
        return response.json().then((jsonData) => {
          throw new Error(showErrorMessage(jsonData));
        });
      }
      if (response.status === 401) {
        handleRefreshToken().then((result) => {
          if (result) {
            fetchFileList();
          } else {
            navigate("/");
          }
        });
      }
    });
};
```

400 코드면 에러 발생 이유를 body에서 뽑아 출력, 401이면 토큰 갱신 요청

- 로그인, 로그아웃 보완

```

})
.then((data) => {
  console.log(data);
  sessionStorage.setItem("accessToken", data.accessToken);
  localStorage.setItem("refreshToken", data.refreshToken);
  goSelect();
})
catch((error) => {

```

로그인 성공시 access 토큰은 세션 스토리지 refresh 토큰은 로컬 스토리지에 저장

```

.then((response) => {
  if (response.status === 200) {
    sessionStorage.removeItem("accessToken");
    localStorage.removeItem("refreshToken");
    alert("로그아웃 되었습니다.");
    return;
  }

```

로그아웃시 redis에서 refresh 토큰 삭제하도록 서버에 요청하고 브라우저에서 모든 토큰 삭제

- Refresh Token 통해 Access Token 재발급 처리

이름	x	헤더	페이로드	미리보기	응답	시작점
team	1		{ "accessToken": "eyJhbGciOiJIUzI1NiJ9.			
bundle.js						
dongguk_logo.png						
invitation						
user						
manifest.json						
refresh-token						
user						

api 요청 응답이 401일 때 refresh token 요청하여 재발급

2. 백엔드

a. 민한결

- DB 암호화

```

@Override
public String convertToDatabaseColumn(String attribute) {
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(SECRET_KEY.getBytes(), ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);
        byte[] encryptedBytes = cipher.doFinal(attribute.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

2 usages  Hankyul Min
@Override
public String convertToEntityAttribute(String dbData) {
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(SECRET_KEY.getBytes(), ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(dbData));
        return new String(decryptedBytes, StandardCharsets.UTF_8);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

```

DB 접근 암호화 로직

dept	name
9Zi8kOHVuH6DRsyTGy5t2FHjvKzxvcabRJKycKr...	SagQ5je35DRWwdpIeWI4Lg==
NULL	NULL

AES로 암호화되어 DB 저장된 모습

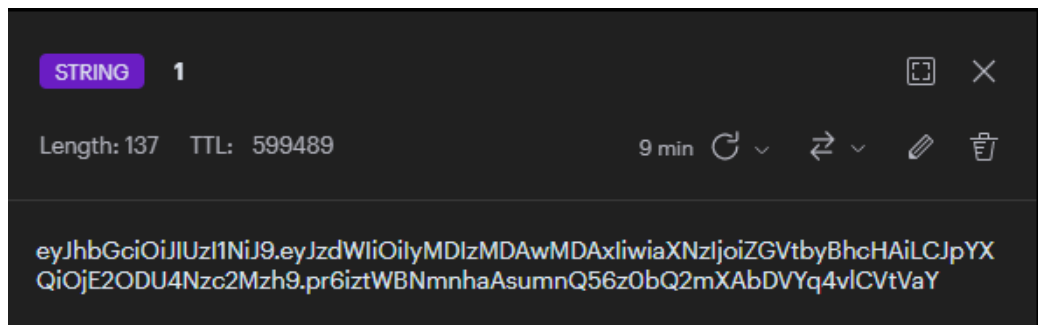
- Refresh Token 구현

```

public Map<String, String> create(User user) {
    Date now = new Date(); // 현재 시간
    Date accessTokenValidity = new Date(now.getTime() + 1000 * 60 * 5); // 5분 뒤 만료
    Date refreshTokenValidity = new Date(now.getTime() + 1000 * 60 * 60 * 24 * 7); // 7일 뒤 만료
    String refreshToken = Jwts.builder()
        .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
        .setSubject(user.getStudentId())
        .setIssuer("demo app")
        .setIssuedAt(now)
        .compact();
    ValueOperations<String, String> vop = redisTemplate.opsForValue();
    vop.set(user.getId().toString(), refreshToken, (timeout: refreshTokenValidity.getTime() - now.getTime(), TimeUnit.MILLISECONDS);
}

```

refresh 토큰 redis에서 저장하는 로직



로그인시 redis에 저장된 refresh 토큰

```

@PostMapping("/refresh-token")
public ResponseEntity<?> refreshToken(@RequestBody UserDto userDto) {
    return userAuthService.refresh(userDto.getRefreshToken());
}

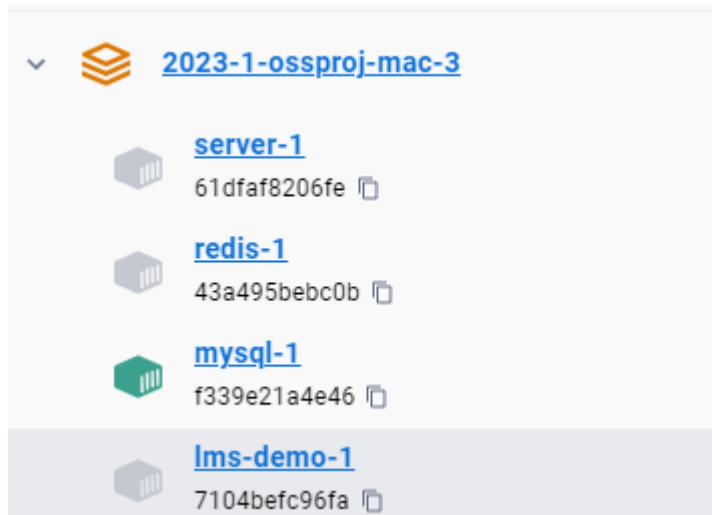
```

access 토큰 만료시 refresh 토큰을 바디에 받아 두 토큰을 재발급

3. 배포

a. 민한결

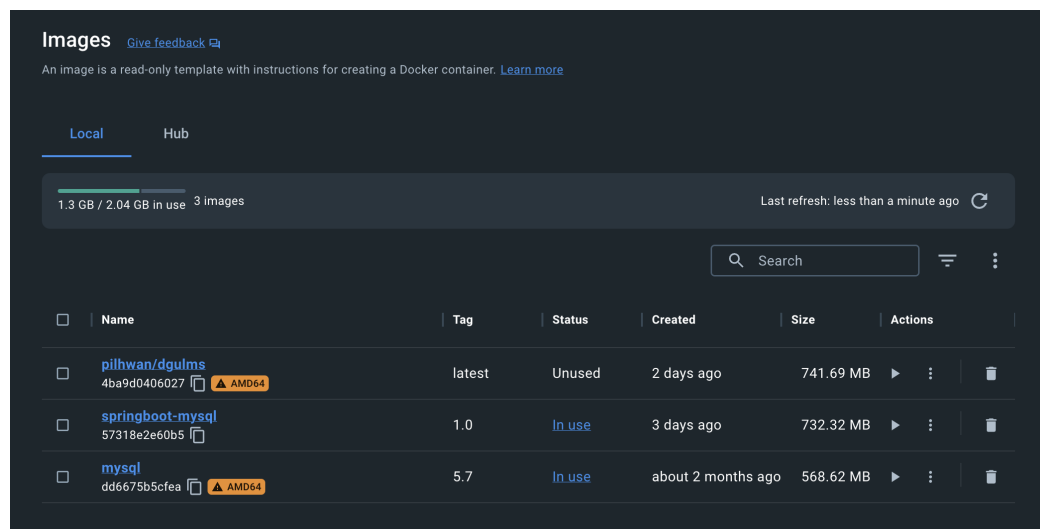
- 프론트 디렉토리 구조 정리, 도커파일 작성하여 도커컴포즈로 로컬에서 테스트



b. 최필환

- 로컬 테스트

프론트에 자잘한 수정사항이 남았기 때문에 백엔드 먼저 테스트 배포




위와 같이 mysql:5.7 version, docker hub에 올릴 스프링 서버인 pilhwan/dgulms 이미지를 생성했다.

위의 이미지를 도커 허브에 push 했다.

pilhwan / dgulms





Description

This repository does not have a description 

 Last pushed: 2 days ago

Tags

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
 latest		Image	2 days ago	2 days ago
 2.0		Image	2 days ago	2 days ago

[See all](#)

[Go to Advanced Image Management](#)

- ec2 상에서 pull하고, 컨테이너 실행

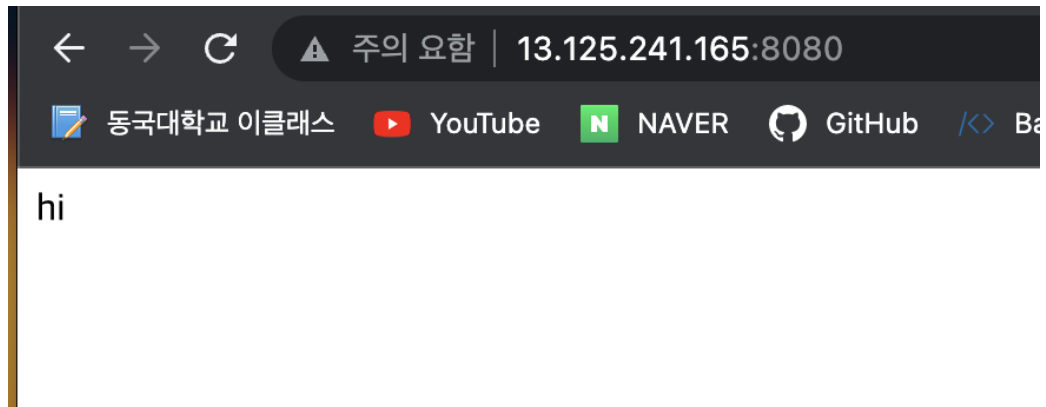
```
ubuntu@ip-172-31-40-93:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
b8f324dff703   pilhwan/dgulms  "java -jar /app.jar"    3 hours ago   Up 56 minutes  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp  springboot-mysql
5924143bc95b   mysql:5.7      "docker-entrypoint.s..." 7 hours ago   Up About an hour  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp  db-mysql
```

- 퍼블릭 접속을 위한 인바운드 규칙 추가

인바운드 규칙 (4)									
<input type="text" value="보안 그룹 규칙 필터"/>									
<input type="checkbox"/>	Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스		
<input type="checkbox"/>	-	sgr-018f612c16acd2e35	IPv4	HTTP	TCP	80	0.0.0.0/0		
<input type="checkbox"/>	-	sgr-02bc18b1e05dadb35	IPv4	SSH	TCP	22	0.0.0.0/0		
<input type="checkbox"/>	-	sgr-01cc8ef0c7716e2d	IPv4	사용자 지정 TCP	TCP	8080	0.0.0.0/0		
<input type="checkbox"/>	-	sgr-0fa6c9d7a8a4d634a	IPv4	사용자 지정 TCP	TCP	3000	0.0.0.0/0		

- 모든 외부 ip에 대해 접속이 가능하도록 인바운드 규칙을 편집했다.

- 퍼블릭 접속(http://{ec2 ipv4}:8080)



- 테스트로 만든 컨트롤러가 반환되는 것을 확인 할 수 있다.

3. 향후 진행 사항

- a. 배포
- b. 보고서 작성
- c. 발표자료 작성