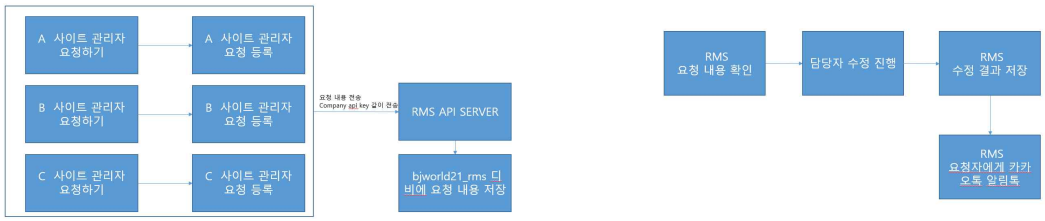
* BJWORLD_RMS API

> 목차

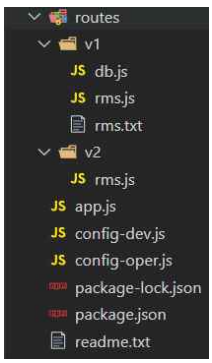
> 1. 프로젝트 개요

참여 인원	인원 : 2명	기간	약 30일	비고	인턴 근무
1. 개요	기존 사내 통합 관리시스템(CMS)에는 외부에서의 요청을 등록하고 관리할 수 있는 API가 존재하지 않아 요청은 메신저로 수신 받고 해당 데이터는 액셀 파일 데이터로 별도 관리하고 있었는데 이러한 불편함을 개선하고자 요청 관리 시스템 서버(RMS)를 개발하여 요청 데이터를 별도의 DB와 서버에서 관리				
2. 개발 언어	CMS - 주요언어: JAVA RMS - javascript, node.js, express, mysql				

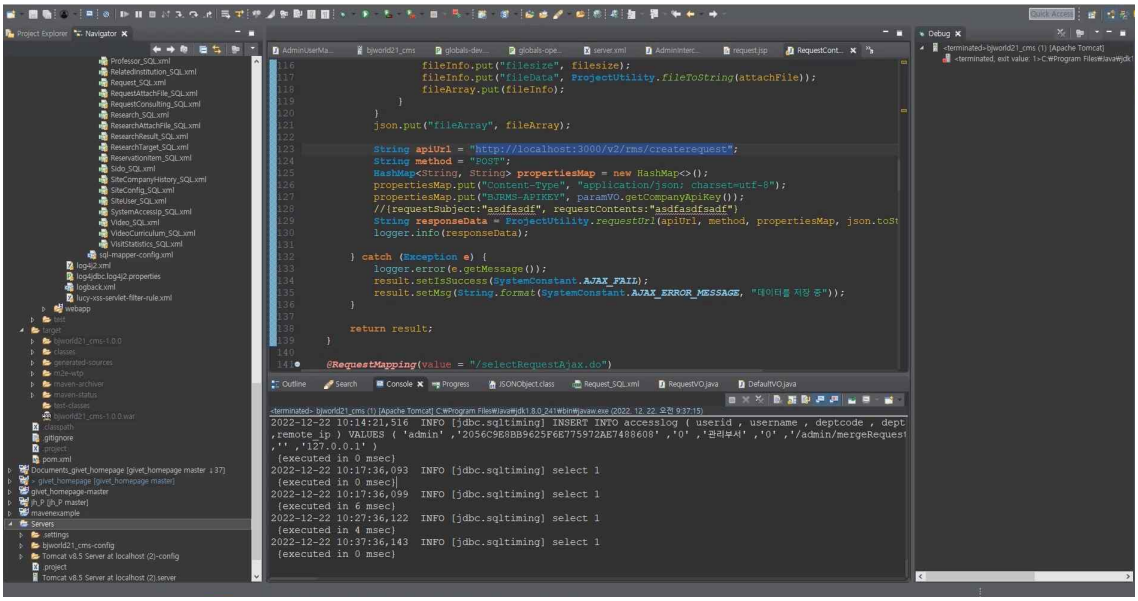
> 2. 프로젝트 기본 설정

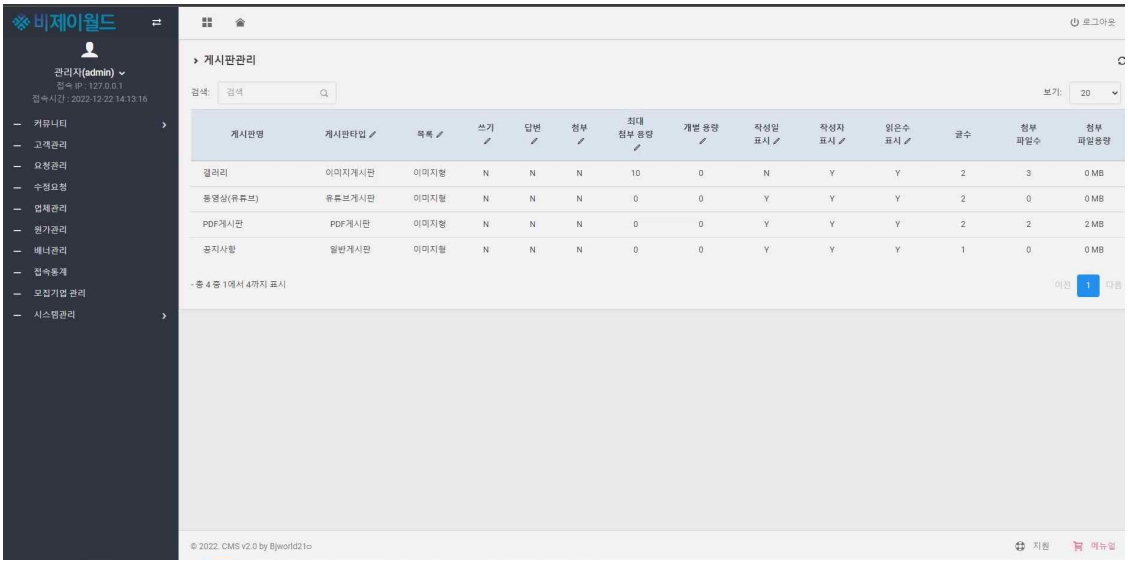
1. 개발 환경	eclipse, sourcetree, putty, MySQL, VSC
2. 기본 설정	- bjworld_cms -> id: admin, pw: admin2!#% - 공유폴더 -> id: bjworld21, pw: bjworld21! - apikey -> lkjsa2gafgag1 - http://ip.bjworld21.com/ -> 개인 PC ip 확인 - everything -> pc 디렉토리 출력 프로그램 - evernote -> 문서 저장 클라우드 서버
3. 톰캣 기본 설정	- Dspring.profiles.active=dev -Djasypt.encryptor.password=ngk2*d2d0ad -Djava.net.preferIPv4Stack=true
4. DB	- db.url = www.bjworld.com:3306 - db.name = bjworld21_rms - db.id = bjworld21_rms - db.pwd = a2jdf@gj9#3d - config.ipcheck = 1 -> 접속 시, ip 인증여부 true
5. 서버 구조 설계	

> 3. 프로젝트 구조

1. 프로젝트 구조	 * v1 1. rms 서버 api를 위한 최초 코드: v1 2. v1 rms.js는 db.js와 연동되어 있음 * v2 -- 완성된 코드 1. v2는 rms.js는 db 파일을 별도로 두지 않음 * config 1. argu 값에 따른 ip, 포트, 파일저장 베이스 폴더의 구분 -> 관리자-사용자 등의 구분을 위하여

2. CMS

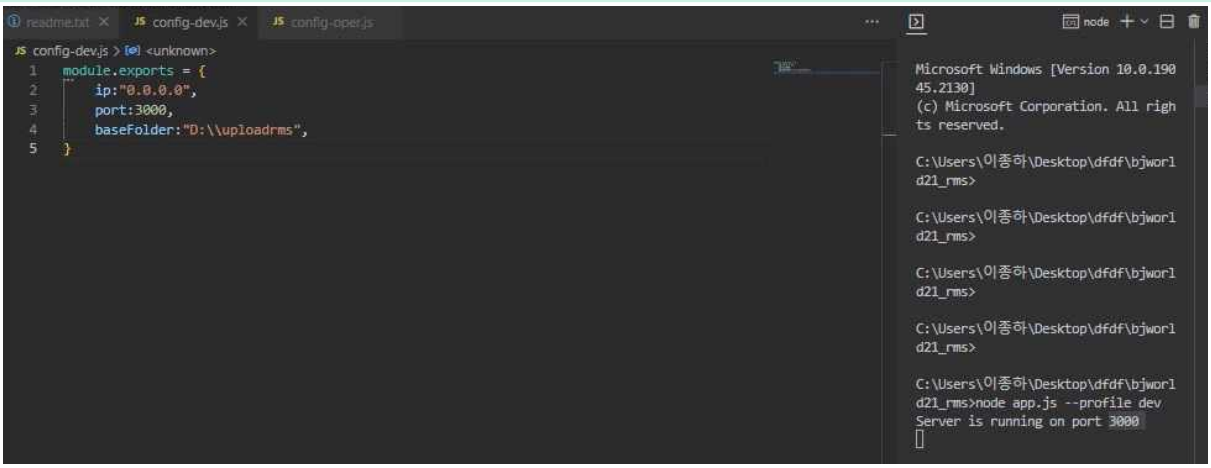




1. 기존의 사내 통합 관리시스템은 JAVA를 메인으로 구성
2. 메인 화면에 보이는 요청 관리 배너 메뉴는 CMS에서 JAVA로 구현
3. RMS와 CMS, 두 서버 간의 연동은 CMS의 requestController.java가 담당

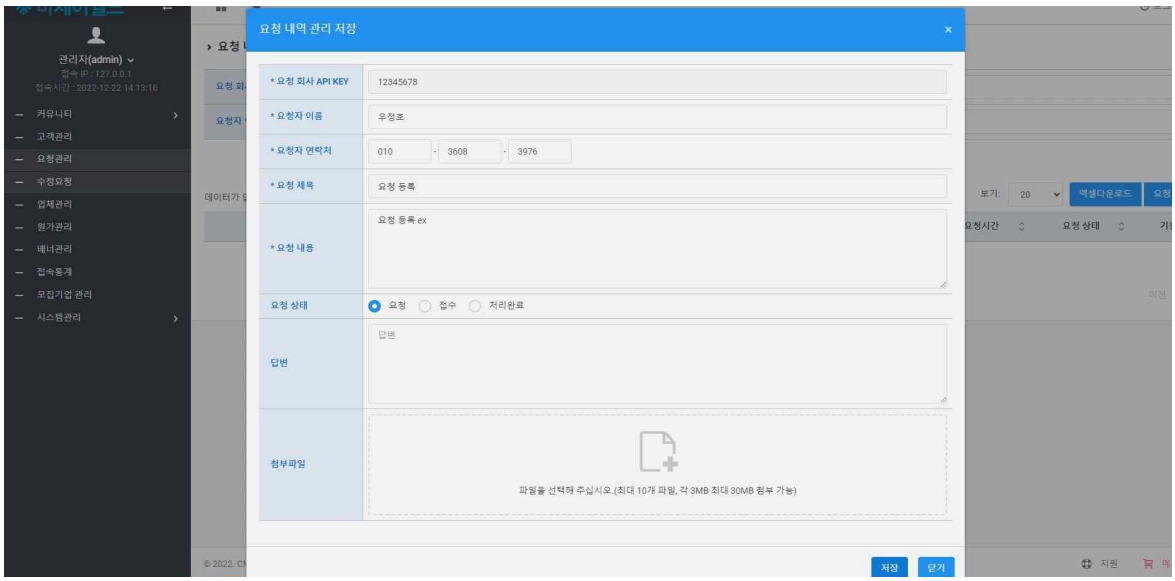
> 4. 프로젝트 기능

1. config - 서버 가동

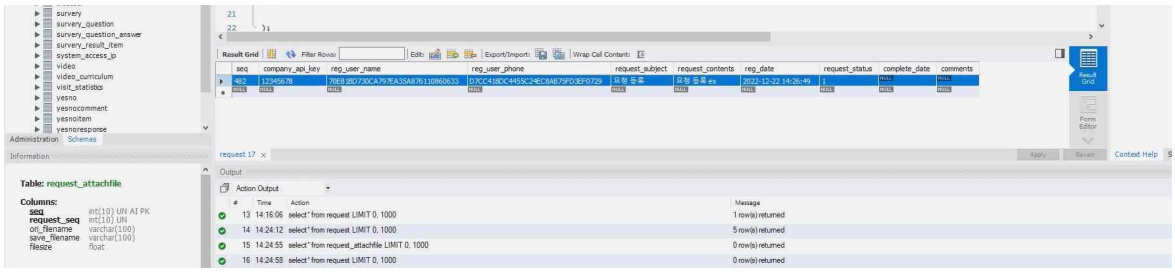


프로젝트 가동 시 node app.js --profile dev 혹은 oper 로 서버 시작, argu 값에 따른 ip, 포트, 파일저장 베이스 폴더의 구분 -> 관리자-사용자 등의 구분을 위하여

2. 요청등록

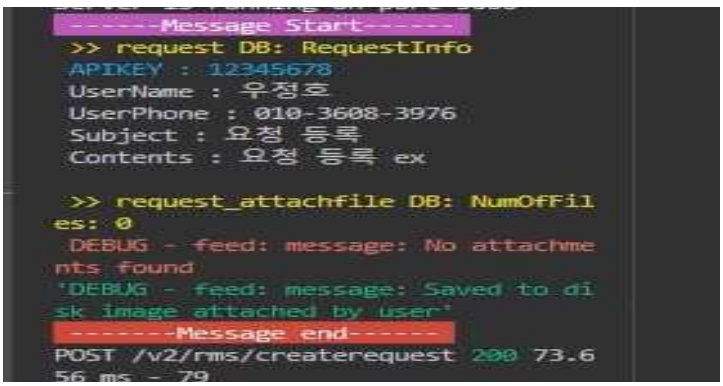


요청 등록 폼은 CMS 서버에서 JAVA로 구현

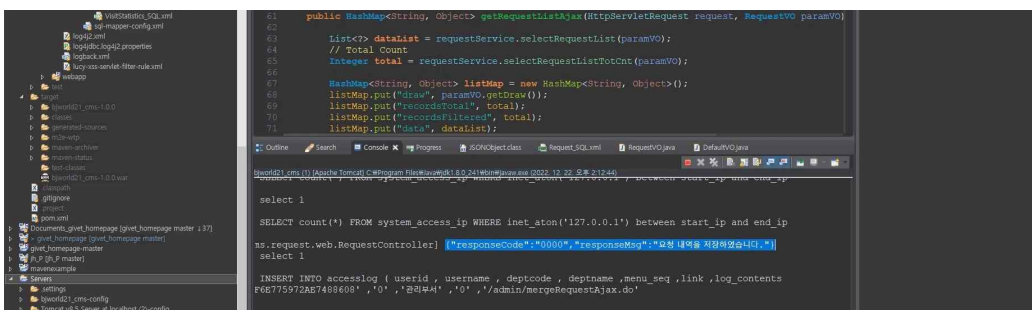


요청 등록 시, DB에 업데이트

메인 화면에 DB 정보 출력



등록 정보 출력

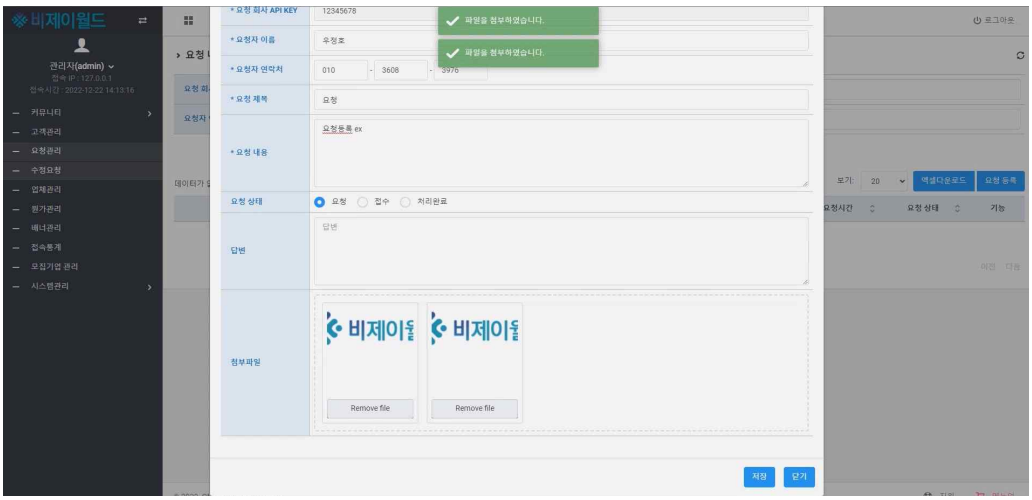


CMS에 정상 응답 출력

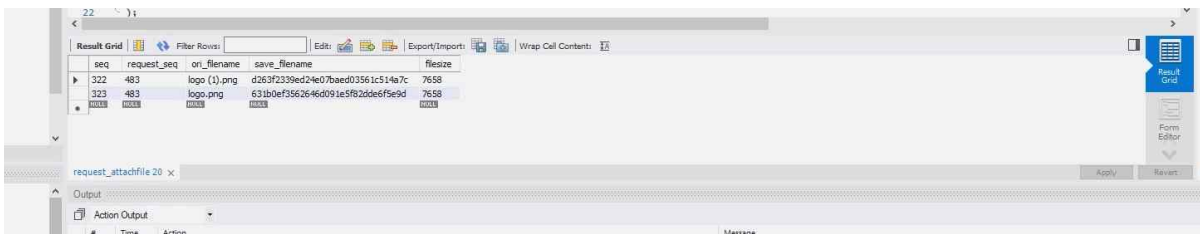
3. 파일 업로드

요청의 수정, 복사, 삭제 기능

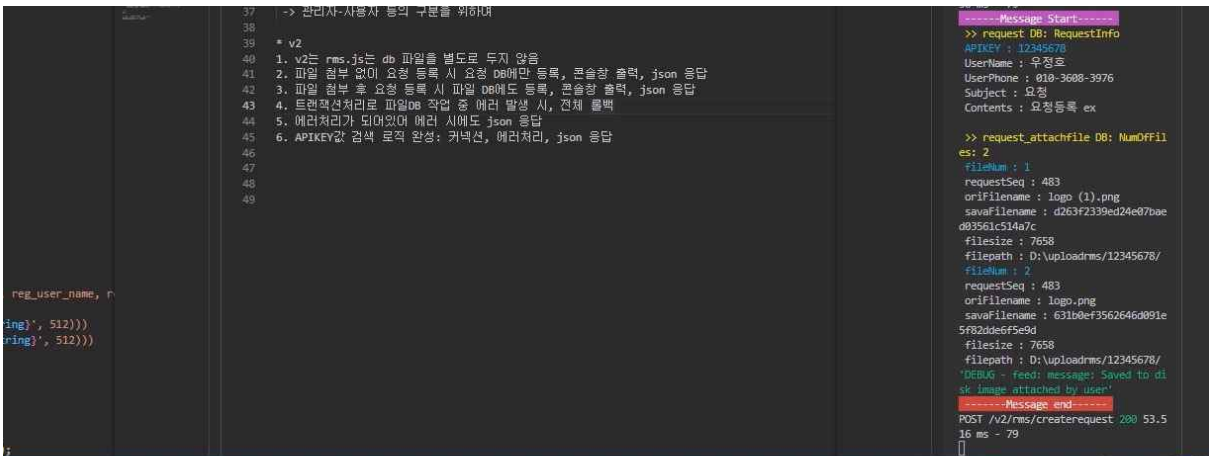
등록 정보 액셀 파일 다운로드



단일 및 다중 파일 업로드



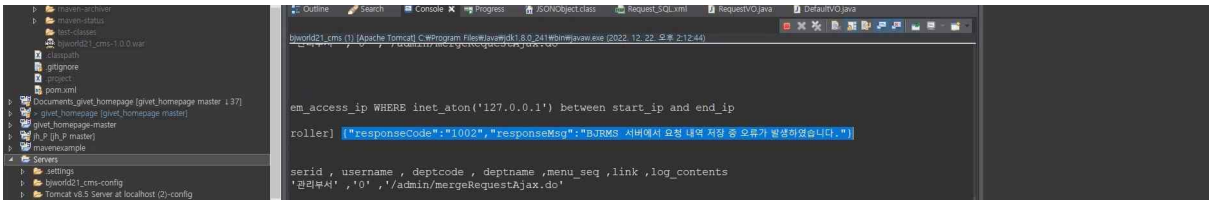
업로드 파일 정보의 DB 업데이트



업로드 파일 정보의 출력

업로드 파일의 다운로드 로직

4. 롤백 수행



요청 등록 정보 혹은 파일 정보 업데이트 과정 중에서 오류 발생 시, 전체 과정이 롤백되는 트랜잭션이 적용되어 있으며 에러 발생하는 경우에 CMS에 에러 응답을 송출

> 5. 전체 코드_v2

1. config-dev.js

```
module.exports= {
    ip:"0.0.0.0",
    port:3000,
    baseFolder:"D:\\uploadrms",
}
```

2. config-oper.js

```
module.exports= {
    ip:"0.0.0.0",
    port:9123,
    baseFolder:"/home/bjworld21_rms/upload",
}
```

3. rms.js

```
// ------- RMS API의 수신/응답, DB 커넥션 담당하며 컨트롤러 역할을 하는 rms.js
/* ------- 코드 구조
0. 변수 및 라이브러리 선언
1. MySql 선언 및 bjworld21_rms DB 연동
2. 요청 내역 및 첨부파일 DB 업데이트 로직
└ 2.0 DB 커넥션
└ 2.1 요청 내역의 DB 업데이트 로직
└ 2.2 요청 내역 콘솔창 출력
└ 2.3 첨부파일 유무에 따른 IF문 처리
└ 2.4 첨부파일 DB 업데이트 로직
└ 2.5 첨부파일 정보 콘솔창 출력
└ 2.6 에러 처리
└ 2.7 커넥션 종료
3. APIKEY 값을 통한 DB 검색 로직
└ 3.0 DB 커넥션
└ 3.1 검색 내역 콘솔창 출력
└ 3.2 검색 내역 DB 조회 로직
└ 3.3 결과값 유무에 따른 IF문 처리
```

```
└ 3.4 에러 처리
└ 3.5 커넥션 종료
99. 라우터 모듈 선언
*/
// 0. 변수 및 라이브러리 선언 =====
// express 라이브러리 선언
var expressrequire('express');
// 라우터 선언
var routerexpress.Router();
// 동기처리
var asyncrequire('async');
// 암복호화 로직
const encString"gi625@3vet1#";
// 파일 다운로드 처리를 위한 express 모듈
const fsrequire('fs');
// 트랜잭션 작업을 위한 mysql2 선언
const mysqlrequire('mysql2/promise');
// Universally Unique IDentifier: RFC4122에 명시된 네트워크 상 id를 위한 표준 규약, public한 화면단에서는
ramdom 한 UUID를 사용하는 것을 권장
const uuid4require('uuid4');
// 콘솔창 글씨 디자인 초기화
let resetFs"\x1b[0m";
// 1. MySql 선언 및 bjworld21_rms DB 연동 =====
const poolmysql.createPool({
  host:'www.bjworld21.com'
user:'bjworld21_rms'
password:'a2jdf@gj9#3d'
port:3306
database:'bjworld21_rms'
connectionLimit:10
});
// 2. 요청 내역 및 첨부파일 DB 업데이트 로직 =====
router.post('/createrequest', async function(req, res, next) {
  // 2.0 DB 커넥션
  const connawait pool.getConnection(); // 커넥션 시작
  try{
    await conn.beginTransaction() // 트랜잭션 적용 시작
    let responseData= {};
    let apiKeyreq.header('BJRMS-APIKEY');
    // 2.1 요청 내역의 DB 업데이트 로직
     const insawait conn.query(`INSERT INTO request (company_api_key, reg_user_name, reg_user_phone,
request_subject, request_contents) VALUES (
'${apiKey}'
, HEX(AES_ENCRYPT('${req.body.regUserName}', SHA2('${encString}', 512)))
, HEX(AES_ENCRYPT('${req.body.regUserPhone}', SHA2('${encString}', 512)))
, '${req.body.requestSubject}'
, '${req.body.requestContents}'
)`
    // 2.2 요청 내역 콘솔창 출력
    console.log('\u001b[45m', '------Message Start------', resetFs, "");
    console.log("\x1b[33m", '>> request DB: RequestInfo');
    console.log("\x1b[36m", "APIKEY : "req.header('BJRMS-APIKEY'));
    console.log(resetFs, "UserName : "req.body.regUserName);
    console.log(resetFs, "UserPhone : "req.body.regUserPhone);
    console.log(resetFs, "Subject : "req.body.requestSubject);
    console.log(resetFs, "Contents : "req.body.requestContents);
    console.log(resetFs, "");
    let requestSeqins[0].insertId; // 요청 내역 DB에 업데이트된 ID값 추출
    let filesreq.body.fileArray;
    console.log("\x1b[33m", ">> request_attachfile DB: NumOfFiles: "files.length, resetFs); // 첨부파일
개수 콘솔창 출력
    // 2.3 첨부파일 유무에 따른 IF문 처리
    if(files.length=== 0) { // 첨부파일이 존재하지 않는 경우
      console.log("\x1b[31m", "DEBUG - feed: message: No attachments found", resetFs);
      responseData= { responseCode:"0000", responseMsg:"요청 내역을 저장하였습니다."};
}
    // 2.4 첨부파일 DB 업데이트 로직
    let validateFileSavetrue;
    for(let i0; i < files.length; i++) {
      let uploadFilefiles[i];
      let saveFilenameuuid4().replace(/-/g, '');
      let extensionuploadFile.saveFilename.substring(uploadFile.saveFilename.lastIndexOf("."));
        await conn.query(`INSERT INTO request_attachfile (request_seq, ori_filename, save_filename,
filesize)
VALUES (
'${requestSeq}'
, '${uploadFile.oriFilename}'
, '${saveFilename}'
```

```
, '${uploadFile.filesize}'
)`);
      let uploadPathglobal.config.baseFolder"/"apiKey"/"; // 파일 업로드 경로 설정
      let saveFullPathuploadPathsaveFilenameextension;
      if(!fs.existsSync(uploadPath)) { // uploadPath 폴더가 존재하지 않는 경우
        fs.mkdirSync(uploadPath); // 폴더를 생성하여 파일저장
}
      try{
        fs.writeFileSync(saveFullPath, Buffer.from(uploadFile.fileData, "base64")) // 파일 암복호화 로직
        // 2.5 첨부파일 정보 콘솔창 출력
        console.log("\x1b[36m", "fileNum : "+ (i1));
        console.log(resetFs, "requestSeq : "requestSeq);
        console.log(resetFs, "oriFilename : "uploadFile.oriFilename);
        console.log(resetFs, "savaFilename : "saveFilename);
        console.log(resetFs, "filesize : "uploadFile.filesize);
        console.log(resetFs, "filepath : "uploadPath);;
} catch(err) { // 에러처리: 에러 발생 시 파일저장 작업 취소
        validateFileSavefalse;
}
}
    if(validateFileSave) { // 파일 정상 저장
      console.dir('DEBUG - feed: message: Saved to disk image attached by user');
      responseData= { responseCode:"0000", responseMsg:"요청 내역을 저장하였습니다."};
} else// 파일 정상 에러
      console.log("\x1b[31m", "Error occurred while saving operation.", resetFs);
      responseData= { responseCode:"1001", responseMsg:"파일 저장 중 오류 발생"};
      throw "error";
}
    await conn.commit() // 커밋
    return res.json(responseData); // CMS 서버에 json 응답
  // 2.6 에러 처리
catch(err) {
    await conn.rollback() // 롤백
    console.log('\u001b[45m', '----Error detected !----', resetFs, "");
     console.log("\x1b[31m", "Error occurred while saving the request history on the BJRMS server.",
resetFs);
    console.error(err);
    responseData= { responseCode:"1002", responseMsg:"BJRMS 서버에서 요청 내역 저장 중 오류가 발생하였습니
다."};
    return res.json(responseData); // CMS 서버에 에러 응답

  // 2.7 커넥션 종료
finally{
    conn.release() // conn 회수
    console.log('\u001b[41m', "-------Message end------", resetFs); // 콘솔창 메시지 종료
}
});


// * 3. APIKEY 값을 통한 DB 검색 로직 =====
router.get('/getrequestcompanylist', async function(req, res, next) {
  // 3.0 DB 커넥션
  const connawait pool.getConnection(); // 커넥션 시작
  try{
    await conn.beginTransaction() // 트랜잭션 적용 시작
    // 3.1 검색 내역 콘솔창 출력
    console.log('\u001b[45m', '------Search Start------', resetFs, "");
    console.log("APIKEY VALUE : "req.header('BJRMS-APIKEY'));
    // 3.2 검색 내역 DB 조회 로직
                const       searchawait       conn.query(`SELECT       *       FROM       request       WHERE
company_api_key='${req.header('BJRMS-APIKEY')}'`, (err, result) =>{
      if(err) throw err;
      callback(result);
});
    // 3.3 결과값 유무에 따른 IF문 처리
    if(search.result.length) { // APIKEY값의 결과가 존재하는 경우
      console.log("\x1b[31m", "DEBUG - feed: message: Search success", resetFs);
      console.log(resetFs, "searchResult : "search.result, resetFs);
      responseData= { responseCode:"0001", responseMsg:"검색에 성공하였습니다.", datas:search.result};
} else// APIKEY값의 결과가 존재하지 않는 경우
      console.log("\x1b[31m", "DEBUG - feed: Result not found", resetFs);
      responseData= { responseCode:"0002", responseMsg:"입력값에 해당하는 결과가 없습니다."};
}
    await conn.commit() // 커밋
    return res.json(responseData); // CMS 서버에 json 응답
  // 3.4 에러 처리
catch(err) {
    await conn.rollback() // 롤백
```

```
    console.log('\u001b[45m', '----Error detected !----', resetFs, "");
    console.log("\x1b[31m", "rror occurred while retrieving from the BJRMS server.", resetFs);
    console.error(err);
     responseData= { responseCode:"1003", responseMsg:"BJRMS 서버에서 검색 작업 중 오류가 발생하였습니
다."};
    return res.json(responseData); // CMS 서버에 에러 응답
  // 3.5 커넥션 종료
finally{
    conn.release() // conn 회수
    console.log('\u001b[41m', "-------Search end-------");
}
});
// 99. 라우터 모듈 선언 =====
module.exportsrouter;
```