>> rest API 서버 [요청관리 시스템]_example

REST API : REST(Representational State Transfer)는 HTTP URI를 통해 자원을 명시하고 HTTP Method(POST, GET, PUT, DELETE, PATCH)를 통해 해당 자원에 대한 CRUD(Create, Read, Update, Delete) 동작을 적용하는 것을 의미한다. REST API는 REST의 원리를 따르는 API를 의미한다.

프로젝트 개요

- 사용언어

node.JS(express), MySQL, HTML(EJS), CSS

- 요청 배경

현재 각 사이트에서 수정요청 사항이 발생할 경우 해당 사이트 담당자는 카톡이나 전화로 개발팀으로 요청을 하고 있고 처리가 완료될 경우 다시 해당 사이트 담당자에게 연락해 처리 여부를 알려 주고 있어 업무 진행에 불편함이 존재하여 각 사이트 관리자메뉴에 요청사항 메뉴를 신설해 요청 내역이 관리되도록 구현

- 프로그램 요청 사항

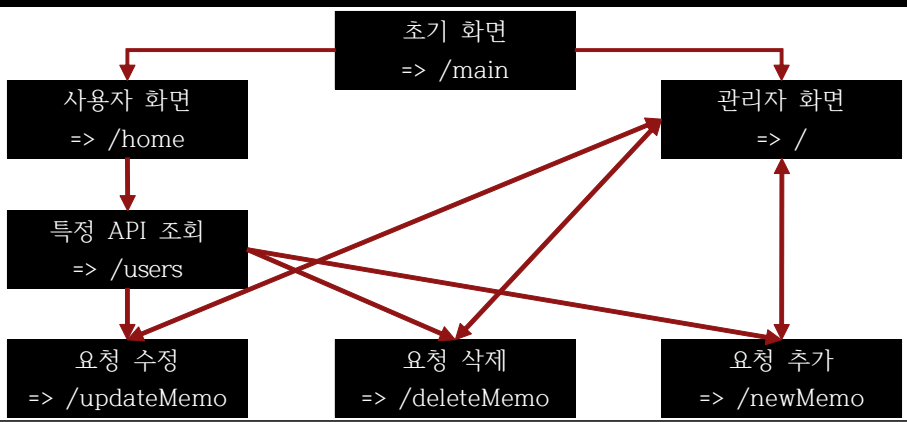
1. 업체관리 :
업체명, 주소, 대표번호, API Key 항목으로 CRUD가 가능해야 함. --> **구현**

2. 업체 계정 관리(고민 중) :
업체 대표 아이디로 요청할 경우 누가 요청한것인지 피드백 주기 어려 각 사이트 관리자 추가시 요청관리시스템에도 등록이 되도록 처리 해야 하나... --> X

3. 요청 관리 :
제목, 내용, 첨부파일(멀티 첨부 가능), 진행상태 항목으로 CRUD 가 가능해야 함 --> **구현**

4. 알림기능(문자 & 카톡 연동) :
요청 내역 등록 시 비제이월드 담당자에게 요청 내용 알림 통보 처리 완료시 요청자에게 알림 통보 --> X

- 자료 흐름도



- 디렉토리 구조

```
node_modules ---------- node.js를 통해 기본적으로 제공되는 모듈 집합
public ---------- view에서 사용되는 CSS파일
 ㄴ stylesheets
      ㄴ style.css
```

```
        ㄴ newMemo.css
        ㄴ stylehome.css
        ㄴ stylemain.css
        ㄴ users.css
        ㄴ part.css
        ㄴ popup.css
uploads ------------------------------------------------------------------ 서버에 파일 업로드 시 저장 되는 경로
routes ------------------------------------------------------------------------- 요청 수신 및 view 렌더링
 ㄴ index.js
 ㄴ user.js
 ㄴ home.js
 ㄴ main.js
 ㄴ part.js
 ㄴ popup.js
views ------------------------------------------------------------------- 라우팅을 통해 브라우저에서 보여지는 화면
 ㄴ error.ejs
 ㄴ index.ejs
 ㄴ newMemo.ejs
 ㄴ updateMemo.ejs
 ㄴ users.ejs
app.js ------------------------------------------------------- view 엔진 및 라우팅 기능을 하는 express 객체 세팅
db.js
package.json
packahe-lock.json
```

- 목차

1. 준비 단계

1.1 설치

npm(nodejs package manager), express-generator, MySQL 설치

1.2 테이블 생성

MySQL에서 요청 데이터를 관리 테이블과 첨부파일 관리 테이블을 생성

- MySQL

```
CREATE TABLE example_01 (
    id INT(11) NOT NULL PRIMARY KEY auto_increment,
    pp VARCHAR(30) NOT NULL,
    adress VARCHAR(30) NOT NULL,
    APIKey INT(50) NOT NULL,
    phone VARCHAR(30) NOT NULL,
```

```
    title VARCHAR(50) NOT NULL,
    request TEXT NOT NULL,
    con VARCHAR(30) NOT NULL,
    create_at timestamp null default null,
    update_at timestamp null default null
);

CREATE TABLE example_attachfile (
    seq INT(11) NOT NULL PRIMARY KEY auto_increment,
    example_seq INT(11) NOT NULL,
    ori_filename VARCHAR(100) NOT NULL,
    save_filename VARCHAR(100) NOT NULL,
    foreign key (seq) references example_01
);
```

2. 구현 단계

2.1 메인 스크립트 작성

자바 스크립트를 이용하여 각 모듈 호출을 통한 메인 스크립트가 되는 코드(app.js), GET-POST 방식을 이용하여 라우터 모듈로 사용할 코드(app.js) 작성

- app.js

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
var app = express();

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var homeRouter = require('./routes/home');
var mainRouter = require('./routes/main');
var popupRouter = require('./routes/popup');
var partRouter = require('./routes/part');

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/home', homeRouter);
app.use('/main', mainRouter);
app.use('/popup', popupRouter);
app.use('/part', partRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
```

```
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
app.listen(3000, () => console.log('서버가 3000번 포트에서 실행되었습니다.'));
```

2.2 DB 연동

MySQL을 통해 데이터베이스와 테이블의 기본 구조를 생성한 이후 모듈의 역할을 수행하는 코드(db.js)를 작성하여 자바 스크립트와 DB를 연동

- db.js

```
const mysql = require('mysql');
const connection = mysql.createConnection({
    host:'www.bjworld21.com',
    user:'bjworld_request',
    password:'reqg9@dg*dg1',
    port:3306,
    database:'bjworld_request',
    dateStrings: 'date'
});
```

2.3 CRUD 주요기능 스크립트 작성

2.3.1 DB 테이블 조회 프로세스

- index.js

```
router.get('/', function(req, res, next) {
  db.getAllMemos((rows) =>{
    res.render('index', { rows: rows });
  });
});
```

- db.js

```
function getAllMemos(callback){
    connection.query(`SELECT * FROM example_01 ORDER BY id DESC`, (err, rows, fields) => {
        if(err) throw err;
        callback(rows);
    });
}
```

2.3.2 테이블 추가 & 파일 업로드

- index.js

```
router.get('/newMemo',  function(req, res, next){
  res.render('newMemo');
});


rrouter.post('/store', [check('id').isInt()
```

```
, check('pp').isByteLength({min:1, max:30})
, check('adress').isByteLength({min:1, max:30})
, check('APIKey').islnt()
, check('phone').isByteLength({min:1, max:30})
, check('title').isByteLength({min:1, max:50})
, check('request').isByteLength({min:1})
, check('con').isByteLength({min:1, max:30})]
, uploader.array("photo", 5)
, function(req, res, next){
    let param = JSON.parse(JSON.stringify(req.body));
    db.insertMemo(param,(result) =>{
      let id = result.insertId;
      console.log(files);
      if (files.length == 1) {
        connection.query(`INSERT INTO example_attachfile (example_seq, ori_filename, save_filename)
        VALUES (
            '${id}','${files[i].originalname}','${files[i].filename}')`);
      } else {
      for(var i=0; i<files.length; i++) {
      connection.query(`INSERT INTO example_attachfile (example_seq, ori_filename, save_filename)
        VALUES (
            '${id}','${files[i].originalname}','${files[i].filename}')`);
        }
      }
    });
    try{
        var files = req.files;
        console.dir('#----------업로드 파일 정보--------#');
        console.dir(files[i]);
        console.dir('#---------------------------------#');
        var originalname = '', filename = '', mimetype = '', size = 0;
        if(Array.isArray(files)){
            console.log("배열 파일 갯수: %d", files.length);

            for(var i=0; i<files.length; i++){
                originalname = files[i].originalname;
                filename = files[i].filename;
                mimetype = files[i].mimetype;
                size = files[i].size;
            }
        }
        console.log("현재 파일 정보: " + originalname + ', ' + filename + ', ' + mimetype + ', ' + size);
        console.dir('#------파일 업로드 완료------#');
        res.redirect('/');
    } catch(err) {
        console.dir(err.stack);
    }
  }
);
```

- db.js

```
function insertMemo(param, callback){
    connection.query(`INSERT INTO example_01 (id, pp, adress, APIKey, phone, title, request, con, create_at,
update_at)
    VALUES (
```

```
            '${param.id}',
            '${param.pp}',
            '${param.adress}',
            '${param.APIKey}',
            '${param.phone}',
            '${param.title}',
            '${param.request}',
            '${param.con}',
            NOW(), NOW()
            )`,
            (err, result) =>{
            if (err) throw err;
            callback();
        });
}
```

2.3.3 테이블 수정 프로세스

- index.js

```
router.get('/updateMemo', (req, res) =>{
  let id = req.query.id;
  db.getMemoById(id, (row)=>{
    if(typeof id === 'undefinde' || row.length <= 0){
      res.status(404).json({error:'undefinde memo'});
    }else{
      res.render('updateMemo',{row:row[0]});
    }
  });
});

//게시글 수정 프로세스
router.post('/updateMemo', [check('id').isInt()
, check('pp').isByteLength({min:1, max:30})
, check('adress').isByteLength({min:1, max:30})
, check('APIKey').isInt()
, check('phone').isByteLength({min:1, max:30})
, check('title').isByteLength({min:1, max:50})
, check('request').isByteLength({min:1})
, check('con').isByteLength({min:1, max:30})]
, (req, res) =>{
  let errs = validationResult(req);
  let param = JSON.parse(JSON.stringify(req.body));
  console.log(req.body);
  let id = param['id'];
  let pp = param['pp'];
  let adress = param['adress'];
  let APIKey = param['APIKey'];
  let phone = param['phone'];
  let title = param['title'];
  let request = param['request'];
  let con = param['con'];

  if(errs['errors'].length > 0){

    db.getMemoById(id, (row)=>{
      res.render('updateMemo',{row:row[0], errs:errs['errors']});
```

```
    });
  }else{
    db.updateMemoById(id, pp, adress, APIKey, phone, title, request, con, () =>{
      res.redirect('/');
      console.dir('#-----수정이 완료되었습니다.------#')
      console.log(param);
    });
  }
});
```

- db.js

```
function getMemoById(id, callback){
    connection.query(`SELECT * FROM example_01 WHERE id='${id}'`, (err,row, fields) =>{
        if(err) throw err;
        callback(row);
    });
}

function updateMemoById(id, pp, adress, APIKey, phone, title, request, con, callback){
    connection.query(`UPDATE example_01 SET
    id='${id}',
    pp='${pp}',
    adress='${adress}',
    APIKey='${APIKey}',
    phone='${phone}',
    title='${title}',
    request='${request}',
    con='${con}',
    update_at=NOW()
    WHERE id='${id}'`,
    (err, result) => {
        if(err) throw err;
        callback();
    });
}
```

2.3.4 테이블 삭제 프로세스

- index.js

```
router.get('/deleteMemo', (req, res) =>{
  let id = req.query.id;
  db.deleteMemoById(id, () =>{
    res.redirect('/');
  });
});
```

- db.js

```
function deleteMemoById(id, callback){
    connection.query(`DELETE FROM example_01 WHERE id=${id}`, (err, result) =>{
        if(err) throw err;
        callback();
    });
}
```

2.3.5 API값을 통한 테이블 조회 프로세스

- home.js

```
router.get('/', check('APIKey').isInt(), function(req, res, next) {
  let APIKey = req.query.APIKey;
  db.getCertainMemos(APIKey, (rows) =>{
      res.render('home', { rows: rows });
```

```
  });
});

module.exports = router;
```

- home.ejs

```
<script>
      function myFunction() {
        var url = 'http://localhost:3000/users/?APIKey=';
        var key = document.getElementById("search_bar").value;
        window.location.href = url + key;

        function api() {
          let apikey = key;
        }
        module.exports = api;
        console.log(apikey);
      }
    </script>
```

2.3.6 팝업 프로세스

- popup.js

```
var express = require('express');
var router = express.Router();



router.get('/', function(req, res, next) {
      res.render('popup', );
});

module.exports = router;
```

- popup.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="author" content="99wo_ow">
    <meta name="description">
    <link href='/stylesheets/popup.css' rel='stylesheet' type="text/css">
    <title>Document</title>
</head>
<body>
    <div id="popup">
        <header class="top">
            <h1 class="infoTit">파일 목록</h1>
        </header>
        <main class="textBox">
            <table class="table_1">
                <tr>
                    <th>순번</th>
                    <th>원본파일명</th>
                    <th>저장파일명</th>
                    <th>파일</th>
                </tr>
                <tr>
                    <td><%= rows[0].seq %></td>
                    <td><%= rows[0].ori_filename %></td>
```

```
                        <td><%= rows[0].save_filename %></td>
                        <td><input id='filename' type="button"
onclick="javascript:location.href='/popup/downloads_1?example_seq=<%=rows[0].example_seq%>'"
value="다운로드"></td>
                    </tr>
                </table>
                <table class="table_2">
                    <tr>
                        <th>순번</th>
                        <th>원본파일명</th>
                        <th>저장파일명</th>
                        <th>파일</th>
                    </tr>
                    <tr>
                        <td><%= rows[1].seq %></td>
                        <td><%= rows[1].ori_filename %></td>
                        <td><%= rows[1].save_filename %></td>
                        <td><input id='filename' type="button"
onclick="javascript:location.href='/popup/downloads_2?example_seq=<%=rows[1].example_seq%>'"
value="다운로드"></td>
                    </tr>
                </table>
            </main>
        </div>
        <footer class="btnBox_todayClose">
            <form method="post"action=""name="pop_form">
                <span id="check"><input type="checkbox"value="checkbox"name="chkbox"id="chkday"
onclick='self.close()'/>
                    <label for="chkday" >확인하였습니다.</label>
                </span>
            </form>
        </footer>
</body>
</html>
```

- popup.js

```
router.get('/', check('example_seq').isInt(), function(req, res, next) {
        let example_seq = req.query.example_seq;
        console.dir('#----현재 페이지: id = ' + example_seq + '----#');
        console.dir('#---------호출한 파일 정보---------#');
        db.selectReq(example_seq, (rows) =>{
          res.render('popup', { rows: rows });
          console.log(rows);
      });
});
```

2.3.7 삭제 완료 알람

- index.js

```
function delalert() {
          alert('성공적으로 삭제되었습니다.');
        }


<input type="button" onclick="javascript:location.href='/deleteMemo?id=<%=row.id%>'; delalert();" value="삭제">
```

2.3.8 업로드 파일 다운로드 프로세스

- popup.js

```
router.get('/downloads_1', function(req, res, next) {
  let uploadPath = process.cwd() + "/uploads/";
```

```
  let example_seq = req.query.example_seq;
  db.selectReq(example_seq,(rows) => {
    if(!rows[0]) {
        alert('첨부된 파일이 없습니다.')
    } else {
        let fileName = rows[0].ori_filename;
        let saveFileName = rows[0].save_filename;
        res.download(uploadPath + saveFileName, fileName);
        console.dir('#----------파일 다운로드 완료----------#');
        console.dir('파일경로 : ' + uploadPath);
        console.dir('파일명 : ' + fileName);
    }
   });
});
```

3. HTML & CSS

3.1 HTML 작성

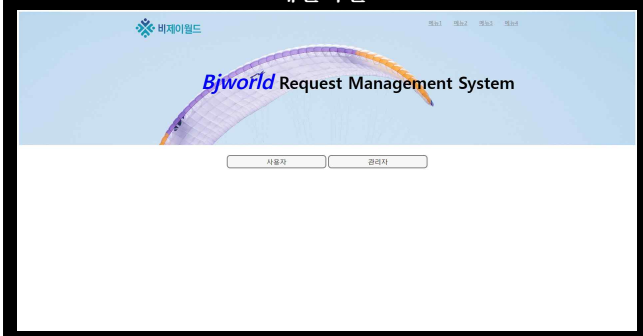
EJS 템플릿 엔진을 통해 HTML, Script 언어를 이용하여 HTML 코드를 작성

3.2 CSS 작성

EJS 파일에 스타일을 적용해 주는 코드 작성

4. 구현 화면

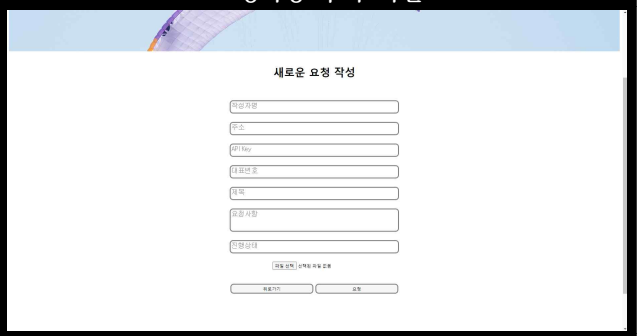
메인화면



관리자 화면

사용자 화면

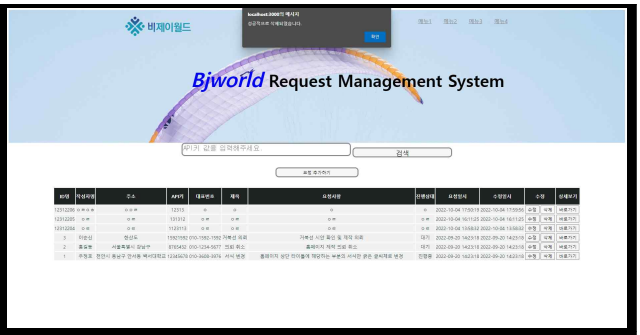
요청사항 추가 화면



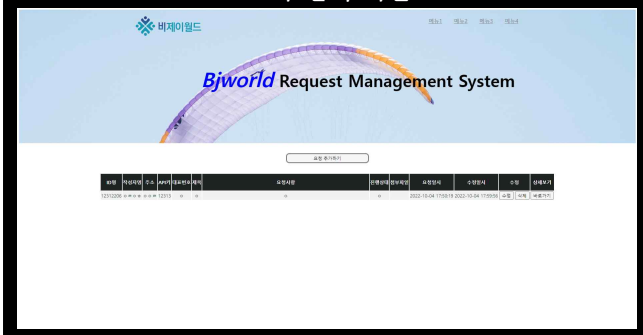
요청사항 수정 화면

요청사항 삭제 화면

API키 검색 화면

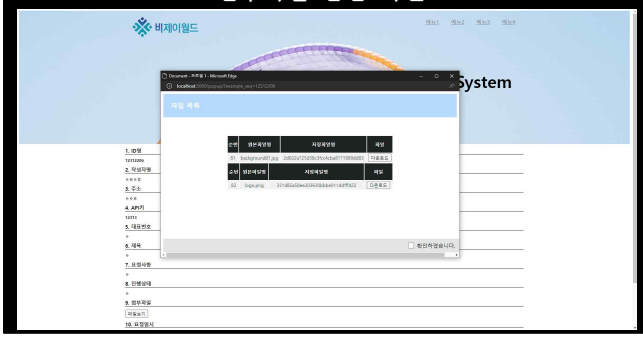


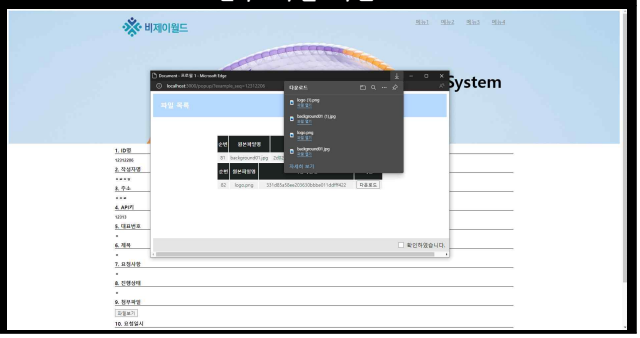
테이블 상세 보기 화면



첨부파일 팝업 화면

첨부 파일 다운로드





5. 보완점

1. 로그인 세션을 통한 사용자-관리자 간 접속 루트 차별
2. 요청사항 수정 화면에서의 첨부파일 수정 기능 추가
3. 진행상태는 관리자 아이디만 수정이 가능하도록 수정
4. 첨부파일마다 개별적인 버튼의 구현이 아닌 express-zip을 이용한 압축 파일의 업로드 및 다운로드 구현
5. 첨부파일명이 무작위로 DB에 저장되는 오류를 개선
6. 요청 수정 및 등록 시 관리자에게 개별 카카오톡, 문자 알람 기능 구현
7. 전반적인 UI 및 시각적인 요소 개선
8. 코드 리펙토링