

05장 실무 - 개발부터 배포까지 실무 응용

106 node.js express 프레임워크 설치하기

- react에서 create-react-app을 설치해 front-end 서버를 간편하게 구동했던 것처럼 node에서는 express를 사용하면 back-end 서버 구현을 쉽게 할 수 있다.

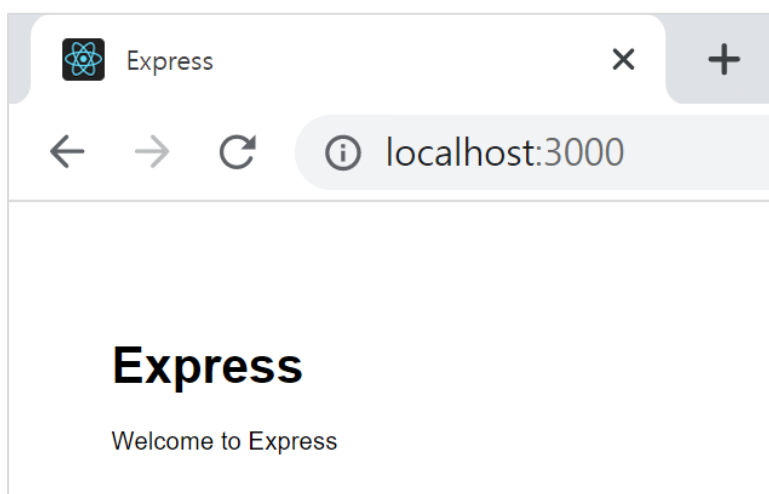
```
// express를 설치한다.  
// D:\dev\workspace\react>npm i -g express-generator  
D:\dev\workspace\react>yarn add express  
  
// express 프로젝트를 생성한다.  
D:\dev\workspace\react>express backend
```

107 node 서버 구동하기

- express 프로젝트를 생성하면 기본적인 웹 프레임워크 디렉터리 구조가 생성된다. React 프로젝트 구조와 마찬가지로 node 경로에는 package.json 파일이 존재하고, 서버 구동에 필요한 패키지들이 작성돼 있다. 작성돼 있는 패키지를 설치하면 서버를 구성할 수 있다.

```
// package.json에 작성된 패키지를 설치한다.  
D:\dev\workspace\react\backend>yarn install  
  
// 서버를 구동한다.  
D:\dev\workspace\react\backend>yarn start
```

- 실행 결과
 - 웹 브라우저에서 [localhost:3000]을 호출하면 다음과 같이 node express 서버가 구동된 것을 확인할 수 있다.



108 node 서버 api 호출하기

- node 경로(D:\dev\workspace\react\server)의 app.js 파일은 react에서 App.js 파일과 동일한 라우팅 기능을 담당한다.
- 1. node.js 는 back-end뿐 아니라 front-end도 구현할 수 있는 언어다. 하지만 front-end 언어로는 node보다 react가 강점이 많기 때문에 node 서버는 back-end api 서버로만 사용할 것이다. node 경로에서 불필요한 front-end 관련 소스 public, views 폴더를 삭제한다.
- 2. node 경로에서 app.js 파일명을 react의 App.js와 구분하기 위해 server.js로 수정한다. server.js에서 불필요한 코드를 삭제하고 다음과 같이 수정한다.
 - 03~04: react의 App.js에서 컴포넌트 파일을 임포트했던 것처럼 require 문법으로 라우터 파일 경로를 변수에 저장한다.
 - 08~09: 호출하는 서버 경로에 따라 line 3~4에서 지정한 파일로 라우팅한다.

[server/server.js]

```
01 var express = require('express');
02
03 var indexRouter = require('./routes/index');
04 var usersRouter = require('./routes/users');
05
06 var app = express();
07
08 app.use('/', indexRouter);
09 app.use('/users', usersRouter);
10
11 module.exports = app;
```

- 3. routes 폴더의 users.js 파일을 보면 /users 경로의 라우터가 구현돼 있다.
 - 05~06: http get 방식으로 request 호출을 받으면 res.send 함수가 텍스트 데이터를 response로 전송한다.

[server/routes/users.js]

```
01 var express = require('express');
02 var router = express.Router();
03
04 /* GET users listing. */
05 router.get('/', function(req, res, next) {
06   res.send('respond with a resource');
07 });
08
09 module.exports = router;
```

- 4. bin 폴더의 www 파일을 열어 다음과 같이 수정한다.
 - 07: app.js 파일명을 server.js로 바꿨기 때문에 server.js로 require하도록 수정한다.
 - 15: react 서버와 node 서버를 동시에 구동하기 위해 node 서버의 포트를 5000번으로 지정한다.

[server/bin/www]

```
01 #!/usr/bin/env node
02
03 /**
```

```

04  * Module dependencies.
05  */
06
07  var app = require('..../server');
08  var debug = require('debug')('server:server');
09  var http = require('http');
10
11  /**
12   * Get port from environment and store in Express.
13   */
14
15  var port = normalizePort(process.env.PORT || '5000');
16  app.set('port', port);
17
18  /**
19   * Create HTTP server.
20   */
21
22  var server = http.createServer(app);
23
24  /**
25   * Listen on provided port, on all network interfaces.
26   */
27
28  server.listen(port);
29  server.on('error', onError);
30  server.on('listening', onListening);
31
32  /**
33   * Normalize a port into a number, string, or false.
34   */
35
36  function normalizePort(val) {
37    var port = parseInt(val, 10);
38
39    if (isNaN(port)) {
40      // named pipe
41      return val;
42    }
43
44    if (port >= 0) {
45      // port number
46      return port;
47    }
48
49    return false;
50  }
51
52  /**
53   * Event listener for HTTP server "error" event.
54   */
55
56  function onError(error) {
57    if (error.syscall !== 'listen') {
58      throw error;
59    }
60
61    var bind = typeof port === 'string'
62      ? 'Pipe ' + port
63      : 'Port ' + port;
64
65    // handle specific listen errors with friendly messages
66    switch (error.code) {
67      case 'EACCES':
68        console.error(bind + ' requires elevated privileges');

```

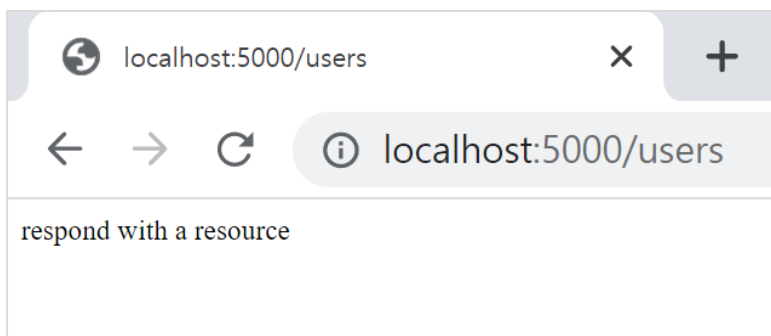
```

69     process.exit(1);
70     break;
71     case 'EADDRINUSE':
72         console.error(bind + ' is already in use');
73         process.exit(1);
74         break;
75     default:
76         throw error;
77     }
78 }
79
80 /**
81  * Event listener for HTTP server "listening" event.
82  */
83
84 function onListening() {
85     var addr = server.address();
86     var bind = typeof addr === 'string'
87         ? 'pipe ' + addr
88         : 'port ' + addr.port;
89     debug('Listening on ' + bind);
90 }

```

■ 실행 결과

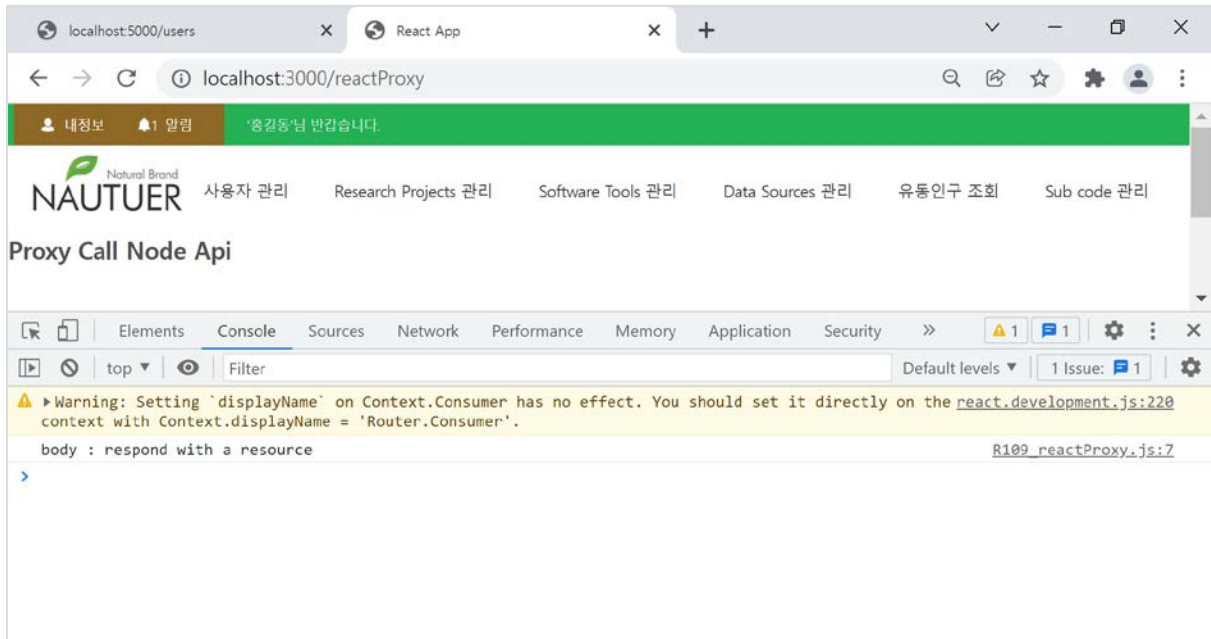
- 웹 브라우저에서 node 서버 주소(localhost:5000)에 users 경로로 호출하면 user.js의 response로 반환한 텍스트가 화면에 표시된다.



109 react 서버와 node 서버를 프록시로 연결하기

- 프록시란 클라이언트가 다른 서버에 간접적으로 접속할 수 있도록 중계해주는 프로그램이다. 예를 드면, 웹 브라우저에 react 서버(localhost:3000)을 띄우고 react 페이지에서 no api 를 호출해 데이터를 표시할 수 있다. 이때 프록시를 설정하면 node 서버 url을 localhost:5000/user가 아닌 /users (localhost:3000/users) 간략하게 사용할 수 있다. 프록시를 보통 보안이나 캐시 목적으로 사용하는데 예제에서는 node 서버 호출 url을 간략히 사용하기 위한 용도로 사용한다.

■ 실행 결과



110 node 서버 api를 get로 json 데이터 호출하기

- react에서 json 데이터를 받아 사용하기 위해서는 node 서버에서 json 형태로 데이터를 response에 담아 보내줘야 한다.
- server/routes/users.js 파일을 수정한다.
 - 05~07: node 서버(localhost:5000)에서 /users 경로를 get 방식으로 호출하면 key가 message이고 value가 node get success인 json 데이터를 response로 전송한다.

[server/routes/users.js]

```
01 var express = require('express');
02 var router = express.Router();
03
04 /* GET users listing. */
05 router.get('/', function(req, res, next) {
06   res.send({ 'message': 'node get success' });
07 });
08
09 module.exports = router;
```

- client/src/components/App.js 파일을 작성한다.
 - 17,27: client 서버(localhost:3000)에서 /AgiGetJson 경로를 호출한다.

[client/src/components/App.js]

```
01 import React, { Component } from 'react';
02 import { Route } from "react-router-dom";
03
04 // css
05 import '../css/new.css';
06
07 // header
08 import HeaderAdmin from './Header/Header admin';
```

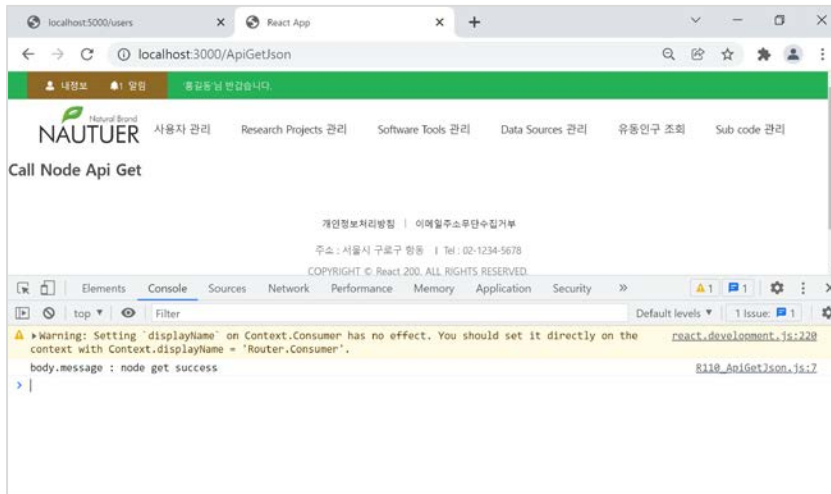
```
09
10 // footer
11 import Footer from './Footer/Footer';
12
13 // login
14 import LoginForm from './LoginForm';
15
16 import reactProxy from './R109_reactProxy';
17 import ApiGetJson from './R110_ApiGetJson';
18
19
20 class App extends Component {
21   render () {
22     return (
23       <div className="App">
24         <HeaderAdmin/>
25         <Route exact path="/" component={LoginForm} />
26         <Route exact path="/reactProxy" component={reactProxy} />
27         <Route exact path="/ApiGetJson" component={ApiGetJson} />
28         <Footer/>
29       </div>
30     );
31   }
32 }
33
34 export default App;
```

- client/src/components/R110_ApiGetJson.js 파일을 작성한다.

[client/src/components/R110_ApiGetJson.js]

```
01 import React, { Component } from 'react';
02
03 class R110_ApiGetJson extends Component {
04   componentDidMount = async () => {
05     const response = await fetch('/users');
06     const body = await response.json();
07     console.log("body.message : "+body.message)
08   }
09
10   render() {
11     return (
12       <><h1>Call Node Api Get</h1></>
13     )
14   }
15 }
16
17 export default R110_ApiGetJson;
```

- 실행 결과



111 node 서버 api를 post로 json 데이터 호출하기

- react에서 post 호출로 json 데이터를 받아오기 위해서는 node 서버에서 post 호출을 받는 라우터 코드를 추가해야 한다.
- server/routes/users.js 파일을 작성한다.

[server/routes/users.js]

```
01 var express = require('express');
02 var router = express.Router();
03
04 /* Post users listing. */
05 router.post('/', function(req, res, next) {
06   res.send({'message': 'node post success'});
07 });
08
09 module.exports = router;
```

- client/src/components/R11_ApiPostJson.js 파일을 작성한다.

[client/src/components/R11_ApiPostJson.js]

```
01 import React, { Component } from 'react';
02 const axios = require('axios');
03
04 class R111_ApiPostJson extends Component {
05   componentDidMount(){
06     axios.post('/users', {
07     })
08     .then( response => {
09       console.log("response.data.message : "+response.data.message)
10     })
11   }
12
13   render() {
14     return (
15       <<h1>Call Node Api Post</h1></>
16     )
17   }
18 }
```

```

18 }
19
20 export default R111_ApiPostJson;

```

- client/src/components/App.js 파일을 작성한다.

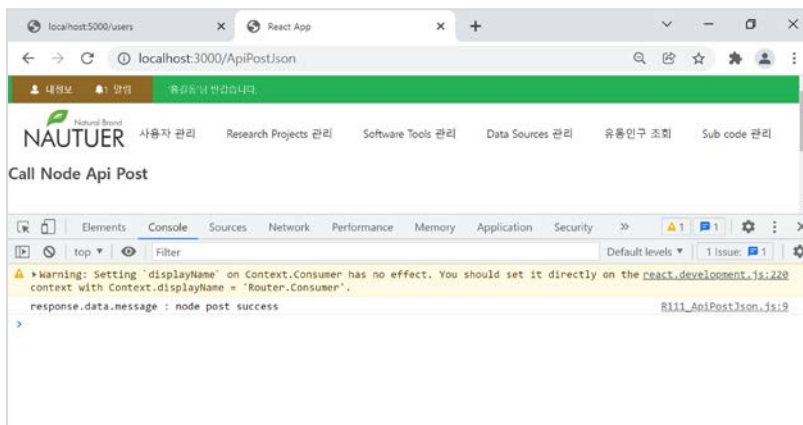
[client/src/components/App.js]

```

01 import React, { Component } from 'react';
02 import { Route } from "react-router-dom";
03
04 // css
05 import '../css/new.css';
06
07 // header
08 import HeaderAdmin from './Header/Header admin';
09
10 // footer
11 import Footer from './Footer/Footer';
12
13 // login
14 import LoginForm from './LoginForm';
15
16 import reactProxy from './R109_reactProxy';
17 import ApiGetJson from './R110_ApiGetJson';
18 import ApiPostJson from './R111_ApiPostJson';
19
20 class App extends Component {
21   render () {
22     return (
23       <div className="App">
24         <HeaderAdmin/>
25         <Route exact path="/" component={LoginForm} />
26         <Route exact path="/reactProxy" component={reactProxy} />
27         <Route exact path="/ApiGetJson" component={ApiGetJson} />
28         <Route exact path="/ApiPostJson" component={ApiPostJson} />
29         <Footer/>
30       </div>
31     );
32   }
33 }
34
35 export default App;

```

- 실행 결과



~~112 concurrently로 react, node 서버 한 번에 구동하기~~

~~113 MYSQL 서버 구축하기 1~~

~~114 MYSQL 서버 구축하기 2~~

~~115 MYSQL 서버 구축하기 3~~

~~116 MYSQL client 설치하기~~

~~117 workbench로 rds 서버 접속하기~~

118 MYSQL 스키마 생성 및 table 생성하기

```
-- root 계정으로 실행한다.
create database react default character set utf8;
create user 'react'@'localhost' identified by 'react';
grant all privileges on react.* to 'react'@'localhost';
create user 'react'@'%' identified by 'react';
grant all privileges on react.* to 'react'@'%';

-- react 계정으로 실행한다.
-- use react;
CREATE TABLE `react_swtool` (
  `swt_code` varchar(100) NOT NULL COMMENT 'SW툴 코드',
  `swt_toolname` varchar(100) DEFAULT NULL COMMENT '툴 이름',
  `swt_function` text COMMENT '상세 기능',
  `swt_imagepath` varchar(100) DEFAULT NULL COMMENT '라벨 이미지 경로',
  `swt_big_imgpath` varchar(100) DEFAULT NULL COMMENT '메인 이미지 경로',
  `swt_comments` text COMMENT '설명',
  `swt_demo_site` varchar(100) DEFAULT NULL COMMENT '데모 URL',
  `swt_manual_path` varchar(100) DEFAULT NULL COMMENT '메뉴얼 파일 경로',
  `swt_github_url` varchar(100) DEFAULT NULL COMMENT 'Github URL',
  `reg_date` varchar(100) DEFAULT NULL COMMENT '등록날짜',
  `reg_user` varchar(100) DEFAULT NULL COMMENT '등록자',
  `update_date` varchar(100) DEFAULT NULL COMMENT '수정날짜',
  `update_user` varchar(100) DEFAULT NULL COMMENT '수정자',
  PRIMARY KEY (`swt_code`)
) engine=innodb default character set = utf8;
```

~~119 MYSQL 스키마 생성 및 table 생성하기~~

~~120 MYSQL 한글 설정 및 Safe 모드 해제하기~~

121 MYSQL 데미 데이터 삽입하기

```
-- react 계정으로 실행한다.
-- use react;

INSERT INTO `react_swtool` VALUES ('USW20200101000000', '툴 이름1', '상세 기능1', '20200101000000_라벨 이미지.png', '20200101000000_메인 이미지.png', '설명1', '데모 URL1', '20200101000000_메뉴얼 파일.docx', 'Github URL1', '20200101000000', 'userA1', '20200102000000', 'userB1');
```

```
INSERT INTO `react_swtool` VALUES ('USW20200102000000', '툴 이름2', '상세 기능2', '20200102000000_라벨 이미지.png', '20200102000000_메인 이미지.png', '설명2', '데모 URL2', '20200102000000_메뉴얼 파일.docx', 'Github URL2', '20200102000000', 'userA2', '20200103000000', 'userB2');

select * from react_swtool;
```

122 NODE 조회 api 만들기 1 - body-parser 패키지 사용하기

- post 방식으로 api를 호출하는 경우, 전달할 데이터들을 request body에 담아 전달할 수 있다. 호출받은 node 서버에서 body-parser 패키지를 사용하면 간편하게 body에 있는 파라미터를 추출해 사용할 수 있다.

```
// node server에 body-parser를 설치한다.
D:\dev\workspace\react\server>yarn add body-parser
```

- server.js 파일

```
[server/server.js]

01 var express = require('express');
02
03 var indexRouter = require('./routes/index');
04 var usersRouter = require('./routes/users');
05 var swtoolRouter = require("./routes/SwtoolRoute");
06
07 var app = express();
08
09 app.use('/', indexRouter);
10 app.use('/users', usersRouter);
11 app.use("/api/Swtool", swtoolRouter);
12
13 const port = process.env.PORT || 5000;
14 app.listen(port, () => console.log(`Listening on port ${port}`));
```

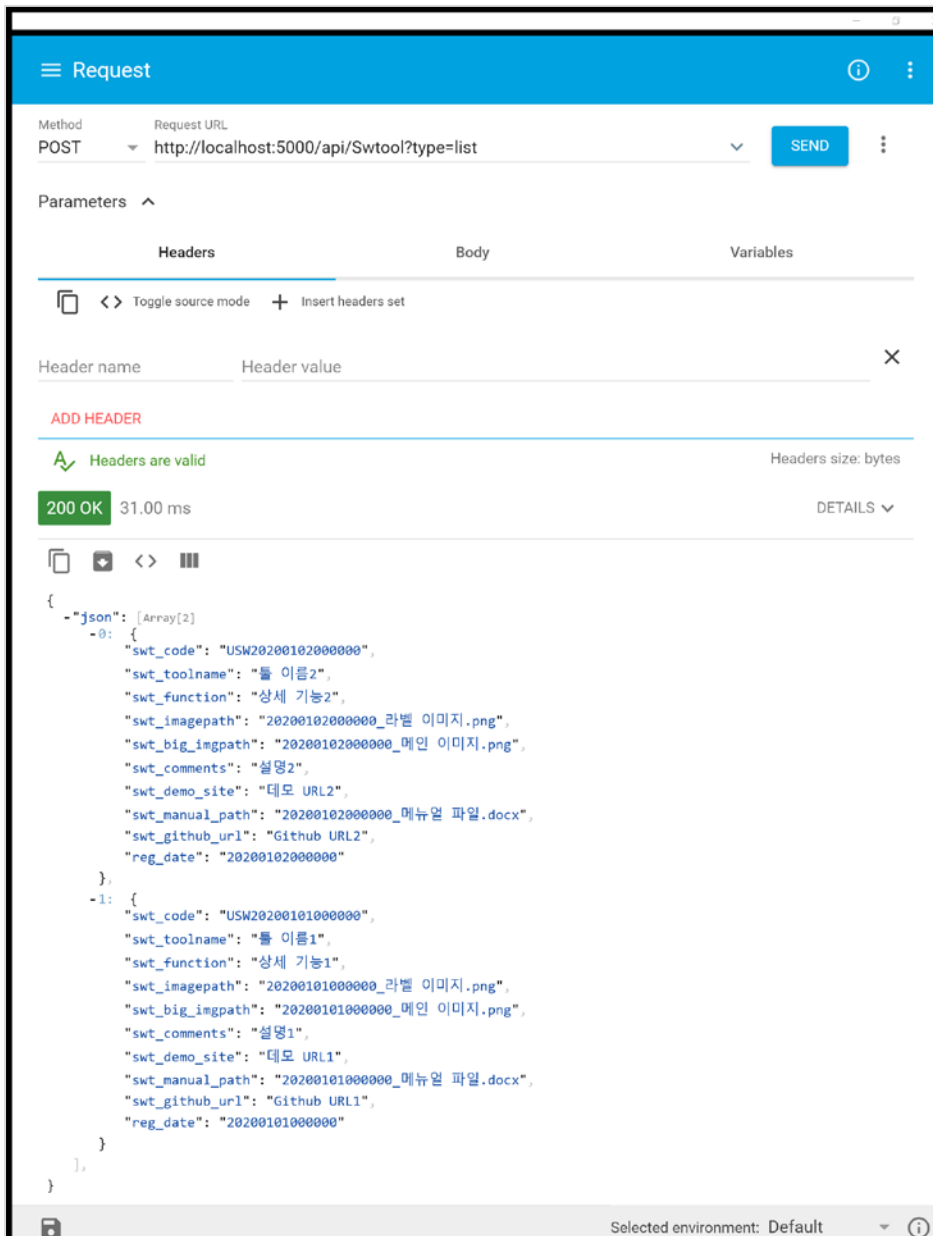
- 실행 결과
 - <http://localhost:5000/api/Swtool?type=list>

123 NODE 조회 api 만들기 2

124 NODE 조회 api 만들기 3

125 NODE 조회 api 만들기 4

- 실행 결과
 - Method: POST
 - Request URL: <http://localhost:5000/api/Swtool?type=list>



126 REACT 조회 페이지 만들기

- react 컴포넌트에서 post 방식으로 axios를 사용해 node api를 호출한다. 호출된 데이터를 가공해 html 코드로 만들고 state 변수에 할당한다. html 코드가 할당된 state 변수를 render 함수를 통해 화면에 표시한다.
- client/src/components/SoftwareToolsManage/SoftwareList.js 파일

```
[client/src/components/SoftwareToolsManage/SoftwareList.js]
```

```
01 import React, { Component } from 'react';
02 import { Link } from 'react-router-dom';
03 import axios from "axios";
04
05 class SoftwareList extends Component {
```

```

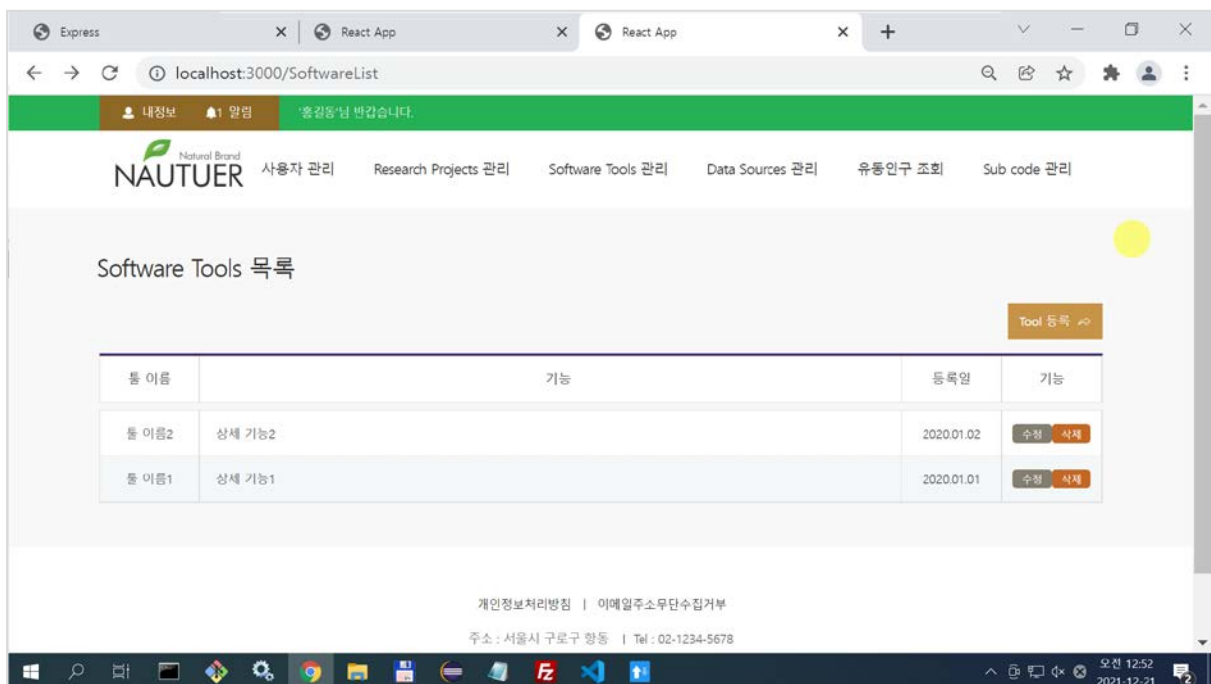
06   constructor(props) {
07       super(props);
08
09       this.state = {
10           responseSwtoolList: '',
11           append_SwtoolList: '',
12       }
13   }
14
15   componentDidMount() {
16       this.callSwToolListApi()
17   }
18
19   callSwToolListApi = async () => {
20       axios.post('/api/Swtool?type=list', {
21       })
22       .then( response => {
23           try {
24               this.setState({ responseSwtoolList: response });
25               this.setState({ append_SwtoolList: this.SwToolListAppend() });
26           } catch (error) {
27               alert('작업중 오류가 발생하였습니다.');
```

```

71         </div>
72     </div>
73
74     <div class="list_cont list_cont_admin">
75         <table class="table_ty1 ad_tlist">
76             <tr>
77                 <th>툴 이름</th>
78                 <th>기능</th>
79                 <th>등록일</th>
80                 <th>기능</th>
81             </tr>
82         </table>
83         <table class="table_ty2 ad_tlist">
84             {this.state.append_SwtoolList}
85         </table>
86     </div>
87 </article>
88 </section>
89 );
90 }
91 }
92
93 export default SoftwareList;

```

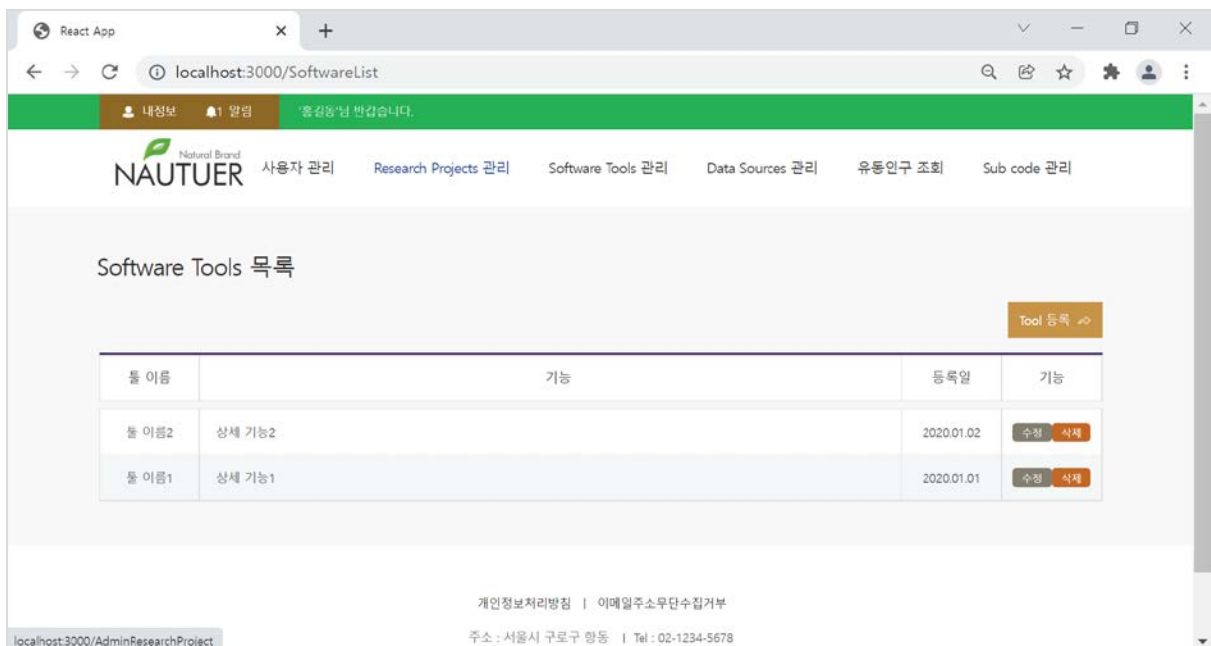
■ 실행 결과



127 DB 연결 풀 구현하기

- 사용자가 웹 사이트에 접속해 DB를 사용하게 되면 node 서버에서 mysql 서버로 connection을 만든다. 그리고 DB 사용이 완료되면 connection을 종료한다. 이때 node 서버와 mysql 서버 사이에 connection을 만드는 작업은 시간적 자원이 소모된다. connection pool은 한 번 맺은 connection을 종료시키지 않고 pool에 저장한다. 그리고 새로운 connection 요청이 오면 pool에 저장된 connection을 제공한다.

■ 실행 결과



128 NODE 등록 api 만들기 1 - insert 라우터 분기하기

- 조회 api를 구현할 때 데이터 연동을 위한 기본 틀이 갖춰졌다. 전달하는 데이터만 달라질 뿐 등록, 수정, 삭제 기능도 동일한 틀 안에서 분기 처리해 사용할 수 있다. react에서 호출할 등록 api는 /api/Swtool?type=save 이다.
- SwtoolRoute.js 파일
 - 34~36: 파라미터로 전달받은 type 값이 save인 경우, insert 쿼리를 실행할 mapper 정보를 body에 할당한다.

[routes/SwtoolRoute.js]

```

01 var express = require('express');
02
03 var router = express.Router();
04 const bodyParser = require('body-parser');
05
06 router.use(bodyParser.urlencoded({ extended: true }));
07 router.use(bodyParser.json());
08
09 router.post('/', (req, res, next) => {
10   var type = req.query.type;
11   if(type == 'list'){
12     //Swtool 리스트 조회
13     try {
14       // Mysql Api 모듈(CRUD)
15       var dbconnect_Module = require('./dbconnect_Module');
16
17       //Mysql 쿼리 호출 정보 입력
18       req.body.mapper = 'SwToolsMapper'; //mybatis xml 파일명
19       req.body.crud = 'select'; //select, insert, update, delete 중에 입력
20       req.body.mapper_id = 'selectSwToolsList';

```

```

21
22     router.use('/', dbconnect_Module);
23     next('route')
24   } catch (error) {
25     console.log("Module > dbconnect error : "+ error);
26   }
27 }else if(type == 'save'){
28   //Swtool 관리자 저장
29   try {
30     // Mysql Api 모듈(CRUD)
31     var dbconnect_Module = require('./dbconnect_Module');
32
33     //Mysql 쿼리 호출정보 입력
34     req.body.mapper = 'SwToolsMapper';//mybatis xml 파일명
35     req.body.crud = 'insert';//select, insert, update, delete 중에 입력
36     req.body.mapper_id = 'insertSwToolsInfo';
37
38     router.use('/', dbconnect_Module);
39     next('route')
40   } catch (error) {
41     console.log("Module > dbconnect error : "+ error);
42   }
43 }
44 });
45
46 module.exports = router;

```

■ dbconnect_Module.js 파일

- 24: SwtoolRoute.js에서 전달받은 req.body.mapper 값인 SwToolsMapper로 사용할 mapper 경로를 설정한다.
- 31: SwtoolRoute.js에서 전달받은 req.body.mapper_id 값인 insertSwToolsInfo로 query 변수를 생성한다.

[routes/dbconnect_Module.js]

```

01 var express = require("express");
02 var router = express.Router();
03 const mysql = require("mysql");
04 const bodyParser = require("body-parser");
05
06 router.use(bodyParser.json());
07
08 // Connection Pool 세팅
09 const pool = mysql.createPool({
10   connectionLimit: 66,
11   waitForConnections: true,
12   host: "react200.cinvalghkckt.ap-northeast-2.rds.amazonaws.com",
13   port: "3306",
14   database: 'react',
15   user: "admin",
16   password: "react200RDS",
17 });
18
19 router.post("/", (req, res) => {
20   const mybatisMapper = require("mybatis-mapper");
21   var param = req.body;
22
23   //mybatis mapper경로 설정
24   mybatisMapper.createMapper(['./models/'+param.mapper+'.xml']);
25   var time = new Date();
26   console.log('## '+time+ ' ##');
27   console.log("\n Called Mapper Name  = "+param.mapper);

```

```

28
29 var format = { language: 'sql', indent: ' ' };
30 //mysql 쿼리 정보 세팅
31 var query = mybatisMapper.getStatement(param.mapper, param.mapper_id, param, format);
32 console.log("\n===== Node Mybatis Query Log Start =====");
33 console.log("* mapper namespc : "+param.mapper+"."+param.mapper_id+" *\n");
34 console.log(query+"\n");
35
36 pool.getConnection(function(err,connection){
37   connection.query(query, function (error, results) {
38     if (error) {
39       console.log("db error***** : "+error);
40     }
41     var time2 = new Date();
42     console.log('## '+time2+ ' ##');
43     console.log('## RESULT DATA LIST ## : \n', results);
44     if(results != undefined){
45       string = JSON.stringify(results);
46       var json = JSON.parse(string);
47       if (req.body.crud == "select") {
48         res.send({ json });
49       }else{
50         res.send("succ");
51       }
52     }else{
53       res.send("error");
54     }
55   }
56   connection.release();
57   console.log("===== Node Mybatis Query Log End =====\n");
58 });
59 })
60 });
61
62 module.exports = router;

```

129 NODE 등록 api 만들기 2 - insert 쿼리 추가, response 처리하기

- xml 파일(mapper)에 쿼리를 추가하는 방식은 select 대신 <insert> 태그를 사용하는 것 외에는 select에서 사용한 것과 같다. 쿼리를 실행한 후 select는 조회된 데이터를 response로 넘기지만 insert는 성공 여부만 전달한다.
- SwToolsMapper.xml 파일
 - 21: <insert> 태그를 사용해 mapperid를 정의한다.
 - 30~32: mybatis에서는 전달받은 파라미터로 if문을 사용할 수 있다. 필수 값이 아닌 컬럼 값이 null이나 공백으로 넘어올 경우에는 테이블에 삽입하지 않는다.
 - 45: 테이블에서 유일한 값을 갖는 기본 키인 swt_code의 값은 USW라는 문자열에 현재 시간 정보를 붙여 생성한다.

[models/SwToolsMapper.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-
03 mapper.dtd">
04 <mapper namespace="SwToolsMapper">
05   <select id="selectSwToolsList">

```



```

06      SELECT
07          swt_code
08      ,   swt_toolname
09      ,   swt_function
10      ,   swt_imagepath
11      ,   swt_big_imgpath
12      ,   swt_comments
13      ,   swt_demo_site
14      ,   swt_manual_path
15      ,   swt_github_url
16      ,   reg_date
17      FROM react.react_swtool
18      ORDER BY update_date DESC
19  </select>
20
21  <insert id="insertSwToolsInfo">
22      INSERT INTO react.react_swtool
23      (
24          swt_code
25      ,   swt_toolname
26      ,   swt_function
27      ,   swt_comments
28      ,   swt_demo_site
29      ,   swt_github_url
30      <if test="is_LabelImg != null && is_LabelImg != ''">
31          ,   swt_imagepath
32      </if>
33      <if test="is_MainImg != null && is_MainImg != ''">
34          ,   swt_big_imgpath
35      </if>
36      <if test="is_MenualName != null && is_MenualName != ''">
37          ,   swt_manual_path
38      </if>
39      ,   reg_date
40      ,   reg_user
41      ,   update_date
42      ,   update_user
43      )
44      VALUES (
45          CONCAT('USW', DATE_FORMAT(now(), '%Y%m%d%H%i%s'))
46      ,   #{is_Swt_toolname}
47      ,   #{is_Swt_function}
48      ,   #{is_Comments}
49      ,   #{is_Swt_demo_site}
50      ,   #{is_Giturl}
51      <if test="is_LabelImg != null && is_LabelImg != ''">
52          ,   #{is_LabelImg}
53      </if>
54      <if test="is_MainImg != null && is_MainImg != ''">
55          ,   #{is_MainImg}
56      </if>
57      <if test="is_MenualName != null && is_MenualName != ''">
58          ,   #{is_MenualName}
59      </if>
60      ,   DATE_FORMAT(now(), '%Y%m%d%H%i%s')
61      ,   #{is_Email}
62      ,   DATE_FORMAT(now(), '%Y%m%d%H%i%s')
63      ,   #{is_Email}
64      )
65  </insert>
66  </mapper>

```

■ dbconnect_Module.js 파일

- 47~51: SwtoolRout.js에서 할당한 req.body.crud 값에 따라 response로 전송할 데이터를 분기 처리한다. 조회(select)인 경우, DB에서 조회한 데이터를 json 형태로 response에 담아 react 페이지로 다시 보내야 한다. 그 밖의 경우(insert, update, delete)인 경우, 쿼리 실행이 성공했을 경우 "succ"라는 문자열을 전송한다.
- 52~54: 쿼리 실행 결과가 undefined라면 쿼리가 정상적으로 실행되지 않은 경우다. 이때는 error라는 문자열을 전송한다.

[routes/dbconnect_Module.js]

```

01 var express = require("express");
02 var router = express.Router();
03 const mysql = require("mysql");
04 const bodyParser = require("body-parser");
05
06 router.use(bodyParser.json());
07
08 // Connection Pool 세팅
09 const pool = mysql.createPool({
10   connectionLimit: 66,
11   waitForConnections: true,
12   host: "react200.cinvalghkckt.ap-northeast-2.rds.amazonaws.com",
13   port: "3306",
14   database: 'react',
15   user: "admin",
16   password: "react200RDS",
17 });
18
19 router.post("/", (req, res) => {
20   const mybatisMapper = require("mybatis-mapper");
21   var param = req.body;
22
23   //mybatis mapper경로 설정
24   mybatisMapper.createMapper(['./models/'+param.mapper+'.xml']);
25   var time = new Date();
26   console.log('## '+time+ ' ##');
27   console.log("\n Called Mapper Name = "+param.mapper);
28
29   var format = { language: 'sql', indent: ' ' };
30   //mysql 쿼리 정보 세팅
31   var query = mybatisMapper.getStatement(param.mapper, param.mapper_id, param, format);
32   console.log("\n===== Node Mybatis Query Log Start =====");
33   console.log("* mapper namespce : "+param.mapper+"."+param.mapper_id+" *\n");
34   console.log(query+"\n");
35
36   pool.getConnection(function(err,connection){
37     connection.query(query, function (error, results) {
38       if (error) {
39         console.log("db error***** : "+error);
40       }
41       var time2 = new Date();
42       console.log('## '+time2+ ' ##');
43       console.log('## RESULT DATA LIST ## : \n', results);
44       if(results != undefined){
45         string = JSON.stringify(results);
46         var json = JSON.parse(string);
47         if (req.body.crud == "select") {
48           res.send({ json });
49         }else{
50           res.send("succ");
51         }
52       }else{
53         res.send("error");
54       }
55     });
56   });
57 }

```

```

55
56     connection.release();
57     console.log("===== Node Mybatis Query Log End =====\n");
58 });
59 })
60 });
61
62 module.exports = router;

```

130 REACT 등록 페이지 만들기

- react 컴포넌트에서 post 방식으로 fetch를 사용해 node api를 호출한다. insert 쿼리에 사용할 데이터를 json 형태로 변경해 전달하고 쿼리가 정상적으로 실행된다면 완료 후 리스트 페이지로 이동한다.

- SoftwareView.js 파일

[client/src/components/SoftwareToolsManage/SoftwareView.js]

```

01 import React, { Component } from 'react';
02 import { Link } from 'react-router-dom';
03 import $ from 'jquery';
04 import Swal from 'sweetalert2'
05
06 class SoftwareView extends Component {
07     submitClick = async (type, e) => {
08
09         this.Swt_toolname_checker = $('#is_Swt_toolname').val();
10         this.Swt_demo_site_checker = $('#is_Swt_demo_site').val();
11         this.Giturl_checker = $('#is_Giturl').val();
12         this.Comments_checker = $('#is_Comments').val();
13         this.Swt_function_checker = $('#is_Swt_function').val();
14
15         this.fnValidate = (e) => {
16             if(this.Swt_toolname_checker === '') {
17                 $('#is_Swt_toolname').addClass('border_validate_err');
18                 alert('툴 이름을 다시 확인해주세요.')
19                 return false;
20             }
21             $('#is_Swt_toolname').removeClass('border_validate_err');
22
23             if(this.Swt_demo_site_checker === '') {
24                 $('#is_Swt_demo_site').addClass('border_validate_err');
25                 alert('데모 URL을 다시 확인해주세요.')
26                 return false;
27             }
28             $('#is_Swt_demo_site').removeClass('border_validate_err');
29
30             if(this.Giturl_checker === '') {
31                 $('#is_Giturl').addClass('border_validate_err');
32                 alert('Github URL을 다시 확인해주세요.')
33                 return false;
34             }
35             $('#is_Giturl').removeClass('border_validate_err');
36
37             if(this.Comments_checker === '') {
38                 $('#is_Comments').addClass('border_validate_err');
39                 alert('설명을 다시 확인해주세요.')
40                 return false;
41             }

```

```

42      $('#is_Comments').removeClass('border_validate_err');
43
44      if(this.Swt_function_checker === '') {
45          $('#is_Swt_function').addClass('border_validate_err');
46          alert('상세기능을 다시 확인해주세요.')
47          return false;
48      }
49      $('#is_Swt_function').removeClass('border_validate_err');
50      return true;
51  }
52
53  if(this.fnValidate()){
54      var jsonstr = $("form[name='frm']").serialize();
55      jsonstr = decodeURIComponent(jsonstr);
56      var Json_form = JSON.stringify(jsonstr).replace(/\\"/gi, '')
57      Json_form = "{\\" + Json_form.replace(/\&/g, '\&', '\\').replace(/=/gi, '\":'+\\"}";
58
59      try {
60          const response = await fetch('/api/Swtool?type='+type, {
61              method: 'POST',
62              headers: {
63                  'Content-Type': 'application/json',
64              },
65              body: Json_form,
66          });
67          const body = await response.text();
68          if(body == "succ"){
69              if(type == 'save'){
70                  this.sweetalertSucc('Software Tools 등록이 완료되었습니다.', false)
71              }
72              setTimeout(function() {
73                  this.props.history.push('/SoftwareList');
74              }.bind(this),1500
75          );
76          }else{
77              alert('작업중 오류가 발생하였습니다.')
78          }
79      } catch (error) {
80          alert('작업중 오류가 발생하였습니다.')
81      }
82  }
83  };
84
85  sweetalertSucc = (title, showConfirmButton) => {
86      Swal.fire({
87          position: 'bottom-end',
88          icon: 'success',
89          title: title,
90          showConfirmButton: showConfirmButton,
91          timer: 1000
92      })
93  }
94
95  render () {
96      return (
97          <section class="sub_wrap">
98              <article class="s_cnt mp_pro_li ct1">
99                  <div class="li_top">
100                      <h2 class="s_tit1">Software Tools 등록/수정</h2>
101                  </div>
102                  <div class="bo_w re1_wrap re1_wrap_writer">
103                      <form name="frm" id="frm" action="" onsubmit="" method="post" >
104                          <input id="is_Swtcode" type="hidden" name="is_Swtcode" />
105                          <input id="is_Email" type="hidden" name="is_Email" value="guest" />
106                      <article class="res_w">

```

```

107         <p class="ment" style={{ "text-align": "right" }}>
108             <span class="red">(*)</span>표시는 필수입력사항 입니다.
109         </p>
110         <div class="tb_outline">
111             <table class="table_ty1">
112                 <tr>
113                     <th>
114                         <label for="is_Swt_toolname">툴 이름<span
115 class="red">(*)</span></label>
116                     </th>
117                     <td>
118                         <input type="text" name="is_Swt_toolname"
119 id="is_Swt_toolname" class="" />
120                     </td>
121                 </tr>
122                 <tr>
123                     <th>
124                         <label for="is_Swt_demo_site">데모 URL<span
125 class="red">(*)</span></label>
126                     </th>
127                     <td>
128                         <input type="text" name="is_Swt_demo_site"
129 id="is_Swt_demo_site" class="" />
130                     </td>
131                 </tr>
132                 <tr>
133                     <th>
134                         <label for="is_Giturl">Github URL<span
135 class="red">(*)</span></label>
136                     </th>
137                     <td>
138                         <input type="text" name="is_Giturl" id="is_Giturl"
139 class="" />
140                     </td>
141                 </tr>
142                 <tr>
143                     <th>
144                         <label for="is_Comments">설명<span
145 class="red">(*)</span></label>
146                     </th>
147                     <td>
148                         <textarea name="is_Comments" id="is_Comments" rows=""
149 cols=""></textarea>
150                     </td>
151                 </tr>
152                 <tr class="div_tb_tr fileb">
153                     <th>
154                         메뉴얼 파일 #1
155                     </th>
156                     <td class="fileBox fileBox_w1">
157                         <label for="uploadBtn1"
158 class="btn_file">파일선택</label>
159                         <input type="text" id="manualfile" class="fileName
160 fileName1" readonly="readonly" placeholder="선택된 파일 없음"/>
161                         <input type="file" id="uploadBtn1" class="uploadBtn
162 uploadBtn1" onChange={e => this.handleFileInput('manual',e)}/>
163                         <div id="upload_manual">
164                             </div>
165                     </td>
166                 </tr>
167                 <tr>
168                     <th>
169                         메인 이미지
170                     </th>
171                     <td className="fileBox fileBox1">

```

```

172                                     <label htmlFor='imageSelect'
173   className="btn_file">파일선택</label>
174                                     <input type="text" id="imagefile" className="fileName
175   fileName1" readOnly="readonly" placeholder="선택된 파일 없음"/>
176                                     <input type="file" id="imageSelect"
177   className="uploadBtn uploadBtn1" onChange={e => this.handleFileInput('file',e)}/>
178                                     <div id="upload_img">
179                                     </div>
180                                     </td>
181                                   </tr>
182                                   <tr>
183                                     <th>
184                                       라벨 이미지
185                                     </th>
186                                     <td className="fileBox fileBox2">
187                                       <label htmlFor='imageSelect2'
188   className="btn_file">파일선택</label>
189                                       <input type="text" id="imagefile2" className="fileName
190   fileName1" readOnly="readonly" placeholder="선택된 파일 없음"/>
191                                       <input type="file" id="imageSelect2"
192   className="uploadBtn uploadBtn1" onChange={e => this.handleFileInput('file2',e)}/>
193                                       <div id="upload_img2">
194                                       </div>
195                                       </td>
196                                     </tr>
197                                   <tr>
198                                     <th>
199                                       <label for="is_Swt_function">상세 기능<span
200   class="red">(*)</span></label>
201                                     </th>
202                                     <td>
203                                       <textarea name="is_Swt_function" id="is_Swt_function"
204   rows="" cols=""></textarea>
205                                     </td>
206                                   </tr>
207                                 </table>
208                                 <div class="btn_confirm mt20" style={{"margin-bottom": "44px"}}>
209                                   <Link to={'/SoftwareList'} className="bt_ty bt_ty1
210   cancel_ty1">취소</Link>
211                                   <a href="javascript:" className="bt_ty bt_ty2 submit_ty1
212   saveclass" onClick={(e) => this.submitClick('save', e)}>저장</a>
213                                   </div>
214                                 </div>
215                               </article>
216                             </form>
217                           </div>
218                         </article>
219                       </section>
220                     );
221                   }
222                 }
223
224   export default SoftwareView;

```

■ 실행 결과

React App

localhost:3000/SoftwareTools/register

NAUTUER

사용자 관리 Research Projects 관리 Software Tools 관리 Data Sources 관리 유동인구 조회 Sub code 관리

Software Tools 등록/수정

이름(*)

주소 (URL*)

GitHub URL(*)

설명(*)

메뉴얼 파일 #1

메뉴얼 이미지

파일 이미지

상세 기능(*)

취소 저장

개인정보처리방침 | 이용약관주소문의접거부

주소 : 서울시 구로구 장동 111 | TEL : 02-1234-5678

COPYRIGHT © React 200. ALL RIGHTS RESERVED.

131 REACT 상세 조회 페이지 만들기

- 리스트 페이지에는 전체 데이터 목록이 된다. 리스트에서 상세 데이터를 조회할 행의 [수정] 버튼을 누르면 상세 조회 페이지로 이동하도록 구현할 수 있다.
- 실행 결과

135 REACT 수정 페이지 만들기

- ```
[client/src/components/SoftwareToolsManage/SoftwareView.js]
```

24 / 61



```

06
07 class SoftwareView extends Component {
08 constructor(props) {
09 super(props);
10 this.state = {
11 before_swcode: props.match.params.swcode
12 }
13 }
14
15 componentDidMount () {
16 if(this.state.before_swcode == 'register'){
17 $('#.modifyclass').hide()
18 }else{
19 this.callSwToolInfoApi()
20 $('#.saveclass').hide()
21 }
22 }
23
24 callSwToolInfoApi = async () => {
25 axios.post('/api/Swtool?type=list', {
26 is_Swtcode: this.state.before_swcode,
27 })
28 .then(response => {
29 try {
30 var data = response.data.json[0]
31 $('#is_Swt_toolname').val(data.swt_toolname)
32 $('#is_Swt_demo_site').val(data.swt_demo_site)
33 $('#is_Giturl').val(data.swt_github_url)
34 $('#is_Comments').val(data.swt_comments)
35 $('#is_Swt_function').val(data.swt_function)
36 } catch (error) {
37 alert('작업중 오류가 발생하였습니다.')
38 }
39 })
40 .catch(error => {alert('작업중 오류가 발생하였습니다.');"return false; });
41 }
42
43 submitClick = async (type, e) => {
44
45 this.Swt_toolname_checker = $('#is_Swt_toolname').val();
46 this.Swt_demo_site_checker = $('#is_Swt_demo_site').val();
47 this.Giturl_checker = $('#is_Giturl').val();
48 this.Comments_checker = $('#is_Comments').val();
49 this.Swt_function_checker = $('#is_Swt_function').val();
50
51 this.fnValidate = (e) => {
52 if(this.Swt_toolname_checker === '') {
53 $('#is_Swt_toolname').addClass('border_validate_err');
54 alert('툴 이름을 다시 확인해주세요.')
55 return false;
56 }
57 $('#is_Swt_toolname').removeClass('border_validate_err');
58
59 if(this.Swt_demo_site_checker === '') {
60 $('#is_Swt_demo_site').addClass('border_validate_err');
61 alert('데모 URL을 다시 확인해주세요.')
62 return false;
63 }
64 $('#is_Swt_demo_site').removeClass('border_validate_err');
65
66 if(this.Giturl_checker === '') {
67 $('#is_Giturl').addClass('border_validate_err');
68 alert('Github URL을 다시 확인해주세요.')
69 return false;
70 }

```

```

71 $('#is_Giturl').removeClass('border_validate_err');
72
73 if(this.Comments_checker === '') {
74 $('#is_Comments').addClass('border_validate_err');
75 alert('설명을 다시 확인해주세요.')
76 return false;
77 }
78 $('#is_Comments').removeClass('border_validate_err');
79
80 if(this.Swt_function_checker === '') {
81 $('#is_Swt_function').addClass('border_validate_err');
82 alert('상세기능을 다시 확인해주세요.')
83 return false;
84 }
85 $('#is_Swt_function').removeClass('border_validate_err');
86 return true;
87 }
88
89 if(this.fnValidate()){
90 var jsonstr = $("form[name='frm']").serialize();
91 jsonstr = decodeURIComponent(jsonstr);
92 var Json_form = JSON.stringify(jsonstr).replace(/\\"/gi, '')
93 Json_form = "{\"" + Json_form.replace(/\&/g, '\&').replace(/=/gi, '\="')+\""}";
94
95 try {
96 const response = await fetch('/api/Swttool?type='+type, {
97 method: 'POST',
98 headers: {
99 'Content-Type': 'application/json',
100 },
101 body: Json_form,
102 });
103 const body = await response.text();
104 if(body == "succ"){
105 if(type == 'save'){
106 this.sweetalertSucc('Software Tools 등록이 완료되었습니다.', false)
107 }else if(type == "modify"){
108 this.sweetalertSucc('Software Tools 수정이 완료되었습니다.', false)
109 }
110 setTimeout(function() {
111 this.props.history.push('/SoftwareList');
112 }.bind(this),1500
113);
114 }else{
115 alert('작업중 오류가 발생하였습니다.')
116 }
117 } catch (error) {
118 alert('작업중 오류가 발생하였습니다.')
119 }
120 }
121 };
122
123 sweetalertSucc = (title, showConfirmButton) => {
124 Swal.fire({
125 position: 'bottom-end',
126 icon: 'success',
127 title: title,
128 showConfirmButton: showConfirmButton,
129 timer: 1000
130 })
131 }
132
133 render () {
134 return (
135 <section class="sub_wrap">

```

```

136 <article class="s_cnt mp_pro_li ct1">
137 <div class="li_top">
138 <h2 class="s_tit1">Software Tools 등록/수정</h2>
139 </div>
140 <div class="bo_w re1_wrap re1_wrap_writer">
141 <form name="frm" id="frm" action="" onsubmit="" method="post" >
142 <input id="is_Swtcode" type="hidden" name="is_Swtcode" />
143 <input id="is_Email" type="hidden" name="is_Email" value="guest" />
144 <input id="is_beforeSwtcode" type="hidden" name="is_beforeSwtcode"
145 value={this.state.before_swtcode} />
146 <article class="res_w">
147 <p class="ment" style={{ "text-align": "right" }}>
148 (*)표시는 필수입력사항 입니다.
149 </p>
150 <div class="tb_outline">
151 <table class="table_ty1">
152 <tr>
153 <th>
154 <label for="is_Swt_toolname">툴 이름(*)</label>
156 </th>
157 <td>
158 <input type="text" name="is_Swt_toolname"
159 id="is_Swt_toolname" class="" />
160 </td>
161 </tr>
162 <tr>
163 <th>
164 <label for="is_Swt_demo_site">데모 URL(*)</label>
166 </th>
167 <td>
168 <input type="text" name="is_Swt_demo_site"
169 id="is_Swt_demo_site" class="" />
170 </td>
171 </tr>
172 <tr>
173 <th>
174 <label for="is_Giturl">Github URL(*)</label>
176 </th>
177 <td>
178 <input type="text" name="is_Giturl" id="is_Giturl"
179 class="" />
180 </td>
181 </tr>
182 <tr>
183 <th>
184 <label for="is_Comments">설명(*)</label>
186 </th>
187 <td>
188 <textarea name="is_Comments" id="is_Comments" rows=""
189 cols=""></textarea>
190 </td>
191 </tr>
192 <tr class="div_tb_tr fileb">
193 <th>
194 메뉴얼 파일 #1
195 </th>
196 <td class="fileBox fileBox_w1">
197 <label for="uploadBtn1"
198 class="btn_file">파일선택</label>
199 <input type="text" id="manualfile" class="fileName
200 fileName1" readonly="readonly" placeholder="선택된 파일 없음"/>

```

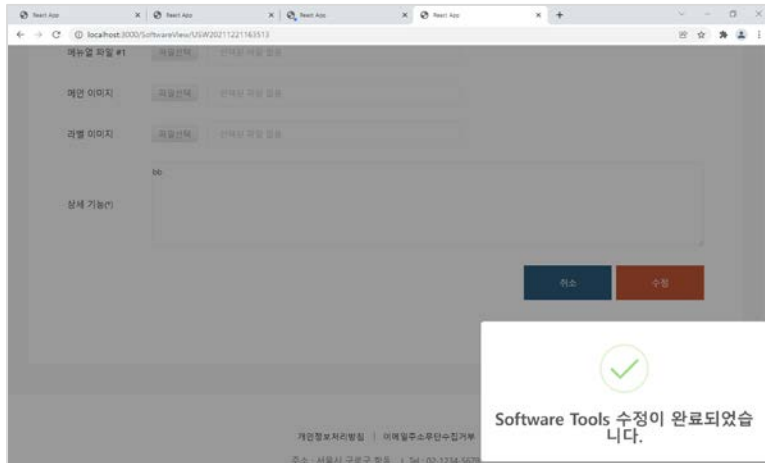
```

201 <input type="file" id="uploadBtn1" class="uploadBtn
202 uploadBtn1" onChange={e => this.handleFileInput('manual',e)}>
203 <div id="upload_manual">
204 </div>
205 </td>
206 </tr>
207 <tr>
208 <th>
209 메인 이미지
210 </th>
211 <td className="fileBox fileBox1">
212 <label htmlFor='imageSelect'
213 className="btn_file">파일선택</label>
214 <input type="text" id="imagefile" className="fileName
215 fileName1" readOnly="readonly" placeholder="선택된 파일 없음"/>
216 <input type="file" id="imageSelect"
217 className="uploadBtn uploadBtn1" onChange={e => this.handleFileInput('file',e)}>
218 <div id="upload_img">
219 </div>
220 </td>
221 </tr>
222 <tr>
223 <th>
224 라벨 이미지
225 </th>
226 <td className="fileBox fileBox2">
227 <label htmlFor='imageSelect2'
228 className="btn_file">파일선택</label>
229 <input type="text" id="imagefile2" className="fileName
230 fileName1" readOnly="readonly" placeholder="선택된 파일 없음"/>
231 <input type="file" id="imageSelect2"
232 className="uploadBtn uploadBtn1" onChange={e => this.handleFileInput('file2',e)}>
233 <div id="upload_img2">
234 </div>
235 </td>
236 </tr>
237 <tr>
238 <th>
239 <label for="is_Swt_function">상세 기능(*)</label>
241 </th>
242 <td>
243 <textarea name="is_Swt_function" id="is_Swt_function"
244 rows="" cols=""></textarea>
245 </td>
246 </tr>
247 </table>
248 <div class="btn_confirm mt20" style={{"margin-bottom": "44px"}}>
249 <Link to={'/SoftwareList'} className="bt_ty bt_ty1
250 cancel_ty1">취소</Link>
251 <a href="javascript:" className="bt_ty bt_ty2 submit_ty1
252 saveclass"
253 onClick={(e) => this.submitClick('save', e)}>저장
254 <a href="javascript:" className="bt_ty bt_ty2 submit_ty1
255 modifyclass"
256 onClick={(e) => this.submitClick('modify', e)}>수정
257 </div>
258 </div>
259 </article>
260 </form>
261 </div>
262 </article>
263 </section>
264);
265 }

```

```
266 }
267
268 export default SoftwareView;
```

## ■ 실행 결과



## 136 NODE 삭제 api 만들기 1

## 137 NODE 삭제 api 만들기 2

## 138 REACT 리스트 페이지 삭제 기능 구현하기

- react 컴포넌트에서 post 방식으로 axios를 사용해 node api를 호출한다. delete 쿼리의 조건절에 사용할 변수를 json 형태를 전달하고 쿼리가 정상적으로 실행됐다면 리스트 페이지 데이터를 다시 불러온다.

## ■ SoftwareList.js 파일

[client/src/components/SoftwareToolsManage/SoftwareList.js]

```
01 import React, { Component } from 'react';
02 import { Link } from 'react-router-dom';
03 import axios from "axios";
04 import Swal from 'sweetalert2'
05
06 class SoftwareList extends Component {
07 constructor(props) {
08 super(props);
09
10 this.state = {
11 responseSwtoolList: '',
12 append_SwtoolList: '',
13 }
14 }
15
16 componentDidMount() {
17 this.callSwToolListApi()
18 }
19 }
```

```

20 callSwToolListApi = async () => {
21 axios.post('/api/Swtool?type=list', {
22 })
23 .then(response => {
24 try {
25 this.setState({ responseSwtoolList: response });
26 this.setState({ append_SwtoolList: this.SwToolListAppend() });
27 } catch (error) {
28 alert('작업중 오류가 발생하였습니다.');
```

```

29 }
30 })
31 .catch(error => {alert('작업중 오류가 발생하였습니다.');
```

```

32 }
33
34 SwToolListAppend = () => {
35 let result = []
36 var SwToolList = this.state.responseSwtoolList.data
37
38 for(let i=0; i<SwToolList.json.length; i++){
39 var data = SwToolList.json[i]
40
41 var date = data.reg_date
42 var year = date.substr(0,4)
43 var month = date.substr(4,2)
44 var day = date.substr(6,2)
45 var reg_date = year + '.'+month+'.'+day
46
47 result.push(
48 <tr class="hidden_type">
49 <td>{data.swt_toolname}</td>
50 <td>{data.swt_function}</td>
51 <td>{reg_date}</td>
52 <td>
53 <Link to={'/SoftwareView/'+data.swt_code}
54 className="bt_c1 bt_c2 w50_b">수정</Link>
55 <a href="#n" class="bt_c1 w50_b" id={data.swt_code}
56 onClick={e => this.deleteSwtool(e)}>삭제
57 </td>
58 </tr>
59)
60 }
61 return result
62 }
63
64 deleteSwtool = (e) => {
65 var event_target = e.target
66 this.sweetalertDelete('정말 삭제하시겠습니까?', function() {
67 axios.post('/api/Swtool?type=delete', {
68 is_SwtCd : event_target.getAttribute('id')
69 })
70 .then(response => {
71 this.callSwToolListApi()
72 }).catch(error => {alert('작업중 오류가 발생하였습니다.');
```

```

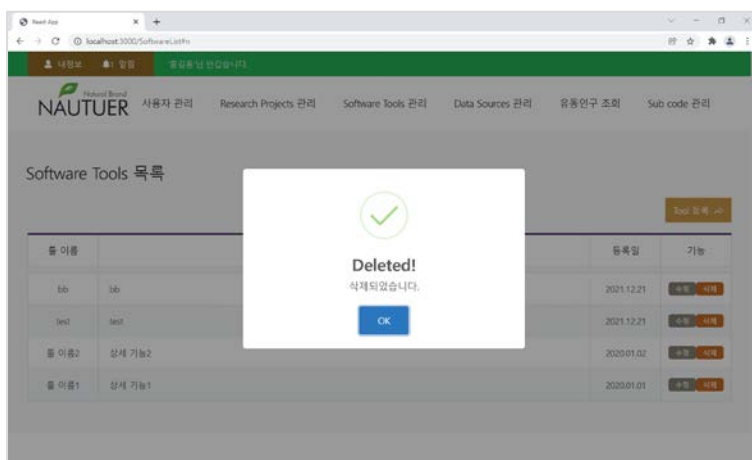
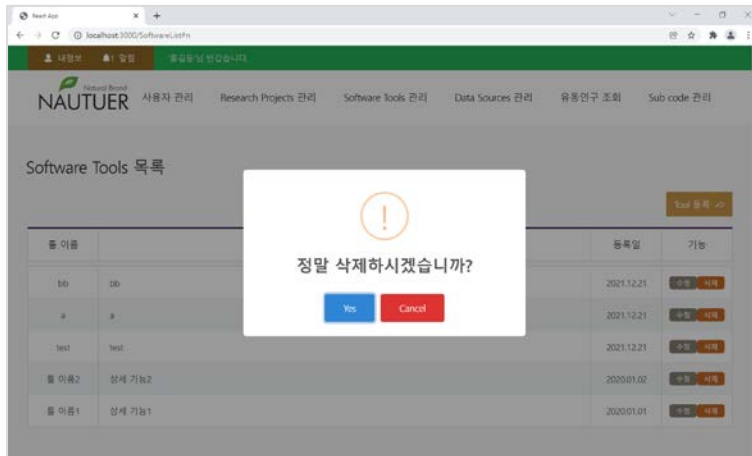
73 }.bind(this))
74 }
75
76 sweetalertDelete = (title, callbackFunc) => {
77 Swal.fire({
78 title: title,
79 text: "",
80 icon: 'warning',
81 showCancelButton: true,
82 confirmButtonColor: '#3085d6',
83 cancelButtonColor: '#d33',
84 confirmButtonText: 'Yes'
```

```

85 }).then((result) => {
86 if (result.value) {
87 Swal.fire(
88 'Deleted!',
89 '삭제되었습니다.',
90 'success'
91)
92 }else{
93 return false;
94 }
95 callbackFunc()
96 })
97 }
98
99 render () {
100 return (
101 <section class="sub_wrap" >
102 <article class="s_cnt mp_pro_li ct1 mp_pro_li_admin">
103 <div class="li_top">
104 <h2 class="s_tit1">Software Tools 목록</h2>
105 <div class="li_top_sch af">
106 <Link to={'/SoftwareView/register'} className="sch_bt2 wi_au">Tool 등록</Link>
107 </div>
108 </div>
109
110 <div class="list_cont list_cont_admin">
111 <table class="table_ty1 ad_tlist">
112 <tr>
113 <th>툴 이름</th>
114 <th>기능</th>
115 <th>등록일</th>
116 <th>기능</th>
117 </tr>
118 </table>
119 <table class="table_ty2 ad_tlist">
120 {this.state.append_SwtoolList}
121 </table>
122 </div>
123 </article>
124 </section>
125);
126 }
127 }
128
129 export default SoftwareList;

```

## ■ 실행 결과



## 139 파일, 이미지 업로드 api 만들기

- 문서나 이미지와 같은 파일을 서버 경로에 업로드하기 위해 multer라는 패키지를 사용한다.

## 140 REACT 등록 페이지 만들기 - 파일, 이미지 업로드 api 호출하기

- react 페이지에서 선택한 파일을 node api를 통해 node 경로에 업로드한다. 파일이 정상적으로 업로드됐다면 파일명과 미리보기 이미지를 표시한다.
- SoftwareView.js 파일

[client/src/components/SoftwareToolsManage/SoftwareView.js]

```
01 import React, { Component } from 'react';
02 import { Link } from 'react-router-dom';
03 import axios from "axios";
04 import $ from 'jquery';
05 import Swal from 'sweetalert2'
06
07 class SoftwareView extends Component {
08 constructor(props) {
09 super(props);
10 this.state = {
```



```

11 before_swcode: props.match.params.swcode,
12 selectedFile: null,
13 }
14 }
15
16 componentDidMount () {
17 if(this.state.before_swcode == 'register'){
18 $('.modifyclass').hide()
19 }else{
20 this.callSwToolInfoApi()
21 $('.saveclass').hide()
22 }
23 }
24
25 callSwToolInfoApi = async () => {
26 axios.post('/api/Swtool?type=list', {
27 is_Swtcode: this.state.before_swcode,
28 })
29 .then(response => {
30 try {
31 var data = response.data.json[0]
32 $('#is_Swt_toolname').val(data.swt_toolname)
33 $('#is_Swt_demo_site').val(data.swt_demo_site)
34 $('#is_Giturl').val(data.swt_github_url)
35 $('#is_Comments').val(data.swt_comments)
36 $('#is_Swt_function').val(data.swt_function)
37 var manualName = data.swt_manual_path.replace('/swmanual/', '')
38 var fileName = data.swt_big_imgpath.replace('/image/', '')
39 var fileName2 = data.swt_imagepath.replace('/image/', '')
40 $('#upload_img').prepend('')
41 $('#upload_img2').prepend('')
42
43 $('#imagefile').val(fileName)
44 $('#imagefile2').val(fileName2)
45 $('#manualfile').val(manualName)
46
47 if($('#uploadimg').attr('src').indexOf("null") > -1){
48 $('#uploadimg').hide()
49 }
50 if($('#uploadimg2').attr('src').indexOf("null") > -1){
51 $('#uploadimg2').hide()
52 }
53 } catch (error) {
54 alert('작업중 오류가 발생하였습니다.')
55 }
56 })
57 .catch(error => {alert('작업중 오류가 발생하였습니다.');
```

```

76 if(this.Swt_demo_site_checker === '') {
77 $('#is_Swt_demo_site').addClass('border_validate_err');
78 alert('데모 URL을 다시 확인해주세요.')
79 return false;
80 }
81 $('#is_Swt_demo_site').removeClass('border_validate_err');
82
83 if(this.Giturl_checker === '') {
84 $('#is_Giturl').addClass('border_validate_err');
85 alert('Github URL을 다시 확인해주세요.')
86 return false;
87 }
88 $('#is_Giturl').removeClass('border_validate_err');
89
90 if(this.Comments_checker === '') {
91 $('#is_Comments').addClass('border_validate_err');
92 alert('설명을 다시 확인해주세요.')
93 return false;
94 }
95 $('#is_Comments').removeClass('border_validate_err');
96
97 if(this.Swt_function_checker === '') {
98 $('#is_Swt_function').addClass('border_validate_err');
99 alert('상세기능을 다시 확인해주세요.')
100 return false;
101 }
102 $('#is_Swt_function').removeClass('border_validate_err');
103 return true;
104 }
105
106 if(this.fnValidate()){
107 var jsonstr = $("form[name='frm']").serialize();
108 jsonstr = decodeURIComponent(jsonstr);
109 var Json_form = JSON.stringify(jsonstr).replace(/\\"/gi, '')
110 Json_form = "{\\" + Json_form.replace(/\&/g, '\&').replace(/=/gi, '\="')+\"}";
111
112 try {
113 const response = await fetch('/api/Swtool?type='+type, {
114 method: 'POST',
115 headers: {
116 'Content-Type': 'application/json',
117 },
118 body: Json_form,
119 });
120 const body = await response.text();
121 if(body == "succ"){
122 if(type == 'save'){
123 this.sweetalertSucc('Software Tools 등록이 완료되었습니다.', false)
124 }else if(type == "modify"){
125 this.sweetalertSucc('Software Tools 수정이 완료되었습니다.', false)
126 }
127 setTimeout(function() {
128 this.props.history.push('/SoftwareList');
129 }.bind(this),1500
130);
131 }else{
132 alert('작업중 오류가 발생하였습니다.')
133 }
134 } catch (error) {
135 alert('작업중 오류가 발생하였습니다.')
136 }
137 }
138 };
139
140 sweetalertSucc = (title, showConfirmButton) => {

```

```

141 Swal.fire({
142 position: 'bottom-end',
143 icon: 'success',
144 title: title,
145 showConfirmButton: showConfirmButton,
146 timer: 1000
147 })
148 }
149
150 handleFileInput(type, e){
151 if(type == 'file'){
152 $('#imagefile').val(e.target.files[0].name)
153 }else if(type == 'file2'){
154 $('#imagefile2').val(e.target.files[0].name)
155 }else if(type == 'manual'){
156 $('#manualfile').val(e.target.files[0].name)
157 }
158 this.setState({
159 selectedFile : e.target.files[0],
160 })
161 setTimeout(function() {
162 if(type == 'manual'){
163 this.handlePostManual()
164 }else{
165 this.handlePostImage(type)
166 }
167 }.bind(this),1
168);
169 }
170
171 handlePostManual(){
172 const formData = new FormData();
173 formData.append('file', this.state.selectedFile);
174 return axios.post("/api/upload?type=uploads/swmanual/", formData).then(res => {
175 this.setState({manualName : res.data.filename})
176 $('#is_ManualName').remove()
177 $('#upload_manual').prepend('<input id="is_ManualName" type="hidden"'
178 + 'name="is_ManualName" value="/swmanual/'+this.state.manualName+'"/>')
179 }).catch(error => {
180 alert('작업중 오류가 발생하였습니다.', error, 'error', '닫기')
181 })
182 }
183
184 handlePostImage(type){
185 const formData = new FormData();
186 formData.append('file', this.state.selectedFile);
187 return axios.post("/api/upload?type=uploads/image/", formData).then(res => {
188 if(type == 'file'){
189 this.setState({fileName : res.data.filename})
190 $('#is_MainImg').remove()
191 $('#uploading').remove()
192 $('#upload_img').prepend('')
194 $('#upload_img').prepend('<input id="is_MainImg" type="hidden"'
195 + 'name="is_MainImg" value="/image/'+this.state.fileName+'"/>')
196 }else if(type == 'file2'){
197 this.setState({fileName2 : res.data.filename})
198 $('#is_LabelImg').remove()
199 $('#uploading2').remove()
200 $('#upload_img2').prepend('')
202 $('#upload_img2').prepend('<input id="is_LabelImg" type="hidden"'
203 + 'name="is_LabelImg" value="/image/'+this.state.fileName2+'"/>')
204 }
205 }).catch(error => {

```

```

206 alert('작업중 오류가 발생하였습니다.')
```

```

207 })
```

```

208 }
```

```

209
```

```

210 render () {
```

```

211 return (
```

```

212 <section class="sub_wrap">
```

```

213 <article class="s_cnt mp_pro_li ct1">
```

```

214 <div class="li_top">
```

```

215 <h2 class="s_tit1">Software Tools 등록/수정</h2>
```

```

216 </div>
```

```

217 <div class="bo_w re1_wrap re1_wrap_writer">
```

```

218 <form name="frm" id="frm" action="" onsubmit="" method="post" >
```

```

219 <input id="is_Swtcode" type="hidden" name="is_Swtcode" />
```

```

220 <input id="is_Email" type="hidden" name="is_Email" value="guest" />
```

```

221 <input id="is_beforeSwtcode" type="hidden" name="is_beforeSwtcode"
```

```

222 value={this.state.before_swtcode} />
```

```

223 <article class="res_w">
```

```

224 <p class="ment" style={{ "text-align": "right" }}>
```

```

225 (*)표시는 필수입력사항 입니다.
```

```

226 </p>
```

```

227 <div class="tb_outline">
```

```

228 <table class="table_ty1">
```

```

229 <tr>
```

```

230 <th>
```

```

231 <label for="is_Swt_toolname">툴 이름<span
```

```

232 class="red">(*)</label>
```

```

233 </th>
```

```

234 <td>
```

```

235 <input type="text" name="is_Swt_toolname"
```

```

236 id="is_Swt_toolname" class="" />
```

```

237 </td>
```

```

238 </tr>
```

```

239 <tr>
```

```

240 <th>
```

```

241 <label for="is_Swt_demo_site">데모 URL<span
```

```

242 class="red">(*)</label>
```

```

243 </th>
```

```

244 <td>
```

```

245 <input type="text" name="is_Swt_demo_site"
```

```

246 id="is_Swt_demo_site" class="" />
```

```

247 </td>
```

```

248 </tr>
```

```

249 <tr>
```

```

250 <th>
```

```

251 <label for="is_Giturl">Github URL<span
```

```

252 class="red">(*)</label>
```

```

253 </th>
```

```

254 <td>
```

```

255 <input type="text" name="is_Giturl" id="is_Giturl"
```

```

256 class="" />
```

```

257 </td>
```

```

258 </tr>
```

```

259 <tr>
```

```

260 <th>
```

```

261 <label for="is_Comments">설명<span
```

```

262 class="red">(*)</label>
```

```

263 </th>
```

```

264 <td>
```

```

265 <textarea name="is_Comments" id="is_Comments" rows=""
```

```

266 cols=""></textarea>
```

```

267 </td>
```

```

268 </tr>
```

```

 <tr class="div_tb_tr fileb">
```

```

 <th>
```

```

 메뉴얼 파일 #1
 </th>
 <td class="fileBox fileBox_w1">
 <label for="uploadBtn1"

class="btn_file">파일 선택</label>

 <input type="text" id="manualfile" class="fileName
fileName1"
 readonly="readonly" placeholder="선택된 파일 없음"/>
 <input type="file" id="uploadBtn1" class="uploadBtn
uploadBtn1"
 onChange={e => this.handleFileInput('manual',e)}>
 <div id="upload_manual">
 </div>
 </td>
</tr>
<tr>
 <th>
 메인 이미지
 </th>
 <td className="fileBox fileBox1">
 <label htmlFor='imageSelect'

className="btn_file">파일 선택</label>

 <input type="text" id="imagefile" className="fileName
fileName1"
 readOnly="readonly" placeholder="선택된 파일 없음"/>
 <input type="file" id="imageSelect"
className="uploadBtn uploadBtn1"
 onChange={e => this.handleFileInput('file',e)}>
 <div id="upload_img">
 </div>
 </td>
</tr>
<tr>
 <th>
 라벨 이미지
 </th>
 <td className="fileBox fileBox2">
 <label htmlFor='imageSelect2'

className="btn_file">파일 선택</label>

 <input type="text" id="imagefile2" className="fileName
fileName1"
 readOnly="readonly" placeholder="선택된 파일 없음"/>
 <input type="file" id="imageSelect2"
className="uploadBtn uploadBtn1"
 onChange={e => this.handleFileInput('file2',e)}>
 <div id="upload_img2">
 </div>
 </td>
</tr>
<tr>
 <th>
 <label for="is_Swt_function">상세 기능<span

class="red">(*)</label>

 </th>
 <td>
 <textarea name="is_Swt_function" id="is_Swt_function"

rows="" cols=""></textarea>

 </td>
</tr>
</table>
<div class="btn_confirm mt20" style={{"margin-bottom": "44px"}}>
 <Link to={'/SoftwareList'} className="bt_ty bt_ty1

cancel_ty1">취소</Link>

 <a href="javascript:" className="bt_ty bt_ty2 submit_ty1

saveclass"

```

```

 onClick={e => this.submitClick('save', e)}>저장
 <a href="javascript:" className="bt_ty bt_ty2 submit_ty1
modifyclass"
 onClick={e => this.submitClick('modify', e)}>수정
 </div>
 </div>
 </article>
 </form>
 </div>
 </article>
 </section>
);
 }
 }

 export default SoftwareView;

```

141 REACT 등록 페이지 만들기 - 업로드 경로를 DB에 insert하기

142 REACT 상세 조회 페이지 만들기 - 파일, 이미지명 표시하기

143 REACT 수정 페이지 만들기 - 파일, 이미지 업로드 api 호출하기

144 REACT 수정 페이지 만들기 - 업로드 경로 DB에 업데이트하기

#### ■ SoftwareView.js 파일

[client/src/components/SoftwareToolsManage/SoftwareView.js]

```

1 import React, { Component } from 'react';
2 import { Link } from 'react-router-dom';
3 import axios from "axios";
4 import $ from 'jquery';
5 import Swal from 'sweetalert2'
6
7 class SoftwareView extends Component {
8 constructor(props) {
9 super(props);
10 this.state = {
11 before_swtcode: props.match.params.swtcode,
12 selectedFile: null,
13 }
14 }
15
16 componentDidMount () {
17 if(this.state.before_swtcode == 'register'){
18 $('.modifyclass').hide()
19 }else{
20 this.callSwToolInfoApi()
21 $('.saveclass').hide()
22 }
23 }
24
25 callSwToolInfoApi = async () => {
26 axios.post('/api/Swtool?type=list', {
27 is_Swtcode: this.state.before_swtcode,

```

```

28 })
29 .then(response => {
30 try {
31 var data = response.data.json[0]
32 $('#is_Swt_toolname').val(data.swt_toolname)
33 $('#is_Swt_demo_site').val(data.swt_demo_site)
34 $('#is_Giturl').val(data.swt_github_url)
35 $('#is_Comments').val(data.swt_comments)
36 $('#is_Swt_function').val(data.swt_function)
37
38 var manualName, fileName, fileName2;
39 if (data.swt_manual_path != null) manualName =
40 data.swt_manual_path.replace('/swmanual/', '')
41 if (data.swt_big_imgpath != null) fileName = data.swt_big_imgpath.replace('/image/', '')
42 if (data.swt_imagepath != null) fileName2 = data.swt_imagepath.replace('/image/', '');
43 $('#upload_img').prepend('')
44 $('#upload_img2').prepend('')
45
46 $('#imagefile').val(fileName)
47 $('#imagefile2').val(fileName2)
48 $('#manualfile').val(manualName)
49
50 if($('#uploadimg').attr('src').indexOf("null") > -1){
51 $('#uploadimg').hide()
52 }
53 if($('#uploadimg2').attr('src').indexOf("null") > -1){
54 $('#uploadimg2').hide()
55 }
56 } catch (error) {
57 alert(error);
58 alert('작업중 오류가 발생하였습니다.')
59 }
60 })
61 .catch(error => {alert('작업중 오류가 발생하였습니다.');
```

## ■ 실행 결과

## 리액트 (프론트엔드 개발)

Test App

Test App

Test - Google Search

Google - Google Search

Google - Google Search

[localhost:3000/Software/View/5/W20211221193557](#)

🏠

🔍

🌟

🔖

내정보

알림

회원탈퇴 및 비밀번호변경

NAUTUER

Natural Brand

사용자 관리

Research Projects 관리

Software Tools 관리

Data Sources 관리

유동연구 조회

Sub code 관리

Software Tools 등록/수정

기타사항은 필수입력사항입니다.

툴 이름(\*)

vscode

대포 URL(\*)

https://code.visualstudio.com/

Github URL(\*)

https://github.com/vscode/vscode

설명(\*)

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Wikipedia

예시 이미지 #1

복합선택

20211221193552\_8.jpg

복합선택

20211221193552\_8.jpg

예시 이미지 #2

복합선택

20211221193552\_vscode.gif

복합선택

20211221193552\_vscode.gif

라벨 이미지

EXPLORER

16 styles.js

128

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

50



## 145 회원 정보 table 생성하기

```
-- react 계정으로 작업한다.
use react;
CREATE TABLE `react_user` (
 `username` varchar(100) DEFAULT NULL COMMENT '사용자 이름',
 `userorg` varchar(100) DEFAULT NULL COMMENT '소속기관',
 `useremail` varchar(100) COMMENT '이메일',
 `userpassword` varchar(100) DEFAULT NULL COMMENT '로그인 비밀번호',
 `usermajor` varchar(100) DEFAULT NULL COMMENT '전공',
 `userphone` varchar(100) DEFAULT NULL COMMENT '휴대전화번호',
 `userflag` varchar(100) DEFAULT NULL COMMENT '승인여부',
 `reg_date` varchar(100) DEFAULT NULL COMMENT '등록날짜',
 `reg_user` varchar(100) DEFAULT NULL COMMENT '등록자',
 `update_date` varchar(100) DEFAULT NULL COMMENT '수정날짜',
 `update_user` varchar(100) DEFAULT NULL COMMENT '수정자',
 PRIMARY KEY (`useremail`)
);
ALTER TABLE react.react_user convert to charset utf8;
```

## 146 NODE 회원 가입 api 만들기 1

- 회원 가입을 하면 다양한 데이터를 입력받게 된다. 이 중 로그인 비밀번호를 그대로 DB에 넣어 보관하는 것은 매우 위험하다. 비밀번호는 복호화할 필요가 없기 때문에 해시 함수로 암호화한다. 해시 알고리즘은 암호화만 가능하고 복호화는 불가능한데 bcrypt 패키지를 사용하면 해시 암호화를 간편하게 구현할 수 있다.

```
// bcrypt를 설치한다.
D:\dev\workspace\react\react200-lab\146>yarn add bcrypt
```

- UsersRoute.js 파일
  - 05~06: bcrypt 패키지를 require해 사용할 수 있도록 한다. bcrypt에서 지원하는 salt의 크기를 10으로 설정한다. salt는 해커가 비밀번호를 유추하기 어렵도록 한 번 더 암호화하는 문자열이다. bcrypt 패키지 내부적으로 salt의 크기만큼의 임의의 문자열을 생성한다.
  - 20~22: type 파라미터의 값이 signup일 경우, 회원 정보를 insert하는 mapper 정보를 req.body에 할당한다.
  - 24~25: req.body.is\_Password는 암호화되지 않은 비밀번호다. 값이 있다면 bcrypt 암호화를 실행한다.
  - 26: genSalt 함수는 line 6에서 지정한 크기(10)의 salt를 생성한다.
  - 27: hash 함수로 salt 값을 사용해 암호화되지 않은 비밀번호(myPlaintextPassword)를 암호화한다. 암호화된 비밀번호가 콜백 함수에서 hash 변수로 반환된다.
  - 28: 암호화된 비밀번호(hash)를 다시 req.body.is\_Password 변수에 할당해 다음 라우터(dbconnect\_Module)로 전달한다.

[routes/UsersRoute.js]

```
01 var express = require('express');
02 var router = express.Router();
03 const bodyParser = require('body-parser');
04
05 const bcrypt = require('bcrypt');
```

```

06 const saltRounds = 10;
07
08 router.use(bodyParser.json());
09 router.use(bodyParser.urlencoded({ extended: true }));
10
11 router.post('/', (req, res, next) => {
12 var type = req.query.type;
13 if(type == "signup"){
14 //회원가입 정보 삽입
15 try {
16 // Mysql Api 모듈(CRUD)
17 var dbconnect_Module = require('./dbconnect_Module');
18
19 //Mysql 쿼리 호출정보 입력
20 req.body.mapper = 'UserMapper';//mybatis xml 파일명
21 req.body.crud = 'insert';//select, insert, update, delete 중에 입력
22 req.body.mapper_id = 'insertUser';
23
24 var myPlaintextPassword = req.body.is_Password;
25 if(myPlaintextPassword != '' && myPlaintextPassword != undefined){
26 bcrypt.genSalt(saltRounds, function(err, salt) {
27 bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {
28 req.body.is_Password = hash;
29 router.use('/', dbconnect_Module);
30 next('route')
31 });
32 });
33 }else{
34 router.use('/', dbconnect_Module);
35 next('route')
36 }
37 } catch (error) {
38 console.log("Module > dbconnect error : "+ error);
39 }
40 }else if(type == "dplicheck"){
41 //이메일 중복체크
42 try {
43 // Mysql Api 모듈(CRUD)
44 var dbconnect_Module = require('./dbconnect_Module');
45
46 //Mysql 쿼리 호출정보 입력
47 req.body.mapper = 'UserMapper';//mybatis xml 파일명
48 req.body.crud = 'select';//select, insert, update, delete 중에 입력
49 req.body.mapper_id = 'selectUserDpliCheck';
50 router.use('/', dbconnect_Module);
51 next('route')
52 } catch (error) {
53 console.log("Module > dbconnect error : "+ error);
54 }
55 }
56 });
57
58 module.exports = router;

```

## 147 NODE 회원 가입 api 만들기 2 - insert 쿼리 추가하기

- 회원 가입 페이지에서 나눠 전달한 휴대전화 번호와 이메일 주소를 mysql CONCAT 함수를 사용해 가공한 후 삽입한다.

## 148 REACT 회원 가입 페이지 만들기 1 - 입력 form 만들기

- 회원 정보를 저장하는 node api 호출 시 전달할 변수들을 <form> 태그 안에 위치시킨다. <form> 태그로 감싸진 <input> 태그와 <select> 태그의 데이터는 별도의 변수 선언 없이 편리하게 파라미터로 전달할 수 있다.
- Register.js
  - 5~6: pattern1은 숫자를 의미한다. 입력된 글자가 숫자가 아니라면 가장 마지막에 추가된 문자를 제외하고 <input> 태그에 남긴다. 예를 들어, 1a를 입력했다면 1만 남게 된다.
  - 17~20: 이메일의 @ 앞부분을 <input> 태그에 입력한다. 유효성 체크에서 input 값을 입력하지 않고 [회원 가입] 버튼을 누르면 <input> 태그에 빨간색 테두리가 생긴다. emailKeyPress 함수는 입력 창에 키를 입력했을 때 실행되고 빨간색 테두리를 제거한다.
  - 97~105: <input> 태그에 max와 maxlength 속성을 추가하면 숫자의 입력 범위를 지정할 수 있다. Max 값이 9999이면 0000부터 9999까지의 숫자를 입력할 수 있다.

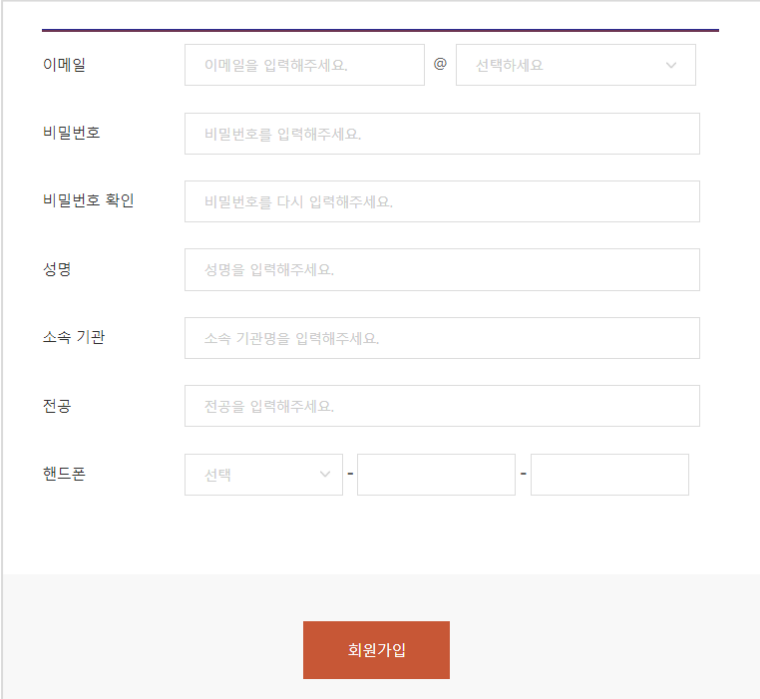
[client/src/components/Register/Register.js]

```

1 ...(<생략>)...
2 mustNumber = (id) => {
3 var pattern1 = /[0-9]/;
4 var str = $('#'+id).val();
5 if(!pattern1.test(str.substr(str.length - 1, 1))){
6 $('#'+id).val(str.substr(0, str.length-1));
7 }
8 }
9 ...(<생략>)...
10
11 <form method="post" name="frm">
12 <div className="re1_wrap">
13 <div className="re_cnt ct2">
14 <table className="table_ty1">
15 <tr className="re_email">
16 <th>이메일</th>
17 <td>
18 <input id="email_val" type="text"
19 name="is_Useremail1"
20 placeholder="이메일을 입력해주세요."
21 onPress={this.emailKeyPress}/>
22 @
23 <select id="email2_val" name="is_Useremail2"
24 className="select_ty1">
25 <option value="">선택하세요</option>
26 <option
27 value='naver.com'>naver.com</option>
28 <option
29 value='hanmail.net'>hanmail.net</option>
30 <option value='nate.com'>nate.com</option>
31 <option
32 value='hotmail.com'>hotmail.com</option>
33 <option
34 value='gmail.com'>gmail.com</option>
35 <option
36 value='yahoo.co.kr'>yahoo.co.kr</option>
37 <option
38 value='yahoo.com'>yahoo.com</option>
39 </select>
40 </td>

```

- 실행 결과



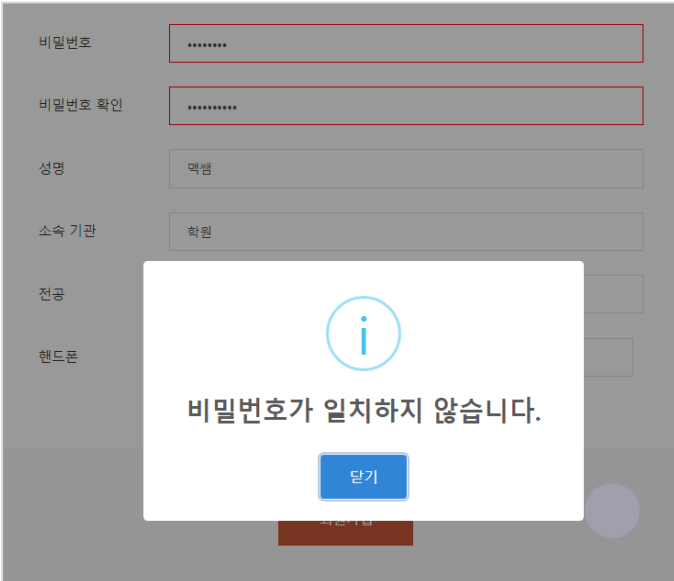
A user registration form with the following fields:

- 이메일**: Email input field with placeholder "이메일을 입력해주세요." and a dropdown menu for domain selection with placeholder "선택하세요".
- 비밀번호**: Password input field with placeholder "비밀번호를 입력해주세요."
- 비밀번호 확인**: Confirm password input field with placeholder "비밀번호를 다시 입력해주세요."
- 성명**: Full name input field with placeholder "성명을 입력해주세요."
- 소속 기관**: Affiliation input field with placeholder "소속 기관명을 입력해주세요."
- 전공**: Major input field with placeholder "전공을 입력해주세요."
- 핸드폰**: Phone number input field with a dropdown for country code, and two input boxes for the number.

At the bottom, there is a red button labeled "회원가입" (Sign Up).

## 149 REACT 회원 가입 페이지 만들기 2 - 유효성 체크하기

- DB에 정합성 높은 데이터를 삽입하기 위해서 node api 호출 전 입력 받은 값이 유효한지 검사한다. 하나의 입력 값이라도 잘못된 데이터가 들어왔다면 node api 호출을 하지 않고 재입력 사유에 대해 메시지를 표시한다.
- 실행 결과



The registration form is shown with a modal error message overlay. The message is:

**비밀번호가 일치하지 않습니다.**

The modal also contains a blue button labeled "닫기" (Close).

The form fields in the background are:

- 비밀번호**: Password input field (masked with dots).
- 비밀번호 확인**: Confirm password input field (masked with dots).
- 성명**: Full name input field (placeholder: "맥썸").
- 소속 기관**: Affiliation input field (placeholder: "학원").
- 전공**: Major input field.
- 핸드폰**: Phone number input field.

## 150 REACT 회원 가입 페이지 만들기 3 - 아이디 중복 체크하기

- 회원 정보에서 아이디는 유일한 값이어야 한다. 회원 정보를 삽입하기 전 중복 체크를 위한 node api를 호출하고 DB에 동일한 계정이 존재하는지 확인한다.
- Register.js
  - 2: 모든 입력 값이 유효성 검사를 통과하면 fnValidate() 함수는 true를 반환한다.
  - 4~6: 중복 체크 node api(/api/register?type=dplicheck)를 호출하면서 파라미터로 입력 받은 이메일 컴럼의 값을 전달한다.
  - 9~18: 새로운 계정이라면 회원 정보 삽입 node api를 호출하는 fnSignInsert() 함수를 실행한다.

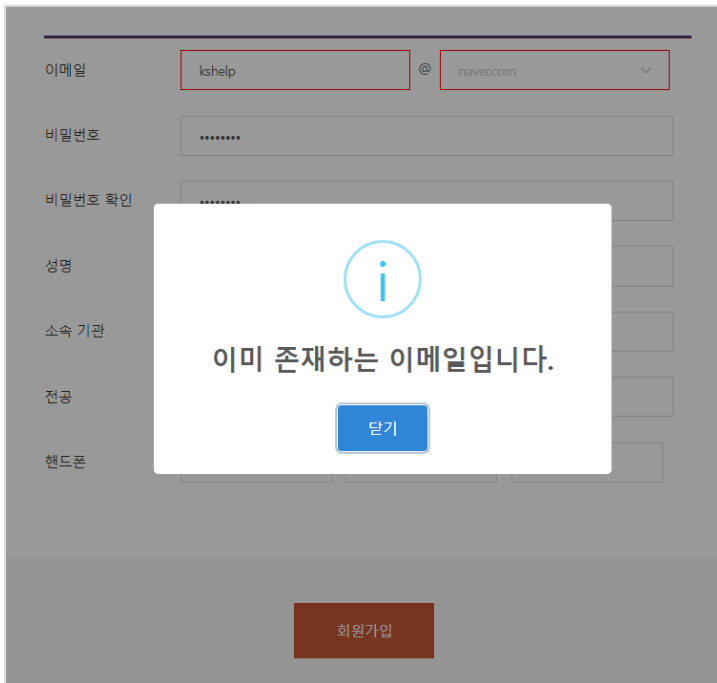
[client/src/components/Register/Register.js]

```

1 ...(<생략>)...
2 if(this.fnValidate()){
3 this.state.full_email = this.email_val_checker+'@'+this.email2_val_checker
4 axios.post('/api/register?type=dplicheck', {
5 is_Email: this.email_val_checker+'@'+this.email2_val_checker
6 })
7 .then(response => {
8 try {
9 const dupli_count = response.data.json[0].num;
10 if(dupli_count !== 0){
11 $('#email_val').addClass('border_validate_err');
12 $('#email2_val').addClass('border_validate_err');
13 this.sweetalert('이미 존재하는 이메일입니다.', '', 'info', '닫기')
14 }else{
15 $('#email_val').removeClass('border_validate_err');
16 $('#email2_val').removeClass('border_validate_err');
17 this.fnSignInsert('signup', e)
18 }
19 } catch (error) {
20 this.sweetalert('작업중 오류가 발생하였습니다.', error, 'error', '닫기')
21 }
22 })
23 .catch(response => { return false; });
24 }
25 ...(<생략>)...

```

- 실행 결과



## 151 REACT 회원 가입 페이지 만들기 4 - 회원 가입 api 호출하기

- 중복 체크가 완료된 회원 정보를 회원 가입 api 호출을 통해 파라미터로 전달한다.
- Register.js 파일

[client/src/components/Register/Register.js]

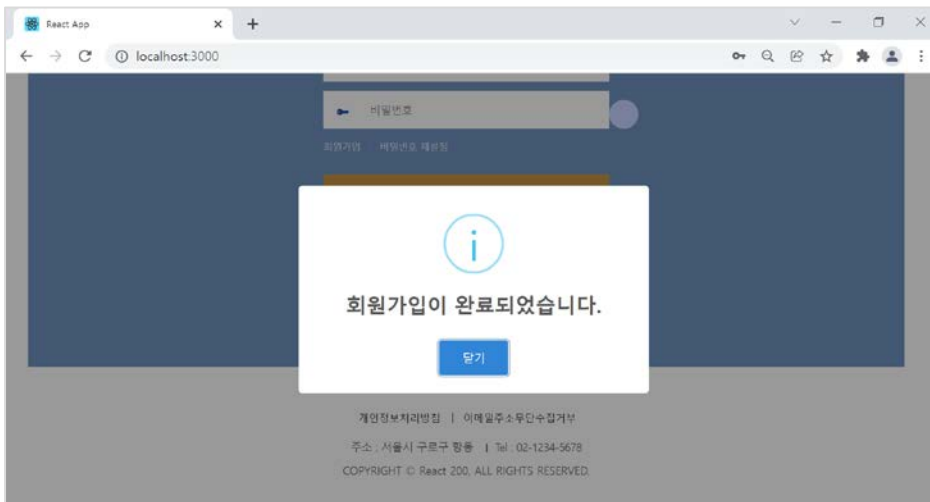
```
01 import React, { Component } from 'react';
02 import axios from "axios";
03 import Swal from 'sweetalert2'
04 import $ from 'jquery';
05
06 class Register extends Component {
07 constructor (props) {
08 super(props);
09 this.state = {
10 }
11 }
12
13 submitClick = async (type, e) => {
14
15 this.email_val_checker = $('#email_val').val();
16 this.email2_val_checker = $('#email2_val').val();
17 this.pwd_val_checker = $('#pwd_val').val();
18 this.pwd_cnf_val_checker = $('#pwd_cnf_val').val();
19 this.name_val_checker = $('#name_val').val();
20 this.org_val_checker = $('#org_val').val();
21 this.major_val_checker = $('#major_val').val();
22 this.phone1_val_checker = $('#phone1_val').val();
23 this.phone2_val_checker = $('#phone2_val').val();
24 this.phone3_val_checker = $('#phone3_val').val();
25
26 this.fnValidate = (e) => {
27 var pattern1 = /[0-9]/;
```

```

28 var pattern2 = /[a-zA-Z]/;
29 var pattern3 = /[~!@#$%^&*()_+|<>?:{}]/;
30
31 if(this.email_val_checker === '') {
32 $('#email_val').addClass('border_validate_err');
33 this.sweetalert('이메일 주소를 다시 확인해주세요.', '', 'info', '닫기')
34 return false;
35 }
36 if(this.email_val_checker.search(/\s/) !== -1) {
37 $('#email_val').addClass('border_validate_err');
38 this.sweetalert('이메일 공백을 제거해 주세요.', '', 'info', '닫기')
39 return false;
40 }
41 $('#email_val').removeClass('border_validate_err');
42
43 if(this.email2_val_checker === '') {
44 $('#email2_val').addClass('border_validate_err');
45 this.sweetalert('이메일 주소를 다시 확인해주세요.', '', 'info', '닫기')
46 return false;
47 }
48 $('#email2_val').removeClass('border_validate_err');
49
50 if(this.pwd_val_checker === '') {
51 $('#pwd_val').addClass('border_validate_err');
52 this.sweetalert('비밀번호를 입력해주세요.', '', 'info', '닫기')
53 return false;
54 }
55 ... (생략) ...

```

## ■ 실행 결과



## 152 NODE 로그인 api 만들기 1 - 라우터 분기, 쿼리 추가하기

- 로그인 api를 호출할 때 필수적으로 아이디와 비밀번호를 파라미터로 전달한다. 전달받은 비밀번호는 암호화되지 않았기 때문에 DB에 저장된 회원 비밀번호와 비교할 수 없다. 비교를 위해 아이디만으로 쿼리를 조회한다. 그리고 파라미터로 전달된 비밀번호를 암호화해서 DB에서 조회된 비밀번호와 비교해야 한다.

## 153 NODE 로그인 api 만들기 2 - bcrypt로 비밀번호 비교하기

- 회원 가입 api에서는 bcrypt의 hash 함수를 사용해 비밀번호를 암호화해서 DB에 저장했다. bcrypt의 compare 함수를 사용하면 파라미터로 전달받은 비밀번호를 같은 방식으로 암호화해 DB에서 조회한 비밀번호와 비교한다.
- dbconnect\_Module.js 파일
  - 17~26: bcrypt의 compare 함수로 파라미터로 전달받은 비밀번호(req.body.is\_Pasword)와 DB에서 조회한 암호화된 비밀번호(json[0].userpassword)를 비교한다. 두 비밀번호가 일치한다면 login\_flag 변수값이 true로, 일치하지 않는다면 false가 반환된다.

[routes/dbconnect\_Module.js]

```

01 ...(생략)...
02 connection.query(query, function (error, results) {
03 if (error) {
04 console.log("db error***** : "+error);
05 }
06 var time2 = new Date();
07 console.log('## '+time2+ ' ##');
08 console.log('## RESULT DATA LIST ## : \n', results);
09 if(results != undefined){
10 string = JSON.stringify(results);
11 var json = JSON.parse(string);
12 if (req.body.crud == "select") {
13 if (param.mapper_id == "selectLoginCheck") {
14 if (json[0] == undefined) {
15 res.send(null);
16 } else {
17 bcrypt.compare(req.body.is_Password, json[0].userpassword, function(
18 err,
19 login_flag
20) {
21 if (login_flag == true) {
22 res.send({ json });
23 } else {
24 res.send(null);
25 }
26 });
27 }
28 } else {
29 res.send({ json });
30 }
31 }else{
32 res.send("succ");
33 }
34 }else{
35 res.send("error");
36 }
37 }
38 connection.release();
39 console.log("===== Node Mybatis Query Log End =====\n");
40 });
41 })
42 } catch (error) {
43 console.log("pool error : "+error);
44 }
45 });
46
47 module.exports = router;

```



## 154 REACT 로그인 페이지 만들기 - 로그인 api 호출하기

- 로그인 node api 호출 시 쿼리에 필요한 변수들을 axios의 post 함수 파라미터로 전달한다. 로그인 아이디와 비밀번호를 입력받아 json 형태로 할당해 request로 전달한다.
- LoginForm.js 파일
  - 12~13: 아이디나 비밀번호가 입력되지 않았다면 재입력 메시지를 표시한다.
  - 15~18: 로그인 api를 호출하고 파라미터로 로그인 아이디(is\_Email)와 비밀번호(is\_Password)를 전달한다.
  - 20~22: 로그인 api 호출 결과 반환된 response에서 아이디(usermail), 회원명(username), 암호화된 비밀번호(userpassword)를 각각 변수에 할당한다.
  - 24~25: 일치하는 회원 정보가 DB에 존재해 회원 정보가 정상적으로 반환됐다면 로그인 성공 메시지를 표시한다.
  - 45~47: 로그인이 정상적으로 완료되면 Software Tools 목록(/SoftwareList)으로 페이지를 이동한다. 로그인 성공 메시지를 1초 표시하고 페이지 이동을 하기 위해 setTimeout 함수를 사용한다.

[client/src/components/LoginForm.js]

```

1 import React, { Component } from 'react';
2 import { Link } from 'react-router-dom';
3 import axios from "axios";
4 import cookie from 'react-cookies';
5 import Swal from 'sweetalert2';
6 import $ from 'jquery';
7
8 class LoginForm extends Component {
9 submitClick = (e) => {
10 this.email_val = $('#email_val').val();
11 this.pwd_val = $('#pwd_val').val();
12 if(this.email_val === '' || this.pwd_val === ''){
13 this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info', '닫기')
14 }else{
15 axios.post('/api/LoginForm?type=signin', {
16 is_Email: this.email_val,
17 is_Password: this.pwd_val
18 })
19 .then(response => {
20 var userid = response.data.json[0].useremail
21 var username = response.data.json[0].username
22 var upw = response.data.json[0].userpassword
23
24 if(userid != null && userid != ''){
25 this.sweetalert('로그인 되었습니다.', '', 'info', '닫기')
26 const expires = new Date()
27 expires.setMinutes(expires.getMinutes() + 60)
28
29 axios.post('/api/LoginForm?type=SessionState', {
30 is_Email: userid,
31 is_UserName: username,
32 })
33 .then(response => {
34 cookie.save('userid', response.data.token1
35 , { path: '/', expires })
36 cookie.save('username', response.data.token2
37 , { path: '/', expires })
38 cookie.save('userpassword', upw

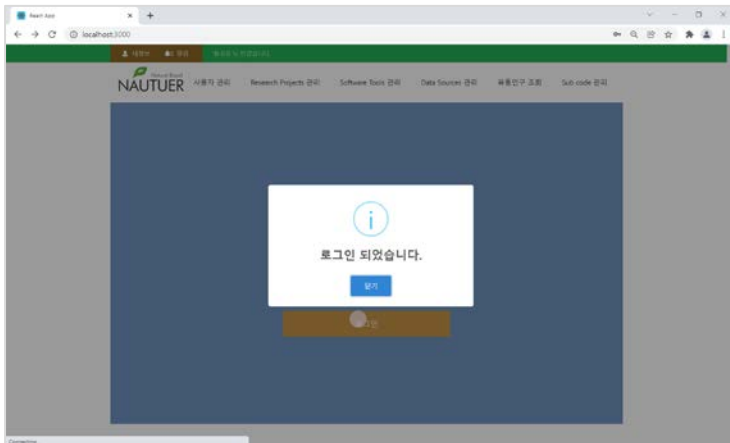
```

```

39 , { path: '/', expires })
40 })
41 .catch(error => {
42 this.sweetalert('작업중 오류가 발생하였습니다.', error, 'error', '닫기');
43 });
44
45 setTimeout(function() {
46 window.location.href = '/SoftwareList';
47 }, bind(this), 1000);
48 }else{
49 this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info', '닫기')
50 }
51 })
52 .catch(error => {this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info',
53 '닫기')});
54 }
55 }
56
57 sweetalert = (title, contents, icon, confirmButtonText) => {
58 Swal.fire({
59 title: title,
60 text: contents,
61 icon: icon,
62 confirmButtonText: confirmButtonText
63 })
64 }
65
66 render () {
67 return (
68 <section className="main">
69 <div className="m_login">
70 <h3>
71 <h3>LOGIN</h3>
72 <div className="log_box">
73 <div className="in_ty1">
74
75 <input type="text" id="email_val" placeholder="이메일" />
76 </div>
77 <div className="in_ty1">
78
79
80
81 <input type="password" id="pwd_val" placeholder="비밀번호" />
82 </div>
83 <ul className="af">
84 <Link to={'/register'}>회원가입</Link>
85 <li className="pwr_b">비밀번호 재설정
86
87 <div className="s_bt" type="" onClick={(e) => this.submitClick(e)}>로그인</div>
88 </div>
89 </div>
90 </section>
91);
92 }
93 }
94
95 export default LoginForm;

```

## ■ 실행 결과



## 155 회원 정보 암호화 api 만들기 - jwt로 회원 정보 암호화하기

- 로그인 정보가 정상적으로 확인됐다면 쿠키에 회원 정보를 저장해 로그인을 유지해야 한다. 이때 회원 정보가 그대로 쿠키에 저장되면 위험하기 때문에 jsonwebtoken 패키지를 사용해 암호화한다.

```
// jsonwebtoken을 설치한다.
D:\dev\workspace\react\react200-lab\155>yarn add jsonwebtoken
```

- jwt.js 파일
  - 1~2: json 객체(jwtObj)를 생성해 key(secret)와 value(react200)을 할당한다. value 값을 비밀 키가 되는 텍스트로 임의의 값으로 정하고 외부에 표시되지 않도록 관리한다.

```
[ignorefile/jwt.js]
```

```
1 let jwtObj = {};
2 jwtObj.secret = "react200"
3 module.exports = jwtObj
```

- UsersRoute.js 파일
  - 79~82: jsonwebtoken 패키지의 sign 함수를 사용해 아이디의 암호화된 토큰을 생성한다.

```
[routes/UsersRoute.js]
```

```
1 var express = require('express');
2 var router = express.Router();
3 const bodyParser = require('body-parser');
4
5 const bcrypt = require('bcrypt');
6 const saltRounds = 10;
7
8 let jwt = require("jsonwebtoken");
9 let secretObj = require("../ignorefile/jwt");
10
11 router.use(bodyParser.json());
12 router.use(bodyParser.urlencoded({ extended: true }));
13
14 router.post('/', (req, res, next) => {
15 var type = req.query.type;
```

```

16 if(type == "signup"){
17 //회원가입 정보 삽입
18 try {
19 // Mysql Api 모듈(CRUD)
20 var dbconnect_Module = require('./dbconnect_Module');
21
22 //Mysql 쿼리 호출정보 입력
23 req.body.mapper = 'UserMapper';//mybatis xml 파일명
24 req.body.crud = 'insert';//select, insert, update, delete 중에 입력
25 req.body.mapper_id = 'insertUser';
26
27 var myPlaintextPassword = req.body.is_Password;
28 if(myPlaintextPassword != '' && myPlaintextPassword != undefined){
29 bcrypt.genSalt(saltRounds, function(err, salt) {
30 bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {
31 req.body.is_Password = hash;
32 router.use('/', dbconnect_Module);
33 next('route')
34 });
35 });
36 }else{
37 router.use('/', dbconnect_Module);
38 next('route')
39 }
40 } catch (error) {
41 console.log("Module > dbconnect error : "+ error);
42 }
43 }else if(type == "dplicheck"){
44 //이메일 중복체크
45 try {
46 // Mysql Api 모듈(CRUD)
47 var dbconnect_Module = require('./dbconnect_Module');
48
49 //Mysql 쿼리 호출정보 입력
50 req.body.mapper = 'UserMapper';//mybatis xml 파일명
51 req.body.crud = 'select';//select, insert, update, delete 중에 입력
52 req.body.mapper_id = 'selectUserDpliCheck';
53 router.use('/', dbconnect_Module);
54 next('route')
55 } catch (error) {
56 console.log("Module > dbconnect error : "+ error);
57 }
58 }else if(type == "signin"){
59 //로그인 조회
60 try {
61 // Mysql Api 모듈(CRUD)
62 var dbconnect_Module = require('./dbconnect_Module');
63
64 //Mysql 쿼리 호출정보 입력
65 req.body.mapper = 'UserMapper';//mybatis xml 파일명
66 req.body.crud = 'select';//select, insert, update, delete 중에 입력
67 req.body.mapper_id = 'selectLoginCheck';
68
69 router.use('/', dbconnect_Module);
70 next('route')
71
72 } catch (error) {
73 console.log("Module > dbconnect error : "+ error);
74 }
75 }else if(type == "SessionState"){
76 var userid = req.body.is_Email
77 var name = req.body.is_UserName
78 try {
79 let token1 = jwt.sign(
80 { email: userid },

```

```

81 secretObj.secret,
82 { expiresIn: '60m' })
83
84 let token2 = jwt.sign(
85 { username: name },
86 secretObj.secret,
87 { expiresIn: '60m' })
88 res.send({"token1":token1, "token2":token2});
89 } catch (error) {
90 res.send(error)
91 }
92 }
93 ... (생략) ...

```

## 156 REACT 쿠키로 로그인 유지하기 - 쿠키에 회원 정보 저장하기

- 로그인 api를 호출해 쿠키에 저장할 데이터(아이디, 회원명, 암호화된 비밀번호)가 반환됐다. 아이디와 회원명은 jwt로 암호화했다. bcrypt로 암호화된 비밀번호는 유출돼도 로그인에 사용할 수 없기 때문에 그대로 사용해도 된다. 세가지 데이터를 react-cookies의 save 함수로 쿠키에 저장한다.

```

// react-cookies를 설치한다.
D:\dev\workspace\react\react200-lab\156>yarn add react-cookies

```

- LoginForm.js 파일

```
[client/src/components/LoginForm.js]
```

```

1 import React, { Component } from 'react';
2 import { Link } from 'react-router-dom';
3 import axios from "axios";
4 import cookie from 'react-cookies';
5 import Swal from 'sweetalert2';
6 import $ from 'jquery';
7
8 class LoginForm extends Component {
9 submitClick = (e) => {
10 this.email_val = $('#email_val').val();
11 this.pwd_val = $('#pwd_val').val();
12 if(this.email_val === '' || this.pwd_val === ''){
13 this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info', '닫기')
14 }else{
15 axios.post('/api/LoginForm?type=signin', {
16 is_Email: this.email_val,
17 is_Password: this.pwd_val
18 })
19 .then(response => {
20 var userid = response.data.json[0].useremail
21 var username = response.data.json[0].username
22 var upw = response.data.json[0].userpassword
23
24 if(userid != null && userid != ''){
25 this.sweetalert('로그인 되었습니다.', '', 'info', '닫기')
26 const expires = new Date()
27 expires.setMinutes(expires.getMinutes() + 60)
28
29 axios.post('/api/LoginForm?type=SessionState', {

```

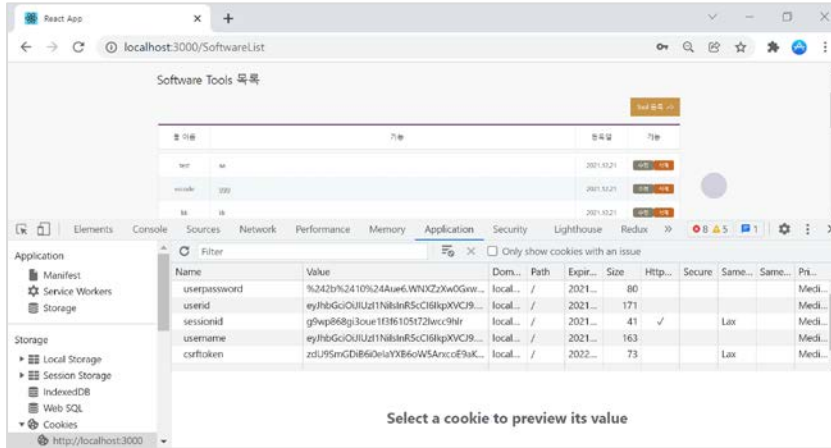
```

30 is_Email: userid,
31 is_UserName: username,
32 })
33 .then(response => {
34 cookie.save('userid', response.data.token1
35 , { path: '/', expires })
36 cookie.save('username', response.data.token2
37 , { path: '/', expires })
38 cookie.save('userpassword', upw
39 , { path: '/', expires })
40 })
41 .catch(error => {
42 this.sweetalert('작업중 오류가 발생하였습니다.', error, 'error', '닫기');
43 });
44
45 setTimeout(function() {
46 window.location.href = '/SoftwareList';
47 }.bind(this),1000);
48 }else{
49 this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info', '닫기')
50 }
51 })
52 .catch(error => {this.sweetalert('이메일과 비밀번호를 확인해주세요.', '', 'info',
53 '닫기')}});
54 }
55 }
56
57 sweetalert = (title, contents, icon, confirmButtonText) => {
58 Swal.fire({
59 title: title,
60 text: contents,
61 icon: icon,
62 confirmButtonText: confirmButtonText
63 })
64 }
65
66 render () {
67 return (
68 <section className="main">
69 <div className="m_login">
70 <h3>
71 LOGIN</h3>
72 <div className="log_box">
73 <div className="in_ty1">
74
75 <input type="text" id="email_val" placeholder="이메일" />
76 </div>
77 <div className="in_ty1">
78
79
80
81 <input type="password" id="pwd_val" placeholder="비밀번호" />
82 </div>
83 <ul className="af">
84 <Link to={'/register'}>회원가입</Link>
85 <li className="pwr_b">비밀번호 재설정
86
87 <div className="s_bt" type="" onClick={(e) => this.submitClick(e)}>로그인</div>
88 </div>
89 </div>
90 </section>
91);
92 }
93 }
94

```

```
95 export default LoginForm;
```

## ■ 실행 결과



## 157 회원 정보 복호화 api 만들기 - jwt로 회원 정보 복호화하기

- 보안상의 이유로 암호화해 쿠키에 저장했던 회원 정보를 사용해야 할 때가 있다. 화면에 회원 정보를 표시해야 하거나 로그인 인증이 다시 필요한 시점이다. 회원 정보의 복호화에는 암호화를 할 때 사용했던 비밀 키가 사용된다.
- LoginForm.js 파일
  - 8~9: jsonwebtoken 패키지를 require해 사용할 수 있도록 한다. 비밀 키가 저장된 파일 (/ignorefile/jwt.js)의 json 객체를 secretObj 변수에 할당한다.
  - 97~99: 회원 정보가 존재할 경우, jsonwebtoken 패키지의 verify 함수를 사용해 회원 정보를 복호화한다. 암호화된 회원 정보(token1, token2)를 비밀 키(secretObj.secret)를 사용해 복호화하고 각각 변수(decoded1, decoded2)에 할당한다.
  - 100: 복호화된 이메일(decoded1.email)과 회원명(decoded2.username)을 json 형태로 response에 담아 전달한다.

```
[routes/UsersRoute.js]
```

```
1 var express = require('express');
2 var router = express.Router();
3 const bodyParser = require('body-parser');
4
5 const bcrypt = require('bcrypt');
6 const saltRounds = 10;
7
8 let jwt = require("jsonwebtoken");
9 let secretObj = require("../ignorefile/jwt");
10
11 router.use(bodyParser.json());
12 router.use(bodyParser.urlencoded({ extended: true }));
13
14 router.post('/', (req, res, next) => {
15 var type = req.query.type;
16 if(type == "signup"){
17 //회원가입 정보 삽입
18 try {
19 // Mysql Api 모듈(CRUD)
```

```

20 var dbconnect_Module = require('./dbconnect_Module');
21
22 //Mysql 쿼리 호출정보 입력
23 req.body.mapper = 'UserMapper';//mybatis xml 파일명
24 req.body.crud = 'insert';//select, insert, update, delete 중에 입력
25 req.body.mapper_id = 'insertUser';
26
27 var myPlaintextPassword = req.body.is_Password;
28 if(myPlaintextPassword != '' && myPlaintextPassword != undefined){
29 bcrypt.genSalt(saltRounds, function(err, salt) {
30 bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {
31 req.body.is_Password = hash;
32 router.use('/', dbconnect_Module);
33 next('route')
34 });
35 });
36 }else{
37 router.use('/', dbconnect_Module);
38 next('route')
39 }
40 } catch (error) {
41 console.log("Module > dbconnect error : "+ error);
42 }
43 }else if(type == "dplicheck"){
44 //이메일 중복체크
45 try {
46 // Mysql Api 모듈(CRUD)
47 var dbconnect_Module = require('./dbconnect_Module');
48
49 //Mysql 쿼리 호출정보 입력
50 req.body.mapper = 'UserMapper';//mybatis xml 파일명
51 req.body.crud = 'select';//select, insert, update, delete 중에 입력
52 req.body.mapper_id = 'selectUserDpliCheck';
53 router.use('/', dbconnect_Module);
54 next('route')
55 } catch (error) {
56 console.log("Module > dbconnect error : "+ error);
57 }
58 }else if(type == "signin"){
59 //로그인 조회
60 try {
61 // Mysql Api 모듈(CRUD)
62 var dbconnect_Module = require('./dbconnect_Module');
63
64 //Mysql 쿼리 호출정보 입력
65 req.body.mapper = 'UserMapper';//mybatis xml 파일명
66 req.body.crud = 'select';//select, insert, update, delete 중에 입력
67 req.body.mapper_id = 'selectLoginCheck';
68
69 router.use('/', dbconnect_Module);
70 next('route')
71 } catch (error) {
72 console.log("Module > dbconnect error : "+ error);
73 }
74 }else if(type == "SessionState"){
75 var userid = req.body.is_Email
76 var name = req.body.is_UserName
77 try {
78 let token1 = jwt.sign(
79 { email: userid },
80 secretObj.secret,
81 { expiresIn: '60m' })
82
83 let token2 = jwt.sign(

```



```

85 { username: name },
86 secretObj.secret,
87 { expiresIn: '60m' })
88 res.send({"token1":token1, "token2":token2});
89 } catch (error) {
90 res.send(error)
91 }
92 }else if(type == "SessionConfirm"){
93 try {
94 let token1 = req.body.token1;
95 let token2 = req.body.token2;
96
97 if(token1 != undefined && token1 != '' && token2 != undefined && token2 != ''){
98 let decoded1 = jwt.verify(token1, secretObj.secret);
99 let decoded2 = jwt.verify(token2, secretObj.secret);
100 res.send({"token1":decoded1.email, "token2":decoded2.username});
101 }else{
102 res.send({"token1":"","token2":""});
103 }
104 } catch (error) {
105 res.send(error)
106 }
107 }else if(type == "SessionSignin"){
108 // 쿠키 정보로 사용자 인증
109 try {
110 // Mysql Api 모듈(CRUD)
111 var dbconnect_Module = require('./dbconnect_Module');
112 //Mysql 쿼리 호출정보 입력
113 req.body.mapper = 'UserMapper';//mybatis xml 파일명
114 req.body.crud = 'select';//select, insert, update, delete 중에 입력
115 req.body.mapper_id = 'selectSessionLoginCheck';
116
117 router.use('/', dbconnect_Module);
118 next('route')
119
120 } catch (error) {
121 console.log("Module > dbconnect error : "+ error);
122 }
123 }
124 });
125
126 module.exports = router;

```

## 158 로그인 회원에게 권한 허용하기

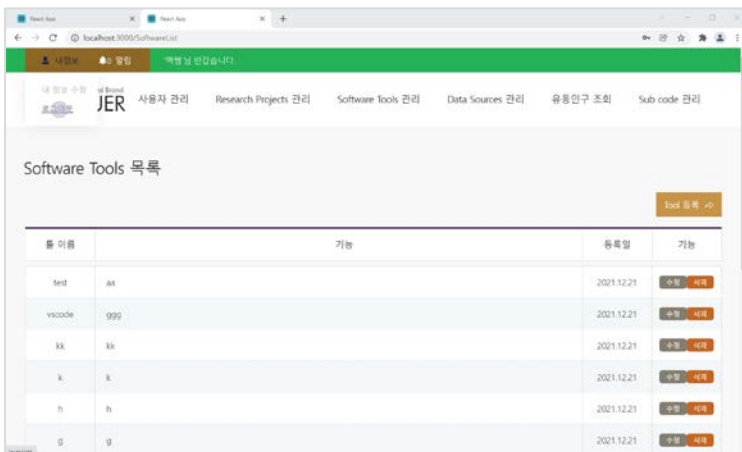
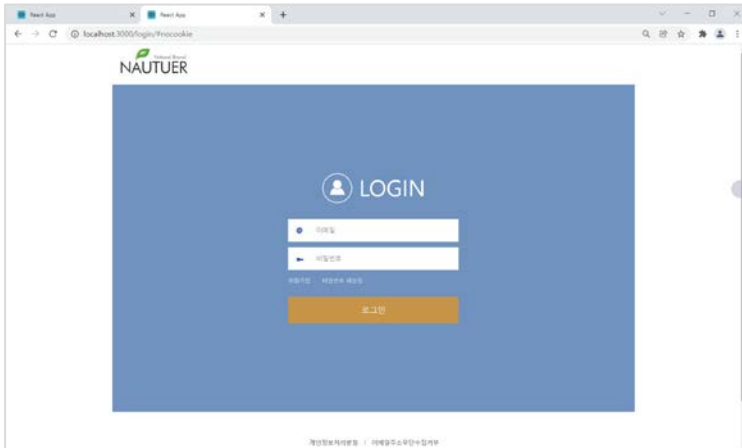
- 쿠키에 저장됐던 데이터(아이디, 회원명, 암호화된 비밀번호)를 react-cookies의 load 함수로 불러온다. 불러온 아이디와 암호화된 비밀번호로 사용자 인증을 하고 인증이 완료된 경우에만 특정 페이지에 접근 권한을 허용한다.

## 159 로그인 세션 시간 관리하기

- 로그인이 성공하면 쿠키 시간이 60분 할당된다. 로그인 이후 사용자가 페이지 새로 고침을 하게 되면 그 시점부터 쿠키 시간을 새로 60분 할당한다. 쿠키 시간 재할당 코드는 모든 페이지에서 표시되는 header에 구현한다.

## 160 로그인 상태에서만 header 표시하기, 로그아웃 구현하기

- 로그인 되지 않은 상태에서는 회원 정보, 로그아웃 버튼, 페이지 내비게이션 영역을 표시하지 않는다.
- 실행 결과



## 161 비밀번호 재설정 메일 템플릿 만들기

- 로그인 비밀번호를 잊어버린 경우, 웹 사이트의 아이디로 사용되는 이메일 인증을 통해 비밀번호를 재설정할 수 있다. 이메일 템플릿은 사용자가 이메일로 받게 되는 내용인데 html을 사용하면 웹사이트 링크를 걸 수 있는 태그를 사용할 수 있다.

## 162 메일 발송 api 만들기 - nodemailer 패키지 사용하기

- 메일 발송 api를 호출할 때 파라미터로 메일 제목, 이메일, 비밀번호를 전달한다. 메일 템플릿의 [비밀번호 변경하기] 버튼 링크를 웹 사이트 주소에 이메일과 비밀번호 토큰을 넣은 값으로 치환한다.

```
// 메일 템플릿 파일을 불러오기 위해 fs 패키지와 메일 발송을 위해 nodemailer 패키지를 설치한다.
D:\dev\workspace\react\react200-lab\162>yarn add nodemailer
D:\dev\workspace\react\react200-lab\162>yarn add fs
```

## 163 비밀번호 조회 api 만들기 - 라우터 분기, 쿼리 추가하기

- 비밀번호를 잊은 상태이기 때문에 사용자 정보를 조회할 수 없다. 아이디와 회원명으로 비밀번호 재설정에 필요한 암호화된 비밀번호를 조회한다.

## 164 아이디와 회원으로 비밀번호 조회 api 호출하기

- 비밀번호 재설정 메일을 발송하기 위해 비밀번호가 아닌 값으로 회원 정보를 조회해야 한다. 아이디와 회원명을 조건으로 암호화된 비밀번호를 조회한다. 메일 발송 함수를 호출하면서 파라미터로 아이디와 암호화된 비밀번호를 전달한다.

## 165 비밀번호 재설정하기 - 이메일 발송 api 호출하기

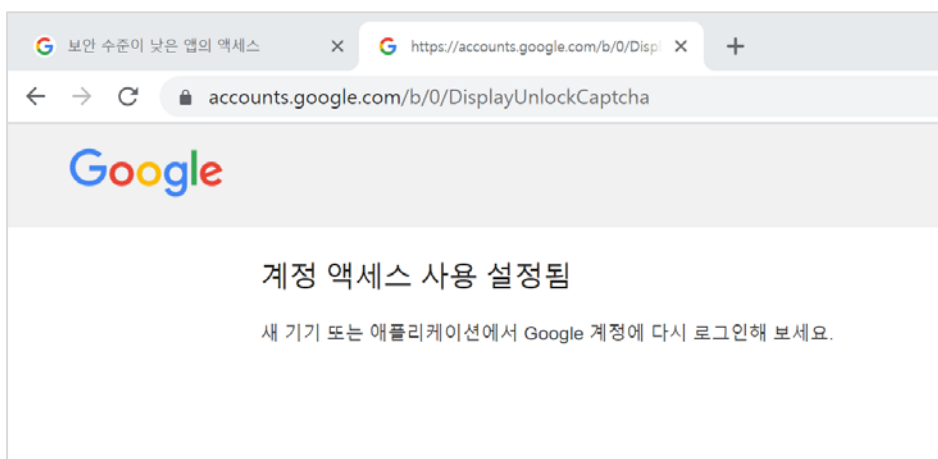
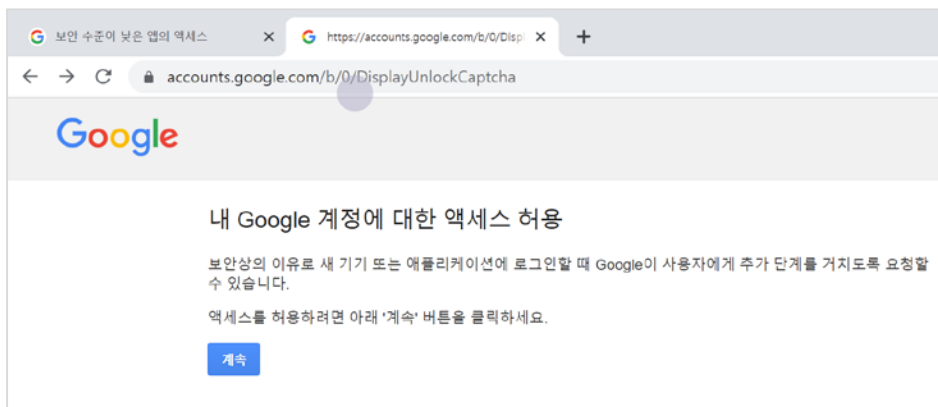
- 아이디(이메일)와 회원명으로 조회된 암호화된 비밀번호와 수신자 이메일 주소, 메일 제목으로 파라미터에 담아 api를 호출한다.

## 166 GMAIL 계정 접근 권한 허용하기

- 웹 사이트에서 구글 계정의 자원을 사용해 이메일을 발송한다. 구글에서는 기본 설정으로 보안이 증명되지 않은 웹 사이트에서 구글 자원에 접근하는 것을 제한한다. 접근을 허용하기 위해 사용할 계정으로 로그인을 하고 다음 화면에서 권한을 허용한다.
- 보안 수준이 낮은 앱의 액세스 링크로 접속해 다음과 같이 보안 수준이 낮은 앱 허용을 사용으로 변경한다.
  - <https://myaccount.google.com/lesssecureapps?pli=1>

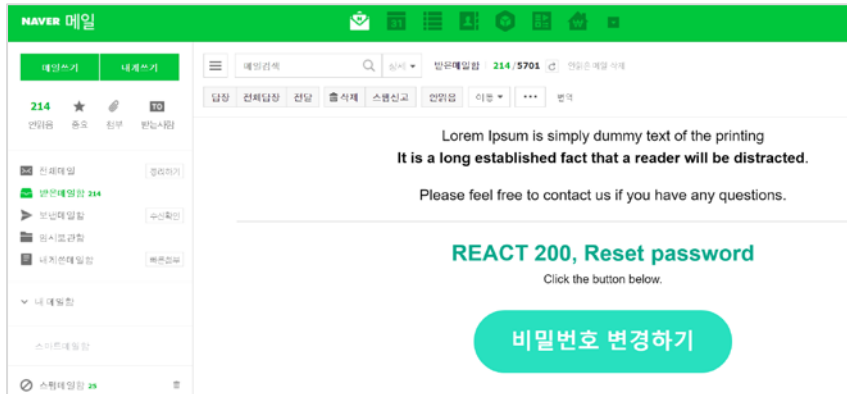


- 내 Google 계정에 대한 액세스 허용 링크로 접속한 후 [계속] 버튼을 눌러 권한을 허용한다.
  - <https://accounts.google.com/b/0/DisplayUnlockCaptcha>



## 167 수신된 이메일과 버튼 링크 확인하기

## ■ 실행 결과



## 168 이메일 토큰으로 사용자 인증하기

- 새로운 비밀번호로 변경할 수 있는 페이지를 만들고 인증된 사용자만 접근을 허용한다.

## 169 비밀번호 수정 api 만들기

- 비밀번호를 잊은 사용자가 이메일 인증을 완료하면 새로운 비밀번호를 설정할 수 있다. 아이디로 사용자 데이터를 조회해 비밀번호 컬럼을 변경한다.

## 170 비밀번호 수정 api 호출하기

- 비밀번호 수정 api를 호출하는 방식은 전달하는 파라미터 수만 줄었을 뿐 회원 가입 api를 호출하는 방식과 동일하다. update 쿼리에 필요한 아이디(이메일)와 새로운 비밀번호를 파라미터로 전달한다.