웹 개발 보고서 – React 게시판

컴퓨터공학과 김남우

ICT 융복합 연구소

목차

[reactboard](file:///C:\Users\Master\Desktop\1차년도_WEB_보고서_최민.docx#_Toc147867630) 2

[server.js](file:///C:\Users\Master\Desktop\1차년도_WEB_보고서_최민.docx#_Toc147867643) 3

client[.js](file:///C:\Users\Master\Desktop\1차년도_WEB_보고서_최민.docx#_Toc147867631) 5

[public](file:///C:\Users\Master\Desktop\1차년도_WEB_보고서_최민.docx#_Toc147867635) 5

index.html 5

[src](file:///C:\Users\Master\Desktop\1차년도_WEB_보고서_최민.docx#_Toc147867635) 6

App.js 6

index.js 8

components 9

List.js 9

Write.js 14

View.js 20

Edit.js 23

Delete.js 29

DeleteView.js 33

Pagination.js 37

Timestamp.js 39

withStyles.js 40

# react\_board

이 프로그램은 React와 Material-UI를 사용하여 구현된 게시판 웹 사이트입니다. 사용자는 게시글을 생성, 조회, 수정, 삭제할 수 있으며, 삭제된 게시글을 복원할 수도 있습니다. 또한 페이지네이션을 통해 여러 페이지의 게시글을 탐색할 수 있습니다.

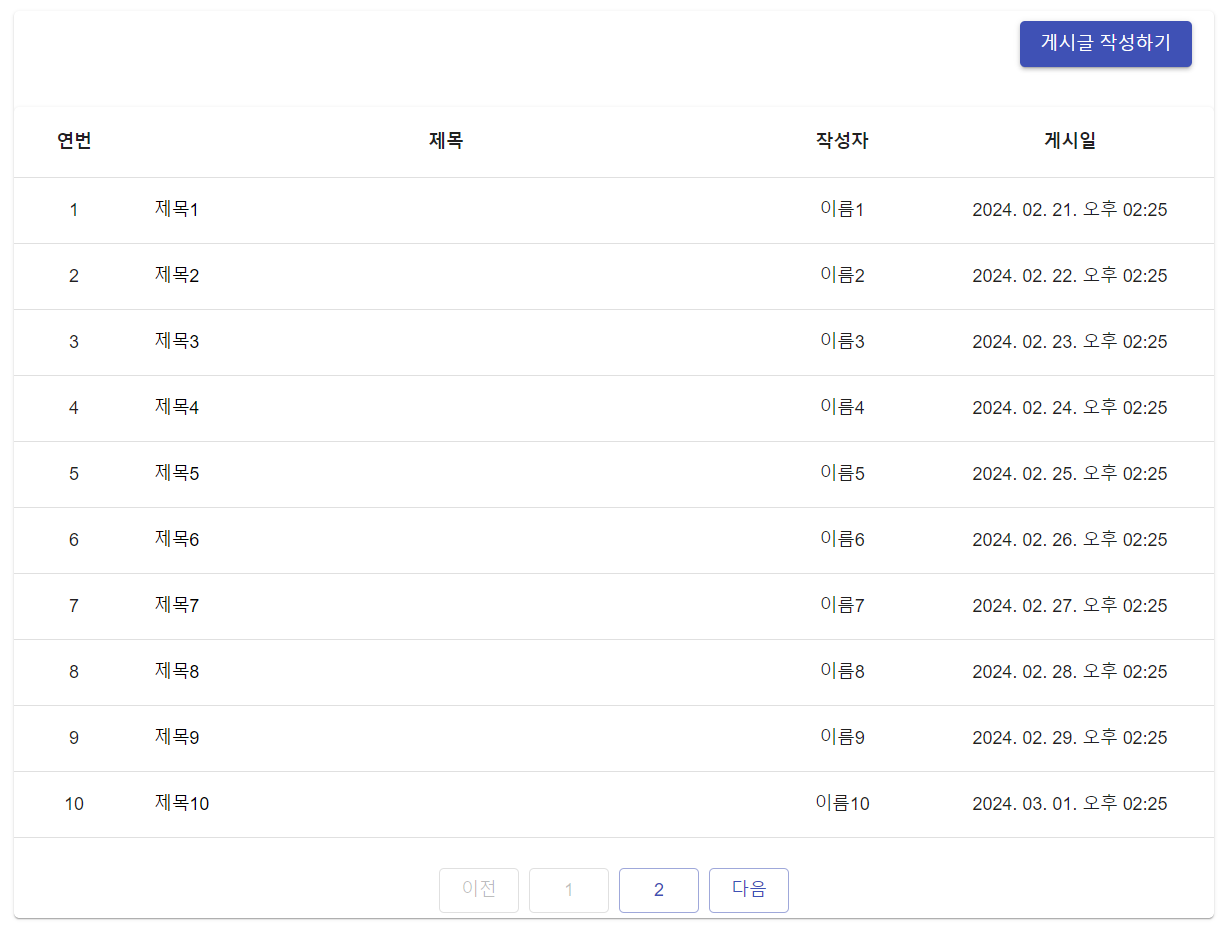


그림 1 - 게시판 메인화면

**프로그램의 기능**

- 게시글 목록 : 게시글은 제목, 작성자, 게시일 및 삭제일과 함께 테이블 형식으로 표시

- 게시글 생성 : 사용자는 제목과 내용, 작성자의 이름을 입력하여 새로운 게시글을 생성

- 게시글 조회 : 게시글 제목을 클릭하면 게시글의 세부 정보를 볼 수 있는 페이지로 이동

- 게시글 수정 : 세부 정보 페이지에서 수정 버튼을 클릭하여 게시글을 수정

- 게시글 삭제 : 세부 정보 페이지에서 삭제 버튼을 클릭하여 게시글을 삭제

- 게시글 복원 : 삭제된 게시글은 복원 버튼을 클릭하여 복원

- 페이지네이션 : 여러 페이지의 게시글을 표시하기 위해 페이지네이션 제공

## server.js

|  |  |
| --- | --- |
| **require** | **설명** |
| express | Node.js웹 애플리케이션을 빠르고 간편하게 개발하기 위한 프레임워크 |
| body-parser | 요청의 본문을 파싱하여 서버에서 사용할 수 있는 형태로 제공하는 express의 미들웨어 |
| fs | 파일 시스템과 상호작용하기 위한 메서드를 제공하는 Node.js 내장 모듈 |
| mysql2 | MySQL 서버와의 효율적인 통신을 지원하는 Node.js용 MySQL 클라이언트 라이브러리 |
| createPool | mysql2 모듈에서 제공되며, MySQL 데이터베이스에 대한 연결 풀을 생성하는 메서드 |

**app.use(bodyParser.json());**

- HTTP 요청 본문(body)이 JSON 형식일 경우, 데이터를 파싱하여 JavaScript 객체로 변환

- 변환된 데이터는 req.body 객체에 저장되어 라우트 핸들러에서 사용 가능

**app.use(bodyParser.urlencoded({ extended: true }));**

- 마찬가지로 URL-encoded 형식일 경우, 데이터를 파싱하여 JavaScript 객체로 변환

- { extended: true }: 중첩된 객체를 파싱할 수 있도록 하는 옵션

**const data = fs.readFileSync('./database.json');**

- fs 모듈을 사용하여 현재 디렉토리의 database.json 파일을 동기적으로 읽어옴

- readFileSync 함수: 파일을 읽어오고 그 내용을 반환

**const conf = JSON.parse(data);**

- 읽어온 JSON 형식의 데이터를 JavaScript 객체로 파싱

- JSON.parse 함수: database.json 파일의 내용을 JavaScript 객체로 변환

**app.get('/api/posts', (req, res) => { … });**

- /api/posts 경로로 들어오는 GET 요청에 대한 핸들러

- 데이터베이스의 posts 테이블에서 모든 글을 조회한 뒤, 결과를 클라이언트에게 전송

- res.send(rows)를 통해 데이터베이스 쿼리 결과인 rows를 클라이언트에게 응답으로 전송

**app.get('/api/posts/:num', (req, res) => { … });**

- /api/posts /:num 경로로 들어오는 GET 요청에 대한 핸들러

**app.post('/api/posts', (req, res) => { ... });**

- /api/posts 경로로 들어오는 POST 요청에 대한 핸들러

**app.patch('/api/posts/:num', (req, res) => { … });**

- /api/posts/:num 경로로 들어오는 PATCH 요청에 대한 핸들러

**app.delete('/api/posts/:num', (req, res) => { … });**

- /api/posts/:num 경로로 들어오는 DELETE요청에 대한 핸들러

**app.get('/api/delete', (req, res) => { … });**

- /api/delete경로로 들어오는 GET 요청에 대한 핸들러

**app.get('/api/delete/:num', (req, res) => { … });**

- /api/delete/:num경로로 들어오는 GET요청에 대한 핸들러

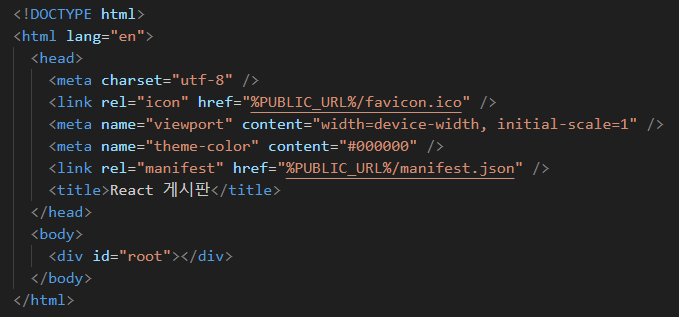
**app.delete('/api/delete/:num', (req, res) => { … });**

- /api/delete/:num'경로로 들어오는 DELETE요청에 대한 핸들러

## client

### public

#### Index.html



**<div id="root"></div>**

- 실제 내용이 들어가는 부분

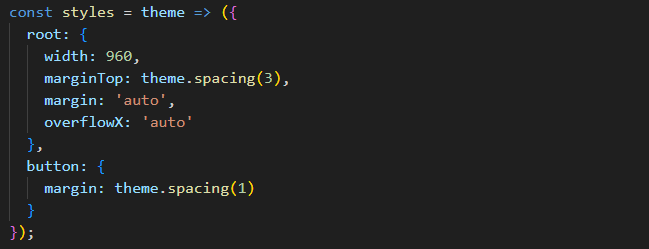
- React 앱이 렌더링될 DOM 요소를 지정하는 <div> 요소

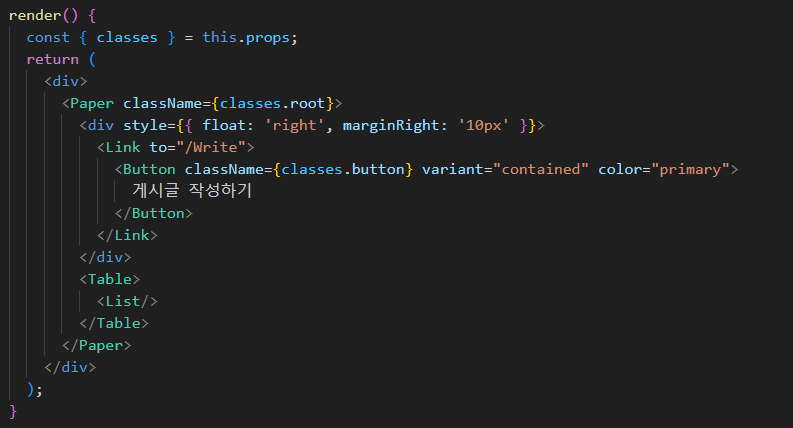
### src

#### App.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 라이브러리의 React 객체와 React 컴포넌트 클래스 |
| ./components/List | List 컴포넌트 import |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/styles | Material-UI 라이브러리에서 제공하는 withStyles 함수 |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |
| Link from react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트 |

**styles설정**





- Paper 컴포넌트를 사용하여 페이지를 감싸고, 그 안에 컨텐츠를 배치, classes.root를 통해 스타일을 적용

- Link 컴포넌트를 사용하여 "게시글 작성하기" 버튼을 누를 시 "/Write" 경로로 이동

- List 컴포넌트를 렌더링하여 게시글 목록을 표시

#### Index.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 라이브러리의 React 객체 |
| react-dom/client | React DOM 모듈에서 ReactDOM 객체를 가져와 React 애플리케이션을 실제 DOM에 렌더링 |
| ./App | App.js의 App 컴포넌트, 메인 애플리케이션 컴포넌트 |
| react-router-dom | React Router 라이브러리의 BrowserRouter, Routes, Route 객체 |
| ./components/View | View.js 파일의 View 컴포넌트 |
| ./components/Wirte | Wirte.js 파일의 Wirte 컴포넌트 |
| ./components/Edit | Edit.js 파일의 Edit 컴포넌트 |
| ./components/Delete | Delete.js 파일의 Delete 컴포넌트 |
| ./components/DeleteView | DeleteView.js 파일의 DeleteView 컴포넌트 |

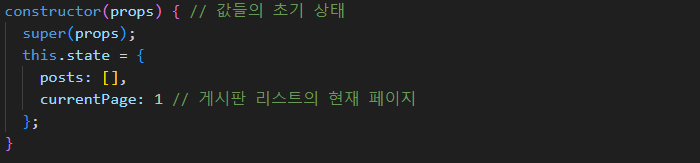
**라우터**

|  |  |
| --- | --- |
| **라우터** | **설명** |
| / | 메인 화면의 URL에 대한 요청을 처리하는JavaScript 함수 또는 컨트롤러를 연결 |
| /View/:num | 게시글 자세히보기와 연관있는 URL에 대한 요청을 처리하는 JavaScript 함수 또는 컨트롤러를 연결 |
| /Write | 게시글 작성과 연관있는 URL에 대한 요청을 처리하는 JavaScript 함수 또는 컨트롤러를 연결 |
| /Edit/:num | 게시글 수정과 연관있는 URL에 대한 요청을 처리하는 JavaScript 함수 또는 컨트롤러를 연결 |
| /delete | 게시글 삭제와 연관있는 URL에 대한 요청을 처리하는 JavaScript 함수 또는 컨트롤러를 연결 |
| /delete/:num | 게시글 삭제 자세히보기와 연관있는 URL에 대한 요청을 처리하는 JavaScript 함수 또는 컨트롤러를 연결 |

#### components

##### List.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 라이브러리의 React 객체와 React 컴포넌트 클래스 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableCell | Material-UI 라이브러리에서 제공하는 TableCell 컴포넌트 |
| ./Timestamp | 시간대를 표시하기 위한 Timestamp 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| Link from react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트 |
| ./Pagination | 페이지를 구현하기 위한 Pagination 컴포넌트 |





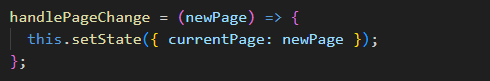
**: 서버로부터 게시글 목록을 가져오는 비동기 fetchPosts 메서드**

- fetch 함수를 사용하여 '/api/posts' 엔드포인트로 GET 요청

- 응답(response)을 JSON 형식으로 파싱하여 데이터를 추출

- setState 메서드를 사용하여 컴포넌트의 상태를 업데이트

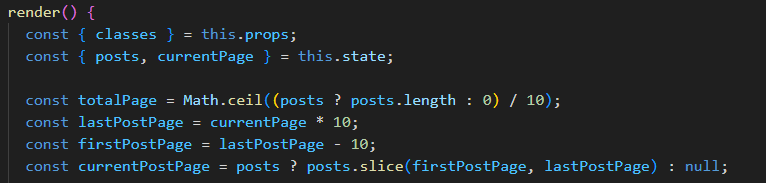
- 게시글 목록을 응답으로 받은 데이터로 설정



**: 페이지 변경이 발생했을 때 페이지 변화를 업데이트 하기위한 메서드**

- 새로운 페이지 번호를 인자로 받고, 현재 페이지 번호를 업데이트하기 위해 setState 메서드 사용

- 새로운 페이지 번호가 newPage로 전달되어 상태의 currentPage를 업데이트



**: List 컴포넌트의 렌더링 메서드(render)**

- totalPage : 전체 페이지 수를 계산하며, 게시글이 존재하면 posts.length를, 그렇지 않으면 0을 10으로 나누어 반올림(Math.ceil())한 값을 사용

- lastPostPage : 현재 페이지에서 마지막 게시글의 인덱스를 계산

- firstPostPage : 현재 페이지에서 첫 번째 게시글의 인덱스를 계산

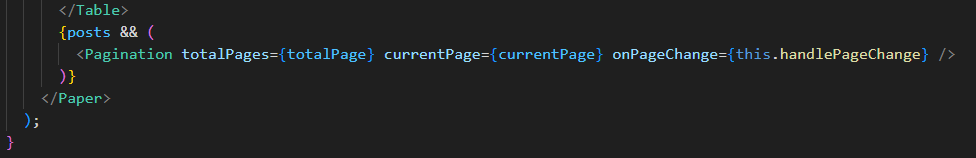
- currentPostPage : 현재 페이지에 해당하는 게시글 목록, posts.slice()를 사용하여 firstPostPage인덱스부터 lastPostPage인덱스 이전까지의 일부를 추출, 만약 posts가 존재하지 않는다면, null로 설정



**: currentPostPage 배열에 있는 각 요소 post에 대해 반복문을 실행**

- 각 셀에는 게시글의 번호, 제목, 작성자, 작성일을 표시

- 해당 제목을 클릭하면 게시글의 상세 페이지로 이동



**: 조건부 렌더링 구현**

- posts 배열이 존재할 때만 Pagination 컴포넌트 렌더링

- <Pagination … /> : Pagination 컴포넌트를 렌더링

- 총 페이지 수, 현재 페이지, 페이지 변경 핸들러(onPageChange) 전달

- 이를 통해 페이지 변경

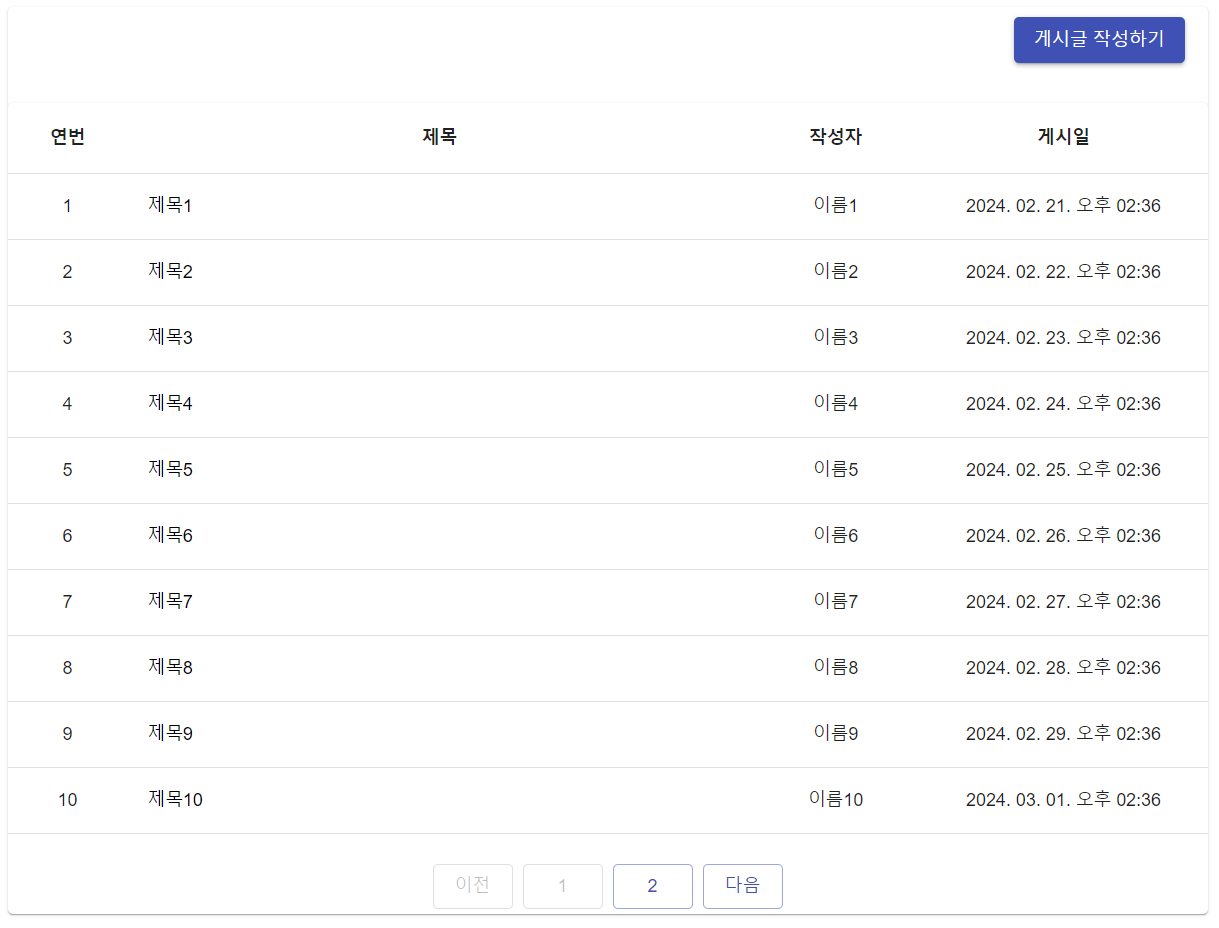
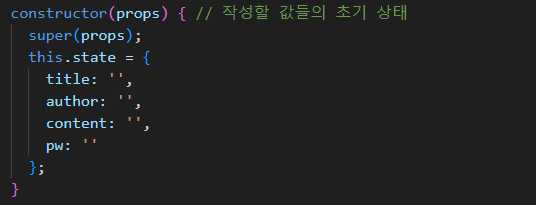
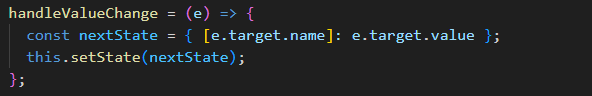


그림 2 - 게시판 리스트 (임의의 데이터 20개가 삽입된 상태)

##### Write.js

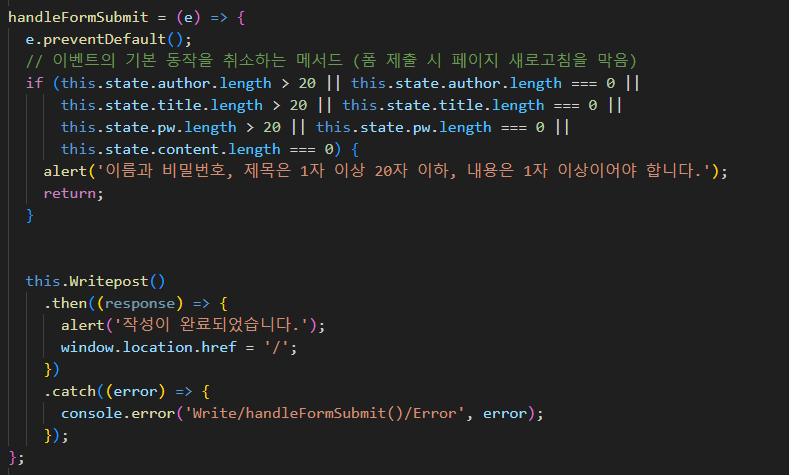
|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 라이브러리의 React 객체 |
| axios | HTTP 요청과 응답 처리를 위한 Axios 라이브러리의 axios 객체 |
| @material-ui/core/TextField | Material-UI 라이브러리에서 제공하는 TextField 컴포넌트 |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |
| Link from react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트 |





**: 입력 필드의 값이 변경될 때 호출되는 메서드**

- e.target.name을 사용하여 변경된 입력 필드의 이름에 해당하는 상태 값을 업데이트



**: 폼 제출 시 호출되는 메서드**

- if문을 이용해 입력 값의 유효성을 검사

- Writepost 메서드를 호출하여 서버로 데이터 전송

- POST 요청이 성공적으로 처리되면 .then() 블록을 실행하여 작성이 완료되었음을 알림

- POST 요청이 실패할 경우 .catch() 블록 실행



**: 게시글을 작성하기 위해 서버로 데이터를 전송하는 메서드**

- FormData 객체를 생성해 HTML 폼 데이터를 쉽게 조작하고 서버로 전송

- FormData에 ‘title’, ‘pw’, ‘author’, ‘content’ 값을 추가하고 this.state를 통해 컴포넌트 상태에서 가져옴

- Axios 요청에 대한 설정 객체를 ‘content-type': 'application/json’ 정의

- Axios의 post 메서드를 사용하여 POST 요청, URL, FormData 객체, 설정 객체 전달

- 이 메서드는 Promise를 반환하며, 요청이 성공하면 완료된 Promise를 반환

- 요청에 대한 응답은 handleFormSubmit 메서드의 .then() 블록에서 처리



**: 게시글을 작성하는 폼을 렌더링하는 render 메서드**

- TextField 컴포넌트를 사용하여 각각의 텍스트 입력 필드 이름(author), 비밀번호(pw), 제목(title), 내용(content)을 생성

- value 속성은 해당 필드의 값으로, onChange 속성를 사용해 입력 값이 변경될 때 handleValueChange 메서드 호출

- 작성 버튼을 눌렀을 때는 onClick 속성을 사용해 handleFormSubmit 메서드 호출

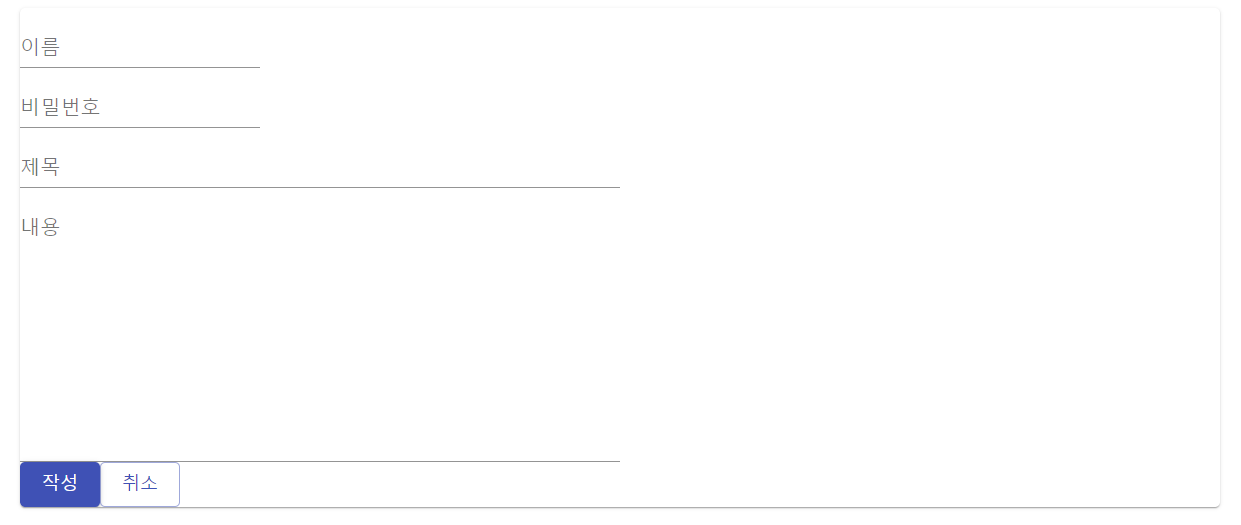


그림 3 - 작성 화면

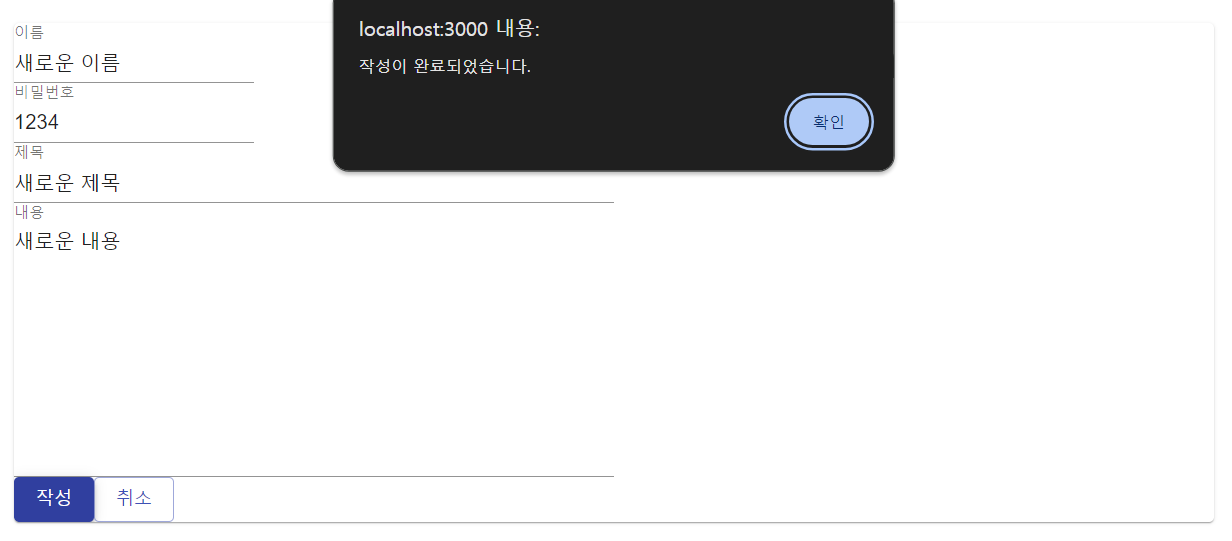


그림 4 - 새로운 데이터를 작성한 후 작성 버튼 클릭

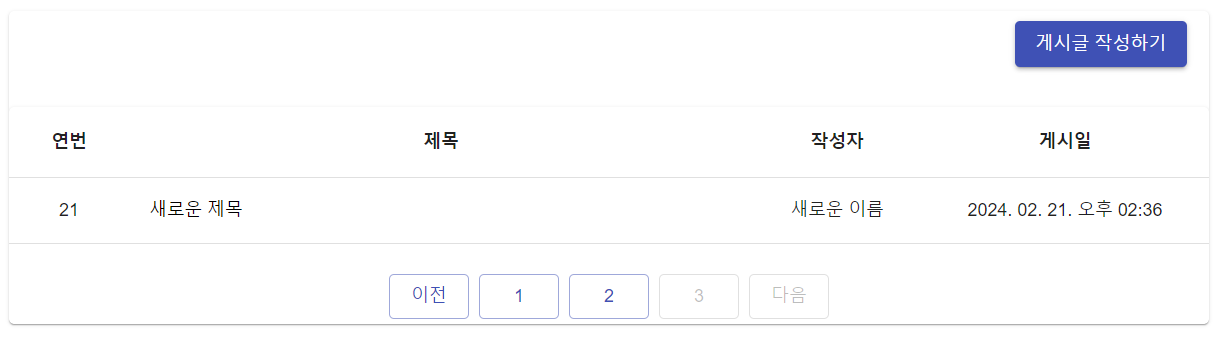


그림 5 - 새로운 페이지에 데이터가 추가된 화면

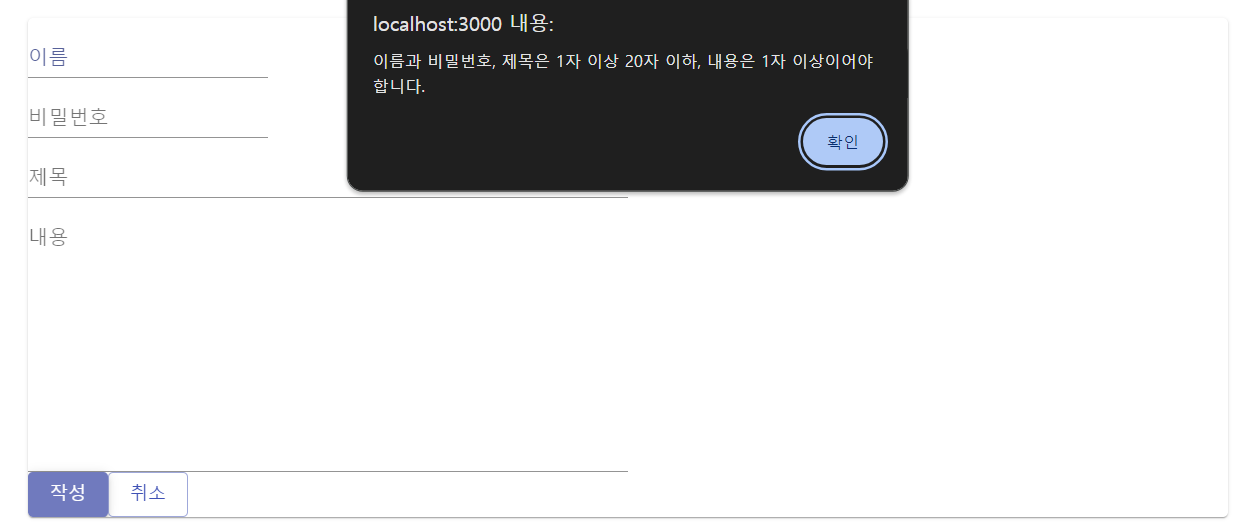


그림 6 - 내용을 입력하지 않고 작성 버튼 클릭 시

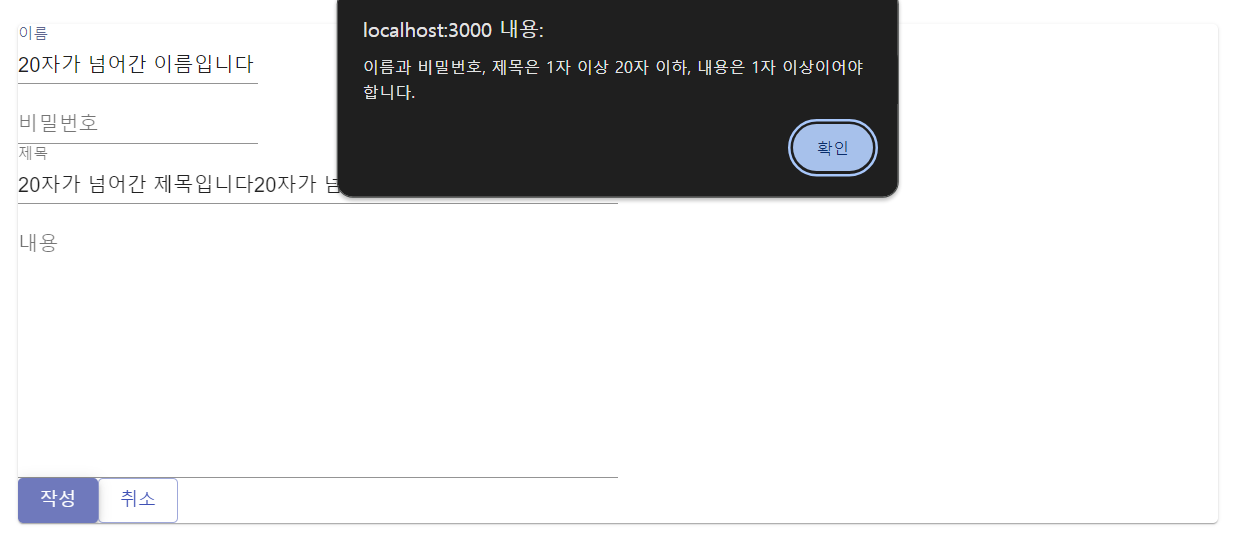
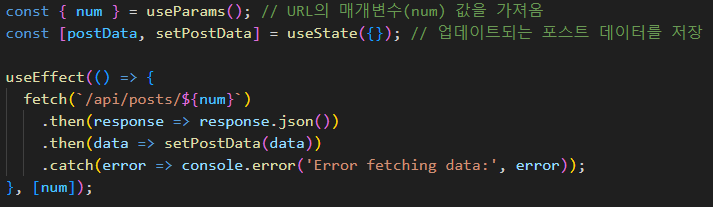


그림 7 - 입력한 내용이 20자가 넘었을 시

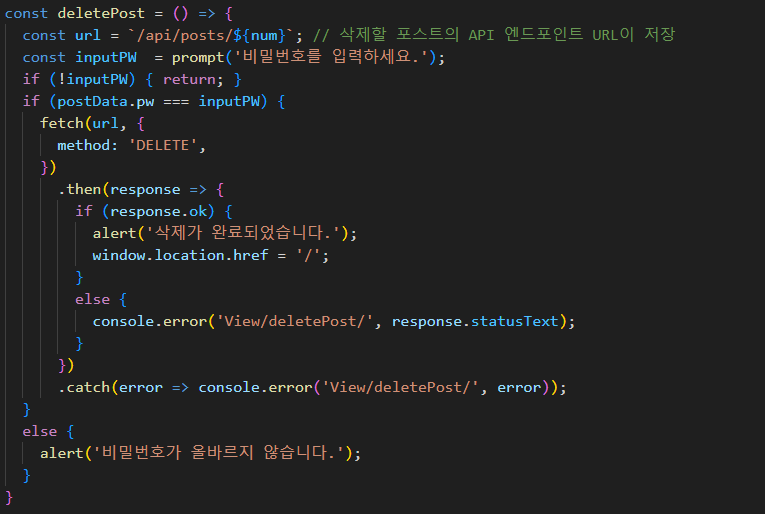
##### View.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 객체와 useEffect 함수, useState 함수 |
| react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트, useParams 훅 |
| @material-ui/core/TextField | Material-UI 라이브러리에서 제공하는 TextField 컴포넌트 |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| ./Timestamp | 시간대를 표시하기 위한 Timestamp 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |

****

**: 컴포넌트가 렌더링될 때와 num이 변경될 때마다 실행되는 효과를 정의**

- num이 변경될 때마다 새로운 포스트 데이터를 가져오기 위해 API로 요청

****

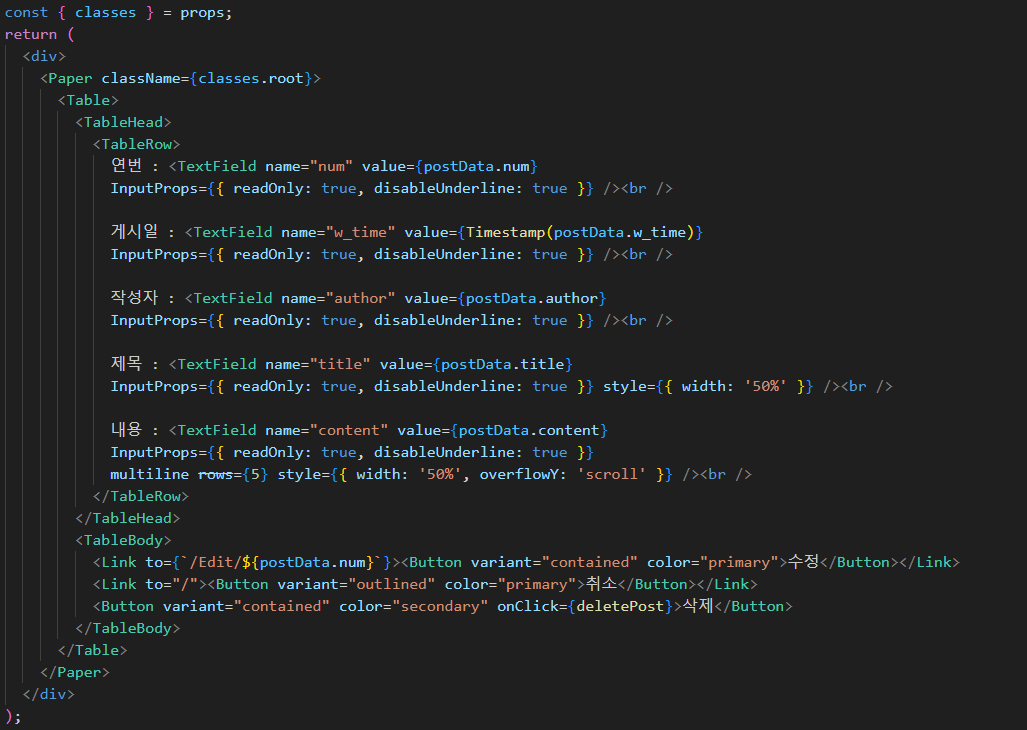
**: 게시글을 삭제하는 기능의 함수를 정의**

- inputPW : 사용자로부터 입력 받은 비밀번호 저장

- prompt() 함수를 사용하여 비밀번호를 입력

- 입력받은 비밀번호와 포스트의 비밀번호를 비교하여 일치하는지 확인

- fetch() 함수를 사용하여 서버에 해당 포스트를 삭제하는 DELETE 요청

****

**: <TableHead> : 각각의 <TextField>는 연번, 게시일, 작성자, 제목, 내용을 표시**

- 각각의 값은 postData 객체에서 가져와서 표시

- 수정할 수 없도록 InputProps를 통해 readOnly: true로 설정

- disableUnderline: true로 밑줄이 없도록 스타일 지정

**: <TableBody> : 수정 및 삭제 버튼이 포함**

- 포스트를 수정할 페이지로 이동시키는 링크 구현

- 삭제 버튼은 deletePost 함수를 호출하여 해당 포스트 삭제

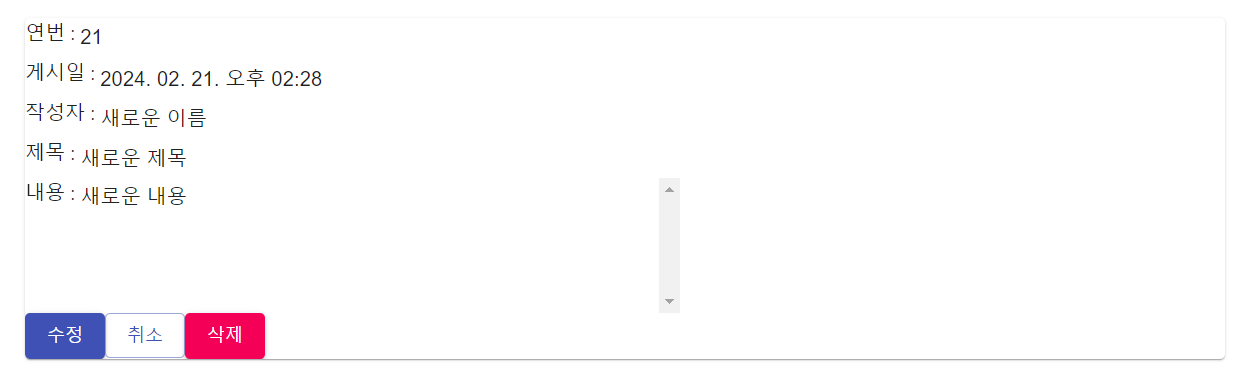
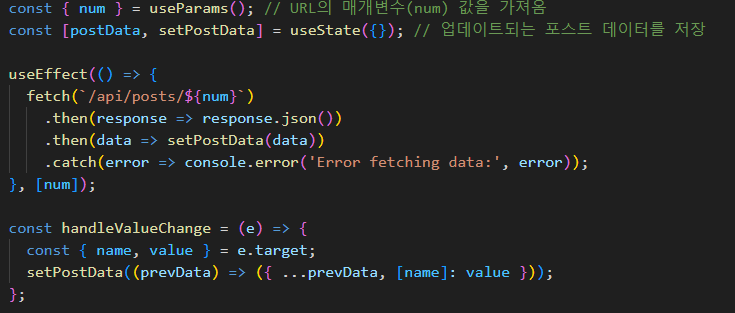


그림 8 - 상세 페이지 화면

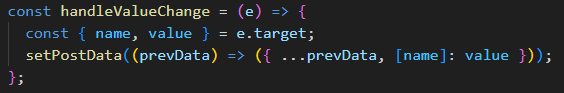
##### Edit.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 객체와 useEffect 함수, useState 함수 |
| react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트, useParams 훅 |
| @material-ui/core/TextField | Material-UI 라이브러리에서 제공하는 TextField 컴포넌트 |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| ./Timestamp | 시간대를 표시하기 위한 Timestamp 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |

****

**: 컴포넌트가 렌더링될 때와 num이 변경될 때마다 실행되는 효과를 정의**

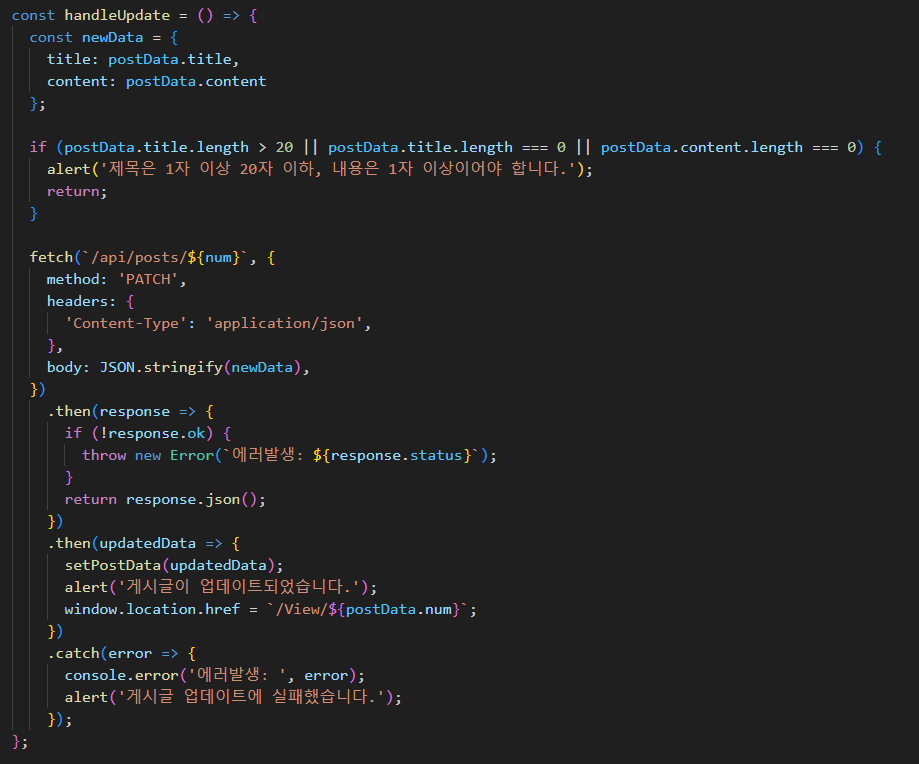
- num이 변경될 때마다 새로운 포스트 데이터를 가져오기 위해 API로 요청

****

**: input 요소의 값이 변경될 때 호출되며, 변경된 값을 상태에 반영하는 함수**

- e.target을 통해 변경된 input 요소의 이름(name)과 값(value)을 추출

- setPostData를 사용해 상태 업데이트

****

**: 게시글을 업데이트하는 함수**

- postData 객체에서 title과 content를 가져와 newData 객체에 저장

- 입력된 데이터가 유효한지 확인

- 유효성 검사를 통과한 경우, fetch 함수를 사용하여 서버로 PATCH 요청

- /api/posts/${num} 엔드포인트로 게시글 번호(num)를 포함하여 요청

- 요청은 JSON 형식으로 데이터를 전송

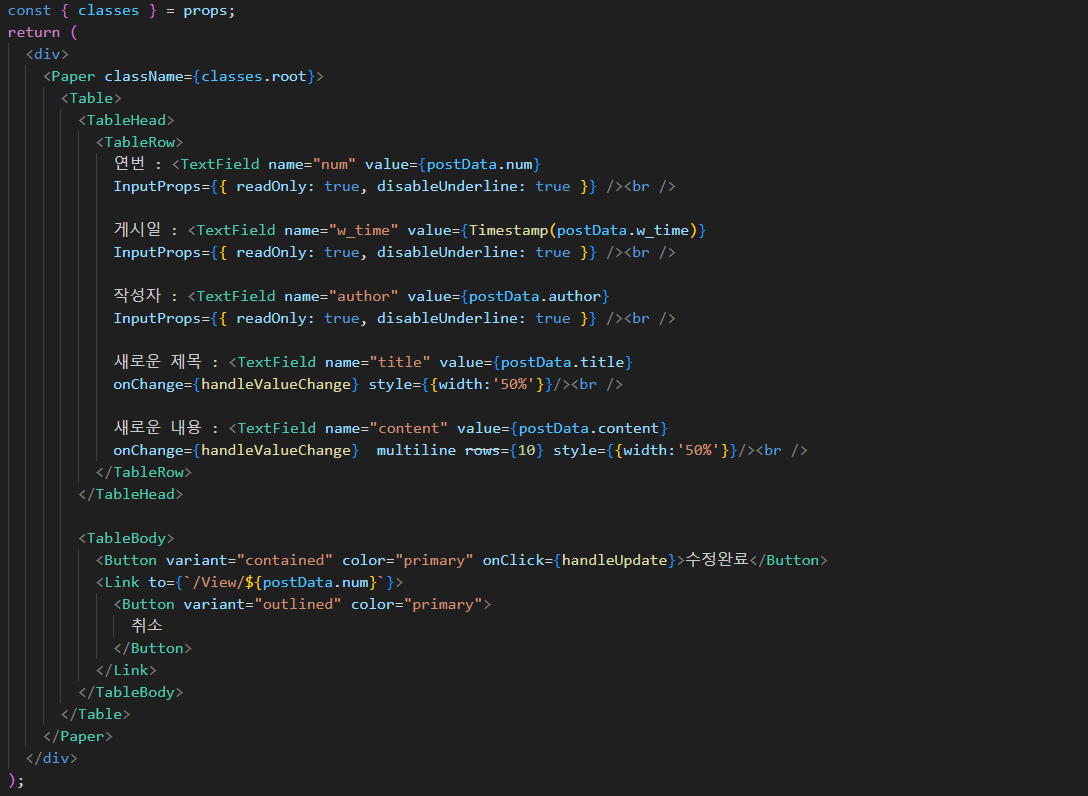
- 서버로부터 응답을 받은 후, 응답이 정상이면 JSON 형식으로 데이터를 추출

- 업데이트된 데이터를 상태에 설정

- 사용자에게 업데이트가 완료되었음을 알리는 메시지를 표시

- window.location.href를 사용하여 페이지를 다시 렌더링

- 업데이트된 게시글을 보여주는 페이지(/View/${postData.num})로 이동

****

**: <TableHead> : 각각의 <TextField>는 연번, 게시일, 작성자, 제목, 내용을 표시**

- 새로운 제목 및 내용은 게시글을 수정할 수 있는 TextField로 렌더링

- 값을 변경할 때마다 onChange 이벤트 핸들러가 handleValueChange 함수를 호출

- 상태를 업데이트하여 사용자가 입력한 값을 반영

- 수정완료 버튼을 클릭하면 handleUpdate 함수가 호출

- 수정된 데이터를 서버에 전송, 업데이트가 성공하면 사용자에게 알림 표시

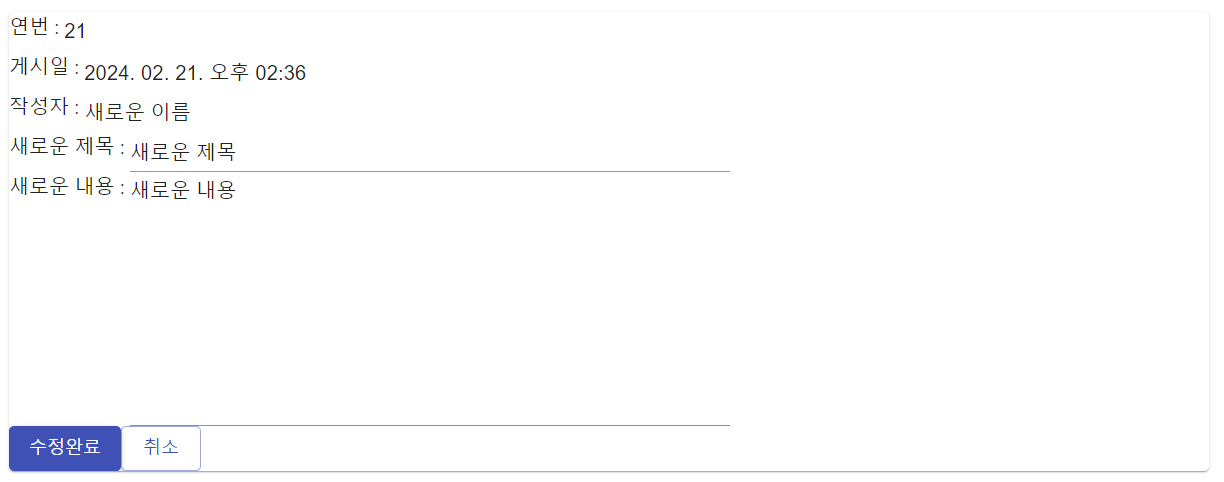


그림 9 - 수정 화면



그림 10 - 내용을 수정한 후 수정완료 버튼 클릭



그림 11 - 수정이 완료되면 상세 페이지로 자동 이동

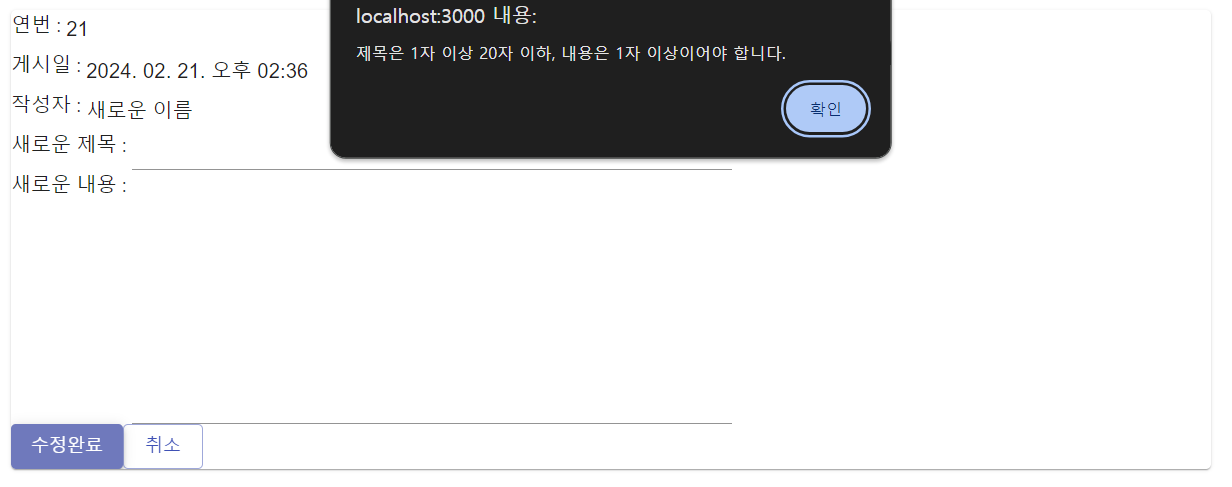


그림 12 - 수정 시에도 동일한 유효성 검사

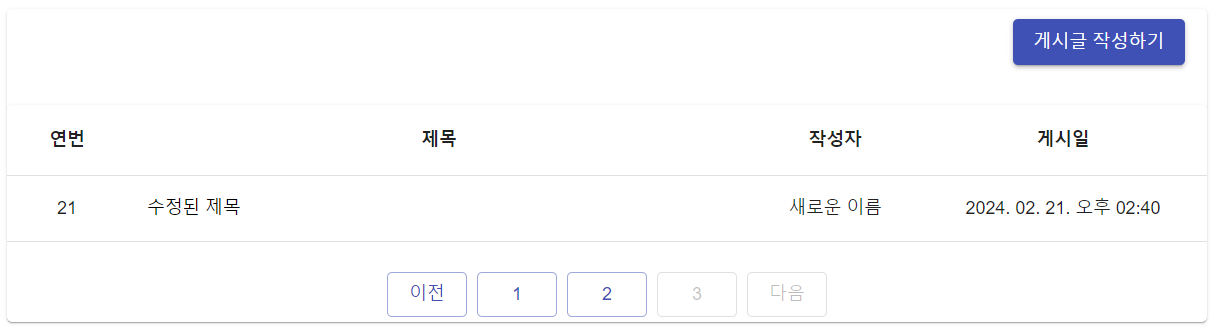
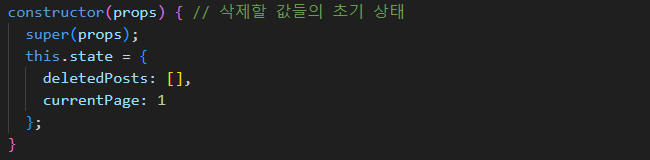
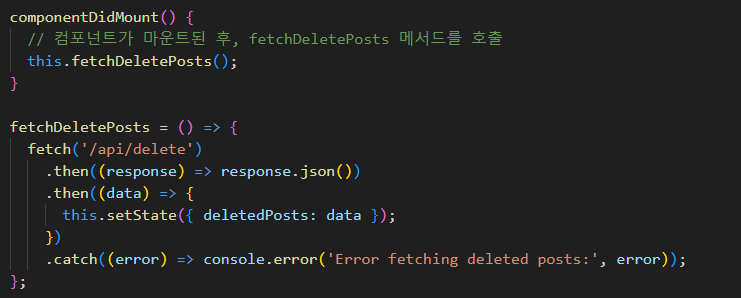


그림 13 - 수정이 완료되고 난 후 리스트 화면

##### Delete.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 라이브러리의 React 객체와 React 컴포넌트 클래스 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableCell | Material-UI 라이브러리에서 제공하는 TableCell 컴포넌트 |
| ./Timestamp | 시간대를 표시하기 위한 Timestamp 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| Link from react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트 |
| ./Pagination | 페이지를 구현하기 위한 Pagination 컴포넌트 |





**: React 클래스 컴포넌트에서 삭제된 게시물을 가져오는 메서드**

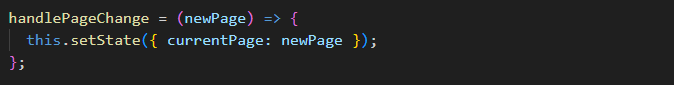
- 클라이언트 측에서 /api/delete 엔드포인트로 GET 요청

- 서버로부터 받은 응답을 JSON 형식으로 파싱

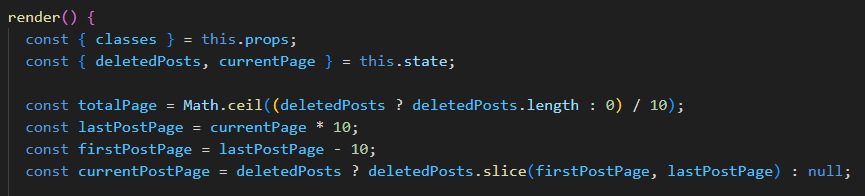
- 파싱된 데이터를 사용하여 컴포넌트 상태 업데이트

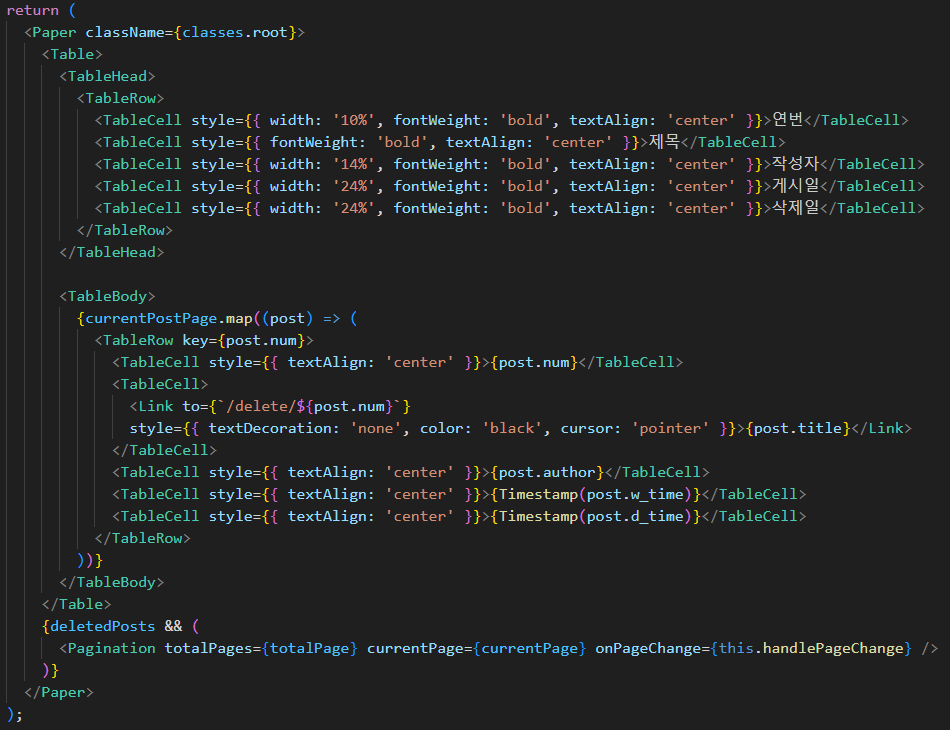
- setState 메서드를 사용하여 삭제된 게시글의 데이터 저장

- UI에 삭제된 게시글 표시



**: 페이지 변경을 처리하는 메서드**





**: 삭제된 게시물을 표시하기 위한 UI 렌더링**

- 핵심 기능과 메서드는 List.js의 return()과 동일

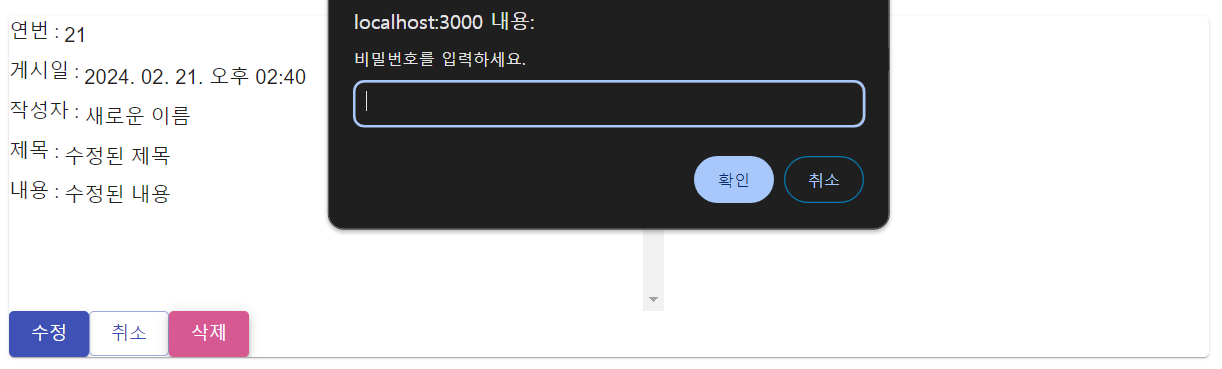


그림 14 - 삭제 시 비밀번호 입력

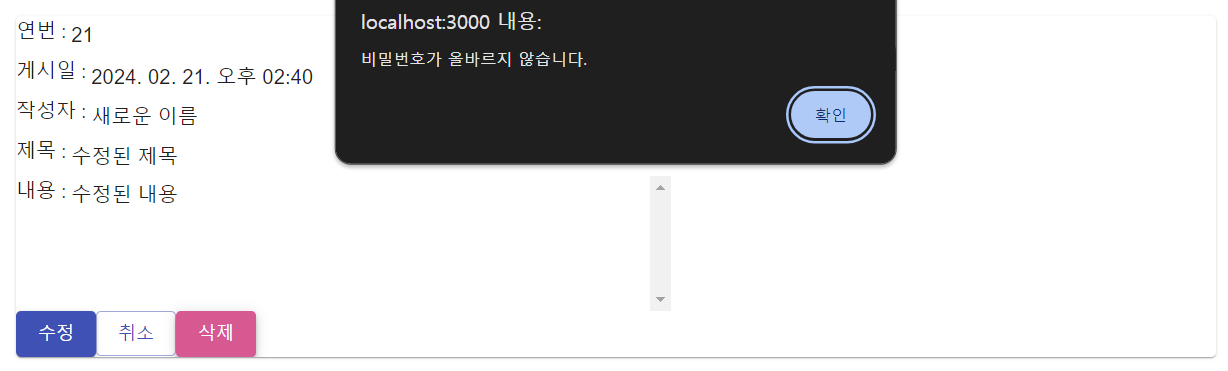


그림 15 - 비밀번호가 올바르지 않을 시

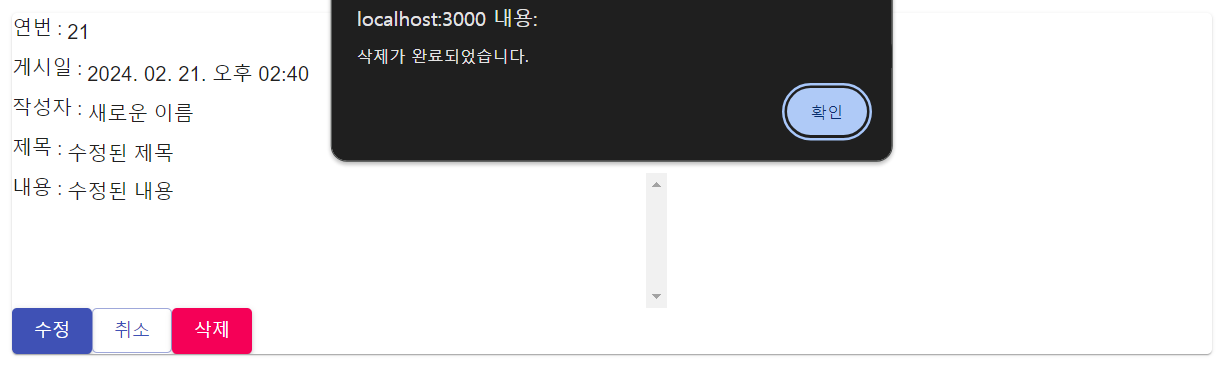


그림 16 - 비밀번호가 일치하여 삭제 완료

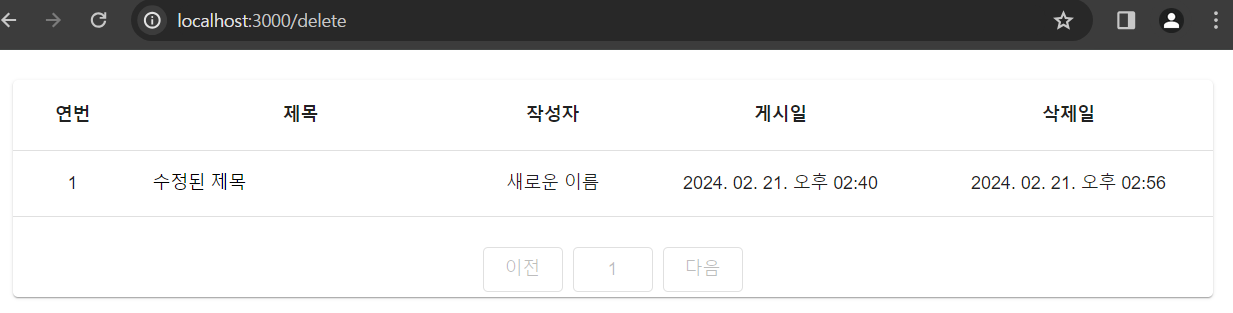
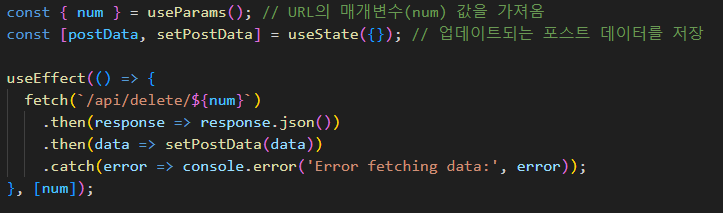


그림 17 - 삭제 후 삭제 게시글 보관 테이블로 이동

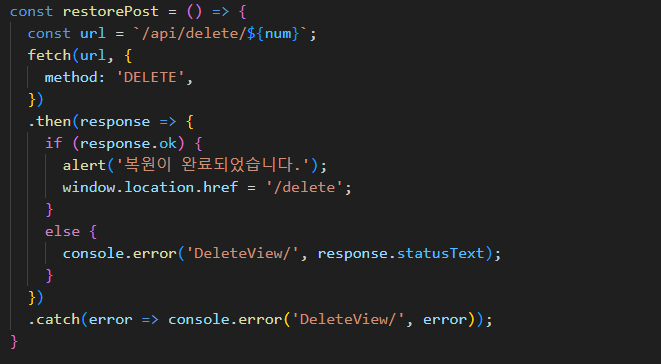
##### DeleteView.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | React 객체와 useEffect 함수, useState 함수 |
| react-router-dom | React Router 라이브러리에서 제공하는 Link 컴포넌트, useParams 훅 |
| @material-ui/core/TextField | Material-UI 라이브러리에서 제공하는 TextField 컴포넌트 |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |
| @material-ui/core/Paper | Material-UI 라이브러리에서 제공하는 Paper 컴포넌트 |
| @material-ui/core/Table | Material-UI 라이브러리에서 제공하는 Table 컴포넌트 |
| @material-ui/core/TableHead | Material-UI 라이브러리에서 제공하는 TableHead 컴포넌트 |
| @material-ui/core/TableRow | Material-UI 라이브러리에서 제공하는 TableRow 컴포넌트 |
| @material-ui/core/TableBody | Material-UI 라이브러리에서 제공하는 TableBody 컴포넌트 |
| ./Timestamp | 시간대를 표시하기 위한 Timestamp 컴포넌트 |
| ./withStyles | 컴포넌트에 스타일을 적용하기 위한withStyles 함수 |



**: 컴포넌트가 렌더링될 때와 num이 변경될 때마다 실행되는 효과를 정의**

- num이 변경될 때마다 새로운 포스트 데이터를 가져오기 위해 API로 요청



**: 게시글을 복원하는 함수**

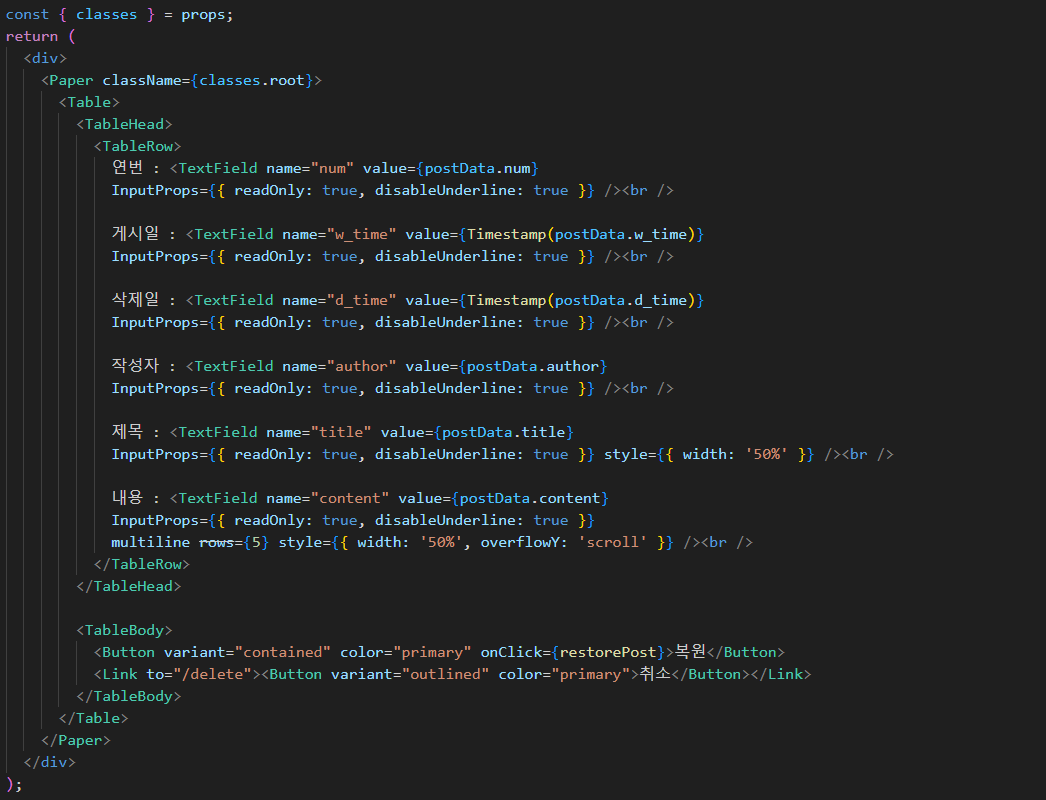
- 복원할 게시글의 엔드포인트 URL 생성

- num을 사용하여 복원할 게시글의 번호 포함

- 생성된 URL로 DELETE 요청하여 해당 게시글 삭제

- DELETE 요청에 대한 응답을 처리하여 응답이 성공적으로 이루어졌을 때 복원이 완료되었음을 알리는 알림 메시지 표시

- window.location.href를 사용하여 삭제된 게시글 목록 페이지(/delete)로 이동

****

**: DeleteView 컴포넌트에서 렌더링**

- 핵심 기능과 메서드는 View.js의 return()과 동일



그림 18 - 삭제된 게시글의 상세 페이지 화면

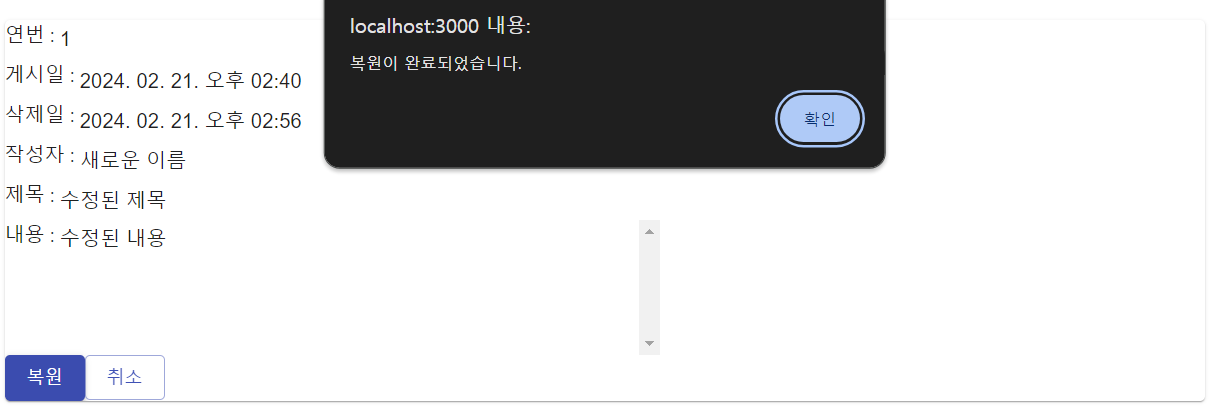


그림 19 - 복원 버튼 클릭 시

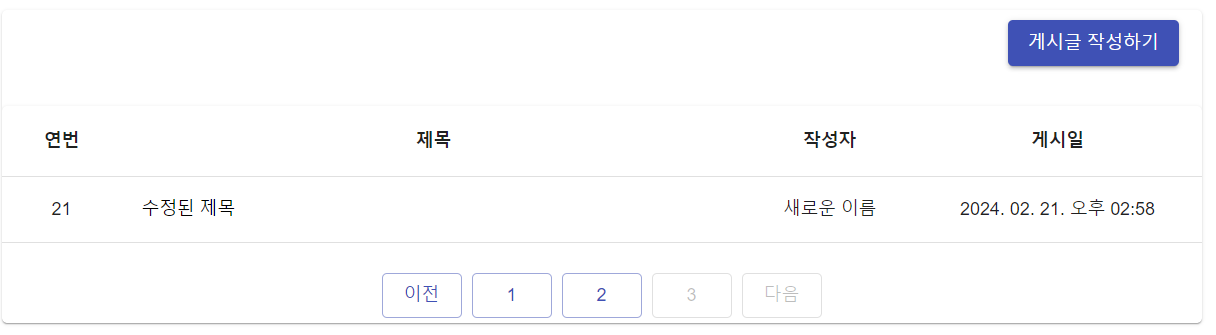
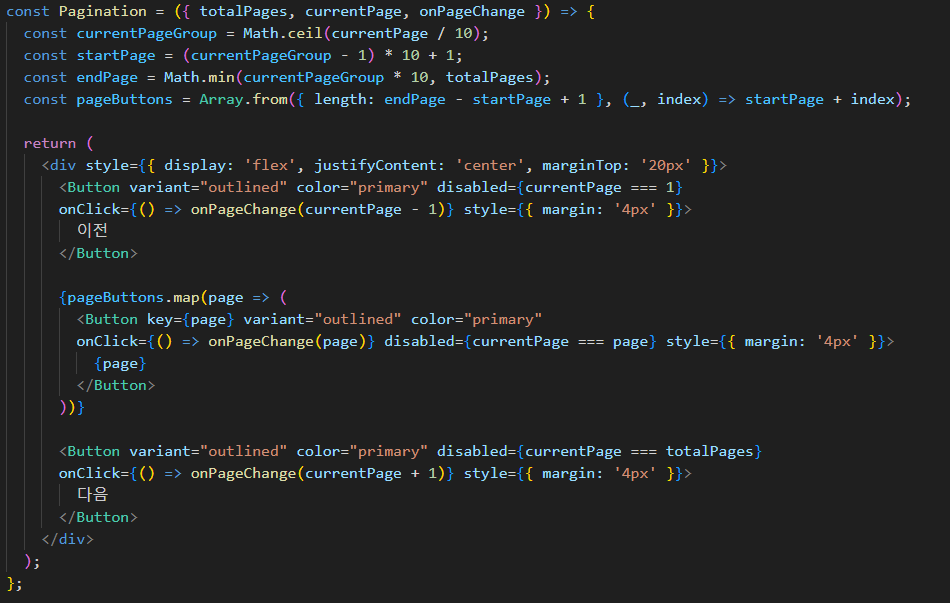


그림 20 - 복원이 완료된 리스트 화면

##### Pagination.js

|  |  |
| --- | --- |
| **import** | **설명** |
| react | 컨트롤 시프트 C, V |
| @material-ui/core/Button | Material-UI 라이브러리에서 제공하는 Button 컴포넌트 |



**: 페이지네이션을 구현하는 React 함수 컴포넌트**

- currentPageGroup : 현재 페이지가 속한 페이지 그룹 결정. 한 페이지 그룹에는 10개의 페이지가 포함되며, 현재 페이지를 10으로 나누어 올림한 값으로 페이지 그룹을 결정

- startPage : 현재 페이지 그룹에서 첫 번째 페이지의 번호를 결정. 각 페이지 그룹은 일반적으로 10개의 페이지로 구성되며, 현재 페이지 그룹 번호에서 1을 뺀 후 10을 곱하고 1을 더하여 첫 번째 페이지 번호를 계산

- endPage : 현재 페이지 그룹에서 마지막 페이지의 번호를 결정. 현재 페이지 그룹의 마지막 페이지 번호는 다음 페이지 그룹의 시작 페이지보다 하나 작거나 같으므로 currentPageGroup \* 10과 totalPages 중 작은 값을 선택하여 마지막 페이지 번호를 결정

- pageButtons : 시작 페이지부터 끝 페이지까지의 번호를 가진 페이지 버튼을 생성. Array.from() 메서드를 사용하여 범위 내의 숫자 배열 생성, 각 숫자에 대해 시작 페이지를 더하여 페이지 버튼 배열을 생성.

- return구문 : 페이지네이션 UI 렌더링, '이전' 및 '다음' 버튼을 누를 때마다 현재 페이지에서 -1 또는 +1씩 이동

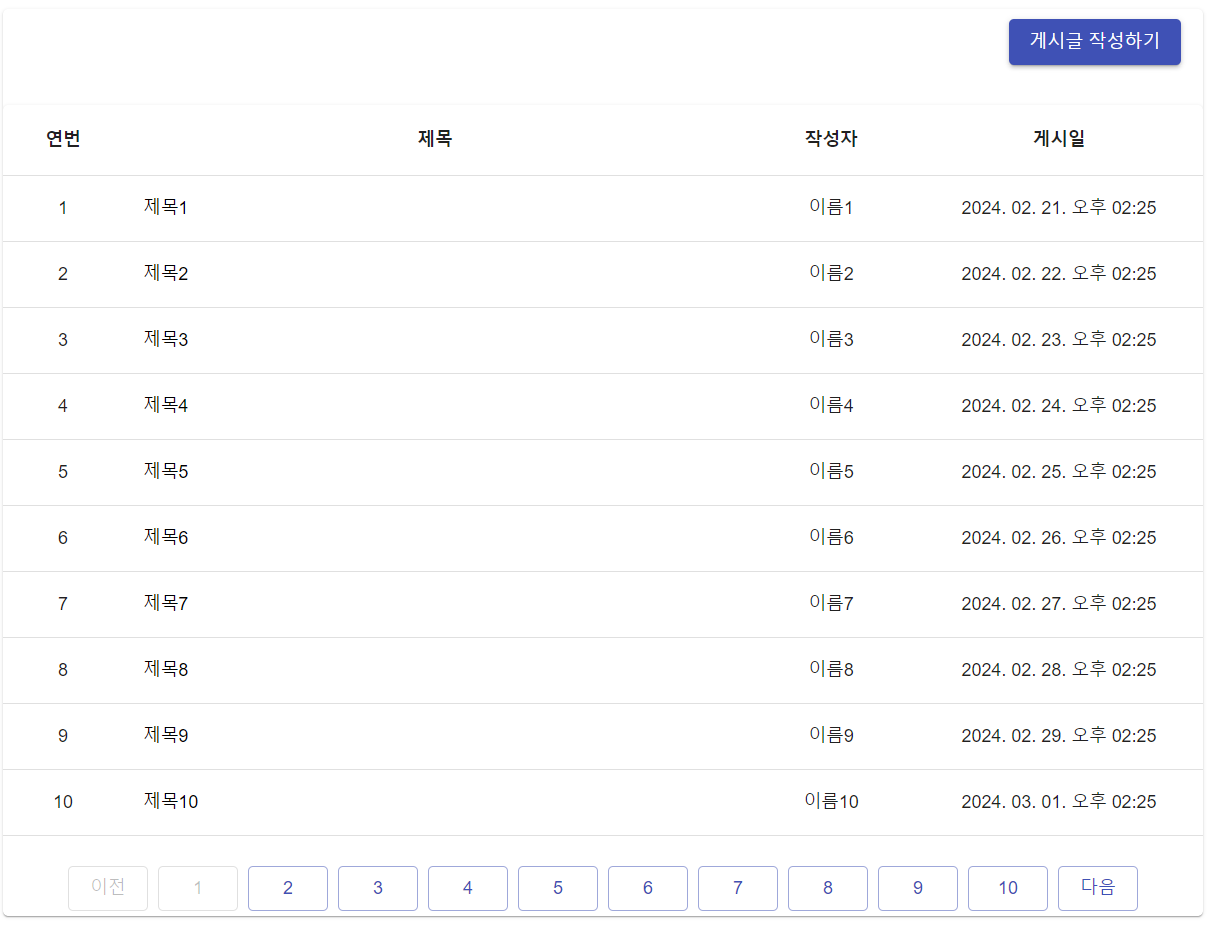


그림 21 - 게시글이 100개가 넘어가 페이지가 10페이지 이상으로 넘어갔을 시 1

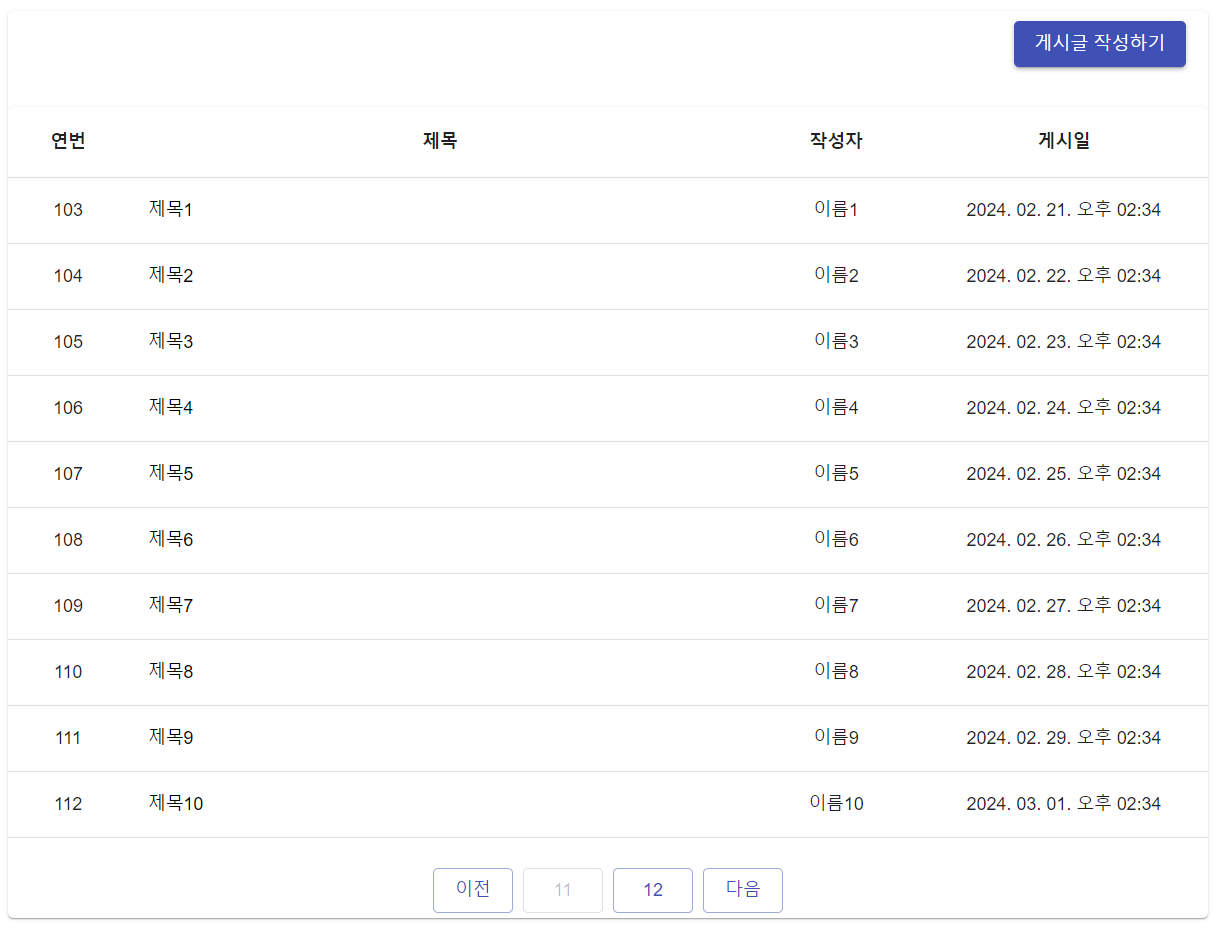
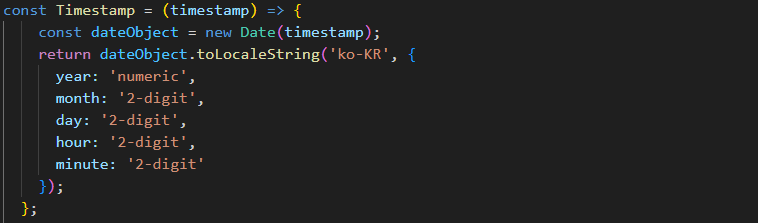


그림 22 - 게시글이 100개가 넘어가 페이지가 10페이지 이상으로 넘어갔을 시2

##### Timestamp.js



**: 타임스탬프를 받아와 한국 시간대에 맞는 형식으로 변환하는 함수**

- new Date(timestamp) : 입력된 타임스탬프를 사용하여 JavaScript의 Date 객체 생성

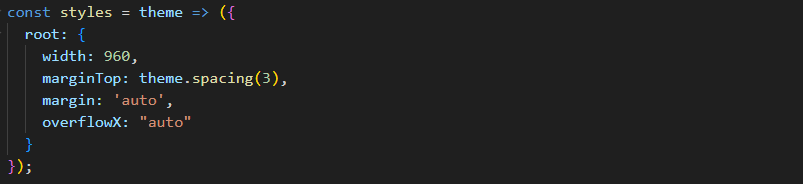
- toLocaleString 메서드를 사용하여 날짜와 시간을 한국 시간대에 맞는 로케일로 변환

- 위 코드에서는 한국 로케일인 'ko-KR'을 사용하였고, 이렇게 변환된 값은 '2024.02.21. 오후 03:45'와 같은 형식으로 반환

양식의 맨 위

##### withStyles.js

|  |  |
| --- | --- |
| **import** | **설명** |
| @material-ui/core/styles | Material-UI 라이브러리에서 제공하는 withStyles  컴포넌트 |



**: Material-UI에서 사용되는 스타일 객체 정의**

- 테마(theme) 객체를 매개변수로 받음

- 스타일 객체 내에 정의된 root 클래스는 컴포넌트의 루트 요소에 적용

- 루트 요소의 너비를 960픽셀로 지정

- 루트 요소의 위쪽 여백을 테마의 spacing 값을 3단위만큼 설정

- theme.spacing() 함수 : 테마에서 지정된 여백 값을 반환

- 루트 요소의 좌우 여백을 자동으로 설정하여 요소를 수평 가운데에 배치

- overflowX: "auto": 가로 방향으로 오버플로우가 발생할 경우 스크롤바 표시양식의 맨 위