웹 시스템 설계 HW#4

추민솔 TA
cmss1217@ajou.ac.kr

2025-11-25

목차

1 과제

1.1 목표

- React의 `useEffect`를 사용하여 렌더링 이후에 실행되는 동작을 구현하는 방법을 익힌다.
- MongoDB를 사용하는 `Backend` 구조를 이해하고, Document-oriented 데이터 모델을 경험한다.

1.2 과제 개요

- 본 과제는 **HW#3**의 확장 버전으로, 화면 구성과, 주요 기능(예약 조회, 생성, 삭제 등)은 그대로 유지한다.
- `Backend`는 기존의 **파일 기반 저장** 방식을 **MongoDB document 기반 저장**으로 변경하여, 동일한 REST API를 MongoDB collection과 연동되도록 구현한다.
- `Frontend`는 `useEffect`를 사용해 상태 바 숨김, 삭제 지연 등과 같은 렌더링 이후 동작을 상태 변화와 연결되도록 재구성한다.

1.3 과제 준비

이번 과제에서는 `MongoDB Community Edition`과 `MongoDB` 데이터를 시각적으로 확인하기 위한 GUI 도구인 `MongoDB Compass`의 사용을 권장한다. `MongoDB Compass`는 실제 데이터가 collection에 어떻게 저장되는지 등을 GUI에서 바로 확인할 수 있어 Backend 디버깅에 유용하다.

본 문서의 설치 안내는 `MongoDB` 공식 문서 *MongoDB Community Edition 설치*를 참고하여 작성하였다.
https://www.mongodb.com/ko-kr/docs/v7.0/administration/install-community/

macOS에서 설치

1. Homebrew가 설치되어 있지 않다면, https://brew.sh 를 참고하여 먼저 Homebrew를 설치한다.
2. 터미널을 열고, MongoDB Homebrew tap을 추가한다.

```
brew tap mongodb/brew
brew update
```

3. MongoDB 7.0 Community Edition을 설치한다.

```
brew install mongodb-community@7.0
```

4. MongoDB 서버를 macOS 서비스로 실행한다.

```
brew services start mongodb-community@7.0
```

5. 서비스 상태를 확인하고 싶다면 다음 명령으로 확인할 수 있다.

```
brew services list
```

6. MongoDB Compass를 설치한다.
   - MongoDB 공식 사이트에서 macOS용 `MongoDB Compass`를 다운로드하여 설치한다.
   - Compass를 실행한 뒤, `mongodb://127.0.0.1:27017` 으로 접속해 로컬 MongoDB 서버 연결이 정상적으로 이루어지는지 확인한다.

Windows에서 설치

Windows에서는 MongoDB 설치 마법사(`.msi`)를 이용해 MongoDB Community Edition과 MongoDB Compass를 설치한다.

1. MongoDB 다운로드 및 설치
   - `MongoDB Download Center`에서 `Community Server`를 선택한다.
   - Version은 7.0, Platform은 `Windows`, Package는 `msi`를 선택한 뒤 다운로드 한다.
   - 설치된 `.msi` 설치 마법사를 실행하여 설치를 완료한다.
2. 서비스 설정
   - `Service Configuration` 단계에서 `Install MongoD as a Service` 옵션을 선택한다.
   - 기본 서비스 이름(`MongoDB`)과 기본 데이터/로그 디렉토리는 그대로 사용해도 무방하다.
   - 이 옵션을 선택하면 Windows 시작 시 `mongod`가 자동으로 백그라운드에서 실행된다.
3. MongoDB Compass 설치
   - 설치 마법사에서 `Install MongoDB Compass` 옵션이 표시되면 그대로 두고 함께 설치한다. (표시되지 않는 경우, MongoDB 사이트에서 따로 다운로드해 설치해도 된다.)
4. 설치 완료 및 실행 확인
   - 설치가 완료되면 Windows의 `Services` 관리 도구에서 `MongoDB` 서비스가 실행 중인지 확인할 수 있다.
   - MongoDB Compass를 실행한 뒤, `mongodb://127.0.0.1:27017` 주소로 접속하여 로컬 MongoDB 서버에 정상적으로 연결되는지 확인한다.

1.4 과제 코드 구성

본 과제는 서버(`Backend`)와 클라이언트(`Frontend`)로 구성된다.

- `Backend`

  - `server.js` Express 기반 REST API 서버. MongoDB와 연결하여 비행편/예약 정보를 조회, 생성, 삭제한다.

- `Frontend`
  - `src/App.jsx` 클라이언트 최상위 페이지. 비행편/예약 상태를 관리하고, `useEffect`를 사용해 초기 데이터 로딩, 상태 바, 삭제 지연을 구현한다.
  - 하위 컴포넌트 (기존 HW#3 코드를 그대로 사용 or 수정 가능)
    * `src/components/FlightMeta.jsx` 비행편 메타 정보(편명, 시간 등) 표시.
    * `src/components/ReservationForm.jsx` 이름/좌석 입력, 형식/범위 검증, 제출 및 로딩 상태 관리.
    * `src/components/ReservationTable.jsx` 예약 목록 렌더링, 지연 삭제 및 삭제 취소 처리.
    * `src/index.css` 스타일시트. 전역 및 컴포넌트 스타일 정의.

1.5 구현 요구사항

본 과제에서는 `Frontend`, `Backend`의 구현이 요구되며, 일부 코드는 HW#3의 코드를 그대로 사용 가능하다.

1.5.1 클라이언트 구현 (`Frontend`)

클라이언트는 항공편 예약 데이터 loading과 예약 생성, 예약 목록 표시, 지연 삭제 및 삭제 취소, 상태 알림 표시까지 전체 상호작용을 담당한다. 구현 과정에서 HW#3에서 구현된 코드를 재사용할 수 있다.

각 항목 별 구현 사항

`src/App.jsx`

- 비행편 정보와 예약 목록을 서버에서 불러오는 초기 로딩 로직은 `useEffect`를 사용해 컴포넌트 마운트 시 한 번 실행되도록 구현한다.
- 상태 바(StatusBar)에 표기할 메시지 상태를 관리하고, `useEffect`를 사용하여 성공/실패 등 메시지가 표시된 뒤 일정 시간이 지나면 자동으로 상태 바가 숨겨지도록 구현한다.
- 예약 삭제 버튼을 클릭했을 때 즉시 삭제하지 않고, 삭제까지 남은 시간을 상태로 관리한 뒤, `useEffect`를 사용하여 남은 시간을 감소시키고, 시간이 만료되면 실제 삭제 요청을 보내도록 구현한다.
- 삭제 대기 상태에서 삭제 취소 버튼을 클릭하면, 해당 예약에 대해 대기 중이던 삭제가 취소되고, 목록이 원래대로 유지되도록 구현한다.

1.5.2 서버 구현 (`Backend`)

서버는 이전 과제에서 구현한 Express 기반 REST API를 바탕으로 하며, Route 경로와, 응답 형식은 그대로 유지하여도 된다. 단, 데이터 관리를 기존 파일 기반에서 MongoDB 기반으로 변경해야 한다.

각 항목 별 구현 사항

MongoDB 연동

- mongodb 드라이버를 사용하여 MongoDB에 접속하고, 하나의 데이터베이스 안에 비행편 collection(flight)와 예약 collection(reservations)를 사용한다.
- 서버 코드에서는 process.env.MONGO_URL, process.env.DB_NAME을 사용하여 접속 정보를 읽어야 한다. 이때 환경 변수는 OS에 직접 설정해도 되지만, 편의를 위해 .env 파일과 dotenv 패키지를 사용 가능하다.
- Backend 디렉토리에 .env 파일을 생성하여, 아래와 같이 환경 변수를 정의한다.

```
# .env
MONGO_URL=mongodb://127.0.0.1:27017     # MongoDB Server Connection URL (Example)
DB_NAME=flightapp                       # Database Name
```

- dotenv 패키지를 사용해 .env 파일을 로드해도 된다. 예를 들어, 다음과 같이 설정할 수 있다.

```
// server.js (Example)

// .env 파일을 읽어오기 위한 dotenv 패키지를 불러온다.
const dotenv = require("dotenv");

// 현재 프로세스에서 .env 내용을 읽어 process.env.*에 채워넣는다.
dotenv.config();

// 이후에는 process.env.변수명 형태로 .env 파일에 정의한 값을 읽을 수 있다.
const MONGO_URL = process.env.MONGO_URL;
const DB_NAME  = process.env.DB_NAME;
```

- 서버 시작 시 MongoDB에 연결하고, 이후 각 route에서 해당 collection을 사용해 예약 조회/생성/삭제를 수행하도록 한다.

REST API Route

- GET /api/flight
  MongoDB의 flight collection에서 한 개의 flight document를 조회하여 JSON으로 반환한다.
- GET /api/reservations
  reservations collection에서 모든 예약 document를 조회하여 배열로 반환한다.
- POST /api/reservations
  요청 본문으로 전달된 이름과 좌석 값을 검증하고, 유효한 값일 경우, 새 예약을 MongoDB의 reservations collection에 추가한다.
- DELETE /api/reservations/:id
  파라미터로 전달된 id에 해당하는 예약을 reservations collection에서 삭제한다.

1.6 제출물 및 제출 요구사항

- 구현 코드는 `node_modules` 파일 삭제 후 `Backend`, `Frontend` 모두 제출
- 과제 보고서 (`report.pdf`)
  - `Frontend` - 렌더링, `useState`, `useEffect` 간 상태/데이터 연결 방식
  - `Backend` - `Express` <-> `MongoDB` 연동 방식
  - 참고한 사이트 목록 (링크 포함).
  - Generative AI 사용 명시: 과제 구현에 있어 AI를 사용했다면 어떤 질문을 했고, 어떤 답변을 받았으며, 코드의 어떤 부분에 사용하였는지 구체적으로 명시.
- 제출물은 코드와 보고서를 HW4_학번_이름.zip 형식으로 압축하여 제출 (미 준수 시 감점 가능)

```
HW4_202301234_홍길동.zip/
├── report.pdf
├── backend/
│   ├── server.js
│   ├── package.json
│   └── .env
└── frontend/
    └── node_modules 폴더를 제외한 전체 코드
```

- 제출 기한: 12/02 (화) 23:59 (지각 제출 시 D+1 -40%, D+2 -70%, D+3 -100% 감점)