



헬스장 관리 시스템 - React

기간	2025.12.30 ~ 2024.01.12
사용 기술	Axios CSS3 Git JSP Java JavaScript MyBartis MySQL Node.js React Spring boot SpringSecurity Swagger VisualStudioCode



프로젝트 주제



스마트 헬스장 통합관리 시스템 - React

주요 기능



- PT 예약 및 운동 계획표
- 실시간 회원 등록
- QR 코드를 통한 출입 관리 시스템
- 스마트 기술 기반 효율적 운영

기존 MVC 패턴으로 제작한 프로젝트를 React 및 Restful api로 전환



기존 MVC 패턴에서 REST API 패턴으로 전환해 클라이언트와 서버를 완전히 분리했습니다.

Controller에서 restController로 변경하여 View를 렌더링하여 반환하는 방식에서 데이터만 반환하고 클라이언트가 해당 데이터를 사용하여 화면을 동적으로 구성하는 방식으로 전환

기술 및 구현 방법

컴포넌트별 스타일 충돌을 방지하고 유지 보수성을 높이기 위해 모듈화를 진행

```
<div class="centerdhkm">
  <div class="input-group my-2">
    <label for="password">비밀번호</label>
    <input type="password" class="form-control" id="password" name="password" placeholder="비밀번호" />
  </div>

  <div style="margin-top: 5px;" class="input-group my-2">
    <label for="passwordCheck">비밀번호 확인</label>
    <input type="password" class="form-control" id="passwordCheck" name="passwordCheck" placeholder="비밀번호 확인" />
  </div>
```



```
return (
  <body className={`${styles.body}`}>

  <div className={`${styles.container1}`}> </div>
  <div className="py-4 px-5 text-center">
    
    <h1 className={styles.h1}>비밀번호 변경</h1>
  </div>

  <form id="form" onSubmit={(e) => newPassword(e)}>
    <div className={styles.centerdhkm}>
```

• CSS 모듈 적용

컴포넌트별 스타일 충돌을 방지하고 유지 보수성을 높이기 위해 css 모듈화 진행

담당 역할 및 어필

TEAM MEMBERS



김도현

- 회원 관리
- 문의사항 게시판



오승원

- 메인화면 구성
- 출석체크 (QR)
- 출석내역
- 출석랭킹
- 이용권내역



이세진

- 이용권(상품)
- 트레이너 프로필
- 결제
- 구매내역
- 배출조회



조하은

- 회원 일정 관리
- 트레이너 코멘트 관리
- 캘린더 API 연동



홍성운

- 예약 기능
- 캘린더 API 연동
- 관리자 페이지 레이아웃

회원관리, 문의사항 게시판, 커스텀 에러페이지

회원관리

DETAIL OF THIS PAGE 053

로그인 및 회원가입



회원가입

- 회원의 입력 정보를 전달받아 DB에 등록
유효성 검사를 실시해 중복된 계정이나
잘못된 정보가 들어가지 않도록 설계

로그인

- 회원이 입력한 정보와 DB의 정보를 비교하여
유효한 회원일시 계정 정보를 JWT토큰에 저장해
로그인 상태를 유지

Rest API로 전환하며 서버와 클라이언트를 분리하며 유저 정보를 세션에 저장할수 없는 만큼 유저 정보를 JWT(JSON Web Token)에 담아 클라이언트측에 저장하고 서버와 통신하며 데이터를 전달해 로그인 여부를 확인하게 설계했습니다.

```
// 쿠키에 저장된 jwt 를 읽어와서 로그인 처리
const autoLogin = async () => {
  // 로딩 시작
  setIsLoading(true);

  // 쿠키에서 jwt 가져오기
  const jwt = Cookies.get("jwt") || localStorage.getItem("jwt");

  if (!jwt) {
    // jwt가 있는 경우 바로 로딩 종료
    setIsLoading(false);
    return;
  }

  const authorization = `Bearer ${jwt}`;
  api.defaults.headers.common.Authorization = authorization;

  try {
    // 사용자 정보 요청
    const response = await auth.info();

    if (response.data === 'UNAUTHORIZED' || response.status === 401) {
      console.error('jwt가 만료되었거나 인증에 실패하였습니다.');
      logoutSetting(); // 인증 실패 시 로그아웃 처리
      setIsLoading(false); // 로딩 종료
      return;
    }
  } catch (error) {
    // 인증 실패: 로그인 세팅
    loginSetting(authorization, response.data);
  }
}
```

```
useEffect(() => {
  if (!isLoading) {
    // 로딩 중일 때는 아무 동작도 하지 않음
    return;
  }

  // 로딩 완료 후 로그인 여부 확인
  if (!isLogin) {
    Swal.alert('로그인을 시도해주세요', '로그인 화면으로 이동합니다.', 'warning', () => {
      navigate('/login');
    });
    return;
  }

  // 로그인되어 있다면 userInfo를 확인
  if (userInfo && userInfo.no) {
    setUserNo(userInfo.no);
    console.log(`userInfo.no: ${userInfo.no}`);
  } else {
    console.log(`userInfo가 없거나 userInfo.no가 없습니다.`);
  }
}, [isLoading, userInfo, navigate]);
```

React 같은 경우 클라이언트측 토큰이 전달받기전에 화면이 먼저 구현되어 데이터를 제대로 전달받지 못하는 경우가 발생하기에 jwt토큰을 가져오기 전까지 로딩이 진행되고 jwt 토큰을 전달 받은 후에 화면이 구현되게 설계하였습니다.

```

import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vite.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    // 프록시 설정
    proxy: {
      '/api': {
        target: 'http://localhost:8080', // (port) 서버 주소
        changeOrigin: true,           // 요청헤더의 Host 를 변경
        secure: false,               // https 지원 여부
        rewrite: (path) => path.replace(/^\api/, '')
      }
    }
  }
})

```

```

// 회원 목록
export const list = (keyword, page) => api.get('/user/list?keyword=${keyword}&pageNumber=${page}')

// 회원가입
export const join = (data) => api.post('/user', data)

// 로그인
export const login = (id, password) => {
  return api.get(`/login?id=${id}&password=${password}`)
}

// 회원 정보
export const info = () => api.get('/user/info')

// 회원 조회
export const select = (no) => api.get('/admin/update/${no}')

// 회원 수정
export const update = (data) => api.put('/user', data)

// 회원 정보 수정(아드민)
export const adminUpdate = (data) => api.put('/admin/update', data)

// 회원 탈퇴
export const remove = (no) => api.delete('/user/${no}')

// [아이디] 중복검사
export const checkId = (id) => api.get('/check/${id}')

```

서버와 연결은 proxy 설정을 통해 5173 포트에서 8080포트 서버로 요청 할 수 있도록 설정 하였습니다.

The screenshot shows the Swagger UI interface with two main sections: **user-controller** and **login-controller**.

- user-controller:**
 - PUT /users**
 - POST /users**
 - GET /users/info**
 - DELETE /users/{username}**
- login-controller:**
 - POST /login**
 - GET /user**

A dropdown menu at the top left shows the generated server URL: [http://localhost:8080 - Generated server url](http://localhost:8080).

클라이언트와 서버간 문제 발생시 확인이 어려운 경우에는 swagger api를 활용해 테스트 하며 문제점을 찾아 진행하였습니다.

아이디 및 비밀번호 찾기

- 회원이 입력한 정보를 토대로 DB 정보와 비교해 아이디를 찾을 수 있다.
- 비밀번호 찾기
- 회원이 입력한 정보를 토대로 DB와 비교 후 정보가 일치하면 비밀번호 변경 폼에 이동
- 페이지 이동시 보안을 위해 비밀번호 토큰을 생성 토큰이 없으면 입력한 정보가 맞더라도 변경 불가

마이페이지(내 정보)

내 정보(회원정보) → 회원정보 수정 → 비밀번호 변경

회원이 입력한 정보들을 한눈에 확인할 수 있는 메인화면

수정을 누른 다음 활용해 비밀번호를 이 곳에서 회원정보를 수정해 쉽게 할당

비밀번호 변경

비밀번호 변경 시에는 같은 비밀번호로는 사용할 수 없으며 이전 비밀번호와 새 비밀번호가 동일하거나 비밀번호 변경은 최근 30일 내로는 적용하지 않기 때문에 변경이 가능하도록 구현했습니다.

회원 목록 및 회원 정보 수정

회원목록

- 홈페이지를 기반한 회원 목록을 확인할 수 있는 페이지
- 관리자 회원정보 수정 및 삭제
- 회원 정보를 수정 및 삭제할 수 있는 관리자 버튼
- 회원과 트레이너나 관리를 분류할 수 있다.

고객문의 게시판

게시글 목록

- 회원이 게시글을 받은 경우 답변(답변 한회)은 표시
- 게시판에 올라온 글을 확인할 수 있도록 처리
- 답변을 적용하여 게시글의 작성기준, 게시글 개수 선, 검색 기능을 통해 원하는 내용을 확인할 수 있도록 설계

문의사항 C.R.U.D

게시글 C.R.U.D

- 원하는 내용을 입력하여 게시글을 작성해 다른 사람에게 알리거나 자신의 게시글을 확인할 수 있음
- 자신이 작성한 게시글에 좋아요를 달면도 함께 확인됨

답변 C.R.U.D

- 답변은 C.R.U.D 작업을 이용하여 게시글의 데이터 새로고침 없이도 실시간으로 데이터를 업데이트할 수 있도록 구현해
- 제공된 툴을 활용해 데이터를 수정하거나 삭제

마이페이지(문의내역)

내 문의내역

- 원하는 작성한 문의글을 원하는 차례로 조회할 수 있는 페이지
- 답변 내용 확인이 가능한 헤더 글을 클릭시 해당 글의 상세 페이지로 이동하고 해당 페이지에서 분의도 가능하게 설계

이외의 내용은 이전 프로젝트와 동일합니다.

프로젝트 개발환경 및 툴

개발환경 (LANGUAGE&TOOL)

Spring Boot

Spring SECURITY

Java

MySQL

GitHub

node.js

Gradle

HTML5

React

개발환경 React, node.js 추가

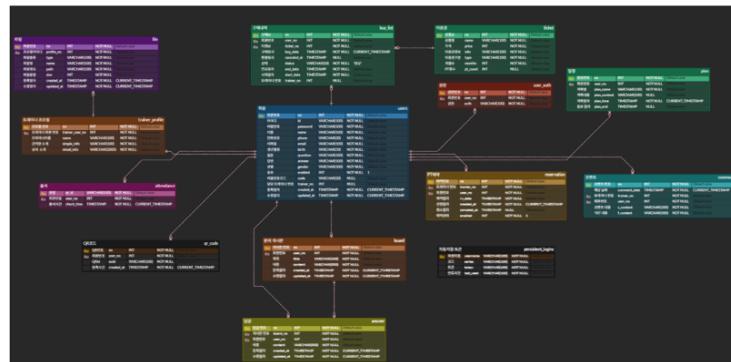
기술 스택

- 사용 언어
 - HTML5, CSS3, JavaScript, Java, SQL
- 서버 런타임 환경
 - Node.js
- 프레임워크
 - React, Bootstrap, Spring Boot, Spring Security, MyBatis
- 개발 도구
 - Visual Studio Code, MySQL Workbench
- 라이브러리
 - Jquery, Axios, Lombok, Devtool
- DB
 - MySQL
- 참조 API
 - SweetAlert, jwt, Springdoc
- 협업 Tools
 - Trello, GitHub, GoogleDrive



프로젝트 설계 ERD

ERD



프로젝트 보완 및 개선사항

- 소비자가 PT예약, 등록, 문의게시판 답변 등록 등 사용자가 알 수 있도록 알림 기능 추가 필요
- 소셜 로그인 기능 추가 필요
- 트레이너와 원활한 소통을 위한 채팅 기능 추가 필요
- 초반에 설계한 친구 시스템 미구현으로 기능 추가 필요



느낀점

- 프로젝트를 진행하면서 스크립트 및 API에 대한 이해도가 많이 부족하다는 것을 느꼈고 이로인해 구현하고 싶은 기능들을 많았지만 제대로 구현하지 못해 아쉬었던 것 같습니다.
- 추후 스크립트와 API 관련해 익숙치 않은 부분들에 대해 더욱 다양한 코드를 작성하며 적응해가야 할 것 같습니다.
- 이번 프로젝트를 진행하며 초반 프로젝트 구상을 탄탄히 하더라도 개발 과정에서 수정 사항이 생길 수밖에 없다는 것을 깨닫게 되었고 또한 팀원들 간 활발한 의사소통으로 각자 잘 모르는 부분을 공유하며 안되던 기능을 하나씩 구현해나가는 성취감을 느낄 수 있었던 값진 경험을하게 된 것 같습니다.