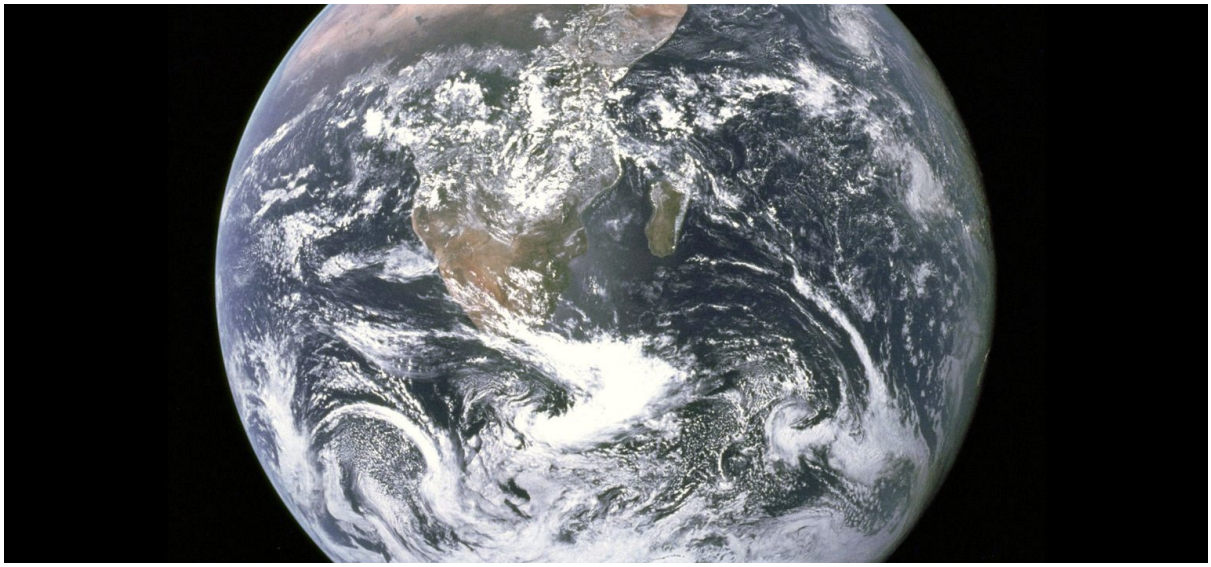


결과보고서

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/deac6e6b-fdfa-49eb-a869-adde4e63d510/4._2021-2_스터디상생플러스_결과보고서.hwp

1. 연구 개요

▼ 목적 및 필요성

본 프로젝트는 물품 대여 서비스를 기반으로 한 웹 사이트를 제작하는 것이 목표이다. 기존의 일반적인 방법으로는 대여 물품을 수기로 작성함으로써 겪는 불편함이 있는 것이 일반적이다. 그렇게 되면 실수를 하기에 쉬운 조건이기 때문에 대여자와 관리자 사이에 제대로 된 소통의 문제가 불가피하다. 우리는 이러한 대여 서비스를 사용하고자 하는 관리자(기업 & 단체 입장)와 대여자가 타깃이며, 물품 대여에 있어 실생활에 적용 가능한 웹페이지를 만들고자 한다.

▼ 구현 방법

2. 연구 내용

▼ 프론트엔드(front-end)

```
<Header></Header>
<div className="MainContent">
    <LoginBox>
    </LoginBox>
```

[그림] component 사용1

```
<SignUpBox
    signUp="관리자"
    closeModal={closeRegisterModal}
/>
```

[그림] component 사용2

Header, LoginBox, SignUpBox 등 위 그림처럼 원하는 부분에 필요한 만큼 component를 불러와서 사용함으로써 코드의 효율성을 높인다. 같은 코드가 불필요하게 반복되는 것을 막으며, 간결하고 reveiw하기 좋은 형태의 코드가 된다.

▼ 백엔드(back-end)

한계점 전 :

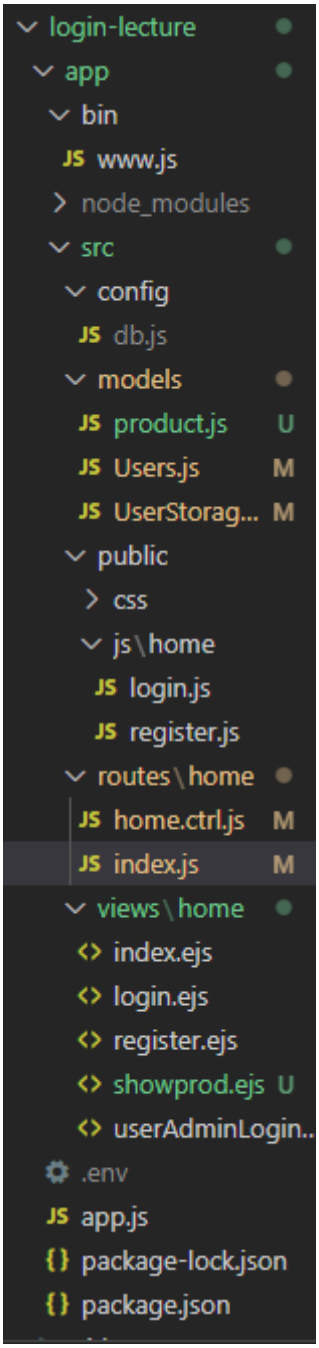
백엔드에서는 주제로 서버 클라이언트 통신 방식이해를 목적으로, 개념과 기술스택은 REST API nodemon ,node.js, javascript, Express.js, 디자인패턴 MVC 등을 적용했다.

```
router.get('/', ctrl.output.home);
router.get('/login', ctrl.output.login);
router.get('/register', ctrl.output.register);
// router.get('/showprod', ctrl.output.showprod);

router.post('/login', ctrl.process.login);
router.post('/register', ctrl.process.register);
// router.post('/product', ctrl.process.showprod);
```

REST API 에 대한 get과 post 구현

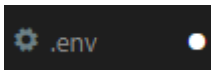
```
"use strict";

const id = document.querySelector("#id"),
psword = document.querySelector("#psword"),
loginBtn = document.querySelector("button");

loginBtn.addEventListener('click', login);

function login() {
    const req = {
        id: id.value,
        psword: psword.value,
    };
    fetch("/login", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
        },
      body : JSON.stringify(req),
    }).then((res) => res.json())
    .then((res) => {
        if (res.success) {
            if(res.checkadmin == 1){
                location.href ="/register";
            }
            else{
            location.href ="/";
            }
        } else {
            alert(res.msg);
         }
    })
    .catch((err) => {
        console.error(("로그인 중 에러 발생"));
    });
}
```

Node.js 문법을 사용한 login logic 구현



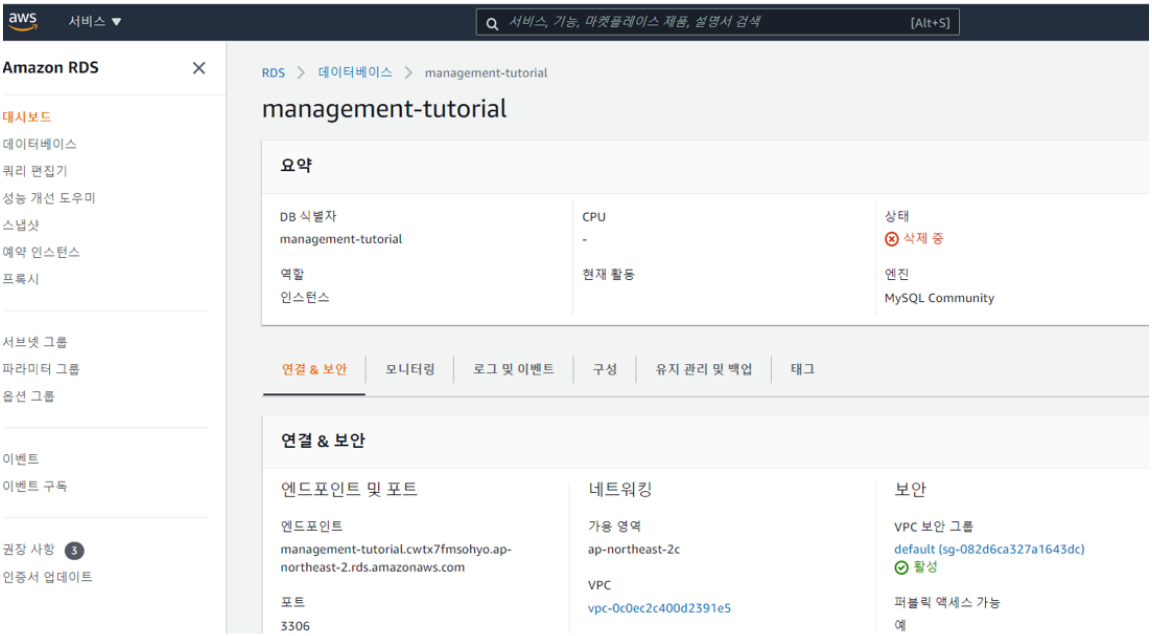
MVC 패턴을 적용한 파일 관리

AWS RDS 를 이용한 데이터베이스 클라우딩



```
PORT=3000

DB_HOST="management-tutorial.cwtx7fmsohyo.ap-northeast-2.rds.amazonaws.com"
DB_USER="user"
DB_PASSWORD=""
DB_DATABASE="login"
```

환경변수를 이용한 보안처리



특히 처음 백엔드를 개발하는 부분이어서, AWS RDS를 관리하는 부분에 있어서도 추가비용을 계산하지 못해 서버를 종료해야만 하는 시행착오를 겪기도 하였고, DB관리에 있어서도 실제 DB랑 AWS, 그리고 NODE.js 와 연동하는 부분에 있어서도 연동이 잘 안되거나, 오류가 생기는 등의시행착오를 겪기도 하였다.

개발부분에 있어서 목적은 학생회에서 물품을 대여해줄때 항상 물품대여기록을 기록해야하고, 누가 대여했고, 반납을 했는지도, 정확한 재산관리가 되지않아서, 프로그램을 만들기로 목적을 정했었다. 처음 서버 포트를 열고 전체적인 틀을 잡기위해서 모델-뷰-콘트롤러 MVC패턴을 적용해서 역할을 나누었다.

모델-뷰-컨트롤러는 소프트웨어 디자인 패턴인데, 뷰 부분을 프론트엔드, 모델부분을 DB관리자, 컨트롤러를 백엔드가 담당했다. 이것을 사용한 이유는 역할분담을 확실히해야하는 부분도 있지만, 개발중에 발생하는 문제들을 쉽게 정의하기위해 개발자들사이에서 규약을 정하는 이유이기도 하다.

사용자가 버튼을 클릭하는 등의 이벤트가 프론트엔드에서 발생하면 그것에 대한 반응을 서버를 통해서 보내줘야하는데, 위에서 소개했던 Express.js 를 이용해서 프론트엔드에서 그에 대한 올바른 response를 보내주기 위해 백엔드는 그에맞게 api를 정의해야한다. 통신 부분외의 논리적인 부분은 node.js 를 이용해서 클래스 개념을 적용해 메소드를 호출하고, 데이터베이스 Query문을 이용해서 데이터를 CRUD 하고, 그것에 대한 response를 Model이 받아와서 Controller 를 거쳐 View 를 통해 사용자에게 화면이 보여지게 된다.

한계점은 처음 개발스택을 공부하면서 개발을 진행하다보니 node.js 문법자체에 이해도가 떨어지기도 했고, 파일관리나 mvc 패턴을 잘 이해하지 못한상태로 접근하다보니, 프론트엔드랑 api와 파일연동 부분에있어서 문제점을 느꼈다. 결국 이전 코드를 응용해서 프론트엔드 파일에 직접 코드를 다시 정의하다보니, 작성도 편하고, 훨씬 간단해져서 연구의 목적을 잃지 않고, 과정을 잘 헤쳐나갔던 것 같다.

한계점 후 :

3. 연구 결론

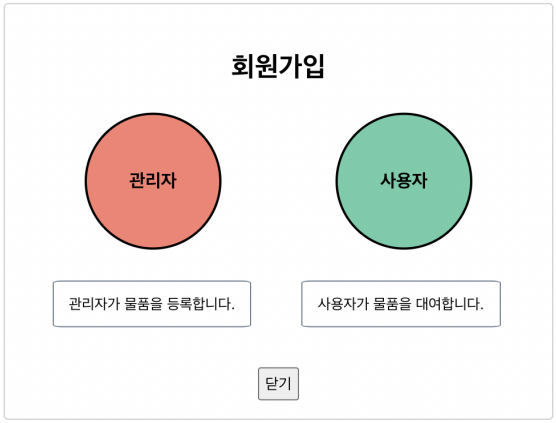
▼ 최종구현

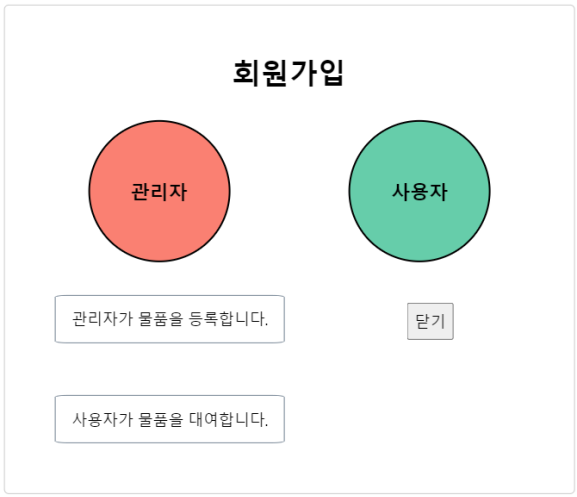
▼ 한계점

- css

모바일 환경이 시작되면서 장치도 다양해져, 해상도의 크기 및 비율이 제각각 다른 경우가 생겼다. 이전에는 모니터에 맞춰 1024x768, 1280x960 등의 동일한 비율에 대해서만 대응하면 됐지만, 지금은 비율도 다양해졌을 뿐 아니라 높은 해상도에 비해 모바일 디바이스의 픽셀 크기 자체가 비교도 안될 만큼 작은 경우도 있다. 즉, 페이지 설계 시 고려할 사항들이 많아진 것이다. 이번 프로젝트에서도 이런걸로 인한 문제가 발생했다.

노트북별로 다른 디스플레이 크기로 인해 같은 코드여도 보여지는 비율이 다르게 나타나 아이콘이나 버튼들이 우리의 본래 목적으로 움직이지 않는 문제가 생겼다. 해당 과정에서 이를 완벽하게 해결해 구현해내지 못한 한계점이 있다.






```
.joinText{
  float: left;
  font-size: small;
  padding: 2.5% 3%;
  margin: 5% 5%;
  border: slategrey solid 1px;
  border-radius: 5%;
}
```

- 프론트 엔드

1. javascript 문법 공부 부족함

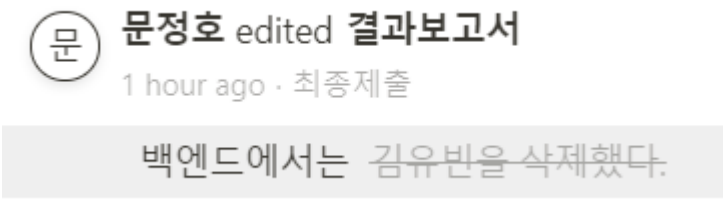
2. react를 많이 다뤄보지 않아 다양한 라이브러리를 알지 못함 - table 만들 때 아쉬웠음
3. component 사용이 미숙하여 자유롭게 coponent를 불러와 사용하지 못함(react의 장점인 component를 제대로 사용하지 못했다)
4. (공통) back-end와 연결하는 부분 미숙함 - 한 프로젝트 내에서 연결할 수밖에 없었음
5. (협업) front, back, design part를 나누면서 github 관리를 완벽하게 하지 못함

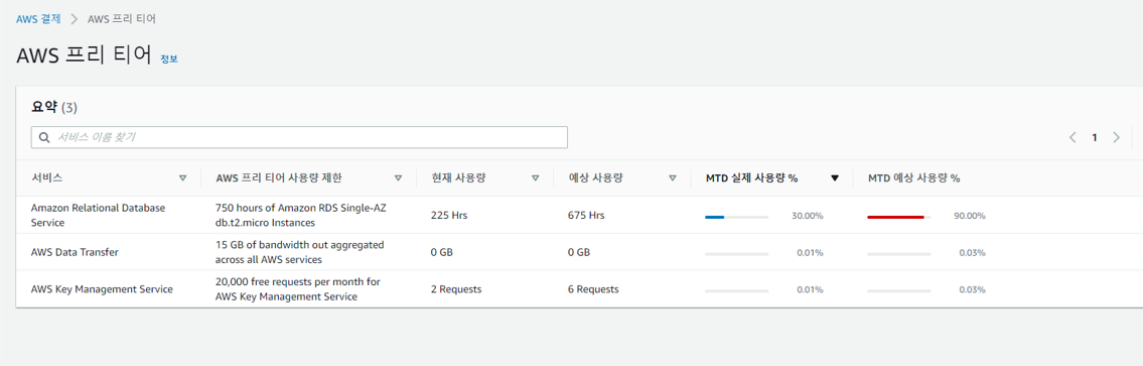
- 백엔드

1. 배포 실패(nodejs 공부 부족)
2. 클라우드 서비스 이용 불가
3. javascript (문법) 공부 부족함
4.

배포를 목적으로 DataBase를 클라우드 서비스에 생성하여 웹사이트를 배포하는데 있어 원활한 서비스를 제공하는 것을 목표로 하였으나, 개인 목적 상의 사용에 있어서 한계점이 생겨 개인 컴퓨터 서버에 DataBase를 생성하는 방향으로 진행하게 되었다.

개인 컴퓨터 DataBase 서버를 이용하게 될 경우, Data를 꺼내오기 위해서는 Data가 들어있는 DataBase 서버가 있는 개인 컴퓨터가 켜져 있어야 한다. 그렇기 때문에 배포를 하게 될 경우, 개인 컴퓨터가 항시 켜져있어야 하기에 서버를 운영하는데에 있어 현실적인 문제에 직면하게 된다. 그렇기 때문에 배포를 목적으로 하기 위해서는 클라우드 DataBase가 필수적인데,





▼ 발전 가능성

- css 부분에서 발생한 기기별 화면 크기 차이로 발생한 컴포넌트 위치 및 사이즈 차이 문제를 해결하고자 반응형 웹을 알아보았다. 반응형 웹이란 디바이스 종류에 따라 웹페이지의 크기가 자동적으로 재조정되는 것을 말한다. 이는 어떠한 환경에서도 그에 맞게 사이즈가 변화되어 사용자가 보기 편리하게 만드는 웹인 것이다. 오직 하나의 HTML 소스 만으로 특정 장치에 최적화된 환경에 사용자에게 제공할 수 있다. 이를 제대로 익혀 프로젝트에 적용시켜 본다면 누가 어떤 기기에서 우리의 웹페이지를 실행시켜 본다고 하더라도 차이 없는, 개발자가 의도했던 UI를 볼 수 있을 것이다.
- react 공부를 더 해서 더 효율적이고 멋진 코드를 짜자!
- github 관리의 중요성을 깨닫게 됐으니 앞으로 협업할 때 팀원들과 함께 코드 관리를 멋지게 하도록 해야겠다.

4. 첨부 자료

▼ 참고문헌 목록

▼ 연구 과정 증빙자료