LH공공임대주택 정보조회 반응형 웹앱

LH Public Housing Information Inquiry Responsive Webapp

H201926137 홍혜원

목차 a table of contents

목차 a table of contents

Part 1

작품 제작 동기

제작 배경

- 웹/모바일 환경에 구애받지 않고 공공임대주택의 단지 정보 및 공고 현황을 편리하게 확인하고 싶다.
- 기존에 존재하는 웹페이지의 경우, PC 환경에 맞춰 개발되어 있어 모바일 환경에선 사용이 불편한 상태이다.

필요성

- 공공임대주택은 공고 현황을 수시로 확인해 봐야 하기 때문에 언제 어디서든 접속할 수 있는 휴대용 기기에 최적화 된 서비스가 필요하다.
- 공공임대주택의 혜택이 필요한 사회 취약 계층의 경우 가구내 PC 또는 모바일 기기가 없을 가능성이 있으므로 가능한 다양한 환경에 대응할 수 있어야 한다.

추진 방향

- 웹/모바일 환경에 최적화 된 효율적이고 직관적인 UI
- ~~빠른 필터 & 검색 속도 (데이터 형태 이슈로 수정)~~

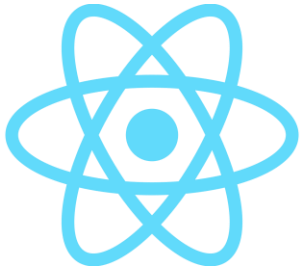
Part 2

주차별 계획표

구 분	LH공공임대주택 정보조회 반응형 웹앱														
	1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주	13주	14주	15주
계 획	개발 방법 검색 및 목표 수립			디자인 완료											
개 발					개발 착수				1차 완료						최종 완료
진 단										Test 진행			기능 개선		
평 가											실 사용 평가				

Part 3

제작에 필요한 도구



React Web App

주요 도구

- Javascript(es6): Visual Studio Code

기타 사용 소프트웨어

- Html, Css
- Node, React, Axios
- 포토샵

필요 장비

- 개발 및 테스트용 PC 1대 (window),
- 테스트용 모바일 기기 2대 (안드로이드, ios)

Part 4

구성 및 동작 원리

구성

- LH 공공임대주택 공고 리스트 조회
- 공고 리스트 검색
- LH 공공임대주택 공고문 상세 조회

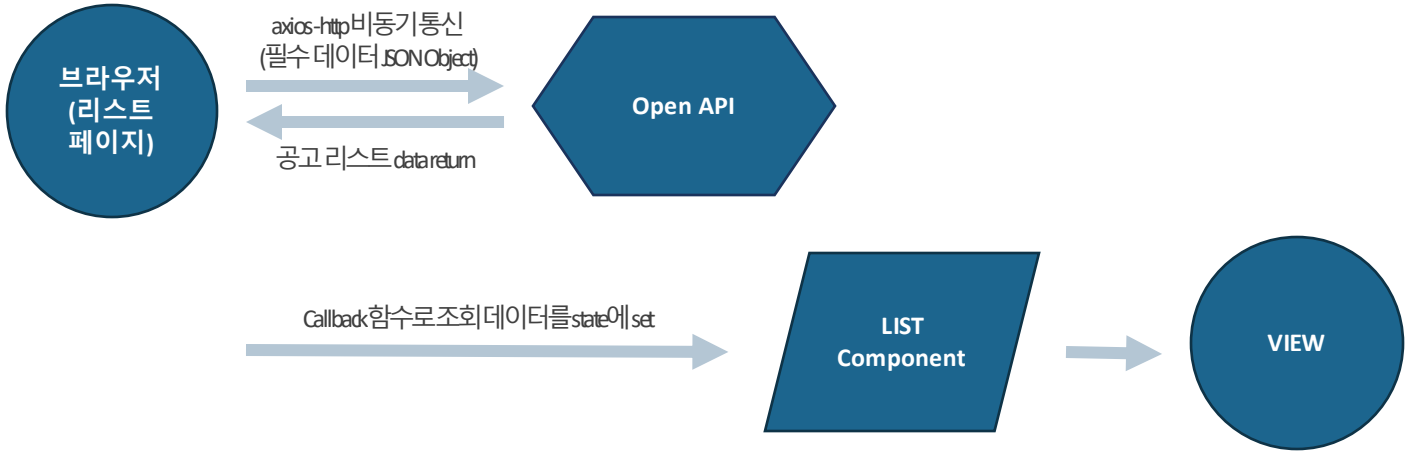
동작 원리

- 공고 리스트 조회 및 공고문 상세 조회 :
  JS AXIOS를 활용해 공공데이터포털의 오픈 api와 통신하여 데이터를 조회
- ~~공고 리스트 정렬 : 데이터 테이블 라이브러리 활용 (데이터 형태 이슈로 수정)~~

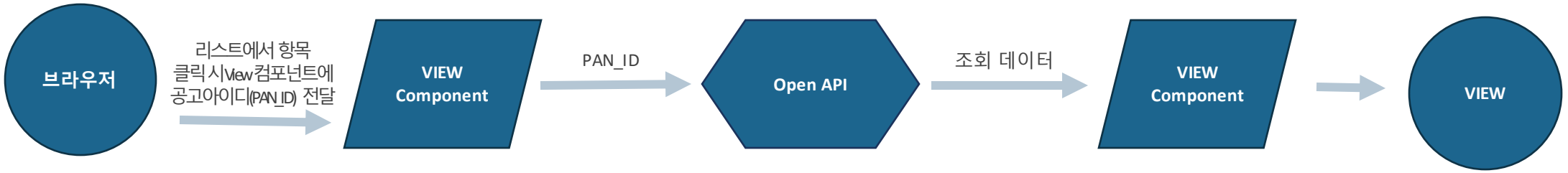
Part 5

프로그램 흐름도

리스트 조회 - 검색할 때마다 http 통신을 하지않아 검색속도가 빠르고 데이터 사용량이 적다(데이터 형태 이슈로 수정)



상세정보 조회



Part 6

구현된 소프트웨어 사진

1-1. 리스트 페이지 웹 (~1920px)

1-2. 리스트 페이지 모바일 (400px~)

Part 6

구현된 소프트웨어 사진

2-1. 상세 페이지 웹 (~1920px)

2-2. 상세 페이지 모바일 (400px~)

소스의 주요 부분과 동작 설명

1-1. 리스트 조회 기능

-React axios, Express 미들웨어를 활용해 외부 api와 비동기 통신)

Server.js

```
const express = require('express');
const cors = require('cors');
const axios = require('axios');
const app = express();

// CORS 설정해주기
app.use(cors({ credentials: true, origin: 'http://localhost:3000' }));

// 요청 본문에 있는 데이터를 객체로 변환해주기
app.use(express.urlencoded({ extended: true }));

// JSON 파싱해주기
app.use(express.json());

app.get('/B552555/lhLeaseNoticeInfo1/lhLeaseNoticeInfo1', async (req, res) => {
  try {

    res.header('Content-Type', 'application/json; charset=utf-8');

    // API에 GET 요청 보내기
    const apiResponse = await axios.get('http://apis.data.go.kr/B552555/lhLeaseNoticeInfo1/lhLeaseNoticeInfo1', {
      params: req.query,
    });

    // API 응답에서 필요한 데이터 추출 또는 가공
    const responseData = {
      message: 'GET request successfully!',
      data: apiResponse.data,
    };

    // 클라이언트에게 JSON 형식으로 응답
    res.json(responseData);
  } catch (error) {
    console.error('Error during relay:', error);
    res.json({ error: error.message });
  }
});

app.get('/B552555/lhLeaseNoticeDtlInfo1/getLeaseNoticeDtlInfo1', async (req, res) => {···
});

// 서버 시작
const port = 9999;
app.listen(port, () => {
  console.log(`Proxy server is running on http://localhost:${port}`);
});
```

리액트앱 구동 서버(http://localhost:9999)
CORS(Cross-origin resource sharing) 허용

브라우저의 cors 정책 이슈를 피하기 위해 Node js로 구동한 서버단에서 외부api와 통신

Node js로 미들웨어 서버(http://localhost:9999)가동

이 때, reqUrl은 미들웨어 서버인 http://localhost:9999로 작성
reqUrl: 'http://localhost:9999/B552555/lhLeaseNoticeInfo1/lhLeaseNoticeInfo1'

ListContainer.js

```
// 조회 함수
const requestApi = async (requrl, reqData) => {
    let loading = document.querySelector(".blackBg");
    try {

        if(loading) loading.style.display = "block";
        const response = await axios.get( requrl, {
                params: reqData,
                withCredentials: true
            });

        console.log(response.data);
        setDataList(response.data.data);
        if(loading) loading.style.display = "none";

    } catch (error) {
        console.error('API 요청 에러:', error);
        if(loading) loading.style.display = "none";
    }
};
```

※ 브라우저에서 직접 외부 API와 통신을 시도하면 브라우저 보안 정책 상 **CORS(Cross-origin resource sharing)에러***가 발생하기 때문에 브라우저 내에서 구동되는 리액트 앱과 외부 API의 통신을 중계해줄 미들웨어를 구현해야 했다.

* **CORS(Cross-origin resource sharing)** - 보안상의 이유로 브라우저는 스크립트 단에서 실행 된 HTTP **교차 출처 요청****을 거부한다.
(단, 서버에서 교차 출처 요청을 허용했고, 브라우저에서 올바른 cors 헤더가 포함된 경우 허용)

** **교차 출처 요청** - 요청 브라우저와 서버간의 출처가 다른 경우

소스의 주요 부분과 동작 설명

1-2. 리스트 조회 기능

- state 관리를 통해 비동기 통신 후 Html DOM 내에서 필요한 부분만 다시 렌더링 시키기

ListContainer.js

```
// state set
const [initFlg, setInitFlg] = useState(true);
const [dataList, setDataList] = useState([]);
```

→ 리스트 state 초기값 및 setter 선언

...

ListContainer.js

```
// 조회 함수
const requestApi = async (reqUrl, reqData) => {
    let loading = document.querySelector(".blackBg");
    try {

        if(loading) loading.style.display = "block";
        const response = await axios.get( reqUrl, {
                params: reqData,
                withCredentials: true
            });

        setDataList(response.data.data);
        if(loading) loading.style.display = "none";

    } catch (error) {
        console.error('API 요청 에러:', error);
        if(loading) loading.style.display = "none";
    }
};
```

→ 통신 성공 시 리스트 state 값 갱신

...

ListContainer.js

```
    return (
        <div id="wrap">
            <Header></Header>
            <div className='container'>
                <Filter
                    onSearch={onSearch}
                    onChangePostDate={onChangePostDate}
                    onChangeCloseDate={onChangeCloseDate}
                    startPostDate={startPostDate}
                    endPostDate={endPostDate}
                    startCloseDate={startCloseDate}
                    endCloseDate={endCloseDate}
                />
                <List
                    dataList={dataList}
                    onSearch={onSearch}
                />
            </div>
            <div className='blackBg'>
                <div className='loading'></div>
            </div>
        </div>
    );
```

→ List 컴포넌트에 props로 전달
state 값 갱신 시 List 컴포넌트만 다시 렌더링

Props로 받아온 리스트 데이터 state

List.js

```
function List {dataList, onSearch}) {
```

...

List.js

```
    {
        dsList && dsList.length > 0 ?
        dsList.map((item) => {
            return (
                <li key={item.PAN_ID}>
                    <Link
                        to={"/view"}
                        state={{
                            SPL_INF_TP_CD: item.SPL_INF_TP_CD,
                            CCR_CNNT_SYS_DS_CD: item.CCR_CNNT_SYS_DS_CD,
                            PAN_ID: item.PAN_ID,
                            UPP_AIS_TP_CD: item.UPP_AIS_TP_CD,
                            AIS_TP_CD: item.AIS_TP_CD
                        }}>
                        <div key={item.PAN_ID+"_RNUM"}>{item.RNUM}<input type='hidden' value={item.SPL_INF_TP_CD}/></div>
                        <div key={item.PAN_ID+"_PAN_NM"}>{item.PAN_NM}</div>
                        <div key={item.PAN_ID+"_UPP_AIS_TP_NM"}>{item.UPP_AIS_TP_NM}</div>
                        <div key={item.PAN_ID+"_CNP_CD_NM"}>{item.CNP_CD_NM}</div>
                        <div key={item.PAN_ID+"_PAN_SS"}>{item.PAN_SS}</div>
                        <div key={item.PAN_ID+"_PAN_NT_ST_DT"}>{item.PAN_NT_ST_DT}</div>
                        <div key={item.PAN_ID+"_CLSG_DT"}>{item.CLSG_DT}</div>
                    </Link>
                </li>
            );
        }) :
        (<li key={"empty"} className='emptyList'>검색 결과가 없습니다.</li>)
    }
```

리스트 배열 크기만큼 반복문을
돌면서 DOM에 리스트를 출력한다

※ **기존 웹 프로젝트에선** 서버와 통신 후, 갱신된 데이터를 보여주기 위해선 **화면 전체를 다시 로드**해줘야 했기 때문에 속도도 느리고 스크롤이나 검색조건 등 화면의 모든 요소가 초기화 되었기 때문에 사용자에게 불편함이 있었다.

해당 앱은 리액트로 작성되었기 때문에 서버와 비동기 통신 후 **state를 갱신하면 리스트 부분만 다시 렌더링** 되기 때문에 이와 같은 불편함이 제거 되었다.

소스의 주요 부분과 동작 설명

1-3. 리스트 검색 기능

- 부모 컴포넌트와 자식 컴포넌트간 함수 및 props 주고받기

ListContainer.js

```
// 검색
const onSearch = (pgSzParam, pageParam) => {
    if(!initFlg){

        let panNm = document.querySelector("input[name='PAN_NM']").value; //공고명
        let uppAisTp = document.querySelector("select[name='UPP_AIS_TP_CD']").value; //공고유형
        let cnp = document.querySelector("select[name='CNP_CD']").value; //지역
        let panSs = document.querySelector("select[name='PAN_SS']").value; //공고 상태
        let panStDt = moment(startPostDate).format("YYYYMMDD"); //게시시작기간
        let panEdDt = moment(endPostDate).format("YYYYMMDD"); //게시끝기간
        let clsgStDt = moment(startCloseDate).format("YYYYMMDD"); //마감시작기간
        let clsgEdDt = moment(endCloseDate).format("YYYYMMDD"); //마감끝기간
        let pgSz = pgSzParam ? pgSzParam : dataList[0].dsSch[0].PG_SZ; //한페이지결과수
        let page = pageParam ? pageParam : dataList[0].dsSch[0].PAGE; //페이지번호

        let searchData = {
            serviceKey: serviceKey,
            PAN_NM: panNm,
            UPP_AIS_TP_CD: uppAisTp,
            CNP_CD: cnp,
            PAN_SS: panSs,
            PAN_ST_DT: panStDt,
            PAN_ED_DT: panEdDt,
            CLSG_ST_DT: clsgStDt,
            CLSG_ED_DT: clsgEdDt,
            PG_SZ: pgSz,
            PAGE: page
        }

        requestApi(reqUrl, searchData);
    }
```

부모 컴포넌트 (리스트 페이지)에서
사용자가 작성한 검색 조건을
요청 데이터에 설정한 뒤,
데이터 조회 요청을 하는 함수 작성

...

ListContainer.js

```
    return (
        <div id="wrap">
            <Header></Header>
            <div className='container'>
                <Filter
                    onSearch={onSearch}
                    onChangePostDate={onChangePostDate}
                    onChangeCloseDate={onChangeCloseDate}
                    startPostDate={startPostDate}
                    endPostDate={endPostDate}
                    startCloseDate={startCloseDate}
                    endCloseDate={endCloseDate}
                />
                <List
                    dataList={dataList}
                    onSearch={onSearch}
                />
            </div>
            <div className='blackBg'>
                <div className='loading'></div>
            </div>
        </div>
    );
```

setter를 포함한 함수도
state와 마찬가지로 자식
컴포넌트에게 props로 전달

부모 컴포넌트로부터 전달받은 함수, state, setter

Filter.js

```
function Filter({onSearch, onChangePostDate, onChangeCloseDate, startPostDate, endPostDate, startCloseDate, endCloseDate}) {

    return (
```

...

Filter.js

```
    </ul>
    <div className='filterFoot'>
        <button onClick={()=> onSearch()}>
            조회하기
            <FontAwesomeIcon icon={faSearch}/>
        </button>
    </div>
```

버튼에 부모 컴포넌트에서
받아온 함수를 클릭 이벤트로
적용

...

Filter.js

```
<li key={"PAN_DT"}>
    <label>게시 기간</label>
    <DatePicker
        locale={ko}
        selected={startPostDate}
        onChange={(update) => {
            onChangePostDate(update);
        }}
        startDate={startPostDate}
        endDate={endPostDate}
        dateFormat="yyyy/MM/dd"
        selectsRange
    />
</li>
<li key={"CLSG_DT"}>
    <label>마감 기간</label>
    <DatePicker
        locale={ko}
        selected={startCloseDate}
        onChange={(update) => {
            onChangeCloseDate(update);
        }}
        startDate={startCloseDate}
        endDate={endCloseDate}
        dateFormat="yyyy/MM/dd"
        selectsRange
    />
</li>
```

DatePicker에 부모 컴포넌트의 state setter를
change 이벤트로 적용하여 부모 컴포넌트의
state 값을 제어

소스의 주요 부분과 동작 설명

2-1. 상세 데이터 조회

- React 라우터로 컴포넌트간 state 주고받기

List.js

```
<ul className='listBody'>
    {
        dsList && dsList.length > 0 ?
        dsList.map((item) => {
            return (
                <li key={item.PAN_ID}>
                    <Link
                        to={"/view"}
                        state={{
                            SPL_INF_TP_CD: item.SPL_INF_TP_CD,
                            CCR_CNNT_SYS_DS_CD: item.CCR_CNNT_SYS_DS_CD,
                            PAN_ID: item.PAN_ID,
                            UPP_AIS_TP_CD: item.UPP_AIS_TP_CD,
                            AIS_TP_CD: item.AIS_TP_CD
                        }}>
                        <input key={item.PAN_ID+"_PAN"} type='hidden' value={item.SPL_INF_TP_CD}/>
                        <div key={item.PAN_ID+"_PAN_NM"}>{item.PAN_NM}</div>
                        <div key={item.PAN_ID+"_UPP_AIS_TP_NM"}>{item.UPP_AIS_TP_NM}</div>
                        <div key={item.PAN_ID+"_CNP_CD_NM"}>{item.CNP_CD_NM}</div>
                        <div key={item.PAN_ID+"_PAN_SS"}>{item.PAN_SS}</div>
                        <div key={item.PAN_ID+"_PAN_NT_ST_DT"}>{item.PAN_NT_ST_DT}</div>
                        <div key={item.PAN_ID+"_CLSG_DT"}>{item.CLSG_DT}</div>
                    </Link>
                </li>
            );
        }) :
        (<li key={"empty"} className='emptyList'>검색 결과가 없습니다.</li>)
    }
</ul>
```

리액트 라우터에서 제공하는 Link 컴포넌트의 기능으로 다음 페이지에 상세 정보 조회에 필요한 데이터를 전달

Link의 props를 이러한 형태로 설정하면 다음 페이지에서 to의 value는 location.pathname에, state의 value는 location.state에 들어있다

※ 리액트 라우터에는 a태그의 기능을 대신하는 Link 컴포넌트를 제공해주고 있다 (build후 배포 시 Link 컴포넌트는 a태그로 변환된다)

리액트는 싱글 페이지 앱에 특화된 라이브러리로, 페이지가 이동하는 것처럼 보이지만 사실 패스를 인식하여 화면 요소만 교체하는 것이기 때문에 location, histot 등의 객체를 기존의 웹 프로젝트 환경처럼 사용할 수 없다.

따라서 이를 보완하기 위해 리액트 라우터에서 기존의 location 객체와 유사한 기능을 수행하는 useLocation()를 제공하고 있다.

location을 사용하기 위해 리액트 라우터에서 제공하는 useLocation을 import

ViewContainer.js

```
import View from '../component/View';
import { useLocation } from 'react-router-dom';
import axios from 'axios';
```

...

ViewContainer.js

```
const location = useLocation();

// 조회 함수
const requestApi = async () => {
  let loading = document.querySelector(".blackBg");
  try {
      document.querySelector(".blackBg").style.display = "block";
      let reqData = {
        serviceKey: serviceKey,
        SPL_INF_TP_CD: location.state.SPL_INF_TP_CD,
        CCR_CNNT_SYS_DS_CD: location.state.CCR_CNNT_SYS_DS_CD,
        PAN_ID: location.state.PAN_ID,
        UPP_AIS_TP_CD: location.state.UPP_AIS_TP_CD,
        AIS_TP_CD: location.state.AIS_TP_CD
      }
      const response = await axios.get( reqUrl, {
        params: reqData,
        withCredentials: true
```

useLocation() 선언

이전 페이지에서 보낸 data들을 location으로부터 꺼내서 요청 데이터에 set

※ 상세 데이터 조회를 위한 통신 기능은 1-1과 동일하므로 설명 생략

소스의 주요 부분과 동작 설명

2-2. 상세 데이터 노출

- 컴포넌트를 활용한 모듈화

ViewContainer.js

```
return (
  <div id="wrap">
    <Header page={"view"}></Header>
    <div className='container'>
      {
        dsCtrtPlc ?
        <View viewId={"dsCtrtPlc"}  viewTit={"접수처정보"} viewDataNm={dsCtrtPlcNm} viewData={dsCtrtPlc}></View> :
        ""
      }
      {
        dsSbdNm ?
        <View viewId={"dsSbd"}  viewTit={"단지정보"} viewDataNm={dsSbdNm} viewData={dsSbd}></View> :
        ""
      }
      {
        dsAhflInfo ?
        <View viewId={"dsAhflInfo"} viewTit={"첨부파일정보"} viewDataNm={dsAhflInfoNm} viewData={dsAhflInfo}></View> :
        ""
      }
      {
        dsSbdAhfl ?
        <View viewId={"dsSbdAhfl"}  viewTit={"단지별첨부파일정보"} viewDataNm={dsSbdAhflNm} viewData={dsSbdAhfl}></View> :
        ""
      }
      {
        dsSplScdl ?
        <View viewId={"dsSplScdl"}  viewTit={"공급일정"} viewDataNm={dsSplScdlNm} viewData={dsSplScdl}></View> :
        ""
      }
      {
        dsEtcInfo ?
        <View viewId={"dsEtcInfo"}  viewTit={"기타정보"} viewDataNm={dsEtcInfoNm} viewData={dsEtcInfo}></View> :
        ""
      }
    </div>
    <div className='blackBg'>
        <div className='loading'></div>
    </div>
  </div>
);
```

유사한 형태의 요소가 반복되므로 View 컴포넌트로 모듈화하여 코드 재사용

JSX에서 if문이 switch문을 사용할 수 없으므로 삼항연산자로 데이터가 있는 경우에만 화면에 출력

View.js

```
function View({viewId, viewTit, viewDataNm, viewData}) {

    let dataHtml = "";

    if(viewDataNm && viewDataNm.length > 0){
        if(viewId == "dsAhflInfo"){…
        }else if(viewId == "dsSbdAhfl"){…
        }else if(viewId == "dsSbd"){…
        }else if(viewId == "dsSplScdl"){…
        }else{
            let viewDataNmObj = viewDataNm[0];
            Object.keys(viewDataNmObj).map((nmItem)=>{
                dataHtml += "<h4>"+viewDataNmObj[nmItem]+"</h4>";
                viewData.map((dataItem)=>{
                    if(dataItem[nmItem]){
                        dataHtml += "<p>"+dataItem[nmItem]+"</p>";
                    }else{
                        dataHtml += "<p>-</p>";
                    }
                });
            });
        }
    }

    return (
        <div className="viewContainer component">
            <h3 className='viewTit'>
                <FontAwesomeIcon icon={faCaretRight}/>
                <span>{viewTit}</span>
            </h3>
            <div id={viewId+'ViewTalbe'} dangerouslySetInnerHTML={{__html: dataHtml}}></div>
        </div>
    );
}

export default View;
```

조회해온 데이터를 가공하여 화면에 노출

(리액트 프로젝트는 사이트간 스크립팅 공격에 노출되기 쉽기 때문에 innerText 대신 dangerouslysetinnerhtml를 사용)

Part 8

결론

리액트를 통해 LH공공임대주택 공고 조회 서비스를 만들게 된 계기는
LH공공임대주택 정책을 통해 좋은 매물을 얻기 위해선 수시로 공고를 확인해야만 하는데, 기존의 서비스들은은 모바일에 친화적인 UI로 만들어져 있지 않아서 휴대용 디바이스로는 접근성이 떨어진다고 생각했기 때문이었습니다.

실제로 리액트를 활용해 어플리케이션을 제작해보니, 기존에 주로 사용했던 언어인 java를 활용한 웹 프로젝트와 비교하여 훨씬 가볍고 속도면에서도 뛰어난데다, 비교적 유연한 언어인 javascript를 알아가는 과정도 흥미로웠습니다.
어려웠던 부분은 리액트 앱이 브라우저 내에서 구동되는 프론트엔드 프로젝트이다보니 브라우저의 정책에서 자유롭지 못했던 부분들이었는데, 외부 api와 통신할 때 겪었던 CORS 에러가 그 중 하나였습니다.

또한 데이터의 구성을 파악하고 적절한 가공 방법을 설계하는데 난항을 겪어 도중에 설계를 완전히 바꾸게 된다던가,
화면에 출력해야 할 데이터의 양은 방대한데, 화면이 큰 웹에서 뿐만 아니라 화면이 작은 모바일에서도 데이터가 명확하게 보여야 했기에 데이터를 어떤 형태로 노출시킬 것인지, 만약 생략해야 된다면 어떤 데이터가 더 중요한지 등
데이터의 처리 방법을 고민하는데 가장 많은 시간이 소요되었습니다.

웹 개발자로 취업에 성공하였으나 실무 경험이 짧았던 제게 이번 과제는 업무의 범위를 보는 시야를 넓혀준 좋은 경험이었습니다.
서버 구축 없이 조회만 실행하는 작은 규모의 서비스였지만 만족할만한 완성도로 프로젝트를 완수하는 것은 쉽지 않은 일이었습니다.
하나의 좋은 프로그램을 만들기 위해선 단순히 코딩만 잘해서 될 것이 아니라 데이터를 정확하게 파악하고 효율적으로 관리하는 능력과,
데이터, 개발 언어, 개발 환경, 개발 목표 등 모든 요소에 대한 이해를 바탕으로 만들어진 완벽한 기획, 그리고 화면을 깔끔하고 가독성 좋게 구성하는 디자인 능력 등 제가 눈치채지 못했던 다양한 사람들의 다양한 고민과 노력들이 필요했음을 깨닫게 되었습니다.

이번 기회를 통해 깨달은 것을 회사에서 실무 활동을 할 때, 함께 일하는 팀원들과 더 원활한 커뮤니케이션을 하는데에 활용하도록 하겠습니다. 해당 프로그램은 앞으로도 계속 수정, 보완해가며 실제로 출시가 가능할 정도의 완성도까지 발전 시켜보고 싶습니다.

참고문헌

https://velog.io/@leewooseong/react-router-dom-v6%EC%82%AC%EB%9D%BC%EC%A7%84-match-location-history-props
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
https://velog.io/@leobt/%EB%AF%B8%EB%93%A4%EC%9B%A8%EC%96%B4%EC%99%80-Express
https://velog.io/@wjddnjswjd12/nodejs-express%EB%AF%B8%EB%93%A4%EC%9B%A8%EC%96%B4%EB%9E%80
https://velog.io/@kho5420/Web-API%EA%B7%B8%EB%A6%AC%EA%B3%A0-EndPoint
https://ts2ree.tistory.com/207
https://www.microsoft.com/ko-kr/security/business/security-101/what-is-an-endpoint
https://jinyisland.kr/post/middleware/
https://velog.io/@nemo/string-to-jsx
https://velog.io/@kaya53/React-React%EC%97%90%EC%84%9C-innerHTML-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0
https://okayoon.tistory.com/entry/React-DOM%EC%97%98%EB%A6%AC%EB%A8%BC%ED%8A%B8%EC%97%90-%ED%85%8D%EC%8A%A4%ED%8A%B8%EC%82%BD%EC%9E%85%ED%95%98%EA%B8%B0innerHTML%EB%A7%90%EA%B3%A0dangerouslySetInnerHTML%EB%A5%BC-%EC%82%AC%EC%9A%A9%ED%95%98%EC%9E%90
https://twojobui.tistory.com/10
https://mjn5027.tistory.com/80#google_vignette
https://hianna.tistory.com/402
https://m.blog.naver.com/PostView.naver?blogId=on21life&logNo=222354181592&categoryNo=0&proxyReferer=
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map
https://velog.io/@yonghk423/%EA%B0%9D%EC%B2%B4Object%EC%86%8D%EC%84%B1property-%EA%B0%9C%EC%88%98-%EA%B5%AC%ED%95%98%EA%B8%B0
https://hianna.tistory.com/452
https://www.data.go.kr/
https://velog.io/@nemo/useLocation
https://velog.io/@pearpearb/TIL-28-Link-to%EB%A1%9C-props%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%A0%84%EB%8B%AC%ED%95%98%EA%B8%B0
https://velog.io/@sham/Router-Props-link%EB%A1%9C-%EC%A0%84%EB%8B%AC%ED%95%98%EB%8A%94-props
https://mchch.tistory.com/140
https://devgabriel.tistory.com/31#google_vignette&gsc.tab=0
https://velog.io/@dlrdmswjd322/Express-response-Invalid-status-code%EC%98%A4%EB%A5%98
https://hianna.tistory.com/469
https://velog.io/@kms0206/Map-ForEach%EC%97%90-%EB%8C%80%ED%95%98%EC%97%AC
https://zoetechlog.tistory.com/134
https://joonpyohong.tistory.com/entry/JS-%EB%8F%99%EC%A0%81%EC%9C%BC%EB%A1%9C-%EC%83%9D%EC%84%B1%EB%90%9C-%ED%83%9C%EA%B7%B8%EC%97%90-%EC%9D%B4%EB%B2%A4%ED%8A%B8-%EA%B1%B8%EA%B8%B0
https://wonyoung2257.tistory.com/4
https://seokdl.tistory.com/8
https://velog.io/@dlruddms5619/React-useEffect%EC%B2%AB-%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%8B%9C-%ED%95%A8%EC%88%98-%EC%8B%A4%ED%96%89-%EB%A7%89%EA%B8%B0
https://velog.io/@gyutato/%ED%8A%B8%EB%9F%AC%EB%B8%94%EC%8A%88%ED%8C%85-React-useEffect%EC%9D%98%EC%A1%B4%EC%84%B1-%EB%B0%B0%EC%97%B4%EC%97%90-props%EB%84%A3%EA%B8%B0
https://velog.io/@hjthgus777/React%EB%8B%A4%EC%8B%9C-%ED%95%9C%EB%B2%88-useCallback%EC%9D%84-%ED%8C%8C%ED%97%A4%EC%B3%90%EB%B3%B4%EC%9E%90
https://mchch.tistory.com/140
https://pottat0.tistory.com/entry/momentjs-react%EB%A1%9C-%EB%82%A0%EC%A7%9C%EB%A5%BC-%EA%B4%80%EB%A6%AC%ED%95%98%EB%8A%94-%EC%89%BD%EA%B3%A0-%EB%B9%A0%EB%A5%B8-%EB%B0%A9%EB%B2%95
https://www.altcademy.com/blog/how-to-format-datetime-in-reactjs/
https://www.npmjs.com/package/react-moment#formatting
https://steemit.com/hive-101145/@realmankwon/react
https://sudo-minz.tistory.com/30