리액트 프로그래밍

COVID19

이승아

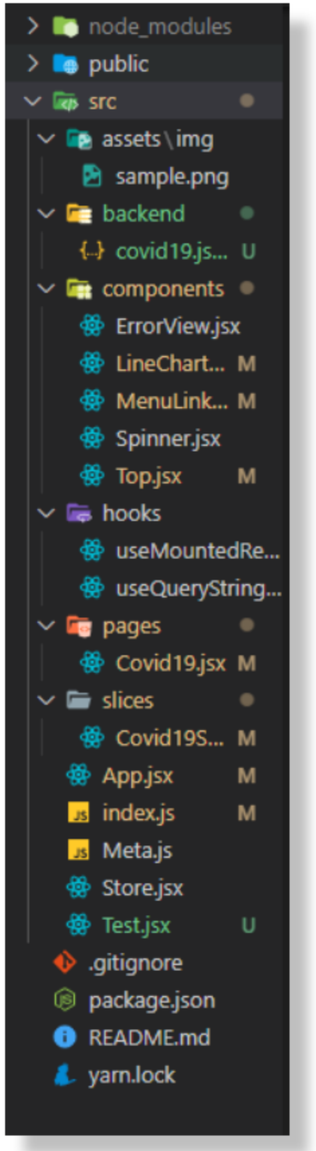
INDEX

01

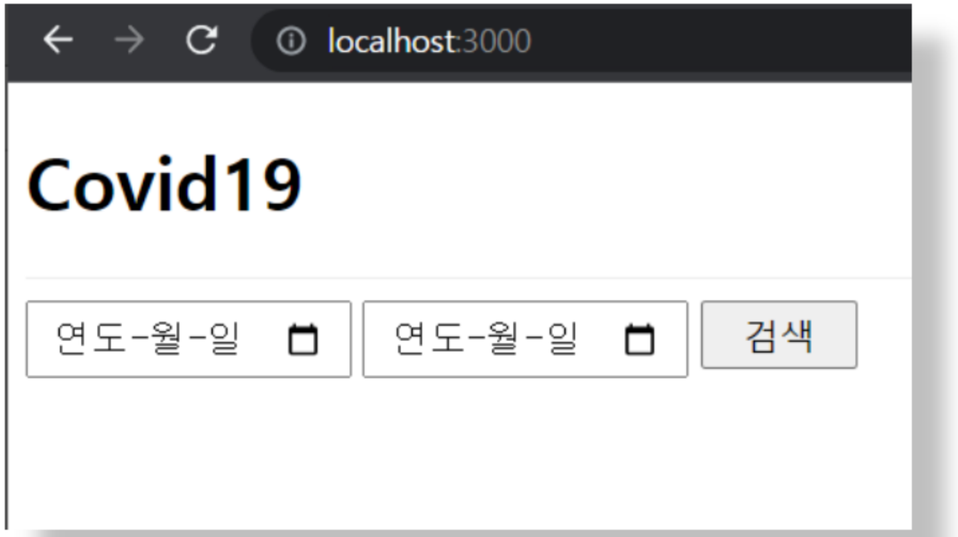
구현 결과

01 구현결과

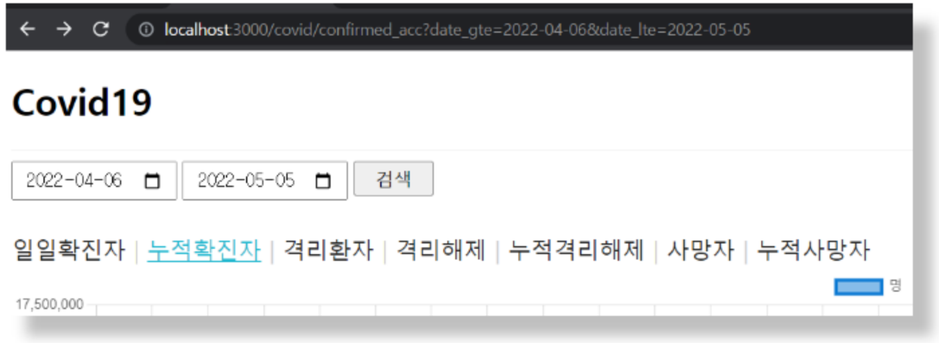
파일 폴더



첫 화면

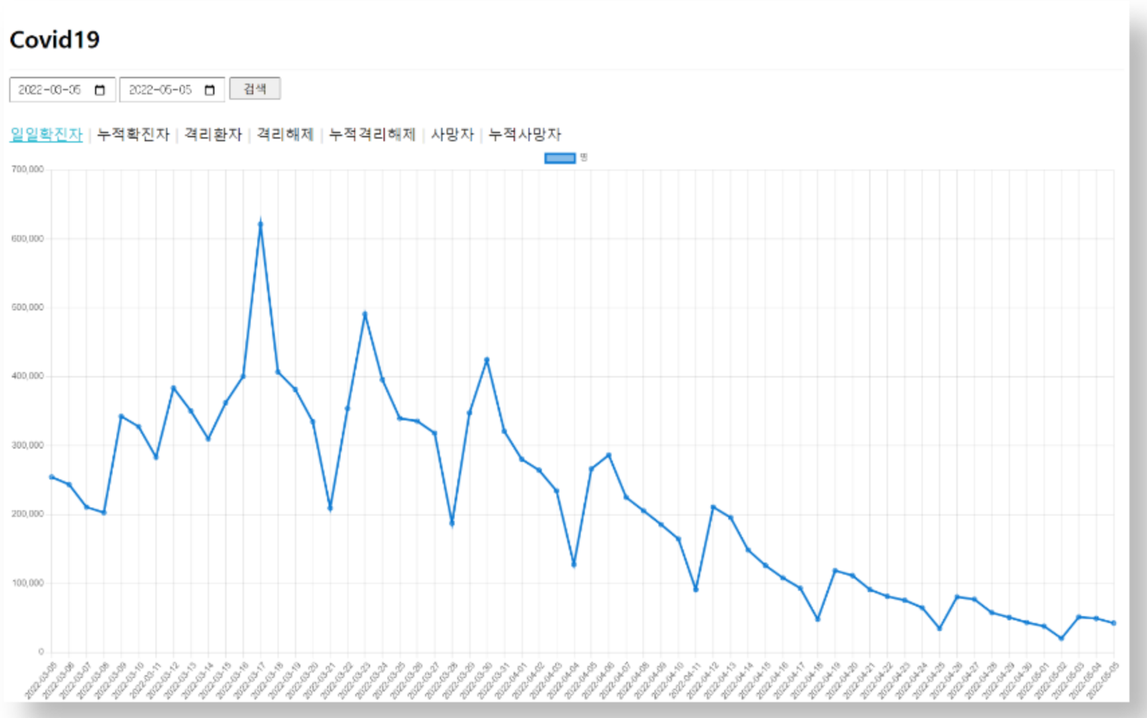


날짜 및 메뉴 선택시 링크 변화

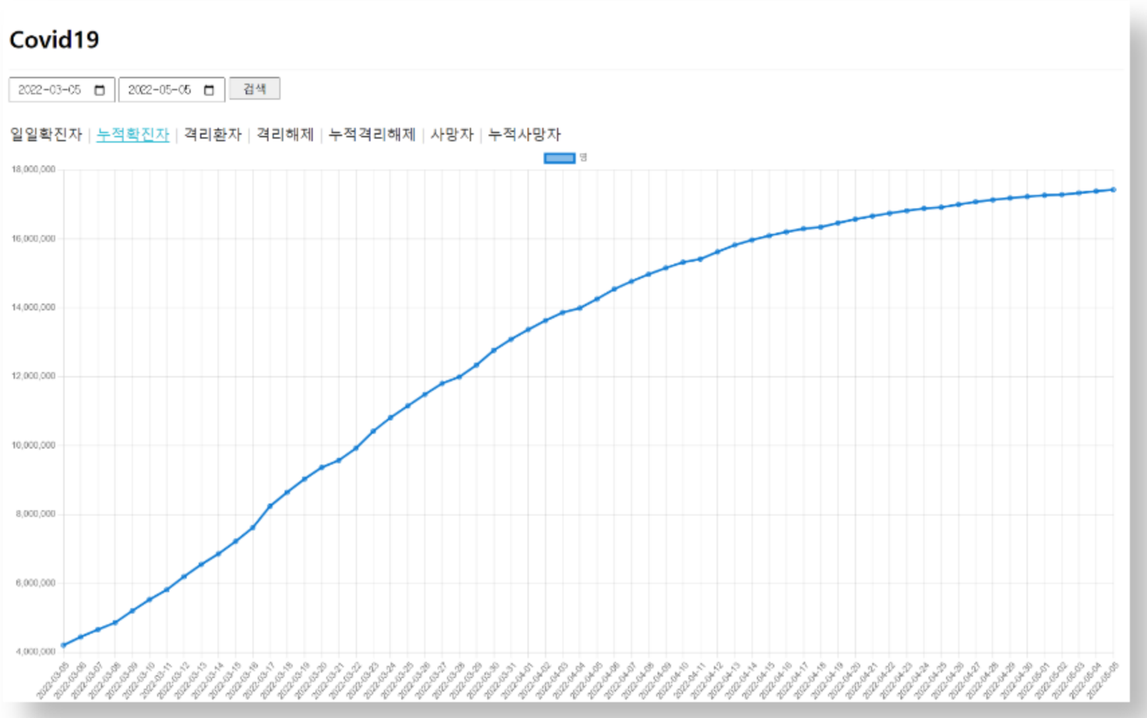


01 구현결과

일일확진자

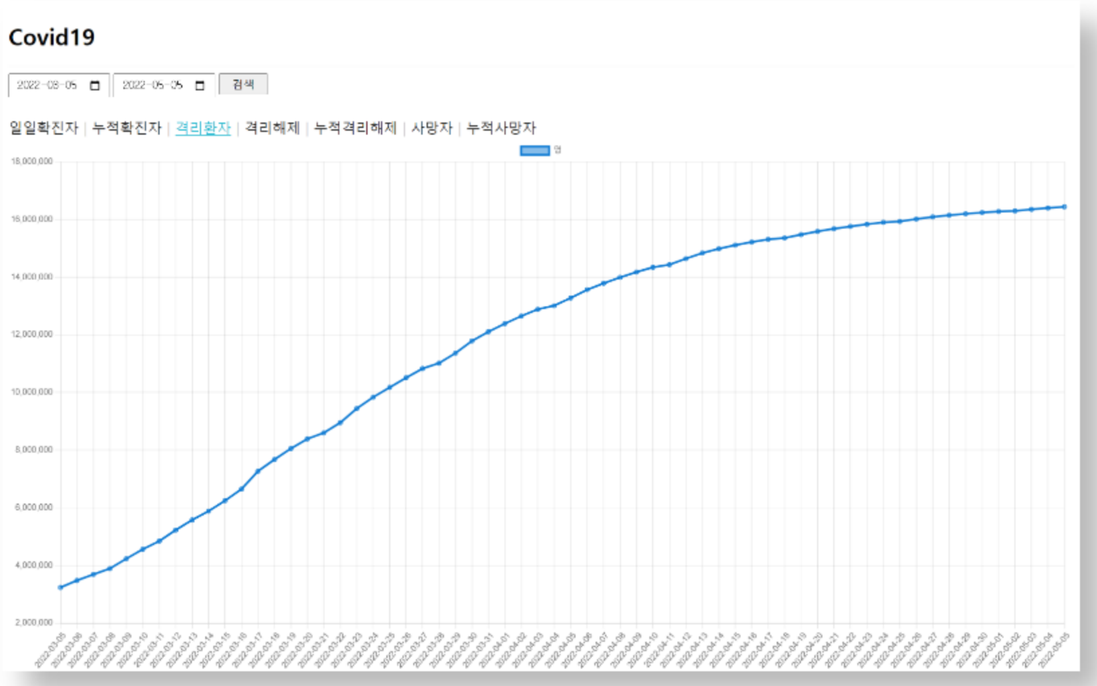


누적확진자



01 구현결과

격리환자

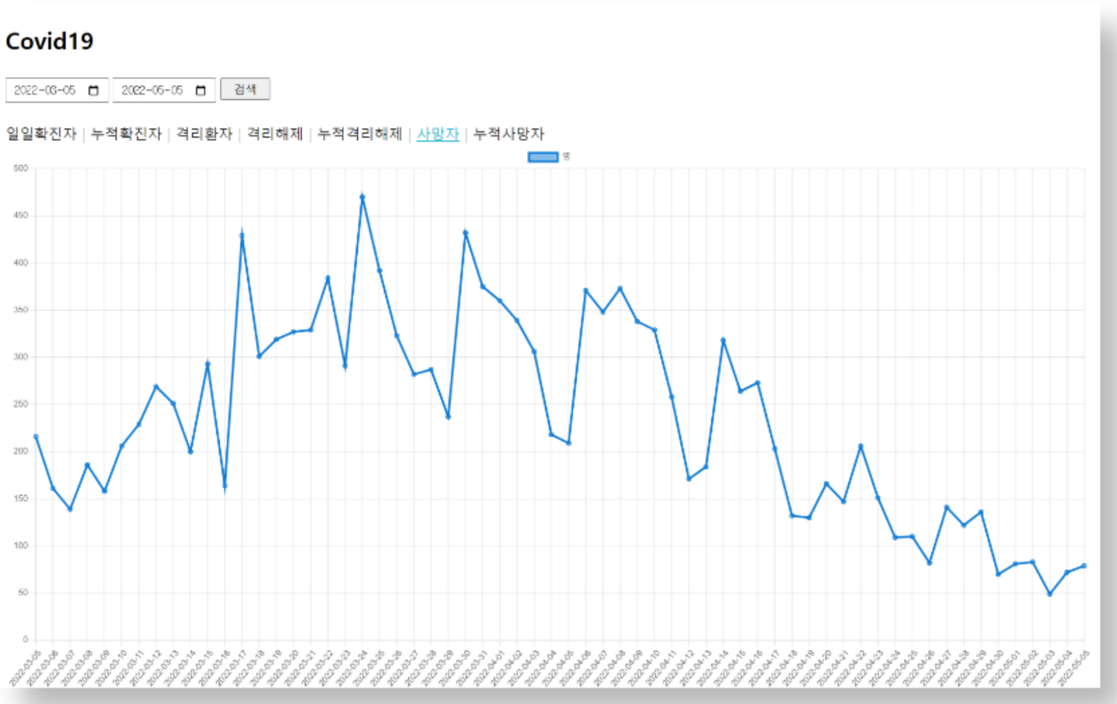


격리해제

01 구현결과

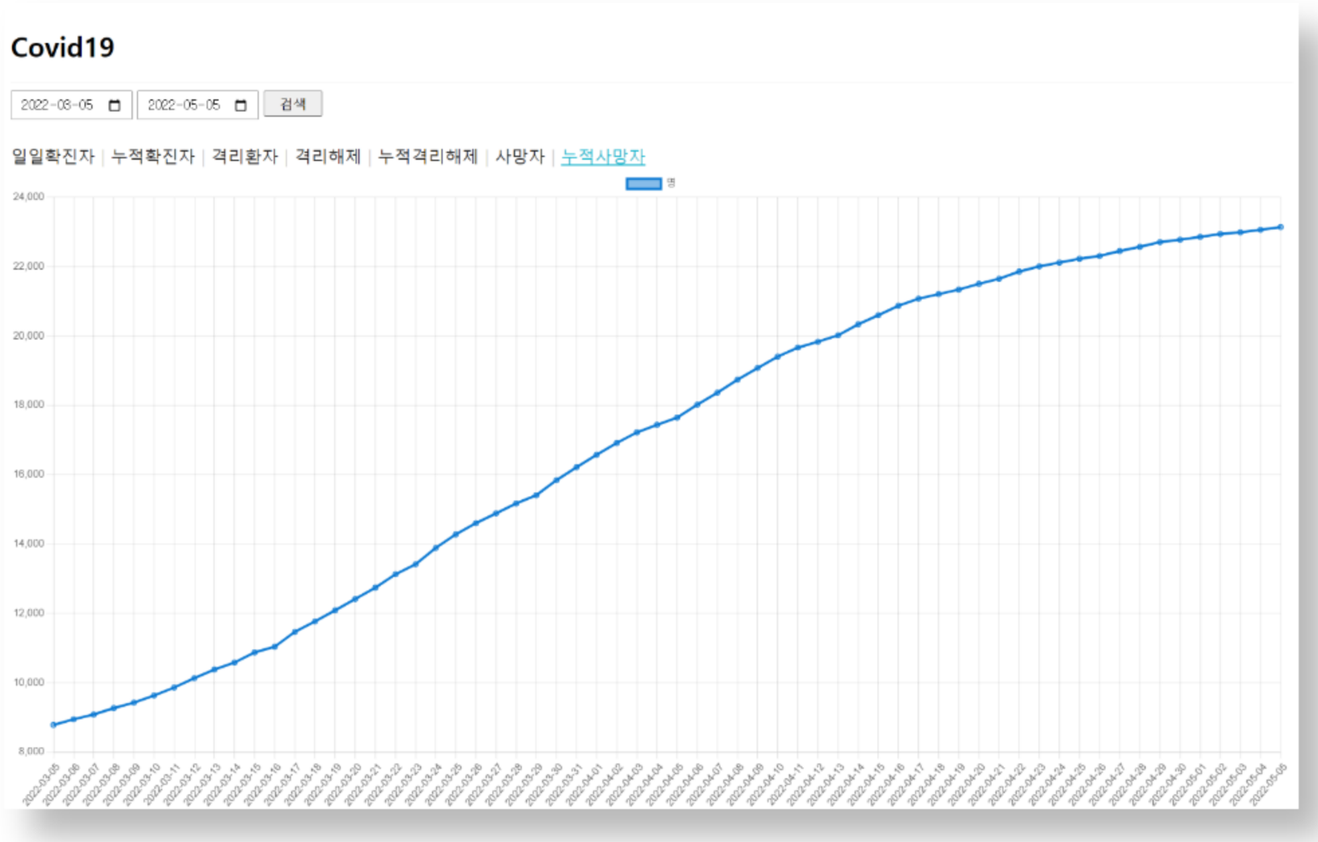
누적격리해제

사망자



01 구현결과

누적사망자



02

구현 결과에 따른 소스코드

01. App.jsx, Index.jsx
02. Store.jsx, Meta.js
03. Covid19Slice.jsx, Covid19.jsx
04. Top.jsx, LineChart.jsx
05. Spinner.jsx, Menu.jsx
06. useQuerystring.jsx, Error.jsx

02 구현결과에 따른 소스코드

App.jsx

```jsx
import React,{ memo } from 'react';
import { Routes, Route } from 'react-router-dom';

import Top from './components/Top';
import Covid19 from './pages/Covid19';

const App = memo(() =>{
    return (
      <div>
        <Top/>
        <Routes>
        <Route path="/covid/*" element={<Covid19 />} />
        </Routes>
      </div>
  );
});

export default App;
```

Index.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';



// test 완료 후, 다시 /를 빼서 App으로 변경해놓기
/**/
import App from './App';
/*/
import App from './Test';
/**/
//test를 위해 App의 경로를 Test로 수정해놓음
import Meta from './Meta';
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './Store';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <Meta/>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </Provider>
  </React.StrictMode>
);
```

02 구현결과에 따른 소스코드

Store.jsx

```jsx
import { configureStore } from "@reduxjs/toolkit";
import Covid19Slice from "./slices/Covid19Slice";

const store = configureStore({
    reducer:{
        covid19:Covid19Slice
    },
    middleware: (getDefaultMiddleware)=>getDefaultMiddleware({serializableCheck:false}),
    devTools: true
});

export default store;
```

Meta.jsx

```jsx
import React from 'react'
import { Helmet, HelmetProvider } from 'react-helmet-async';
import sample from './assets/img/sample.png';

const Meta = (props) => {
  return (
      <HelmetProvider>
        <Helmet>
            <meta charSet='utf-8'/>
            {/* SEO 태그 */}
            <title>{props.title}</title>
            <meta name='description' content={props.description}/>
            <meta name='keywords' content={props.keywords}/>
            <meta name='author' content='website'/>
            <meta name='og:type' content={props.title}/>\
            <meta name='og:description' content={props.description}/>
            <meta name='og:url' content={props.url}/>
            <meta name='og:image' content={props.image}/>
            <link rel='shortcut icon' href={props.image} type='image/png' />
            <link rel='icon' href={props.image} type="image/png"/>
        </Helmet>
    </HelmetProvider>
  );
};
    Meta.defaultProps={
        title: 'React 시험 ',
        description: 'Covid19 data를 활용한 React 시험입니다.',
        author:'이승아',
        image:sample,
        url:window.location.href
    };

export default Meta
```

02 구현결과에 따른 소스코드

Covid19Slice.jsx

```jsx
import {createSlice, createAsyncThunk} from "@reduxjs/toolkit"
import axios from 'axios';

export const getCovid19 = createAsyncThunk("covid19/getCovid19", async(payload,{rejectWithValue})=>{
  let result = null;

  try{
    result = await axios.get('http://localhost:3001/covid19',{
    params:{
      date_gte: payload.gte,
      date_lte: payload.lte,
    }
    });
    console.log(payload.gte, payload.lte)
    // 에러가 발생하더라도 HTTP 상태 코드는 200으로 응답이 오기 때문에 catch문이 동작하지 않는다.
    // 그러므로 직접 에러를 감지해야 한다.
    if (result.data.faultInfo !== undefined){
      const err = new Error();
      err.response = {status:500, statusText: result.data.faultInfo.message};
      throw err;
    }
  }catch(err) {
    result = rejectWithValue(err.response);
  }
  return result;
});

const Covid19Slice = createSlice({
  name: 'covid19',
  initialState: {
    data: null,
    loading: false,
    error:null
  },
  reducers: {},
  extraReducers: {
    [getCovid19.pending]: (state, {payload})=>{
      return{ ...state, loading: true}
    },
    [getCovid19.fulfilled]: (state,{payload})=>{
      return{
        data: payload?.data,
        loading: false,
        error: null
      }
    },
    [getCovid19.rejected]: (state,{payload})=>{
      return{
        data: payload?.date,
        loading:false,
        error:{
          code: payload?.status ? payload.status:500,
          message: payload?.statusText ? payload.statusText:'Server Error'
        }
      }
    }
  },
});

export default Covid19Slice.reducer;
```

Covid19.jsx

```jsx
import React,{ memo } from 'react';
import { Routes, Route} from 'react-router-dom';
import { useSelector, useDispatch } from 'react-redux';
import { getCovid19 } from '../slices/Covid19Slice';

import { useQueryString} from '../hooks/useQueryString';

import Spinner from '../components/Spinner';
import ErrorView from '../components/ErrorView';
import LineChartView from '../components/LineChartView';
import MenuLink from '../components/MenuLink';
/* import dayjs from 'dayjs'; */



const Covid19 = memo(() =>{
  const {data, loading, error} = useSelector((state) => state.covid19);
    const dispatch = useDispatch();
    const {date_gte, date_lte} = useQueryString();

    React.useEffect(()=>{
      dispatch(getCovid19({
        gte: date_gte + "T00:00:00Z",
        lte: date_lte +"T00:00:00Z",
      }))
  },[dispatch, date_gte, date_lte]);

/*   const [chartData, setChartData] = React.useState([]);
  const mountedRef = React.useRef(false);
  React.useEffect(() => {
    setTimeout(() => {
      mountedRef.current = true;
      const newData = {
        date: [],
      };
      data.forEach((v, i) => {
        newData.date.push(dayjs(v.date).format("YYYY-MM-DD"));
      });
      setChartData(newData);
      console.log(newData)
    }, []);
  }, [data]);
console.log(chartData) */

    return (
      <div>
        <nav>
            <MenuLink to={`/covid/confirmed?date_gte=${date_gte}&date_lte=${date_lte}`}>일일확진자</MenuLink>
            <MenuLink to={`/covid/confirmed_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적확진자</MenuLink>
            <MenuLink to={`/covid/active?date_gte=${date_gte}&date_lte=${date_lte}`}>격리환자</MenuLink>
            <MenuLink to={`/covid/released?date_gte=${date_gte}&date_lte=${date_lte}`}>격리해제</MenuLink>
            <MenuLink to={`/covid/released_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적격리해제</MenuLink>
            <MenuLink to={`/covid/death?date_gte=${date_gte}&date_lte=${date_lte}`}>사망자</MenuLink>
            <MenuLink to={`/covid/death_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적사망자</MenuLink>
        </nav>
      <Spinner visible={loading}/>

        {error ? (
            <ErrorView error={error}/>
        ): (
          data && (
            <Routes>
              <Route path='/:category' element={<LineChartView chartData={data}/>}/>
            </Routes>
          )
      )}
      </div>
  );
});

export default Covid19;
```

02 구현결과에 따른 소스코드

Top.jsx

```jsx
import React,{memo, useCallback, useState} from 'react'
import {useNavigate} from 'react-router-dom';

import styled from 'styled-components';



const Form = styled.form`
  background-color: #fff;
  display: flex;
  border-top: 1px solid #eee;
  padding: 10px 0;
  margin: 0;
  margin-bottom: 20px;

  input, button{
    display: block;
    margin-right: 5px;
    font-size: 16px;
    padding: 0 10px;
    height: 30px;
  }
  button{
    width: 70px;
    font-size: 16px;
    flex: none;
  }
`;

const Top=memo(()=> {
  const navigate = useNavigate();
  /*   const dispatch = useDispatch(); */
  // 하루 전 날짜의 데이터 값을 불러오기 위한 설정
  const [targetDt, setTargetDt] = useState({
    date_gte: "",
    date_lte: "",
  });

  // 페이지가  열린 직후와 날짜값이 변경되는 경우 리덕스 액션함수 디스패치 --> Ajax 호출
  /*     React.useEffect(()=>{
      dispatch(getCovid19({targetDt: targetDt}));
  }, [dispatch, targetDt]); */

  // 드롭다운의 선택이 변경된 경우의 리액트
  const onDateChange = useCallback((e) => {
    e.preventDefault();
    // 선택값으로 상태값을 갱신한다. --> React.useEffect()에 의해 액션함수가 디스패치 된다.
    setTargetDt((targetDt) => ({
      ...targetDt,
      [e.target.name]: e.target.value,
    }));
  }, []);

  const onSubmit = (e) => {
    e.preventDefault();
    if (!(targetDt.date_gte && targetDt.date_lte)) return;
    navigate(
      `/covid/confirmed?date_gte=${targetDt.date_gte}&date_lte=${targetDt.date_lte}`
    );
  };


  return (
    <div>
      <h1>Covid19</h1>
      <Form onSubmit={onSubmit}>
        <input type="date" name="date_gte" onChange={onDateChange} />
        <input type="date" name="date_lte" onChange={onDateChange} />
        <button type="submit">검색</button>
      </Form>

    </div>
  );
});

export default Top;
```

LineChart.jsx

```jsx
import React,{memo} from 'react'
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from 'chart.js';
import { Line } from 'react-chartjs-2';
import { useParams } from 'react-router-dom';
import dayjs from 'dayjs';
import useMountedRef from '../hooks/useMountedRef';

ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend
);
const LineChartView = memo (({chartData})=> {

  const params = useParams();
  const {category} = params;
  const mountedRef = useMountedRef();

  console.log(category)
    /** 그래프 옵션 */
    const options={
        //indexAxis: 그래프 축 식별자
        responsive: true,
        plugins: {
          legend: {
            position: 'top'
          },
        },
      };

    /** chart에 표시될 데이터 (막대그래프용) */
    const [data, setData] = React.useState({
        labels: [],
        datasets: [
          {
            label: '명',
            // 선택 옵션에 따라  data 값 영수가 달라짐
            data: [],
            borderColor: 'rgb(13, 122, 211)',
            backgroundColor: 'rgba(13, 122, 211, 0.5)',
          }
        ],
      });

    /** chart에 표시될 데이터 (막대그래프용) */
    const [data, setData] = React.useState({
        labels: [],
        datasets: [
          {
            label: '명',
            // 선택 옵션에 따라  data 값 영수가 달라짐
            data: [],
            borderColor: 'rgb(13, 122, 211)',
            backgroundColor: 'rgba(13, 122, 211, 0.5)',
          }
        ],
      });

  React.useEffect(() => {
    if(mountedRef.current){
      mountedRef.current = true;
      const newData = {
        labels: [],
        datasets: [],
      };
      chartData.forEach((data) => {
        newData.labels.push(dayjs(data.date).format("YYYY-MM-DD"));
        newData.datasets.push(data[category]);
      });

      setData((prevData)=>({
        ...prevData,
        labels: newData.labels,
        datasets: [
          {
            label: '명',
            // 선택 옵션에 따라  data 값 영수가 달라짐
            data: newData.datasets,
          }
        ],
      }));
      console.log(newData)
    };
  }, [mountedRef, chartData, category]);


  return <Line data={data} options={options} />
});

export default LineChartView;
```

02 구현결과에 따른 소스코드

Spinner.jsx

```jsx
import React from 'react'
import PropTypes from 'prop-types';
import styled from 'styled-components';

/* 로딩바 뒤에 표시된 반투명 막 */
// -> http://mhnpd.github.io/react-loader-spinner/
import { Bars } from 'react-loader-spinner';

/** 로딩바 뒤에 표시된 반투명 막 */
const TransLayer = styled.div`
    position: fixed;
    left: 0;
    top: 0;
    z-index: 9999;
    background-color: #0003;
    width: 100%;
    height: 100%;
`;

const Spinner = ({visible, color, width, height}) => {
  return (
    <>
        {visible&&(
            <TransLayer>
                <Bars
                    color={color}
                    height={height}
                    width={width}
                    wrapperStyle={{
                        position:'absolute',
                        zIndex:10000,
                        left:'50%',
                        top: '50%',
                        marginLeft:(-width/2)+'px',
                        marginTop:(-height/2)+'px'
                    }}
                />
            </TransLayer>
        )}
    </>
  );
};
/** 기본값 정의 */
Spinner.defaultProps={
    visible:false,
    color: '#0Gf',
    width:100,
    height: 100
};

/** 데이터 타입 설정 */
Spinner.propTyes ={
    visible: PropTypes.bool.isRequired,
    color: PropTypes.string,
    width: PropTypes.number,
    height: PropTypes.number,
};

export default React.memo (Spinner);
```

Menu.jsx

```jsx
import React from 'react';
import styled from 'styled-components';
import { NavLink } from 'react-router-dom';

/** 메뉴링크 --> NavLink: 현재 머물고 있는 페이지와 관련된 링크에 CSS 적용 */
const MenuLinkContainer = styled (NavLink)`
/* styled component를 이용하여 NavLink를 확장을 시킴
--> 기본적인 CSS는 styled와 결합하고, URL과 mapping 되었을 경우 active가 따로 적용됨  */

    font-size: 20px;
    cursor: pointer;
    text-decoration: none;
    padding-bottom: 2px;
    color: #222;

    /* CSS의 가상클래스 hover */
    &:hover{
        color:#22b8cf;
    }

    &:after{
        content:'|';
        display: inline-block;
        padding: 0 7px;
        color: #ccc;
    }

    &:last-child{
        &:after{
            /* 글자색을 흰색으로 지정하여 화면에서 숨긴다. */
            color:#fff;
        }
    }

    /*
        URL이 현재 메뉴를 가르키는 경우(클릭이 아닌 점에 주의)
        활성 메뉴에 적용되는 기본 클래스 이름이 'active'이다.
    */
   &.active{
       text-decoration: underline;
       color:#22b8cf;

       &:after{
           /* 흰색 선을 추가하여 .active에서 지정한 border를 덮을 수 있도록 지정한다.(가림효과) */
           border-bottom: 4px solid #fff !important;
       }
   }

`;
/* to=NavLink 이고, Children은 그대로 넣는다
그래서 <MenuLinkContainer to={to}>{children}</MenuLinkContainer>를 사용하면
링크를 재사용할 수 있다. */
const MenuLink =({to, children})=> <MenuLinkContainer to={to}>{children}</MenuLinkContainer>
export default React.memo(MenuLink);

/* 위에와 같은 코드를 한번 더 축약한다면 -->
export default ({to, children})=> <MenuLinkContainer to={to}>{children}</MenuLinkContainer> */
```

02 구현결과에 따른 소스코드

useQuerystring.jsx

```jsx
import {useLocation} from 'react-router-dom';

const useQueryString = ()   =>{
    //QueryString 부사열 추출함
    const {search} = useLocation();
    //QueryString 문자열을 객체로 변환
    const params = new  URLSearchParams(search);
    // 모든 key와 value의 쌍을 for...in 반복문으로 처리 가능한 [key, value]쌍의 배열로 반환함.
    // params.entries()는 반복문을 돌릴 수 있는 객체가 됨
    const entries = params.entries();
    const result ={}

    // 추출한 배열을 반복문으로 처리하여 JSON객체로 변환함
    for(const [key, value] of entries){  // each 'entries' is a [key, value]
        // 리턴할 빈 객체에 key와 value을 쌓은 후
        result[key] = value;
    }
    // 리턴해줄 때 hook을 하나 만듦
    return result;
};

export {useQueryString};
```

Error.jsx

```jsx
import React,{memo} from "react";

const ErrorView = memo(({error})=>{
    return(
        <div>
            <h1>Oops~!!! {error.code} Error.</h1>
            <hr/>
            <p>{error.message}</p>
        </div>
    );
});

export default ErrorView;
```

03

문제점 및 소감

03 문제점 및 소감

코드의 흐름을 따라 가는게 중요한데, 시작을 하는 과정에서 흐름을 완전히 이해하고 작업을 하기보다. 예제로 사용했던 것들을 위주로 코드를 작성하다보니. 에러나 그런 것들이 발생 했을 때, 해결을 하는데 어려움을 많이 겪었습니다. 근데 이걸 포기하면 이 과정을 하는 것에 무의미함을 느낄 거 같다는 생각이 들어. 주어진 시간 내에 최선을 다 해보고 싶어. 많이 찾아보고 정말 모르는 것들은 여기저기 물어보면서 문제를 해결을 하려 하였습니다. 그 과정 속에서 코드의 흐름을 이해하는 법에 대해서 더 자세히 알게 되었고, 그래서 문제를 끝까지 풀 수 있었습니다. 많은 우여곡절로 며칠을 이 문제만 붙잡고 있었으나, 그 과정이 힘든 만큼 재미있었던 거 같았습니다.

특히, LineChart를 구현하는 과정 속에서 labels와 dataset data를 불러오는 과정에 흐름 상 어디에 두는게 좋을까? 고민을 해서 여러 방법으로 코드를 작성해보면서 최종적으로 결과를 얻어 낼 수 있었는데. 그 과정에서 코드를 어떻게 짜는 것이 조금 더 쉽게 짤 수 있는지를 많이 고민 했던 과정이었습니다.