1. Home

import classNames from 'classnames/bind';

import React, { useEffect, useState, useRef } from 'react';

import { motion } from 'framer-motion';

import axios from 'axios';

import { useSelector } from 'react-redux';

import styles from './Home.module.scss'

import { PieChart } from '~/component/Chart';

import Detail from '~/component/Detail';

const cx = classNames.bind(styles);

function Home() {

    const [stations, setStations] = useState([]);

    const [devices, setDevices] = useState([]);

    const selectedStationIdRef = useRef(null);

    const unitValue = useSelector((*state*) => *state*.select.selectValueUnit);

    const fieldValue = useSelector((*state*) => *state*.select.selectValueField);

    const fetchStations = async () => {

        try {

            const response = await axios.get('/api/home/station', {

                params: { unit: unitValue, field: fieldValue }

            });

            const stationsData = response.data;

            setStations(stationsData);

            if(selectedStationIdRef.current === null) {

                const firstStationId = stationsData?.[0]?.StationCode ?? null;

                fetchDevices(firstStationId); *// Lấy thiết bị cho trạm đầu tiên*

            }

            if(selectedStationIdRef.current != null) {

                fetchDevices(selectedStationIdRef.current); *// Lấy thiết bị cho trạm đầu tiên*

            }

        } catch (err) {

            console.error('Error fetching stations:', err);

        }

    };

    const fetchDevices = async (*id*) => {

        try {

            const response = await axios.get('/api/home/device', {

                params: { stationCode: *id* }

            });

            const devicesData = response.data;

            setDevices(devicesData);

        } catch (err) {

            console.error('Error fetching devices:', err);

        }

    };

    const handleBoxClick = (*id*) => {

        fetchDevices(*id*);

        selectedStationIdRef.current = *id*;

    };

    useEffect(() => {

        selectedStationIdRef.current = null;

        fetchStations();

*// Tạo interval để gọi fetchStations mỗi 2 phút (120000 milliseconds)*

        const interval = setInterval(() => {

            fetchStations();

        }, 5000); *// 120000 ms = 2 phút*

*// Cleanup interval khi component unmount hoặc khi unitValue, fieldValue thay đổi*

        return () => clearInterval(interval);

*// eslint-disable-next-line react-hooks/exhaustive-deps*

    }, [unitValue, fieldValue]);

    if (stations && stations.length > 0) {

        return (

            <>

                <div *className*='w-100 d-flex'>

                    <div *className*='w-100 d-flex align-items-center'>

                        <span *className*="status-item within-threshold">Đang hoạt động</span>

                        <span *className*="status-item threshold-warning">Dừng hoạt động</span>

                        <span *className*="status-item threshold-exceeded">Lỗi</span>

                        <span *className*="status-item signal-lost">Mất tín hiệu</span>

                    </div>

                </div>

                <div *className*={`ps-4 pe-4 pt-2 pb-2 d-flex flex-column ${cx('content-right')}`}>

                    <div *className*='row mb-2'>

                        {stations.map((*item*) => (

                            <div *key*={*item*.StationCode} *className*='col-2 cursor-pointer mb-3' *onClick*={() => handleBoxClick(*item*.StationCode)}>

                                <div *className*={`w-100 bg-body-tertiary rounded-3 p-2 ${cx("box-item")}`}>

                                    <PieChart *className*="w-100" *data*={*item*} />

                                    <label *className*={`fw-bold fs-5 lh-base mt-3 ps-1 pe-1 ${cx("label-text")}`}>{*item*.StationName} ({*item*.TotalDevices})</label>

                                </div>

                            </div>

                        ))}

                    </div>

                    <motion.div

*initial*={{ y: 100, opacity: 0 }}

*animate*={{ y: 0, opacity: 1 }}

*transition*={{ duration: 0.5 }}

                    >

                        <Detail *data*={devices} />

                    </motion.div>

                </div>

            </>

        )

    } else {

        return (

            <>

                <div *className*='w-100 d-flex'>

                    <div *className*='w-100 d-flex align-items-center'>

                        <span *className*="status-item within-threshold">Đang hoạt động</span>

                        <span *className*="status-item threshold-warning">Dừng hoạt động</span>

                        <span *className*="status-item threshold-exceeded">Lỗi</span>

                        <span *className*="status-item signal-lost">Mất tín hiệu</span>

                    </div>

                </div>

                <div *className*={`ps-4 pe-4 pt-2 pb-2 d-flex flex-column ${cx('content-right')}`}>

                    <h4 *className*="w-100 text-center">Không có dữ liệu</h4>

                </div>

            </>

        )

    }

}

export default Home;

2. Location

import classNames from "classnames/bind";

*// import { useLocation } from 'react-router-dom';*

import React, { useEffect, useState, useCallback } from 'react';

import axios from "axios";

import { useSelector } from 'react-redux';

import styles from "./Location.module.scss";

*// import Map from "~/component/Map";*

*// import { listLocations } from '~/services/LocationsServices';*

import Accordion from 'react-bootstrap/Accordion';

import imageMapping from '~/common/imgMapping'

const cx = classNames.bind(styles);

function Location() {

    const [imgLocation, setImgLocation] = useState();

    const [stations, setStations] = useState();

    const [dataDevice, setDataDevice] = useState();

    const [deviceTotal, setDeviceTotal] = useState();

    const unitValue = useSelector((*state*) => state.select.selectValueUnit);

    const fieldValue = useSelector((*state*) => state.select.selectValueField);

    const fetchData = async () => {

        try {

            const stationsResponse = await axios.get('/api/home/station', {

                params: { unit: unitValue, field: fieldValue }

            });

            const dataDeviceResponse = await axios.get('/api/common/location');

            console.log(stationsResponse.data.filter(*x* => x.StationCode === '03')[0])

            setStations(stationsResponse.data);

            setDeviceTotal(stationsResponse.data[0]);

            setDataDevice(dataDeviceResponse.data);

            if (stationsResponse.data.length > 0) {

                const firstStation = stationsResponse.data[0];

                setActiveButton(firstStation.StationName);

                setImgLocation(firstStation.Location);

            }

        } catch (err) {

            console.error('Error fetching stations:', err);

        }

    };

    const [activeButton, setActiveButton] = useState(null);

    const handleButtonClick = useCallback((*data*) => {

        setDeviceTotal(stations.filter(*x* => x.StationCode === data.StationCode)[0]);

        setActiveButton(data.StationName);

        setImgLocation(data.Location)

*// eslint-disable-next-line react-hooks/exhaustive-deps*

    }, [stations]); *// Chỉ tạo lại hàm khi component mount*

    useEffect(() => {

        fetchData();

*// eslint-disable-next-line react-hooks/exhaustive-deps*

    }, [unitValue, fieldValue])

    return (

        <div *className*='d-flex'>

            <div *className*="position-relative flex-grow-1">

                {*/\* <Map lat={lat} lon={lon} zoom={10} data={machines} mapEvent={updateVisibleMachines} setMap={setMap} /> \*/*}

                <img *className*="w-100 object-fit-cover" *src*={imageMapping[imgLocation]} *alt*='bản đồ' *style*={{ height: '100vh', }} />

                {*/\* <div className='d-flex align-items-center position-absolute status-bar top-0 right-0'>*

*<span className="status-item signal-lost">Mất tín hiệu</span>*

*<span className="status-item threshold-exceeded">Lỗi</span>*

*<span className="status-item threshold-warning">Dừng hoạt động</span>*

*<span className="status-item within-threshold">Đang hoạt động</span>*

*</div> \*/*}

            </div>

            <div *className*="col-3 pt-2 ps-2 pe-2">

                <div>

                    <div *className*={cx('title')}>

                        <span *className*="d-block">Trạng thái</span>

                    </div>

                    <div *className*={`d-flex flex-wrap align-items-center ${cx("status-bar")}`}>

                        <div *className*={`within-threshold position-relative ${cx("status-item")}`}>

                            {*/\* <input className="form-check-input position-absolute" type="checkbox" value="" id="signalLost" defaultChecked={true} /> \*/*}

                            <label *className*="form-check-label d-flex flex-column justify-content-end h-100" *htmlFor*="withinThreshold">

                                <span *className*="mb-3">Đang hoạt động</span>

                                <span *className*="mb-2">{deviceTotal?.Status === 'NS' ?? 0 ? 0 : deviceTotal?.TotalActive ?? 0}</span>

                            </label>

                        </div>

                        <div *className*={`threshold-warning position-relative ${cx("status-item")}`}>

                            {*/\* <input className="form-check-input position-absolute" type="checkbox" value="" id="thresholdExceeded" defaultChecked={true} /> \*/*}

                            <label *className*="form-check-label d-flex flex-column justify-content-end h-100" *htmlFor*="thresholdWarning">

                                <span *className*="mb-3">Dừng hoạt động</span>

                                <span *className*="mb-2">{deviceTotal?.Status === 'NS' ?? 0 ? 0 : deviceTotal?.TotalStopped ?? 0}</span>

                            </label>

                        </div>

                        <div *className*={`border position-relative ${cx("status-item")}`}>

                            {*/\* <input className="form-check-input position-absolute" type="checkbox" value="" id="thresholdWarning" defaultChecked={true} /> \*/*}

                            <label *className*="form-check-label d-flex flex-column justify-content-end h-100" *htmlFor*="thresholdWarning">

                                <span *className*="mb-3 text-black">Mực nước</span>

                                <span *className*="mb-2 text-black">{dataDevice ? dataDevice[0]?.MucNuoc : 0} m</span>

                            </label>

                        </div>

                        <div *className*={`border position-relative ${cx("status-item")}`}>

                            {*/\* <input className="form-check-input position-absolute" type="checkbox" value="" id="withinThreshold" defaultChecked={true} /> \*/*}

                            <label *className*="form-check-label d-flex flex-column justify-content-end h-100" *htmlFor*="thresholdWarning">

                                <span *className*="mb-3 text-black">Lưu lượng nước</span>

                                <span *className*="mb-2 text-black">{dataDevice ? dataDevice[0]?.LuuLuongNuoc : 0} m³/h</span>

                            </label>

                        </div>

                    </div>

                    <div>

                        <div *className*={cx('title')}>

                            <span *className*="d-block">Tất cả</span>

                        </div>

                        <div *className*={cx('list-container')}>

                            <div *className*="input-group rounded mt-2 mb-2 position-relative pe-1">

                                <input *type*="search" *className*="form-control rounded bg-transparent ps-5" *placeholder*="Search" *aria-label*="Search" *aria-describedby*="search-addon" />

                                <span *className*="input-group-text h-100 border-0 bg-transparent position-absolute" *id*="search-addon">

                                    <i *className*="fas fa-search"></i>

                                </span>

                            </div>

                            <Accordion *defaultActiveKey*="0" *className*={cx("accordion-container")}>

                                <Accordion.Item *eventKey*="0">

                                    <Accordion.Header *className*={cx("header-list")}>Nước sạch</Accordion.Header>

                                    <Accordion.Body>

                                        <div *className*={`list-group list-group-flush ${cx('button-group')}`}>

                                            {stations?.map((*item*, *index*) => (

                                                <button

*key*={*index*}

*onClick*={() => handleButtonClick(*item*)}

*type*="button"

*className*={classNames(

                                                        'list-group-item list-group-item-action',

                                                        { 'active': *item*.StationName === activeButton }

                                                    )}

                                                >

                                                    {*item*.StationName}

                                                </button>

                                            ))}

                                        </div>

                                    </Accordion.Body>

                                </Accordion.Item>

                            </Accordion>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    )

}

export default Location;

3. HomeController

import db from "../Dbcontext.js";

import {

    GET\_STATUS\_BY\_UNIT\_AND\_FIELD,

    GET\_DEVICES\_BY\_STATION,

    GET\_DATA\_LUU\_LUONG\_NUOC,

    GET\_DATA\_TRANG\_THAI,

    GET\_DATA\_MUC\_NUOC

} from "../query/homeQuery.js";

export const getStations = async (*req*, *res*) => {

    const { unit, field } = req.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_STATUS\_BY\_UNIT\_AND\_FIELD, [field, unit], (*err*, *rows*) => {

                if (err) {

                    reject(err);

                    return;

                }

                resolve(rows);

            });

        });

        res.status(200).json(data);

    } catch (err) {

        console.error('getStations\_homeController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

};

export const getDeviceByStation = async (*req*, *res*) => {

    const { stationCode } = *req*.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_DEVICES\_BY\_STATION, [stationCode], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getStations\_homeController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

};

export const getDataMucNuoc = async (*req*, *res*) => {

    const { date } = *req*.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_DATA\_MUC\_NUOC, [date], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getStations\_homeController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

}

export const getDataLuuLuongNuoc = async (*req*, *res*) => {

    const { date } = *req*.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_DATA\_LUU\_LUONG\_NUOC, [date], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getStations\_homeController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

}

export const getDataTrangThai = async (*req*, *res*) => {

    const { date } = *req*.query;

    const startTime = `${date} 00:00:00`;

    const endtTime = `${date} 23:59:59`;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_DATA\_TRANG\_THAI, [date, startTime, endtTime], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getStations\_homeController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

}

4. CommonController

import moment from 'moment/moment.js'

import db from '../Dbcontext.js'

import { GET\_ALL\_FIELDS } from '../query/fieldQuery.js'

import { GET\_ALL\_UNIT } from '../query/unitQuery.js'

import { INSERT\_TABLE\_DEVICEDATA, GET\_MUCNC\_LUULUONG } from '../query/commonQueries.js'

import { GET\_DATA\_DEVICE\_USAGE\_BY\_DAY } from '../query/statisticsQuery.js'

export const getAllUnit = async (*req*, *res*) => {

    const { unitCode } = req.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_ALL\_UNIT, [unitCode, unitCode, unitCode], (*err*, *rows*) => {

                if (err) {

                    reject(err);

                    return;

                }

                resolve(rows);

            });

        });

        res.status(200).json(data);

    } catch (err) {

        console.error('getAllUnit\_commonController:', err);

        res.status(500).json({ message: 'Internal server error' });

    }

};

export const getAllField = async (*req*, *res*) => {

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_ALL\_FIELDS, [], (*err*, *rows*) => {

                if (err) {

                    reject(err);

                    return;

                }

                resolve(rows);

            });

        });

        res.status(200).json(data);

    } catch (err) {

        console.error('getAllField\_commonController:', error);

*res*.status(500).json({ message: 'Internal server error' });

    }

};

export const getMucNCLuuLuong = async (*req*, *res*) => {

    const currentDate = moment().format('YYYY-MM-DD');

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_MUCNC\_LUULUONG, [currentDate], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getMucNCLuuLuong\_commonController:', err);

*res*.status(500).json({ message: 'Internal server error' });

    }

}

export const insertDeviceData = (*data*) => {

    const currentDate = moment().format('YYYY-MM-DD HH:mm:ss');

    return new Promise((*resolve*, *reject*) => {

        db.run(INSERT\_TABLE\_DEVICEDATA, [*data*[0], *data*[2], *data*[1], currentDate], (*err*) => {

            if (*err*) {

                console.log(*err*.message);

                reject(*err*);

                return;

            }

            if (*data*[0] === '#00000001') {

*// Update D03 với status là 'A', D02 với status là 'S'*

                updateDeviceStatus('D03', 'A');

                updateDeviceStatus('D02', 'S');

            } else if (*data*[0] === '#00000010') {

*// Update D03 với status là 'S', D02 với status là 'A'*

                updateDeviceStatus('D03', 'S');

                updateDeviceStatus('D02', 'A');

            }

        })

    });

}

*// Hàm để thực hiện thao tác update*

const updateDeviceStatus = (*deviceID*, *status*) => {

    return new Promise((*resolve*, *reject*) => {

        const updateQuery = `UPDATE Devices SET Status = ? WHERE DeviceCode = ?`;

        db.run(updateQuery, [*status*, *deviceID*], (*err*) => {

            if (*err*) {

                console.log(*err*.message);

                reject(*err*);

            } else {

                resolve();

            }

        });

    });

};

*// biểu đồ trang thống kê*

export const getDataDeviceUsageByDay = async (*req*, *res*) => {

    const { dateStart, dateEnd } = *req*.query;

    try {

        const data = await new Promise((*resolve*, *reject*) => {

            db.all(GET\_DATA\_DEVICE\_USAGE\_BY\_DAY, [dateStart, dateEnd, dateStart, dateEnd], (*err*, *rows*) => {

                if (*err*) {

                    reject(*err*);

                    return;

                }

                resolve(*rows*);

            });

        });

*res*.status(200).json(data);

    } catch (err) {

        console.error('getMucNCLuuLuong\_commonController:', err);

*res*.status(500).json({ message: 'Internal server error' });

    }

}