**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHÓA D20**

**🙟🕮🙝**



**Báo Cáo Lập Trình Web**

**Đề tài**

**Hệ thống thống kê nhân sự**

**Nhóm Lớp: 09**

**Giảng viên: Dương Trần Đức**

**Sinh viên:**

* Lê Phan Thanh Bình - B20DCCN093
* Nguyễn Đức Duy - B20DCCN154
* Nguyễn Thị Ngọc Hân - B20DCCN232

***Hà Nội , 2023***



**Phần 1: Giao diện+ Front End**

**Trang app(chứa thành phần gốc của ứng dụng):**

import AppLoading from '@/assets/appLoading.json';

import { Button, ConfigProvider, notification, Row, Spin, Timeline } from 'antd';

import vi\_VN from 'antd/lib/locale/vi\_VN';

import Lottie from 'lottie-react';

import moment from 'moment';

import React from 'react';

import styled from 'styled-components';

import LocalStorage from './apis/LocalStorage';

import GlobalStyle from './config/global.style';

import { BOX\_SHADOW } from './config/theme';

import { APP\_LOADING, SET\_BG\_APP, SET\_COUNT\_NOTI, SET\_INFO, SET\_KIOTVIETS, SET\_SOCKET } from './context/types';

import MainPage from './features/MainPage';

import useCallContext from './hooks/useCallContext';

import { appService } from './service';

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

import axios from 'axios';

const DEV\_TYPE = import.meta.env.VITE\_DEVOPS\_TYPE;

moment.utc().locale('vi');

function App() {

    const { state, dispatch } = useCallContext();

    const [role, setRole] = React.useState('');

    const headers = {

        'Content-Type': 'application/json',

        Authorization: `Bearer ${LocalStorage.getToken()}`,

    };

    const location = useLocation();

    // loading when going to app

    React.useEffect(() => {

        setTimeout(() => {

            dispatch({ type: APP\_LOADING, payload: false });

        }, 2500);

    }, []);

    React.useLayoutEffect(() => {

        if (LocalStorage.getToken()) {

            axios.get('http://26.75.181.165:8080/getrole', { headers }).then((res) => {

                setRole(res.data.role);

                dispatch({

                    type: SET\_INFO,

                    payload: { ...res.data[Object.keys(res.data)[0]], role: res.data.role },

                });

            });

        }

        if (LocalStorage.getBG()) {

            dispatch({

                type: SET\_BG\_APP,

                payload: JSON.parse(LocalStorage.getBG() as any),

            });

        }

    }, []);

    return (

        <SpinLoadingStyled

            spinning={state.appLoading}

            indicator={

                <ContainerLoading>

                    <div style={{ height: '600px', width: '600px' }}>

                        <Lottie animationData={AppLoading} loop={true} />

                    </div>

                </ContainerLoading>

            }

        >

            <ConfigProvider locale={vi\_VN}>

                <MainPage role={role} />

            </ConfigProvider>

            {/\* define default style \*/}

            <GlobalStyle />

        </SpinLoadingStyled>

    );

}

const SpinLoadingStyled = styled(Spin)`

    &&& {

        top: 0;

        left: 0;

    }

`;

const ContainerLoading = styled.div`

    top: 0 !important;

    margin: 0 !important;

    left: 0 !important;

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    background-color: white;

`;

const ContainerLoadingSync = styled.div`

    top: 0 !important;

    margin: 0 !important;

    left: 0 !important;

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    position: fixed;

    background-color: rgba(0, 0, 0, 0.4);

    z-index: 9999;

`;

const ContainerLoad = styled.div`

    background-color: white;

    border-radius: 20px;

    box-shadow: ${BOX\_SHADOW};

    padding: 40px;

    max-width: 500px;

`;

export default App;

1. Import các thành phần và thư viện:
   * **AppLoading** được import từ **@/assets/appLoading.json** là một file JSON chứa dữ liệu của animation.
   * **Button**, **ConfigProvider**, **notification**, **Row**, **Spin**, **Timeline** được import từ thư viện Ant Design.
   * **vi\_VN** được import từ **antd/lib/locale/vi\_VN** là định dạng ngôn ngữ tiếng Việt cho Ant Design.
   * **Lottie** được import từ **lottie-react** để hiển thị animation.
   * **moment** được import từ **moment** để làm việc với thời gian.
   * **React**, **styled**, **LocalStorage**, **GlobalStyle**, **useCallContext**, **useLocation** và **axios** là các thư viện và hooks được sử dụng trong ứng dụng.
2. Định nghĩa hằng số **DEV\_TYPE** để lưu trữ giá trị của biến môi trường **VITE\_DEVOPS\_TYPE**.
3. Sử dụng hook **useCallContext** để lấy giá trị **state** và **dispatch** từ context.
4. Sử dụng hook **useState** để lưu trữ giá trị **role** của người dùng.
5. Định nghĩa biến **headers** chứa các thông tin header gửi kèm trong các yêu cầu HTTP, bao gồm **'Content-Type'** và **'Authorization'** dựa trên token được lưu trữ trong **LocalStorage**.
6. Sử dụng hook **useLocation** để lấy giá trị của **location**.
7. Sử dụng hook **useEffect** để đảm bảo rằng trạng thái **appLoading** trong **state** được thiết lập thành **false** sau khoảng thời gian 2.5 giây. Điều này cho phép hiển thị màn hình tải trong một khoảng thời gian ngắn khi ứng dụng khởi động.
8. Sử dụng hook **useLayoutEffect** để thực hiện các công việc sau khi rendering đã hoàn tất:
   * Kiểm tra xem người dùng đã đăng nhập hay chưa bằng cách kiểm tra sự tồn tại của token trong **LocalStorage**. Nếu đã đăng nhập, gửi yêu cầu HTTP GET đến URL **'http://26.75.181.165:8080/getrole'** để lấy thông tin về
9. Trả về JSX để hiển thị giao diện của ứng dụng:

- Sử dụng thành phần **SpinLoadingStyled** từ thư viện Ant Design để hiển thị màn hình tải. Thành phần này được sử dụng để bao bọc các phần tử con và hiển thị màn hình tải khi **state.appLoading** là **true**.

- Truyền prop **spinning** với giá trị **state.appLoading** vào thành phần **SpinLoadingStyled**.

- Sử dụng prop **indicator** để cung cấp nội dung của màn hình tải. Trong trường hợp này, nội dung là một thành phần **ContainerLoading** bao gồm một đoạn mã JSX để hiển thị animation.

- Sử dụng thành phần **ConfigProvider** từ thư viện Ant Design để cung cấp cấu hình địa phương cho các thành phần Ant Design. Trong trường hợp này, cấu hình địa phương được đặt thành tiếng Việt (**vi\_VN**).

- Hiển thị thành phần **MainPage** với prop **role** được truyền vào. Thành phần này sẽ hiển thị giao diện chính của ứng dụng dựa trên vai trò của người dùng.

- Hiển thị thành phần **GlobalStyle** để định nghĩa các kiểu CSS toàn cục cho ứng dụng.

1. Định nghĩa các kiểu CSS cho các thành phần:

**- SpinLoadingStyled** là một thành phần **Spin** với một số kiểu CSS tùy chỉnh để đặt vị trí và kích thước của nó.

**- ContainerLoading** là một thành phần **div** có các kiểu CSS để căn giữa nội dung của nó trong màn hình và tạo một phông nền trắng cho nó.

**- ContainerLoadingSync** và **ContainerLoad** là các thành phần **div** tương tự như **ContainerLoading**, nhưng được sử dụng để hiển thị màn hình tải cho các tác vụ đồng bộ khác nhau.

**Trang MainPage(thành phần chính của ứng dụng)**

import LocalStorage from '@/apis/LocalStorage';

import { routerPage } from '@/config/contants.routes';

import { PublicRoutes } from '@/config/Lazy.routes';

import { AccountantRoutes, AdminRoutes, AuthRoutes, EditorRoutes } from '@/config/routes';

import { ADMIN } from '@/contants';

import PageLayout from '@/layout';

import React from 'react';

import { useLocation, useNavigate, useRoutes } from 'react-router-dom';

const switchRoute = (role: string) => {

    // 'group'

    switch (role) {

        case ADMIN.main:

            return AdminRoutes;

        case ADMIN.employee:

            return AccountantRoutes;

        default:

            return PublicRoutes;

    }

};

// config routes

const MainPage = ({ role }: { role: string }) => {

    const navigate = useNavigate();

    const { pathname } = useLocation();

    let element = useRoutes(LocalStorage.getToken() ? switchRoute(role) : AuthRoutes);

    const [logged, setLogged] = React.useState(false);

    React.useEffect(() => {

        if (!role) return;

    }, [role]);

    React.useEffect(() => {

        // nếu đăng nhập và domain không webview và domain không public

        // if (LocalStorage.getToken() && pathname.includes('webview') && pathname.includes('public')) {

        if (LocalStorage.getToken()) {

            setLogged(true);

            if (pathname === routerPage.register || pathname === routerPage.login) {

                // return switchSidebar(role)?.[0]?.key;

                if (role === 'ROLE\_EMPLOYEE') {

                    return navigate('/calendar');

                }

                if (role === 'ROLE\_ADMIN') {

                    return navigate('/');

                }

            }

            // navigate(pathname);

        } else {

            switch (pathname) {

                case routerPage.register:

                    navigate(routerPage.register);

                    break;

                default:

                    navigate(routerPage.login);

                    break;

            }

        }

    }, [logged, pathname, role]);

    return element;

};

export default PageLayout(MainPage);

1. Truyền prop **role** vào thành phần **MainPage** để xác định vai trò của người dùng.
2. Sử dụng hook **useLocation** và **useNavigate** từ thư viện **react-router-dom** để lấy vị trí hiện tại và điều hướng trang.
3. Định nghĩa hàm **switchRoute** để xác định và trả về các tuyến dẫn tương ứng dựa trên vai trò của người dùng. Hàm này sẽ nhận giá trị **role** và sử dụng câu lệnh **switch** để xác định vai trò và trả về các tuyến dẫn tương ứng. Nếu vai trò không khớp với bất kỳ trường hợp nào, thì tuyến dẫn **PublicRoutes** sẽ được trả về.
4. Sử dụng hook **useRoutes** từ thư viện **react-router-dom** để ánh xạ các tuyến dẫn và hiển thị nội dung tương ứng. Thành phần **element** sẽ chứa nội dung được trả về từ hook **useRoutes**, dựa trên vai trò và trạng thái đăng nhập.
5. Sử dụng hook **useEffect** để kiểm tra và điều hướng trang dựa trên trạng thái đăng nhập, vai trò và vị trí hiện tại của trang. Nếu người dùng đã đăng nhập (**LocalStorage.getToken()** trả về **true**), thì **logged** sẽ được đặt thành **true**. Sau đó, ta sẽ xác định xem người dùng có được điều hướng đến trang đăng ký (**routerPage.register**) hay trang đăng nhập (**routerPage.login**) hay không. Nếu vai trò là "ROLE\_EMPLOYEE", ta sẽ điều hướng đến trang **/calendar**. Nếu vai trò là "ROLE\_ADMIN", ta sẽ điều hướng đến trang gốc (**/**). Nếu không phải trường hợp trên, ta sẽ điều hướng đến **pathname** hiện tại.
6. Nếu người dùng chưa đăng nhập (**LocalStorage.getToken()** trả về **false**), ta sẽ kiểm tra **pathname** hiện tại và điều hướng đến trang đăng ký (**routerPage.register**) nếu **pathname** trùng khớp, hoặc điều hướng đến trang đăng nhập (**routerPage.login**) trong trường hợp khác.

**Trang login (giao diện trang đăng nhâp)**

import LocalStorage from '@/apis/LocalStorage';

import { Notification, wait } from '@/utils';

import { errorValidPhone } from '@/utils/validation';

import { Button, Form, Input } from 'antd';

import React from 'react';

import Wrapper from '../../Wrapper';

import InfoLogin from '../components/InfoLogin';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

const LoginPage = () => {

    const [loading, setLoading] = React.useState(false);

    const navigate = useNavigate();

    const handleSubmit = async (data: any) => {

        setLoading(true);

        const res = await axios.post('http://26.75.181.165:8080/login', {

            username: data?.username,

            password: data?.password,

        });

        if (res.status) {

            const token = res.headers['authorization'];

            LocalStorage.setToken(token);

            const headers = {

                'Content-Type': 'application/json',

                Authorization: `Bearer ${LocalStorage.getToken()}`,

            };

            Notification('success', 'Đăng nhập thành công');

            wait(1500).then(() => {

                window.location.reload();

                setLoading(false);

            });

        } else {

            setLoading(false);

        }

    };

    return (

        <div style={{ height: '100vh', width: '100vw' }}>

            <div className="gx-app-login-wrap">

                <div className="gx-app-login-container">

                    <Wrapper loading={loading}>

                        <div className="gx-app-login-main-content">

                            <InfoLogin />

                            <div className="gx-app-login-content">

                                <Form onFinish={handleSubmit} className="gx-signin-form gx-form-row">

                                    <Form.Item

                                        name="username"

                                        rules={[{ required: true, message: 'Vui lòng nhập tài khoản' }]}

                                    >

                                        <Input placeholder="Nhập tài khoản" />

                                    </Form.Item>

                                    <Form.Item

                                        className="gx-mb-1"

                                        name="password"

                                        rules={[{ required: true, message: 'Vui lòng nhập mật khẩu!' }]}

                                    >

                                        <Input.Password type="password" placeholder="Nhập mật khẩu" />

                                    </Form.Item>

                                    <Form.Item className="gx-mt-4 gx-d-flex gx-justify-content-end">

                                        <Button type="primary" htmlType="submit" className="gx-mb-0">

                                            Đăng nhập

                                        </Button>

                                    </Form.Item>

                                </Form>

                            </div>

                        </div>

                    </Wrapper>

                </div>

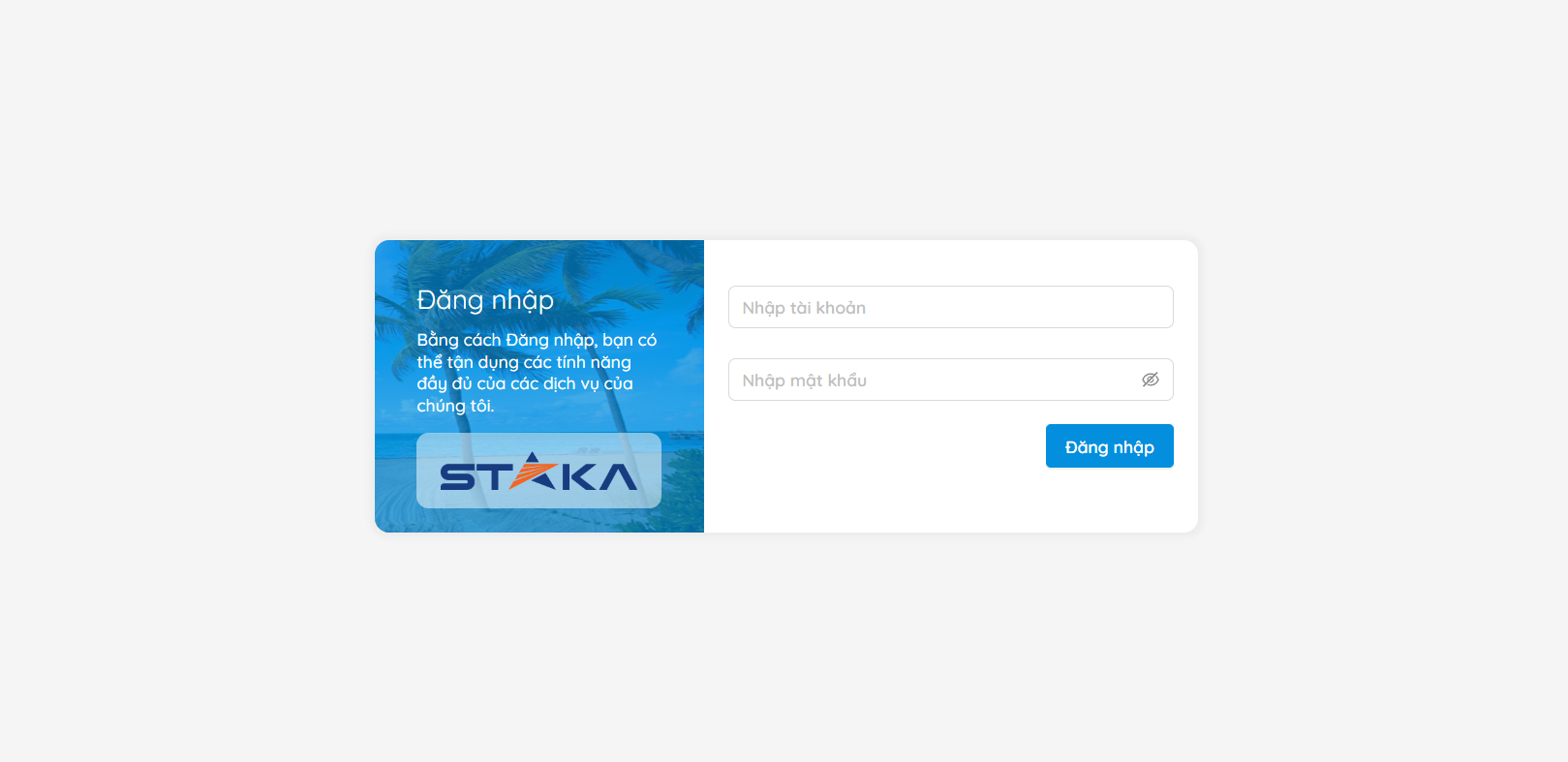
            </div>

        </div>

    );

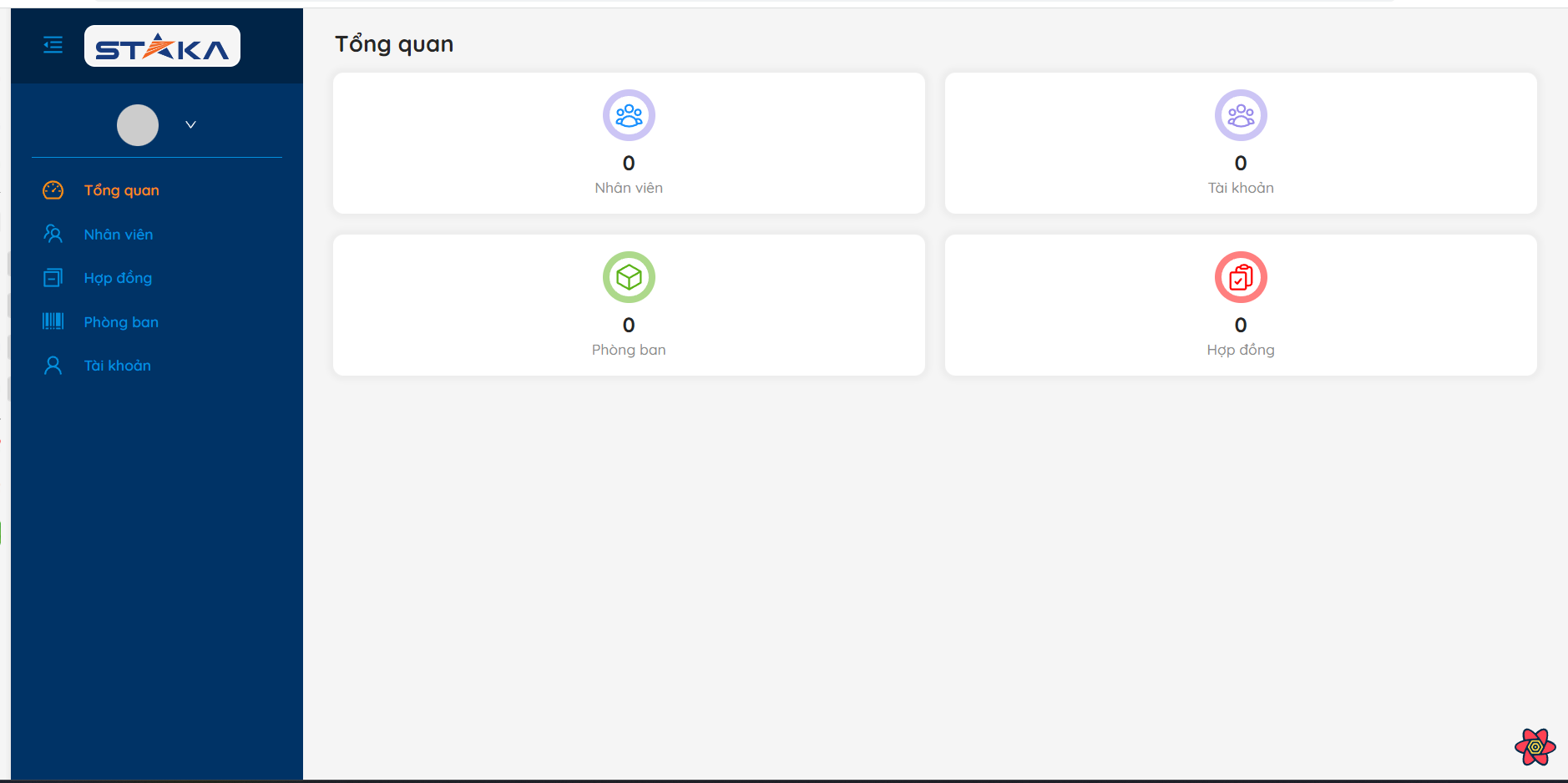
};

export default LoginPage;

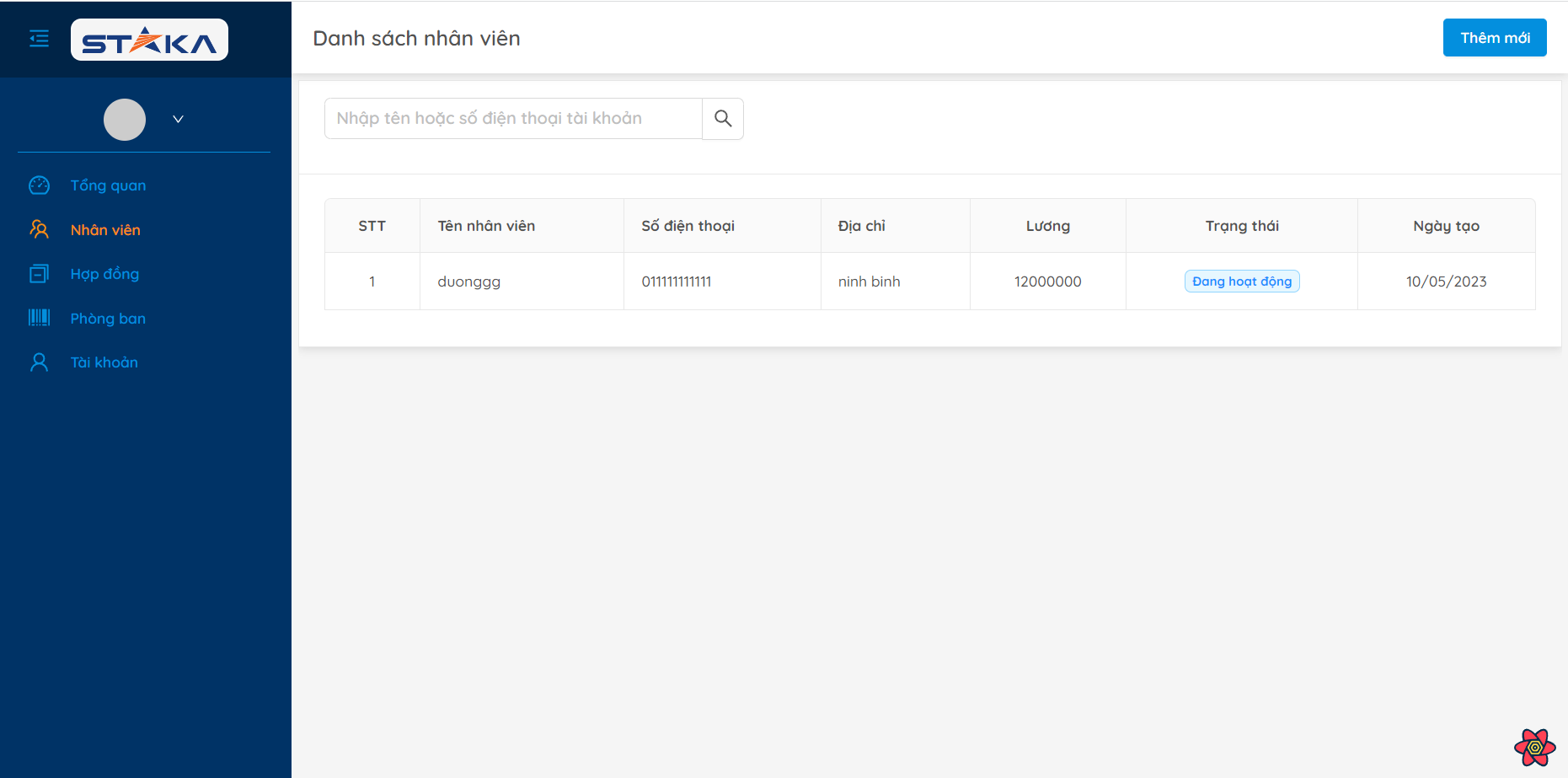
1. Đầu tiên, ta khai báo các biến state bằng cách sử dụng **React.useState()**. Trong trường hợp này, có biến **loading** để xác định trạng thái tải của trang và **setLoading** để cập nhật giá trị của biến **loading**. Có cũng có biến **navigate** được tạo bằng cách sử dụng **useNavigate()** từ thư viện **react-router-dom** để điều hướng trang.
2. Hàm **handleSubmit** được gọi khi người dùng gửi form đăng nhập. Trong hàm này, trạng thái **loading** được đặt thành **true** để hiển thị biểu tượng tải trên trang. Sau đó, ta gửi một yêu cầu POST đến URL '<http://26.75.181.165:8080/login>' để xác thực thông tin đăng nhập người dùng. Dữ liệu đăng nhập gửi đi là **username** và **password** từ form.
3. Sau khi nhận được kết quả từ yêu cầu POST, ta kiểm tra **res.status** để xác định xem đăng nhập có thành công hay không. Nếu thành công, ta lưu token nhận được từ phản hồi vào Local Storage bằng **LocalStorage.setToken(token)**. Sau đó, ta cập nhật tiêu đề và giá trị token trong header bằng cách sử dụng biến **headers**.
4. Một thông báo thành công được hiển thị bằng hàm **Notification** từ thư viện **antd**.
5. Tiếp theo, ta sử dụng hàm **wait** để chờ 1,5 giây trước khi tải lại trang bằng cách sử dụng **window.location.reload()**. Cuối cùng, trạng thái **loading** được đặt lại thành **false** để ẩn biểu tượng tải.
6. Trong phần giao diện, trang được chia thành các phần tử HTML và các thành phần của **antd** như **Form**, **Input**, và **Button**. Ta sử dụng **Form** để xây dựng form đăng nhập với hai trường **username** và **password**. Các trường này đều có quy tắc yêu cầu không được để trống.
7. Khi người dùng nhấp vào nút "Đăng nhập", hàm **handleSubmit** sẽ được gọi để xử lý việc gửi dữ liệu đăng nhập.
8. Toàn bộ trang được bọc bên trong một **Wrapper** component để xử lý trạng thái tải và hiển thị nội dung tương ứng.
9. CSS được áp dụng để định dạng giao diện của trang. Đoạn mã CSS này có thể được tìm thấy trong các file CSS tương ứng và được áp dụng cho các phần tử HTML trong trang. Ví dụ: **gx-app-login-wrap**, **gx-app-login-container**, **gx-app-login-main-content**, **gx-app-login-content**, **gx-signin-form**, **gx-mb-1**, **gx-mt-4**,...

Giao diện trang admin khi đăng nhập

**Trang tổng quan**



**Trang nhân viên**



import CardComponent from '@/components/CardComponent';

import ClearFilter from '@/components/ClearFilter';

import ClearFilterLoading from '@/components/ClearFilter/ClearFilter.Loading';

import TableComponent from '@/components/TableComponent';

import TopBar from '@/components/TopBar';

import useCallContext from '@/hooks/useCallContext';

import { DeleteOutlined, EditOutlined } from '@ant-design/icons';

import Container from '@/layout/Container';

import { downloadFile, handleObjectEmpty, wait, Notification } from '@/utils';

import React, { useState } from 'react';

import { useQuery } from 'react-query';

import { IFilter } from '../../voucher/type';

import { columns, DataTypeEmployee } from '../components/Employee.Config';

import Description from '../components/Description';

import Filter from '../components/Filter';

import { CustomerService } from '../service';

import axios from 'axios';

import Buttons from '../../voucher/components/Buttons';

import { Button, Form, Input, Popconfirm, Space, message } from 'antd';

import EmployeeFormPage from './EmployeeFormPage';

import useDebounce from '@/hooks/useDebounce';

import LocalStorage from '@/apis/LocalStorage';

import { useNavigate } from 'react-router-dom';

import { routerPage } from '@/config/contants.routes';

const initialFilterQuery = {};

const EmployeePage = () => {

    const { Search } = Input;

    const navigate = useNavigate();

    const { state } = useCallContext();

    const [callBack, setCallBack] = useState(false);

    const [isModalOpen, setIsModalOpen] = useState(false);

    const [values, setValues] = useState<DataTypeEmployee | null>();

    const [filterQuery, setFilterQuery] = React.useState<any>('');

    const [page, setPage] = React.useState(1);

    const token = LocalStorage.getToken();

    const headers = {

        'Content-Type': 'application/json',

        Authorization: `Bearer ${token}`,

    };

    const {

        data: employee,

        refetch,

        isRefetching,

    } = useQuery<any>(['employee', page, filterQuery], () =>

        axios.get(`http://26.75.181.165:8080/employee/allemployees/${filterQuery}`, { headers })

    );

    const initialValue = {

        name: '',

        username: '',

        salary: '',

        email: '',

        avatar: '',

        phoneNumber: '',

        password: '',

    };

    const [form] = Form.useForm();

    const [loadingClearFilter, setLoadingClearFilter] = React.useState(false);

    React.useEffect(() => {

        refetch();

    }, [state.syncLoading]);

    const onSearch = (search: string) => {

        setFilterQuery(search);

        setPage(1);

    };

    const onClearFilter = () => {

        setLoadingClearFilter(true);

        wait(1500).then(() => {

            setFilterQuery('');

            setPage(1);

            setLoadingClearFilter(false);

        });

    };

    const handleCallBack = () => {

        setCallBack(!callBack);

    };

    const handleOk = () => {

        setIsModalOpen(false);

    };

    const formReset = () => {

        form.setFieldsValue(initialValue);

    };

    const handleCancel = () => {

        setValues(null);

        setIsModalOpen(false);

        formReset();

    };

    return (

        <>

            <TopBar

                title="Danh sách nhân viên"

                extra={[

                    loadingClearFilter ? <ClearFilterLoading key="clear\_filter" /> : <></>,

                    <Button

                        key="add\_employee"

                        onClick={() => {

                            setIsModalOpen(true);

                        }}

                        className="gx-mb-0"

                        type="primary"

                    >

                        Thêm mới

                    </Button>,

                ]}

            />

            <Container>

                <CardComponent

                    title={

                        loadingClearFilter ? (

                            <ClearFilterLoading key="clear\_filter" />

                        ) : (

                            <Space>

                                <Search

                                    size="large"

                                    placeholder="Nhập tên hoặc số điện thoại tài khoản"

                                    onChange={(e: any) => {

                                        onSearch(e?.target?.value);

                                    }}

                                    style={{ width: 400 }}

                                />

                            </Space>

                        )

                    }

                >

                    <TableComponent

                        reLoadData={() => refetch()}

                        showTotalResult

                        loading={isRefetching}

                        page={page}

                        rowSelect={false}

                        onChangePage={(\_page) => setPage(\_page)}

                        dataSource={employee ? employee.data : []}

                        onRowClick={(record: { id: number }) => navigate(`${routerPage.employee}/${record.id}`)}

                        columns={[...columns(page)]}

                        total={employee && employee?.paging?.totalItemCount}

                    />

                    <EmployeeFormPage

                        handleCallBack={handleCallBack}

                        values={values}

                        handleCancel={handleCancel}

                        handleOk={handleOk}

                        open={isModalOpen}

                        refetch={refetch}

                    />

                </CardComponent>

            </Container>

            <ClearFilter

                hidden={

                    Object.values(handleObjectEmpty(filterQuery))?.filter(

                        (item: any) => item !== undefined && item !== ''

                    ).length > 0

                }

                onClick={onClearFilter}

            />

        </>

    );

};

export default EmployeePage;

1. Sử dụng các thành phần như **CardComponent**, **TableComponent**, **TopBar** và **ClearFilter** để hiển thị danh sách nhân viên và các chức năng tương ứng trên trang.
2. Sử dụng hook **useCallContext** để lấy trạng thái và dispatch từ ngữ cảnh gọi.
3. Sử dụng hook **useState** để quản lý trạng thái và điều khiển mở/đóng của modal.
4. Sử dụng hook **useQuery** từ thư viện **react-query** để thực hiện truy vấn dữ liệu từ API. Dữ liệu nhân viên được lấy từ **axios.get** với URL **http://26.75.181.165:8080/employee/allemployees/${filterQuery}** và headers chứa token

- const token = LocalStorage.getToken();

    const headers = {

        'Content-Type': 'application/json',

        Authorization: `Bearer ${token}`,

    };

* . **token** được lấy từ Local Storage thông qua hàm **LocalStorage.getToken()**.
* **headers** là một đối tượng chứa thông tin xác thực gồm loại nội dung của yêu cầu (**'Content-Type': 'application/json'**) và mã thông báo xác thực (**'Authorization'**).

const {

        data: employee,

        refetch,

        isRefetching,

    } = useQuery<any>(['employee', page, filterQuery], () =>

        axios.get(`http://26.75.181.165:8080/employee/allemployees/${filterQuery}`, { headers })

    );

* **data: employee** là dữ liệu nhân viên được trả về từ API.
* **refetch** là một hàm được cung cấp bởi **useQuery** để gửi lại yêu cầu API và cập nhật dữ liệu.
* **isRefetching** là một biến đánh dấu xem yêu cầu API đang được thực hiện hay không.
* Yêu cầu HTTP được gửi bằng phương thức GET đến **http://26.75.181.165:8080/employee/allemployees/${filterQuery}** với các thông tin xác thực trong **headers**.
* **form** được khởi tạo bằng **Form.useForm()** để quản lý dữ liệu và trạng thái của form.
* Biến **loadingClearFilter** là một trạng thái để đánh dấu xem quá trình xóa bộ lọc đang diễn ra hay không.

   const onSearch = (search: string) => {

        setFilterQuery(search);

        setPage(1);

    };

* Hàm **onSearch** được sử dụng để xử lý sự kiện tìm kiếm. Khi người dùng thực hiện tìm kiếm, giá trị **search** được truyền vào hàm và sau đó cập nhật giá trị **filterQuery** (biến lưu trữ các thông tin tìm kiếm) với giá trị **search** mới và đặt lại trang **page** về 1.

const onClearFilter = () => {

        setLoadingClearFilter(true);

        wait(1500).then(() => {

            setFilterQuery('');

            setPage(1);

            setLoadingClearFilter(false);

        });

    };

* Trước khi xóa bộ lọc, biến **loadingClearFilter** được đặt thành **true** để hiển thị trạng thái xóa đang diễn ra.
* Hàm **wait** được gọi để tạm dừng trong 1.5 giây trước khi thực hiện các thay đổi.
* Sau khi tạm dừng, giá trị **filterQuery** được đặt lại thành chuỗi rỗng và trang **page** được đặt lại thành 1.
* Cuối cùng, biến **loadingClearFilter** được đặt lại thành **false** để hiển thị trạng thái xóa đã hoàn thành.

 const handleOk = () => {

        setIsModalOpen(false);

    };

    const formReset = () => {

        form.setFieldsValue(initialValue);

    };

    const handleCancel = () => {

        setValues(null);

        setIsModalOpen(false);

        formReset();

    };

* **handleOk** được gọi khi người dùng xác nhận hoàn tất hành động trong modal. Trạng thái **isModalOpen** được đặt thành **false** để đóng modal.
* **formReset** được gọi để đặt lại giá trị của form thành giá trị ban đầu.
* **handleCancel** được gọi khi người dùng hủy bỏ hành động trong modal. Biến **values** và **isModalOpen** được đặt lại thành giá trị ban đầu và form được đặt lại thành giá trị ban đầu bằng cách gọi **formReset**.

1. Sử dụng hook **useForm** từ thư viện **antd** để quản lý trạng thái và giá trị của form.
2. Sử dụng hook **useEffect** để tái fetch dữ liệu khi **state.syncLoading** thay đổi.
3. Xử lý các sự kiện như tìm kiếm (**onSearch**), xóa bộ lọc (**onClearFilter**), và các sự kiện khác.
4. Hiển thị các thành phần như **Search**, **TableComponent** và **EmployeeFormPage** để hiển thị dữ liệu và thực hiện thao tác thêm mới, chỉnh sửa và xóa nhân viên.
5. Sử dụng **EmployeeFormPage** để hiển thị modal cho việc thêm mới hoặc chỉnh sửa nhân viên.
6. Cuối cùng, hiển thị thành phần **ClearFilter** để cho phép người dùng xóa bộ lọc nếu có.

Đây là trang danh sách nhân viên đơn giản trong ứng dụng, cho phép người dùng tìm kiếm, xem danh sách và thao tác với nhân viên.

**Trang thêm nhân viên**

import React from 'react';

import { Button, Checkbox, Form, Input, InputNumber, Modal, Row, Select, Space, Spin, message } from 'antd';

import styled from 'styled-components';

import { Rule } from 'antd/lib/form';

import { useForm } from 'antd/lib/form/Form';

import axios from 'axios';

import { DataTypeEmployee } from '../components/Employee.Config';

import FormComponent from '@/components/FormComponent';

import SaveButton from '@/components/Button/Save.Button';

import FormItemComponent from '@/components/FormComponent/FormItemComponent';

import LocalStorage from '@/apis/LocalStorage';

const REG\_PHONE = /((^(\+84|84|0|0084){1})(3|5|7|8|9))+([0-9]{8})$/;

const MOBI = /((^(\+84|84|0|0084){1})(3)(2|3|4|5|6|7|8|9))+([0-9]{7})$/;

const VINA = /((^(\+84|84|0|0084){1})(8)(2|3|4|5|6|8|9))+([0-9]{7})$/;

const VIETTEL = /((^(\+84|84|0|0084){1})(9)(3|4|1|6|7|8|9|0))+([0-9]{7})$/;

const SEVEN = /((^(\+84|84|0|0084){1})(7)(0|6|7|8|9))+([0-9]{7})$/;

const FIVE = /((^(\+84|84|0|0084){1})(5)(9))+([0-9]{7})$/;

const Option = Select;

const initialValue = {

    phoneNumber: '',

    password: '',

    username: '',

    address: '',

    createdAt: '',

    name: '',

    salary: '',

};

const token = LocalStorage.getToken();

const headers = {

    'Content-Type': 'application/json',

    Authorization: `Bearer ${token}`,

};

export const formItemLayout = {

    labelCol: {

        xs: { span: 24 },

        sm: { span: 8 },

    },

    wrapperCol: {

        xs: { span: 24 },

        sm: { span: 16 },

    },

};

const EmployeeFormPage = ({

    handleCancel,

    open,

    values,

    handleCallBack,

    refetch,

}: {

    handleLoading?: any;

    handleUnLoading?: any;

    handleCallBack?: any;

    handleCancel?: any;

    handleOk?: any;

    refetch?: any;

    open?: any;

    values?: DataTypeEmployee | any;

}) => {

    const [form] = Form.useForm();

    React.useEffect(() => {

        if (values) {

            form.setFieldsValue({ ...values });

        }

    }, [values]);

    const formReset = () => {

        form.setFieldsValue(initialValue);

    };

    const onSubmit = async (data: DataTypeEmployee) => {

        if (!values) {

            const dataForm = {

                address: data?.address,

                password: data?.password,

                createdAt: data?.createdAt?.trim(),

                phoneNumber: data?.phoneNumber,

                name: data?.name?.trim(),

                status: 'ok',

                salary: data?.salary,

                username: data?.username?.trim(),

            };

            try {

                const res = await axios.put('http://26.75.181.165:8080/admin/newemployee', dataForm, { headers });

                onCancel();

                refetch();

                handleCallBack();

                message.success('Thêm mới nhân viên thành công');

            } catch (error) {

                console.log(error);

            }

        }

    };

    const onCancel = () => {

        handleCancel();

        formReset();

    };

    return (

        <Modal

            title={values ? 'Sửa nhân viên' : 'Thêm nhân viên'}

            open={open}

            onCancel={handleCancel}

            footer={null}

            width="36%"

        >

            <FormComponent

                {...formItemLayout}

                form={form}

                onSubmit={(value: DataTypeEmployee) => {

                    onSubmit(value);

                }}

            >

                <Row gutter={[20, 0]}>

                    <FormItemComponent

                        title="name"

                        label="Họ và tên"

                        inputField={<Input placeholder="Nhập họ và tên" />}

                        rules={[{ required: true, message: 'Vui lòng nhập họ và tên' }]}

                    />

                    <FormItemComponent

                        label="Số điện thoại"

                        title="phoneNumber"

                        rules={[

                            { required: true, message: 'Vui lòng nhập số điện thoại' },

                            {

                                validator(\_: Rule, value: number) {

                                    if (!value?.toString()?.trim()) return Promise.resolve();

                                    if (

                                        (!value?.toString()?.match(SEVEN) &&

                                            !value?.toString()?.match(MOBI) &&

                                            !value?.toString()?.match(VIETTEL) &&

                                            !value?.toString()?.match(VINA) &&

                                            !value?.toString()?.match(FIVE)) ||

                                        isNaN(Number(value))

                                    ) {

                                        return Promise.reject(new Error('Vui lòng nhập đúng định dạng số điện thoại!'));

                                    }

                                    return Promise.resolve();

                                },

                            },

                        ]}

                        inputField={<Input placeholder="Nhập số điện thoại" />}

                    />

                    <FormItemComponent

                        label="Địa chỉ"

                        title="address"

                        rules={[{ required: true, message: 'Vui lòng nhập địa chỉ' }]}

                        inputField={<Input placeholder="Nhập địa chỉ" />}

                    />

                    <FormItemComponent

                        label="Mức lương"

                        title="salary"

                        rules={[{ required: true, message: 'Vui lòng nhập mức lương' }]}

                        inputField={<Input placeholder="Nhập mức lương" />}

                    />

                    <FormItemComponent

                        title="username"

                        label="Tên tài khoản"

                        inputField={<Input placeholder="Nhập tài khoản nhân viên" />}

                        rules={[{ required: true, message: 'Vui lòng nhập tài khoản nhân viên' }]}

                    />

                    <FormItemComponent

                        label="Mật khẩu"

                        title="password"

                        rules={[{ required: true, message: 'Vui lòng nhập mật khẩu' }]}

                        inputField={<Input.Password placeholder="\*\*\*\*\*\*" disabled={values} />}

                    />

                </Row>

                <Row style={{ width: '100%' }} align="bottom" justify="end">

                    <Space>

                        <Button type="default" onClick={onCancel}>

                            Thoát

                        </Button>

                        <SaveButton htmlType="submit" />

                    </Space>

                </Row>

            </FormComponent>

        </Modal>

    );

};

export default EmployeeFormPage;

1. Sử dụng các thành phần từ **antd** như **Form**, **Input**, **Select**, **Button**, **Modal**, **Row**, **Space**, **Spin**, **message** để xây dựng giao diện modal.
2. Sử dụng hook **useForm** từ thư viện **antd** để quản lý trạng thái và giá trị của form.
3. Sử dụng hook **useEffect** để cập nhật giá trị cho form khi **values** thay đổi.
4. Xử lý sự kiện khi người dùng nhấn nút "Thêm nhân viên" hoặc "Sửa nhân viên" trong form.

  React.useEffect(() => {

        if (values) {

            form.setFieldsValue({ ...values });

        }

    }, [values]);

Khi giá trị **values** thay đổi (thường là khi sửa đổi dữ liệu nhân viên), **form.setFieldsValue({ ...values })** được gọi để đặt giá trị của form bằng giá trị mới của **values**.

 const formReset = () => {

        form.setFieldsValue(initialValue);

    };

Khi cần đặt lại form (ví dụ: sau khi hoàn tất thao tác hoặc khi hủy bỏ), **form.setFieldsValue(initialValue)** được gọi để đặt giá trị của form bằng giá trị ban đầu **initialValue**.

 const onSubmit = async (data: DataTypeEmployee) => {

        if (!values) {

            const dataForm = {

                address: data?.address,

                password: data?.password,

                createdAt: data?.createdAt?.trim(),

                phoneNumber: data?.phoneNumber,

                name: data?.name?.trim(),

                status: 'ok',

                salary: data?.salary,

                username: data?.username?.trim(),

            };

            try {

                const res = await axios.put('http://26.75.181.165:8080/admin/newemployee', dataForm, { headers });

                onCancel();

                refetch();

                message.success('Thêm mới nhân viên thành công');

            } catch (error) {

                console.log(error);

            }

        }

    };

Kiểm tra nếu **values** không tồn tại, tức là trường hợp tạo mới nhân viên:, Tạo một đối tượng **dataForm** chứa thông tin nhân viên từ các trường dữ liệu trong form:

* Dữ liệu nhân viên được lấy từ các trường dữ liệu trong form, ví dụ: **data?.address**, **data?.password**, **data?.createdAt?.trim()**, v.v.
* Các giá trị được chuẩn hóa bằng cách sử dụng **.trim()** để xóa bỏ các khoảng trắng thừa ở đầu và cuối chuỗi.

Yêu cầu PUT được gửi đến **http://26.75.181.165:8080/admin/newemployee** với dữ liệu **dataForm** và thông tin xác thực trong **headers**.

Kết quả của yêu cầu được lưu trong biến **res**

Trong trường hợp gửi yêu cầu thành công, các bước sau được thực hiện:

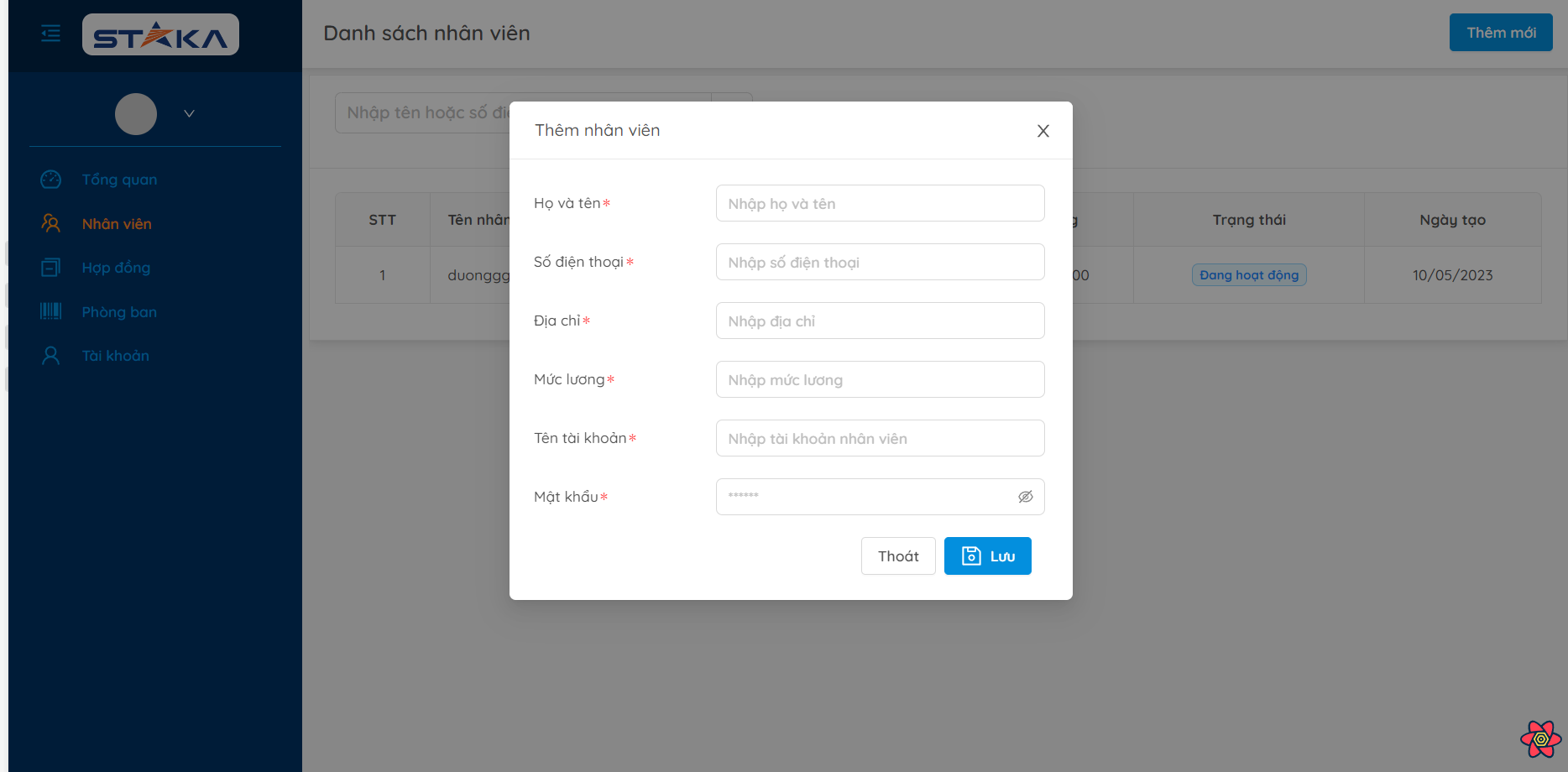
Gọi hàm **onCancel** để đóng modal và đặt lại giá trị của form:

Gọi hàm **refetch** để gửi lại yêu cầu API và cập nhật dữ liệu nhân viên:

Hiển thị thông báo thành công sử dụng **message.success**:

Trong trường hợp gửi yêu cầu gặp lỗi, lỗi được ghi vào console để kiểm tra và xử lý sau.

1. Xử lý sự kiện khi người dùng nhấn nút "Hủy" trong form để đóng modal và đặt lại giá trị ban đầu cho form.
2. Hiển thị giao diện modal với các trường thông tin nhân viên như Họ và tên, Số điện thoại, Địa chỉ, Mức lương, Tên tài khoản và Mật khẩu.
3. Sử dụng các thành phần như **Input**, **Input.Password**, **Select** để người dùng có thể nhập thông tin.
4. Sử dụng **FormItemComponent** để tạo các trường thông tin trong form với các thuộc tính như **title**, **label**, **inputField** và **rules**.
5. Xử lý sự kiện khi người dùng nhấn nút "Thoát" hoặc "Lưu" trong form để đóng modal hoặc gửi yêu cầu cập nhật thông tin.
6. Hiển thị nút "Lưu" bằng thành phần **SaveButton** để người dùng có thể lưu thay đổi thông tin.



Chi tiết nhân viên

1. Sử dụng các thành phần từ **react**, **react-query**, **react-router-dom**, **moment**, **@ant-design/icons**, **antd**, **@fullcalendar/react**, **@fullcalendar/daygrid** để xây dựng giao diện trang chi tiết.
2. Sử dụng hook **useParams** từ **react-router-dom** để lấy giá trị tham số từ URL.
3. Sử dụng hook **useState** để quản lý trạng thái của bộ lọc (**filterQuery**) với giá trị ban đầu là **initialFilterQuery**.
4. Sử dụng hook **useNavigate** từ **react-router-dom** để điều hướng trang.
5. Sử dụng hook **useQuery** từ **react-query** để gửi yêu cầu lấy dữ liệu nhân viên và dữ liệu cuộc hẹn.
6. Sử dụng thành phần **TopBar** để hiển thị tiêu đề trang và nút "Xóa nhân viên".
7. Sử dụng thành phần **Popconfirm** từ **antd** để xác nhận việc xóa nhân viên.
8. Sử dụng **axios.get** để gửi yêu cầu lấy thông tin nhân viên và cuộc hẹn từ API.

-const {

        data: employee,

        refetch,

        isRefetching,

    } = useQuery<any>(['employee', filterQuery], () =>

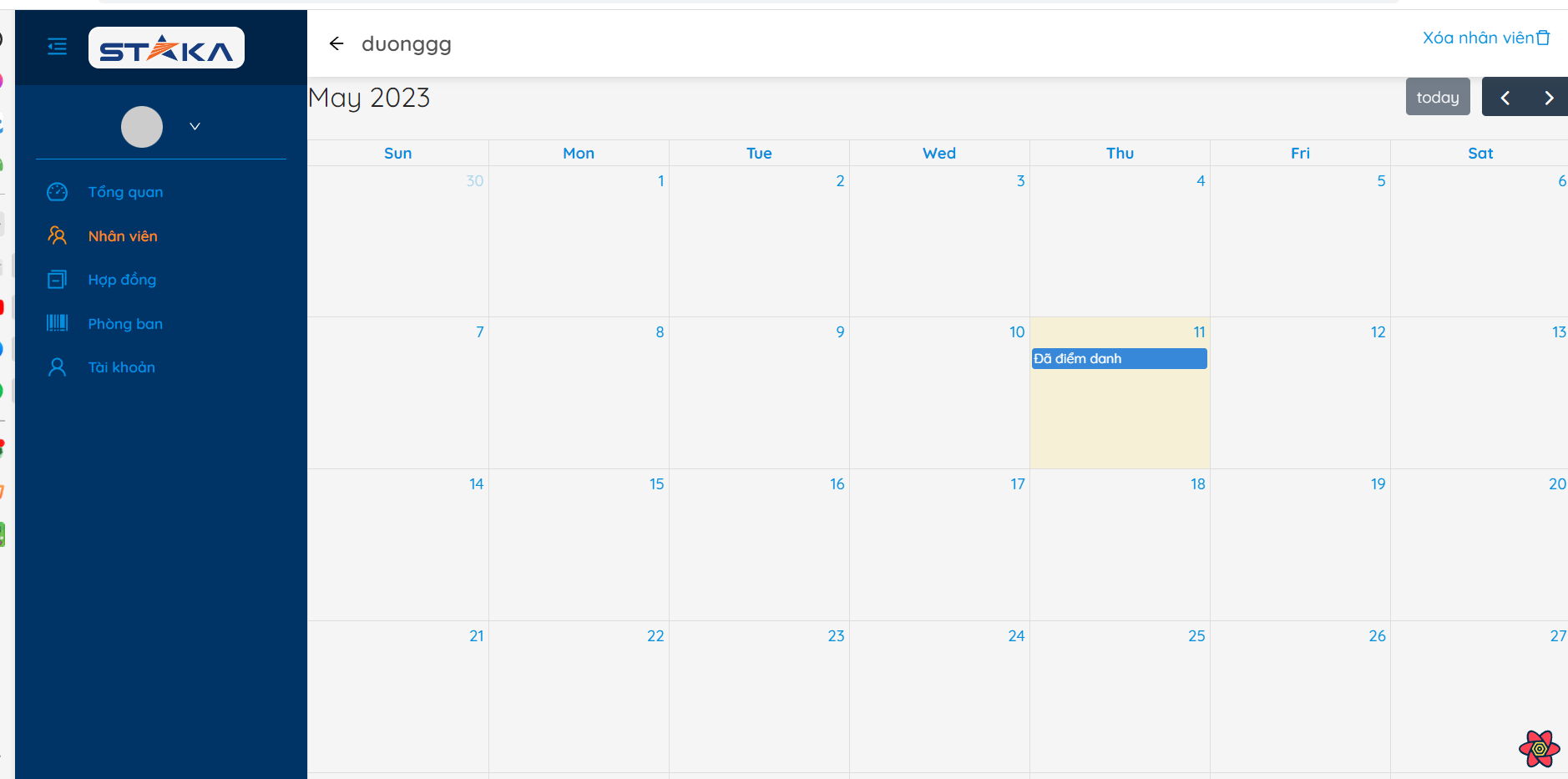
        axios.get(`http://26.75.181.165:8080/employee/${id}`, { ...filterQuery, headers })

    );

* **data: employee** là dữ liệu nhân viên được trả về từ API.
* **refetch** là một hàm được cung cấp bởi **useQuery** để gửi lại yêu cầu API và cập nhật dữ liệu.
* **isRefetching** là một biến đánh dấu xem yêu cầu API đang được thực hiện hay không.
* const { data: rolling } = useQuery<any>(['rolling', id], () =>
* axios.get(`http://26.75.181.165:8080/admin/getrollingupinf/${id}/${5}/${'2023'}`, { headers })
* );
* **data: rolling** là dữ liệu cuộc hẹn đã điểm danh được trả về từ API.
* const confirm = async (record: any) => {
* const res = await axios.delete(`http://26.75.181.165:8080/employee/${record}`, { headers });
* message.success('Đã xóa nhân viên thành công');
* refetch();
* navigate('/employee');
* };
* Hàm này gửi một yêu cầu HTTP DELETE đến API để xóa nhân viên.
* Sau khi xóa thành công, hàm gọi **refetch()** để cập nhật dữ liệu nhân viên và chuyển hướng đến trang danh sách nhân viên (**'/employee'**).

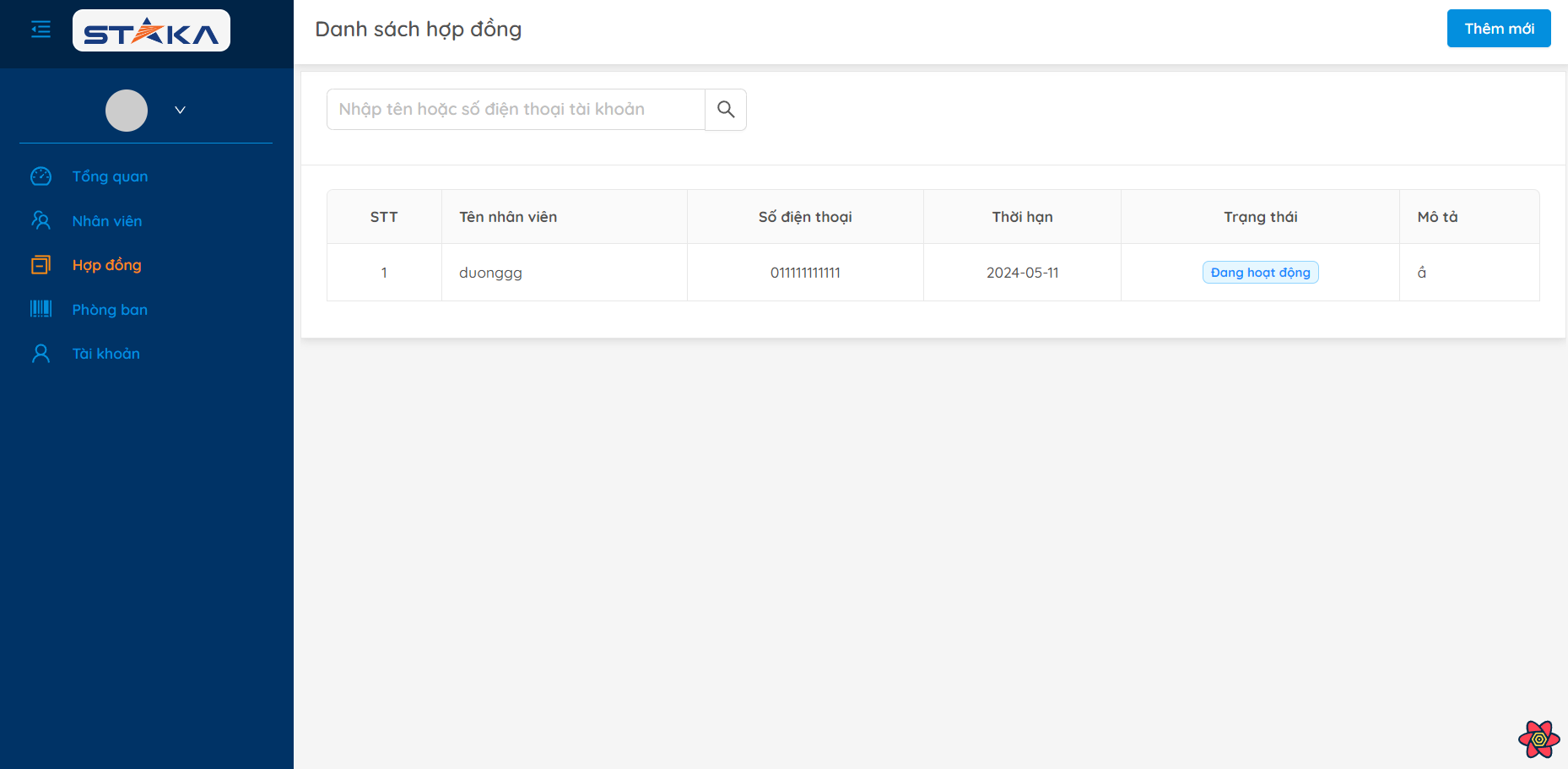
1. Sử dụng thành phần **FullCalendar** từ **@fullcalendar/react** để hiển thị lịch và sự kiện đã điểm danh.
2. Xử lý sự kiện khi người dùng xác nhận xóa nhân viên và gửi yêu cầu xóa nhân viên đến API.
3. Hiển thị tiêu đề trang và nút "Xóa nhân viên" bằng thành phần **TopBar**.
4. Hiển thị lịch và sự kiện đã điểm danh bằng thành phần **FullCalendar**.

Trang chi tiết nhân viên hiển thị thông tin của nhân viên và lịch cuộc hẹn đã điểm danh. Người dùng có thể xóa nhân viên thông qua nút "Xóa nhân viên".

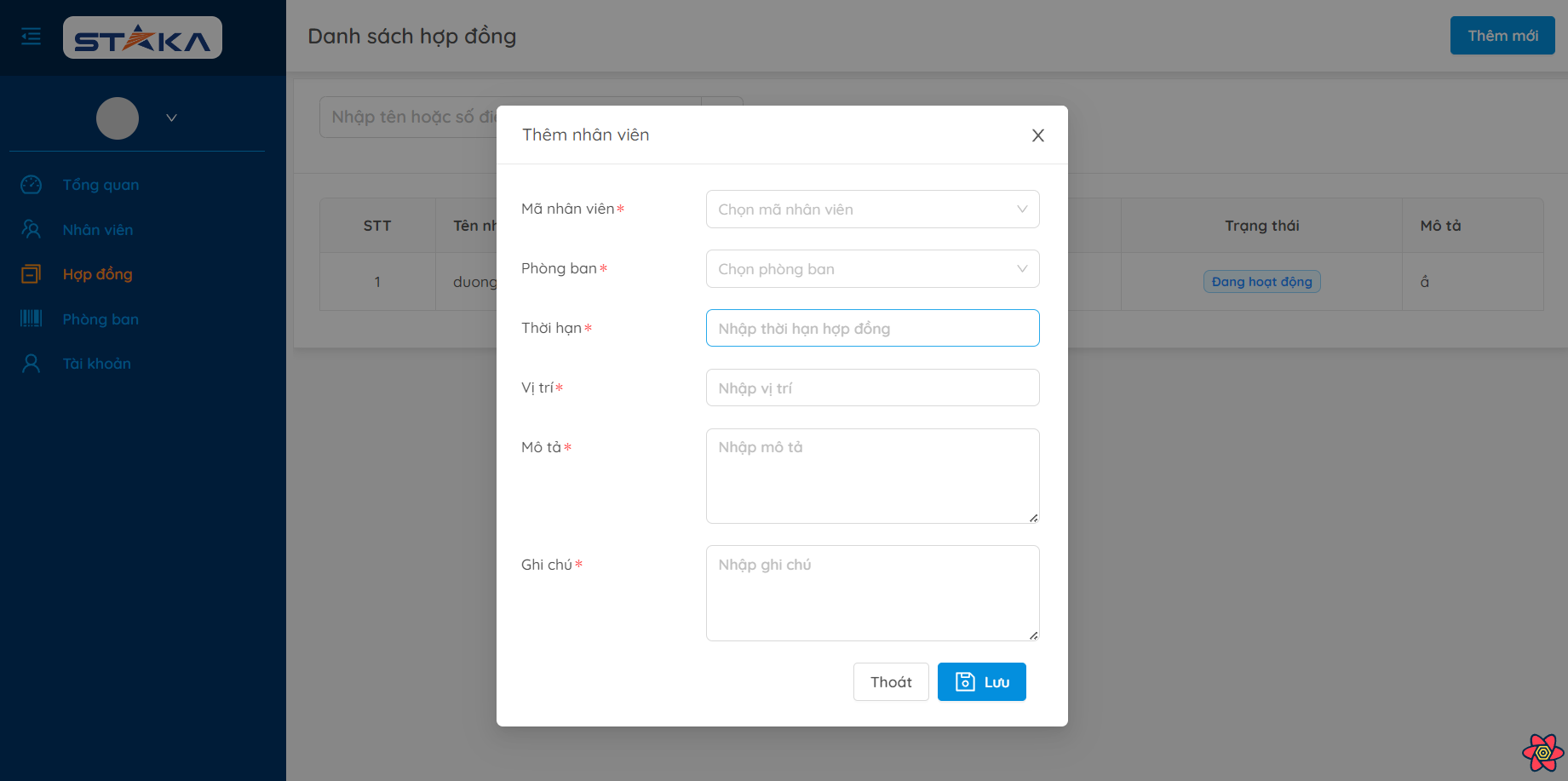


Giao diện trang hợp đồng ( có thể xem danh sách hợp đồng và thêm mới 1 hợp đồng , tìm kiếm tên nhân viên của hợp đồng đó)

Phần code và logic tương tự trang nhân viên chỉ thay đổi dữ liệu

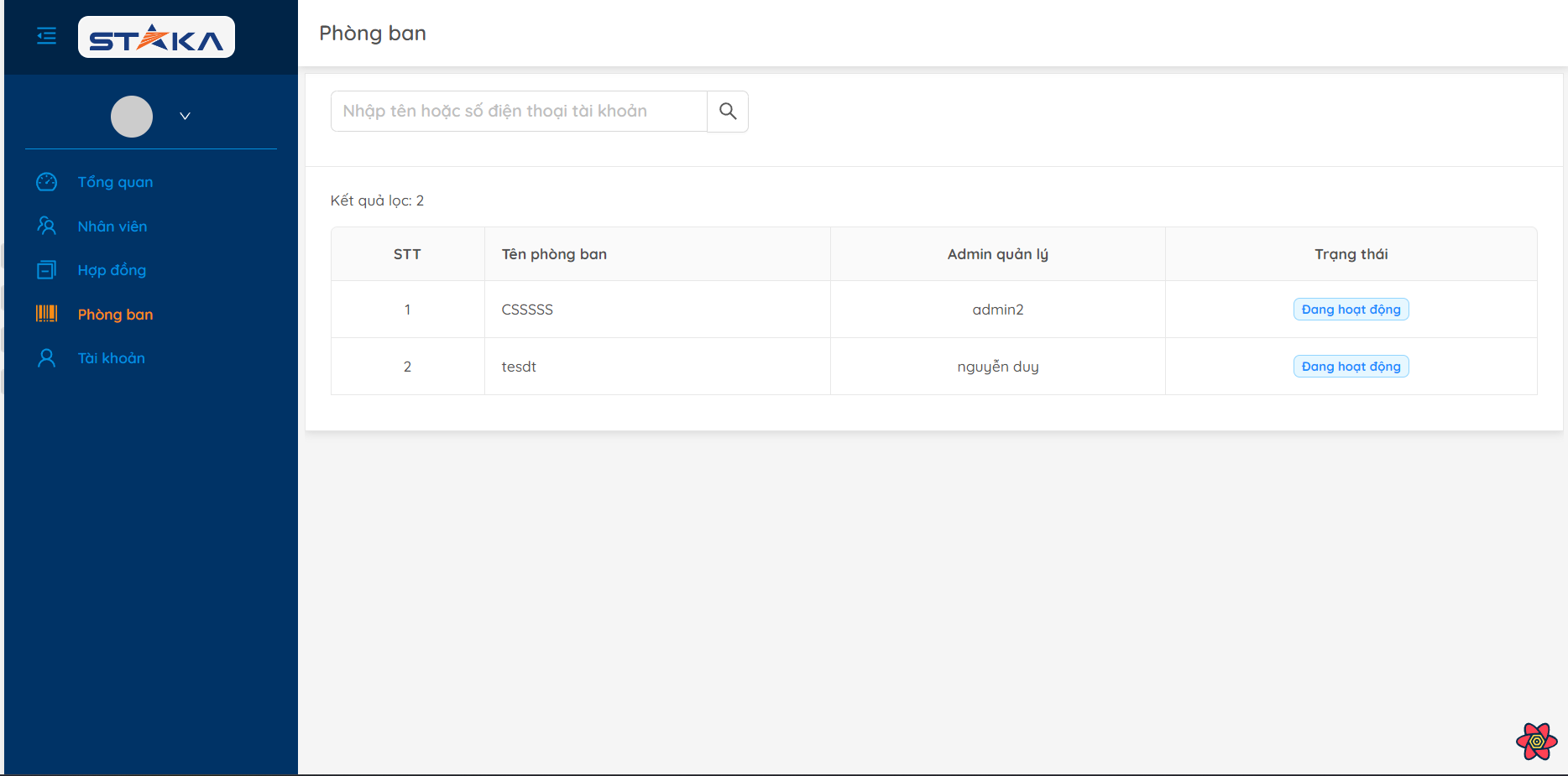


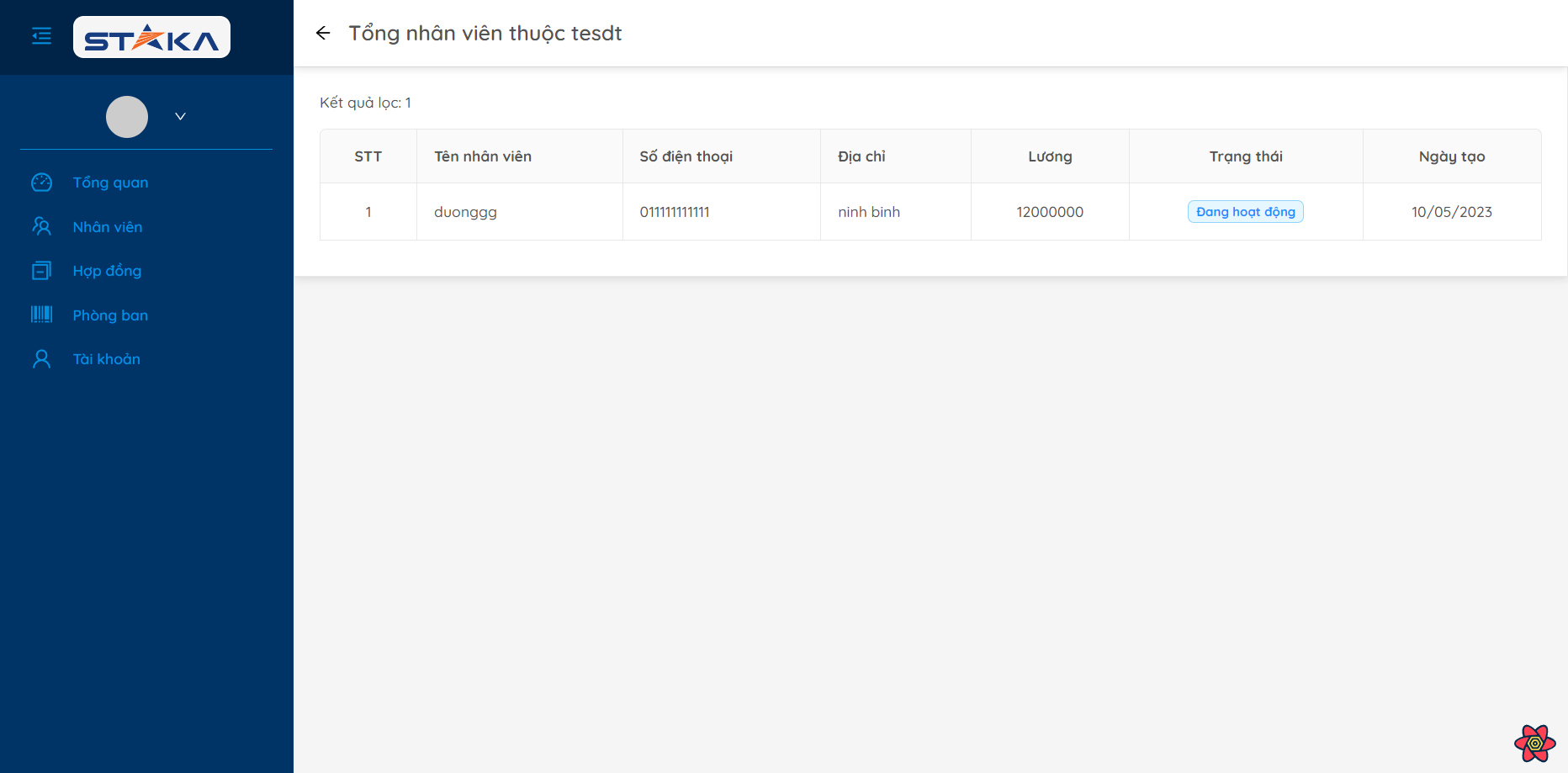
Form thêm mới 1hợp đồng.



Giao diện trang phòng ban (có thể xem danh sách phòng ban, click vào 1 dòng ở danh sách có thể xem tổng số nhân viên của phòng ban đó, tìm kiếm theo tên phòng ban)

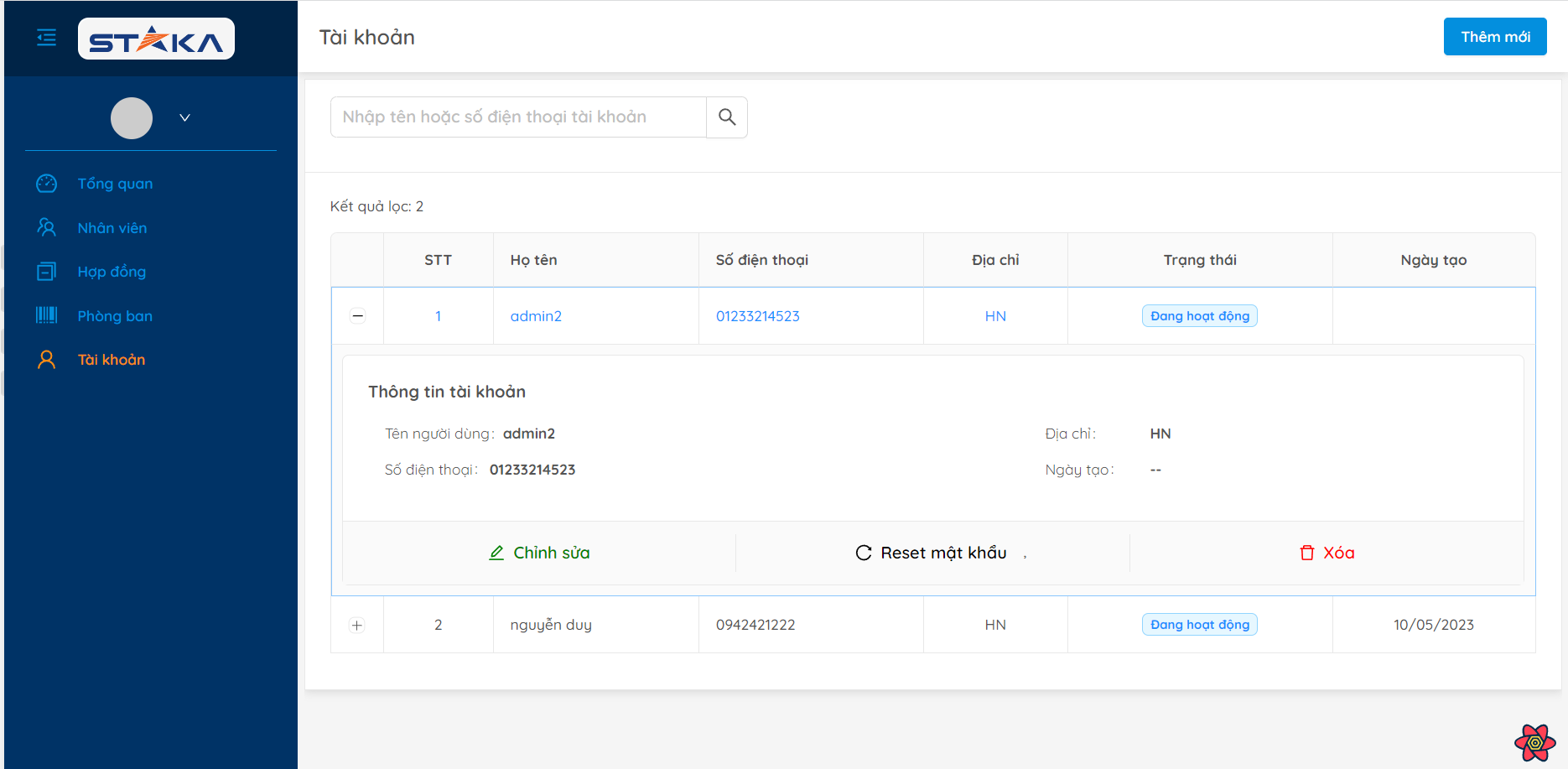
Phần code và logic cũng tương tự trang nhân viên



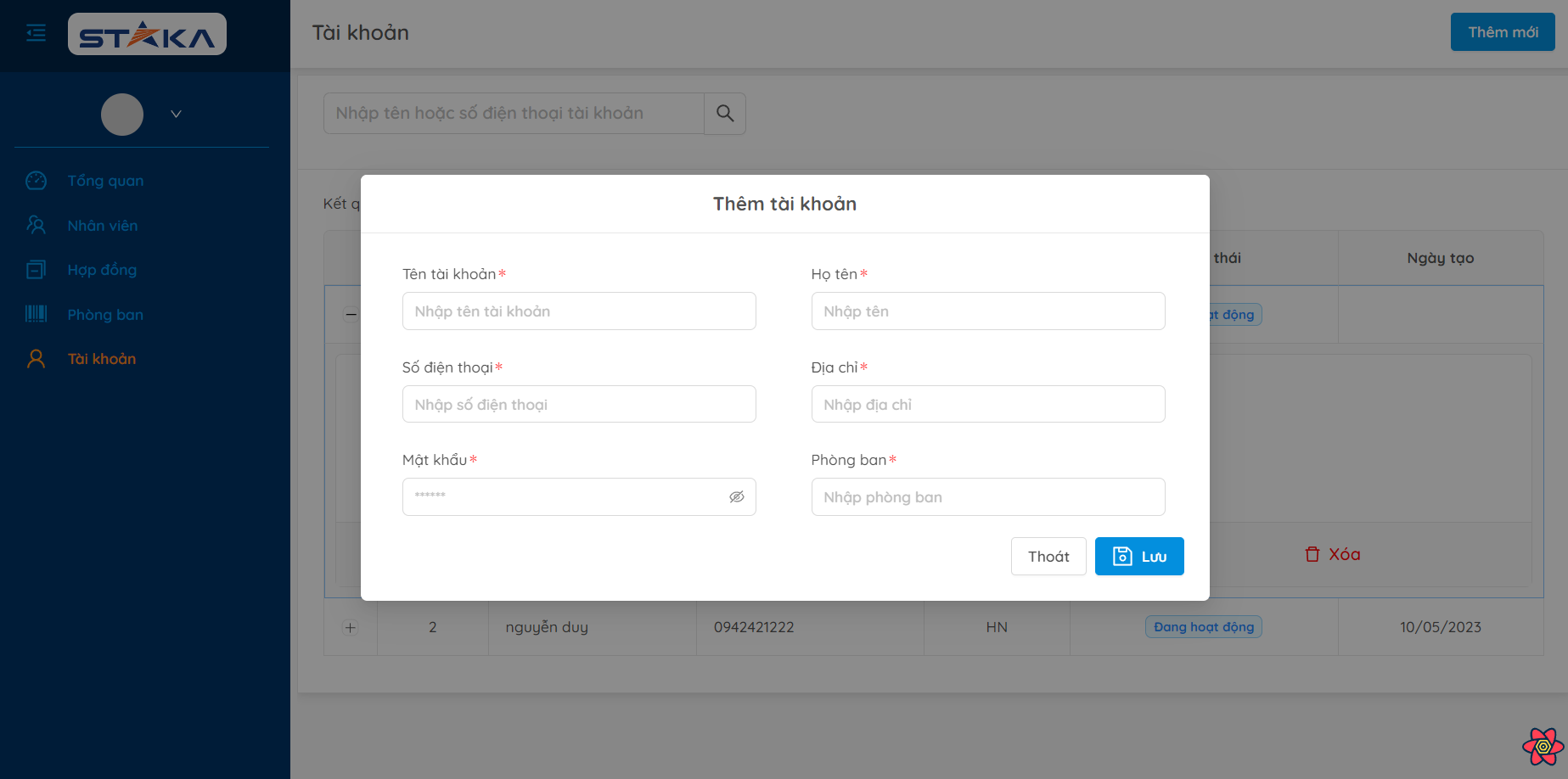


Giao diện trang tài khoản(có thể thêm mới 1 tài khoản, chỉnh sửa, xóa và đặt lại mật khẩu mặc định là ‘123’, tìm kiếm theo tên tài khoản)

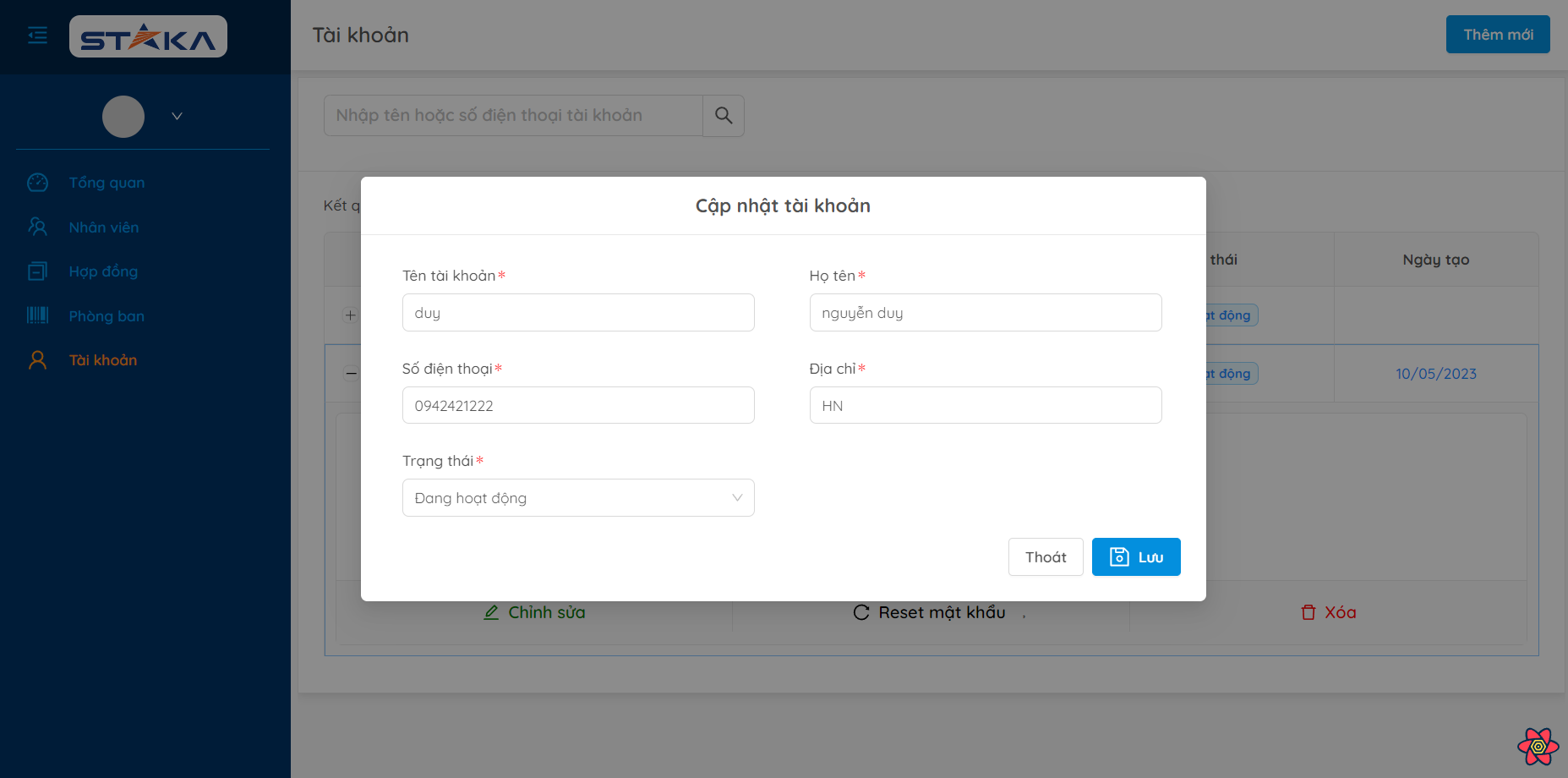
Phần code và logic tương tự trang nhân viên



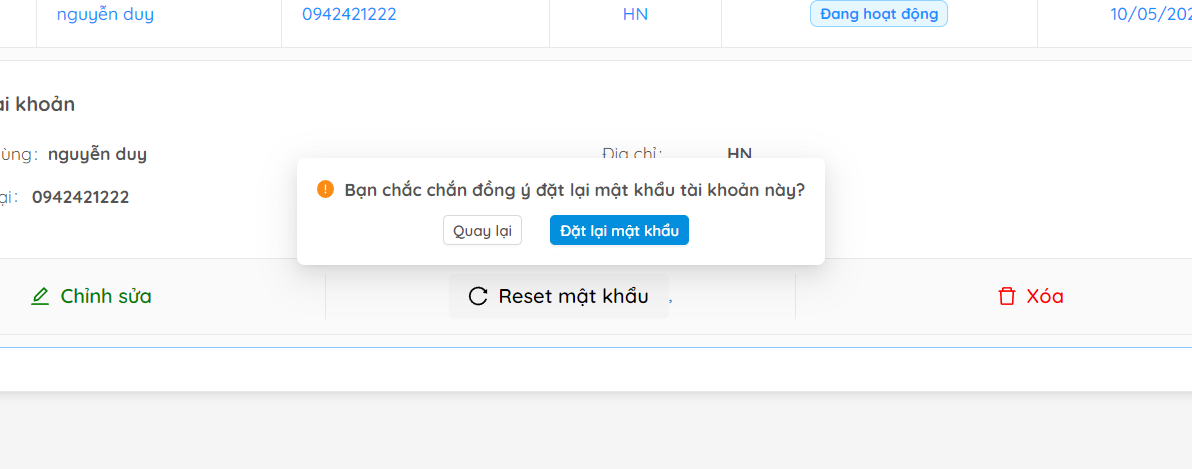
Form thêm mới tài khoản



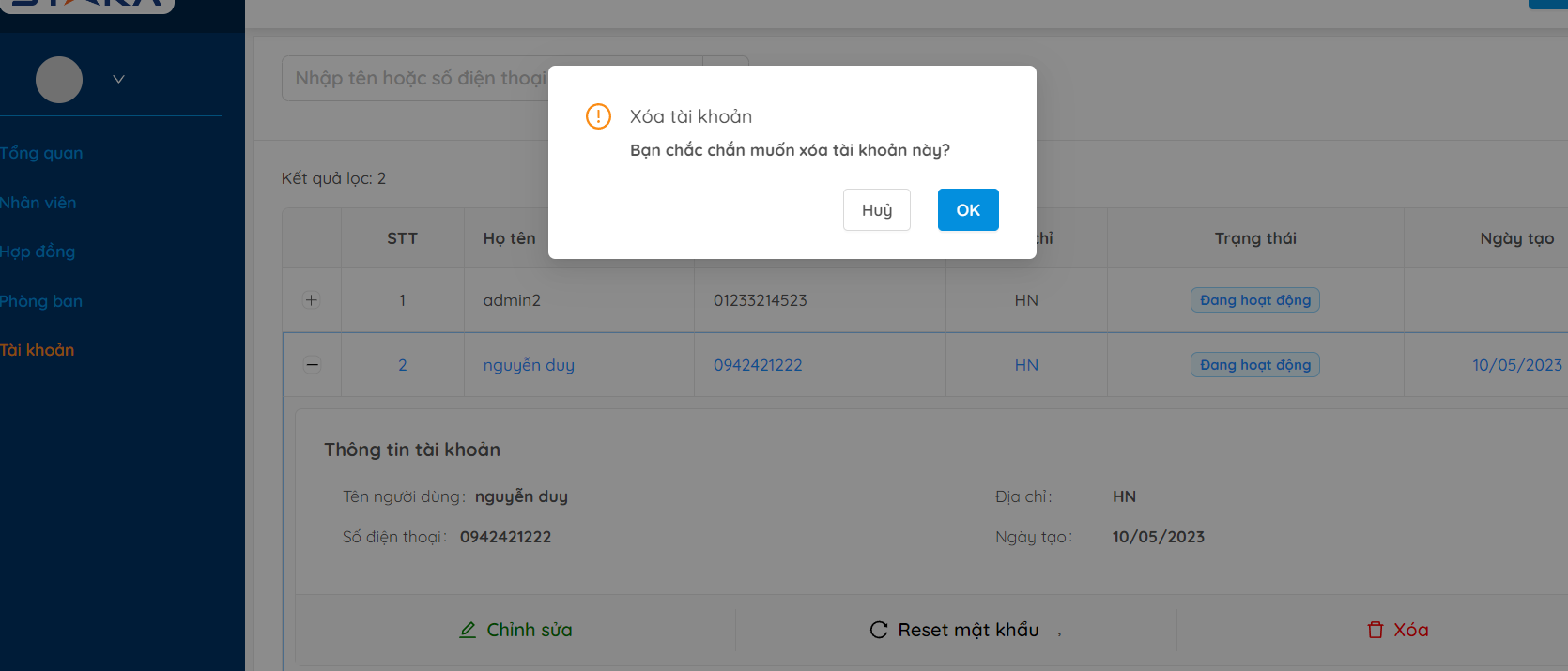
Form chỉnh sửa tài khoản



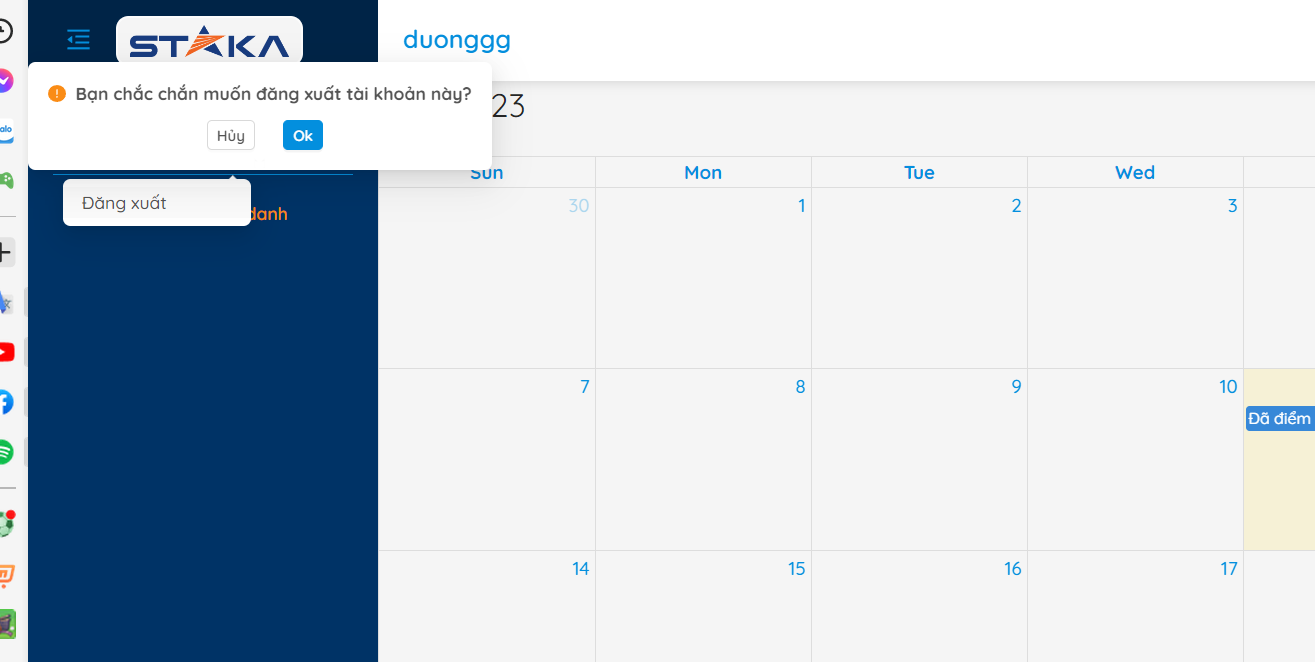
Khi ấn reset mật khẩu



Khi ấn nút xóa

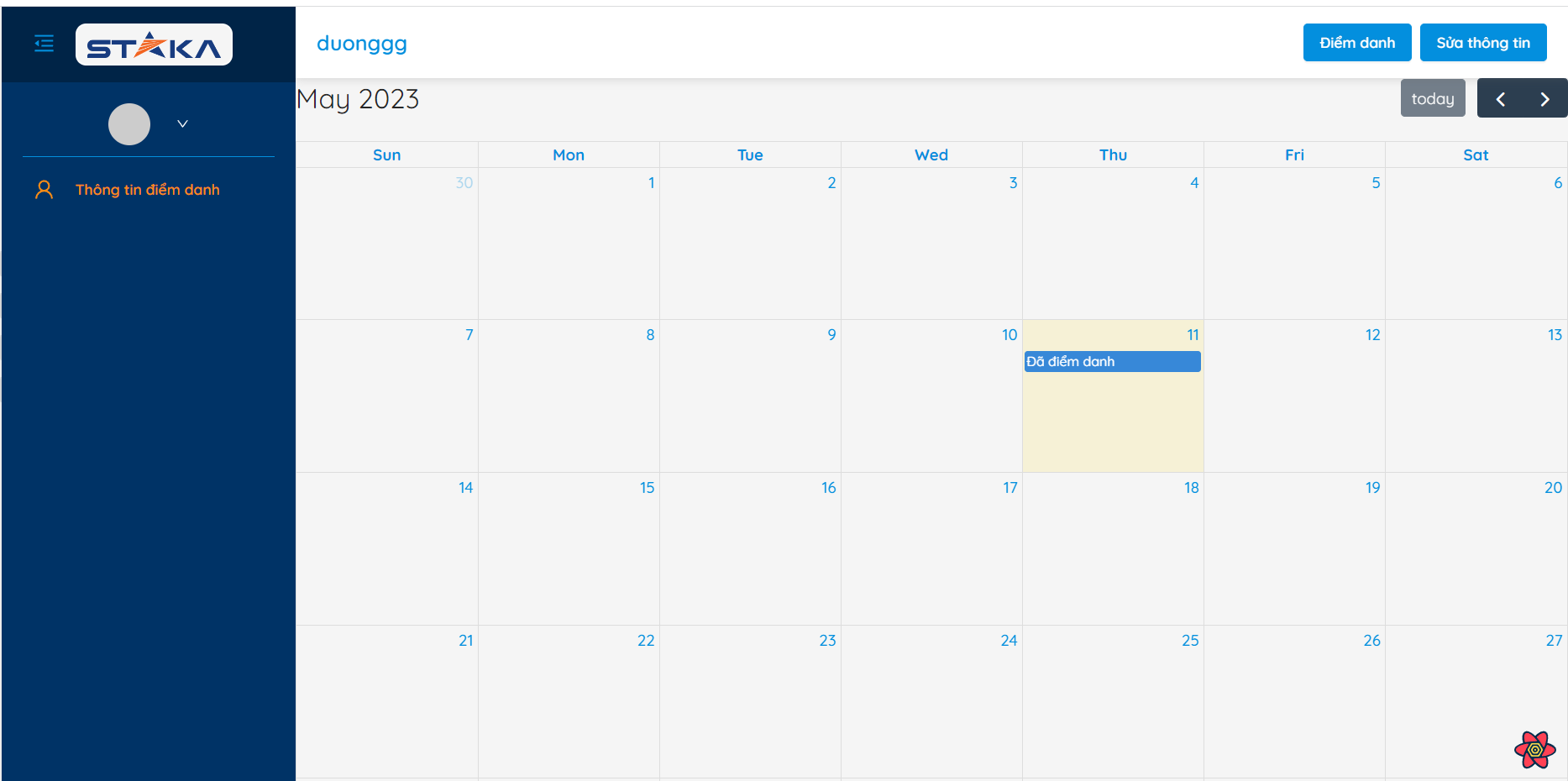


Khi muốn đăng xuất

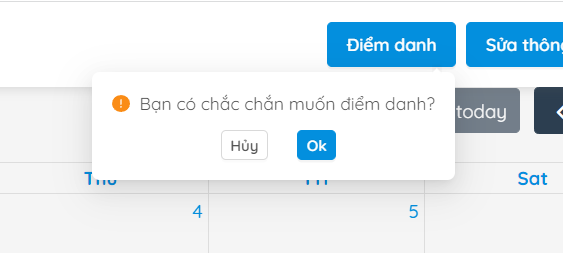


Khi dăng nhập vào nick nhân viên

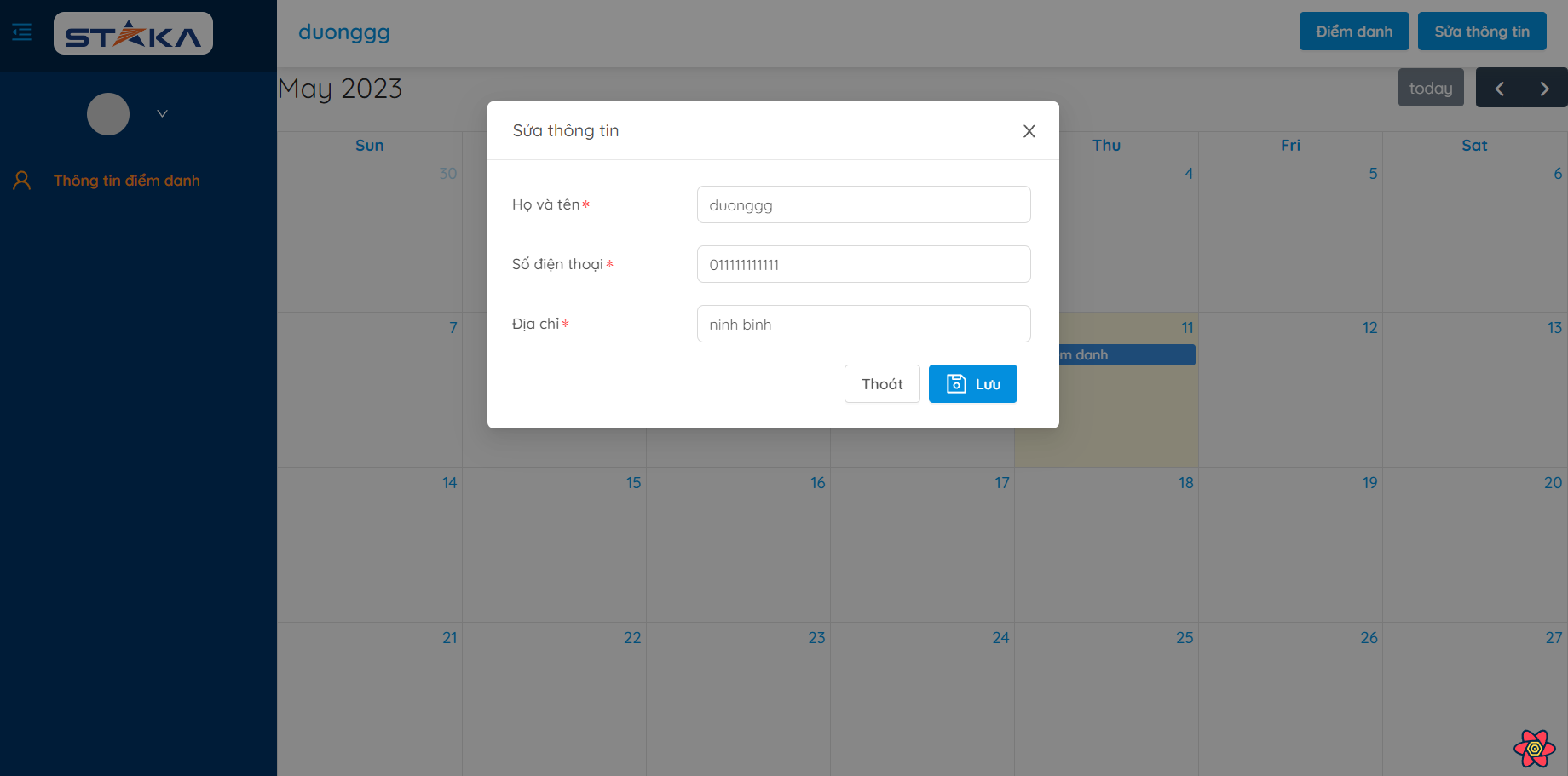
Giao diện(có thể sửa thông tin cá nhân và điểm danh)



Khi click điểm danh



Khi click buttton sửa thông tin(hiện ra các thông tin cũ có thể sửa)



Code:

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

import { ClearOutlined } from '@ant-design/icons';

import { Button, Form, Input, Modal, Popconfirm, Row, Space, message } from 'antd';

import { useQuery } from 'react-query';

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

import moment from 'moment';

import FullCalendar from '@fullcalendar/react';

import dayGridPlugin from '@fullcalendar/daygrid';

import TopBar from '@/components/TopBar';

import Topbar from '@/layout/Content/Topbar';

import axios from 'axios';

import LocalStorage from '@/apis/LocalStorage';

import Buttons from '../../voucher/components/Buttons';

import FormItemComponent from '@/components/FormComponent/FormItemComponent';

import FormComponent from '@/components/FormComponent';

import SaveButton from '@/components/Button/Save.Button';

const initialFilterQuery = {

    search: '',

    status: '',

    customerName: '',

    typeService: '',

    provinceId: '',

    districtId: '',

    form: '',

    vote: '',

    startDate: '',

    endDate: '',

};

export const formItemLayout = {

    labelCol: {

        xs: { span: 24 },

        sm: { span: 8 },

    },

    wrapperCol: {

        xs: { span: 24 },

        sm: { span: 16 },

    },

};

const CalendarPage: React.FC<any> = () => {

    const [form] = Form.useForm();

    const [isModalOpen, setIsModalOpen] = useState(false);

    const token = LocalStorage.getToken();

    const headers = {

        'Content-Type': 'application/json',

        Authorization: `Bearer ${token}`,

    };

    const {

        data: Infor,

        refetch,

        isRefetching,

    } = useQuery<any>(['employee'], () =>

        axios.get(`http://26.75.181.165:8080/employee/getrollingupinf/${5}/${'2023'}`, { headers })

    );

    React.useEffect(() => {

        if (Infor) {

            form.setFieldsValue({

                name: Infor?.data?.name,

                address: Infor?.data?.address,

                phoneNumber: Infor?.data?.phonenumber,

            });

        }

    }, [Infor]);

    const formReset = () => {

        form.setFieldsValue({

            name: Infor?.data?.name,

            address: Infor?.data?.address,

            phoneNumber: Infor?.data?.phonenumber,

        });

    };

    const handleCancel = () => {

        setIsModalOpen(false);

        formReset();

    };

    useEffect(() => {

        refetch();

    }, []);

    const requestData = {

        name: '',

    };

    const onSubmit = async (data: any) => {

        const dataForm = {

            address: data?.address,

            phoneNumber: data?.phoneNumber,

            name: data?.name?.trim(),

        };

        try {

            const res = await axios.post(`http://26.75.181.165:8080/employee/newemployee`, dataForm, {

                headers,

            });

            handleCancel();

            refetch();

            message.success('Cập nhật thông tin thành công');

        } catch (error) {

            console.error(error);

        }

    };

    const confirm = async (e: any) => {

        try {

            const res = await axios.post('http://26.75.181.165:8080/employee/rollingup', requestData, { headers });

            // handleCallBack();

            if (res?.data?.status === 'ok') message.success('Đã điểm danh thành công');

            else message.error('Hôm nay bạn đã điểm danh rồi');

            refetch();

        } catch (error) {

            console.log(error);

        } finally {

        }

    };

    const events = Infor?.data?.alldates?.map((item: any, index: number) => ({

        title: 'Đã điểm danh',

        id: item.id,

        date: item.date,

    }));

    const cancel = (e: any) => {};

    return (

        <>

            <TopBar

                title={<a onClick={() => setIsModalOpen(true)}>{Infor?.data?.name}</a>}

                extra={[

                    <Popconfirm

                        placement="topRight"

                        title="Bạn có chắc chắn muốn điểm danh?"

                        onConfirm={() => {

                            confirm('1');

                        }}

                        onCancel={cancel}

                        okText="Ok"

                        cancelText="Hủy"

                    >

                        <a

                            style={{

                                fontSize: '1.6rem',

                            }}

                            href=""

                        >

                            <Button type="primary">Điểm danh</Button>

                        </a>

                    </Popconfirm>,

                    <Button onClick={() => setIsModalOpen(true)} type="primary">

                        Sửa thông tin

                    </Button>,

                ]}

            />

            <FullCalendar plugins={[dayGridPlugin]} initialView="dayGridMonth" events={events} />

            <Modal title="Sửa thông tin" open={isModalOpen} onCancel={handleCancel} footer={null} width="36%">

                <FormComponent

                    {...formItemLayout}

                    form={form}

                    onSubmit={(value: any) => {

                        onSubmit(value);

                    }}

                >

                    <Row gutter={[20, 0]}>

                        <FormItemComponent

                            title="name"

                            label="Họ và tên"

                            inputField={<Input placeholder="Nhập họ và tên" />}

                            rules={[{ required: true, message: 'Vui lòng nhập họ và tên' }]}

                        />

                        <FormItemComponent

                            label="Số điện thoại"

                            title="phoneNumber"

                            rules={[{ required: true, message: 'Vui lòng nhập số điện thoại' }]}

                            inputField={<Input placeholder="Nhập số điện thoại" />}

                        />

                        <FormItemComponent

                            label="Địa chỉ"

                            title="address"

                            rules={[{ required: true, message: 'Vui lòng nhập địa chỉ' }]}

                            inputField={<Input placeholder="Nhập địa chỉ" />}

                        />

                    </Row>

                    <Row style={{ width: '100%' }} align="bottom" justify="end">

                        <Space>

                            <Button type="default" onClick={handleCancel}>

                                Thoát

                            </Button>

                            <SaveButton htmlType="submit" />

                        </Space>

                    </Row>

                </FormComponent>

            </Modal>

        </>

    );

};

export default CalendarPage;

Giải thích:

1. Import các thành phần cần thiết từ thư viện **antd**, **@fullcalendar/react**, **axios**, và **LocalStorage**.
2. Khai báo và khởi tạo các biến và state cần sử dụng:
   * **form**: Dùng để quản lý form sử dụng trong modal.
   * **isModalOpen**: State để xác định trạng thái mở/đóng của modal.
   * **token**: Lấy token từ **LocalStorage**.
   * **headers**: Định nghĩa header cho các yêu cầu HTTP sử dụng token.
   * **Infor**: State lưu trữ thông tin cá nhân của nhân viên được lấy từ API.
   * **refetch**: Hàm được cung cấp bởi **useQuery** để gửi lại yêu cầu API và cập nhật dữ liệu.
   * **isRefetching**: Trạng thái đang gửi lại yêu cầu API.
   * Sử dụng **useEffect** để cập nhật giá trị của form khi thông tin cá nhân thay đổi:

React.useEffect(() => {

        if (Infor) {

            form.setFieldsValue({

                name: Infor?.data?.name,

                address: Infor?.data?.address,

                phoneNumber: Infor?.data?.phonenumber,

            });

        }

    }, [Infor]);

1. Định nghĩa hàm **formReset** để đặt lại giá trị của form:

  const formReset = () => {

        form.setFieldsValue({

            name: Infor?.data?.name,

            address: Infor?.data?.address,

            phoneNumber: Infor?.data?.phonenumber,

        });

    };

1. Định nghĩa hàm **handleCancel** để đóng modal và đặt lại giá trị của form:

 const handleCancel = () => {

        setIsModalOpen(false);

        formReset();

    };

1. Sử dụng **useEffect** để gửi yêu cầu API và lấy dữ liệu thông tin cá nhân khi trang được tải:

  useEffect(() => {

        refetch();

    }, []);

1. Định nghĩa đối tượng **requestData** chứa dữ liệu yêu cầu khi điểm danh:

const requestData = {

        name: '',

    };

1. Định nghĩa hàm **onSubmit** để gửi yêu cầu API để cập nhật thông tin cá nhân:

  const onSubmit = async (data: any) => {

        const dataForm = {

            address: data?.address,

            phoneNumber: data?.phoneNumber,

            name: data?.name?.trim(),

        };

        try {

            const res = await axios.post(`http://26.75.181.165:8080/employee/newemployee`, dataForm, {

                headers,

            });

            handleCancel();

            refetch();

            message.success('Cập nhật thông tin thành công');

        } catch (error) {

            console.error(error);

        }

    };

1. Định nghĩa hàm `confirm` để gửi yêu cầu API để điểm danh:

const confirm = async (e: any) => {

        try {

            const res = await axios.post('http://26.75.181.165:8080/employee/rollingup', requestData, { headers });

            // handleCallBack();

            if (res?.data?.status === 'ok') message.success('Đã điểm danh thành công');

            else message.error('Hôm nay bạn đã điểm danh rồi');

            refetch();

        } catch (error) {

            console.log(error);

        } finally {

        }

    };

1. Sử dụng dữ liệu từ **Infor** để tạo danh sách các sự kiện trên lịch:

const events = Infor?.data?.alldates?.map((item: any, index: number) => ({

        title: 'Đã điểm danh',

        id: item.id,

        date: item.date,

    }));

1. Định nghĩa các sự kiện và giao diện trang:
   * Hiển thị tiêu đề và nút điểm danh, chỉnh sửa thông tin.
   * Sử dụng **FullCalendar** để hiển thị lịch làm việc.
   * Sử dụng **Modal** để hiển thị form chỉnh sửa thông tin cá nhân.
   * Gọi **FormComponent** để tạo form chỉnh sửa thông tin cá nhân.
   * Gọi **Popconfirm** để xác nhận khi điểm danh.
   * Sử dụng **message** từ thư viện **antd** để hiển thị thông báo thành công hoặc lỗi.
2. Cuối cùng, xuất **CalendarPage** là một component React.

**Phần 2: Cơ sở dữ liệu + Back End:**

* + Cơ sở dữ liệu:

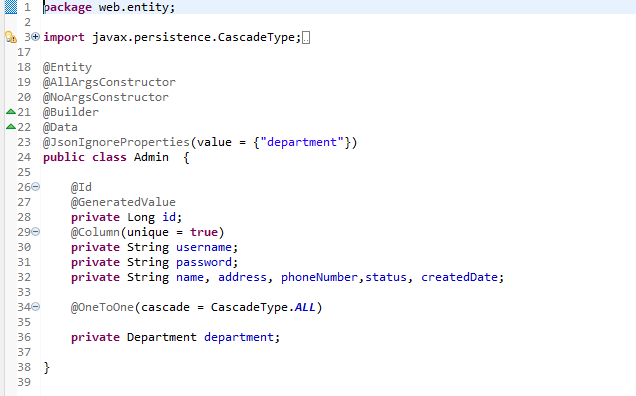
A picture containing text, screenshot, diagram, font

Description automatically generated

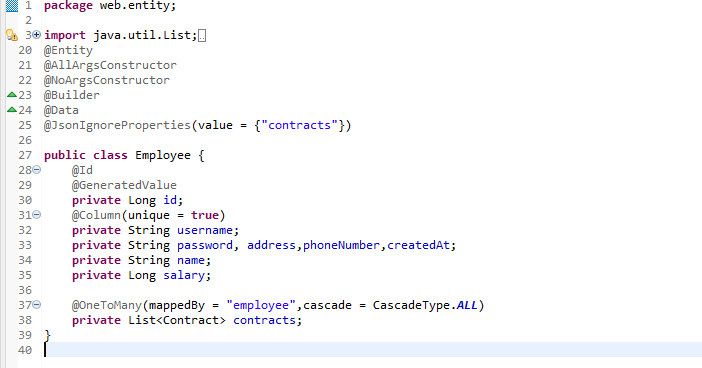
* + Back End

1. Pakage Entity

Class Admin:



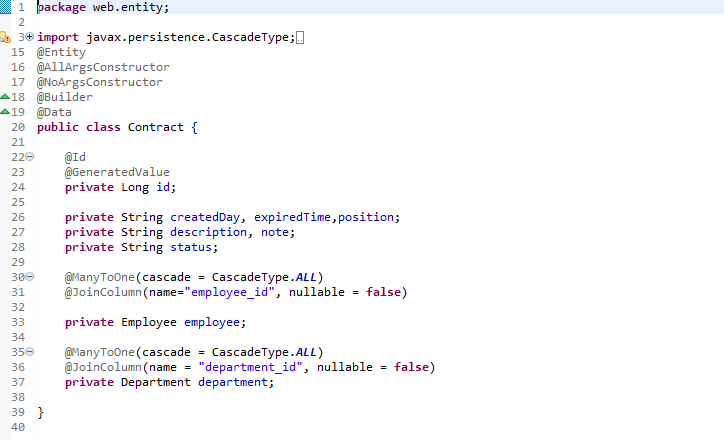
Class Employee:



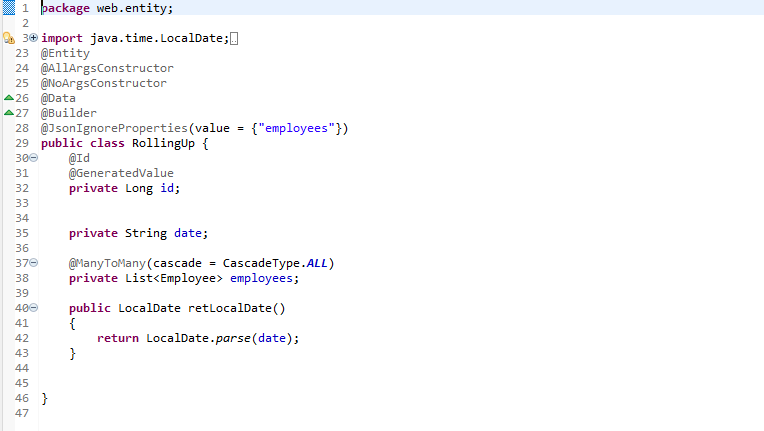
Class Department:



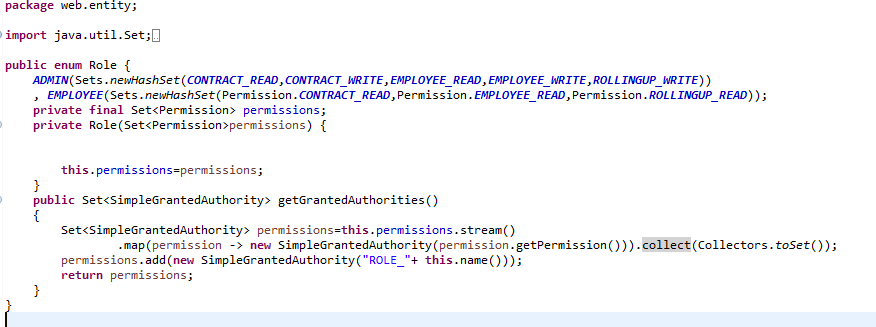
Class Contract:



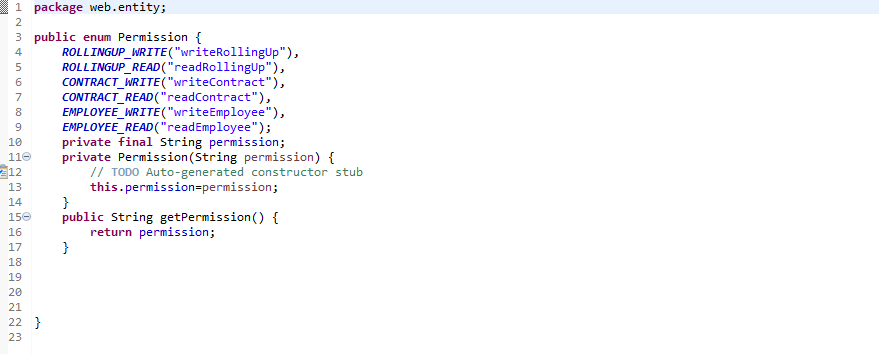
Class RollingUp:



Class Role:

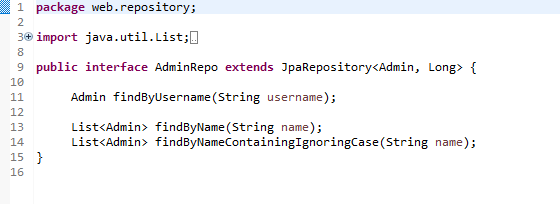


Class Permission:

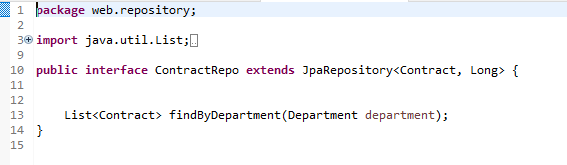


2. Package Repository

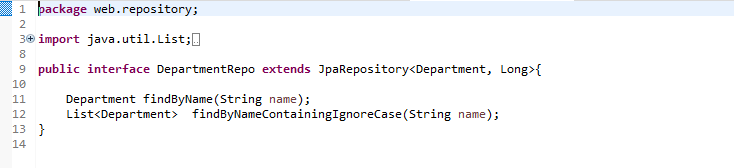
Class AdminRepo:



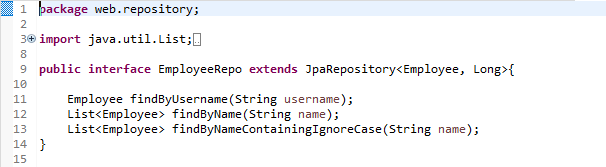
Class ContractRepo:



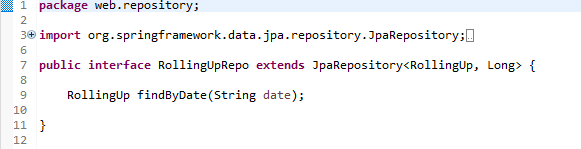
Class DepartmentRepo:



Class EmployeeRepo:

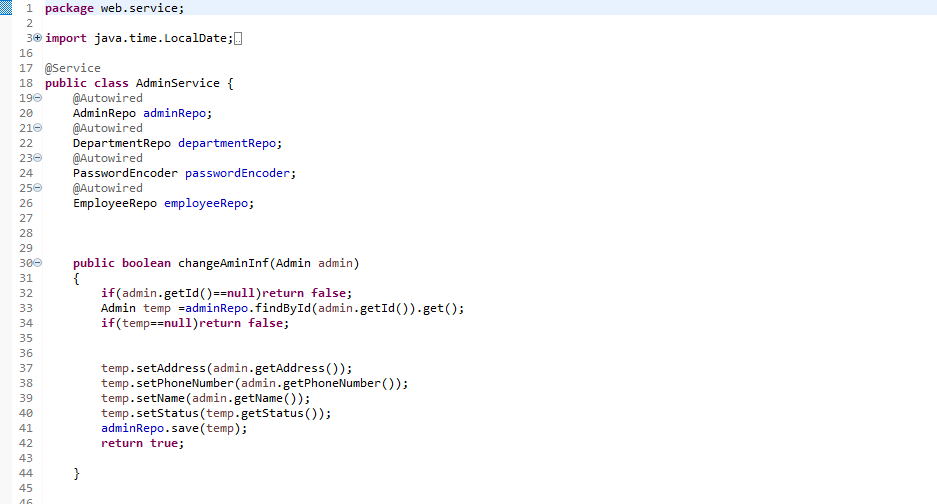


Class RollingUpRepo:



3.Package Service:

Class AdminService:

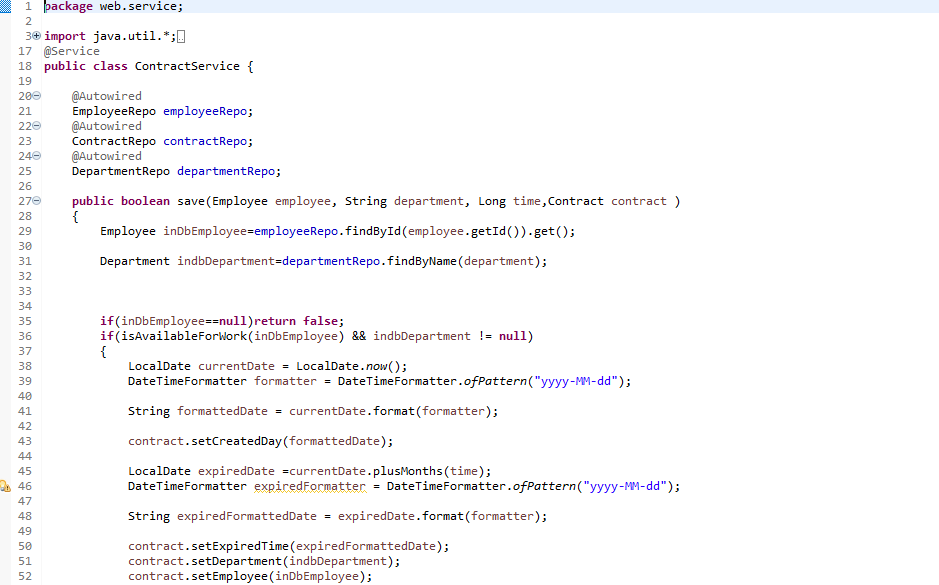






* Hàm changeAdminInf(Admin admin): Cho phép sửa thông tin của 1 admin
* Hàm save(Admin admin, String department): Cho phép lưu 1 admin và 1 department mới vào csdl
* Hàm adminResetPassword(Long id): cho phép reset mật khẩu của 1 admin về lại thành 123
* Hàm employeeResetPassword(Long id): cho phép reset mật khẩu của 1 employee về lại thành 123

Class ContractService:







* Hàm save(Employee employee, String department, Long time,Contract contract ) cho phép tạo 1 hợp đồng mới.
* Hàm isAvailableForWork(Employee employee): cho phép kiểm tra nhân viên có thể tạo hợp đồng mới hay không(hợp đồng gần đây nhất vẫn chưa hết hạn thì không thể nhận hd mới)
* Hàm getAllEmployeesBelongingToDepartment(String departmentName): trả về 1 danh sách những nhân viên hiện đang làm việc cho ban phòng nào.
* Hàm allContractsBelongingToEmployeeNameIgnoreCase(String name): trả về 1 danh sách hợp đồng theo tên được truyền vào

Class EmployeeService:

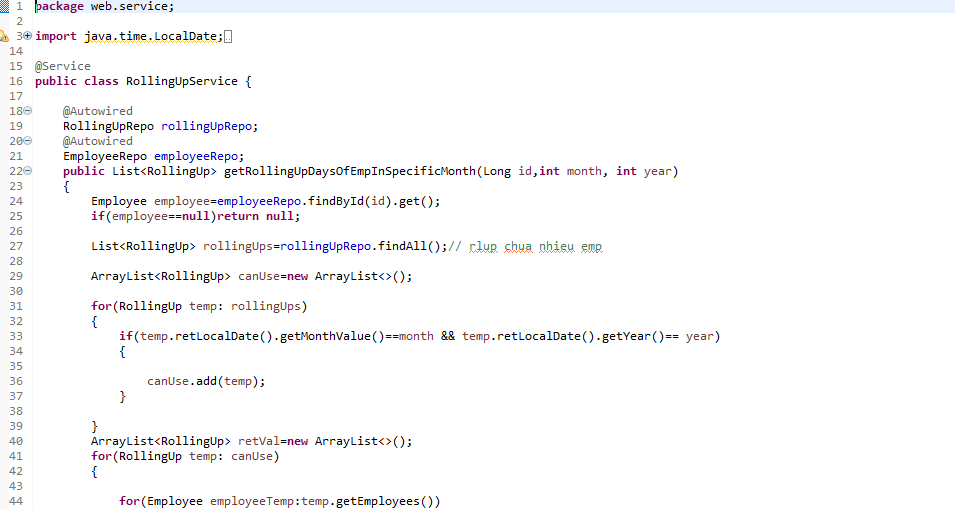






* Hàm employeesBelongToDepartment(Department department) : trả về 1 danh sách nhân viên đã và đang ở trong 1 ban phòng
* Hàm changeEmpInfo(Employee employee): cho phép thay đổi thông tin 1 nhân viên
* Hàm save(Employee employee): cho phép lưu 1 nhân viên mới vào csdl

Class RollingUpService





* Hàm getRollingUpDaysOfEmpInSpecificMonth(Long id,int month, int year): lấy ra những ngày mà 1 nhân viên điểm danh trong tháng và năm xác định

4.Package jwt

Class JwtUsernameAndPasswordAuthenticationFilter:





* Hàm attemptAuthentication(HttpServletRequest request, HttpServletResponse response): xác thực đăng nhập trong thân request user gửi đến.
* Hàm successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain, Authentication authResult): sau khi xác thực thành công, tạo token, gắn token vào header của response.

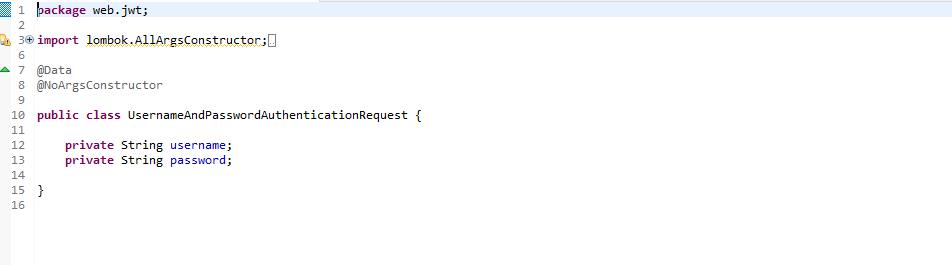
Class JwtTokenVerifier:





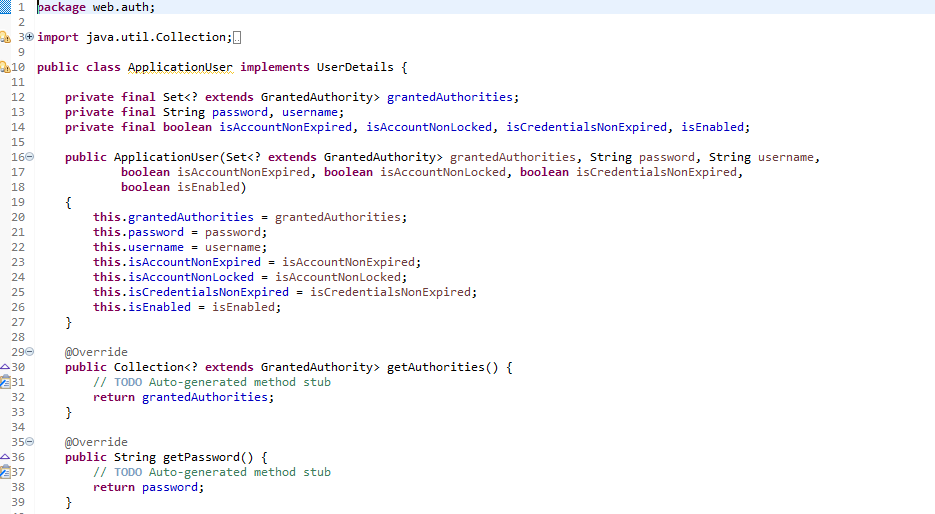
* Hàm doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain): Yêu cầu mỗi request gửi kèm token, xác thực token để lấy phân quyền.

Class UsernameAndPasswordAuthenticationRequest:



- 1 class để hỗ trợ việc tạo token.

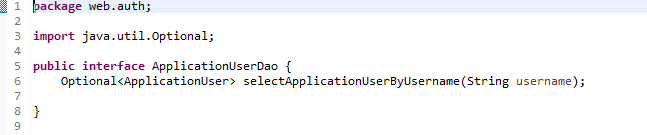
5. Package Auth





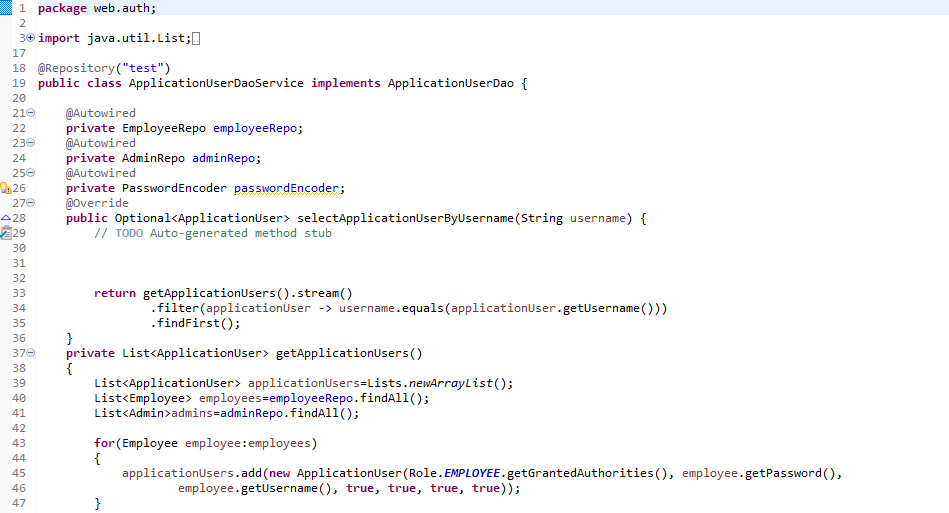
- 1 class extends class UserDetails để chứa thông tin những user trong database để xác thực đăng nhập.

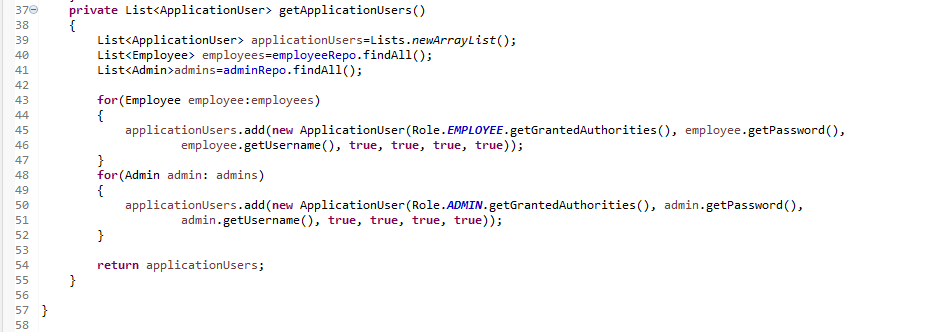
Interface ApplicationUserDao:



- 1 class để hỗ trợ Dependency Injection

Class ApplicationUserDaoService:

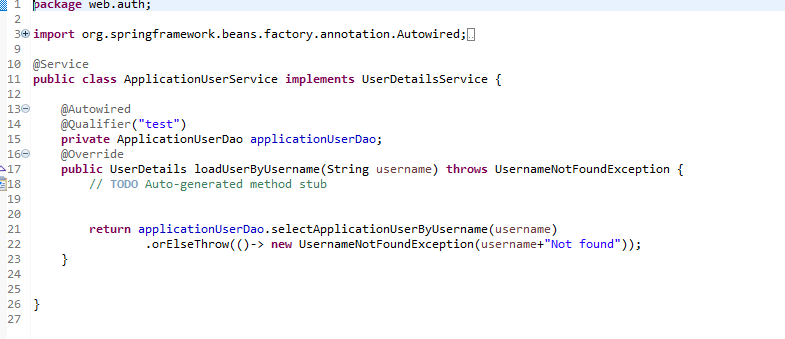




- Hàm selectApplicationUserByUsername(String username): Lấy ra user dựa trên username

- getApplicationUsers(): Lấy ra 1 list user trong database, những user trong bảng admin thì sẽ có quyền ROLE\_ADMIN, còn trong bảng employee thì có quyền ROLE\_EMPLOYEE.

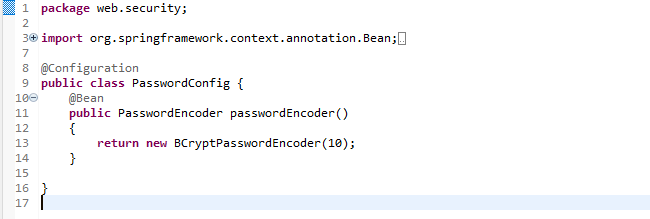
Class ApplicationUserService:



- 1 class implements interface UserDetailsService để hỗ trợ cấu hình security

6. Package Security

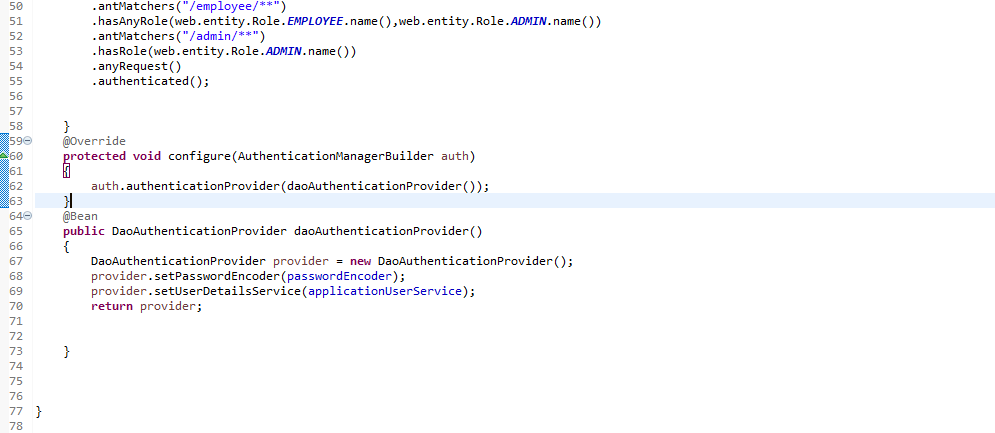
Class PasswordConfig:



* Sử dụng Bcypt làm passwordEncoder

Class ApplicationSecurityConfig:





* Hàm configure(HttpSecurity http): Nâng CORS lên tầm security, disable cross site request forgery, thêm filter để xác thực token, cho phép user với quyền ROLE\_ADMIN vào những endpoints có bắt đầu bằng /admin/\*\*, user có quyền ROLE\_EMPLOYEE vào những endpoints có bắt đầu bằng /employee/\*\*.
* Hàm configure (AuthenticationManagerBuilder auth): chọn xác thực do mình tự custom.
* Hàm DaoAuthenticationProvider daoAuthenticationProvider(): đánh dấu là @Bean để spring container khởi tạo nó.

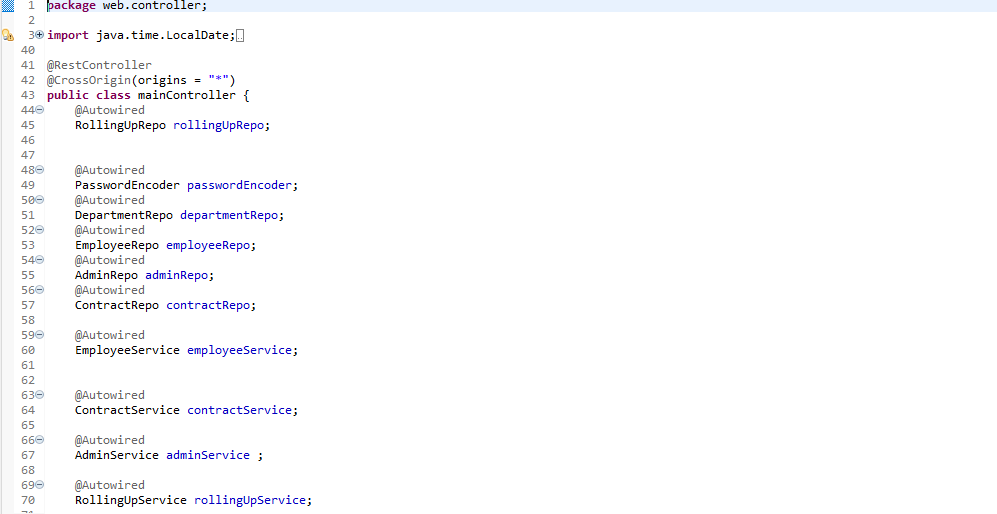
Class AppConfig:



* Class để config CORS

7. Package Controller

Class mainController:





























**Các endpoints:**

/getrole : get -> dùng để lấy role trả về 1 json

/employee/allemployees: get -> 1 mảng toàn bộ nhân viên

/getid: get-> lấy ra id của admin or nv

/countall: get -> trả về mọi thông số

/employee/newemployee: post-> sửa thông tin 1 employee, trong thân Phải có đủ mọi thuộc tính của employee hiện tại.

/employee/allemployees/{name}: get -> trả về 1 list nhân viên theo tên không phải username nhé

/employee/{id}: get -> bỏ dấu ngoặc đi, thay id bằng 1 số nhé, dùng để lấy nhân viên theo id, trả về 1 json

/employee/{id}: delete ->bỏ dấu ngoặc đi, thay id bằng 1 số nhé, dùng xóa nhân viên, không trả về gì

/employee/rollingup: post -> đăng nhập acc của nhân viên vào, vào link chưa điểm danh thì status là ok, ngược lại là rolledUp

/employee/employeesbelongingdepartment/{departmentname}: get -> lấy tất cả nhân viên thuộc 1 department, cái departmentname ấy sẽ là 1 cái String, nó tương ứng với 1 trong mấy cái tên department trong database

/employee/getrollingupinf/{month}/{year}: get-> lấy thông tin điểm danh khi log acc employee

/employee/allemployees/department/{department}: get -> lấy employees thuộc 1 department

/admin/newemployee: put -> thêm mới 1 nhân viên trong thân phải có đủ trường thuộc tính của nhân viên, status là ok thì mới thêm đc nhé

/admin/alladmins: get -> 1 mảng toàn bộ admin

/admin/{id}: get -> bỏ dấu ngoặc đi, thay id bằng 1 số nhé, lấy admin theo id, trả vê 1 json

/admin/{id}: delete -> bỏ dấu ngoặc đi, thay id bằng 1 số nhé, xóa admin theo id, k trả về gì

/admin/newadmin/{departmentname}: put -> thêm mới admin, trong thân phải có đủ trường thuộc tính của admin, status phải là ok nhé, cái departmentname ấy sẽ là 1 cái String, departmentname này không được trùng với bất cứ cái department nào trong database(vì mỗi admin sẽ thầu 1 cái department nghĩa là thêm mới admin là có thêm 1 department mới)

/admin/newcontract/{departmentname}/{time}/{idemp}: put -> thêm 1 contract, trong thân k cần cái trường createdDay và expiredTime(backend tạo r), cái departmentname ấy sẽ là 1 cái String, nó tương ứng với 1 trong mấy cái tên department trong database, time là 1 số nguyên là số tháng kí hợp đồng, idemp là mã nhân viên đã tồn tại trong database

/admin/resetpass/employee/{id}: put -> đặt lại pass cho nhân viên về lại 123 theo id

/admin/resetpass/admin/{id}: put-> đặt lại pass cho admin về lại 123 theo id

/admin/alladmins/{name}: get-> trả về 1 list admin theo tên không phải username nhé

/admin/newadmin/{departmentname}: post -> sửa thông tin 1 admin, trong thân Phải có đủ mọi thuộc tính của admin hiện tại Nhất là id nhé, departmentname đặt bừa đi vì ông muốn cái api này giống vói thêm admin

/admin/getrollingupinf/{id}/{month}/{year}: get-> trả về 1 mảng chuỗi dạng

/department/alldepartments/{name}: get-> lấy thông tin department bằng 1 phần tên

/department/alldepartments: get -> trả về tất cả các department trong db

/contract/allcontracts/{name}: get -> lấy contracts chỉ bằng 1 phần tên nhân viên

/contract/allcontracts : get -> trả về tất cả contract trong db