À mình quên ko bảo bạn chạy lên và test xem chức năng ok chưa.

Bạn ấn vào chỗ tìm kiếm của vs code để tìm kiếm đoạn code mình note nhé.

Bạn Muốn hiểu đc code cần hiểu đc 3 ý sau :

1: mô hình hoạt động website như nào

Khi mà mình truy cập vào website:

Ui:(Browser) => sẽ gửi yêu cầu lên server

Server sẽ tiếp nhận yêu cầu và trả về phản hồi

Server và ui sẽ giao tiếp với nhau qua api: <http://localhost:3000/list-employee> đây cx đc coi 1 api

(tất cả api nằm trong thưc mục petrolimex\petrolimex-ui\src\services\apis.js)

Dựa vào các api và ở tab netword sẽ biết đc các yêu cầu nào gửi lên server

2: restfull api: có 4 phương thức thường đc dùng để làm việc với api

-get : lấy dữ liệu=> vd: danh sách nhân viên, danh sách chấm công,..)

-put(sửa dữ liệu) => vd: chức năng sửa nhân viên

-delete(xóa dữ liệu) => vd: chức năng xóa nhân viên

-post : (tạo dữ liệu) => vd: tạo nhân viên

=> tùy thuộc vào các api mà nó sẽ sử dụng các phương thức tương ứng

=> cái này trong api của project có thể hiện rõ ở chức năng danh sách nhân viên có chức năng **thêm sử xóa**

3: Mô hình MVC => thường mở dụng thêm thành MVCS (project này đang dùng)

**Model (M - Model):**

**Chức năng:** Đại diện cho dữ liệu và logic liên quan đến dữ liệu. => tương tác với db(MySql)

**View (V - View): (UI)**

**Chức năng:** Hiển thị thông tin cho người dùng và tương tác với họ.

**Controller (C - Controller):**

**Chức năng:** Quản lý luồng điều khiển của ứng dụng. tiếp nhận req, phản hôi res . nếu cần sẽ chọc đến **Service**

**Service (S - Service):**

**Chức năng:** Nếu cần, Service sẽ chọc đến Model.

Bạn cố hiểu 3 ý sau có thể tham khảo trên mạng miễn sau hiểu đc 3 ý trên

Giải thích code

Project sử dụng các thư viện và frameword sau:

Db: mysql(xampp)

Server: nodejs (experss, sequelize,jsonwebtoken,mysql2,bcryptjs)

Ui: Reactjs(Redux,React router dom ,react-toastify, react-hook-form, js-cookie, material,axios)

* **Đây là code bên server**

**Chức năng đăng nhập**

Đầu tiên bạn cứ đọc từ file web: **project\petrolimex\petrolimex-server\src\route\web.js**

Đến file controller : **project\petrolimex\petrolimex-server\src\controllers\useController.js**

Đến file server: **project\petrolimex\petrolimex-server\src\services\useServices.j**

**File web:**

 router.post("/api-login", handleLoginUser);

Đăng ký một api (/api-login) cho phương thức POST và liên kết nó với hàm handleLoginUser. Điều này có nghĩa là khi có một yêu cầu POST đến đường dẫn /api-login, hàm handleLoginUser sẽ được gọi để xử lý yêu cầu. rồi nó sẽ gọi đến controller:

**File controller:**

const handleLoginUser = async (**req**, **res**) => {

    try {

        const userInfor = req.body;

        const data = await handleServiceLoginUser(userInfor);

        if (data.accessToken && data.refreshToken) {

            res.cookie("accessToken", data.accessToken, {

*// httpOnly: true,*

*// secure: true,*

            });

            res.cookie("refreshToken", data.refreshToken, {

*// httpOnly: true,*

*// secure: true,*

            });

        }

        delete data.accessToken;

        delete data.refreshToken;

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

    } catch (error) {

        console.log(error);

        return res.status(200).json(error.message);

    }

}

Nó sẽ nhận dữ liệu từ ô input (email và password) từ ui gửi lên bằng phương thức post

Và nó sẽ tạo ra **accessToken, refreshToken ,**và lưu vào cookies

const handleServiceLoginUser = (**userInfor**) => {

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

        try {

            let data = await userCheckEmail(userInfor.email);

*// console.log(data);*

            if (!data) {

                resolve({

                    statusCode: 4,

                    message: "Your login information is incorrect",

                });

                return;

            }

            if (data) {

                const passwordHash = data.password\_hash;

                const userId = data.id;

                const password = await useDecodePassword(userInfor.password, passwordHash);

                if (password && data) {

                    const accessToken = await useAccessToken(userId);

                    const refreshToken = await useRefreshToken(userId);

                    delete data.password\_hash;

                    resolve({

                        statusCode: 2,

                        user: data,

                        accessToken: accessToken,

                        refreshToken: refreshToken,

                    });

                } else {

                    resolve({

                        statusCode: 4,

                        message: "Your login information is incorrect",

                    });

                }

            } else {

                resolve({

                    statusCode: 4,

                    message: "Your login information is incorrect",

                });

            }

        } catch (error) {

            console.log(error);

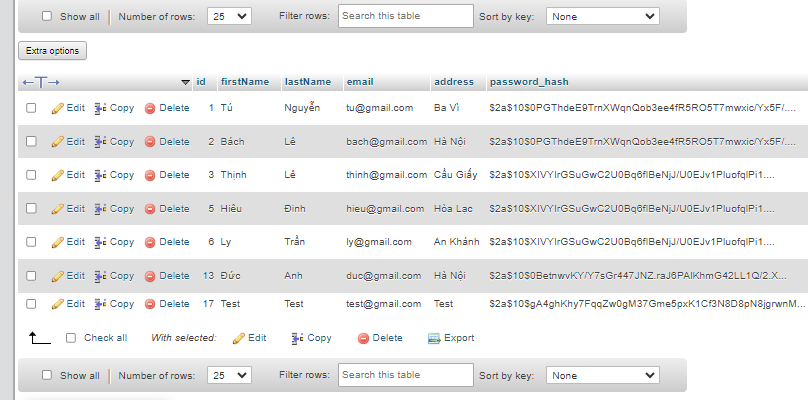
            reject(error);

        }

    });

}

* Check xem user đó đã tồn tại trong ứng dụng hay chưa và phải giải mã password trong db vì passord trong db là chuỗi has để bảo bảo mật khẩu người dùng tránh bị lộ



Nếu password nhập đúng với password đã được has thì mình sẽ cho login

File :petrolimex-ui\src\pages\Login\Login.js

 if (response && response.data.statusCode === 2) {

                    setAccessToken(Cookies.get('accessToken'));

                    setRefreshToken(Cookies.get('refreshToken'));

                }

Nếu đúng thông tin của nhân viên sẽ cho đăng nhập và xét 2 **accessToken và refreshToken** vào cookies để xác định và check xem nhân viên đã đăng nhập chưa. Ko có 2 biến này sẽ check đc là nhân viên đó chưa đăng nhập

  if (response && response.data.statusCode === 4) {

                    toast.error('Thông Tin Tài Khoản Hoặc Mật Khẩu Không Chính Xác', {

                        position: "top-right",

                        autoClose: 5000,

                        hideProgressBar: false,

                        closeOnClick: true,

                        pauseOnHover: true,

                        draggable: true,

                        progress: undefined,

                        theme: "light",

                    });

                }

Nếu thông tin sai thì nó sẽ thông báo 1 toast ra giao diện

A screen shot of a message

Description automatically generated

  useEffect(() => {

        if (stateAccessToken && stateRefreshToken) {

            window.location.href = config.routes.profile;

        }

    }, [navigate, stateAccessToken, stateRefreshToken]);

Đoạn có này thể hiện nếu có **accessToken và refreshToken** chuyển hướng đến trang profile

Tại sao mình lại nói nhiều đến cái **accessToken và refreshToken** vì ngoài liên quan đến đăng nhập. mình sẽ check đc xem nhân viên đó có phải hệ thống của mình tạo ra hay ko => liên quan đến bảo mật.

**Chức năng tạo nhân viên**

**Code sau nằm ở file: project\petrolimex\petrolimex-ui\src\layouts\Header\index.js**

* Chức năng này chỉ có quản lý và của hàng trưởng mới sử dụng đc.

const menus = [

    {

        id: 1,

        name: 'Danh Sách Nhân Viên',

        role: true,

        router: config.routes.employee

    },

    {

        id: 2,

        name: "Chấm Công",

        role: false,

        router: config.routes.timekeeping

    },

    {

        id: 3,

        name: 'Lương',

        role: false,

        router: config.routes.salary

    },

    {

        id: 4,

        name: 'Báo Cáo',

        role: false,

        router: config.routes.report

    },

    {

        id: 5,

        name: 'Phân Quyền Và Chia Ca',

        role: true,

        router: config.routes.ofTheChief

    },

];

{

                    isLogin ?

                        <ul className={style.headerMenus}>

                            {

                                menus && menus.map((**menu**) => {

                                    if (menu.role && userInfor && userInfor.roleId === 'R0') {

                                        return null; *// Nếu menu có role true và roleId là R0, không render menu này*

                                    }

                                    return (

                                        <li key={menu.id} className={style.headerMenuItem}>

                                            <Link to={menu.router} key={menu.id} className={style.headerMenuItem}>

                                                <span>{menu.name}</span>

                                            </Link>

                                        </li>

                                    )

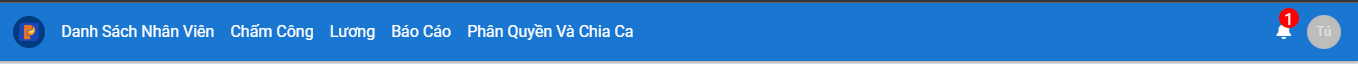
                                })

                            }

                        </ul>

                        : <React.Fragment></React.Fragment>

                }



Đoạn code trên nó sẽ hiển thị ra giao diện các name: (vd: danh sách nhân viên, chấm công…)



Và mình sẽ phân quyền xem nhân viên nào sẽ nhìn thấy name nào: ví dụ quản lý sẽ nhìn thấy tất cả các name còn nhân viên sẽ hiển thị như sau. Dựa vào biến **role : true** thì nhân viên sẽ ko thấy name đó ví dụ ảnh đăng nhập bằng quyền nhân viên



Code sau nằm file: **project\petrolimex\petrolimex-ui\src\pages\Employee\index.js**

     <div className={style.headerEmployee}>

                    <Button variant="contained" disableElevation size="large" sx={{ fontSize: '18px' }} onClick={handleCreateUser}>Tạo Nhân Viên</Button>

                    <Button variant="contained" disableElevation size="large" sx={{ fontSize: '18px' }} onClick={() => handleOpen('notification')}>Tạo Thông Báo</Button>

                </div>

Đoạn code trên sẽ hiển thị ra 2 nut tạo nhân viên và tạo thông báo

A blue rectangles with white text

Description automatically generated

Khi ấn vào nút tạo nhân viên thì nó là sự kiện click chuột thì nó sẽ chạy code sau:  
**onClick={handleCreateUser}>**

 const handleCreateUser = () => {

        navigate(config.routes.register)

    }

Và nó sẽ chuyển hướng đến trang register(tạo nhân viên)

Code sau nằm ở file :**project\petrolimex\petrolimex-ui\src\pages\Register\Register.js**

  const schema = yup.object().shape({

        firstName: yup.string().required("Trường Tên Là Bắt Buộc!"),

        lastName: yup.string().required("Trường Họ Là Bắt Buộc!"),

        address: yup.string().required("Trường Địa Chỉ Là Bắt Buộc!"),

        email: yup.string().email().required("Trường Email Là Bắt Buộc!"),

        password: yup.string().min(4).max(20).required("Trường Mật Khẩu Là Bắt Buộc, Lớn hơn 4 và nhỏ hon 20"),

        confirmPassword: yup

            .string()

            .required("Trường Xác Nhận Mật Khẩu Là Bắt Buộc"),

        dob: yup.string().required("Năm Sinh Là Bắt Buộc!"),

        mobile: yup.string().required('Trường Số Điện Thoại Là Bắt Buộc!'),

        gender: yup.string().required("Giới Tính Là Bắt Buộc!"),

        home\_town: yup.string().required("Quê Hương Là Bắt Buộc!"),

        cccd: yup.string().required('Trường Căn Cước Là Bắt Buộc!'),

        nation: yup.string().required("Quốc Gia Là Bắt Buộc!"),

        education: yup.string().required("Học Vấn Là Bắt Buộc!"),

    });

    const {

        register,

        handleSubmit,

        formState: { errors },

    } = useForm({

        resolver: yupResolver(schema),

    });

  <div className={style.loginWapper}>

                <div className={style.bodyWapper}>

                    <div className={style.headeWapper}>

                        <h1>Register An Account</h1>

                        <p>

                            Register an account to enjoy all the services <br /> without any ads for free!

                        </p>

                    </div>

                    <div className={style.formRegister}>

                        <form onSubmit={handleSubmit(onSubmit)}>

                            <div className={style.formGroup}>

                                <span>Tên</span><br />

                                <input type="text" placeholder="Tên..." {...register("firstName")} className={style.inputRegister} />

                                <p className={style.errorMessage}>{errors.firstName?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Họ</span><br />

                                <input type="text" placeholder="Họ..." {...register("lastName")} className={style.inputRegister} />

                                <p className={style.errorMessage}>{errors.lastName?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Email</span><br />

                                <input type="text" placeholder="Email..." {...register("email")} className={style.inputRegister} />

                                <p className={style.errorMessage}>{errors.email?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Địa Chỉ</span><br />

                                <input type="text" placeholder="Địa chỉ..." {...register("address")} className={style.inputRegister} />

                                <p className={style.errorMessage}>{errors.address?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Mật Khẩu</span><br />

                                <input

                                    type="password"

                                    placeholder="Mật Khẩu..."

                                    {...register("password")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.password?.message ? 'Trường Mật Khẩu Là Bắt Buộc, Lớn hơn 4 và nhỏ hon 20' : ''}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Xác Thực Mật Khẩu</span><br />

                                <input

                                    type="password"

                                    placeholder="Xác Thực Mật Khẩu..."

                                    {...register("confirmPassword")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.confirmPassword?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Năm Sinh</span><br />

                                <input

                                    type="text"

                                    placeholder="Năm Sinh..."

                                    {...register("dob")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.dob?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Số Điện Thoại </span><br />

                                <input

                                    type="text"

                                    placeholder="Số Điện Thoại ..."

                                    {...register("mobile")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.mobile?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Giới Tính</span><br />

                                <input

                                    type="text"

                                    placeholder="Giới Tính..."

                                    {...register("gender")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.gender?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Quê Hương</span><br />

                                <input

                                    type="text"

                                    placeholder="Quê Hương..."

                                    {...register("home\_town")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.home\_town?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Căn Cước</span><br />

                                <input

                                    type="text"

                                    placeholder="Căn Cước..."

                                    {...register("cccd")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.cccd?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Quốc Gia</span><br />

                                <input

                                    type="text"

                                    placeholder="Quốc Gia..."

                                    {...register("nation")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.nation?.message}</p>

                            </div>

                            <div className={style.formGroup}>

                                <span>Học Vấn</span><br />

                                <input

                                    type="text"

                                    placeholder="Học Vấn..."

                                    {...register("education")}

                                    className={style.inputRegister}

                                />

                                <p className={style.errorMessage}>{errors.education?.message}</p>

                            </div>

                            <input type="submit" value="Đăng Ký" className={style.submitBtn} />

                        </form>

                    </div>

                    <div className={style.footerWapper}>

                        <p>

                            Already Have An Account?{' '}

                            <Link to={config.routes.login} className={style.navLink}>

                                Login

                            </Link>

                        </p>

                    </div>

                </div>

            </div>

Code trên nó sẽ hiển thị mấy o input(tên, họ…) và lỗi tương ứng với các ô input khi mình nhập sai:

A screen shot of a computer

Description automatically generated

Khi mình nhập đúng sẽ thông tin ở ô input đúng nó sẽ gửi lên server thông tin mình nhập ở ô input

const resultRegister = await handRegisterUser(data)

file service ở server: **petrolimex\petrolimex-server\src\services\useServices.js**

const data = await userCheckEmail(email);

*// console.log('data', data);*

            const hashPassword = await useHasPassword(password);

            if (!data) {

                await db.User.create({

                    firstName: firstName,

                    lastName: lastName,

                    email: email,

                    password\_hash: hashPassword,

                    cccd: cccd,

                    dob: dob,

                    education: education,

                    gender: gender,

                    home\_town: home\_town,

                    nation: nation,

                    mobile: mobile,

                    address: address

                });

                resolve({ statusCode: 2, message: "create user successful" });

            } else {

                resolve({ statusCode: 4, message: `${data.email} Your have not` });

            }

Check xem email đã tồn tại chưa và has passowrod của nhân viên chứ ko để nguyên mật khẩu của nhân viên vì sẽ nguy cơ bị lộ mật khẩu cx như để bảo mật passoword của nhân viên.

Nếu email chưa có trong db và đã has đc pasword và sẽ lưu vào db

  await db.User.create({

                    firstName: firstName,

                    lastName: lastName,

                    email: email,

                    password\_hash: hashPassword,

                    cccd: cccd,

                    dob: dob,

                    education: education,

                    gender: gender,

                    home\_town: home\_town,

                    nation: nation,

                    mobile: mobile,

                    address: address

                });

Nó sẽ tạo các thông tin nhập từ ô input vào db: **db.User.create (**db: đại diện cho mysql,User đại diện cho bảng user trong db và create là tạo**)**

A screenshot of a computer

Description automatically generated

Nếu nó là 4 thì email đã tồn tại hiển thị thông báo sau:

  if (resultRegister && resultRegister?.data?.statusCode === 4) {

                toast.error("Email Này Đã Đăng Ký, Vui Lòng Sử Dụng Email Khác Để Khác Ký Tài Khoản", {

                    position: "top-right",

                    autoClose: 5000,

                    hideProgressBar: false,

                    closeOnClick: true,

                    pauseOnHover: true,

                    draggable: true,

                    progress: undefined,

                    theme: "light",

                });

            }

A screenshot of a computer

Description automatically generated

Nếu là statcode = 2:

navigate(config.routes.employee)

chuyển hướng về trang nhân viên

**chức năng tạo thông báo**

khi ấn vào nút tạo thông báo:

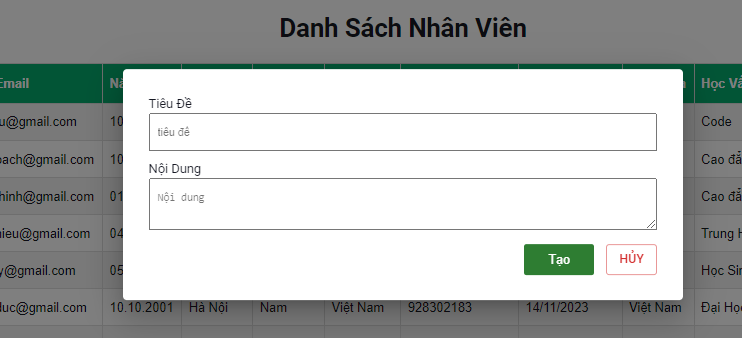
                    <Button variant="contained" disableElevation size="large" sx={{ fontSize: '18px' }} onClick={() => handleOpen('notification')}>Tạo Thông Báo</Button>

 const handleOpen = (**type**) => {

        setOpen(true);

    }

Nó sẽ mở lên 1 modal sau:



Khi ấn hủy nó sẽ đóng modal lại

Khi chưa nhập gì nhấn tạo sẽ báo lỗi :

   const schema = yup.object().shape({

        title: yup.string().required("Title is Required!"),

        content: yup.string().required("Content is Required!"),

    });

A screenshot of a computer

Description automatically generated

khi nhập đúng tiêu đề và nội dụng ấn nút tạo sẽ gửi lên server:

  await db.Notification.create({

                userId: userId,

                title: notificationData.title,

                content: notificationData.content

            })

            resolve({ statusCode: 2, message: "create notification successful" });

sẽ lưu vào trong db:

A screenshot of a computer

Description automatically generated

và hiển thị thông báo

A screen shot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

Danh sách thông báo đã tạo:

   <Menu

                                        sx={{ mt: '45px', fontSize: '16px', }}

                                        id="menu-appbar"

                                        anchorEl={bellNoti}

                                        anchorOrigin={{

                                            vertical: 'top',

                                            horizontal: 'right',

                                        }}

                                        keepMounted

                                        transformOrigin={{

                                            vertical: 'top',

                                            horizontal: 'right',

                                        }}

                                        open={Boolean(bellNoti)}

                                        onClose={handleCloseBellMenu}

                                        disableScrollLock={true}

                                    >

                                        {

                                            listNotification && listNotification.map((**notification**) => {

                                                return (

                                                    <MenuItem onClick={handleCloseBellMenu} sx={{ fontSize: '16px', display: 'flex', flexDirection: 'column', justifyContent: 'flex-start' }} key={notification?.id}>

                                                        <h2>{notification?.title}</h2>

                                                        <p>{notification?.content}</p>

                                                    </MenuItem>

                                                )

                                            })

                                        }

                                    </Menu>

Code trên nó sẽ hiển thị ra danh sách thông báo như trên ảnh

Và biến **listNotification** mình sẽ lấy từ server tương đương với phương thức get:

 const allNotification = await db.Notification.findAll({

                order: [["id", "DESC"]],

            })

            resolve({

                statusCode: 2,

                message: allNotification,

            });

Server nó sẽ lấy tất cả các thông báo và sắp xếp theo id (lớn đến nhỏ vì vậy khi tạo mới thông báo nó sẽ hiển thị thông báo mới nhất ở trên đầu) từ db là trả về cho ui

Chức năng sửa nhân viên:

 if (userData?.roleId === 'R1' && roleId === "R2") {

            toast.warn("Bạn Không Thể Sửa Người Có Quyền lớn hơn mình", {

                position: "top-right",

                autoClose: 5000,

                hideProgressBar: false,

                closeOnClick: true,

                pauseOnHover: true,

                draggable: true,

                progress: undefined,

                theme: "colored",

            });

        }

code trên check xem cửa hàng trưởng có quyền edit thông tin của quản lý ko:

  const findUser = await db.User.findOne({

                where: {

                    id: users.users.userId

                },

                attributes: {

                    exclude: ["password\_hash", 'refresh\_token'],

                },

                raw: false,

                nest: true,

            })

Code trên Check xem nhân viên đó có tồn tại trong db ko. Và sẽ ko lấy password và refresh\_toen trong db

Nếu tìm thấy:

  if (findUser) {

                findUser.firstName = users.users.firstName

                findUser.lastName = users.users.lastName

                findUser.address = users.users.address

                findUser.dob = users.users.dob

                findUser.home\_town = users.users.home\_town

                findUser.gender = users.users.gender

                findUser.education = users.users.education

                findUser.nation = users.users.nation

                findUser.mobile = users.users.mobile ? users.users.mobile : null

                findUser.cccd = users.users.cccd

            }

            await findUser.save();

nó sẽ sửa thông tin mà mình sửa vào trong db vào lưu(save)

**Chức năng xóa nhân viên:**

 if (userData && userData.id === userId) {

            toast.warn("Bạn Không Thể Xóa Chính Mình", {

                position: "top-right",

                autoClose: 5000,

                hideProgressBar: false,

                closeOnClick: true,

                pauseOnHover: true,

                draggable: true,

                progress: undefined,

                theme: "colored",

            });

        } else if (userData?.roleId === 'R1' && roleId === "R2") {

            toast.error("Bạn Không Thể Xóa Người Có Quyền Lơn Hơn Mình", {

                position: "top-right",

                autoClose: 5000,

                hideProgressBar: false,

                closeOnClick: true,

                pauseOnHover: true,

                draggable: true,

                progress: undefined,

                theme: "colored",

            })

        }

Code trên check ko thể xóa chính mình cx như có quyền lớn hơn mình

 await db.User.destroy({

                where: {

                    id: deleteUserId.userId

                }

            })

            resolve({

                statusCode: 2,

                message: `the user in deleted`,

            });

Bên server sẽ xóa nhân viên đó ở trong db

<h1 className={style.title}>Danh Sách Nhân Viên</h1>

                <table className={style.customers}>

                    <tbody>

                        <tr>

                            <th>Mã Nhân Viên</th>

                            <th>Full Name</th>

                            <th>Email</th>

                            <th>Năm Sinh</th>

                            <th>Địa Chỉ</th>

                            <th>Giới Tính</th>

                            <th>Quê Quán</th>

                            <th>Căn Cước</th>

                            <th>Ngày Vào Làm</th>

                            <th>Quốc Gia</th>

                            <th>Học Vấn</th>

                            <th>Chức Vụ</th>

                            <th>Actions</th>

                        </tr>

                        {allUser && allUser.map(**user** => (

                            <tr key={user.id}>

                                <td>Mã Nhân Viên : {user.id}</td>

                                <td>{`${user.lastName} ${user.firstName}`}</td>

                                <td>{user.email || "Chưa có thông tin"}</td>

                                <td>{user.dob || "Chưa có thông tin"}</td>

                                <td>{user.address || "Chưa có thông tin"}</td>

                                <td>{user.gender || "Chưa có thông tin"}</td>

                                <td>{user.home\_town || "Chưa có thông tin"}</td>

                                <td>{user.cccd || "Chưa có thông tin"}</td>

                                <td>{new Date(user.createdAt).toLocaleDateString('vi-VN') || "Chưa có thông tin"} </td>

                                <td>{user.nation || "Chưa có thông tin"}</td>

                                <td>{user.education || "Chưa có thông tin"}</td>

                                <td>{user?.roleData?.role\_name || "Chưa có thông tin"}</td>

                                <td className={style.btnAction}>

                                    <Button variant="contained" onClick={() => handleEditUser(user.id, user.roleId)}>Sửa</Button>

                                    <Button variant="outlined" color="error" onClick={() => handleDeleUser(user.id, user.roleId)}>

                                        Xóa

                                    </Button>

                                </td>

                            </tr>

                        ))}

                    </tbody>

                </table>

Code trên sẽ hiển thị ra ui

A screenshot of a computer

Description automatically generated

Chấm công:

 const title = `Mã Nhân Viên : ${userData?.id} - ${userData.lastName} ${userData.firstName} ,

                Giờ Vào ${startTime.hours}h${startTime.minutes}p Ngày ${startTime.day} Tháng ${startTime.month} Năm ${startTime.year}

            `

 const title = `Mã Nhân Viên : ${userData?.id} - ${userData.lastName} ${userData.firstName} ,

            Giờ Ra ${endTime.hours}h${endTime.minutes}p Ngày ${endTime.day} Tháng ${endTime.month} Năm ${startTime.year}

        `

Code treeb Nó sẽ lấy đc các thông tin như sau: mã nhân viên và tên, họ , giờ vào, giờ ra và sẽ gửi lên server

  await db.Timekeeping.create({

                userId: timekeeping.userId,

                time: timekeeping.time,

                type: timekeeping.type

            })

            resolve({ statusCode: 2, message: "create timekeeing successful" });

code trên server sẽ lưu vào db gồm mã nhân viên gồm mã nhân viên thời gian và giờ vào giờ ra

**chức năng tính lương**

  const calculateYearsWorked = (**startDate**, **currentSalary**, **increasePerCycle**) => {

*// console.log(startDate, currentSalary, increasePerCycle);*

        const today = new Date();

        const start = new Date(startDate);

        const differenceInTime = today.getTime() - start.getTime();

        const differenceInYears = differenceInTime / (1000 \* 3600 \* 24 \* 365.25); *// Số năm*

        const cycles = Math.floor(differenceInYears / 5); *// Số chu kỳ 5 năm đã trôi qua*

        console.log(cycles);

        if (cycles > 0) {

*// const newSalary = currentSalary + (cycles \* increasePerCycle);*

            const newSalary = parseInt(currentSalary) + parseInt(increasePerCycle);

*// console.log(newSalary);*

            const formattedNumber = new Intl.NumberFormat('vi-VN', { style: 'currency', currency: 'VND' }).format(newSalary);

*// console.log(formattedNumber);*

            return formattedNumber;

        } else {

            const result = new Intl.NumberFormat('vi-VN', { style: 'currency', currency: 'VND' }).format(currentSalary);

            return result;

        }

    };

Code trên Tính xem thâm niên của nhân viên là bao lâu đê quyết định tăng lương

**Chức năng báo cáo:**

  <table>

                                <tbody>

                                    <tr>

                                        <th>STT</th>

                                        <th>Ngày Bán</th>

                                        <th>Số Săng Bán Được</th>

                                        <th>Giá</th>

                                        <th>Thành Tiền</th>

                                    </tr>

                                    {

                                        listSales && listSales.map((**sale**, **index**) => {

                                            return (

                                                < tr key={sale.id}>

                                                    <td>{index + 1}</td>

                                                    <td>{sale?.day\_for\_sale}</td>

                                                    {*/\* <td>{sale?.price}</td> \*/*}

                                                    {*/\* <td>{sale?.sales\_figures\_day \* sale?.price} vnd</td> \*/*}

                                                    <td>{sale?.sales\_figures\_day}</td>

                                                    <td>{new Intl.NumberFormat('vi-VN', { style: 'currency', currency: 'VND' }).format(sale?.price)}</td>

                                                    <td>{new Intl.NumberFormat('vi-VN', { style: 'currency', currency: 'VND' }).format(sale?.sales\_figures\_day \* sale?.price)}</td>

                                                </tr>

                                            )

                                        })

                                    }

                                </tbody>

                            </table>

Code trên sẽ hiển thị ra danh sách báo cáo

<Button variant="contained" onClick={exportToExcel} sx={{ fontSize: '16px' }}>Xuất Excel</Button>

const exportToExcel = () => {

        const data = listSales.map((**sale**, **index**) => {

            const formattedDate = formatDate(sale.day\_for\_sale);*// Chuyển đổi ngày tháng thành đối tượng Date*

            return {

                'STT': index + 1,

                'Ngày Bán': formattedDate, *// Định dạng lại ngày tháng*

                'Số Săng Bán Được': sale.sales\_figures\_day,

                'Giá': sale.price,

                'Thành Tiền': sale.sales\_figures\_day \* sale.price + ' vnd'

            };

        });

        const worksheet = XLSX.utils.json\_to\_sheet(data);

        const workbook = XLSX.utils.book\_new();

        XLSX.utils.book\_append\_sheet(workbook, worksheet, 'SalesData');

        XLSX.writeFile(workbook, 'sales\_data.xlsx');

        saveExcelDb(JSON.stringify(workbook));

    };

Code trên chức năng xuất excel;

 <Button variant="contained" color="success" onClick={ResetSales} sx={{ fontSize: '16px' }}>Reset</Button>

 const ResetSales = async () => {

        const response = await apiResetSales({ TIME\_RESET, isResetting: true })

        console.log(response)

        if (response?.data?.statusCode === 2) {

            const response = await apiResetSales({ TIME\_RESET, isResetting: false })

            console.log(response);

        }

*// setIsResetting(false)*

*// console.log(response?.data?.statusCode === 2)*

    }

 const timeInSeconds = parseInt(timeRest.TIME\_RESET); *// Chuyển đổi giá trị thời gian sang số nguyên*

            if (timeRest.isResetting) {

                resetJob = cron.schedule(`\*/${timeInSeconds} \* \* \* \* \*`, async () => {

                    try {

                        await db.Sales.destroy({

                            where: {}, *// Điều kiện để xóa, rỗng để xóa hết dữ liệu*

                            truncate: true *// Chọn truncate để xóa dữ liệu nhanh hơn*

                        });

                        resolve({ statusCode: 2, message: 'Reset sale successful' });

                    } catch (error) {

                        resolve({ statusCode: 4, message: 'Reset sale error' });

                    }

                });

Code trên thể hiện chức năng **reset:** check xem đã hết thời gian chưa nếu hết thời gian thì nó sẽ xóa ở trong db

Chức năng phân quyền và chia ca:

<tbody>

                        <tr>

                            <th>Mã Nhân Viên</th>

                            <th>Full Name</th>

                            <th>Email</th>

                            <th>Năm Sinh</th>

                            <th>Chức Vụ</th>

                            <th>Actions</th>

                        </tr>

                        {allUser && allUser.map(**user** => (

                            <tr key={user.id}>

                                <td>Nhân Viên : {user.id}</td>

                                <td>{`${user.lastName} ${user.firstName}`}</td>

                                <td>{user.email || "Chưa có thông tin"}</td>

                                <td>{user.dob || "Chưa có thông tin"}</td>

                                <td>{user?.roleData?.role\_name || "Chưa có thông tin"}</td>

                                <td className={style.btnAction}>

                                    <Button variant="contained" onClick={() => handleOpen('auth', user.id, user.roleId)} sx={{ fontSize: '12px' }} >

                                        Phân Quyền

                                    </Button>

                                    <Button variant="contained" onClick={() => handleOpen('shifts', user.id, user.roleId)} sx={{ fontSize: '12px' }} >

                                        Chia Ca

                                    </Button>

                                </td>

                            </tr>

                        ))}

                    </tbody>

Code trên hiển thị ra danh sách nhân viên

 if (userData.id === userId) {

                    toast.warn("Bạn Không Thể Thêm Quyền Cho Chính Mình", {

                        position: "top-right",

                        autoClose: 5000,

                        hideProgressBar: false,

                        closeOnClick: true,

                        pauseOnHover: true,

                        draggable: true,

                        progress: undefined,

                        theme: "colored",

                    });

                    setOpen(false)

                }

                if (userData?.roleId === 'R1' && roleId === "R2") {

                    toast.warn("Bạn Không Thể Thêm Quyền Cho Người Có Quyền Lớn Hơn Mình", {

                        position: "top-right",

                        autoClose: 5000,

                        hideProgressBar: false,

                        closeOnClick: true,

                        pauseOnHover: true,

                        draggable: true,

                        progress: undefined,

                        theme: "colored",

                    });

                    setOpen(false)

                }

Code trên check có thêm quyền cho mình và người có quyền lớn hơn mình ko. Và thông báo ra

 const userData = await db.User.findOne({

                where: {

                    id: role.userId

                },

                attributes: ['id', 'roleId'],

                raw: false,

                nest: true,

            })

*// console.log(role);*

*// console.log(userData);*

            if (userData) {

                userData.roleId = role.role

            }

            await userData.save();

code trên thể hiện thêm quyền

**chức năng chia ca:**

await db.Shift.create({

                userId: dataShift.userId,

                time: dataShift?.data?.shifts

            })

const response = await apiCreateShift({ data: data, userId: userId })

            if (response && response?.data?.statusCode === 2) {

                toast.success("Tạo Ca Làm Thành Công", {

                    position: "top-right",

                    autoClose: 5000,

                    hideProgressBar: false,

                    closeOnClick: true,

                    pauseOnHover: true,

                    draggable: true,

                    progress: undefined,

                    theme: "colored",

                });

                setOpen(false)

            }

Tạo ca thành công sẽ hiển thị ra ui

Và lưu vào db:

await db.Shift.create({

                userId: dataShift.userId,

                time: dataShift?.data?.shifts

            })