Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**РАЗРАБОТКА ПРОГРАММЫ**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Выполнили:

студенты гр.ИСТ-19-1б, ИСТ-19-2б

Баяндин А.В., Михайлишин А.А.

Принял:

ст. преподаватель, Банников Р.Ю.

**Пермь 2023**

РЕАЛИЗАЦИЯ ПРОГРАММЫ

**Цель работы:** запрограммировать обозначенный в ТЗ функционал с учётом созданной БД и разработанных структур классов.

**Задачи:**

1. Перенести пользовательский интерфейс в код;
2. Прописать всю логику программы, описанную в техническом задании.
3. Разработка программы
   1. Создание UI-компонентов

Для правильного выполнения поставленных задач, необходим UI-пакет базовых компонентов, таких как кнопка, текстовой поле и т.д. Разработчиками было принято решение написать этот пакет самостоятельно. Были написаны следующие компоненты:

1. Button – кнопка (прил. 1);
2. ExportButton – кнопка экспорта (прил. 2);
3. InputText- поле для текстового ввода (прил. 3);
4. InputCheckBox – поле переключателя (прил. 4);
5. Modal – модальное окно с различными данными (прил. 5);
6. Table – таблица (использована библиотека react-table).
   1. Инициализация и создание хранилища состояний

В списке технологий, которые предлагались для работы над системой, был менеджер состояний Redux и его дополнение Redux Toolkit. Они и были использованы для работы над системой.

Для их корректной работы необходимо их подключить, инициализировать и создать.

* 1. Вёрстка страниц

Для работы над системой был выбран фреймворк React, который поддерживает и поощряет функциональное программирование. Исходя из чего, классовых компонентов в системы будет очень мало.

Система, на этапе проработки интерфейса, была разделена на четыре части:

1. Страница авторизации (прил. 6);
2. Домашняя страница (прил. 7);
3. Панель управления (прил. 8);
4. Страница с таблицей (прил. 9).

Каждая страница имеет свою логику, поэтому они были вынесены в отдельные компоненты, т.е. каждая страница – отдельный компонент, со своей логикой.

Переходы между страницами возможны благодаря библиотеке react-router-dom, которая позволяет без перезагрузки страницы менять используемые компонент для его отрисовки.

* 1. Проработка логики страниц

После вёрстки страниц, необходимо соединить их с серверной частью, чтобы получать данные, а также иметь возможность их изменять. В качестве технологии общения, была выбрана библиотека axios, которая позволяет отправлять асинхронные запросы на сервер. Выглядит это следующим образом:

export const fetchUserEvents = () => (dispatch: (arg0: IGetUserEvents) => void) => {

    request(RequestTypesEnum.get, RequestApiEnum.getUserEvents, null)

        .then(res => {

            const { data } = res;

            if (!data) {

                return null;

            }

            dispatch({

                type: TABLE\_GET\_USER\_EVENTS,

                userEvents: data,

            })

        })

}

То есть у нас есть встроенная функция request, которая принимает три параметра:

1. Метод запроса: get, pos, delete, put и т.д;
2. API, по которому нужно проводить запрос;
3. Тело запроса (подходит только для post).

Запрос возвращает нам объект типа Promise, из которого мы можем узнать статус запроса и забрать из него данные.

* 1. Добавление реакции на ошибочный ввод

Ошибочный ввод бывает разный, и разработчики могут упростить этап тестирования тем, что наложат правила на вводимые значения. Так, реакции можно разделить на два вида.

* + 1. Валидация полей

Один из способов добавления реакций программы на ввод некорректных данных – валидация полей.

Валидация полей – наложение правил на действия пользователя системы.

Так, например, для страницы авторизации были созданы следующие правила:

1. Логин:
   * Только английские символы;
   * Обязательно к заполнению;
   * Только строки (нельзя использовать только цифры и первый символ – буква.
2. Пароль:
   * Любые символы;
   * Обязательно к заполнению;
   * Не менее 6-ти символов.

Если валидация полей выдает ошибку, то пользователь это увидит. В случае с КИС «ППО ЕвроХим» пользователь не сможет пройти аутентификаю пока не заполнить правильно все поля. Для удобной работы с валидацией были использованы следующие библиотеки:

* React-Hook-Form;
* Yup.

Выглядит это следующим образом:

export const schema = yup.object({

    username: yup

        .string()

        .required('Поле не может быть пустым'),

    password: yup

        .string()

        .required('Поле не может быть пустым')

        .min(6, 'Не менее 6 символов')

});

* + 1. Сообщение об ошибке

Ещё одним способом оповещения пользователя об ошибке является прерывание работы системы и его информирование. С этой работой хорошо справляется модальное окно, которое блокирует всю страницу и выводит отдельное окно с сообщением об ошибке (рис. 1).

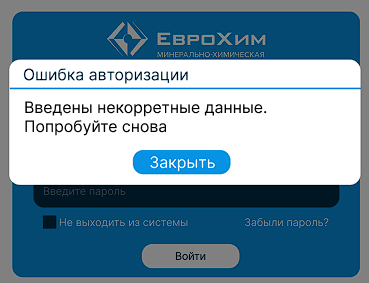


Рис. 1 – Модальное окно с ошибкой

1. Выводы

В данной лабораторной работе была разработана программа информационной системы ППО ЕвроХим. Реализован основной функционал: аутентификация, вид под разными пользователями (в зависимости от роли), редактировании базы данных, формирование отчетов, реакция системы на ошибочный ввод.

Приложение

UI-компонент «Button»

interface IButton {

    onClick?: any;

    title: string;

    id?: string;

    className?: string;

}

const Button: React.FC<IButton> = ({

    onClick,

    id,

    className = '',

    title

}) => {

    const classNames = cx('button', className)

    return (

        <button className={classNames} id={id} onClick={onClick}>{title}</button>

    )

}

export { Button };

UI-компонент «ExportButton»

interface IExportButton {

    className?: string;

    url: string;

}

const ExportButton: React.FC<IExportButton> = ({ className, url }) => {

    const classNames = cx('export-button', className);

    return (

        <a className={classNames} href={url}>

            <Row noFlex ai={AlignItemsTypes.center}>

                <Row className="export-button\_\_title" fullHeight ai={AlignItemsTypes.center}>Экспорт</Row>

                <FontAwesomeIcon className="export-button\_\_icon" icon={faDownload} />

            </Row>

        </a>

    )

}

export { ExportButton };

UI-компонент «InputText»

interface IInputText {

    id: string;

    type: InputTypesEnum;

    name?: string;

    label?: string;

    onClick?: any;

    onChange?: any;

    classNameInput?: string;

    classNameLabel?: string;

    classNameContainer?: string;

    placeholder?: string;

    error?: any;

    register?: any;

    value?: string;

}

const InputText: React.FC<IInputText> = ({

    id,

    type = "text",

    name = '',

    label,

    onChange,

    classNameInput = '',

    classNameLabel = '',

    classNameContainer = '',

    placeholder = '',

    error,

    register,

    value,

    ...props

}) => {

    const inputClassNames = cx('input-text', classNameInput);

    const labelClassNames = cx('input-text\_\_label', classNameLabel);

    const containerClassNames = cx('input-text\_\_wrapper', classNameContainer);

    return (

        <Column className={containerClassNames}>

            <Row ai={AlignItemsTypes.center} jc={JustifyContentTypes.spaceBetween}>

                {label && <label className={labelClassNames} htmlFor={id}>{label}</label>}

                {error && <p className="input-text\_\_error">{error.message}</p>}

            </Row>

            <input

                id={id}

                name={name}

                className={inputClassNames}

                placeholder={placeholder}

                type={type}

                onChange={onChange}

                value={value}

                {...register}

                {...props}

            />

        </Column>

    )

}

export { InputText };

UI-компонент «InputCheckBox»

interface IInputCheckbox {

    id: string;

    name?: string;

    label?: string;

    onClick?: any;

    onChange?: any;

    classNameInput?: string;

    classNameLabel?: string;

    classNameContainer?: string;

    checked?: boolean;

    errors?: any;

}

const InputCheckbox: React.FC<IInputCheckbox> = ({

    id,

    name = '',

    label,

    onChange,

    classNameInput = '',

    classNameLabel = '',

    classNameContainer = '',

    checked,

    errors,

    ...props

}) => {

    const inputClassNames = cx('input-checkbox', classNameInput);

    const labelClassNames = cx('input-checkbox\_\_label', classNameLabel);

    const containerClassNames = cx('input-checkbox\_\_wrapper', classNameContainer);

    return (

        <Row className={containerClassNames} ai={AlignItemsTypes.center} noFlex>

            <input

                className={inputClassNames}

                id={id}

                name={name}

                type="checkbox"

                checked={checked}

                onChange={onChange}

                {...props}

            />

            <label className={labelClassNames} htmlFor={id}>{label}</label>

        </Row>

    )

}

export { InputCheckbox };

UI-компонент «Modal»

interface IModalProps {

    type: ModalTypes;

    closeModal: any;

    option: any;

}

const Modal: React.FC<IModalProps> = ({type, closeModal, option}) => {

    return (

        <Column ai={AlignItemsTypes.center} jc={JustifyContentTypes.center} className="modal-window">

            <Column className='modal-window\_\_wrapper'>

                <FontAwesomeIcon className='modal-window\_\_close' onClick={() => closeModal()} icon={faXmark} />

                {getModal(type, closeModal, option)}

            </Column>

        </Column>

    );

}

export { Modal };

Страница авторизации

interface IAuthPage {

    auth: boolean;

}

const AuthPage: React.FC<IAuthPage> = ({ auth }) => {

    const [rememberLogInData, setRememberLogInData] = useState<boolean>(false);

    const navigate = useNavigate();

    const dispatch = useDispatch();

    useEffect(() => {

        if (auth) {

            navigate("/home");

        }

    }, [auth, navigate])

    const {

        register,

        handleSubmit,

        formState: {

            errors

        }

    } = useForm<ILoginData>({

        mode: 'onSubmit',

        resolver: yupResolver(schema)

    });

    const onChangeHandler = () => {

        setRememberLogInData(!rememberLogInData);

    }

    const onSubmit: SubmitHandler<ILoginData> = data => {

        // @ts-ignore

        dispatch(fetchUser(data, rememberLogInData));

    }

    return (

        <Column className="auth-page" fullHeight ai={AlignItemsTypes.center} jc={JustifyContentTypes.center}>

            <Column className="auth-page\_\_body" ai={AlignItemsTypes.center}>

                <Row className="auth-page\_\_company" noFlex>

                    <HomeSvgSelector icon={ImageEnum.logo} />

                </Row>

                <p className="auth-page\_\_title">Авторизация</p>

                <form className="auth-page\_\_form" onSubmit={handleSubmit(onSubmit)}>

                    <Column className="auth-page\_\_fields">

                        <Input

                            type={InputTypesEnum.text}

                            id="user-login"

                            label="Логин"

                            classNameInput='auth-page\_\_form-input'

                            classNameLabel='auth-page\_\_form-label'

                            placeholder='Введите логин'

                            error={errors.username}

                            register={register('username')}

                        />

                        <Input

                            type={InputTypesEnum.password}

                            id="user-password"

                            name="password"

                            label="Пароль"

                            classNameInput='auth-page\_\_form-input'

                            classNameLabel='auth-page\_\_form-label'

                            placeholder='Введите пароль'

                            error={errors.password}

                            register={register('password')}

                        />

                    </Column>

                    <Row className="manage-panel" jc={JustifyContentTypes.spaceBetween} ai={AlignItemsTypes.center}>

                        <Input

                            type={InputTypesEnum.checkbox}

                            id="user-leave"

                            label="Не выходить из системы"

                            classNameContainer='manage-panel\_\_leave'

                            checked={rememberLogInData}

                            onChange={onChangeHandler}

                        />

                        <p className="manage-panel\_\_forgot">Забыли пароль?</p>

                    </Row>

                    <Button title="Войти" className="auth-page\_\_onSubmit" />

                </form>

            </Column>

        </Column>

    )

}

const mapStateToProps = (state: any) => {

    const { user } = state;

    return {

        auth: user.auth,

    }

}

export default connect(mapStateToProps)(AuthPage);

Домашняя страница

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

    const withHeader = true;

    return (

        <>

            {withHeader && <Header />}

            <Row className="home-page" jc={JustifyContentTypes.spaceAround} fullHeight>

                {/\* <HomeHelp /> \*/}

                <HomeNews />

                <HomeEvents />

            </Row>

        </>

    )

}

export { HomePage };

Панель управления

const DashboardLayout = () => {

    return (

        <Column className="dashboard-layout" fullHeight ai={AlignItemsTypes.center}>

            <p className="dashboard-layout\_\_title">Панель управления</p>

            <Column className="dashboard-layout\_\_controls" noFlex ai={AlignItemsTypes.center} fullHeight>

                {ControlRows.map(({id, panels}) => {

                    return <DashboardRow key={id} panels={panels} />

                })}

            </Column>

        </Column>

    )

}

export { DashboardLayout };

Страница с таблицей

interface IDashboardMore {

    page: IDashboardPage;

    users: IUserModel[] | null;

    awards: IAwardsModel[] | null;

    events: IEventsModel[] | null;

    userAwards: IUserAwardsModel[] | null;

    userEvents: IUserEventsModel[] | null;

}

const DashboardMore: React.FC<IDashboardMore> = ({

    page,

    users,

    awards,

    events,

    userAwards,

    userEvents

}) => {

    const { id, title, exportUrl, tableConfig, isClickable } = page;

    const [selectedRowIndex, setSelectedRowIndex] = useState<string | null>(null);

    const dispatch = useDispatch();

    const tableData = getTableData(id, users, awards, events, userAwards, userEvents);

    useEffect(() => {

        // @ts-ignore

        dispatch(fetchUsers())

        // @ts-ignore

        dispatch(fetchAwards())

        // @ts-ignore

        dispatch(fetchEvents())

        // @ts-ignore

        dispatch(fetchUserAwards())

        // @ts-ignore

        dispatch(fetchUserEvents())

    }, [dispatch])

    const navigate = useNavigate();

    const onBackButtonHandler = () => {

        return navigate('/dashboard');

    }

    const closeModalHandler = () => {

        // @ts-ignore

        dispatch(closeModal());

    }

    const onAddMemberButtonHandler = () => {

        // @ts-ignore

        dispatch(openModal(ModalTypes.addUser, closeModalHandler, null));

    }

    const onRemoveMemberButtonHandler = () => {

        if (!selectedRowIndex || !tableData || id !== DashboardPagesUrlEnum.users) {

            return null;

        }

        // @ts-ignore

        const ids = tableData[Number(selectedRowIndex)].id;

        // @ts-ignore

        dispatch(deleteUser(ids));

    }

    const onChangeMemberButtonHandler = () => {

        if (!tableData || !selectedRowIndex) {

            return null;

        }

        const choosenUser = tableData[Number(selectedRowIndex)];

        // @ts-ignore

        dispatch(openModal(ModalTypes.editUser, closeModalHandler, choosenUser));

    }

    return (

        <Column className="dashboard-more">

            <Row className="dashboard-more\_\_work-elements" ai={AlignItemsTypes.center}>

                <FontAwesomeIcon className="dashboard-more\_\_back" icon={faArrowLeft} onClick={onBackButtonHandler} />

                <p className="dashboard-more\_\_title">{title}</p>

                <ExportButton url={exportUrl} />

                {id === DashboardPagesUrlEnum.users && (

                    <Row

                        className="dashboard-more\_\_user-edit"

                        ai={AlignItemsTypes.center}

                        jc={JustifyContentTypes.spaceBetween}

                    >

                        <FontAwesomeIcon

                            className="dashboard-more\_\_button-user-plus"

                            icon={faUserPlus}

                            onClick={onAddMemberButtonHandler}

                        />

                        <FontAwesomeIcon

                            className="dashboard-more\_\_button-user-minus"

                            icon={faUserMinus}

                            onClick={onRemoveMemberButtonHandler}

                        />

                        <FontAwesomeIcon

                            className="dashboard-more\_\_button-user-edit"

                            icon={faUserPen}

                            onClick={onChangeMemberButtonHandler}

                        />

                    </Row>

                )}

            </Row>

            <Table

                config={tableConfig}

                tableData={tableData}

                isClickable={isClickable}

                selectedRowIndex={selectedRowIndex}

                setSelectedRowIndex={setSelectedRowIndex}

            />

        </Column>

    )

}

const mapStateToProps = (state: any) => {

    const { table } = state;

    return {

        users: table.users,

        awards: table.awards,

        events: table.events,

        userAwards: table.userAwards,

        userEvents: table.userEvents

    }

}

export default connect(mapStateToProps)(DashboardMore);