**Лабораторна робота № 10**

**Тема:** Основи роботи із бібліотекою Redux.js

**Мета:** Навчитись працювати з Redux.js.

**Завдання**

1. Потрібно розробити React проект із використанням Redux, для роботи із вашою

API, що була створена на практичній роботі №6.

**Хід роботи**

**yarn create react-app lr10 --template typescript**

App.tsx

import { Provider as ReduxProvider } from 'react-redux';

import './App.css';

import Home from './pages/Home';

import store from './store';

function App() {

  return (

    <ReduxProvider store={store}>

      <div className="App">

        <Home />

      </div>

    </ReduxProvider>

  );

}

export default App;

Home.jsx

import { useCallback, useState, useRef } from 'react';

import { useDispatch } from 'react-redux';

import get from '../services/api/get';

import addClient from '../services/api/addClient';

import deleteClient from '../services/api/deleteClient';

import {

    getClientsFailure, getClientsStart, getClientsSuccess,

    addClientStart, addClientSuccess, addClientFailure,

    deleteClientFailure, deleteClientStart, deleteClientSuccess

} from '../store/actions/clients';

import './Home.css';

function Home() {

    const name = useRef();

    const kindOfProperty = useRef();

    const address = useRef();

    const contactPerson = useRef();

    const dispatch = useDispatch();

    const [clients, setClients] = useState([]);

    const getClients = useCallback(() => {

        dispatch(getClientsStart());

        get().then((data) => {

            setClients(data.result);

            dispatch(getClientsSuccess(data));

        }).catch((error) => {

            dispatch(getClientsFailure(error));

        });

    }, []);

const removeClient = useCallback((e) => {

        dispatch(deleteClientStart());

        deleteClient(e.target.closest('button').dataset.id).then((data) => {

            getClients();

            dispatch(deleteClientSuccess(data));

        }).catch((error) => {

            dispatch(deleteClientFailure(error));

        });

    }, []);

    const submitForm = useCallback((e) => {

        e.preventDefault();

        dispatch(addClientStart());

        addClient({

            kindOfProperty: kindOfProperty.current.value,

            name: name.current.value,

            contactPerson: contactPerson.current.value,

            address: address.current.value

        }).then((data) => {

            getClients();

            dispatch(addClientSuccess(data));

        }).catch((error) => {

            dispatch(addClientFailure(error));

        });

    }, []);

    return (

        <div className="Home">

            <button id='fetch\_button' onClick={getClients}>Fetch all clients</button>

            {

                !!clients.length && <table>

                    <thead>

                        <tr>

                            <th>name</th>

                            <th>kindOfProperty</th>

                            <th>address</th>

                            <th>contactPerson</th>

                        </tr>

                    </thead>

                    <tbody>{clients.map((client) => <tr key={client.id}>

                        <td>{client.name}</td>

                        <td>{client.kindOfProperty}</td>

                        <td>{client.address}</td>

                        <td>{client.contactPerson}</td>

                        <td><button data-id={client.id} onClick={removeClient}><svg width="24" height="24" xmlns="http://www.w3.org/2000/svg" fillRule="evenodd" clipRule="evenodd"><path d="M19 24h-14c-1.104 0-2-.896-2-2v-17h-1v-2h6v-1.5c0-.827.673-1.5 1.5-1.5h5c.825 0 1.5.671 1.5 1.5v1.5h6v2h-1v17c0 1.104-.896 2-2 2zm0-19h-14v16.5c0 .276.224.5.5.5h13c.276 0 .5-.224.5-.5v-16.5zm-9 4c0-.552-.448-1-1-1s-1 .448-1 1v9c0 .552.448 1 1 1s1-.448 1-1v-9zm6 0c0-.552-.448-1-1-1s-1 .448-1 1v9c0 .552.448 1 1 1s1-.448 1-1v-9zm-2-7h-4v1h4v-1z"/>

      </svg>

</button></td>

                    </tr>)}

                    </tbody>

                </table>

            }

            <form className="form flex flex-direction\_column" onSubmit={submitForm}>

                <label htmlFor="name">Name</label>

                <input ref={name} id="name" type="text" required />

                <label htmlFor="kindOfProperty">Kind of property</label>

                <input ref={kindOfProperty} id="kindOfProperty" type="text" required />

                <label htmlFor="address">Address</label>

                <input ref={address} id="address" type="text" required />

                <label htmlFor="contactPerson">Contact person</label>

                <input ref={contactPerson} id="contactPerson" type="telephone" required />

                <button type="submit">Submit</button>

            </form>

        </div>

    );

}

export default Home;

Home.css

.Home {

    display: flex;

    flex-direction: column;

    align-items: center;

    padding: 30px;

}

.flex{

    display: flex;

}

.form {

    width: fit-content;

    border: #ecdcfb 1px solid;

    padding: 1rem;

    border-radius: 10px;

    display: grid;

    gap: 5px;

    position: sticky;

    top: 20px;

}

#fetch\_button {

    padding: 10px 15px;

    background-color: darkmagenta;

    color: white;

    font-size: 15px;

    border: 3px solid blueviolet;

    border-radius: 10%;

    font-weight: bold;

}

table {

    margin-top: 30px;

}

table,

th,

td {

    border: 1px solid blueviolet;

    border-collapse: collapse;

}

td {

    padding: 10px;

}

th {

    padding: 15px;

}

services/api/addClient

import axios from "axios";

export default async function addClient(client){

    const response = await axios.post("http://localhost:8080/api/clients/register", client);

    return response.data;

}

services/api/deleteClient

import axios from "axios";

export default async function deleteClient(id){

    console.log(id);

    const response = await axios.delete(`http://localhost:8080/api/clients/${id}`);

    return response.data;

}

services/api/get

import axios from "axios";

export default async function get(){

    const response = await axios.get("http://localhost:8080/api/clients");

    return response.data;

}

store/constants.js

export const CLIENTS\_GET\_START = 'CLIENTS\_GET\_START';

export const CLIENTS\_GET\_SUCCESS = 'CLIENTS\_GET\_SUCCESS';

export const CLIENTS\_GET\_FAILURE = 'CLIENTS\_GET\_FAILURE';

export const CLIENT\_ADD\_START = 'CLIENT\_ADD\_START';

export const CLIENT\_ADD\_SUCCESS = 'CLIENT\_ADD\_SUCCESS';

export const CLIENT\_ADD\_FAILURE = 'CLIENT\_ADD\_FAILURE';

export const CLIENT\_DELETE\_START = 'CLIENT\_DELETE\_START';

export const CLIENT\_DELETE\_SUCCESS = 'CLIENT\_DELETE\_SUCCESS';

export const CLIENT\_DELETE\_FAILURE = 'CLIENT\_DELETE\_FAILURE';

store/actions/clients.ts

import { Client } from '../../types'

import { CLIENTS\_GET\_FAILURE, CLIENTS\_GET\_START, CLIENTS\_GET\_SUCCESS,

    CLIENT\_ADD\_START, CLIENT\_ADD\_SUCCESS, CLIENT\_ADD\_FAILURE,

    CLIENT\_DELETE\_START, CLIENT\_DELETE\_SUCCESS, CLIENT\_DELETE\_FAILURE } from '../constants'

export function getClientsStart(): { type: typeof CLIENTS\_GET\_START } {

    return {

        type: CLIENTS\_GET\_START

    }

}

export function getClientsSuccess(clients: Client[]): { type: typeof CLIENTS\_GET\_SUCCESS, clients:Client[] } {

    return {

        clients,

        type: CLIENTS\_GET\_SUCCESS

    }

}

export function getClientsFailure(error: Error): { type: typeof CLIENTS\_GET\_FAILURE, error:Error } {

    return {

        error,

        type: CLIENTS\_GET\_FAILURE

    }

}

export function addClientStart(): { type: typeof CLIENT\_ADD\_START } {

    return {

        type: CLIENT\_ADD\_START

    }

}

export function addClientSuccess(clients: Client[]): { type: typeof CLIENT\_ADD\_SUCCESS, clients:Client[] } {

    return {

        clients,

        type: CLIENT\_ADD\_SUCCESS

    }

}

export function addClientFailure(error: Error): { type: typeof CLIENT\_ADD\_FAILURE, error:Error } {

    return {

        error,

        type: CLIENT\_ADD\_FAILURE

    }

}

export function deleteClientStart(): { type: typeof CLIENT\_DELETE\_START } {

    return {

        type: CLIENT\_DELETE\_START

    }

}

export function deleteClientSuccess(clients: Client[]): { type: typeof CLIENT\_DELETE\_SUCCESS, clients:Client[] } {

    return {

        clients,

        type: CLIENT\_DELETE\_SUCCESS

    }

}

export function deleteClientFailure(error: Error): { type: typeof CLIENT\_DELETE\_FAILURE, error:Error } {

    return {

        error,

        type: CLIENT\_DELETE\_FAILURE

    }

}

store/reducers/clients.ts

import { Client } from "../../types";

import { getClientsFailure, getClientsStart, getClientsSuccess,

    addClientStart, addClientSuccess, addClientFailure,

    deleteClientStart, deleteClientFailure, deleteClientSuccess

} from "../actions/clients";

import { CLIENTS\_GET\_FAILURE, CLIENTS\_GET\_START, CLIENTS\_GET\_SUCCESS, CLIENT\_ADD\_START, CLIENT\_ADD\_SUCCESS, CLIENT\_ADD\_FAILURE, CLIENT\_DELETE\_START, CLIENT\_DELETE\_FAILURE, CLIENT\_DELETE\_SUCCESS } from "../constants";

export interface ClientsState {

    error: Error | null;

    isLoading: boolean;

    clients: Client[];

}

export default function clientsReducer(

    state: ClientsState = { error: null, isLoading: false, clients: [] },

    action: ReturnType<typeof getClientsStart>

        | ReturnType<typeof getClientsSuccess>

        | ReturnType<typeof getClientsFailure>

        | ReturnType<typeof addClientStart>

        | ReturnType<typeof addClientSuccess>

        | ReturnType<typeof addClientFailure>

        | ReturnType<typeof deleteClientStart>

        | ReturnType<typeof deleteClientSuccess>

        | ReturnType<typeof deleteClientFailure>) {

    switch (action.type) {

        case CLIENTS\_GET\_START: return { ...state, isLoading: true };

        case CLIENTS\_GET\_SUCCESS: return { ...state, error: null, isLoading: false, clients: action.clients };

        case CLIENTS\_GET\_FAILURE: return { ...state, error: action.error, isLoading: false, clients: [] };

        case CLIENT\_ADD\_START: return { ...state, isLoading: true };

        case CLIENT\_ADD\_SUCCESS: return { ...state, error: null, isLoading: false, clients: action.clients };

        case CLIENT\_ADD\_FAILURE: return { ...state, error: action.error, isLoading: false, clients: [] };

        case CLIENT\_DELETE\_START: return { ...state, isLoading: true };

        case CLIENT\_DELETE\_SUCCESS: return { ...state, error: null, isLoading: false, clients: action.clients };

        case CLIENT\_DELETE\_FAILURE: return { ...state, error: action.error, isLoading: false, clients: [] };

    }

}

store/reducers/index.ts

import { combineReducers } from "redux";

import clientsReducer from "./clients";

const reducer = combineReducers({ clients: clientsReducer });

export default reducer;

store/index.ts

import { createStore } from "redux";

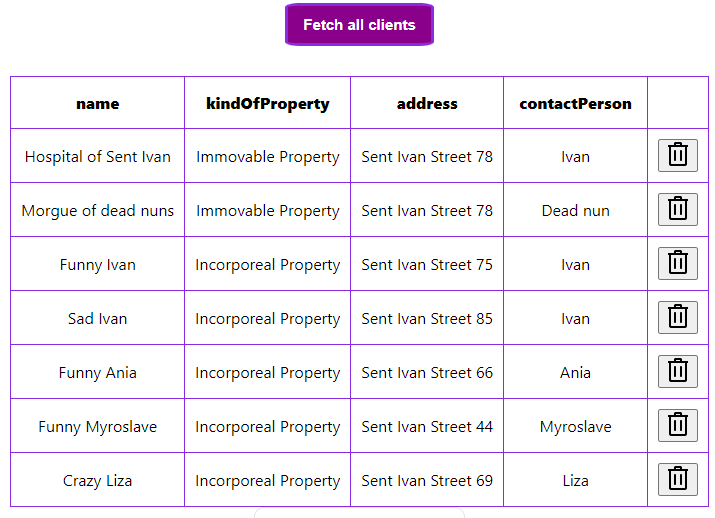
import reducer from "./reducers";

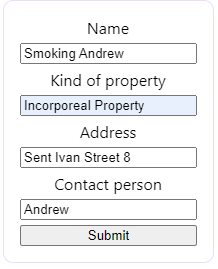
const store = createStore(reducer);

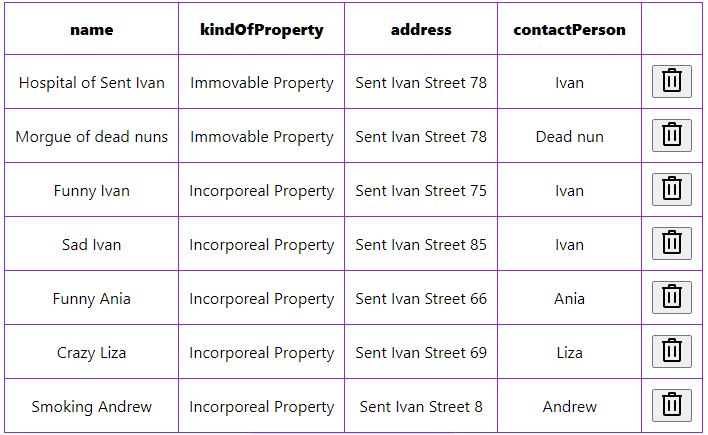
export default store;

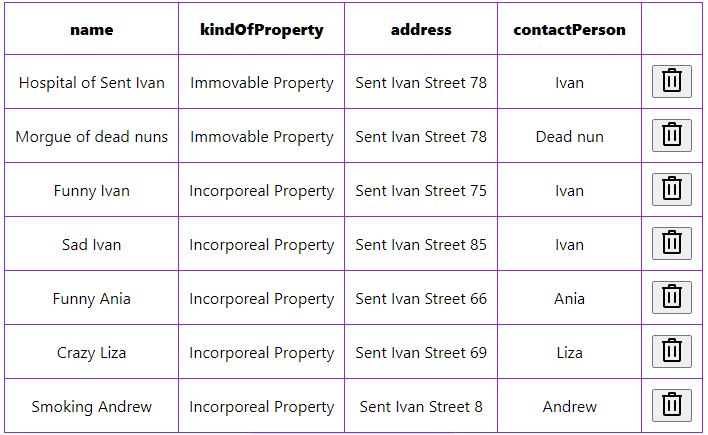
Результат:











**Висновок:** Навчилась працювати з Redux.