/\* eslint-disable no-unused-vars \*/

import React from "react";

import styles from "../searchBar/SearchBar.module.css";

import { useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import { getRecipesName, getRecipesAll } from "../../redux/action";

import img from "../assets/food-logo.png";

function SearchBar({ setCurrentPage }) {

  const dispatch = useDispatch();

  const [name, setName] = useState("");

  function handleInput(evt) {

    evt.preventDefault();

    setName(evt.target.value);

    setCurrentPage(1);

    // dispatch(getRecipesAll(evt.target.value));

  }

  function handleSubmit(evt) {

    evt.preventDefault();

    if (name) {

      dispatch(getRecipesName(name));

      setName("");

    }

  }

  return (

    <div>

      <div>

        <img className={styles.imgLogo} src={img} alt="Img Not Found"></img>

      </div>

      <input

        type="text"

        placeholder="Search Name ..."

        onChange={(evt) => handleInput(evt)}

      />

      <button

        className={styles.searchButton}

        type="submit"

        onClick={(evt) => handleSubmit(evt)}

      >

        Search

      </button>

    </div>

  );

}

export { SearchBar };

action}

import axios from "axios";

export const GET\_RECIPES\_NAME = "GET\_RECIPES\_NAME";

export const GET\_RECIPES\_ID = "GET\_RECIPES\_ID";

export const GET\_TYPES = "GET\_TYPES";

export const GET\_RECIPES = "GET\_RECIPES";

export const FILTER\_BY\_DIETS = "FILTER\_BY\_DIETS";

export const SET\_DEFAULT\_CARD = "SET\_DEFAULT\_CARD";

export const FILTER\_BY\_RESOURCES = "FILTER\_BY\_RESOURCES";

export const FILTER\_BY\_ORDER = "FILTER\_BY\_ORDER";

export const FILTER\_BY\_SEARCHBAR = "FILTER\_BY\_SEARCHBAR";

export const ORDER\_BY\_SCORE = "ORDER\_BY\_SCORE";

export const POST\_RECIPES = "POST\_RECIPES";

export const GET\_DATABASE = "GET\_DATABASE";

export const GET\_STATE\_ID = "GET\_STATE\_ID";

const URL = "http://localhost:3001";

export function getRecipesAll() {

  return function (dispatch) {

    axios

      .get(`${URL}/recipes`)

      .then((json) => {

        return dispatch({

          type: GET\_RECIPES,

          payload: json.data,

        });

      })

      .catch((error) => {

        console.log(error.message);

      });

  };

}

export function getRecipesName(name) {

  return async function (dispatch) {

    try {

      const recipes = await axios.get(`${URL}/recipes?name=${name}`);

      console.log(recipes.data);

      return dispatch({

        type: GET\_RECIPES\_NAME,

        payload: recipes.data,

      });

    } catch (error) {

      console.log(error);

    }

  };

}

export function postRecipes(info) {

  return async function (dispatch) {

    try {

      const response = await axios.post(`${URL}/recipes`, info);

      console.log(response);

      return dispatch({

        type: POST\_RECIPES,

        payload: response.data,

      });

    } catch (error) {

      console.log(error.message);

    }

  };

}

export function getRecipesId(id) {

  return async function (dispatch) {

    try {

      const response = await axios.get(`${URL}/recipes/${id}`);

      console.log(response);

      return dispatch({

        type: GET\_RECIPES\_ID,

        payload: response.data,

      });

    } catch (err) {

      console.log(err);

    }

  };

}

export function getTypes() {

  return function (dispatch) {

    try {

      axios.get(`${URL}/diets`).then((types) =>

        dispatch({

          type: GET\_TYPES,

          payload: types.data,

        })

      );

    } catch (error) {

      console.log(error);

    }

  };

}

export function getDatabase() {

  return async function (dispatch) {

    try {

      let dataBase = await axios.get(`${URL}/recipes/dates`);

      return dispatch({

        type: GET\_DATABASE,

        payload: dataBase.data,

      });

    } catch (error) {

      console.log(error);

    }

  };

}

export function getFilterByDiets(payload) {

  return {

    type: FILTER\_BY\_DIETS,

    payload,

  };

}

export function setDefaultCards() {

  return {

    type: SET\_DEFAULT\_CARD,

  };

}

export function filterByResources(payload) {

  return {

    type: FILTER\_BY\_RESOURCES,

    payload: payload,

  };

}

export function filterByOrder(payload) {

  return {

    type: FILTER\_BY\_ORDER,

    payload: payload,

  };

}

export function orderByScore(payload) {

  return {

    type: ORDER\_BY\_SCORE,

    payload: payload,

  };

}

export function searchBarName(payload) {

  return {

    type: FILTER\_BY\_SEARCHBAR,

    payload: payload,

  };

}

export function searchId(payload) {

  return {

    type: GET\_STATE\_ID,

    payload: payload,

  };

}

import {

  GET\_RECIPES,

  GET\_RECIPES\_NAME,

  GET\_RECIPES\_ID,

  GET\_TYPES,

  FILTER\_BY\_DIETS,

  SET\_DEFAULT\_CARD,

  FILTER\_BY\_RESOURCES,

  FILTER\_BY\_ORDER,

  FILTER\_BY\_SEARCHBAR,

  ORDER\_BY\_SCORE,

  POST\_RECIPES,

  GET\_DATABASE,

  GET\_STATE\_ID,

} from "../redux/action";

const inicialState = {

  recipes: [],

  recipesAll: [],

  types: [],

  detail: [],

  addReset: [],

};

const rootReducer = (state = inicialState, action) => {

  switch (action.type) {

    case GET\_RECIPES:

      return {

        ...state,

        recipes: action.payload,

        recipesAll: action.payload,

      };

    case GET\_RECIPES\_NAME:

      const addRecipe = state.recipesAll;

      const nameRecipes = addRecipe.map((recipe) => recipe.name);

      const newRecipeAdd = action.payload?.map((recipe) => {

        if (!nameRecipes.includes(recipe.name)) {

          addRecipe.push(recipe);

        }

        return addRecipe;

      });

      return {

        ...state,

        recipes: action.payload,

        // recipesAll: newRecipeAdd,

      };

    case GET\_RECIPES\_ID:

      return {

        ...state,

        detail: action.payload,

      };

    case POST\_RECIPES:

      return {

        ...state,

        addReset: action.payload,

      };

    case GET\_TYPES:

      return {

        ...state,

        types: action.payload,

      };

    case FILTER\_BY\_DIETS:

      const all = state.recipesAll;

      const getFilterByDiets =

        action.payload === "all"

          ? all

          : all.filter((recipe) => {

              if (recipe.diets.length > 0)

                if (recipe.diets.find((element) => element === action.payload))

                  return recipe;

            });

      return {

        ...state,

        recipes: getFilterByDiets,

      };

    case SET\_DEFAULT\_CARD:

      const recipes = state.recipesAll;

      return {

        ...state,

        recipes: recipes,

      };

    case GET\_DATABASE:

      const datBase = state.recipesAll.filter((recipe) => {

        if (recipe.hasOwnProperty("idApi")) return recipe;

      });

      const joinAll = datBase.concat(action.payload);

      return {

        ...state,

        dates: action.payload,

        recipes: joinAll,

        recipesAll: joinAll,

      };

    case FILTER\_BY\_RESOURCES:

      const { id } = action;

      const allRecipes1 = state.recipesAll;

      const statusFiltered2 =

        action.payload === "Filter by Source"

          ? allRecipes1.filter((el) => typeof el.id === "string")

          : allRecipes1.filter((el) => typeof el.id !== "number");

      return {

        ...state,

        recipes:

          action.payload === "api"

            ? allRecipes1.filter((el) => typeof el.id === "number")

            : statusFiltered2,

      };

    case FILTER\_BY\_ORDER:

      const recypesByOrder =

        action.payload === "up"

          ? state.recipesAll.sort((a, b) => {

              if (a.title.toLowerCase() > b.title.toLowerCase()) return 1;

              else return -1;

            })

          : state.recipesAll.sort((a, b) => {

              if (a.title.toLowerCase() < b.title.toLowerCase()) return 1;

              else return -1;

            });

      return {

        ...state,

        recipes: recypesByOrder,

      };

    case ORDER\_BY\_SCORE:

      const recypesByScore =

        action.payload === "SSc"

          ? state.recipesAll.sort((a, b) => {

              if (a.score - b.score < 0) return 1;

              else return -1;

            })

          : state.recipesAll.sort((a, b) => {

              if (a.healthScore - b.healthScore < 0) return 1;

              else return -1;

            });

      return {

        ...state,

        recipes: recypesByScore,

      };

    case FILTER\_BY\_SEARCHBAR:

      const filtSearch = state.recipesAll;

      const filtOnState = filtSearch.filter((recipe) => {

        let name = recipe.title.toLowerCase();

        if (name.includes(action.payload)) return recipe;

      });

      return {

        ...state,

        recipes: filtOnState,

      };

    case GET\_STATE\_ID:

      const filtId = state.recipesAll;

      const Idfind = filtId.find((recipe) => {

        if (typeof action.payload === "number") {

          if (recipe.idApi === action.payload) return recipe;

        } else {

          if (recipe.id === action.payload) return recipe;

        }

      });

      return {

        ...state,

        detail: Idfind,

      };

    default:

      return state;

  }

};

export default rootReducer;

store

import { createStore, applyMiddleware } from "redux";

import thunk from "redux-thunk";

import { composeWithDevTools } from "redux-devtools-extension";

import rootReducer from "../redux/reducer";

export const store = createStore(

  rootReducer,

  composeWithDevTools(applyMiddleware(thunk))

);

App import "./App.css";

import { BrowserRouter, Route, Switch } from "react-router-dom";

import LandingPage from "./components/landingPage/LandingPage";

import Home from "./components/home/Home.jsx";

import Form from "./components/form/Form.jsx";

import Detail from "./components/detail/Detail.jsx";

import NotFound from "./components/error404/Error404";

function App() {

  return (

    <BrowserRouter>

      <div className="App">

        <Switch>

          <Route exact path="/" component={LandingPage} />

          <Route path="/home" component={Home} />

          <Route path="/recipes" component={Form} />

          <Route path="/detail/:id" component={Detail} />

          <Route component={() => <NotFound />} />

        </Switch>

      </div>

    </BrowserRouter>

  );

}

export default App;