Serchbar

import React, { useState } from "react";

import style from "./SearchBar.module.css";

function SearchBar({ onSearch }) {

  const [characterId, setCharacterId] = useState("");

  const handleCase = (event) => {

    setCharacterId(event.target.value);

  };

  const handleRandomCharacter = () => {

    const randomId = Math.floor(Math.random() \* 826) + 1;

    onSearch(randomId);

    setCharacterId("");

  };

  return (

    <div className={style.SearchBar}>

      <>

        <input

          type="search"

          value={characterId}

          onChange={handleCase}

          className={style.border}

          placeholder="Search..."

        />

        <button onClick={() => onSearch(characterId)} className={style.btn}>

          Add

        </button>

        <button

          onClick={() => handleRandomCharacter(characterId)}

          className={style.btn}

        >

          Random

        </button>

      </>

    </div>

  );

}

export default SearchBar;

import { Link } from "react-router-dom";

import style from "./Card.module.css";

import { connect } from "react-redux";

import { addFavorite, removeFavorite } from "../../redux/action";

import { useState, useEffect } from "react";

function Card(props) {

  console.log(props);

  const {

    id,

    name,

    species,

    gender,

    image,

    onClose,

    detailId,

    addFavorite,

    removeFavorite,

    myFavorites,

  } = props;

  const [isFav, setIsFav] = useState(false);

  useEffect(() => {

    myFavorites.forEach((fav) => {

      if (fav.id === id) {

        setIsFav(true);

        //console.log(isFav);

      }

    });

  }, [myFavorites]);

  const handleFavorite = () => {

    if (isFav) {

      setIsFav(false);

      removeFavorite(id);

    } else {

      setIsFav(true);

      addFavorite({

        id,

        name,

        species,

        gender,

        image,

        onClose,

        detailId,

        addFavorite,

        removeFavorite,

      });

    }

  };

  return (

    <div className={style.container}>

      {isFav ? (

        <button

          onClick={() => {

            handleFavorite(id);

          }}

        >

          ❤️

        </button>

      ) : (

        <button

          onClick={() => {

            handleFavorite();

          }}

        >

          🤍

        </button>

      )}

      <div className={style.back}>

        <button onClick={onClose} className={style.btnDelete}>

          X

        </button>

        <Link to={`/detail/${detailId}`}>

          <h2 className={style.title}> {name}</h2>

          <h2 className={style.title}>{gender}</h2>

          <img src={image} alt={name} />

          <h2 className={style.title}>{species}</h2>

        </Link>

      </div>

    </div>

  );

}

const mapDispatchToProps = (dispatch) => {

  return {

    addFavorite: (character) => {

      dispatch(addFavorite(character));

    },

    removeFavorite: (id) => {

      dispatch(removeFavorite(id));

    },

  };

};

const mapStateToProps = (state) => {

  return {

    myFavorites: state.myFavorites,

  };

};

export default connect(mapStateToProps, mapDispatchToProps)(Card);

action

import axios from "axios";

export const ADD\_FAVORITE = "ADD\_FAVORITE";

export const REMOVE\_FAVORITE = "REMOVE\_FAVORITE";

export const FILTER = "FILTER";

export const ORDER = "ORDER";

export const addFavorite = (character) => {

  const endpoint = "http://localhost:3001/rickandmorty/fav";

  return (dispatch) => {

    try {

      axios.post(endpoint, character).then(({ data }) => {

        return dispatch({

          type: "ADD\_FAVORITE",

          payload: data,

        });

      });

    } catch (error) {

      return error.message;

    }

  };

};

export const removeFavorite = (id) => {

  const endpoint = "http://localhost:3001/rickandmorty/fav/" + id;

  return (dispatch) => {

    try {

      axios.delete(endpoint).then(({ data }) => {

        return dispatch({

          type: "REMOVE\_FAVORITE",

          payload: data,

        });

      });

    } catch (error) {

      return error.message;

    }

  };

};

// const endpoint = "http://localhost:3001/rickandmorty/fav";

// export const addFavorite = (character) => {

//   return (dispatch) => {

//     axios.post(endpoint, character).then(({ data }) => {

//       return dispatch({

//         type: "ADD\_FAVORITE",

//         payload: data,

//       });

//     });

//   };

// };

// export const removeFavorite = (id) => {

//   return (dispatch) => {

//     axios.delete(`${endpoint}/${id}`).then(({ data }) => {

//       return dispatch({

//         type: "REMOVE\_FAVORITE",

//         payload: data,

//       });

//     });

//   };

// };

export const filterCards = (gender) => {

  return { type: FILTER, payload: gender };

};

export const orderCards = (id) => {

  return { type: ORDER, payload: id };

};

import { ADD\_FAVORITE, REMOVE\_FAVORITE, FILTER, ORDER } from "./action";

const initialState = {

  myFavorites: [],

  allCharacters: [],

};

const rootReducer = (state = initialState, { type, payload }) => {

  switch (type) {

    case ADD\_FAVORITE:

      return { ...state, myFavorites: payload, allCharacters: payload };

    case REMOVE\_FAVORITE: {

      return {

        ...state,

        myFavorites: payload,

        allCharacters: payload,

      };

    }

    case FILTER:

      const allCharsFiltered = state.allCharacters.filter(

        (char) => char.gender === payload

      );

      return {

        ...state,

        myFavorites: allCharsFiltered,

      };

    case ORDER: {

      if (payload === "Ascendente") {

        return {

          ...state,

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

            return a.id - b.id;

          }),

        };

      } else {

        return {

          ...state,

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

            return b.id - a.id;

          }),

        };

      }

    }

    default:

      return { ...state };

  }

};

export default rootReducer;

import { createStore, applyMiddleware, compose } from "redux";

import rootReducer from "./reducer";

import ThunkMiddleware from "redux-thunk";

const composeEnhancer = window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

const store = createStore(

  rootReducer,

  composeEnhancer(applyMiddleware(ThunkMiddleware))

);

export default store;