**SERVER**

SERVER CON WEBSERVER

const http = require("http");

const getCharById = require("./controllers/getCharById.js");

const PORT = 3001;

http.createServer((req, res) => {

    //Core...Setea el header, y le permite acceso total a cualquier url

    res.setHeader('Access-Control-Allow-Origin', '\*');

    if(req.url.includes("/rickandmorty/character")){

        const id = req.url.split("/").pop();

        getCharById(res, id);

    }

}).listen(PORT, "localhost");

//prueba del server, para saber si me trae los chars

//http://localhost:3001/rickandmorty/character/444

/\*

if(req.url.includes("/rickandmorty/character")){

    //const id = req.url.split("/").pop().at(-1);  empieza al final

    const id = req.url.split("/").pop();

    const character = characters.filter (char => char.id === Number(id));  //tambien pude usar .pop, .shift

    //character: [ {id:1}]

    //char.id tipo: number,  id tipo:string

    res

    .writeHead(200, {"content-type": "application-json"})

    .end(JSON.stringify(character[0]));

    //character es un array de objetos, le paso el [0] para que llegue como objeto al front

    //asi no paso todo el personaje con un arreglo sino solo ese personaje como obj

    //req.url= localhos:3001/rickandmorty/character/4

    //   split [localhos:3001, rickandmorty, character, 4]

}

\*/

**CONTROLLERS**

GetCharById USANDO WEBSERVER

const axios = require("axios");

const getCharById = (res, id) => {

    axios.get(`https://rickandmortyapi.com/api/character/${id}`)

    .then(response => response.data)

    .then(data => {

        const character = {

            id: data.id,   //id: id?

            name: data.name,

            gender: data.gender,

            species: data.species,

            origin: data.origin,

            image: data.image,

            status: data.status

        };

        //El response no hace falta transformarlo a JSON pq estoy usando axios, si usara fetch si habria que transformarlo

        //seteamos el Header

        res

        .writeHead(200, {"Content-type": "application/json"})

        .end(JSON.stringify(character));

    })

    .catch((error) =>

    res

    .writeHead(500, {"Content-type": "text/plain"})

    .end(`Error 500. Character with id ${id} not found`) //error.message

    //tambien lo puedo hacer de esta manera:

    //.end({"message": "Error"})    como(JSON)

    //Desde el front puedo chequear si me llega un objeto con propiedad message,

    //es que hubo un error en el back al buscar el char

    );

}

module.exports = getCharById;

GetCharById USANDO PROMESAS

//Promsesas

/\*

module.exports = (req, res) => {

    const {id} = req.params;

    axios.get(URL + id)

    .then(response => {

        const { status, name, gender, species, origin, image } = response.data;

        const character = {

                id, name, gender, species, origin, image, status

            }

            return character.name

            ? res.status(200).json(character)

            : res.status(404).send("Not found")

            })

    .catch (error => {

        return res.status(500).send(error.message);

    })

};

\*/

CARPETA utils ARCHIVO users.js (eliminado por cambios en el server)

module.exports = [

    {email: "ejemplo@gmail.com", password: "123456"}

]

ARCHIVO controllers/handleFavorites.js (eliminado por cambios en el server)

let myFavorites = [];  // [ {id:1}, {id:3}, ...]

const postFav = (req, res) => {

    myFavorites.push(req.body);

    return res.status(200).json(myFavorites); //envio un  [ favorites ]

    //return para que JS no siga leyendo

}

const deleteFav = (req, res) => {

    const { id } = req.params;

    myFavorites = myFavorites.filter( charFav => charFav.id !== id );

    //charFav.id llega como string, el igual que id

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

}

module.exports = {

    postFav,

    deleteFav

}

ARCHIVO controllers/login.js (eliminado por cambios en el server)

const users = require("../utils/users.js"); //[ {-} ]

module.exports = (req, res) => {

    const {email, password} = req.query;

    let access = false;

    users.forEach((user) => {

        //if(user.email === email && user.password === password) access = true;

        user.email === email && user.password === password

        ? access = true

        : null;  //window.alert("");  //se ejecuta el alert del front, en la funcion login

    })

    return res.status(200).json({ access }); // envio  {access: true/false}

}

**CLIENT**

FUNCION LOGIN SIN USAR PROMESAS

//funcion login sin usar promesas

   /\*

   function onSubmit(userData) { //recibe el objeto con el input del login

      if (userData.password === PASSWORD && userData.email === EMAIL) {

         setAccess(true);

         navigate("/home"); //redirije a la pagina home

      }else {

         alert("Wrong email or password");

      }

   }

   \*/

FUNCION LOGIN USANDO PROMESAS

   //Usando Promises

   /\*

   function login(userData) {  //onSubmit

      const { email, password } = userData;

      const URL = 'http://localhost:3001/rickandmorty/login/';

      axios(URL + `?email=${email}&password=${password}`).then(({ data }) => {

         const { access } = data; //data= {access: false}

         setAccess(data); //access = true o false

         access && navigate('/home');

      });

   }

   \*/

OnSearch USANDO PROMESAS (SUSTITUÍ POR ASYNCAWAIT)

//Usando Promises

   /\*

   function onSearch(id) {

      axios(`http://localhost:3001/rickandmorty/character/${id}`).then(({ data }) => {

         //console.log(data); //objeto con los datos del personaje

         //console.log(id); //2

         if (!characters.some( character => character.id === data.id )){

            if (data.name) {

                  setCharacters((characters) => [...characters, data]);

            } else {

               window.alert('¡There are no characters with this ID!');

               }

            } else {

               window.alert('¡There is already a character with this ID!');

            }

      }).catch( error => {

         window.alert("¡The character with the entered ID does not exist!")

      })

   }

   \*/

ACTIONS

AddFav SIN PROMESAS

/\*

export const addFav = (character) => { //retorna unobjeto con la info del personaje

    return {

        type: ADD\_FAV,

        payload: character

    }

}

\*/

AddFav USANDO PROMESAS

/\*

export const addFav = (character) => {

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

    return (dispatch) => {

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

          return dispatch({

             type: 'ADD\_FAV',

             payload: data,

          });

       });

    };

 };

\*/

AddFav antes de aplicar express (mas reciente)

// case ADD\_FAV:

        //     return{

        //         ...state,

        //         myFavorites: [...state.allCharacters, action.payload],

        //         allCharacters: [...state.allCharacters, action.payload]   //para no ir pisando mientras filtremos

        //     }

RemoveFav SIN PROMESAS

 /\*

export const removeFav = (id) => {

    return{

        type: REMOVE\_FAV,

        payload: id

    }

}

\*/

RemoveFav USANDO PROMESAS

/\*

export const removeFav = (id) => {

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

    return (dispatch) => {

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

          return dispatch({

             type: 'REMOVE\_FAV',

             payload: data,

       });

       });

    };

 };

\*/

RemoveFav antes de aplicar express (más reciente)

// case REMOVE\_FAV:

        //     const filteredFavs = state.allCharacters.filter((favorite) => favorite.id !== Number(action.payload));

        //     return{

        //         ...state,

        //         //myFavorites: state.myFavorites.filter( (character) => character.id !== Number(action.payload) )

        //         myFavorites: filteredFavs,

        //         allCharacters: filteredFavs  //Elimina tanto en favoritos como en todos los cards

        //     }