*(ReactJS chỉ xử lý ở tầng VIEW (render ra giao diện bên ngoài))*

*(Các thiết lập đầu tiên nên làm : aliases)*

**=== Install ===**

npm install jquery popper.js bootstrap@4.6.1 --save

index.js: bootstrap.min.css > index.css > (scss) > App > jquery > popper.js > bootstrap.js

**=== Xây dựng cơ bản cấu trúc HomeTemplate ===**

(src/containers/HomeTemplate/\_components (NavbarHome, FooterHome, ...), HomePage, About Page, ProductsPage(Product),PageNotFound, ...)

**=== Xây dựng bộ định tuyến, tạo đường dẫn đến các page vừa tạo ===**

npm install react-router-dom@5.3.0 --save

- Sử dụng Route của react-router-dom để định tuyến cho mỗi đường dẫn, cấu trúc cơ bản cho mỗi Route:

<*Route* *path*="/" *component*={HomePage} />

- Sử dụng BrowserRouter của react-router-dom để định nghĩa routing, bao bọc nội dung trong App.js (kèm theo cặp thẻ <Switch> của react-router-dom để định tuyến cho đường dẫn mà end-user nhập sai sẽ dẫn về trang Page Not Found – đặt <Route> của Page Not Found ở cuối thẻ <Switch>)

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

import AboutPage from "./containers/HomeTemplate/AboutPage";

import HomePage from "./containers/HomeTemplate/HomePage";

import PageNotFound from "./containers/HomeTemplate/PageNotFound";

import ListMoviesPage from "./containers/HomeTemplate/ListMoviesPage";

function App() {

  return (

    <*BrowserRouter*>

      <*Switch*>

        <*Route* *exact*={true} *path*="/" *component*={HomePage} />

        <*Route* *path*="/about" *component*={AboutPage} />

        <*Route* *path*="/list-movies" *component*={ListMoviesPage} />

        <*Route* *path*="" *component*={PageNotFound} />

      </*Switch*>

    </*BrowserRouter*>

  );

}

export default App;

- Phát triển web có thể rất nhiều trang => phát sinh nhiều thẻ <Route> => tạo thư mục src/routes để quản lý bộ định tuyến, bao gồm nhiều array quản lý thuộc tính exact, path, component cho từng Template.

- Thay vì duyệt từng mảng array để render ra nhiều thẻ <Route>, ta duyệt mảng và render ra theo từng Template component (truyền props vào Template component trong đó chứa thẻ <Route> và các component chung như Navbar, Footer, ...) để phân luồng cho từng Template

import AboutPage from "../containers/HomeTemplate/AboutPage";

import HomePage from "../containers/HomeTemplate/HomePage";

import HomeTemplate from "../containers/HomeTemplate";

import ListMoviesPage from "../containers/HomeTemplate/ListMoviesPage";

// import { Route } from "react-router-dom";

const routesHome = [

  {

    exact: true,

    path: "/",

    component: HomePage

  },

  {

    exact: false,

    path: "/about",

    component: AboutPage

  },

  {

    exact: false,

    path: "/list-movies",

    component: ListMoviesPage

  }

]

// export const renderRoutesHome = () => {

//   return routesHome.map((route, index) => {

//     return <Route

//       key={index}

//       exact={route.exact}

//       path={route.path}

//       component={route.component}

//     />

//   })

// }

// Thay vì duyệt mảng và return về các thẻ <Route/>, ta return về component <HomeTemplate/> để phân luồng với các template sau này, đồng thời truyền props vào

export const renderRoutesHome = () => {

  return routesHome.map((*route*, *index*) => {

    return <*HomeTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

Trong component HomeTemplate return về:

import React, { Fragment } from 'react';

import { Route } from 'react-router-dom';

import NavbarHome from './\_components/Navbar';

export default function HomeTemplate(*props*) {

  const { exact, path, component } = *props*;

  return (

    <*Fragment*>

      <*NavbarHome* />

      <*Route* *exact*={exact} *path*={path} *component*={component} />

    </*Fragment*>

  )

}

Khi đó App.js được viết lại:

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

import PageNotFound from "./containers/HomeTemplate/PageNotFound";

import { renderRoutesHome } from "./routes";

function App() {

  return (

    <*BrowserRouter*>

      <*Switch*>

        {renderRoutesHome()}

        <*Route* *path*="" *component*={PageNotFound} />

      </*Switch*>

    </*BrowserRouter*>

  );

}

export default App;

(Lỗi chưa xử lý: VD khi vào path localhost:3000/about/abc => không về trang PageNotFound)

**=== Xây dựng Navbar (Footer) cho HomeTemplate, các thẻ <a> liên kết với đường dẫn vừa tạo ===**

- Tạo nhanh bằng navbar của bootstrap. Tuy nhiên phải thay toàn bộ thẻ <a> (có thuộc tính href làm refresh lại page => phá vỡ cấu trúc SPA) bằng Link/NavLink của react-router-dom; thay thế href='...' bằng to='...'

(NavLink có props là activeClassName để đặt tên class khi thẻ <li> đang được active (thêm props exact ở thẻ <li> đầu tiên để tránh nhầm lẫn cho các case tiếp theo - tương tự lúc khai báo bộ định tuyến)

import React from 'react';

import { Link, NavLink } from 'react-router-dom';

export default function NavbarHome() {

  return (

    <nav *className*="navbar navbar-expand-lg navbar-light bg-light">

      <*Link* *className*="navbar-brand" *to*="/">Home Navbar</*Link*>

      <button *className*="navbar-toggler" *type*="button" *data-toggle*="collapse" *data-target*="#navbarSupportedContent01" *aria-controls*="navbarSupportedContent" *aria-expanded*="false" *aria-label*="Toggle navigation">

        <span *className*="navbar-toggler-icon" />

      </button>

      <div *className*="collapse navbar-collapse" *id*="navbarSupportedContent01">

        <ul *className*="navbar-nav mr-auto">

          <li *className*="nav-item">

            <*NavLink* *exact* *activeClassName*="active" *className*="nav-link" *to*="/">Home</*NavLink*>

          </li>

          <li *className*="nav-item">

            <*NavLink* *activeClassName*="active" *className*="nav-link" *to*="/about">About</*NavLink*>

          </li>

          <li *className*="nav-item">

            <*NavLink* *activeClassName*="active" *className*="nav-link" *to*="/list-movies">List Movies</*NavLink*>

          </li>

        </ul>

      </div>

    </nav>

  )

}

**=== Xây dựng trang List Movies Page => thao tác với API bằng Axios ===**

npm install axios –save

- Sử dụng Hook useEffect() của stateless component (tương đương với các life-cycle componentWillUnmount, componentDidMount, componentDidUpdate trong statefull component) để render từ axios trả về

import axios from 'axios';

import React, { useEffect } from 'react';

import Movie from './Movie';

export default function ListMoviesPage() {

  useEffect(() => {

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyPhim/LayDanhSachPhim?maNhom=GP02",

      method: "GET",

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        console.log(*result*.data.content);

      })

      .catch(*error* => {

        console.log(*error*);

      })

  }, []);

  return (

    <div *className*='container'>

      <div *className*='row'>

        <*Movie*/>

        <*Movie*/>

        <*Movie*/>

        <*Movie*/>

        <*Movie*/>

      </div>

    </div>

  )

}

- Vấn đề đặt ra: cần lấy result.data từ API trả về để duyệt mảng và render ra các component Movie, (trong statefull component có thể tạo state và setState, khi state thay đổi sẽ render lại lần mới và có data để map()); trong stateless component => sử dụng Hooks useState; đồng thời tạo thêm component <Loading/> để thông báo cho end-user biết đang trong quá trình pending khi gọi API => ListMoviesPage được viết lại:

import axios from 'axios';

import React, { useEffect, useState } from 'react';

import Loading from '../../../components/Loader';

import Movie from './Movie';

export default function ListMoviesPage() {

  const [data, setData] = useState({

    listMovie: null,

    loading: false,

    error: null

  }); //Một số trường hợp nên tách riêng từng property để dễ quản lý việc render lại của các child component => tốt cho performance

  const handleRenderData = () => {

    if (data.listMovie) {

      return data.listMovie.map(*movie* => {

        return <*Movie* *key*={*movie*.maPhim} *movie*={*movie*} />

      })

    }

  };

  useEffect(() => {

    setData({ listMovie: null, loading: true, error: null }); // Request

    // Set waiting time

    const waitingTime = setTimeout(() => {

      setData({ listMovie: null, loading: false, error: null }); // (Tính năng thêm)

      return alert("Time out!");

    }, 5000);

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyPhim/LayDanhSachPhim?maNhom=GP02",

      method: "GET",

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        clearTimeout(waitingTime);

        setData({ listMovie: *result*.data.content, loading: false, error: null }); // Success

      })

      .catch(*error* => {

        setData({ listMovie: null, loading: false, error: *error*.response.data.message }); // Failed

        alert(*error*.response.data.message); //Do vẫn chưa thoát function scope nên hàm setData dù đã làm thay đổi data.error nhưng thời điểm capture vẫn là null => nếu alert(data.error) vẫn hiện kết quả null => có thể sử dụng useState với chức năng nhưng componentDidMount để alert(data.error)

        clearTimeout(waitingTime);

      })

  }, []);

  if (data.loading) {

    return <*Loading* />

  }

  return (

    <div *className*='container'>

      <div *className*='row'>

        {handleRenderData()}

      </div>

    </div>

  )

}

Component Movie

import React from 'react';

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

export default function Movie({ *movie* }) {

  // const { movie } = props;

  const MAX\_TITLE\_LENGTH = 18;

  return (

    <div *className*="col-md-3 p-3">

      <div *className*="card overflow-hidden">

        <img *className*="card-img-top d-inline-block" *src*={*movie*.hinhAnh} *alt*={*movie*.biDanh} *style*={{ height: 350, minWidth: "100%" }} />

        <div *className*="card-body">

          <h5 *className*="card-title">{*movie*.tenPhim.substring(0, MAX\_TITLE\_LENGTH)}{*movie*.tenPhim.length > MAX\_TITLE\_LENGTH ? "..." : ""}</h5>

         <*Link* *className*='btn btn-success' *to*="./">Detail</*Link*>

        </div>

      </div>

    </div>

  )

}

**=== Xây dựng components dùng chung cho project (Loader – thông báo trạng thái pending khi đang get API) ===**

(src > components > loader)

npm install styled-components --save để code bằng styled hoặc chèn bằng file css

import React from 'react';

import styled, { keyframes } from 'styled-components';

const spin = keyframes`

  0% {

    transform: rotate(0deg);

  }

  100% {

    transform: rotate(360deg);

  }

`

const Loader = styled.div`

  margin: auto;

  margin-top: 1rem;

  border: 7px solid #f3f3f3;

  border-radius: 50%;

  border-top: 7px solid ${(*props*) => (*props*.color ? "#3498db" : "red")};

width: 60px;

height: 60px;

animation: ${spin} 1s linear infinite;

`

// Nếu component được truyền vào props là color thì màu #3498db, nếu không có props là color sẽ có màu đỏ

const Notification = styled.p`

margin - top: 1rem;

text - align: center;

`

export default function Loading() {

  return (

    <div>

      <*Loader* *color* />

      <*Notification*>

        Loading...

      </*Notification*>

    </div>

  )

}

**=== Sử dụng thư viện MUI (tương tự như Bootstrap, nhưng MUI sử dụng cấu trúc JSS - CSS in JS) ===**

npm install @mui/material @emotion/react @emotion/styled @mui/icons-material

- Demo về cách sử dụng MUI trong AboutPage (để làm sau + SCSS?)

**=== Scroll to top (sử dụng thư viện react-scroll-to-top) ===**

npm i react-scroll-to-top

**=== REDUCER + Nâng cấp lưu trữ state của ListMoviesPage bằng reducer ===**

npm install redux react-redux –save

- Tạo store lưu trữ global (rootReducer):

(src/reducers/index.js) => tạo store global và các reducer con nằm trong thư mục của từng component

Tại file index.js của reducers

import { combineReducers } from "redux"; //Combine các reducer con

export const rootReducer = combineReducers({

  // Nơi chứa các reducer của components ...

});

Tại file index.js lớn nhất của project, tạo store bằng redux và dùng <Provider> của react-redux để connect 2 thư viện với nhau, <Provider> truyền vào props là store vừa tạo và bao bọc <App/> bên trong:

import React from 'react';

import ReactDOM from 'react-dom';

import 'bootstrap/dist/css/bootstrap.min.css';

import './index.css';

import App from './App';

import 'jquery/dist/jquery.min.js';

import 'popper.js/dist/popper.min.js';

import 'bootstrap/dist/js/bootstrap.min.js'

import reportWebVitals from './reportWebVitals';

import { createStore } from 'redux';//Create store bằng redux

import { rootReducer } from './reduces';//Import rootReducer

import { Provider } from 'react-redux';//Connect react và redux

// const store = createStore(rootReducer);

// Khi không sử dụng API => tạo store với tham số là rootReducer; tham số thứ 2 có thể có hoặc ko, dùng cho extension redux dev tool trên trình duyệt để xem cho dễ dàng

const store = createStore(rootReducer,window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_());

ReactDOM.render(

  <*Provider* *store*={store}>

    <*App* />

  </*Provider*>,

  document.getElementById('root')

  );

  reportWebVitals();

- Tạo store (reducer) cho ListMoviesPage:

(src/containers/HomeTemplate/ListMoviesPage/modules/(actions.js, constants.js, reducer.js))

Trong ListMoviesPage: lấy store (state: data, loading, error) của trang index.js mang qua reducer.js để tạo state ban đầu và structure của reducer cho component ListMoviesPage

const initialState = {

  listMovie: null,

  loading: false,

  error: null

}

// Sử dụng default params, nếu không truyền tham số thì sẽ lấy initialState là tham số mặc định, tham số thứ 2 là action để component truyền cho store xử lý

const listMoviesReducer = (*state* = initialState, *action*) => {

  switch (*action*.type) {

    // case ...(khi dispatch các type (+ payload) lên rootReducer => dựa vào action đó có các case là action.type để phân loại xem thuộc reducer con nào và xử lý các logic tiếp theo)

    default:

      return {...*state*}; //Sau khi thay đổi trong state => clone lại state mới và return state mới này ra để component biết store có thay đổi hay không

  }

}

export default listMoviesReducer;

Cập nhật reducer của ListMoviesPage vào rootReducer để combine

import { combineReducers } from "redux";

import listMoviesReducer from "../containers/HomeTemplate/ListMoviesPage/modules/reducer";

export const rootReducer = combineReducers({

  listMoviesReducer,

});

Để biết được có bao nhiêu action trong reducer của ListMoviesPage => xem trong file index.js của ListMoviesPage ban đầu có các vị trí nào cần làm thay đổi state (xem tại các thời điểm //Request, //Success,//Failed)

Trong file actions.js:

const actListMoviesRequest = () => {

  return {

    type: ActionType.LIST\_MOVIES\_REQUEST,

  }

};

const actListMoviesSuccess = (*data*) => {

  return {

    type: ActionType.LIST\_MOVIES\_SUCCESS,

    payload: *data*

  }

};

const actListMoviesFailed = (*error*) => {

  return {

    type: ActionType.LIST\_MOVIES\_FAILED,

    payload: *error*

  }

};

export { actListMoviesRequest, actListMoviesSuccess, actListMoviesFailed }

Trong file reducer.js :

const initialState = {

  listMovie: null,

  loading: false,

  error: null

}

const listMoviesReducer = (*state* = initialState, *action*) => {

  switch (*action*.type) {

    case "LIST\_MOVIES\_REQUEST": {

*state*.listMovie = null;

*state*.loading = true;

*state*.error = null;

      return { ...*state* };

    }

    case "LIST\_MOVIES\_SUCCESS": {

*state*.listMovie = *action*.payload;

*state*.loading = false;

*state*.error = null;

      return { ...*state* };

    }

    case "LIST\_MOVIES\_FAILED": {

*state*.listMovie = null;

*state*.loading = false;

*state*.error = *action*.payload;

      return { ...*state* };

    }

    default:

      return { ...*state* };

  }

}

export default listMoviesReducer;

=> Tạo sẵn các constants và cập nhật lại actions/reducer/index.js của ListMoviesPage để dễ quản lý code nếu có sự thay đổi:

Trong constants.js :

// const LIST\_MOVIES\_REQUEST = 'LIST\_MOVIES\_REQUEST';

// const LIST\_MOVIES\_SUCCESS = 'LIST\_MOVIES\_SUCCESS';

// const LIST\_MOVIES\_FAILED = 'LIST\_MOVIES\_FAILED';

// Thông thường sẽ thêm prefix là @.../ để dễ nhận biết các constants đó dành cho chức năng gì và không duplicate với các action type khác

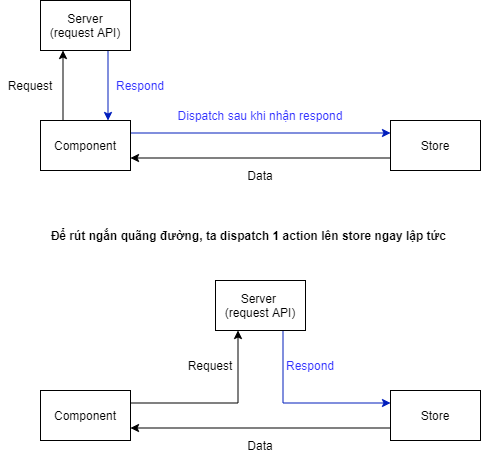
const LIST\_MOVIES\_REQUEST = '@listmovies/LIST\_MOVIES\_REQUEST';

const LIST\_MOVIES\_SUCCESS = '@listmovies/LIST\_MOVIES\_SUCCESS';

const LIST\_MOVIES\_FAILED = '@listmovies/LIST\_MOVIES\_FAILED';

export { LIST\_MOVIES\_REQUEST, LIST\_MOVIES\_SUCCESS, LIST\_MOVIES\_FAILED }

Trong actions.js được viết lại (với file actions.js là khu vực middleWare – ta sẽ call API tại middleWare để tăng performance => tuy nhiên gọi API sẽ dẫn đến quá trình bất đồng bộ tại middleWare trong khi đó Reducer chỉ chấp nhận xử lý cho quá trình đồng bộ => phải đảm bảo nhận response từ server mới dispatch qua reducer ?? -> chưa hiểu : do dispatch bên trong .then() hoặc .catch() có nghĩa là đã nhận response???) :



import axios from 'axios';

import \* as ActionType from './constants';

export const actFetchListMovies = () => {

  // Call API ngay tại Action creator để tăng performance (không phải đi qua trung gian là component trước khi dispatch lên store). Tuy nhiên việc gọi API từ server là bất đồng bộ => không chắc khi dispatch lên reducer thì response đã trả về hay chưa => sử dụng middleWare (dispatch action lên reducer phải qua middleWare để đảm bảo rằng khi tới được reducer thì response từ server đã được trả về - có nhiều loại middleWare, ở đây ta sử dụng redux-thunk => xem tại file index của project)

  return (*dispatch*) => {

    // Request

    dispatch(actListMoviesRequest());

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyPhim/LayDanhSachPhim?maNhom=GP02",

      method: "GET",

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        dispatch(actListMoviesSuccess(*result*.data.content));

      })

      .catch(*error* => {

        dispatch(actListMoviesFailed(*error*));

      });

  }

}

const actListMoviesRequest = () => {

  return {

    type: ActionType.LIST\_MOVIES\_REQUEST,

  }

};

const actListMoviesSuccess = (*data*) => {

  return {

    type: ActionType.LIST\_MOVIES\_SUCCESS,

    payload: *data*

  }

};

const actListMoviesFailed = (*error*) => {

  return {

    type: ActionType.LIST\_MOVIES\_FAILED,

    payload: *error*

  }

};

Chưa handle được vấn đề đặt thêm setTimeout để ngăn chặn axios trả về then hoặc catch => xử lý pending time trong API?

reducer.js được viết lại :

import \* as ActionType from './constants';

const initialState = {

  listMovie: null,

  loading: false,

  error: null

}

const listMoviesReducer = (*state* = initialState, *action*) => {

  switch (*action*.type) {

    case ActionType.LIST\_MOVIES\_REQUEST: {

*state*.listMovie = null;

*state*.loading = true;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LIST\_MOVIES\_SUCCESS: {

*state*.listMovie = *action*.payload;

*state*.loading = false;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LIST\_MOVIES\_FAILED: {

*state*.listMovie = null;

*state*.loading = false;

*state*.error = *action*.payload.response.data.message;

      return { ...*state* };

    }

    default:

      return { ...*state* };

  }

}

export default listMoviesReducer;

index.js của ListMoviesPage được viết lại:

import React, { useEffect, useState } from 'react';

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

import Loading from '../../../components/Loader';

import Movie from './Movie';

import { actFetchListMovies } from './modules/actions';

export default function ListMoviesPage() {

  const dispatch = useDispatch();

  // Get data from reducer

  const listMovie = useSelector(*state* => *state*.listMoviesReducer.listMovie);

  const loading = useSelector(*state* => *state*.listMoviesReducer.loading);

  const error = useSelector(*state* => *state*.listMoviesReducer.error);

  const handleRenderData = () => {

    if (listMovie) {

      return listMovie.map(*movie* => {

        return <*Movie* *key*={*movie*.maPhim} *movie*={*movie*} />

      })

    }

  };

  useEffect(() => {

    dispatch(actFetchListMovies());

  }, []);

  if (loading) {

    return <*Loading* />

  }

  return (

    <div *className*='container'>

      <div *className*='row'>

        {handleRenderData()}

      </div>

    </div>

  )

}

**- TUY NHIÊN:** khi call API sẽ gặp quá trình pending => dispatch lên store của redux sẽ bị lỗi do Reducer CHỈ XỬ LÝ NHỮNG TIẾN TRÌNH ĐỒNG BỘ (thời điểm dispatch lên store và lấy dữ liệu từ store về)=> xử lý bằng thunk của redux-thunk, applyMiddleWare và compose của redux

npm install redux-thunk –save

Tại file index.js lớn nhất của project được viết lại:

import React from 'react';

import ReactDOM from 'react-dom';

import 'bootstrap/dist/css/bootstrap.min.css';

import './index.css';

import App from './App';

import 'jquery/dist/jquery.min.js';

import 'popper.js/dist/popper.min.js';

import 'bootstrap/dist/js/bootstrap.min.js'

import reportWebVitals from './reportWebVitals';

import { applyMiddleware, compose, createStore } from 'redux';//applyMiddleware và compose để xử lý bất đồng bộ trước khi dispatch lên store của reducer

import { rootReducer } from './reduces';

import { Provider } from 'react-redux';

import thunk from 'redux-thunk';//Đồng bộ hóa dữ liệu trước khi đưa lên store

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

const store = createStore(rootReducer, composeEnhancers(applyMiddleware(thunk)));

ReactDOM.render(

  <*Provider* *store*={store}>

    <*App* />

  </*Provider*>,

  document.getElementById('root')

);

reportWebVitals();

**=== Xử lý cho trang detail của từng Movie ===**

- Tạo thêm trang DetailMoviePage (src/containers/HomeTemplate/DetailMoviePage/index.js)

- Tạo bộ định tuyến cho trang DetailMoviePage => đây là bộ định tuyến động dựa trên id của từng bộ phim để mỗi lần click vào 1 bộ phim sẽ render khác nhau dựa trên id của bộ phim đó

import AboutPage from "../containers/HomeTemplate/AboutPage";

import HomePage from "../containers/HomeTemplate/HomePage";

import HomeTemplate from "../containers/HomeTemplate";

import ListMoviesPage from "../containers/HomeTemplate/ListMoviesPage";

import DetailMoviePage from "../containers/HomeTemplate/DetailMoviePage";

const routesHome = [

  {

    exact: true,

    path: "/",

    component: HomePage

  },

  {

    exact: false,

    path: "/about",

    component: AboutPage

  },

  {

    exact: false,

    path: "/list-movies",

    component: ListMoviesPage

  },

  {

    // Routes cho DetailMoviePage

    exact: false,

    path: "/detail/:id", //:id động do component <Movie/> render ra

    component: DetailMoviePage

  }

]

export const renderRoutesHome = () => {

  return routesHome.map((*route*, *index*) => {

    return <*HomeTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

- Tạo wireframe cơ bản cho trang DetailMoviePage: LƯU Ý: Do component được bao bọc bởi bộ định tuyến <Route> nên sẽ có các props mặc định (history, location, match) **=> Get params from url** để lấy id của Movie và call API

import React, { useEffect } from 'react'

import { useSelector } from 'react-redux'

export default function DetailMoviePage(*props*) {

  // Lấy maPhim (id) từ URL của browser: LƯU Ý: Do component được bao bọc bởi bộ định tuyến <Route> nên sẽ có các props mặc định (history, location, match)

  const movieId = *props*.match.params.id;

  const movie = useSelector(*state* => *state*.listMoviesReducer.listMovie.find(*movie* => *movie*.maPhim == movieId));

  useEffect(() => {

    console.log(movie);

  })

  return (

    <div>DetailMoviePage</div>

  )

}

- Tạo modules cho DetailMoviePage tương tự module của ListMoviePage (nếu trong API của List Movies trả về có đầy đủ thông tin thì không cần tạo thêm reducer do sử dụng được reducer của ListMoviePage để trả về từng Movie dựa trên maPhim (id) của nó)

- Tương tự, sau khi tạo modules, DetailMoviePage được viết lại:

import React, { Fragment, useEffect } from 'react';

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

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

import Loading from '../../../components/Loader';

import { actFetchDetailMovie } from './modules/actions';

export default function DetailMoviePage(*props*) {

  const dispatch = useDispatch();

  // Get data from reducer

  const movie = useSelector(*state* => *state*.detailMovieReducer.movie);

  const loading = useSelector(*state* => *state*.detailMovieReducer.loading);

  const error = useSelector(*state* => *state*.detailMovieReducer.error);

  useEffect(() => {

    const movieId = *props*.match.params.id;

    dispatch(actFetchDetailMovie(movieId));

  }, []);

  const handleRenderData = () => {

    if (error) {

      return <div *className*='m-auto p-3'>{error}</div>

    }

    if (movie) {

      return <*Fragment*>

        <div *className*='col-md-5'>

          <img *className*='img-fluid' *src*={movie.hinhAnh} />

        </div>

        <div *className*='col-md-7'>

          <table *className*='table'>

            <tbody>

              <tr>

                <td *style*={{ width: "30%", fontWeight: "bold" }}>Tên phim</td>

                <td>{movie.tenPhim}</td>

              </tr>

              <tr>

                <td *style*={{ width: "30%", fontWeight: "bold" }}>Mô tả</td>

                <td>{movie.moTa}</td>

              </tr>

              <tr>

                <td *style*={{ width: "30%", fontWeight: "bold" }}>Ngày khởi chiếu</td>

                <td>{**new** *Date*(movie.ngayKhoiChieu).toLocaleDateString()}</td>

              </tr>

            </tbody>

          </table>

          <div *className*='text-right'>

            <*Link* *className*='btn btn-success' *to*={`/booking/${movie.maPhim}`}>Đặt vé</*Link*>

            {/\* Tại đây chưa tạo route cho trang booking, nhưng kiểm tra lại đường dẫn ảnh svg của trang PageNotFound \*/}

          </div>

        </div>

      </*Fragment*>

    };

  }

  if (loading) {

    return <*Loading* />

  }

  return (

    <div *className*='container'>

      <div *className*='row py-3'>

        {handleRenderData()}

      </div>

    </div>

  )

}

Kiểm tra lại path của ảnh svg trong PageNotFound

**=== QUẢN LÝ CHỨC NĂNG LOGIN – ADMIN TEMPLATE ===**

Tương tự HomeTemplate, tạo các folder \_components, … LoginPage, AddUserPage, …

- Định tuyến các routes trong AdminTemplate và route cho chức năng login, trong file index.js của routes:

import AboutPage from "../containers/HomeTemplate/AboutPage";

import HomePage from "../containers/HomeTemplate/HomePage";

import HomeTemplate from "../containers/HomeTemplate";

import ListMoviesPage from "../containers/HomeTemplate/ListMoviesPage";

import DetailMoviePage from "../containers/HomeTemplate/DetailMoviePage";

import AdminTemplate from "../containers/AdminTemplate";

import DashboardPage from "../containers/AdminTemplate/DashboardPage"

const routesHome = [

  {

    exact: true,

    path: "/",

    component: HomePage

  },

  {

    exact: false,

    path: "/about",

    component: AboutPage

  },

  {

    exact: false,

    path: "/list-movies",

    component: ListMoviesPage

  },

  {

    exact: false,

    path: "/detail/:id",

    component: DetailMoviePage

  },

];

const routesAdmin = [

  {

    exact: false,

    path: "/dashboard",

    component: DashboardPage

  },

]

export const renderRoutesHome = () => {

  return routesHome.map((*route*, *index*) => {

    return <*HomeTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

export const renderRoutesAdmin = () => {

  return routesAdmin.map((*route*, *index*) => {

    return <*AdminTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

Trong App.js :

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

import LoginPage from "./containers/AdminTemplate/LoginPage";

import PageNotFound from "./containers/HomeTemplate/PageNotFound";

import { renderRoutesAdmin, renderRoutesHome } from "./routes";

function App() {

  return (

    <*BrowserRouter*>

      <*Switch*>

        {renderRoutesHome()}

        <*Route* *path*="/login" *component*={LoginPage} />

        {renderRoutesAdmin()}

        <*Route* *path*="" *component*={PageNotFound} />

      </*Switch*>

    </*BrowserRouter*>

  );

}

export default App;

Trong actions.js của LoginPage :

import axios from 'axios';

import \* as ActionType from './constants';

export const actLogin = (*userInfo*) => {

  return (*dispatch*) => {

    dispatch(actLoginRequest());

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyNguoiDung/DangNhap",

      method: "POST",

      data: *userInfo*,

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        console.log(*result*.data.content);

        dispatch(actLoginSuccess(*result*.data.content));

      })

      .catch(*error* => {

        dispatch(actLoginFailed(*error*));

      });

  }

}

const actLoginRequest = () => {

  return {

    type: ActionType.LOGIN\_REQUEST,

  }

};

const actLoginSuccess = (*data*) => {

  return {

    type: ActionType.LOGIN\_SUCCESS,

    payload: *data*

  }

};

const actLoginFailed = (*error*) => {

  return {

    type: ActionType.LOGIN\_FAILED,

    payload: *error*

  }

};

Trong reducer.js của LoginPage (export và combine trong rootReducer) :

import \* as ActionType from "./constants";

const initialState = {

  user: null,

  loading: false,

  error: null

}

const userLoginReducer = (*state* = initialState, *action*) => {

  switch (*action*.type) {

    case ActionType.LOGIN\_REQUEST: {

*state*.user = null;

*state*.loading = true;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LOGIN\_SUCCESS: {

*state*.user = *action*.payload;

*state*.loading = false;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LOGIN\_FAILED: {

*state*.user = null;

*state*.loading = false;

*state*.error = *action*.payload;

      return { ...*state* };

    }

    default:

      return { ...*state* }

  }

}

export default userLoginReducer;

Trong index.js của LoginPage :

import React, { useState } from 'react';

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

import Loading from '../../../components/Loader';

import { actLogin } from './modules/actions';

export default function LoginPage() {

  const [infoField, setInfoField] = useState({

    taiKhoan: "",

    matKhau: ""

  });

  const loading = useSelector(*state* => *state*.userLoginReducer.loading);

  const error = useSelector(*state* => *state*.userLoginReducer.error);

  const dispatch = useDispatch();

  const handleLogin = (*event*) => {

*event*.preventDefault();

    dispatch(actLogin(infoField));

  };

  const handleOnChange = (*event*) => {

    setInfoField({ ...infoField, [*event*.target.name]: *event*.target.value });

  };

  const notifications = () => {

    if (loading) {

      return <*Loading* />

    }

    return error && <div *className*='alert alert-danger mt-3'>{error.response.data.content}</div>

  }

  return (

    <div *className*="container">

      <div *className*="row py-3">

        <div *className*="col-md-5 mx-auto">

          <h4 *className*='text-center'>Login Page</h4>

          <form *onSubmit*={handleLogin}>

            <div *className*="form-group">

              <label>Username</label>

              <input *type*="text" *className*="form-control" *name*="taiKhoan" *onChange*={handleOnChange} />

            </div>

            <div *className*="form-group">

              <label>Password</label>

              <input *type*="password" *className*="form-control" *name*="matKhau" *autoComplete*="on" *onChange*={handleOnChange} />

            </div>

            <div *className*='text-center'>

              <button *type*="submit" *className*="btn btn-success">Login</button>

            </div>

          </form>

          <div *className*='text-monospace'>

            <p>Example:</p>

            <p>Admin account: qwerty - 123</p>

            <p>User account: test003 - 1</p>

          </div>

          {notifications()}

        </div>

      </div>

    </div>

  )

}

- Một số cập nhật thêm cho tính năng login :

+ Check nhóm người dùng : nếu user là admin mới được vào trang dashboard, nếu user không thuộc nhóm admin => return và không dispatch lên store

Xem thêm về Promise, async-await:

<https://kipalog.com/posts/Su-khac-biet-cua-2-cach-su-dung-Resolve-Reject-trong-Promise>

<https://ehkoo.com/bai-viet/tat-tan-tat-ve-promise-va-async-await>

+ Lưu trạng thái login : bằng localStorage (=> tính năng đăng xuất sẽ xóa localStorage đó)

+ Redirect (chuyển hướng sang trang dashboard nếu đăng nhập thành công bằng tài khoản admin) : Nếu trong component có thể sử dụng Redirect của react-router-dom để chuyển hướng; ở đây là file js return về function (không phải component) nên có thể sử dụng history.push("/\_link") hoặc history.replace("/\_link") để chuyển hướng; tuy nhiên hàm replace() của history kèm theo chức năng không cho back trở về trang trước đó (trang login) => sẽ phù hợp hơn với chức năng login; history được lấy từ props.history của component truyền vào (khi component được bao bọc trong bộ định tuyến)

- Trong constants.js của LoginPage được viết lại :

const LOGIN\_REQUEST = '@login/LOGIN\_REQUEST';

const LOGIN\_SUCCESS = '@login/LOGIN\_SUCCESS';

const LOGIN\_FAILED = '@login/LOGIN\_FAILED';

const LOGIN\_RESET = '@login/LOGIN\_RESET'; //Thêm tính năng reset trạng thái error sau khi thoát khỏi trang login

export { LOGIN\_REQUEST, LOGIN\_SUCCESS, LOGIN\_FAILED, LOGIN\_RESET }

- Trong actions.js của LoginPage được viết lại :

import axios from 'axios';

import \* as ActionType from './constants';

export const actLogin = (*userInfo*, *history*) => {

  return (*dispatch*) => {

    dispatch(actLoginRequest());

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyNguoiDung/DangNhap",

      method: "POST",

      data: *userInfo*,

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        // Check nhóm người dùng, nếu là admin sẽ tiếp tục dispatch và chuyển hướng sang trang quản trị dashboard

        if (*result*.data.content.maLoaiNguoiDung !== "QuanTri") {

          return *Promise*.reject({

            response: {

              data: {

                content: "Không có quyền truy cập vào trang quản trị"

              }

            }

          });

        }

        // Lưu thông tin đăng nhập tại localStorage

        localStorage.setItem("UserInfo", JSON.stringify(*result*.data.content)); //Lưu thông tin ở localStorage

        // Redirect

*history*.replace("/dashboard"); //Nếu trong component có thể sử dụng Redirect của react-router-dom để chuyển hướng; ở đây là file js return về function (không phải component) nên có thể sử dụng history.push("/\_link") hoặc history.replace("/\_link") để chuyển hướng; tuy nhiên hàm replace() của history kèm theo chức năng không cho back trở về trang trước đó (trang login) => sẽ phù hợp hơn với chức năng login; history được lấy từ props.history của component truyền vào (khi component được bao bọc trong bộ định tuyến)

        dispatch(actLoginSuccess(*result*.data.content));

      })

      .catch(*error* => {

        dispatch(actLoginFailed(*error*));

      });

  }

}

const actLoginRequest = () => {

  return {

    type: ActionType.LOGIN\_REQUEST,

  }

};

const actLoginSuccess = (*data*) => {

  return {

    type: ActionType.LOGIN\_SUCCESS,

    payload: *data*

  }

};

const actLoginFailed = (*error*) => {

  return {

    type: ActionType.LOGIN\_FAILED,

    payload: *error*

  }

};

- Trong reducer.js của LoginPage được viết lại :

import \* as ActionType from "./constants";

const initialState = {

  user: null,

  loading: false,

  error: null

}

const userLoginReducer = (*state* = initialState, *action*) => {

  switch (*action*.type) {

    case ActionType.LOGIN\_REQUEST: {

*state*.user = null;

*state*.loading = true;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LOGIN\_SUCCESS: {

*state*.user = *action*.payload;

*state*.loading = false;

*state*.error = null;

      return { ...*state* };

    }

    case ActionType.LOGIN\_FAILED: {

*state*.user = null;

*state*.loading = false;

*state*.error = *action*.payload;

      return { ...*state* };

    }

    case ActionType.LOGIN\_RESET: {

*state*.error = null;

      return { ...*state* };

    }

    default:

      return { ...*state* }

  }

}

export default userLoginReducer;

- Trang index.js của LoginPage được viết lại :

import React, { useEffect, useState } from 'react';

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

import Loading from '../../../components/Loader';

import { actLogin } from './modules/actions';

import { LOGIN\_RESET } from './modules/constants';

export default function LoginPage(*props*) {

  const [infoField, setInfoField] = useState({

    taiKhoan: "",

    matKhau: ""

  });

  const error = useSelector(*state* => *state*.userLoginReducer.error); //Đang lỗi còn lưu lại ở reducer thông báo lỗi khi đăng nhập thất bại (nếu back về home và trở lại trang login sẽ thấy lỗi) => xử lý: khi thoát khỏi trang sẽ dispatch thêm action để xóa error trong reducer

  const loading = useSelector(*state* => *state*.userLoginReducer.loading);

  // Sử dụng useEffect với cú pháp tương đương componentWillUnmount reset lại error của reducer để khi back lại trang login sẽ không thấy lỗi

  useEffect(() => {

    return () => {

      dispatch({

        type: LOGIN\_RESET

      })

    }

  }, [])

  const dispatch = useDispatch();

  const handleLogin = (*event*) => {

*event*.preventDefault();

    dispatch(actLogin(infoField, *props*.history)); //Truyền data và history vào actions.js

  };

  const handleOnChange = (*event*) => {

    setInfoField({ ...infoField, [*event*.target.name]: *event*.target.value });

  };

  const notifications = () => {

    if (loading) {

      return <*Loading* />

    }

    return error && <div *className*='alert alert-danger mt-3'>{error.response.data.content}</div>

  }

  return (

    <div *className*="container">

      <div *className*="row py-3">

        <div *className*="col-md-5 mx-auto">

          <h4 *className*='text-center'>Login Page</h4>

          <form *onSubmit*={handleLogin}>

            <div *className*="form-group">

              <label>Username</label>

              <input *type*="text" *className*="form-control" *name*="taiKhoan" *onChange*={handleOnChange} />

            </div>

            <div *className*="form-group">

              <label>Password</label>

              <input *type*="password" *className*="form-control" *name*="matKhau" *autoComplete*="on" *onChange*={handleOnChange} />

            </div>

            <div *className*='text-center'>

              <button *type*="submit" *className*="btn btn-success">Login</button>

            </div>

          </form>

          <div *className*='text-monospace'>

            <p>Example:</p>

            <p>Admin account: qwerty - 123</p>

            <p>User account: test003 - 1</p>

          </div>

          {notifications()}

        </div>

      </div>

    </div>

  )

}

**=== Khóa trang dashboard khi chưa đăng nhập bằng tài khoản admin ===**

- Kiểm tra tại AdminTemplate, nếu đã đăng nhập trong localStorage (bằng tài khoản admin) thì mới render ra các <Route> chứa các trang quản trị

- Trang index.js của AdminTemplate được viết lại :

import React, { Fragment } from 'react';

import { Route, Redirect } from 'react-router-dom';

import AdminDashboard from './\_components/Dashboard';

export default function AdminTemplate({ *exact*, *path*, *component* }) {

  // Nếu localStorage đã tồn tại user (đăng nhập bằng admin mới lưu tại localStorage) mới render ra các Route chứa các trang quản trị

  if (localStorage.getItem("UserInfo")) {

    return (

      <*Fragment*>

        <div *className*='row p-3'>

          <div *className*='col-md-2'>

            <*AdminDashboard* />

          </div>

          <div *className*='col-md-10'>

            <*Route* *exact*={*exact*} *path*={*path*} *component*={*component*} />

          </div>

        </div>

      </*Fragment*>

    )

  }

  // Nếu localStorage chưa tồn tại thì redirect sang trang khác (VD: LoginPage)

  // Có thể sử dụng Redirect của react-router-dom trong component hoặc nếu không trong component thì sử dụng props.history.push() hoặc props.history.replace()

  return <*Redirect* *to*="/login" />

}

**=== Chức năng đăng xuất (xóa localStorage) và redirect về trang Login ===**

Tại AdminTemplate truyền thêm flag vào component Sidebar, khi click vào đăng xuất ở component Sidebar thì xóa localStorage đồng thời truyền flag về AdminTemplate là false để render lại và redirect ra ngoài trang LoginPage

- Trong index của AdminTemplate được viết lại :

import React, { Fragment, useEffect, useState } from 'react';

import { Route, Redirect } from 'react-router-dom';

import AdminSidebar from './\_components/Sidebar';

export default function AdminTemplate({ *exact*, *path*, *component* }) {

  const [flag, setFlag] = useState(true);

  const transData = (*data*) => {

    setFlag(*data*);

  }

  if (localStorage.getItem("UserInfo") && flag) {

    return (

      <*Fragment*>

        <div *className*='row p-3'>

          <div *className*='col-md-2'>

            <*AdminSidebar* *transData*={transData} />

          </div>

          <div *className*='col-md-10'>

            <*Route* *exact*={*exact*} *path*={*path*} *component*={*component*} />

          </div>

        </div>

      </*Fragment*>

    )

  }

  return <*Redirect* *to*="/login" />

}

- Trong Sidebar được viết lại :

import React from 'react'

import { Link, NavLink } from 'react-router-dom';

export default function AdminSidebar(*props*) {

  const handleLogout = () => {

    if (window.confirm("Really?")) {

      localStorage.removeItem("UserInfo");

*props*.transData(false);

    }

  }

  return (

    <ul *className*="nav flex-column">

      <li *className*="nav-item">

        <*NavLink* *activeClassName*='active' *className*="nav-link" *to*="/dashboard">Dashboard</*NavLink*>

      </li>

      <li *className*="nav-item">

        <*NavLink* *exact* *activeClassName*="active" *className*="nav-link" *to*="/">Home</*NavLink*>

      </li>

      <li *className*="nav-item">

        <*NavLink* *activeClassName*='active' *className*="nav-link" *to*="/add-user">Add User</*NavLink*>

      </li>

      <li *className*="nav-item">

        <*Link* *className*="nav-link" *style*={{ cursor: "pointer" }} *onClick*={handleLogout}>Logout</*Link*>

      </li>

    </ul>

  )

}

- Tại Navbar của HomeTemplate có thể được viết lại để biết trạng thái đã đăng nhập :

import './style.css';

import React, { Fragment } from 'react';

import { Link, NavLink } from 'react-router-dom';

export default function NavbarHome() {

  const handleRenderLink = () => {

    if (localStorage.getItem("UserInfo")) {

      return <*Link* *className*="nav-link px-1" *to*="/dashboard">Dashboard</*Link*>

    }

    return (

      <*Fragment*>

        <*Link* *className*="nav-link px-1" *to*="/login">Login</*Link*>

        <*Link* *className*="nav-link px-1 disabled" *to*="/">Register</*Link*>

      </*Fragment*>

    )

  }

  return (

    <nav *className*="navbar navbar-expand-lg navbar-light bg-light justify-content-between">

      <div *className*='d-flex'>

        <*Link* *className*="navbar-brand" *to*="/">Logo</*Link*>

        <button *className*="navbar-toggler" *type*="button" *data-toggle*="collapse" *data-target*="#navbarSupportedContent01" *aria-controls*="navbarSupportedContent" *aria-expanded*="false" *aria-label*="Toggle navigation">

          <span *className*="navbar-toggler-icon" />

        </button>

        <div *className*="collapse navbar-collapse" *id*="navbarSupportedContent01">

          <ul *className*="navbar-nav mr-auto">

            <li *className*="nav-item">

              <*NavLink* *exact* *activeClassName*="active" *className*="nav-link" *to*="/">Home</*NavLink*>

            </li>

            <li *className*="nav-item">

              <*NavLink* *activeClassName*="active" *className*="nav-link" *to*="/about">About</*NavLink*>

            </li>

            <li *className*="nav-item">

              <*NavLink* *activeClassName*="active" *className*="nav-link" *to*="/list-movies">List Movies</*NavLink*>

            </li>

          </ul>

        </div>

      </div>

      <div *className*='d-flex'>

        {handleRenderLink()}

      </div>

    </nav>

  )

}

**=== Rút gọn đường dẫn import (thay đổi đường dẫn - Aliases) (sau này nên thiết lập đầu tiên để không phải chỉnh sửa) ===**

- Sử dụng jsconfig.json => tạo file jsconfig.json đồng cấp với các file json khác của thư mục gốc

- Trong file jsconfig.json :

{

  "compilerOptions": {

    "baseUrl": "./src", //Thư mục đường dẫn chứa code hay sử dụng để import

    "paths": {

      "components/\*": ["./src/components/\*"], //Đường dẫn components/... sẽ đại diện cho tất cả thư mục trong ./src/components/...

      "containers/\*": ["./src/containers/\*"],

      "reducers/\*": ["./src/reducers/\*"],

      "routes/\*": ["./src/routes/\*"],

    }

  }

}

- Ví dụ về thay đổi đường dẫn trong file index.js của AddUserPage => các trang khác làm tương tự

Trước :

import Loading from '../../../components/Loader';

Sau:

import Loading from 'components/Loader';

**=== Tăng performance bằng kỹ thuật LAZY-LOAD (CODE-SPLITTING) ===**

Đọc thêm về SPA <https://viblo.asia/p/single-page-application-concept-LzD5dDvo5jY>

- Ban đầu: VD: Đang ở trang HomePage nhưng các trang khác đã load dữ liệu => không tốt về performance

Viết theo SPA (single page application) => chỉ load 1 lần (xem tại F12 -> Network -> JS -> chỉ có 1 file bundle.js duy nhất để load mọi thứ) => thiết lập Lazy loading để khi vào trang nào chỉ load content ở nội dung đó :

- Xử lý lazy load tại các thẻ <Route> => không gọi trực tiếp các component mà thông qua lazy của react

Trong index.js của routes ban đầu :

import AboutPage from "containers/HomeTemplate/AboutPage";

import HomePage from "containers/HomeTemplate/HomePage";

import HomeTemplate from "containers/HomeTemplate";

import ListMoviesPage from "containers/HomeTemplate/ListMoviesPage";

import DetailMoviePage from "containers/HomeTemplate/DetailMoviePage";

import AdminTemplate from "containers/AdminTemplate";

import DashboardPage from "containers/AdminTemplate/DashboardPage"

import AddUserPage from "containers/AdminTemplate/AddUserPage";

const routesHome = [

  {

    exact: true,

    path: "/",

    component: HomePage

  },

  {

    exact: false,

    path: "/about",

    component: AboutPage

  },

  {

    exact: false,

    path: "/list-movies",

    component: ListMoviesPage

  },

  {

    exact: false,

    path: "/detail/:id",

    component: DetailMoviePage

  },

];

const routesAdmin = [

  {

    exact: false,

    path: "/dashboard",

    component: DashboardPage

  },

  {

    exact: false,

    path: "/add-user",

    component: AddUserPage

  },

]

export const renderRoutesHome = () => {

  return routesHome.map((*route*, *index*) => {

    return <*HomeTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

export const renderRoutesAdmin = () => {

  return routesAdmin.map((*route*, *index*) => {

    return <*AdminTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

Trong index.js của routes được viết lại :

import HomeTemplate from "containers/HomeTemplate";

import AdminTemplate from "containers/AdminTemplate";

import { lazy } from "react";

const routesHome = [

  {

    exact: true,

    path: "/",

    component: lazy(() => import("containers/HomeTemplate/HomePage"))

  },

  {

    exact: false,

    path: "/about",

    component: lazy(() => import("containers/HomeTemplate/AboutPage"))

  },

  {

    exact: false,

    path: "/list-movies",

    component: lazy(() => import("containers/HomeTemplate/ListMoviesPage"))

  },

  {

    exact: false,

    path: "/detail/:id",

    component: lazy(() => import("containers/HomeTemplate/DetailMoviePage"))

  },

];

const routesAdmin = [

  {

    exact: false,

    path: "/dashboard",

    component: lazy(() => import("containers/AdminTemplate/DashboardPage"))

  },

  {

    exact: false,

    path: "/add-user",

    component: lazy(() => import("containers/AdminTemplate/AddUserPage"))

  },

]

export const renderRoutesHome = () => {

  return routesHome.map((*route*, *index*) => {

    return <*HomeTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

export const renderRoutesAdmin = () => {

  return routesAdmin.map((*route*, *index*) => {

    return <*AdminTemplate*

*key*={*index*}

*exact*={*route*.exact}

*path*={*route*.path}

*component*={*route*.component}

    />

  })

}

Tuy nhiên thời điểm này khi load lại trang web sẽ bị lỗi do bất đồng bộ, react yêu cầu “Add a <Suspense fallback=...> component higher in the tree to provide a loading indicator or placeholder to display.”

=> Bao bọc tất cả những gì thuộc về route bằng thẻ <Suspense> của react với fallback là component loading (hoặc 1 thẻ html nào đó) để thông báo trong thời gian chờ server response về

Trong App.js được viết lại :

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

import LoginPage from "containers/AdminTemplate/LoginPage";

import PageNotFound from "containers/HomeTemplate/PageNotFound";

import { renderRoutesAdmin, renderRoutesHome } from "routes";

import { Suspense } from "react";

import Loading from "components/Loader";

function App() {

  return (

    <*Suspense* *fallback*={<*Loading* />}>

      <*BrowserRouter*>

        <*Switch*>

          {renderRoutesHome()}

          <*Route* *path*="/login" *component*={LoginPage} />

          {renderRoutesAdmin()}

          <*Route* *path*="" *component*={PageNotFound} />

        </*Switch*>

      </*BrowserRouter*>

    </*Suspense*>

  );

}

export default App;

=> Kiểm tra refresh lại F12/Network/JS => khi mở từng trang sẽ nhận lại 1 file JS riêng cho trang đó

**=== LazyLoad của react-lazyload ===**

npm install react-lazyload --save

<https://viblo.asia/p/toi-uu-hoa-ung-dung-react-voi-lazy-loading-YWOZr83w5Q0>

**=== REFACTOR SERVICE (quản lý API : domain, headers, token, …) ===**

VD khi call API, khi domain bị thay đổi => phải sửa code ở nhiều vị trí gọi link API; hoặc vị trí headers của action để xác thực token => refactor lại để chỉ sửa 1 lần khi cần chỉnh sửa => tạo file quản lý api để thay đổi trong các modules actions có gọi API

<https://github.com/axios/axios> (axios document)

- Axios document:

You can create a new instance of axios with a custom config.

*axios.create([config])*

const instance = axios.create({

  baseURL: 'https://some-domain.com/api/',

  timeout: 1000,

  headers: { 'X-Custom-Header': 'foobar' }

});

The available instance methods are listed below. The specified config will be merged with the instance config.

*axios#request(config)*

*axios#get(url[, config])*

*axios#delete(url[, config])*

*axios#head(url[, config])*

*axios#options(url[, config])*

*axios#post(url[, data[, config]])*

*axios#put(url[, data[, config]])*

*axios#patch(url[, data[, config]])*

*axios#getUri([config])*

interceptor của axios có thể hiểu như một bước tường lưới chặn các request, response của ứng dụng để cho phép kiểm tra, thêm vào header hoặc thay đổi các param của request, response. Nó cho phép chúng ta kiểm tra các token ứng dụng, Content-Type hoặc tự thêm các header vào request. Điều này cho phép chúng ta tận dụng tối đa thao tác chỉnh sửa header, body, param request gửi lên server sao cho hợp lý nhất, bảo mật nhất.

axios.interceptors.request.use(function (*config*) {

  // Do something before request is sent

  return *config*;

}, function (*error*) {

  // Do something with request error

  return *Promise*.reject(*error*);

});

// Add a response interceptor

axios.interceptors.response.use(function (*response*) {

  // Any status code that lie within the range of 2xx cause this function to trigger

  // Do something with response data

  return *response*;

}, function (*error*) {

  // Any status codes that falls outside the range of 2xx cause this function to trigger

  // Do something with response error

  return *Promise*.reject(*error*);

});

// If you need to remove an interceptor later you can.

const myInterceptor = axios.interceptors.request.use(function () {/\*...\*/ });

axios.interceptors.request.eject(myInterceptor);

// You can add interceptors to a custom instance of axios.

const instance = axios.create();

instance.interceptors.request.use(function () {/\*...\*/ });

Áp dụng vào project :

- Trong folder src tạo utils/apiUtils.js để quản lý các vấn đề liên quan axios (import axios)

import axios from "axios";

const api = axios.create({

  baseURL: "https://movienew.cybersoft.edu.vn/api/",

  headers: {

    Authorization: localStorage.getItem("UserInfo") ? `Bearer ${JSON.parse(localStorage.getItem("UserInfo")).accessToken}` : "",

    TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20"

  }

});

api.interceptors.request.use((*config*) => {

  console.log("Message from apiUtils.js before call API");

  return *config*;

});

// Hoặc có thể viết lại

// const api = axios.create({

//   baseURL: "https://movienew.cybersoft.edu.vn/api/",

// });

// api.interceptors.request.use((config) => {

//   config.headers = {

//     Authorization: localStorage.getItem("UserInfo") ? `Bearer ${JSON.parse(localStorage.getItem("UserInfo")).accessToken}` : "",

//     TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

//   };

//   return config;

// });

export default api;

//Lúc này api có thể thay thế cho phương thức call api của axios({...})

// Cách sử dụng:

// api.get(...)

// api.put(...)

// api.post(...)

// api.delete(...)

// ...

- Ví dụ về sự thay đổi của file actions.js trong ListMoviePage được viết lại (tương tự cho những trang khác) :

Ban đầu :

import axios from 'axios';

import \* as ActionType from './constants';

export const actFetchListMovies = () => {

  return (*dispatch*) => {

    dispatch(actListMoviesRequest());

    axios({

      url: "https://movienew.cybersoft.edu.vn/api/QuanLyPhim/LayDanhSachPhim?maNhom=GP02",

      method: "GET",

      headers: {

        TokenCybersoft: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5Mb3AiOiJCb290Y2FtcCAyMCIsIkhldEhhblN0cmluZyI6IjE3LzA3LzIwMjIiLCJIZXRIYW5UaW1lIjoiMTY1ODAxNjAwMDAwMCIsIm5iZiI6MTYyNjcxNDAwMCwiZXhwIjoxNjU4MTYzNjAwfQ.CyAnnc8e2Rp7YmuJCdtEj-Wp7RvlDenB9Dad6NV0R20",

      }

    })

      .then(*result* => {

        dispatch(actListMoviesSuccess(*result*.data.content));

      })

      .catch(*error* => {

        dispatch(actListMoviesFailed(*error*));

      });

  }

}

const actListMoviesRequest = () => {

  return {

    type: ActionType.LIST\_MOVIES\_REQUEST,

  }

};

const actListMoviesSuccess = (*data*) => {

  return {

    type: ActionType.LIST\_MOVIES\_SUCCESS,

    payload: *data*

  }

};

const actListMoviesFailed = (*error*) => {

  return {

    type: ActionType.LIST\_MOVIES\_FAILED,

    payload: *error*

  }

};

Sau khi refactor :

import \* as ActionType from './constants';

import api from 'utils/apiUtils';

export const actFetchListMovies = () => {

  return (*dispatch*) => {

    dispatch(actListMoviesRequest());

    api.get("QuanLyPhim/LayDanhSachPhim?maNhom=GP02")

      .then(*result* => {

        dispatch(actListMoviesSuccess(*result*.data.content));

      })

      .catch(*error* => {

        dispatch(actListMoviesFailed(*error*));

      });

  }

}

const actListMoviesRequest = () => {

  return {

    type: ActionType.LIST\_MOVIES\_REQUEST,

  }

};

const actListMoviesSuccess = (*data*) => {

  return {

    type: ActionType.LIST\_MOVIES\_SUCCESS,

    payload: *data*

  }

};

const actListMoviesFailed = (*error*) => {

  return {

    type: ActionType.LIST\_MOVIES\_FAILED,

    payload: *error*

  }

};

=== Deploy bằng github ===

npm install gh-pages –save

Trong package.json thêm đường dẫn homepage và thêm command predeploy và deploy app

{

  "name": "demo-reactjs-movie",

  "version": "0.1.0",

  "private": true,

  "homepage": "https://huynhanhmtu.github.io/demo-reactjs-movie/",

  "dependencies": ...

  "scripts": {

    "predeploy":"npm run build",

    "deploy": "gh-pages -d build",

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },...

npm run deploy

Trong repositories của github => vào project => settings => pages => source => gh-pages => save (có thể khi deploy lên do đường dẫn ban đầu sẽ khác nên bị chuyển sang trang page not found => thêm route là đường dẫn lúc deploy để trỏ đến trang HomePage)