Bootcamp React 3.0 - Cap. 09

Autenticação e autorização

# Competências

* Formulários
  + React Hook Form
  + Validação de formulário, expressões regulares
  + Mensagens de erro e estilização condicional
* Login OAuth2
* Interceptors
* LocalStorage
  + Acesso a dados
  + JSON parse / stringify
* Estado global com Context API
* Fluxos de autenticação e autorização
  + Rotas protegidas
  + Redirecionamentos de login e de autorização
  + Redirecionamentos especiais para experiência do usuário (UX)
  + Permissionamento em nível de rotas
  + Restrição de conteúdo (UI) baseada em perfil de usuário

# Figma do DSCatalog

<https://www.figma.com/file/cNa2l3TqZXxbU6NBDPruNw/BDS-DSCatalog>

# Planejamento das rotas

| **ROTA** | **Página (componente)** |
| --- | --- |
| / | Home |
| /products | Catalog |
| /products/:productId | ProductDetails |
| /admin | Admin/Products (redirecionamento) |
| **/admin**/auth | **Admin/**Auth/Login (redirecionamento) |
| **/admin**/auth/login | **Admin/**Auth/Login |
| /admin/products | Admin/Products/List |
| /admin/products/create | Admin/Products/Form |
| /admin/products/:productId | Admin/Products/Form |

## Card de login

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

import ButtonIcon from 'components/ButtonIcon';

import './styles.css';

const Login = () => {

return (

<div className="base-card login-card">

<h1>LOGIN</h1>

<form>

<div className="mb-4">

<input

type="text"

className="form-control base-input"

placeholder="Email"

name="username"

/>

</div>

<div className="mb-2">

<input

type="password"

className="form-control base-input "

placeholder="Password"

name="password"

/>

</div>

<Link to="/admin/auth/recover" className="login-link-recover">

Esqueci a senha

</Link>

<div className="login-submit">

<ButtonIcon text="Fazer login" />

</div>

<div className="signup-container">

<span className="not-registered">Não tem Cadastro?</span>

<Link to="/admin/auth/register" className="login-link-register">

CADASTRAR

</Link>

</div>

</form>

</div>

);

};

export default Login;

## Base input

.base-input {

border-radius: 10px;

font-size: 1em;

letter-spacing: -0.015em;

color: #263238;

height: 50px;

}

.base-input::placeholder {

color: #9E9E9E;

}

## Detalhes do card de login

.login-link-recover {

font-size: 1em;

letter-spacing: -0.015em;

color: #407bff;

display: block;

margin-bottom: 90px;

}

.login-link-register {

font-weight: bold;

font-size: 1em;

text-align: center;

letter-spacing: -0.015em;

text-decoration-line: underline;

color: #407bff;

margin-left: 10px;

}

.login-submit {

margin-bottom: 10px;

}

.not-registered {

font-weight: bold;

font-size: 1em;

text-align: center;

letter-spacing: -0.015em;

color: #9e9e9e;

}

.signup-container {

text-align: center;

margin-bottom: 30px;

}

## Tipo AxiosRequestConfig

<https://github.com/axios/axios/blob/master/index.d.ts>

## Biblioteca Query String

yarn add qs @types/qs

## Expressão regular JS para email

/^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i

## LocalStorage

Inspect -> Application

SessionStorage: escopo por aba

LocalStorage: escopo por domínio

\* Somente a aplicação do domínio consegue acessar os dados programaticamente

### Acessando valores no localStorage

localStorage.getItem(<chave>);

localStorage.setItem(<chave>; <valor>);

### Convertendo objeto para string

var pessoa = '{"name": "Maria"}'

var obj = JSON.parse(str)

var str = JSON.stringify(obj)

## Componente de busca de usuários (apenas teste)

import { AxiosRequestConfig } from 'axios';

import { useEffect, useState } from 'react';

import { User } from 'types/user';

import { SpringPage } from 'types/vendor/spring';

import { requestBackend } from 'util/requests';

const Users = () => {

const [page, setPage] = useState<SpringPage<User>>();

useEffect(() => {

const params : AxiosRequestConfig = {

url: '/users',

withCredentials: true,

params: {

page: 0,

size: 12,

},

};

requestBackend(params).then((response) => {

setPage(response.data);

});

}, []);

return (

<div>

{page?.content.map((item) => (

<p key={item.id}>{item.email}</p>

))}

</div>

);

};

export default Users;

## Axios interceptors

<https://github.com/axios/axios>

## React Router Auth Workflow

<https://reactrouter.com/native/example/auth-workflow>

## Componente PrivateRoute

Adaptado de:

<https://reactrouter.com/native/example/auth-workflow>

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

import { isAuthenticated } from 'util/requests';

type Props = {

children: React.ReactNode;

path: string;

};

const PrivateRoute = ({ children, path }: Props) => {

return (

<Route

path={path}

render={() =>

isAuthenticated() ? children : <Redirect to="/admin/auth/login" />

}

/>

);

};

export default PrivateRoute;

## Verificar se o token é válido

* Decodificar o token (pode lançar exceção)
* Verificar se o token não está expirado

## JWT Decoder

yarn add jwt-decode @types/jwt-decode

## Estado global com Context API do React

* Criar o contexto global
* Prover o contexto global
  + Crie um useState no App.tsx
* Acessar o contexto global nos componentes

# Desafio para entregar

**TAREFA: MovieFlix front inicial**

Construa um pequeno app conforme design Figma e os requisitos abaixo. Assista o vídeo explicativo.

## Design Figma

<https://www.figma.com/file/Yul1TnqQXmBsLwm5h5nvJu/MovieFlix-front-inicial>

## Back end

| **Atenção:** este app deverá se comunicar com o seu back end desenvolvido na tarefa **“MovieFlix casos de uso”** do Bootcamp Spring. Se você não fez o Bootcamp Spring, você pode utilizar o back end de referência da escola, publicado no Heroku. Assista o vídeo explicativo.  **Back end Movieflix de referência no Heroku (modo gratuito - leva 1 minuto para subir):**  <https://movieflix-devsuperior.herokuapp.com>  **Collection Postman do back end Movieflix:**  <https://www.getpostman.com/collections/6819494cd1edca40a064>  **Environment no Postman:**  host: **https://movieflix-devsuperior.herokuapp.com**  client-id: **myclientid**  client-secret: **myclientsecret**  username: **bob@gmail.com (VISITOR) ou ana@gmail.com (MEMBER)**  password: **123456**  token |
| --- |

## Requisitos

* Ao acessar a rota raiz do app, deverá aparecer uma **tela de login**.
* Ao fazer login com sucesso, o usuário deverá ser redirecionado para a rota /movies que corresponde à **tela de listagem de filmes**.
* Durante todo o tempo em que o usuário estiver logado, deverá aparecer na barra de navegação um botão “Sair” que deverá realizar o logout do usuário e redirecioná-lo para a tela de login.
* A **tela de listagem de filmes** deve exibir apenas um título, e dois links para os detalhes dos filmes de código 1 e 2 respectivamente.
* Ao clicar em um dos links dos detalhes do filme, o usuário deve ser levado para a rota /movies/:movieId, que corresponde à tela de detalhes do filme.
* Na **tela de detalhes do filme**, caso o usuário possua o perfil **MEMBER**, deve ser exibido um card com um formulário para que o usuário possa entrar com uma avaliação do filme. Se o usuário não possuir o perfil de MEMBER, este card não deve ser exibido.
* Na **tela de detalhes do filme**, abaixo do card de formulário, deve aparecer um outro card com uma listagem de TODAS avaliações daquele filme *(não esqueça o atributo “key” em cada item renderizado)*. Quando um usuário MEMBER informar com sucesso uma nova avaliação, esta listagem deve ser atualizada.
* As telas **listagem de filmes** e **detalhes do filme** devem ser rotas protegidas. Se o usuário não autenticado tentar acessar diretamente essas rotas, ele deve ser redirecionado para a **tela de login**.

## Sugestão de passos para implementar o projeto:

1) Faça o layout da tela de login, respondendo pela rota /

2) Faça a requisição de autenticação e o redirecionamento para a rota /movies

3) Faça também o tratamento para o login inválido

4) Faça a parte de salvar os dados de autenticação no localStorage e das requisições autorizadas, e faça a listagem de filmes (requisição autorizada)

5) Faça a parte de redirecionar para tela de login caso seja acessada uma rota não autorizada

6) Implemente o botão Sair com Context API

7) Faça a tela de detalhes do filme, inicialmente somente com a listagem das avaliações

8) Faça card de formulário de avaliação, e só apresente o mesmo se o usuário logado tiver o perfil MEMBER

9) Implemente a validação do formulário (não aceitar o campo em branco)

10) Implemente o envio do formulário de avaliação e a atualização da listagem de avaliações após o envio do formulário