ROTEIRO PARA CRIAR AUTENTICAÇAO DE UMA API REST COM JWT

1. Com a mesma API criada anteriormente, vamos alterar somente o arquivo index.html incluindo inputs para o usuário inserir o email e a senha para login
2. No arquivo index.html digite :
3. h2>Login</h2>
4. <input *type*="text" *name*="email" *id*="email" *placeholder*="email@teste.com.br">
5. <input *type*="text" *name*="password" *id*="password" *placeholder*="Digite a senha">
6. <button *type*="button" *onclick*="login()">Entrar</button>
7. Dentro do campo de script logo abaixo da tag </body> vamos criar uma variável de configuração do axios, para guardar a informação no headers: authorization do token que foi salvo no localstorage. Essa variável será utilizada na autenticação do usuário para acesso a API.
8. //variavel com configuração do axios
9. var axiosConfig = {
10. headers: {
11. authorization: "Bearer " + localStorage.getItem("token")
12. }
13. }
14. Criamos uma função login() que será chamada no evento onclik() do botão ‘Entrar’.
15. A função login() irá capturar os valores dos inputs de email e password esses valores serão passados como parâmetro para uma rota de autenticação do axios “axios.post(<http://localhost:4040/auth>,{ email, password}
16. Nessa rota a função auth no back-end (ou seja no arquivo index.js que deverá estar rodando) irá verificar se o email e password estão cadastrados e neste casa será gerado um token. Esse token será passado por response. Então o valor do response que é o token será capturado na função login e será salvo como novo item no localStorage.
17. //função para validar o login
18. function login(){
19. var emailField = document.getElementById('email');
20. var passwordField = document.getElementById('password');
21. var email = emailField.value;
22. var password = passwordField.value;
24. //fazendo uma requisiçao via axios para a rota auth passando os
25. //dados do usuário
26. axios.post("http://localhost:4040/auth", {
27. email,
28. password
29. }).then(*res* =>{
30. //verificando oque a requisição está trazendo
31. //console.log(res.data);
32. //carregando o token na variavel token que será utilizada
33. //para autenticação
34. var token = *res*.data.token;
35. //local onde é salvo a informação
36. //a informação somente é apagada quando o histórioco é limpo ou o cache do navegador
37. //cada site possui um localstorage diferente
38. //criando um item
39. localStorage.setItem("token",token);
40. axiosConfig.headers.authorization = "Bearer " + localStorage.getItem("token");
41. // alert("Logado");
42. }).catch(*err* =>{
43. alert("Login incorreto....")
44. });
45. }
46. Após a criação da função login() e a configuração da variável axiosConfig, basta adicionar a variável axiosConfig como parâmetro em todas as rotas que devem possuir autenticação.EX:
47. /carregando itens na pagina
48. //o axios irá executar a url para buscar todos os games
49. axios.get('http://localhost:4040/games', axiosConfig ).then(*response* =>{

52. //rota para cadastrar um novo game, basta informar a rota e os dados
53. axios.post("http://localhost:4040/game", game, axiosConfig).then(*response*=>{
55. //rota utilizando o axios para deletar o item
56. axios.delete("http://localhost:4040/game/"+id, axiosConfig).then(*response*=>{
58. //utilizano o axios para abrir a api e editar o game
59. axios.put("http://localhost:4040/game/"+id, game, axiosConfig).then(*response* =>{
60. alert("Game