ROTEIRO PARA CRIAR AUTENTICAÇAO DE UMA API REST COM JWT

1. Criar sistema de login da API
2. NA pasta APIDEGAMES
3. Abrir o arquivo index.js
4. Inserir no banco de dados “uma tabela de usuários” para realizar a autenticação
5. users:[
6. {
7. id:1,
8. email:"adm@adm.com.br",
9. nome:"admin",
10. senha:"12345"
11. },
12. {
13. id:2,
14. email:"claudisnei@teste.com",
15. nome:"Claudisnei",
16. senha:"teste123"
17. }
18. ]
19. Criar uma rota para autencicação ‘/auth’ a autenticação do JWT utiliza um token. A rota pega o email passado por requisição e verifica se o email está cadastrado no banco de dados.
20. /\*\*rota para autenticação \*/
21. app.post("/auth",(*req*,*res*)=>{
22. var {email, password} = *req*.body;
23. //validações
24. if(email != undefined){
25. //Verificando se o usuário é cadastrado
26. var user = DB.users.find(*us* => *us*.email == email);
27. if(user != undefined){
28. if(user.password === password){
29. *res*.status(200);
30. *res*.json({token: "Token provisório!"});
31. }else{
32. *res*.status(401);
33. *res*.json({err: "Email ou senha inválidos"});
34. }
35. }else{
36. *res*.status(404);
37. *res*.json({err: "Email ou senha inválidos"});
38. }
39. }else{
40. *res*.status(400);
41. *res*.json({err: "O email enviado é inválido."});
42. }
43. });
44. Para gerar o token temos que instalar a biblioteca jsonwebtoken
45. ‘npm install –save jsonwebtoken’
46. Criar uma variável para instanciar a jwt
47. Criar uma chave secreta que será utilizada para gerar o token
48. //instanciando a biblioteca jsonwebtoken
49. const jwt = require("jsonwebtoken");
50. //criando uma chave secreta para o jwt
51. const secretKeyJwt = "ffhhae@#$¨%fdd!@#4";
52. Gerar o token
53. Para gerar o token é utilizado a função ‘sign’ do JWT, nesta função devemos Incluir as informações que se quer carregar no token, aqui estamos passando três JSON que são: o id e email do usuário, a chave secreta que criamos, e o tempo de duração do token.
54. /\*\*deve-se passar as informações necessárias
55. \* , a chave secreta e o prazo que irá expirar
56. \* o token
57. \*/
58. jwt.sign(
59. {
60. id: user.id,
61. email: user.email
62. },
63. secretKeyJwt,
64. {expiresIn:'48h'},(*err*, *token*)=>{
65. if(*err*){
66. *res*.status(400);
67. *res*.json({err: "Falha interna"});
68. }else{
69. *res*.status(200);
70. *res*.json({token: *token*});
71. }
72. }
73. )
74. Protegendo as rotas através dos tokens
75. Vamos trabalhar com midlewares
76. Vamos criar um midleware

Token gerado: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJhZG1AYWRtLmNvbS5iciIsImlhdCI6MTYzMTc1NDMxMywiZXhwIjoxNjMxOTI3MTEzfQ.Es7isxdkYhbQa7IcalBxS69a97Yv4f6P55\_Y7-DZfSM

1. //midleware para realizar a autenticação antes da rota ser executada
2. function auth(*req*, *res*, *next*){
3. //pegando o header de autorização
4. const authToken = *req*.headers['authorization']
5. //validando o token
6. if(authToken !=undefined){
7. //cortando o token
8. const bearer = authToken.split(' ');
9. //pegando somente o token
10. var token = bearer[1];
11. //verificando se o token é valido
12. jwt.verify(token,secretKeyJwt,(*err*, *data*)=>{
13. if(*err*){
14. *res*.status(401);
15. *res*.json({err:"Token inválido!"});
16. }else{
17. // caso o token seja valido.
18. //passando variáveis para a requisição
19. //, estes dados poderão ser acessados
20. //nas rotas que forem autenticadas
21. *req*.token = token;
22. *req*.logedUser = {id: *data*.id, email: *data*.email};
23. //console.log(*data*);
24. next();
25. }
26. });
27. // console.log(bearer);
28. }else{
29. *res*.status(401);
30. *res*.json({err:"Token inválido"});
31. }
32. //console.log(authToken);
33. //se autenticação foi realizada com sucesso a função next que irá
34. //realizar a requisição do usuário
36. }
37. Incluir nas rotas o midleware para realizar a autenticação

Ex: app.get('/games',auth,(*req*,*res*)=>{

*res*.statusCode = 200;

*res*.json(DB.games);

});

1. Onde auth é o midleware