ROTEIRO PARA CRIAÇÃO DE UMA API

1. Baixar o POSTMAN
2. Crie um cadastro na página do Postman
3. Instale o Postman
4. CRIAR UMA API NO VS CODE
5. Criar uma pasta chamada APIDEGAMES
6. Criando a API com o Express
7. Na pasta do projeto crie um novo projeto node digite:
8. ‘npm init’
9. Baixar a biblioteca do Express:
10. ‘npm install express –save’
11. Baixar a biblioteca do body-parser:
12. ‘npm install body-parser –save’
13. Crie o arquivo principal ‘Index.js’
14. No arquivo Index.js
15. Baixe as bibliotecas do express e do body-parser
16. //importando as bibliotecas
17. const express = require("express");
18. const bodyParser = require("body-parser");
19. Criando instancia do express
20. //criando o app
21. const app =  express();
22. iniciando o express com a configuração básica
23. app.use(express.urlencoded({extended:false}));
24. app.use(express.json());
25. Abrir o servidor em uma porta e criar uma mensagem no call-back, lembrando que este código deve sempre estar no final da página.
26. app.listen(45678,()=>{
27. console.log("API rodando")
28. });
29. Criando um banco de dados falso para manipular a API, neste caso vamos utilizar um JSON, porém após a criação da API é recomendado criar um BD com o sequealize para testar.
30. var DB = {
31. games:[
32. {
33. id:23,
34. title: "call of duty MW",
35. year: 2019,
36. price:60
37. }
38. {
39. id:65,
40. title: "Sea of thieves",
41. year: 2018,
42. price:40
43. }
44. {
45. id:2,
46. title: "Minecraft",
47. year: 2012,
48. price:20
49. }
50. ]
51. }
52. Criando o primeiro ‘end point’, que são basicamente rotas na API.
53. Vamos utilizar o verto GET, sendo possível utilizar outros verbos DELETE, PUT
54. Mas deve-se utilizar a lógica de dar nomes corretos na rota segundo os critérios de uma API
55. Deve-se também informar o status Code em todas as rotas de um API isso é um padrão, alista de status code pode ser vista no google.
56. /\*\*primeira rota no primeiro end point que deverá listar todos
57. \* os games do sistema
58. \*/
59. app.get('/games',(*req*,*res*)=>{
60. *res*.status(200);
61. *res*.json(DB.games);
62. });
63. Rodar a aplicação no console
64. ‘nodemon index.js’
65. Caso apresente um erro dizendo que o nodemon.ps1 não pode ser carregado porque a execução do script foi desabilitada neste sistema. Utilize o comando
66. ‘npx nodemon index.js’
67. Abra o navegador com o endereço localhost:4040/games , que foi a porta configurada anteriormente e rota criada.
68. CRIANDO A LISTAGEM DE REGISTRO ÚNICO DE UM GAME E VALIDAÇÃO
69. Rota para buscar um game pelo ID
70. /\*\*rota para devolver ao usuário o game pelo ID \*/
71. app.get('/game/:id',(*req*,*res*)=>{
73. //tratando o id verificando se é numero
74. if(isNaN(*req*.params.id)){
75. *res*.sendStatus(400);
76. }else{
77. var id = parseInt(*req*.params.id);
79. var game = DB.games.find( *game* =>  *game*.id == id );
81. if(game != undefined){
82. *res*.statusCode= 200;
83. *res*.json(game);
84. }else{
85. *res*.sendStatus(404);
86. }
87. }
88. });
89. UTILIZANDO O POSTMAN PARA TESTE DE API’S
90. Digite a url localhost:4040/games na postman selecione o método GET e clique em send
91. Serão exibidos todos os registros no campo body do postman
92. CADASTRANDO DADOS
93. Criando um novo endpoint ou rota para salvar dados
94. Digite
95. /\*\*novo endpoint para cadastrar um novo game \*/
96. app.post('/game',(*req*,*res*)=>{
97. /\* isso dá o mesmo resultado da destruct
98. var title = req.body.title;
99. var year = req.body.year;
100. var price = req.body.price;\*/
101. //utilizando a destruct para facilitar a manipulação de dados
102. var {title, year, price} = *req*.body;
103. //o ideal é validar todos os dados conforme as regras no negocio
104. //e retornar o status code 400 caso não se enquadre nas regras
106. //utilizando o método push que serve para adicionar dados dentro de um array
107. DB.games.push({
108. id:2323,
109. title,
110. year,
111. price
112. })
113. //retornando o status
114. *res*.sendStatus(200);
115. });
116. No post é possível realizar a requisição post sem ter um formulário
117. Basta ir na aba Body do postman selecionar a opção raw, selecionar o formato JSON.
118. Digitar em forma de json o dado que se quer cadastrar e clicar em Send.
119. {
120. “title”:”CS:GO”,
121. “price”: 60,
122. “year”: 2012
123. }
124. Será adicionado o json na variável DB.games .
125. DELETANDO DADOS EM UMA API REST
126. /\*\*novo endpoint para deletar um jogo \*/
127. app.delete('/game/:id',(*req*,*res*)=>{
128. /\*\*validação \*/
129. if(isNaN(*req*.params.id)){
130. *res*.sendStatus(400);
131. }else{
132. var id = parseInt(*req*.params.id);
133. /\*\*se o index for 0 ou outro numero ele existe
134. \* se for -1 ele não existe
135. \*/
136. var index = DB.games.findIndex( *g*=> *g*.id == id);
138. if(index == -1){
139. *res*.sendStatus(404);
140. }else{
141. /\*\*utiliza-se splice para deletar um registro
142. \* em um array. Os parâmetros são o indice do elemento
143. \* que será apagao e a quantidade de itens que serão
144. \* apagados apartir deste indice
145. \*/
146. DB.games.splice(index, 1);
147. *res*.sendStatus(200);
148. }
149. }
150. });
151. NO POSTMAN
152. Digite a url localhost/game/:id, onde o :id será o id do jogo que será deletado. Selecione o verbo DELETE
153. Clique em send, o jogo será apagado. Será retornado uma mensagem, que configuramos na API
154. EDIÇÃO DE DADOS EM UMA API REST
155. A edição pode ser feito utilizado três métodos PUT, PUSH e PATCH
156. No index.js
157. /\*\*endpois para edição de registros será criado uma rota
158. \* utilizando o PUT
159. \*/
160. app.put('/game/id',(*req*,*res*)=>{
161. /\*\*validação \*/
162. if(isNaN(*req*.params.id)){
163. *res*.sendStatus(400);
164. }else{
165. var id = parseInt(*req*.params.id);
166. /\*\*localiza o jogo pelo id
167. \*/
168. var game = DB.games.find( *g*=> *g*.id == id);
170. if(game != undefined){
171. //peqando os dados que estão na requisição
172. var {title, price, year} = *req*.body;
173. /\*\* ideal é fazer as validações conforme a regra
174. \* do negócio
175. \*/
176. if(title !=undefined){
177. game.title = title;
178. }
179. if(price !=undefined){
180. game.price = price;
181. }
182. if(year !=undefined){
183. game.year = year;
184. }
185. *res*.sendStatus(200);
186. }else{
187. *res*.sendStatus(404);
188. }
189. }
190. });
191. NO POSTMAN
192. Digite a url localhost/game/:id, onde o :id será o id do jogo que será editado. Selecione o verbo PUT
193. Vá na aba Body selecionar a opção raw, selecionar o formato JSON.
194. Digitar em forma de json os dados do jogo que se quer editar e clicar em Send.
195. {
196. “title”:”CS:GO”,
197. }