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 statusCode 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*.statusCode = 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:45678/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 na postman
91. CADASTRANDO DADOS
92. Criando um novo endpoint para salvar dados
93. Digite
94. /\*\*novo endpoint para cadastrar um novo game \*/
95. app.post('/game',(*req*,*res*)=>{
96. /\* isso dá o mesmo resultado da destruct
97. var title = req.body.title;
98. var year = req.body.year;
99. var price = req.body.price;\*/
100. //utilizando a destruct para facilitar a manipulação de dados
101. var {title, year, price} = *req*.body;
102. //o ideal é validar todos os dados conforme as regras no negocio
103. //e retornar o status code 400 caso não se enquadre nas regras
105. //utilizando o método push que serve para adicionar dados dentro de um array
106. DB.games.push({
107. id:2323,
108. title,
109. year,
110. price
111. })
112. //retornando o status
113. *res*.sendStatus(200);
114. });
115. No post é possível realizar a requisição post sem ter um formulário
116. Basta ir na aba Body do postman clicar selecionar a opção raw, selecionar o formato JSON.
117. Digitar em forma de json o dado que se quer cadastrar e clicar em Send.
118. {
119. “title”:”CS:GO”,
120. “price”: 60,
121. “year”: 2012
122. }
123. DELETANDO DADOS EM UMA API REST
124. /\*\*novo endpoint para deletar um jogo \*/
125. app.delete('/game/:id',(*req*,*res*)=>{
126. /\*\*validação \*/
127. if(isNaN(*req*.params.id)){
128. *res*.sendStatus(400);
129. }else{
130. var id = parseInt(*req*.params.id);
131. /\*\*se o index for 0 ou outro numero ele existe
132. \* se for -1 ele não existe
133. \*/
134. var index = DB.games.findIndex( *g*=> *g*.id == id);
136. if(index == -1){
137. *res*.sendStatus(404);
138. }else{
139. /\*\*utiliza-se splice para deletar um registro
140. \* em um array. Os parâmetros são o indice do elemento
141. \* que será apagao e a quantidade de itens que serão
142. \* apagados apartir deste indice
143. \*/
144. DB.games.splice(index, 1);
145. *res*.sendStatus(200);
146. }
147. }
148. });
149. NO POSTMAN
150. Execute um request para a rota /game utilizando o método DELETE
151. Passando o id do jogo que se quer apagar ou seja /game/id
152. EDIÇÃO DE DADOS EM UMA API REST
153. A edição pode ser feito utilizado três métodos PUT, PUSH e PATCH
154. No index.js
155. /\*\*endpois para edição de registros será criado uma rota
156. \* utilizando o PUT
157. \*/
158. app.put('/game/id',(*req*,*res*)=>{
159. /\*\*validação \*/
160. if(isNaN(*req*.params.id)){
161. *res*.sendStatus(400);
162. }else{
163. var id = parseInt(*req*.params.id);
164. /\*\*localiza o jogo pelo id
165. \*/
166. var game = DB.games.find( *g*=> *g*.id == id);
168. if(game != undefined){
169. //peqando os dados que estão na requisição
170. var {title, price, year} = *req*.body;
171. /\*\* ideal é fazer as validações conforme a regra
172. \* do negócio
173. \*/
174. if(title !=undefined){
175. game.title = title;
176. }
177. if(price !=undefined){
178. game.price = price;
179. }
180. if(year !=undefined){
181. game.year = year;
182. }
183. *res*.sendStatus(200);
184. }else{
185. *res*.sendStatus(404);
186. }
187. }
188. });
189. NO POSTMAN
190. Execute um request para a rota /game/id utilizando o método PUT
191. Passando o id do jogo que se quer editar.
192. Selecione o método PUT
193. Vá na aba Body selecionar a opção raw, selecionar o formato JSON.
194. Digitar em forma de json o dado que se quer editar e clicar em Send.
195. {
196. “title”:”CS:GO”,
197. }