

GAME'S

BACKEND COM

NODE JS

API'S



PROCESSOS

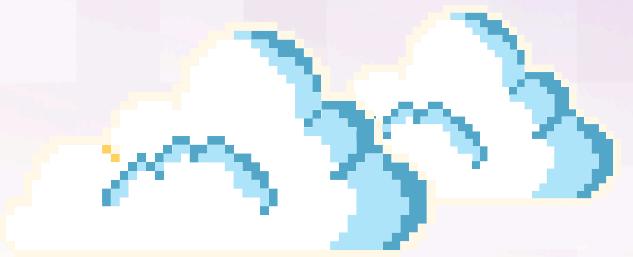
CRIAÇÃO DO
SERVER.JS E
SUPABASE

INSTALAÇÃO DE
PACOTES

INICIALIZAÇÃO DO
SERVIDOR E
CRIAÇÃO DE DADOS

PRIMEIRO PASSO

EU CRIE UM PASTA NA
ÁREA DE TRABALHO

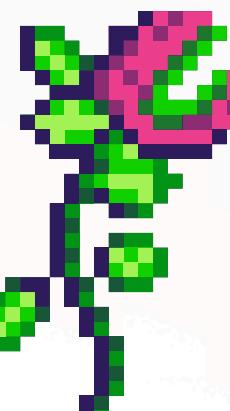


SEGUNDO PASSO



ABRI O VS CODE A ABRIR A
PASTA NELE

YAY



TERCEIRO PASSO

CRIE UMA PASTA BACKEND DENTRO DA PASTA JÁ CRIADA E CRIE UM ARQUIVO SERVER.JS E COLOQUEI ESSE CÓDIGO

```
1 const express = require('express');
2 const cors = require('cors');
3 require('dotenv').config();
4 const { createClient } = require('@supabase/supabase-js');

5
6 const app = express();
7 const PORT = process.env.PORT || 3000;

8
9 // Middleware
10 app.use(cors());
11 app.use(express.json());

12
13 // Inicializar Supabase
14 const supabase = createClient(
15   process.env.SUPABASE_URL,
16   process.env.SUPABASE_KEY
17 );
18
19 // Rota teste
20 app.get('/', (req, res) => {
21   res.json({ mensagem: 'API rodando!' });
22 });

23
24 // GET - Buscar todos os produtos
25 app.get('/produtos', async (req, res) => {
26   try {
27     const { data, error } = await supabase
28       .from('produtos')
29       .select('*');
30     if (error) throw error;
31     res.json(data);
32   } catch (error) {
33     res.status(400).json({ erro: error.message });
34   }
35 });

36
37 // POST - Criar um novo produto
38 app.post('/produtos', async (req, res) => {
39   const { nome, descricao, preco } = req.body;
40   const { data, error } = await supabase
41     .from('produtos')
42     .insert([{ nome, descricao, preco }]);
43   if (error) throw error;
44   res.json(data);
45 });

46
47 // PUT - Atualizar um produto
48 app.put('/produtos/:id', async (req, res) => {
49   const { id } = req.params;
50   const { nome, descricao, preco } = req.body;
51   const { data, error } = await supabase
52     .from('produtos')
53     .update({ nome, descricao, preco })
54     .eq('id', id);
55   if (error) throw error;
56   res.json(data);
57 });

58
59 // DELETE - Deletar um produto
60 app.delete('/produtos/:id', async (req, res) => {
61   const { id } = req.params;
62   const { data, error } = await supabase
63     .from('produtos')
64     .delete()
65     .eq('id', id);
66   if (error) throw error;
67   res.json(data);
68 });

69
70 // Rota para gerar um token de acesso
71 app.get('/token', async (req, res) => {
72   const { data, error } = await supabase
73     .from('tokens')
74     .select('*');
75   if (error) throw error;
76   res.json(data);
77 });

78
79 // Rota para gerar um token de acesso
80 app.get('/tokens', async (req, res) => {
81   const { data, error } = await supabase
82     .from('tokens')
83     .select('*');
84   if (error) throw error;
85   res.json(data);
86 });

87
88 // Rota para gerar um token de acesso
89 app.get('/tokens/:id', async (req, res) => {
90   const { id } = req.params;
91   const { data, error } = await supabase
92     .from('tokens')
93     .select('*')
94     .eq('id', id);
95   if (error) throw error;
96   res.json(data);
97 });

98
99 // Rota para gerar um token de acesso
100 app.get('/tokens/:id', async (req, res) => {
101   const { id } = req.params;
102   const { data, error } = await supabase
103     .from('tokens')
104     .select('*')
105     .eq('id', id);
106   if (error) throw error;
107   res.json(data);
108 });

109
110 // Rota para gerar um token de acesso
111 app.get('/tokens/:id', async (req, res) => {
112   const { id } = req.params;
113   const { data, error } = await supabase
114     .from('tokens')
115     .select('*')
116     .eq('id', id);
117   if (error) throw error;
118   res.json(data);
119 });

120
121 // Rota para gerar um token de acesso
122 app.get('/tokens/:id', async (req, res) => {
123   const { id } = req.params;
124   const { data, error } = await supabase
125     .from('tokens')
126     .select('*')
127     .eq('id', id);
128   if (error) throw error;
129   res.json(data);
130 });

131
132 // Rota para gerar um token de acesso
133 app.get('/tokens/:id', async (req, res) => {
134   const { id } = req.params;
135   const { data, error } = await supabase
136     .from('tokens')
137     .select('*')
138     .eq('id', id);
139   if (error) throw error;
140   res.json(data);
141 });

142
143 // Rota para gerar um token de acesso
144 app.get('/tokens/:id', async (req, res) => {
145   const { id } = req.params;
146   const { data, error } = await supabase
147     .from('tokens')
148     .select('*')
149     .eq('id', id);
150   if (error) throw error;
151   res.json(data);
152 });

153
154 // Rota para gerar um token de acesso
155 app.get('/tokens/:id', async (req, res) => {
156   const { id } = req.params;
157   const { data, error } = await supabase
158     .from('tokens')
159     .select('*')
160     .eq('id', id);
161   if (error) throw error;
162   res.json(data);
163 });

164
165 // Rota para gerar um token de acesso
166 app.get('/tokens/:id', async (req, res) => {
167   const { id } = req.params;
168   const { data, error } = await supabase
169     .from('tokens')
170     .select('*')
171     .eq('id', id);
172   if (error) throw error;
173   res.json(data);
174 });

175
176 // Rota para gerar um token de acesso
177 app.get('/tokens/:id', async (req, res) => {
178   const { id } = req.params;
179   const { data, error } = await supabase
180     .from('tokens')
181     .select('*')
182     .eq('id', id);
183   if (error) throw error;
184   res.json(data);
185 });

186
187 // Rota para gerar um token de acesso
188 app.get('/tokens/:id', async (req, res) => {
189   const { id } = req.params;
190   const { data, error } = await supabase
191     .from('tokens')
192     .select('*')
193     .eq('id', id);
194   if (error) throw error;
195   res.json(data);
196 });

197
198 // Rota para gerar um token de acesso
199 app.get('/tokens/:id', async (req, res) => {
200   const { id } = req.params;
201   const { data, error } = await supabase
202     .from('tokens')
203     .select('*')
204     .eq('id', id);
205   if (error) throw error;
206   res.json(data);
207 });

208
209 // Rota para gerar um token de acesso
210 app.get('/tokens/:id', async (req, res) => {
211   const { id } = req.params;
212   const { data, error } = await supabase
213     .from('tokens')
214     .select('*')
215     .eq('id', id);
216   if (error) throw error;
217   res.json(data);
218 });

219
220 // Rota para gerar um token de acesso
221 app.get('/tokens/:id', async (req, res) => {
222   const { id } = req.params;
223   const { data, error } = await supabase
224     .from('tokens')
225     .select('*')
226     .eq('id', id);
227   if (error) throw error;
228   res.json(data);
229 });

230
231 // Rota para gerar um token de acesso
232 app.get('/tokens/:id', async (req, res) => {
233   const { id } = req.params;
234   const { data, error } = await supabase
235     .from('tokens')
236     .select('*')
237     .eq('id', id);
238   if (error) throw error;
239   res.json(data);
240 });

241
242 // Rota para gerar um token de acesso
243 app.get('/tokens/:id', async (req, res) => {
244   const { id } = req.params;
245   const { data, error } = await supabase
246     .from('tokens')
247     .select('*')
248     .eq('id', id);
249   if (error) throw error;
250   res.json(data);
251 });

252
253 // Rota para gerar um token de acesso
254 app.get('/tokens/:id', async (req, res) => {
255   const { id } = req.params;
256   const { data, error } = await supabase
257     .from('tokens')
258     .select('*')
259     .eq('id', id);
260   if (error) throw error;
261   res.json(data);
262 });

263
264 // Rota para gerar um token de acesso
265 app.get('/tokens/:id', async (req, res) => {
266   const { id } = req.params;
267   const { data, error } = await supabase
268     .from('tokens')
269     .select('*')
270     .eq('id', id);
271   if (error) throw error;
272   res.json(data);
273 });

274
275 // Rota para gerar um token de acesso
276 app.get('/tokens/:id', async (req, res) => {
277   const { id } = req.params;
278   const { data, error } = await supabase
279     .from('tokens')
280     .select('*')
281     .eq('id', id);
282   if (error) throw error;
283   res.json(data);
284 });

285
286 // Rota para gerar um token de acesso
287 app.get('/tokens/:id', async (req, res) => {
288   const { id } = req.params;
289   const { data, error } = await supabase
290     .from('tokens')
291     .select('*')
292     .eq('id', id);
293   if (error) throw error;
294   res.json(data);
295 });

296
297 // Rota para gerar um token de acesso
298 app.get('/tokens/:id', async (req, res) => {
299   const { id } = req.params;
300   const { data, error } = await supabase
301     .from('tokens')
302     .select('*')
303     .eq('id', id);
304   if (error) throw error;
305   res.json(data);
306 });

307
308 // Rota para gerar um token de acesso
309 app.get('/tokens/:id', async (req, res) => {
310   const { id } = req.params;
311   const { data, error } = await supabase
312     .from('tokens')
313     .select('*')
314     .eq('id', id);
315   if (error) throw error;
316   res.json(data);
317 });

318
319 // Rota para gerar um token de acesso
320 app.get('/tokens/:id', async (req, res) => {
321   const { id } = req.params;
322   const { data, error } = await supabase
323     .from('tokens')
324     .select('*')
325     .eq('id', id);
326   if (error) throw error;
327   res.json(data);
328 });

329
330 // Rota para gerar um token de acesso
331 app.get('/tokens/:id', async (req, res) => {
332   const { id } = req.params;
333   const { data, error } = await supabase
334     .from('tokens')
335     .select('*')
336     .eq('id', id);
337   if (error) throw error;
338   res.json(data);
339 });

340
341 // Rota para gerar um token de acesso
342 app.get('/tokens/:id', async (req, res) => {
343   const { id } = req.params;
344   const { data, error } = await supabase
345     .from('tokens')
346     .select('*')
347     .eq('id', id);
348   if (error) throw error;
349   res.json(data);
350 });

351
352 // Rota para gerar um token de acesso
353 app.get('/tokens/:id', async (req, res) => {
354   const { id } = req.params;
355   const { data, error } = await supabase
356     .from('tokens')
357     .select('*')
358     .eq('id', id);
359   if (error) throw error;
360   res.json(data);
361 });

362
363 // Rota para gerar um token de acesso
364 app.get('/tokens/:id', async (req, res) => {
365   const { id } = req.params;
366   const { data, error } = await supabase
367     .from('tokens')
368     .select('*')
369     .eq('id', id);
370   if (error) throw error;
371   res.json(data);
372 });

373
374 // Rota para gerar um token de acesso
375 app.get('/tokens/:id', async (req, res) => {
376   const { id } = req.params;
377   const { data, error } = await supabase
378     .from('tokens')
379     .select('*')
380     .eq('id', id);
381   if (error) throw error;
382   res.json(data);
383 });

384
385 // Rota para gerar um token de acesso
386 app.get('/tokens/:id', async (req, res) => {
387   const { id } = req.params;
388   const { data, error } = await supabase
389     .from('tokens')
390     .select('*')
391     .eq('id', id);
392   if (error) throw error;
393   res.json(data);
394 });

395
396 // Rota para gerar um token de acesso
397 app.get('/tokens/:id', async (req, res) => {
398   const { id } = req.params;
399   const { data, error } = await supabase
400     .from('tokens')
401     .select('*')
402     .eq('id', id);
403   if (error) throw error;
404   res.json(data);
405 });

406
407 // Rota para gerar um token de acesso
408 app.get('/tokens/:id', async (req, res) => {
409   const { id } = req.params;
410   const { data, error } = await supabase
411     .from('tokens')
412     .select('*')
413     .eq('id', id);
414   if (error) throw error;
415   res.json(data);
416 });

417
418 // Rota para gerar um token de acesso
419 app.get('/tokens/:id', async (req, res) => {
420   const { id } = req.params;
421   const { data, error } = await supabase
422     .from('tokens')
423     .select('*')
424     .eq('id', id);
425   if (error) throw error;
426   res.json(data);
427 });

428
429 // Rota para gerar um token de acesso
430 app.get('/tokens/:id', async (req, res) => {
431   const { id } = req.params;
432   const { data, error } = await supabase
433     .from('tokens')
434     .select('*')
435     .eq('id', id);
436   if (error) throw error;
437   res.json(data);
438 });

439
440 // Rota para gerar um token de acesso
441 app.get('/tokens/:id', async (req, res) => {
442   const { id } = req.params;
443   const { data, error } = await supabase
444     .from('tokens')
445     .select('*')
446     .eq('id', id);
447   if (error) throw error;
448   res.json(data);
449 });

450
451 // Rota para gerar um token de acesso
452 app.get('/tokens/:id', async (req, res) => {
453   const { id } = req.params;
454   const { data, error } = await supabase
455     .from('tokens')
456     .select('*')
457     .eq('id', id);
458   if (error) throw error;
459   res.json(data);
460 });

461
462 // Rota para gerar um token de acesso
463 app.get('/tokens/:id', async (req, res) => {
464   const { id } = req.params;
465   const { data, error } = await supabase
466     .from('tokens')
467     .select('*')
468     .eq('id', id);
469   if (error) throw error;
470   res.json(data);
471 });

472
473 // Rota para gerar um token de acesso
474 app.get('/tokens/:id', async (req, res) => {
475   const { id } = req.params;
476   const { data, error } = await supabase
477     .from('tokens')
478     .select('*')
479     .eq('id', id);
480   if (error) throw error;
481   res.json(data);
482 });

483
484 // Rota para gerar um token de acesso
485 app.get('/tokens/:id', async (req, res) => {
486   const { id } = req.params;
487   const { data, error } = await supabase
488     .from('tokens')
489     .select('*')
490     .eq('id', id);
491   if (error) throw error;
492   res.json(data);
493 });

494
495 // Rota para gerar um token de acesso
496 app.get('/tokens/:id', async (req, res) => {
497   const { id } = req.params;
498   const { data, error } = await supabase
499     .from('tokens')
500     .select('*')
501     .eq('id', id);
502   if (error) throw error;
503   res.json(data);
504 });

505
506 // Rota para gerar um token de acesso
507 app.get('/tokens/:id', async (req, res) => {
508   const { id } = req.params;
509   const { data, error } = await supabase
510     .from('tokens')
511     .select('*')
512     .eq('id', id);
513   if (error) throw error;
514   res.json(data);
515 });

516
517 // Rota para gerar um token de acesso
518 app.get('/tokens/:id', async (req, res) => {
519   const { id } = req.params;
520   const { data, error } = await supabase
521     .from('tokens')
522     .select('*')
523     .eq('id', id);
524   if (error) throw error;
525   res.json(data);
526 });

527
528 // Rota para gerar um token de acesso
529 app.get('/tokens/:id', async (req, res) => {
530   const { id } = req.params;
531   const { data, error } = await supabase
532     .from('tokens')
533     .select('*')
534     .eq('id', id);
535   if (error) throw error;
536   res.json(data);
537 });

538
539 // Rota para gerar um token de acesso
540 app.get('/tokens/:id', async (req, res) => {
541   const { id } = req.params;
542   const { data, error } = await supabase
543     .from('tokens')
544     .select('*')
545     .eq('id', id);
546   if (error) throw error;
547   res.json(data);
548 });

549
550 // Rota para gerar um token de acesso
551 app.get('/tokens/:id', async (req, res) => {
552   const { id } = req.params;
553   const { data, error } = await supabase
554     .from('tokens')
555     .select('*')
556     .eq('id', id);
557   if (error) throw error;
558   res.json(data);
559 });

560
561 // Rota para gerar um token de acesso
562 app.get('/tokens/:id', async (req, res) => {
563   const { id } = req.params;
564   const { data, error } = await supabase
565     .from('tokens')
566     .select('*')
567     .eq('id', id);
568   if (error) throw error;
569   res.json(data);
570 });

571
572 // Rota para gerar um token de acesso
573 app.get('/tokens/:id', async (req, res) => {
574   const { id } = req.params;
575   const { data, error } = await supabase
576     .from('tokens')
577     .select('*')
578     .eq('id', id);
579   if (error) throw error;
580   res.json(data);
581 });

582
583 // Rota para gerar um token de acesso
584 app.get('/tokens/:id', async (req, res) => {
585   const { id } = req.params;
586   const { data, error } = await supabase
587     .from('tokens')
588     .select('*')
589     .eq('id', id);
590   if (error) throw error;
591   res.json(data);
592 });

593
594 // Rota para gerar um token de acesso
595 app.get('/tokens/:id', async (req, res) => {
596   const { id } = req.params;
597   const { data, error } = await supabase
598     .from('tokens')
599     .select('*')
600     .eq('id', id);
601   if (error) throw error;
602   res.json(data);
603 });

604
605 // Rota para gerar um token de acesso
606 app.get('/tokens/:id', async (req, res) => {
607   const { id } = req.params;
608   const { data, error } = await supabase
609     .from('tokens')
610     .select('*')
611     .eq('id', id);
612   if (error) throw error;
613   res.json(data);
614 });

615
616 // Rota para gerar um token de acesso
617 app.get('/tokens/:id', async (req, res) => {
618   const { id } = req.params;
619   const { data, error } = await supabase
620     .from('tokens')
621     .select('*')
622     .eq('id', id);
623   if (error) throw error;
624   res.json(data);
625 });

626
627 // Rota para gerar um token de acesso
628 app.get('/tokens/:id', async (req, res) => {
629   const { id } = req.params;
630   const { data, error } = await supabase
631     .from('tokens')
632     .select('*')
633     .eq('id', id);
634   if (error) throw error;
635   res.json(data);
636 });

637
638 // Rota para gerar um token de acesso
639 app.get('/tokens/:id', async (req, res) => {
640   const { id } = req.params;
641   const { data, error } = await supabase
642     .from('tokens')
643     .select('*')
644     .eq('id', id);
645   if (error) throw error;
646   res.json(data);
647 });

648
649 // Rota para gerar um token de acesso
650 app.get('/tokens/:id', async (req, res) => {
651   const { id } = req.params;
652   const { data, error } = await supabase
653     .from('tokens')
654     .select('*')
655     .eq('id', id);
656   if (error) throw error;
657   res.json(data);
658 });

659
660 // Rota para gerar um token de acesso
661 app.get('/tokens/:id', async (req, res) => {
662   const { id } = req.params;
663   const { data, error } = await supabase
664     .from('tokens')
665     .select('*')
666     .eq('id', id);
667   if (error) throw error;
668   res.json(data);
669 });

670
671 // Rota para gerar um token de acesso
672 app.get('/tokens/:id', async (req, res) => {
673   const { id } = req.params;
674   const { data, error } = await supabase
675     .from('tokens')
676     .select('*')
677     .eq('id', id);
678   if (error) throw error;
679   res.json(data);
680 });

681
682 // Rota para gerar um token de acesso
683 app.get('/tokens/:id', async (req, res) => {
684   const { id } = req.params;
685   const { data, error } = await supabase
686     .from('tokens')
687     .select('*')
688     .eq('id', id);
689   if (error) throw error;
690   res.json(data);
691 });

692
693 // Rota para gerar um token de acesso
694 app.get('/tokens/:id', async (req, res) => {
695   const { id } = req.params;
696   const { data, error } = await supabase
697     .from('tokens')
698     .select('*')
699     .eq('id', id);
700   if (error) throw error;
701   res.json(data);
702 });

703
704 // Rota para gerar um token de acesso
705 app.get('/tokens/:id', async (req, res) => {
706   const { id } = req.params;
707   const { data, error } = await supabase
708     .from('tokens')
709     .select('*')
710     .eq('id', id);
711   if (error) throw error;
712   res.json(data);
713 });

714
715 // Rota para gerar um token de acesso
716 app.get('/tokens/:id', async (req, res) => {
717   const { id } = req.params;
718   const { data, error } = await supabase
719     .from('tokens')
720     .select('*')
721     .eq('id', id);
722   if (error) throw error;
723   res.json(data);
724 });

725
726 // Rota para gerar um token de acesso
727 app.get('/tokens/:id', async (req, res) => {
728   const { id } = req.params;
729   const { data, error } = await sup
```

TERCEIRO PASSO

CRIE UMA PASTA BACKEND DENTRO DA PASTA JÁ CRIADA E CRIE UM ARQUIVO SERVER.JS E COLOQUEI ESSE CÓDIGO

```
35 });
36
37 // GET - Buscar produto por ID
38 app.get('/produtos/:id', async (req, res) => {
39   try {
40     const { data, error } = await supabase
41       .from('produtos')
42       .select('*')
43       .eq('id', req.params.id)
44       .single();
45     if (error) throw error;
46     res.json(data);
47   } catch (error) {
48     res.status(400).json({ erro: error.message });
49   }
50 });
51
52 // POST - Criar novo produto
53 app.post('/produtos', async (req, res) => {
54   const { nome, preco} = req.body;
55
56   if (!nome || !preco) {
57     return res.status(400).json({ erro: 'Nome e preço são obrigatórios.' });
58   }
59
60   try {
61     const { data, error } = await supabase
62       .from('produtos')
63       .insert([{ nome, preco }])
64       .select()
65       .single();
```

TERCEIRO PASSO

CRIE UMA PASTA BACKEND DENTRO DA PASTA JÁ CRIADA E CRIE UM ARQUIVO SERVER.JS E COLOQUEI ESSE CÓDIGO

```
66  if (error) throw error;
67  res.status(201).json(data);
68
69 } catch (error) {
70   res.status(400).json({ erro: error.message });
71 }
72 });
73 );
74
75
76 // PUT - Atualizar produto
77 app.put('/produtos/:id', async (req, res) => {
78   try {
79     const { nome, preco} = req.body;
80     const { data, error } = await supabase
81       .from('produtos')
82       .update({ nome, preco})
83       .eq('id', req.params.id)
84       .select();
85     if (error) throw error;
86     res.json(data);
87   } catch (error) {
88     res.status(400).json({ erro: error.message });
89   }
90 });
91
92 // DELETE - Deletar produto
93 app.delete('/produtos/:id', async (req, res) => {
94   try {
95     const { error } = await supabase
96       .from('produtos')
97       .delete()
```

TERCEIRO PASSO

CRIE UMA PASTA BACKEND DENTRO
DA PASTA JÁ CRIADA E CRIE UM
ARQUIVO SERVER.JS E COLOQUEI
ESSE CÓDIGO

```
    .eq('id', req.params.id);
    if (error) throw error;
    res.json({ mensagem: 'Produto deletado' });
} catch (error) {
    res.status(400).json({ erro: error.message });
}
});

app.listen(PORT, () => {
    console.log(`Servidor rodando em http://localhost:\${PORT}`);
});
```

TERCEIRO PASSO

CRIE UMA PASTA BACKEND DENTRO
DA PASTA JÁ CRIADA E CRIE UM
ARQUIVO SERVER.JS E COLOQUEI
ESSE CÓDIGO

```
98  |     .eq(`id`, req.params.id);
99  |     if (error) throw error;
100 |     res.json({ mensagem: 'Produto deletado' });
101 |   } catch (error) {
102 |     res.status(400).json({ erro: error.message });
103 |   }
104 | );
105 |
106 app.listen(PORT, () => {
107   console.log(`Servidor rodando em http://localhost:\${PORT}`);
108 });


```

QUARTO PASSO

CRIAR UM TABELA NO SUPABASE COM
ESSE CÓDIGO

```
1 create table produtos(  
2     id SERIAL PRIMARY KEY,  
3     nome VARCHAR(200),  
4     preco DECIMAL(10,2)  
5 )
```

QUINTO PASSO

CRIAR DENTRO DA PASTA
BACKEND UM ARQUIVO .ENV

DENTRO DO ARQUIVO .ENV
COLOQUE ESSE CÓDIGO

```
SUPABASE_URL=https://mlgbgczbzrtcdtzuhlwss.supabase.co  
SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Im1sZ2JnY3picnRjZHR6dWl  
PORT=3000
```

AGORA SUBSTITUA O SUBBASE_URL E
SUPABASE_KEY POR ESSES:

Project URL

<https://mlgbgczbzrtcdtzuhlwss.supabase.co>

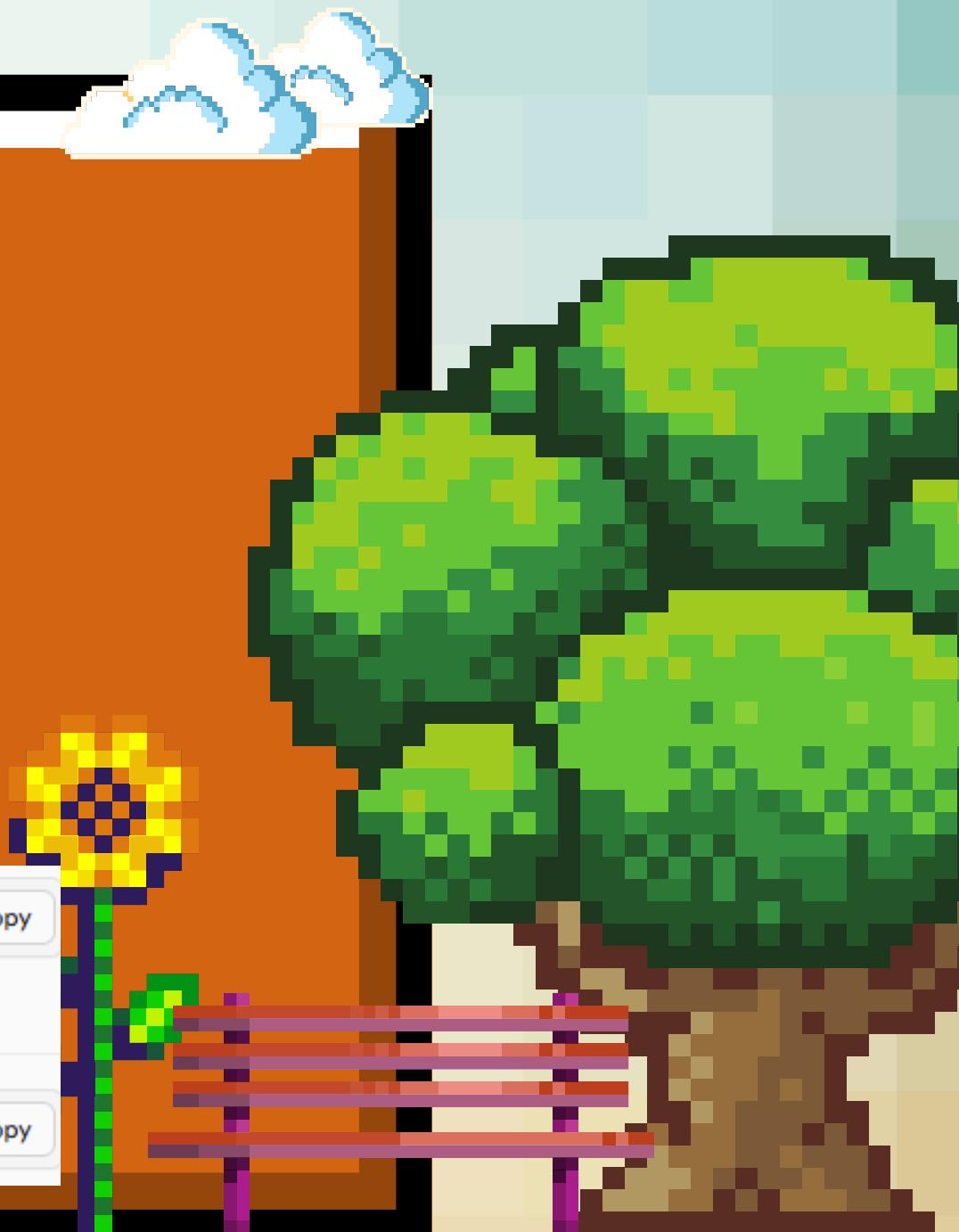
Copy

A RESTful endpoint for querying and managing your database.

API Key

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Im1sZ2JnY3picnRjZHR6dWl

Copy



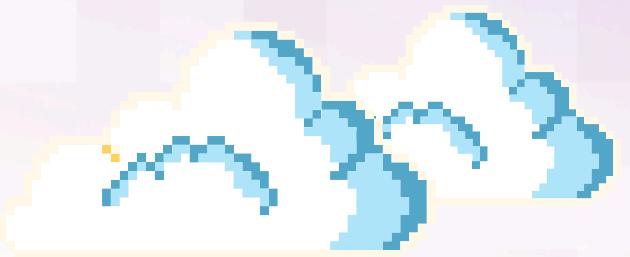
SEXTO PASSO

DENTRO DA PASTA BACKEND ATRAVÉS DO TERMINAL USE OS SEGUINTES COMANDOS PARA INSTALAR OS PACOTES:

- NPM INIT -Y;
- NPM INSTALL EXPRESS;
- NPM INSTALL CORS DOTENV @SUPABASE/SUPABASE-JS;
- NPM INSTALL NODEMON

SETIMO PASSO

AGORA NO TERMINAL USE
O SEGUINTE COMANDO
PARA INICIALIZAR O
SERVIDOR:
`-NODE SERVER.JS`



PASSO FINAL

AGORA ABRA O THUNDER CLIENTE PARA ADICIONAR DADOS A SUA TABELA

POST

Body

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```
1 {  
2   "id": 1,  
3   "nome": "God of War Ragnarok",  
4   "preco": 135  
5 }
```

YAY





THANKS!

END