



SUPLE-FIT

LOJA DE SUPLEMENTOS

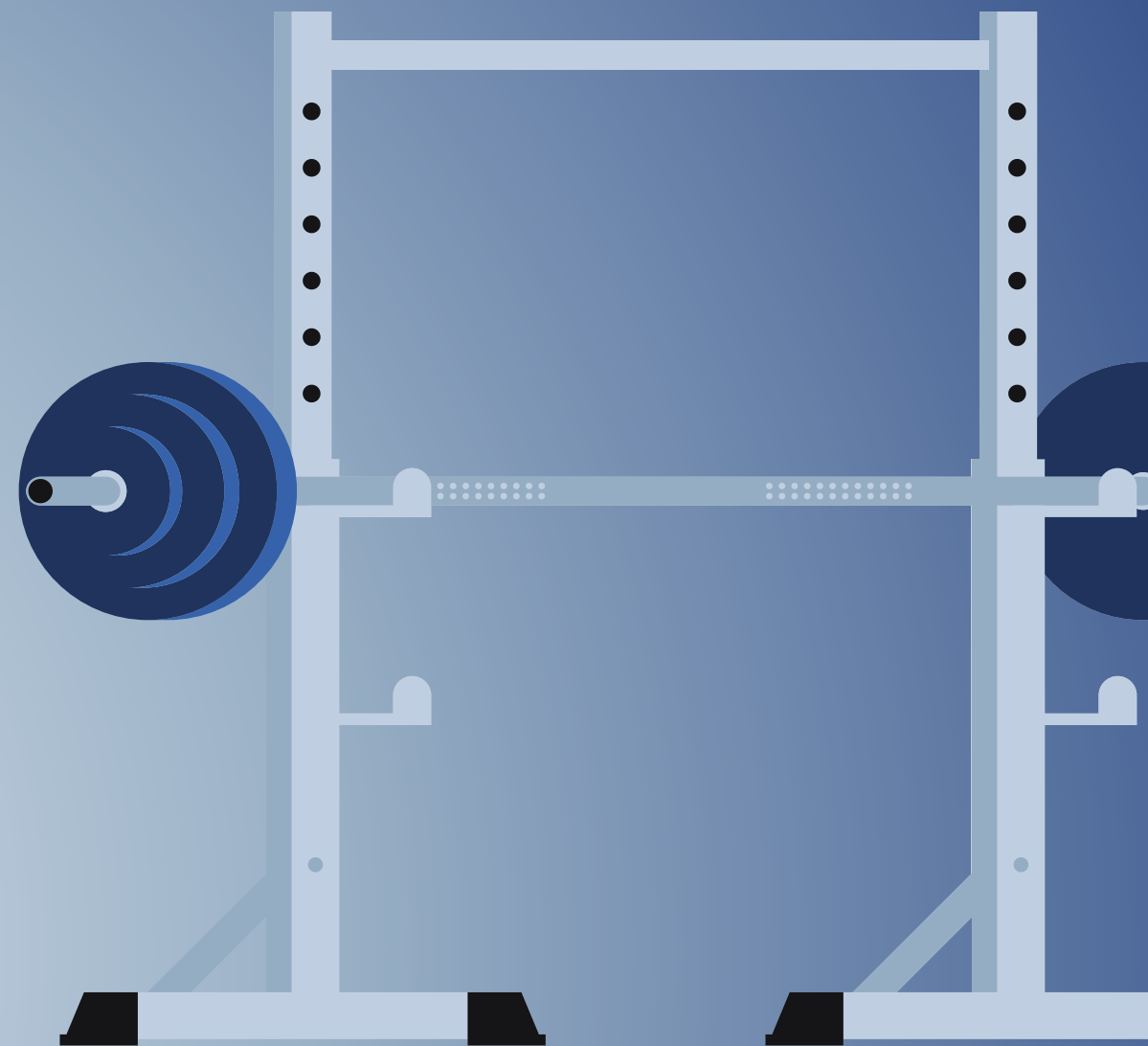
Suplementos alimentares para potencializar seus treinos e alcançar seus objetivos

1º PASSO



Criar a pasta
'LOJASUPLEMENTOS', jogar no VS
Code e em seguida criar as
seguintes pastas e os seguintes
arquivos.

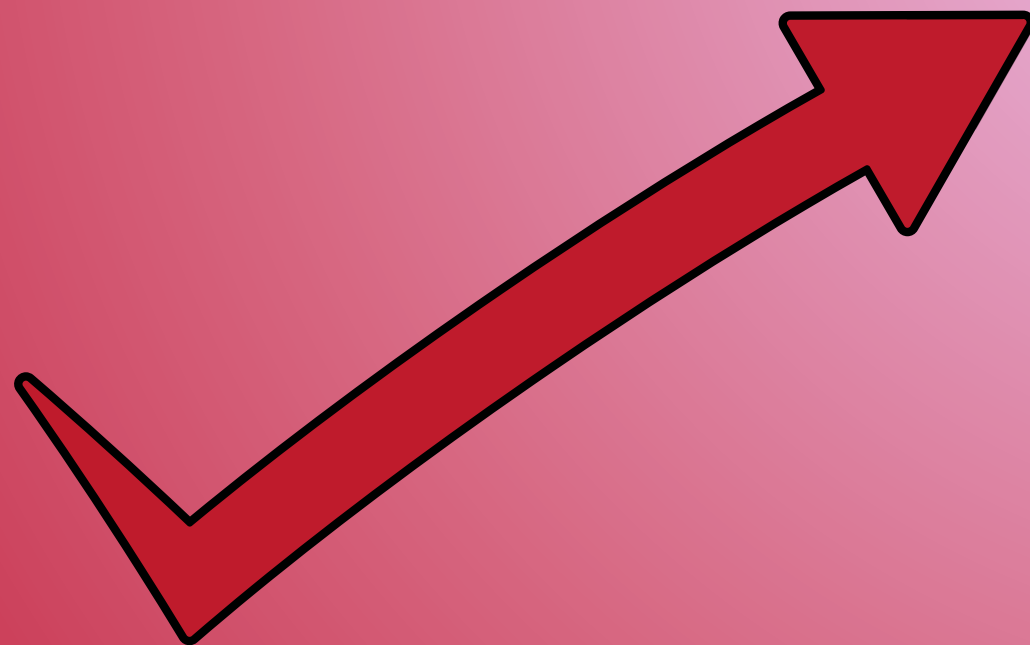
```
projeto-tutorial/  
├── /backend/  
│   ├── server.js  
│   ├── package.json  
│   └── /routes/  
│       └── api.js  
└── /frontend/  
    └── index.html
```



2º PASSO



Abrir o terminal e colocar os seguintes códigos: cd backend, npm init -y e npm install express.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\Aluno\Desktop\lojasuplementos> cd .\backend\
● PS C:\Users\Aluno\Desktop\lojasuplementos\backend> npm init -y
Wrote to C:\Users\Aluno\Desktop\lojasuplementos\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@supabase/supabase-js": "^2.84.0",
    "cors": "^2.8.5",
    "dotenv": "^17.2.3",
    "express": "^5.1.0"
  },
  "devDependencies": {},
  "description": ""
}

● PS C:\Users\Aluno\Desktop\lojasuplementos\backend> npm install express
```



3º PASSO



Colocar o código no arquivo 'server.js'

```
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 // GET - Buscar produto por ID
```

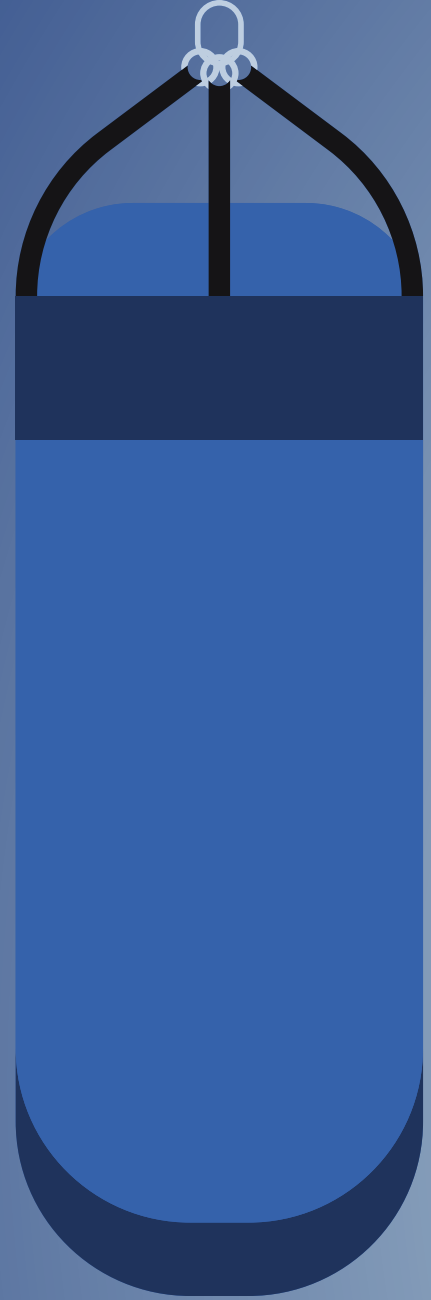
```
37 // GET - Buscar produto por ID
38 app.get('/produtos/:id', async (req, res) => {
39   try {
40     const { data, error } = await supabase
41       .from('produtos')
42       .select('*')
43       .eq('id', req.params.id)
44       .single();
45     if (error) throw error;
46     res.json(data);
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   try {
55     const { nome, categoria, preco, estoque, descricao } = req.body;
56     const { data, error } = await supabase
57       .from('produtos')
58       .insert([
59         { nome, categoria, preco, estoque, descricao }
60       ])
61       .select();
62     if (error) throw error;
63     res.json(data);
64   } catch (error) {
65     res.status(400).json({ erro: error.message });
66   }
67 });
68
69 // PUT - Atualizar produto
70 app.put('/produtos/:id', async (req, res) => {
71   try {
72     const { nome, preco } = req.body;
73     const { data, error } = await supabase
```

```
67 // PUT - Atualizar produto
68 app.put('/produtos/:id', async (req, res) => {
69   try {
70     const { nome, preco } = req.body;
71     const { data, error } = await supabase
72       .from('produtos')
73       .update({ nome, preco })
74       .eq('id', req.params.id)
75       .select();
76     if (error) throw error;
77     res.json(data);
78   } catch (error) {
79     res.status(400).json({ erro: error.message });
80   }
81 });
82
83 // DELETE - Deletar produto
84 app.delete('/produtos/:id', async (req, res) => {
85   try {
86     const { error } = await supabase
87       .from('produtos')
88       .delete()
89       .eq('id', req.params.id);
90     if (error) throw error;
91     res.json({ mensagem: 'Produto deletado' });
92   } catch (error) {
93     res.status(400).json({ erro: error.message });
94   }
95 });
96
97 app.listen(PORT, () => {
98   console.log(`Servidor rodando em http://localhost:${PORT}`);
99 });
```

4º PASSO

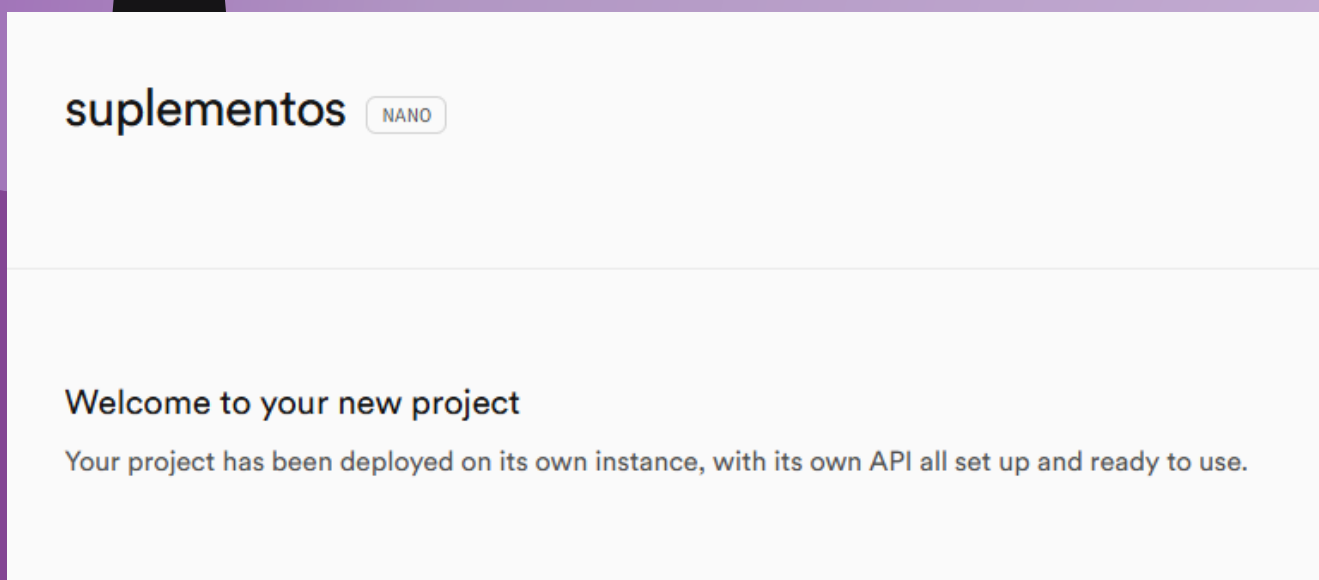
Executar o 'node server.js'

```
PS C:\Users\Aluno\Desktop\lojasuplementos> cd .\backend\  
PS C:\Users\Aluno\Desktop\lojasuplementos\backend> node server.js  
[dotenv@17.2.3] injecting env (3) from .env -- tip: ⚙ suppress all logs with { quiet: true }  
Servidor rodando em http://localhost:3000  
|
```



5º PASSO

Criar novo projeto no supabase, e criar a tabela 'produtos'.



The screenshot shows the Supabase SQL editor for the 'public.produtos' table. The table has four columns: 'id' (int4), 'nome' (text), 'categoria' (text), and an empty column. The table contains five rows of data.

	id int4	nome text	categoria text	
<input type="checkbox"/>	1	Whey Protein Concentrado 1kg	Proteína	
<input type="checkbox"/>	2	Creatina Monohidratada 300g	Creatina	
<input type="checkbox"/>	3	Pré-Treino Explosive 300g	Pré-Treino	
<input type="checkbox"/>	4	Multivitamínico Completo 60 cápsulas	Vitaminas	
<input type="checkbox"/>	5	BCAA 4:1:1 120 cápsulas	Aminoácidos	



6º PASSO

Abrir o arquivo ‘.env’ e
adicionar o URL e a KEY do
seu supabase

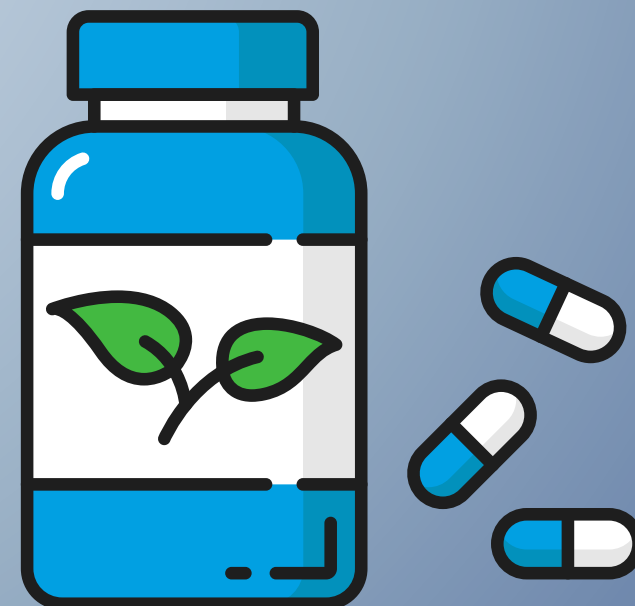
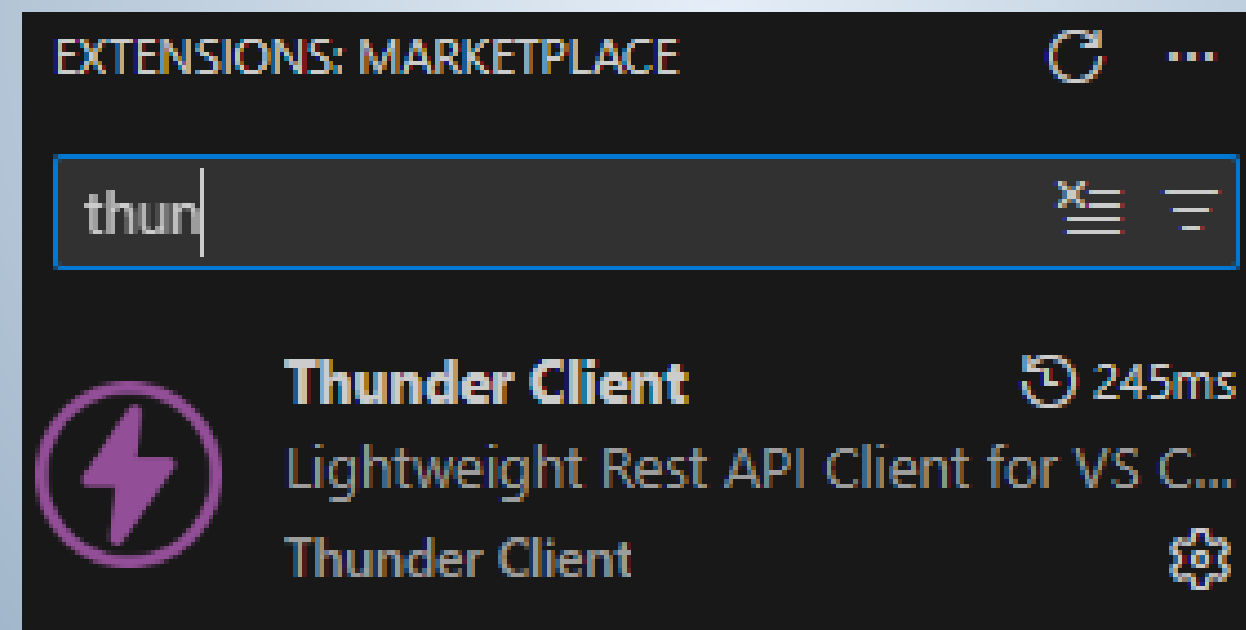
```
JS server.js  TC New Request  .env  X  JS api.js  <> index.js

backend > .env
1  SUPABASE_URL=https://rrgtxwofyasgezugqpx.supabase.co
2  SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYnEiLCJkIj...
3  PORT=3000
```



7º PASSO

Baixar o Thunder Client para poder testar os comandos.



8º PASSO

Testar os comandos:
GET, GET, POST, PUT e DELETE

