

# Consultas

Prof. Me. Ewerton José da Silva

# Login

```
134 },
135 async login(request, response) {
136   try {
137
138     const { usu_email, usu_senha } = request.body;
139
140     const sql = `SELECT usu_id, usu_nome, usu_tipo FROM usuarios
141       WHERE usu_email = ? AND usu_senha = ? AND usu_ativo = 1`;
142
143     const values = [usu_email, usu_senha];
144
145     const usuarios = await db.query(sql, values);
146     const nItens = usuarios[0].length;
147
148     if (nItens < 1) {
149       return response.status(403).json({
150         sucesso: false,
151         mensagem: 'Login e/ou senha inválido.',
152         dados: null,
153       });
154     }
155
156     return response.status(200).json({
157       sucesso: true,
158       mensagem: 'Login efetuado com sucesso',
159       dados: usuarios[0]
160     });
161   } catch (error) {
162     return response.status(500).json({
163       sucesso: false,
164       mensagem: 'Erro na requisição.',
165       dados: error.message
166     });
167   }
168 },
169 }
170 }
```

# Rota do Login

```
router.get('/usuarios', UsuariosController.listarUsuarios);  
router.post('/usuarios', UsuariosController.cadastrarUsuarios); //body  
router.patch('/usuarios/:usu_id', UsuariosController.editarUsuarios); // params (URL) e body  
router.delete('/usuarios/:usu_id', UsuariosController.apagarUsuarios); // params (URL)  
router.delete('/usuarios/del/:usu_id', UsuariosController.ocultarUsuario); // params (URL)  
router.post('/usuarios/login', UsuariosController.login); //body
```



# Teste login

POST ▼ http://localhost:3333/usuarios/login Send

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

```
1 {
2   "usu_email": "thomasfranciscocortereal@kaynak.com.br",
3   "usu_senha": "543@21"
4 }
```



Status: 200 OK Size: 133 Bytes Time: 8 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Login efetuado com sucesso",
4   "dados": [
5     {
6       "usu_id": 1,
7       "usu_nome": "Thomas Francisco Corte Real",
8       "usu_tipo": 0
9     }
10  ]
11 }
```



POST ▼ http://localhost:3333/usuarios/login Send

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

```
1 {
2   "usu_email": "thomasfranciscocortereal@kaynak.com.br",
3   "usu_senha": "543@2"
4 }
```




Status: 403 Forbidden Size: 71 Bytes Time: 4 ms

Response Headers <sup>7</sup> Cookies Results Docs


```
1 {
2   "sucesso": false,
3   "mensagem": "Login e/ou senha inválido.",
4   "dados": null
5 }
```



# Pesquisa com parâmetros

```
async listarIngredientes(request, response) {  
  try {  
    const { ing_nome } = request.body;  
     const ingPesq = ing_nome ? `%${ing_nome}%` : `%%`;   
    const sql = `SELECT  
      ing_id, ing_nome, ing_img, ing_custo_adicional  
    FROM ingredientes  
    WHERE ing_nome like ?`;   
  
    const values = [ingPesq];  
    const ingredientes = await db.query(sql, values);  
    const nItens = ingredientes[0].length;  
  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de ingredientes.',  
      dados: ingredientes[0],  
      nItens  
    });  
  } catch (error) {  
    return response.status(500).json({  
      sucesso: false,  
      mensagem: 'Erro na requisição.',  
      dados: error.message  
    });  
  }  
},
```

# Teste pesquisa com parâmetros

 Ingredientes

**GET** listar  
just now

GET ⌵ http://localhost:3333/ingredientes Send

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "ing_nome": "al"
3 }
```

Status: 200 OK Size: 245 Bytes Time: 4 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de ingredientes.",
4   "dados": [
5     {
6       "ing_id": 3,
7       "ing_nome": "Salmão",
8       "ing_img": "salmao.png",
9       "ing_custo_adicional": "10.00"
10    },
11    {
12      "ing_id": 4,
13      "ing_nome": "Alface",
14      "ing_img": "alface.png",
15      "ing_custo_adicional": "4.50"
16    }
17  ],
18  "nItens": 2
19 }
```

# Pesquisa com múltiplos parâmetros

```
async listarCidades(request, response) {  
  try {  
    const { cid_uf, cid_nome } = request.body;  
    const cidPesq = cid_nome ? `${cid_nome}%` : `%%`;   
    const sql = `SELECT  
      cid_id, cid_nome, cid_uf  
    FROM cidades  
    WHERE cid_uf = ? AND cid_nome like ?`;   
  
    const values = [cid_uf, cidPesq];  
    const cidades = await db.query(sql, values);  
    const nItens = cidades[0].length;  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de cidades.',  
      dados: cidades[0],  
      nItens  
    });  
  } catch (error) {  
    return response.status(500).json({  
      sucesso: false,  
      mensagem: 'Erro na requisição.',  
      dados: error.message  
    });  
  }  
},
```

# Test pesquisa com múltiplos parâmetros

GET ⌵ http://localhost:3333/cidades Send

Query

Headers <sup>2</sup>

Auth

**Body <sup>1</sup>**

Tests

Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "cid_uf" : "SP",
3   "cid_nome": ""
4 }
```

Status: 200 OK Size: 34.75 KB Time: 8 ms

Response

Headers <sup>7</sup>

Cookies

Results

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de cidades.",
4   > "dados": [ ... ],
3231   "nItens": 645
3232 }
```





# Test pesquisa com múltiplos parâmetros

GET ⌵ http://localhost:3333/cidades Send

Query

Headers <sup>2</sup>

Auth

Body <sup>1</sup>

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content Format

```
1 {
2   "cid_uf" : "SP",
3   "cid_nome": "Tup"
4 }
```

Status: 200 OK   Size: 280 Bytes   Time: 6 ms

Response

Headers <sup>7</sup>

Cookies

Results

Docs

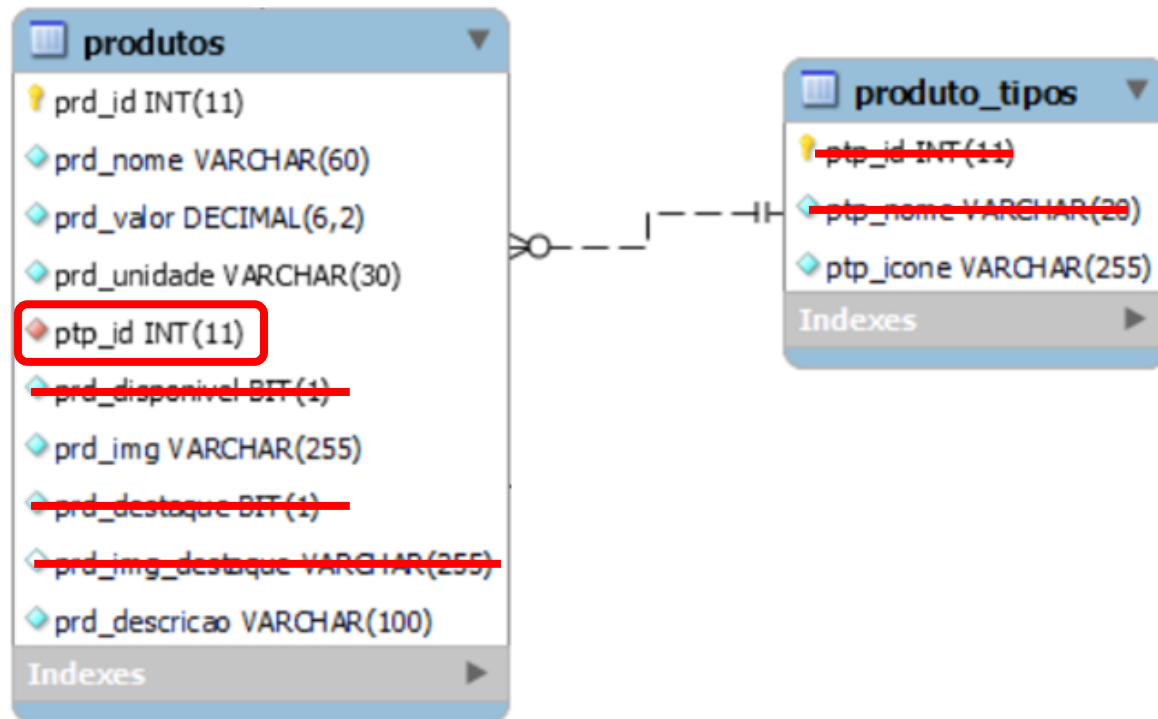
```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de cidades.",
4   "dados": [
5     {
6       "cid_id": 3541,
7       "cid_nome": "Itupeva",
8       "cid_uf": "SP"
9     },
10    {
11      "cid_id": 3884,
12      "cid_nome": "Tupã",
13      "cid_uf": "SP"
14    },
15    {
16      "cid_id": 3885,
17      "cid_nome": "Tupi Paulista",
18      "cid_uf": "SP"
19    },
20    {
21      "cid_id": 3909,
22      "cid_nome": "Votuporanga",
23      "cid_uf": "SP"
24    }
25  ],
26  "nItens": 4
```

# Instrução SQL de retorno simples

```
29     },
30     async listarUfs(request, response) {
31         try {
32             const sql = `SELECT DISTINCT cid_uf FROM cidades ORDER BY cid_uf ASC`;
33             const estados = await db.query(sql);
34
35             return response.status(200).json({
36                 sucesso: true,
37                 mensagem: 'Lista de estados.',
38                 dados: estados[0]
39             });
40         } catch (error) {
41             return response.status(500).json({
42                 sucesso: false,
43                 mensagem: 'Erro na requisição.',
44                 dados: error.message
45             });
46         }
47     },
48 }
49
```

```
router.get('/cidades', CidadesController.listarCidades);
router.get('/ufs', CidadesController.listarUfs);
```

# Uso de Inner Join para chave estrangeira



```
async listarProdutos(request, response) {  
  try {  
    const sql = `SELECT  
      prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade, pdt.ptpicone,  
      prd.prd_img, prd.prd_descricao  
    FROM produtos prd  
    INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id`;   
  
    const produtos = await db.query(sql);  
  
    const nItens = produtos[0].length;  
  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de produtos.',  
      dados: produtos[0],  
      nItens  
    });  
  } catch (error) {  
    // tratamento de erro  
  }  
}
```

# Uso de Inner Join para chave estrangeira

```
1  {
2    "sucesso": true,
3    "mensagem": "Lista de produtos.",
4    "dados": [
5      {
6        "prd_id": 1,
7        "prd_nome": "Lanche de Frango",
8        "prd_valor": "15.00",
9        "prd_unidade": "un.",
10       "ptp_icone": "lanche.svg",
11       "prd_img": "p1.png",
12       "prd_descricao": "Pão, frango desfiado e temperado"
13     },
14     {
15       "prd_id": 2,
16       "prd_nome": "Lanche de Salmão"
```

```
1  {
2    "sucesso": true,
3    "mensagem": "Lista de produtos.",
4  >  "dados": [...],
78   "nItens": 8
79 }
```

# Ajustes para filtro de produtos

```
4      async listarProdutos(request, response) {
5
6          const { prd_nome, ptp_id, prd_valor } = request.body;
7          const prd_disponivel = 1
8          const prdPesqNm = prd_nome ? `%${prd_nome}%` : `%%`;
9          const prdPesqTp = ptp_id ? `%${ptp_id}%` : `%%`;
10
11          try {
12              const sqlMaxVlr = 'SELECT MAX(prd_valor) vlr_max FROM produtos;';
13              const vlrMax = await db.query(sqlMaxVlr);
14              var prdPesqVlr = prd_valor ? prd_valor : parseFloat(vlrMax[0][0].vlr_max) + 1;
15
16          } catch (error) {
17              return response.status(500).json({
18                  sucesso: false,
19                  mensagem: 'Erro na requisição.',
20                  dados: error.message
21              });
22          }
23
24          + new f
```

```

23
24     try {
25         const sql = `SELECT
26             prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade, pdt.ptp_icone,
27             prd.prd_img, prd.prd_descricao
28         FROM produtos prd
29         INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id
30         WHERE prd.prd_disponivel = ? AND prd.prd_nome LIKE ? AND prd.ptp_id LIKE ?
31         AND prd.prd_valor < ?`;
32
33         const values = [prd_disponivel, prdPesqNm, prdPesqTp, prdPesqVlr];
34
35         const produtos = await db.query(sql, values);
36
37         const nItens = produtos[0].length;
38
39         return response.status(200).json({
40             sucesso: true,
41             mensagem: 'Lista de produtos.',
42             dados: produtos[0],
43             nItens
44         });
45     } catch (error) {
46         return response.status(500).json({
47             sucesso: false,
48             mensagem: 'Erro na requisição.',
49             dados: error.message
50         });
51     }
52 },
53 async cadastrarProdutos(request, response) {

```

GET  Send

Query Headers <sup>2</sup> Auth **Body** Tests Pre Run

**JSON** XML Text Form Form-encode GraphQL Binary

JSON Content

Format

1

Status: **200 OK** Size: **1.23 KB** Time: **36 ms**

**Response** Headers <sup>7</sup> Cookies Results Docs {} ≡

```
1  {
2    "sucesso": true,
3    "mensagem": "Lista de produtos.",
4    "dados": [
5      {
6        "prd_id": 1,
7        "prd_nome": "Lanche de Frango",
8        "prd_valor": "15.00",
9        "prd_unidade": "un.",
10       "ptp_icone": "lanche.svg",
11       "prd_img": "p1.png",
12       "prd_descricao": "Pão, frango desfiado e temperado"
13     },
14     {
15       "prd_id": 3,
16       "prd_nome": "Lanche de Salada",
17       "prd_valor": "18.00",
18       "prd_unidade": "un.",
19       "ptp_icone": "lanche.svg",
20       "prd_img": "p3.img",
21       "prd_descricao": "Pão, alface, tomate, rúcula, milho,
                           pepino e aspargo"
```

GET

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "prd_nome" : null,
3   "ptp_id" : 3,
4   "prd_valor" : null
5 }
```

Status: 200 OK Size: 699 Bytes Time: 7 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos.",
4 >  "dados": [...],
42   "nItens": 4
43 }
```

GET

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "prd_nome" : null,
3   "ptp_id" : null,
4   "prd_valor" : 16
5 }
```

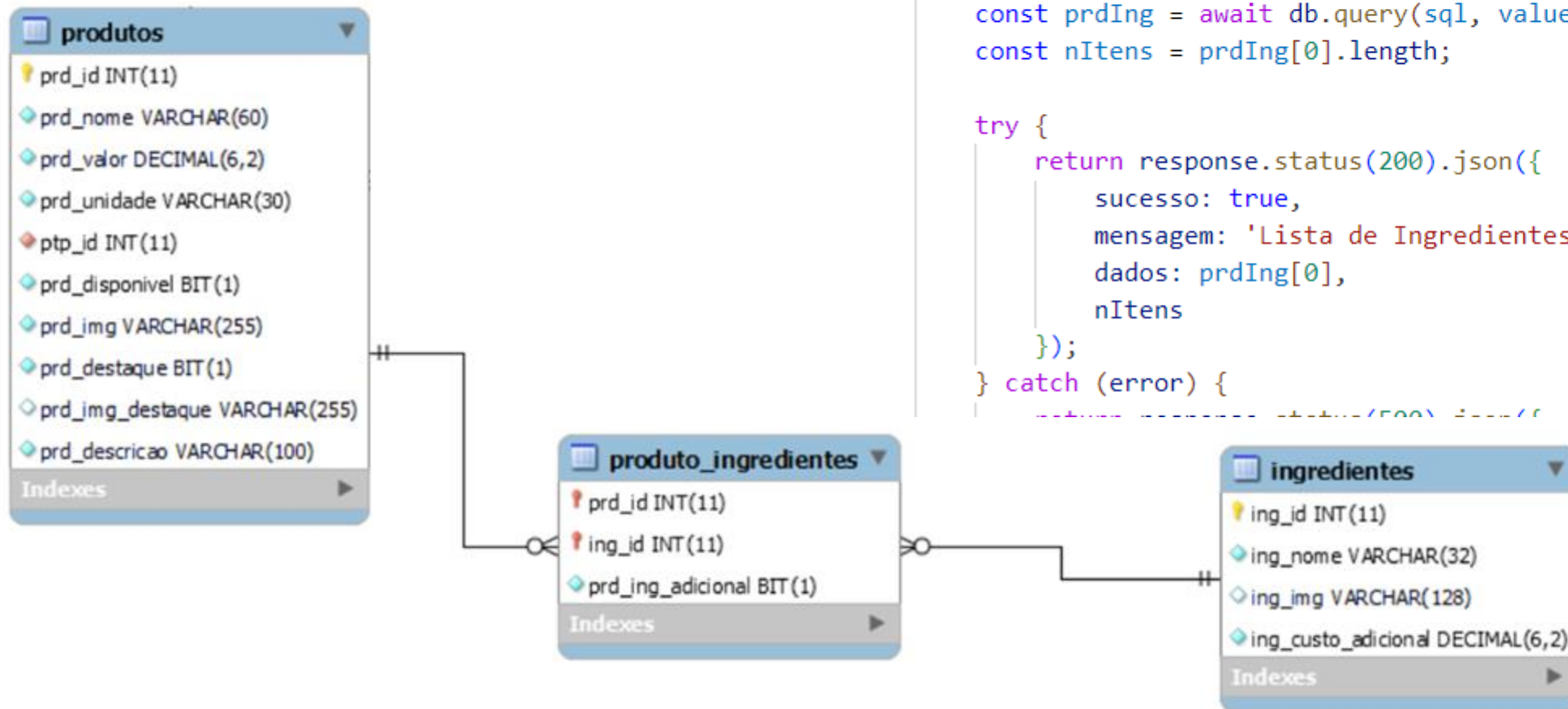
Status: 200 OK Size: 878 Bytes Time: 5 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos.",
4 >  "dados": [...],
51   "nItens": 5
52 }
```



# Uso de Inner Join para relacionamento n:n



```
async listarProdutoIngredientes(request, response) {  
  
  const { prd_id, prd_ing_adicional } = request.body;  
  
  const sql = `SELECT ing.ing_nome, ing.ing_img, ing.ing_custo_adicional  
FROM produto_ingredientes pi  
INNER JOIN ingredientes ing ON ing.ing_id = pi.ing_id  
WHERE pi.prd_id = ? AND pi.prd_ing_adicional = ?`;   
  
  const values = [prd_id, prd_ing_adicional];  
  
  const prdIng = await db.query(sql, values);  
  const nItens = prdIng[0].length;  
  
  try {  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de Ingredientes do produto.',  
      dados: prdIng[0],  
      nItens  
    });  
  } catch (error) {  
    return response.status(500).json({  
      sucesso: false,  
      mensagem: 'Erro ao listar ingredientes do produto.',  
      dados: null,  
      nItens: 0  
    });  
  }  
}
```

GET  Send

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

```
1 {
2   "prd_id": 1,
3   "prd_ing_adicional": 0
4 }
```

Status: 200 OK Size: 227 Bytes Time: 38 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de Ingredientes do produto.",
4   "dados": [
5     {
6       "ing_nome": "Pão",
7       "ing_img": "pao.png",
8       "ing_custo_adicional": "0.00"
9     },
10    {
11      "ing_nome": "Frango",
12      "ing_img": "frango.png",
13      "ing_custo_adicional": "7.00"
14    }
15  ],
16  "nItens": 2
17 }
```

# Uso de Inner Join para relacionamentos em mais de uma tabela

```
async listarClientes(request, response) {
  try {
    const { usu_nome, cli_cel } = request.body;

    const pesqNome = usu_nome ? `%${usu_nome}%` : `%%`;
    const usu_ativo = 1;
    const end_principal = 1;
    const campo = cli_cel ? 'cl.cli_cel =' : 'us.usu_nome LIKE';
    const campoPesq = cli_cel ? cli_cel : pesqNome;

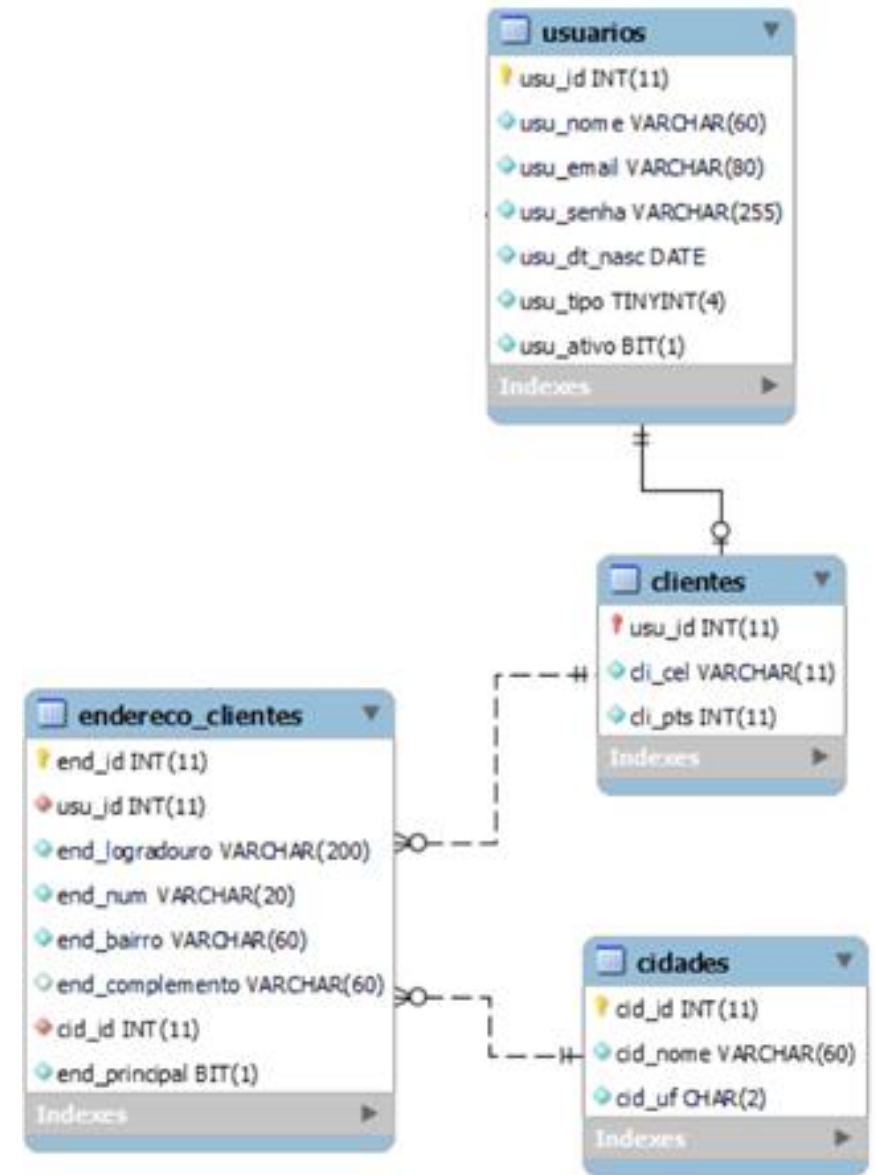
    const sql = `SELECT us.usu_nome, us.usu_dt_nasc, cl.cli_cel, cl.cli_pts, cid.cid_nome
      FROM clientes cl
      INNER JOIN usuarios us ON us.usu_id = cl.usu_id
      INNER JOIN endereco_clientes edcl ON edcl.usu_id = cl.usu_id
      INNER JOIN cidades cid ON cid.cid_id = edcl.cid_id
      WHERE us.usu_ativo = ? AND edcl.end_principal = ? AND ${campo} ?`;

    const values = [usu_ativo, end_principal, campoPesq];

    const clientes = await db.query(sql, values);

    const nItens = clientes[0].length;

    return response.status(200).json({
      sucesso: true,
      mensagem: 'Lista de Clientes.',
      dados: clientes[0],
      nItens
    });
  } catch (error) {
```



GET

http://localhost:3333/clientes

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

1 {

2   "usu\_nome" : null,

3   "cli\_cel" : null

4 }

Format

Status: 200 OK   Size: 533 Bytes   Time: 39 ms

Response

Headers 7

Cookies

Results

Docs

1 {

2   "sucesso": true,

3   "mensagem": "Lista de Clientes.",

4 > "dados": [ ... ],

30   "nItens": 4

31 }

51 •   SELECT \* FROM clientes;

Result Grid

Filter Rows:

	usu_id	cli_cel	cli_pts
▶	4	14911112222	0
	5	14922334444	50
	6	14911113111	0
	7	14911113333	0
	41	11988885678	0
	42	18912345678	0
*	NULL	NULL	NULL

55       WHERE edc.end\_principal = 1 AND us.usu\_ativo = 1;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	end_id	usu_id	end_logradouro	end_num	end_bairro	end_complemento	cid_id	end_principal	end_excluido	usu_id	cli_cel
▶	1	4	Rua dos Salgueiros	645	Mangabeira	Fundos	3884	1	0	4	14911112222
	3	6	Rua Mundico Thomas	39	Treze de Setembro	NULL	3672	1	0	6	14911113111
	6	7	Avenida Brasil	645	Centro	NULL	3884	1	0	7	14911113333
	8	42	Rua Argentina	17	Vila Portuguesa	NULL	3884	1	0	42	18912345678

Só lista clientes com endereços cadastrados

Se pesquisar um usuário que não tem endereço, o mesmo não será apresentado, devido ao uso do campo que apresenta o nome da cidade do cliente na listagem dos dados.

GET ⌵ http://localhost:3333/clientes Send

Query

Headers <sup>2</sup>

Auth

Body <sup>1</sup>

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content Format

1

{

2

"usu\_nome" : null,

3

"cli\_cel" : "11988885678"

4

}

Status: 200 OK    Size: 70 Bytes    Time: 5 ms

Response

Headers <sup>7</sup>

Cookies

Results

Docs

1

{

2

"sucesso": true,

3

"mensagem": "Lista de Clientes.",

4

"dados": [],

5

"nItens": 0

6

}

Se não for necessário apresentar o nome da cidade do cliente na pesquisa é possível listar clientes que não tenham endereço.

Caso o cliente tenha o endereço a pesquisa funciona normalmente. Então recomenda-se obrigar o usuário a cadastrar um endereço para evitar problemas, ou devemos remover o campo de cidade da pesquisa.

GET ⌵ http://localhost:3333/clientes Send

Query

Headers <sup>2</sup>

Auth

Body <sup>1</sup>

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content Format

```
1 {
2   "usu_nome" : null,
3   "cli_cel" : "18912345678"
4 }
```

Status: 200 OK Size: 195 Bytes Time: 7 ms

Response

Headers <sup>7</sup>

Cookies

Results

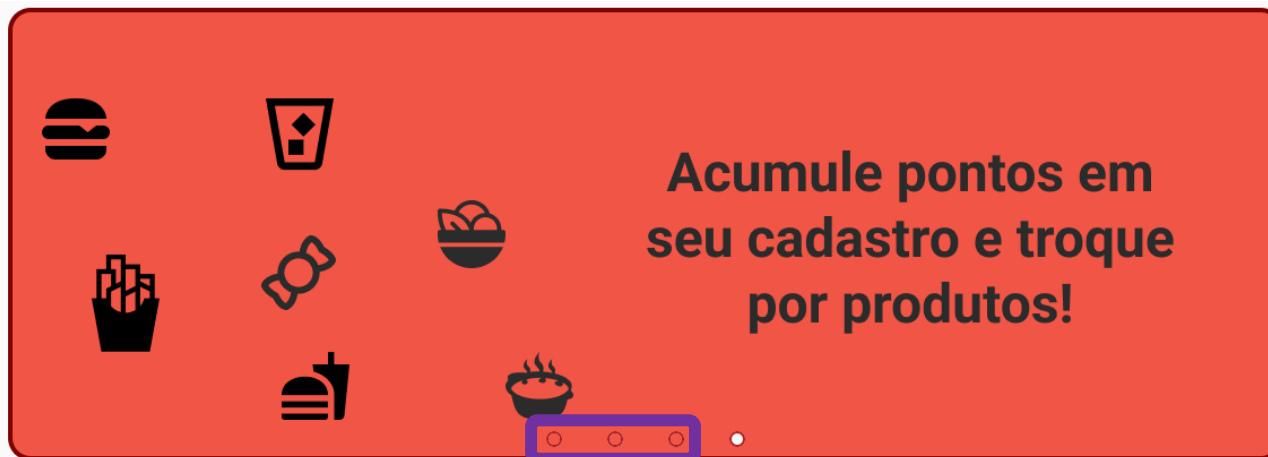
Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de Clientes.",
4   "dados": [
5     {
6       "usu_nome": "Lionel Ronaldo",
7       "usu_dt_nasc": "1988-10-10T03:00:00.000Z",
8       "cli_cel": "18912345678",
9       "cli_pts": 0,
10      "cid_nome": "Tupã"
11    }
12  ],
13  "nItens": 1
14 }
```

# Exemplo random

Existem quatro produtos na promoção!

prd_id	prd_nome	prd_valor	prd_unidade	ptp_id	prd_disponivel	prd_img	prd_destaque	prd_img_destaque	prd_descricao
1	Lanche de Frango	15.00	un.	1	1	p1.png	0	NULL	Pão, frango desfiado e temperado
2	Lanche de Salmão	28.00	un.	1	0	p2.png	1	salmaopromo.png	Pão, filé de salmão temperado com ervas finas
3	Lanche de Salada	18.00	un.	1	1	p3.png	0	NULL	Pão, alface, tomate, rúcula, milho, pepino e asp...
4	Batata frita	17.20	un.	2	1	sem.png	1	batataPromo.png	Batata de qualidade internacional.
5	Suco de Abacaxi	12.00	copo	3	1	sem.png	1	sucoAbacaxiPromo.png	Abacaxi, açúcar e gelo
6	Suco de Uva	15.00	copo	3	1	sem.png	1	sucoUvaPromo.png	Uva, açúcar e gelo
7	Suco de Laranja	12.00	copo	3	1	sem.png	0	NULL	Laranja, açúcar e gelo
8	Suco de Limão	12.00	copo	3	1	sem.png	0	NULL	Limão, açúcar e gelo



É esperado que três produtos sejam apresentados no slider da home do site/app

# Adicionar um método no controle de produtos, com o objeto de listar apenas 3 itens promocionais

JS produtos.js X

controllers > JS produtos.js > ...

```
97     },
98     async listarPromocoes(request, response) {
99         try {
100
101             const sql = `SELECT prd_img_destaque FROM produtos
102                WHERE prd_destaque = 1
103                ORDER BY RAND()
104                LIMIT 3;`;
105
106             const promo = await db.query(sql);
107             const nItens = promo[0].length;
108
109             return response.status(200).json({
110                 sucesso: true,
111                 mensagem: 'Itens na promoção.',
112                 dados: promo[0],
113                 nItens
114             });
115         } catch (error) {
116             return response.status(500).json({
117                 sucesso: false,
118                 mensagem: 'Erro na requisição.',
119                 dados: error.message
120             });
121         }
122     },
123 }
124
```

Depois adicionar uma rota para acessar os produtos da promoção

```
router.get('/produtos', ProdutosController.listarProdutos);
router.get('/produtos/promocoes', ProdutosController.listarPromocoes);
router.post('/produtos', ProdutosController.cadastrarProdutos);
router.patch('/produtos', ProdutosController.editarProdutos);
router.delete('/produtos', ProdutosController.apagarProdutos);
```



# A cada requisição a rota de promoções teremos um resultado diferente.

GET

http://localhost:3333/produtos/promocoes

Send

Status: 200 OK Size: 194 Bytes Time: 4 ms

Response	Headers 7	Cookies	Results	Docs
1 {				
2 "sucesso": true,				
3 "mensagem": "Itens na promoção.",				
4 "dados": [				
5 {				
6   "prd_img_destaque": "sucoAbacaxiPromo.png"				
7 },				
8 {				
9   "prd_img_destaque": "salmaopromo.png"				
10 },				
11 {				
12   "prd_img_destaque": "sucoUvaPromo.png"				
13 }				
14 ],				
15 "nItens": 3				
16 }				

Status: 200 OK Size: 189 Bytes Time: 3 ms

Response	Headers 7	Cookies	Results	Docs
1 {				
2 "sucesso": true,				
3 "mensagem": "Itens na promoção.",				
4 "dados": [				
5 {				
6   "prd_img_destaque": "sucoUvaPromo.png"				
7 },				
8 {				
9   "prd_img_destaque": "batataPromo.png"				
10 },				
11 {				
12   "prd_img_destaque": "salmaopromo.png"				
13 }				
14 ],				
15 "nItens": 3				
16 }				

Status: 200 OK Size: 194 Bytes Time: 4 ms

Response	Headers 7	Cookies	Results	Docs
1 {				
2 "sucesso": true,				
3 "mensagem": "Itens na promoção.",				
4 "dados": [				
5 {				
6   "prd_img_destaque": "sucoUvaPromo.png"				
7 },				
8 {				
9   "prd_img_destaque": "batataPromo.png"				
10 },				
11 {				
12   "prd_img_destaque": "sucoAbacaxiPromo.png"				
13 }				
14 ],				
15 "nItens": 3				
16 }				

# Paginação

controllers > JS produtos.js > ...

```
1  const db = require('../database/connection');
2
3  module.exports = {
4    async listarProdutos(request, response) {
5      {
6        const { page = 1, limit = 5 } = request.query;
7        const inicio = (page - 1) * limit;
8      }
9      const { prd_nome, ptp_id, prd_valor } = request.body;
10     const prd_disponivel = 1;
```

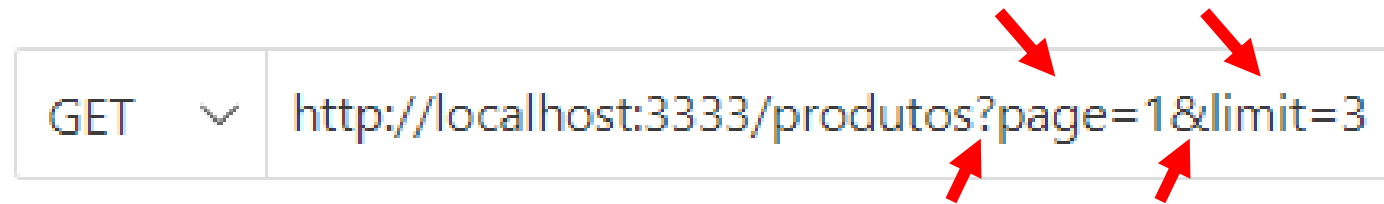
# Alterar a instrução SQL e passagem dos parâmetros

```
const sql = `SELECT
prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade, pdt.ptp_icone,
prd.prd_img, prd.prd_descricao
FROM produtos prd
INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id
WHERE prd.prd_disponivel = ? AND prd.prd_nome LIKE ? AND prd.ptp_id LIKE ?
AND prd.prd_valor < ? ←
LIMIT ?, ?;`;
```

```
const values = [prd_disponivel, prdPesqNm, prdPesqTp, prdPesqVlr, parseInt(inicio), parseInt(limit)];
```



# Inserir paginação na rota da requisição



GET  Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

```
1 {
2   "prd_nome" : null,
3   "ptp_id" : null,
4   "prd_valor" : null
5 }
```

Status: 200 OK Size: 643 Bytes Time: 5 ms

Response Headers 7 Cookies Results Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos.",
4   "dados": [
5     {
6       "prd_id": 1,
7       "prd_nome": "Lanche de Frango",
8       "prd_valor": "15.00",
9       "prd_unidade": "un.",
10      "ptp_icone": "lanche.svg",
11      "prd_img": "p1.png",
12      "prd_descricao": "Pão, frango desfiado e temperado"
13    },
14    {
15      "prd_id": 2,
16      "prd_nome": "Lanche de Salmão",
17      "prd_valor": "28.00",
18      "prd_unidade": "un.",
19      "ptp_icone": "lanche.svg",
20      "prd_img": "p2.png",
21      "prd_descricao": "Pão, filé de salmão temperado com ervas finas"
22    },
23    {
24      "prd_id": 3,
25      "prd_nome": "Lanche de Salada",
26      "prd_valor": "18.00",
27      "prd_unidade": "un.",
28      "ptp_icone": "lanche.svg",
29      "prd_img": "p3.png",
30      "prd_descricao": "Pão, alface, tomate, rúcula, milho, pepino e aspargo"
31    }
32  ],
33   "nItens": 3
34 }
```

GET  Send

Query Headers<sup>2</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

```
1 {
2   "prd_nome" : null,
3   "ptp_id" : null,
4   "prd_valor" : null
5 }
```

Status: 200 OK Size: 583 Bytes Time: 5 ms

**Response** Headers<sup>7</sup> Cookies Results Docs {}

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos.",
4   "dados": [
5     {
6       "prd_id": 4,
7       "prd_nome": "Batata frita",
8       "prd_valor": "17.20",
9       "prd_unidade": "un.",
10      "ptp_icone": "porcao.svg",
11      "prd_img": "batata.png",
12      "prd_descricao": "Batata de qualidade internacional."
13    },
14    {
15      "prd_id": 5,
16      "prd_nome": "Suco de Abacaxi",
17      "prd_valor": "12.00",
18      "prd_unidade": "copo",
19      "ptp_icone": "suco.svg",
20      "prd_img": "sabacaxi.png",
21      "prd_descricao": "Abacaxi, açúcar e gelo"
22    },
23    {
24      "prd_id": 6,
25      "prd_nome": "Suco de Uva",
26      "prd_valor": "15.00",
27      "prd_unidade": "copo",
28      "ptp_icone": "suco.svg",
29      "prd_img": "suva.png",
30      "prd_descricao": "Uva, açúcar e gelo"
31    }
32  ],
33   "nItens": 3
34 }
```