


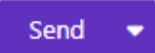
# Consultas

Prof. Me. Ewerton José da Silva



# Instrução SQL de retorno simples

```
async listarUfs(request, response) {  
  try {  
    const sql = `  
      SELECT DISTINCT  
        cid_uf  
      FROM  
        cidades  
      ORDER BY  
        cid_uf ASC;  
    `;  
  
    const [rows] = await db.query(sql);  
  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de estados.',  
      dados: rows  
    });  
  } catch (error) {  
    return response.status(500).json({  
      sucesso: false,  
      mensagem: 'Erro na requisição.',  
      dados: error.message  
    });  
  }  
},
```


# Teste de listagem de estados

GET  http://localhost:3333/cidades/listar-ufs  **200 OK** 48 ms 489 B

Params Body Auth Headers (3) Scripts Docs Preview Headers (8) Cookies Tests (0/0)


No Body  Preview 

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de estados.",
4   "dados": [
5     {
6       "cid_uf": "AC"
7     },
8     {
9       "cid_uf": "AL"
10    },
11    {
12      "cid_uf": "AM"
13    },
14    {
15      "cid_uf": "AP"
16    },
17    {
18      "cid_uf": "BA"
19    },
20    {
21      "cid_uf": "CE"
22    },
23    {
24      "cid_uf": "DF"
25    },
26    {
27      "cid_uf": "ES"
28    },
29    {
30      "cid_uf": "GO"
```




Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de estados.", 
4   "dados": [ 27 ]
87 }
```

# Login


```
async login(request, response) {  
  try {  
  
    const { email, senha } = request.query;   
  
    const sql = `  
      SELECT  
        usu_id, usu_nome, usu_tipo  
      FROM  
        usuarios  
      WHERE  
        usu_email = ? AND usu_senha = ? AND usu_ativo = 1;  
    `;  
  
    const values = [email, senha];  
  
    const [rows] = await db.query(sql, values);  
    const nItens = rows.length;  
  
    if (nItens < 1) {  
      return response.status(403).json({  
        sucesso: false,  
        mensagem: 'Login e/ou senha inválido.',  
        dados: null,  
      });  
    }  
  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Login efetuado com sucesso',  
      dados: rows  
    });  
  } catch (error) {  
    return response.status(500).json({  
      sucesso: false,  
      mensagem: 'Erro na requisição.',  
      dados: error.message  
    });  
  }  
},
```


# Rota do Login

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



# Teste login

GET  http://localhost:3333/login?email=bernacel@terra.com.br&senha=123456

Send 

200 OK

28 ms

133 B



Params Body Auth Headers **3** Scripts Docs

URL PREVIEW


http://localhost:3333/login?email=bernacel%40terra.com.br&senha=123456


QUERY PARAMETERS

Import from URL  Bulk Edit

+ Add  Delete all  Description

name	value
------	-------

GET  http://localhost:3333/login?email=bernacel@terra.com.br&senha=12345

Send 

403 Forbidden

21 ms

71 B



Params Body Auth Headers **3** Scripts Docs

URL PREVIEW

http://localhost:3333/login?email=bernacel%40terra.com.br&senha=12345

QUERY PARAMETERS

Import from URL  Bulk Edit

+ Add  Delete all  Description

name	value
------	-------

Preview Headers **8** Cookies Tests **0 / 0** → More

Preview 

```
1 {  
2   "sucesso": true,  
3   "mensagem": "Login efetuado com sucesso",  
4   "dados": [  
5     {  
6       "usu_id": 7,  
7       "usu_nome": "Bernardo Celestino Brandão",  
8       "usu_tipo": 2  
9     }  
10  ]  
11 }
```

Preview Headers **8** Cookies Tests **0 / 0** →

Preview 

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

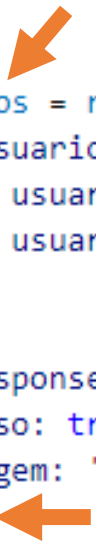
# Devolver dados com nome dos campos tratados

```
const [rows] = await db.query(sql, values);
const nItens = rows.length;

if (nItens < 1) {
  return response.status(403).json({
    sucesso: false,
    mensagem: 'Login e/ou senha inválido.',
    dados: null,
  });
}

const dados = rows.map(usuario => ({
  id: usuario.usu_id,
  nome: usuario.usu_nome,
  tipo: usuario.usu_tipo
})));

return response.status(200).json({
  sucesso: true,
  mensagem: 'Login efetuado com sucesso',
  dados
});
```



# Retorno tratado

GET http://localhost:3333/login?email=bernacel@terra.com.br&senha=123456 Send  200 OK 24 ms 121 B

Params Body Auth Headers 3 Scripts Docs

Preview Headers 8 Cookies Tests 0 / 0

URL PREVIEW

http://localhost:3333/login?email=bernacel%40terra.com.br&senha=123456

QUERY PARAMETERS

Import from URL Bulk Edit

+ Add Delete all Description

name	value			
------	-------	--	--	--

Preview

```
1 {  
2   "sucesso": true,  
3   "mensagem": "Login efetuado com sucesso",  
4   "dados": [  
5     {  
6       "id": 7,  
7       "nome": "Bernardo Celestino Brandão",  
8       "tipo": 2  
9     }  
10  ]  
11 }
```



# Pesquisa com parâmetros

```
async listarIngredientes(request, response) {  
  try {  
    const { nome } = request.query; ←  
    → const ing_nome = nome ? `%${nome}%` : ``;  
    const sql = `  
      SELECT  
        ing_id, ing_nome, ing_img, ing_custo_adicional  
      FROM  
        ingredientes  
      WHERE  
        ing_nome like ?;  
    `;  
  
    const values = [ing_nome];  
  
    const [rows] = await db.query(sql, values);  
    const nItens = rows.length;  
  
    return response.status(200).json({  
      sucesso: true,  
      mensagem: 'Lista de ingredientes.',  
      nItens,  
      dados: rows ←  
    });  
  } 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?nome=al

Send 

Params

Body 

Auth

Headers 

Scripts

Docs

http://localhost:3333/ingredientes?nome=al

```
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      "ing_id": 22,  
19      "ing_nome": "Sal",  
20      "ing_img": "sal.png",  
21      "ing_custo_adicional": "2.00"  
22    }  
23  ],  
24  "nItens": 3  
25 }
```

# Devolver dados com nome dos campos tratados

```
const [rows] = await db.query(sql, values);
const nItens = rows.length;

const dados = rows.map(ingrediente => ({
  id: ingrediente.ing_id,
  nome: ingrediente.ing_nome,
  img: ingrediente.ing_img,
  custo_adicional: ingrediente.ing_custo_adicional
}));

return response.status(200).json({
  sucesso: true,
  mensagem: 'Lista de ingredientes.',
  nItens,
  dados ←
});
```


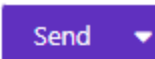


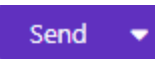
```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de ingredientes.",
4   "nItens": 3,
5   "dados": [
6     {
7       "id": 3,
8       "nome": "Salmão",
9       "img": "salmao.png",
10      "custo_adicional": "10.00"
11    },
12    {
13      "id": 4,
14      "nome": "Alface",
15      "img": "alface.png",
16      "custo_adicional": "4.50"
17    },
18    {
19      "id": 22,
20      "nome": "Sal",
21      "img": "sal.png",
22      "custo_adicional": "2.00"
23    }
24  ]
25 }
```

# Pesquisa com múltiplos parâmetros

```
4  async listarCidades(request, response) {
5    try {
6      const { uf, cidade } = request.query; ←
7
8      // Verificação obrigatória
9      if (!uf) {
10         return response.status(400).json({
11           sucesso: false,
12           mensagem: 'UF (estado) é obrigatório para listar cidades.',
13         });
14      }
15
16      // Formata o nome da cidade para busca parcial
17      const cid_nome = cidade ? `%${cidade}%` : ``; ←
18
19      // Consulta SQL parametrizada
20      const SQL_LISTAR_CIDADES = `
21        SELECT
22          cid_id, cid_nome, cid_uf
23        FROM
24          cidades
25        WHERE
26          cid_uf = ? AND cid_nome LIKE ?;
27      `;
28
29      const values = [uf.toUpperCase(), cid_nome]; ←
30      const [rows] = await db.query(SQL_LISTAR_CIDADES, values);
31
32      const dados = rows.map(municipio => ({
33        uf: municipio.cid_uf,
34        cidade: municipio.cid_nome
35      }));
```

```
36
37
38
39         return response.status(200).json({
40           sucesso: true,
41           mensagem: dados.length > 0 ? 'Lista de cidades encontrada com sucesso.' : 'Nenhuma cidade encontrada com os critérios fornecidos.', ←
42           nItens: dados.length,
43           dados
44         });
45
46       } catch (error) {
47         console.error('Erro ao listar cidades:', error);
48         return response.status(500).json({
49           sucesso: false,
50           mensagem: 'Erro interno ao listar cidades.',
51           dados: error.message
52         });
53       }
54     },
```

# Teste pesquisa com múltiplos parâmetros

GET  <a href="http://localhost:3333/cidades/listar-cidades">http://localhost:3333/cidades/listar-cidades</a> 	<b>400 Bad Request</b> 21 ms 79 B
Params Body Auth Headers <b>3</b> Scripts Docs	Preview Headers <b>8</b> Cookies Tests 0 / 0 → Mock Console
No Body 	Preview 
	<pre>1 { 2   "sucesso": false, 3   "mensagem": "UF (estado) é obrigatório para listar cidades." 4 }</pre>
GET <a href="http://localhost:3333/cidades/listar-cidades?uf=sp">http://localhost:3333/cidades/listar-cidades?uf=sp</a> 	<b>200 OK</b> 119 ms 22.1 KB
Params Body Auth Headers <b>3</b> Scripts Docs	Preview Headers <b>8</b> Cookies Tests 0 / 0 → Mock Console
No Body	Preview
	<pre>1 { 2   "sucesso": true, 3   "mensagem": "Lista de cidades encontrada com sucesso.", 4   "nItens": 645, 5   "dados": [ 6     { 7       "uf": "SP", 8       "cidade": "Adamantina" 9     }, 10    { 11      "uf": "SP", 12      "cidade": "Adolfo" 13    }, 14    { 15      "uf": "SP", 16      "cidade": "Aguai" 17    } 18  ] 19 }</pre>

# Test pesquisa com múltiplos parâmetros




The screenshot shows a REST client interface with a GET request to `http://localhost:3333/cidades/listar-cidades?uf=sp&cidade=tup`. The response is a 200 OK status with a 39 ms response time and 223 B of data. The response body is a JSON object with the following structure:

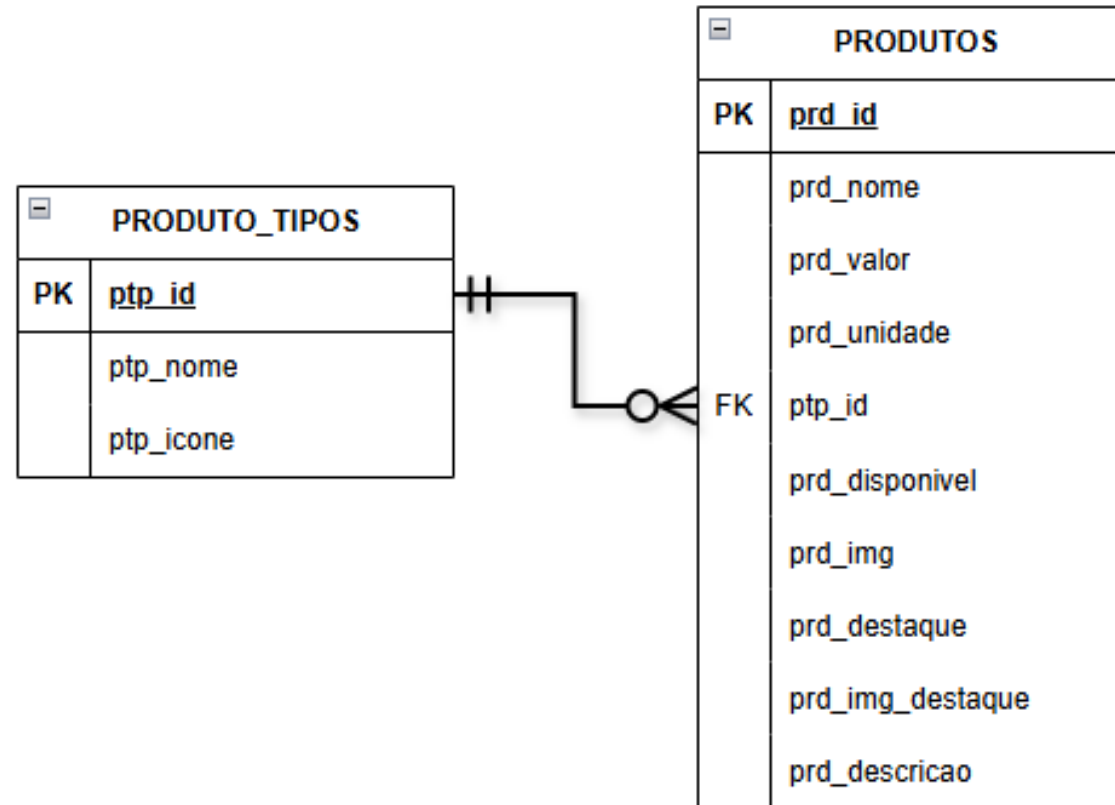
```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de cidades encontrada com sucesso.",
4   "nItens": 4,
5   "dados": [
6     {
7       "uf": "SP",
8       "cidade": "Itupeva"
9     },
10    {
11      "uf": "SP",
12      "cidade": "Tupã"
13    },
14    {
15      "uf": "SP",
16      "cidade": "Tupi Paulista"
17    },
18    {
19      "uf": "SP",
20      "cidade": "Votuporanga"
21    }
22  ]
23 }
```

Two orange arrows point to the query parameters `uf=sp` and `cidade=tup` in the URL. Another orange arrow points to the `"nItens": 4` value in the JSON response.

`http://localhost:3333/cidades/listar-cidades?uf=SP&cidade=Tup`



# Uso de Inner Join para chave estrangeira e query dinâmica



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 Empanado	31.00	un.	1	1	lancheFrangoEmpanado.jpg	0	NULL	Pão e frango empanado
2	Lanche de Salmão	38.00	un.	1	1	lancheSalmao.jpg	0	NULL	Pão, filé de salmão temperado com ervas finas
3	Lanche de Salada	28.00	un.	1	1	lancheVegetariano.jpeg	0	NULL	Pão, alface, tomate, rúcula, milho, pepino e asp...
4	Batata frita	25.40	un.	2	1	fritas.jpg	1	promoGenerica.png	Batata de qualidade internacional.
5	Suco de Abacaxi	12.00	copo	3	1	sucoAbacaxi.png	0	NULL	Abacaxi, açúcar e gelo
6	Suco de Uva	15.00	copo	3	1	sucoUva.jpeg	0	NULL	Uva, açúcar e gelo
7	Suco de Laranja	6.99	copo	3	1	sucoLaranja.jpg	1	promoSucoLaranja.png	Laranja, açúcar e gelo
8	Suco de Limão	12.00	copo	3	1	sucoLimaao.jpg	0	NULL	Limão, açúcar e gelo
9	Biscoito Amanteigado	12.99	un.	4	0	biscoitoAmanteigado.png	0	NULL	Biscoito saboroso
10	Lanche de Peixe	20.99	un.	1	1	lanchePeixe.jpg	1	promoLanchePeixe.png	Feito com peixes do aquário
11	Salada da Casa	16.99	un.	1	1	salada.jpg	1	promoSaladaDaCasa.png	Feito com peixes do aquário
12	Sorvete Chocolate	11.00	un.	4	1	sorveteChocolate.jpeg	0	NULL	Refrescante e delicioso
13	Torta de creme de coco	19.00	un.	4	1	tortaCremeCoco.jpg	0	NULL	Torta doce e cremosa
14	Lanche de bacon	34.00	un.	1	1	hamburger-bacon.jpg	0	NULL	Hamburguer bacon e couve
15	Combo hamburguer + batata	48.00	un.	1	1	hamburger-batata.jpg	0	NULL	Lanche de hamburguer com salada mais uma po...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ptp_id	ptp_nome	ptp_icone
1	Lanche	lanche.svg
2	Porção	porcao.svg
3	Suco	suco.svg
4	Sobremesa	doce.svg
NULL	NULL	NULL



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

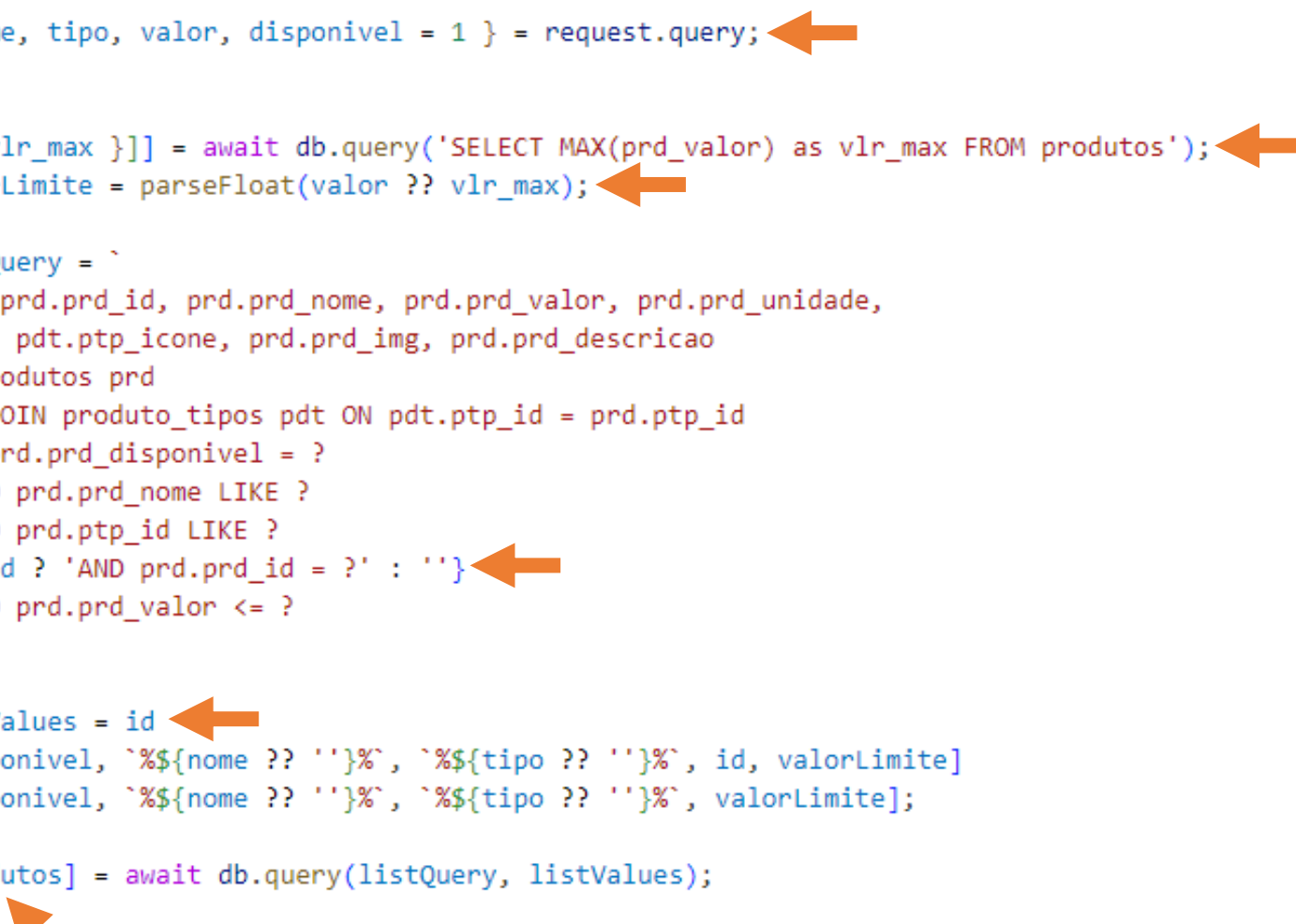
INNER JOIN

    produto\_tipos pdt ON pdt.ptp\_id = prd.ptp\_id;


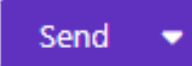
prd_id	prd_nome	prd_valor	prd_unidade	ptp_icone	prd_img	prd_descricao
1	Lanche de Frango Empanado	31.00	un.	lanche.svg	lancheFrangoEmpanado.jpg	Pão e frango empanado
2	Lanche de Salmão	38.00	un.	lanche.svg	lancheSalmao.jpg	Pão, filé de salmão temperado com ervas finas
3	Lanche de Salada	28.00	un.	lanche.svg	lancheVegetariano.jpeg	Pão, alface, tomate, rúcula, milho, pepino e asp...
4	Batata frita	25.40	un.	porcao.svg	fritas.jpg	Batata de qualidade internacional.
5	Suco de Abacaxi	12.00	copo	suco.svg	sucoAbacaxi.png	Abacaxi, açúcar e gelo
6	Suco de Uva	15.00	copo	suco.svg	sucoUva.jpeg	Uva, açúcar e gelo
7	Suco de Laranja	6.99	copo	suco.svg	sucoLaranja.jpg	Laranja, açúcar e gelo
8	Suco de Limão	12.00	copo	suco.svg	sucoLimao.jpg	Limão, açúcar e gelo
9	Biscoito Amanteigado	12.99	un.	doce.svg	biscoitoAmanteigado.png	Biscoito saboroso
10	Lanche de Peixe	20.99	un.	lanche.svg	lanchePeixe.jpg	Feito com peixes do aquário
11	Salada da Casa	16.99	un.	lanche.svg	salada.jpg	Feito com peixes do aquário
12	Sorvete Chocolate	11.00	un.	doce.svg	sorveteChocolate.jpeg	Refrescante e delicioso
13	Torta de creme de coco	19.00	un.	doce.svg	tortaCremeCoco.jpg	Torta doce e cremosa
14	Lanche de bacon	34.00	un.	lanche.svg	hamburger-bacon.jpg	Hamburguer bacon e couve
15	Combo hamburguer + batata	48.00	un.	lanche.svg	hamburger-batata.jpg	Lanche de hamburguer com salada mais uma po...

# Ajustes para filtro de produtos por parâmetro

```
4  async listarProdutos(request, response) {
5
6      const { id, nome, tipo, valor, disponivel = 1 } = request.query;
7
8      try {
9          const [[{ vlr_max }]] = await db.query('SELECT MAX(prd_valor) as vlr_max FROM produtos');
10         const valorLimite = parseFloat(valor ?? vlr_max);
11
12         const listQuery = `
13             SELECT prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade,
14                 pdt.ptp_icone, prd.prd_img, prd.prd_descricao
15             FROM produtos prd
16             INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id
17             WHERE prd.prd_disponivel = ?
18                 AND prd.prd_nome LIKE ?
19                 AND prd.ptp_id LIKE ?
20                 ${id ? 'AND prd.prd_id = ?' : ''}
21                 AND prd.prd_valor <= ?
22         `;
23
24         const listValues = id
25             ? [disponivel, `${nome ?? ''}%`, `${tipo ?? ''}%`, id, valorLimite]
26             : [disponivel, `${nome ?? ''}%`, `${tipo ?? ''}%`, valorLimite];
27
28         const [produtos] = await db.query(listQuery, listValues);
29     }
```




```
30     const dados = produtos.map(produto => ({
31       id: produto.prđ_id,
32       nome: produto.prđ_nome,
33       valor: produto.prđ_valor,
34       unidade: produto.prđ_unidade,
35       icone: produto.ptp_icone,
36       imgProduto: produto.prđ_img,
37       descricao: produto.prđ_descricao
38     }));
39
40     return response.status(200).json({
41       sucesso: true,
42       mensagem: 'Lista de produtos',
43       nItens: dados.length,
44       dados
45     });
46
47   } catch (error) {
48     console.error('Erro ao listar produtos:', error);
49     return response.status(500).json({
50       sucesso: false,
51       mensagem: 'Erro ao listar produtos.',
52       dados: error.message
53     });
54   }
55 },
```


GET  http://localhost:3333/produtos  200 OK 32 ms 2.3 KB



Params Body Auth Headers **3** Scripts Docs




Preview Headers **8** Cookies Tests 0/0 → Mock


URL PREVIEW


http://localhost:3333/produtos 


QUERY PARAMETERS Import from URL  Bulk Edit

+ Add  Delete all  Description


name	value	
		  

Preview 

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 14, 
5   "dados": [
6     {
7       "id": 1,
8       "nome": "Lanche de Frango Empanado",
9       "valor": "31.00",
10      "unidade": "un.",
11      "icone": "lanche.svg",
12      "imgProduto": "lancheFrangoEmpanado.jpg",
13      "descricao": "Pão e frango empanado"
14    },
15    {
16      "id": 2,
17      "nome": "Lanche de Salmão"
```

  
`const { id, nome, tipo, valor, disponivel = 1 } = request.query;`

GET  http://localhost:3333/produtos?disponivel=0

Send 

200 OK

58 ms

229 B

Params

Body

Auth

Headers

3

Scripts

Docs

Preview

Headers

8

Cookies

Tests

0 / 0

→ Mock

URL PREVIEW

http://localhost:3333/produtos?disponivel=0




QUERY PARAMETERS

Import from URL  Bulk Edit

+ Add

 Delete all


 Description

name

value



GET  http://localhost:3333/produtos?id=1

Send 

200 OK

26 ms

242 B

Params

Body

Auth

Headers

3

Scripts

Docs

Preview

Headers

8

Cookies

Tests

0 / 0

→ Mock

URL PREVIEW

http://localhost:3333/produtos?id=1



QUERY PARAMETERS

Import from URL  Bulk Edit

+ Add

 Delete all

 Description

name

value



Preview 

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 1,
5   "dados": [
6     {
7       "id": 9,
8       "nome": "Biscoito Amanteigado",
9       "valor": "12.99",
10      "unidade": "un.",
11      "icone": "doce.svg",
12      "imgProduto": "biscoitoAmanteigado.png",
13      "descricao": "Biscoito saboroso"
14    }
15  ]
16 }
```

Preview 

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 1,
5   "dados": [
6     {
7       "id": 1,
8       "nome": "Lanche de Frango Empanado",
9       "valor": "31.00",
10      "unidade": "un.",
11      "icone": "lanche.svg",
12      "imgProduto": "lancheFrangoEmpanado.jpg",
13      "descricao": "Pão e frango empanado"
14    }
15  ]
16 }
```

GET http://localhost:3333/produtos?nome=salada 200 OK 16 ms 419 B

Params Body Auth Headers 3 0/0 Docs

URL PREVIEW

http://localhost:3333/produtos?nome=salada

QUERY PARAMETERS

Import from URL Bulk Edit

+ Add Delete all Description

name	value			

Preview

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 2,
5   "dados": [
6     {
7       "id": 3,
8       "nome": "Lanche de Salada",
9       "valor": "28.00",
10      "unidade": "un.",
11      "icone": "lanche.svg",
12      "imgProduto": "lancheVegetariano.jpeg",
13      "descricao": "Pão, alface, tomate, rúcula, milho, pepino e aspargo"
14    },
15    {
16      "id": 11,
17      "nome": "Salada da Casa",
18      "valor": "16.99",
19      "unidade": "un.",
20      "icone": "lanche.svg",
21      "imgProduto": "salada.jpg",
22      "descricao": "Feito com peixes do aquário"
23    }
24  ]
25 }
```

GET http://localhost:3333/produtos?nome=salada&valor=20 200 OK 18 ms 224 B

Params Body Auth Headers 3 0/0 Docs

URL PREVIEW

http://localhost:3333/produtos?nome=salada&valor=20

QUERY PARAMETERS

Import from URL Bulk Edit

+ Add Delete all Description

name	value			

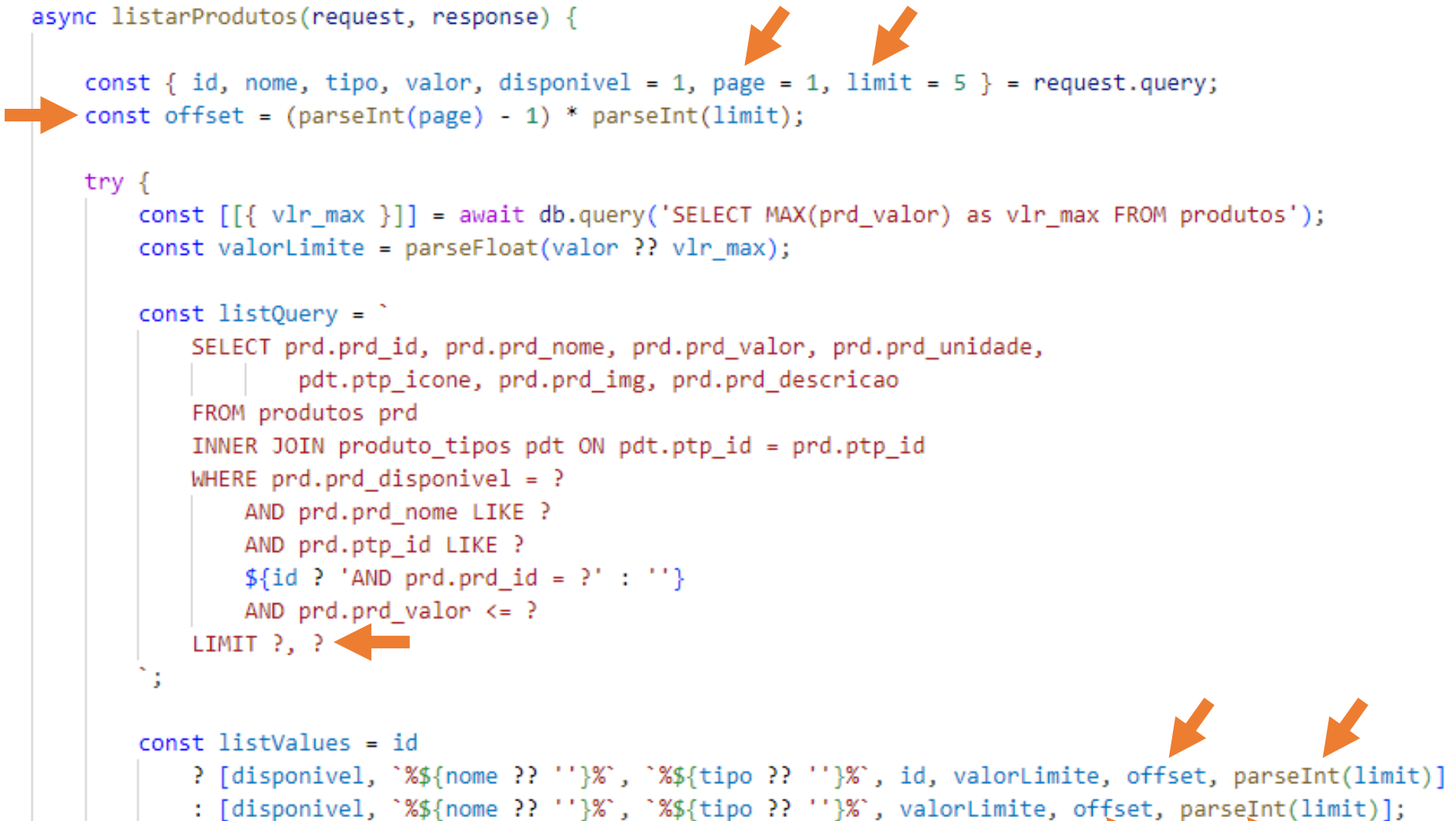
Preview

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 1,
5   "dados": [
6     {
7       "id": 11,
8       "nome": "Salada da Casa",
9       "valor": "16.99",
10      "unidade": "un.",
11      "icone": "lanche.svg",
12      "imgProduto": "salada.jpg",
13      "descricao": "Feito com peixes do aquário"
14    }
15  ]
16 }
```

http://localhost:3333/produtos?nome=salada&valor=20

# Paginação

```
4   async listarProdutos(request, response) {  
5  
6       const { id, nome, tipo, valor, disponivel = 1, page = 1, limit = 5 } = request.query;  
7       const offset = (parseInt(page) - 1) * parseInt(limit);  
8  
9       try {  
10          const [{ vlr_max }] = await db.query('SELECT MAX(prd_valor) as vlr_max FROM produtos');  
11          const valorLimite = parseFloat(valor ?? vlr_max);  
12  
13          const listQuery = `  
14              SELECT prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade,  
15                  |      pdt.ptp_icone, prd.prd_img, prd.prd_descricao  
16              FROM produtos prd  
17              INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id  
18              WHERE prd.prd_disponivel = ?  
19                  AND prd.prd_nome LIKE ?  
20                  AND prd.ptp_id LIKE ?  
21                  ${id ? 'AND prd.prd_id = ?' : ''}  
22                  AND prd.prd_valor <= ?  
23              LIMIT ?, ?  
24          `;  
25  
26          const listValues = id  
27              ? [disponivel, `%${nome} ?? ''`, `%${tipo} ?? ''`, id, valorLimite, offset, parseInt(limit)]  
28              : [disponivel, `%${nome} ?? ''`, `%${tipo} ?? ''`, valorLimite, offset, parseInt(limit)];
```



GET <http://localhost:3333/produtos>

Send

200 OK

18 ms

935 B

Params

Body

Auth

Headers

3

Scripts

Docs

Preview

Headers

8

Cookies

Tests 0 / 0

→ Mock

URL PREVIEW

<http://localhost:3333/produtos>



QUERY PARAMETERS

Import from URL Bulk Edit

+ Add



Delete all



Description

name

value



Preview

```
1 {  
2   "sucesso": true,  
3   "mensagem": "Lista de produtos",  
4   "nItens": 5,  
5   "dados": [  
6     {  
7       "id": 1,  
8       "nome": "Lanche de Frango Empanado",  
9       "valor": "31.00",  
10      "unidade": "un.",  
11      "icone": "lanche.svg",  
12      "imgProduto": "lancheFrangoEmpanado.jpg",  
13      "descricao": "Pão e frango empanado"  
14    },  
15    {  
16      "id": 2,  
17      "nome": "Lanche de Calmão",  
18      "descricao": "Doleta de qualquer internacional."  
19    },  
20    {  
21      "id": 3,  
22      "nome": "Lanche de Bife",  
23      "descricao": "Doleta de qualquer internacional."  
24    },  
25    {  
26      "id": 4,  
27      "nome": "Lanche de Hambúrguer",  
28      "descricao": "Doleta de qualquer internacional."  
29    },  
30    {  
31      "id": 5,  
32      "nome": "Suco de Abacaxi",  
33      "valor": "12.00",  
34      "unidade": "copo",  
35      "icone": "suco.svg",  
36      "imgProduto": "sucoAbacaxi.png",  
37      "descricao": "Abacaxi, açúcar e gelo"  
38    }  
39  ]  
40 }  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```



GET <http://localhost:3333/produtos?page=2&limit=3>

Send

200 OK

16 ms

526 B

Params

Body

Auth

Headers

3

Scripts

Docs

Preview

Headers

8

Cookies

Tests 0 / 0

→ Mock

Co

#### URL PREVIEW

<http://localhost:3333/produtos?page=2&limit=3>



#### QUERY PARAMETERS

Import from URL Bulk Edit

+ Add



Delete all



Description

name

value




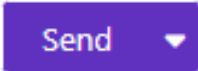
#### PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'





#### Preview



```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 3,
5   "dados": [
6     {
7       "id": 4,
8       "nome": "Batata frita",
9       "valor": "25.40",
10      "unidade": "un.",
11      "icone": "porcao.svg",
12      "imgProduto": "fritas.jpg",
13      "descricao": "Batata de qualidade internacional."
14    },
15    {
16      "id": 5,
17      "nome": "Suco de Abacaxi",
18      "valor": "12.00",
19      "unidade": "copo",
20      "icone": "suco.svg",
21      "imgProduto": "sucoAbacaxi.png",
22      "descricao": "Abacaxi, açúcar e gelo"
23    },
24    {
25      "id": 6,
26      "nome": "Suco de Uva",
27      "valor": "15.00",
28      "unidade": "copo",
29      "icone": "suco.svg",
30      "imgProduto": "sucoUva.jpeg",
31      "descricao": "Uva, açúcar e gelo"
32    }
33  ]
34 }
```


GET  http://localhost:3333/produtos?tipo=1&page=1&limit=3  200 OK 18 ms 621 B


Params Body Auth Headers **3** Scripts Docs


URL PREVIEW  
http://localhost:3333/produtos?tipo=1&page=1&limit=3 


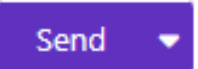
QUERY PARAMETERS Import from URL  Bulk Edit

+ Add  Delete all  Description


Preview  Headers **8** Cookies Tests 0/0


Preview 



```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 3, 
5   "dados": [
6     {
7       "id": 1,
8       "nome": "Tanche de Frango Empanado"
```




GET  http://localhost:3333/produtos?tipo=1&page=3&limit=3  200 OK 17 ms 276 B


Params Body Auth Headers **3** Scripts Docs


URL PREVIEW  
http://localhost:3333/produtos?tipo=1&page=3&limit=3 


QUERY PARAMETERS Import from URL  Bulk Edit

+ Add  Delete all  Description

name	value	
		  

Preview  Headers **8** Cookies Tests 0/0

Preview 

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 1, 
5   "dados": [
6     {
7       "id": 15,
8       "nome": "Combo hamburguer + batata",
9       "valor": "14.99"
```

?7?

# Total de itens

```
4  async listarProdutos(request, response) {
5
6      const { id, nome, tipo, valor, disponivel = 1, page = 1, limit = 5 } = request.query;
7      const offset = (parseInt(page) - 1) * parseInt(limit);
8
9      try {
10         const [[{ vlr_max }]] = await db.query('SELECT MAX(prd_valor) as vlr_max FROM produtos');
11         const valorLimite = parseFloat(valor ?? vlr_max);
12
13         const countQuery = `
14             SELECT COUNT(*) AS total
15             FROM produtos prd
16             INNER JOIN produto_tipos pdt ON pdt.ptp_id = prd.ptp_id
17             WHERE prd.prd_disponivel = ?
18                 AND prd.prd_nome LIKE ?
19                 AND prd.ptp_id LIKE ?
20                 ${id ? 'AND prd.prd_id = ?' : ''}
21                 AND prd.prd_valor <= ?
22         `;
23
24         const countValues = id
25             ? [disponivel, `%${nome} ?? '%`%, `%${tipo} ?? '%`, id, valorLimite]
26             : [disponivel, `%${nome} ?? '%`%, `%${tipo} ?? '%`, valorLimite];
27
28         const [[{ total }]] = await db.query(countQuery, countValues);
29
30         const listQuery = `
31             SELECT prd.prd_id, prd.prd_nome, prd.prd_valor, prd.prd_unidade,
32
33             imgproduto: produto.prd_img,
34             descricao: produto.prd_descricao
35         `;
36
37         response.setHeader('X-Total-Count', total);
38         return response.status(200).json({
39             sucesso: true,
40             mensagem: 'Lista de produtos',
41             nItens: dados.length,
42             dados
43         });
44     } catch (error) {
```

GET http://localhost:3333/produtos?tipo=1&page=1&limit=3

Send

200 OK

27 ms

621 B

Params Body Auth Headers 3 Scripts Docs

Preview Headers 9 Cookies Tests 0/0 → Mock Console

#### URL PREVIEW

http://localhost:3333/produtos?tipo=1&page=1&limit=3

#### QUERY PARAMETERS

Import from URL Bulk Edit

+ Add Delete all Description

name	value
------	-------

#### PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'

#### Preview

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos",
4   "nItens": 3,
5   "dados": [
6     {
7       "imgProduto": "lanchevegetariano.jpeg",
8       "descricao": "Pão, alface, tomate, rúcula, milho, pepino e aspargo"
9     }
10  ]
11 }
```

#### Preview

#### Headers

9

#### Cookies

Tests 0/0

→ Mock

#### Console

#### NAME

#### VALUE

X-Powered-By

Express

Access-Control-Allow-Origin

\*

X-Total-Count

7

Content-Type

application/json; charset=utf-8

Content-Length

621

ETag

W/"26d-rLXUvMGaOLsAfKTntQ4YwuSUKNw"

Date

Sat, 17 May 2025 01:39:30 GMT

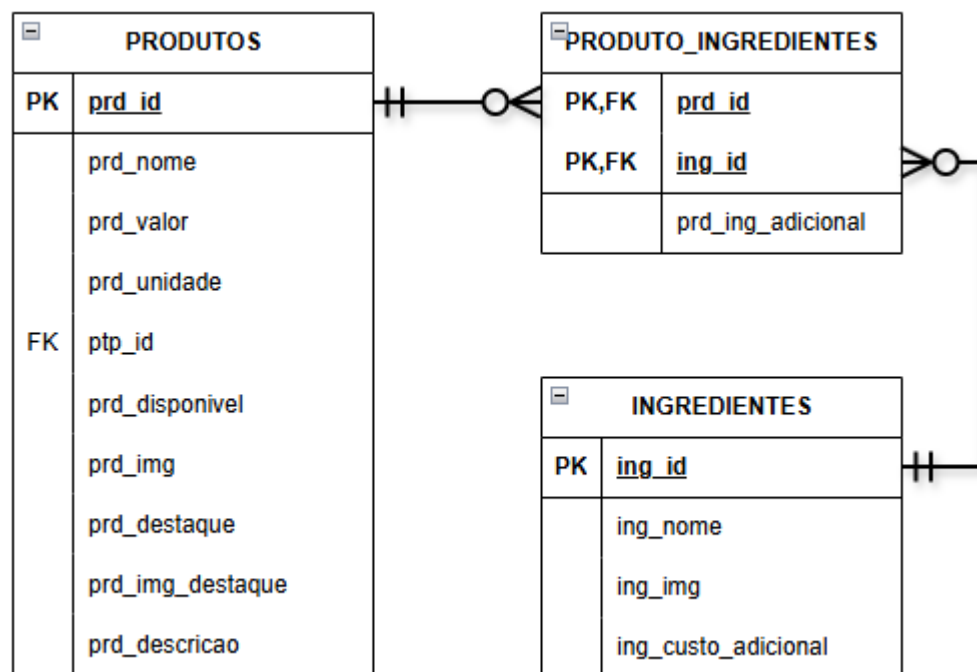
Connection

keep-alive

Keep-Alive

timeout=5

# Uso de Inner Join para relacionamento n:n



SELECT

```
p.prd_id AS id,
p.prd_nome AS nome,
p.prd_valor AS valor,
p.prd_unidade AS unidade,
p.prd_disponivel AS disponivel,
p.prd_img AS imagem,
p.prd_descricao AS descricao,

pdtp.ptp_nome AS nomeTipo,
pdtp.ptp_icone AS iconeTipo,

i.ing_id AS idIngrediente,
i.ing_nome AS nomeIngrediente,
i.ing_img AS imagemIngrediente,
i.ing_custo_adicional AS custoAdicionalIngrediente,

pi.prd_ing_adicional AS adicionalProdutoIngrediente
FROM
    produtos p
JOIN
    produto_ingredientes pi ON pi.prd_id = p.prd_id
JOIN
    ingredientes i ON i.ing_id = pi.ing_id
JOIN
    produto_tipos pdtp ON pdtp.ptp_id = p.ptp_id
WHERE
    p.prd_id = 1;
```

# Itens retornados pela instrução

id	nome	valor	unidade	disponivel	imagem	descricao	idTipo	nomeTipo	iconeTipo	idIngrediente	nomeIngrediente	imagemIngrediente	custoAdicionalIngrediente	adicionalProdutoIngrediente
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	1	Pão	pao.png	0.00	0
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	2	Frango	frango.png	7.00	0
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	3	Salmão	salmão.png	10.00	0
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	4	Alface	alface.png	4.50	0
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	5	Rúcula	rucula.png	4.00	0
1	Lanche de Frango Empanado	31.00	un.	1	lancheFrangoEmpanado.jpg	Pão e frango empanado	1	Lanche	lanche.svg	6	Tomate	tomate.png	5.25	0

```

161 async listarIngredientesDoProduto(request, response) {
162     try {
163         const { id } = request.params;
164
165         const sql = `
166             SELECT
167                 p.prđ_id AS id,
168                 p.prđ_nome AS nome,
169                 p.prđ_valor AS valor,
170                 p.prđ_unidade AS unidade,
171                 p.prđ_disponivel AS disponivel,
172                 p.prđ_img AS imagem,
173                 p.prđ_descricao AS descricao,
174
175                 pdtp.ptp_nome AS nomeTipo,
176                 pdtp.ptp_icone AS iconeTipo,
177
178                 i.ing_id AS idIngrediente,
179                 i.ing_nome AS nomeIngrediente,
180                 i.ing_img AS imagemIngrediente,
181                 i.ing_custo_adicional AS custoAdicionalIngrediente,
182
183                 pi.prđ_ing_adicional = 1 AS adicionalProdutoIngrediente
184             FROM
185                 produtos p
186             JOIN
187                 produto_ingredientes pi ON pi.prđ_id = p.prđ_id
188             JOIN
189                 ingredientes i ON i.ing_id = pi.ing_id
190             JOIN
191                 produto_tipos pdtp ON pdtp.ptp_id = p.ptp_id
192             WHERE
193                 p.prđ_id = ?;
194         `;
195
196

```


```

197     const [rows] = await db.query(sql, [id]);
198
199     if (rows.length === 0) {
200         return response.status(404).json({
201             sucesso: false,
202             mensagem: `Produto com id ${id} não encontrado ou sem ingredientes.`,
203             dados: null
204         });
205     }
206
207     // Extrai dados do produto (só do primeiro registro, pois todos têm os mesmos valores)
208     const produto = {
209         id: rows[0].id,
210         nome: rows[0].nome,
211         valor: parseFloat(rows[0].valor).toFixed(2),
212         unidade: rows[0].unidade,
213         disponivel: !!rows[0].disponivel,
214         img: rows[0].imagem,
215         descricao: rows[0].descricao,
216         tipoNome: rows[0].nomeTipo,
217         tipoIcône: rows[0].iconeTipo,
218         ingredientes: rows.map(row => ({
219             id: row.idIngrediente,
220             nome: row.nomeIngrediente,
221             quantidade: row.imagemIngrediente,
222             unidade: row.custoAdicionalIngrediente,
223             adicional: row.adicionalProdutoIngrediente
224         })),
225     };
226
227     return response.status(200).json({
228         sucesso: true,
229         mensagem: `Ingredientes do produto ${produto.nome}`,
230         dados: produto
231     });
232
233 } catch (error) {
234     return response.status(500).json({
235         sucesso: false,
236         mensagem: 'Erro na requisição.',
237         dados: error.message
238     });
239 }
240 },

```

```
router.get('/produtos', ProdutosController.listarProdutos);
router.post('/produtos', ProdutosController.cadastrarProdutos);
router.patch('/produtos', ProdutosController.editarProdutos);
router.delete('/produtos', ProdutosController.apagarProdutos);
→ router.get('/produtos/:id', ProdutosController.listarIngredientesDoProduto);
```

GET  http://localhost:3333/produtos/1

Send 

200 OK

14 ms

803 B

Params

Body

Auth

Headers  3

Scripts

Docs

Preview

Headers  8

Cookies

Tests 0 / 0

→ Mock

Console

URL PREVIEW

http://localhost:3333/produtos/1



QUERY PARAMETERS

Import from URL  Bulk Edit

+ Add

 Delete all

 Description

name

value



PATH PARAMETERS

 Path parameters are url path segments that start with a colon ':' e.g. ':id'

×

Preview 

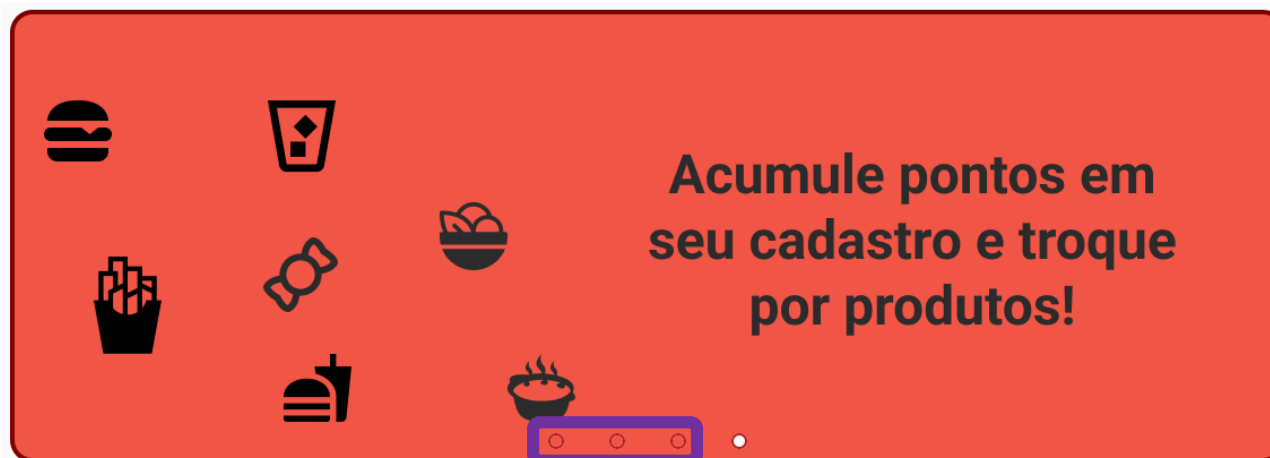
```
1 {
2   "sucesso": true,
3   "mensagem": "Ingredientes do produto Lanche de Frango Empanado",
4   "dados": {
5     "id": 1,
6     "nome": "Lanche de Frango Empanado",
7     "valor": "31.00",
8     "unidade": "un.",
9     "disponivel": true,
10    "img": "lancheFrangoEmpanado.jpg",
11    "descricao": "Pão e frango empanado",
12    "tipoNome": "Lanche",
13    "tipoIcône": "lanche.svg",
14    "ingredientes": [
15      {
16        "id": 1,
17        "nome": "Pão",
18        "quantidade": "pao.png",
19        "unidade": "0.00",
20        "adicional": 0
21      },
22      {
23        "id": 2,
24        "nome": "Frango",
25        "quantidade": "frango.png",
26        "unidade": "7.00",
27        "adicional": 0
28      },
29      {
30        "id": 3,
31        "nome": "Caldeirão"
```



# Exemplo select com dados aleatórios

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


```
async listarPromocoes(request, response) {
  try {
    const sql= `
      SELECT prd_img_destaque AS imgDestaque FROM produtos
      WHERE prd_destaque = 1
      ORDER BY RAND()
      LIMIT 3;
    `;

    const [promo] = await db.query(sql);

    return response.status(200).json({
      sucesso: true,
      mensagem: `Produtos em promoção.`,
      dados: promo
    });
  } catch (error) {
    return response.status(500).json({
      sucesso: false,
      mensagem: 'Erro na requisição.',
      dados: error.message
    });
  }
},
```

Depois adicionar uma rota para acessar os produtos da promoção, **neste exemplo a ordem das rotas importa!**

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



# 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 }				