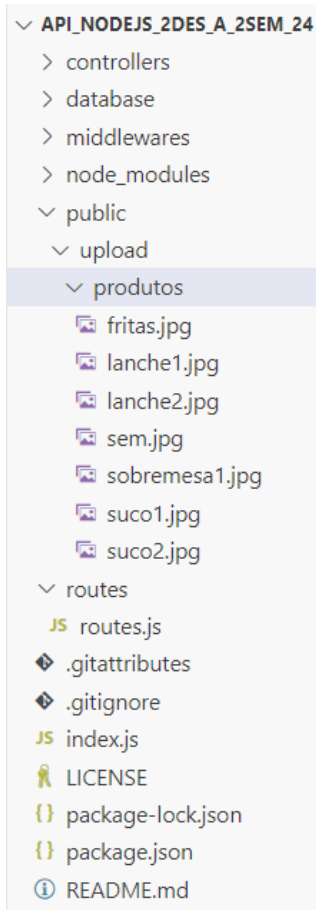


Trabalhando com imagens

Prof. Me. Ewerton José da Silva

Criar pasta publica com imagens que possam ser acessadas de forma externa




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	lanche2.jpg	0	NULL	Pão, frango desfiado e temperado
2	Lanche de Salmão	28.00	un.	1	1	lanche1.jpg	1		Pão, filé de salmão temperado com ervas finas
3	Lanche de Salada	18.00	un.	1	1	sem.jpg	0	NULL	Pão, alface, tomate, rúcula, milho, pepino e asp...
4	Batata frita	17.20	un.	2	1	fritas.jpg	1		Batata de qualidade internacional.
5	Suco de Abacaxi	12.00	copo	3	1	sem.jpg	1		Abacaxi, açúcar e gelo
6	Suco de Uva	15.00	copo	3	1	sem.jpg	1		Uva, açúcar e gelo
7	Suco de Laranja	12.00	copo	3	1	suco1.jpg	0	promoSuco.jpg	Laranja, açúcar e gelo
8	Suco de Limão	12.00	copo	3	1	suco2.jpg	0	NULL	Limão, açúcar e gelo
9	Biscoito	3.99	un.	4	0	sobremesa1.jpg	0	NULL	Biscoito saboroso
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

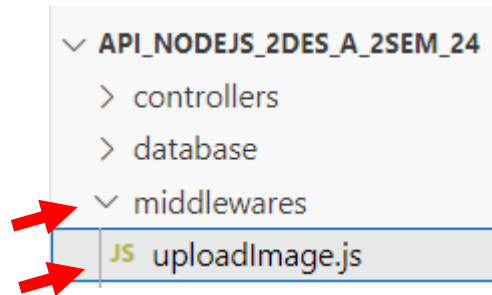
Instalar a biblioteca para upload de imagens e carregamento de imagens
npm i multer
npm i fs-extra

Tornar a pasta public pública.

```
JS index.js > ...
1  const express = require('express');
2  const cors = require('cors');
3
4  const router = require('./routes/routes');
5
6  const app = express();
7  app.use(cors());
8  app.use(express.json());
9  app.use(router);
10
11 // tornando as pasta public acessível para imagens
12 app.use('/public', express.static('public'));
13
14 // const porta = process.env.PORT || 3333;
15 const porta = 3333;
16
17 app.listen(porta, () => {
18 |   console.log(`Servidor iniciado na porta ${porta}`);
19 | });
20
21 app.get('/', (request, response) => {
22 |   response.send('Hello World');
23 | });
24
```



Código para upload de imagens



```
middlewares > JS uploadImage.js > ...
1  const multer = require('multer');
2
3  const storage = multer.diskStorage({
4    destination: function (req, file, cb) {
5      cb(null, './public/upload/produtos/');
6    },
7    filename: function (req, file, cb) {
8      //let data = new Date().toISOString().replace(/:/g, '-') + '-';
9      //let data = Date.now().toString();
10     const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
11     // identificar extensão
12     const ext = file.mimetype === 'image/jpeg' ? '.jpeg' : file.mimetype.slice(file.mimetype.length - 3);
13     cb(null, file.fieldname + '-' + uniqueSuffix + ext);
14     //cb(null, data + '_' + file.originalname);
15   }
16 });
17
18 const fileFilter = (req, file, cb) => {
19   if (file.mimetype === 'image/jpg' || file.mimetype === 'image/jpeg' || file.mimetype === 'image/png') {
20     cb(null, true);
21   } else {
22     cb(null, false);
23   }
24 }
25
26 module.exports = (multer({
27   storage: storage,
28   limits: {
29     fileSize: 1024 * 1024 * 5
30   },
31   fileFilter: fileFilter
32 }));
33
```

Ajuste na rota para chamada do upload

routes > JS routes.js > ...

```
15  const ProdutoTiposController = require('../controllers/produtoTipos');
16
17  → const upload = require('../middlewares/uploadImage');
18
19  router.get('/usuarios', UsuariosController.listarUsuarios);
20  router.post('/usuarios', UsuariosController.cadastrarUsuarios); //body
```

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

Ajuste no controller produtos para listagem das imagens

controllers > JS produtos.js > ...

```
1  const db = require('../database/connection');
2  var fs = require('fs-extra');
3
4  function geraUrl (e) {
5
6      // garantir que valores em branco carreguem algo
7      let img = e.prđ_img ? e.prđ_img : 'sem.jpg';
8      // verifica se imagem existe
9      if (!fs.existsSync('../public/upload/produtos/' + img)) {
10         img = 'sem.jpg';
11     }
12
13     const produto = {
14         prđ_id: e.prđ_id,
15         prđ_nome: e.prđ_nome,
16         ptp_id: e.ptp_id,
17         ptp_nome: e.ptp_nome,
18         prđ_valor: e.prđ_valor,
19         prđ_unidade: e.prđ_unidade,
20         prđ_disponivel: e.prđ_disponivel,
21         prđ_img: 'http://10.67.22.145:3333/public/upload/produtos/' + img,
22         prđ_destaque: e.prđ_destaque,
23         prđ_img_destaque: e.prđ_img_destaque,
24         prđ_descricao: e.prđ_descricao
25     }
26     return produto;
27 }
28
29 module.exports = {
30     async listarProdutos(request, response) {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

D:\TEMP\Ewerton\github\api_nodejs_2des_a_2sem_24>ipconfig

Configuração de IP do Windows

Adaptador Ethernet Ethernet 8:

Sufixo DNS específico de conexão.	: etectupa.local
Endereço IPv6 de link local	: fe80::a50d:1d44:927f:7bb4%14
Endereço IPv4.	: 10.67.22.145
Máscara de Sub-rede	: 255.255.254.0
Gateway Padrão.	: 10.67.22.3

D:\TEMP\Ewerton\github\api_nodejs_2des_a_2sem_24>

Implementando geração da url na listagem

```
try {
  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)];

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

  const nItens = produtos[0].length;

  // chamada para montar url
  const resultado = produtos[0].map(geraUrl);

  return response.status(200).json({
    sucesso: true,
    mensagem: 'Lista de produtos.',
    dados: resultado,
    nItens
  });
} catch (error) {
```

Teste de listagem

GET ⌵ http://localhost:3333/produtos?page=2&limit=3 Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

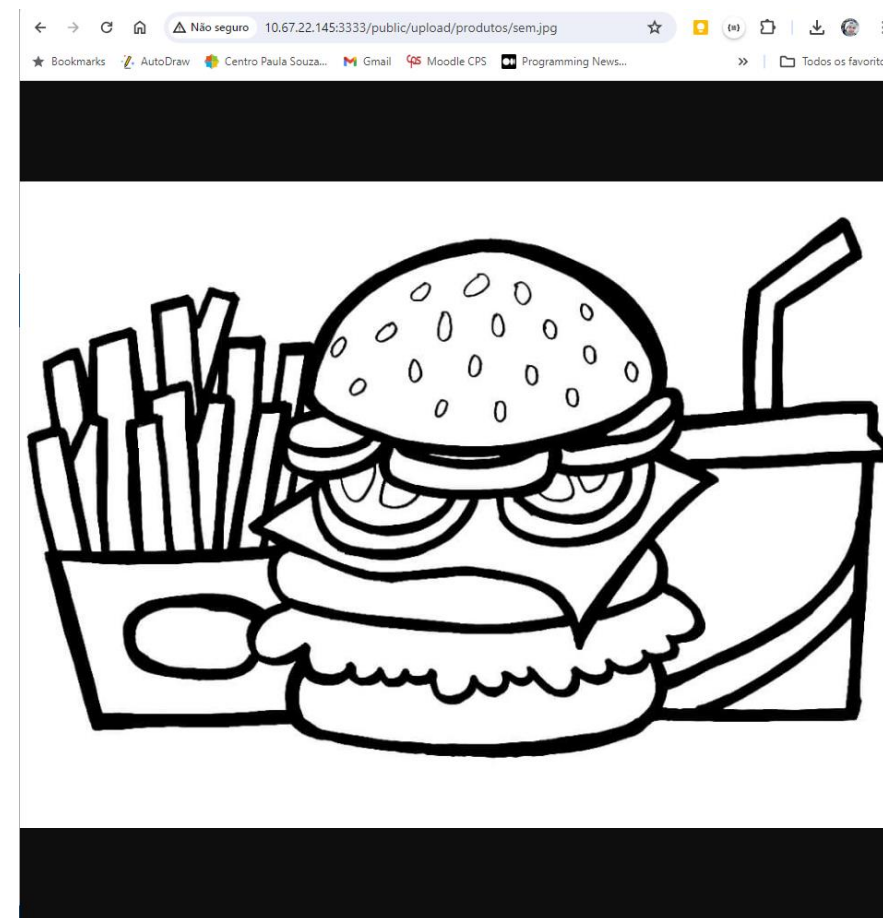
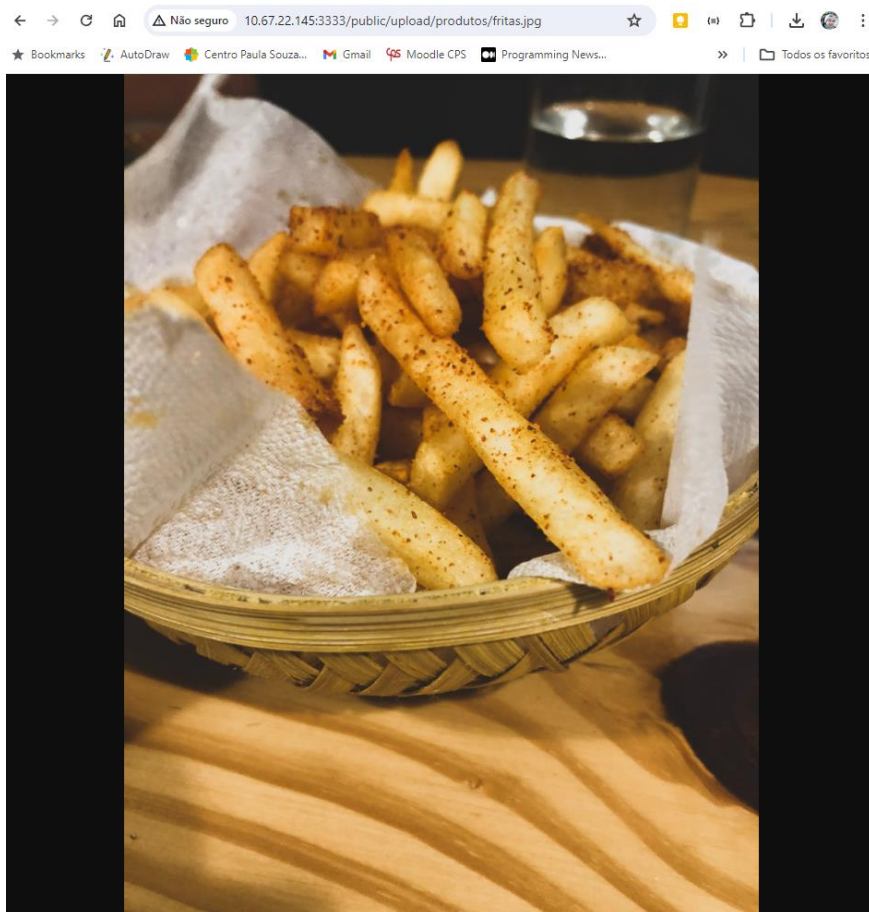
```
1 {
2   "prd_nome" : null,
3   "ptp_id" : null,
4   "prd_valor" : null
5 }
```

Status: 200 OK Size: 650 Bytes Time: 40 ms

Response Headers ⁷ 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      "prd_img": "http://10.67.22.145:3333/public/upload/produtos
11                /fritas.jpg",
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      "prd_img": "http://10.67.22.145:3333/public/upload/produtos/sem
20                .jpg",
21      "prd_descricao": "Abacaxi, açúcar e gelo"
22    },
23    {
24      "prd_id": 6,
25      "prd_nome": "Suco de laranja"
26    }
27  ]
28 }
```


Visualização da url gerada pela API



Contagem de produtos de acordo com o filtro

controllers > JS produtos.js > ...

```
52
53     try {
54
55         // contagem total produtos disponíveis
56         const sqlCount = `SELECT COUNT(*) AS cont_tt_prod FROM produtos prd
57         WHERE prd.prd_disponivel = ? AND prd.prd_nome LIKE ? AND prd.ptp_id LIKE ?
58         AND prd.prd_valor < ?`;
59         const valuesCount = [prd_disponivel, prdPesqNm, prdPesqTp, prdPesqVlr];
60         const ttProdutos = await db.query(sqlCount, valuesCount);
61
62         // Listagem itens
63         const sql = `SELECT
64         prd prd id prd prd nome prd prd valor prd prd unidade prd prd icone
65
66         // chamada para montar url
67         const resultado = produtos[0].map(geraUrl);
68
69         // total de produtos no cabeçalho
70         response.header('X-Total-Count', ttProdutos[0][0].cont_tt_prod);
71
72         return response.status(200).json({
```

Verificando total de itens no cabeçalho

GET ▼ localhost:3333/produtos Send

Query

Headers ²

Auth

Body

Tests

Pre Run

Query Parameters

☐

parameter

value

Status: 200 OK Size: 1.09 KB Time: 77 ms

Response

Headers ⁸

Cookies

Results

Docs

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de produtos.",
4 >  "dados": [ ... ],
46  "nItens": 5
47 }
```

Status: 200 OK Size: 1.09 KB Time: 77 ms

Response

Headers ⁸

Cookies

Results

Docs

Response Headers

Header	Value
x-powered-by	Express
access-control-allow-origin	*
x-total-count	8
content-type	application/json; charset=utf-8

Insert com imagens


```
async cadastrarProdutos(request, response) {
  try {
    const { nome, valor, unidade, tipo, disponivel, descricao } = request.body;
    const destaque = 0;
    const img_destaque = null;
    const img = request.file.filename;

    // instrução sql para inserção
    const sql = `INSERT INTO produtos
      (prd_nome, prd_valor, prd_unidade, prd_tipo, prd_disponivel, prd_img, prd_destaque, prd_img_destaque, prd_descricao)
      VALUES
      (?, ?, ?, ?, ?, ?, ?, ?, ?)`;

    // definição de array com os parâmetros que receberam os valores do front-end
    const values = [nome, parseFloat(valor), unidade, parseInt(tipo), parseInt(disponivel), img, destaque, img_destaque, descricao];

    // executa a instrução de inserção no banco de dados
    const confirmacao = await db.query(sql, values);
    // Exibe o id do registro inserido
    const prd_id = confirmacao[0].insertId;
    // Mensagem de retorno no formato JSON
    const dados = {
      id: prd_id,
      nome,
      valor: parseFloat(valor).toFixed(2),
      unidade,
      tipo,
      disponivel,
      img: 'http://localhost:3333/public/upload/produtos/' + img
    };
  }
}
```

...

```
        return response.status(200).json({
            sucesso: true,
            mensagem: 'Produto cadastrado com sucesso.',
            dados 
        });
    } catch (error) {
        return response.status(500).json({
            sucesso: false,
            mensagem: 'Erro na requisição.',
            dados: error.message
        });
    }
},
```

Inserindo imagem na API

The screenshot displays a REST client interface with a POST request to `localhost:3333/produtos`. The request body is configured as a form with the following fields:

Field	Value
nome	Sorvete de chocolate
valor	12.00
unidade	un.
tipo	4
disponivel	1
descricao	3 bolas na tigela
field name	value

Below the form fields, a file named `img` is attached, with the file path `sorveteChocolate.jpg` visible. A red bracket on the left groups the form fields and the file section.

The response status is `200 OK`, with a size of `255 Bytes` and a time of `8 ms`. The response body is a JSON object:

```
{
  "sucesso": true,
  "mensagem": "Produto cadastrado com sucesso.",
  "dados": {
    "id": 11,
    "nome": "Sorvete de chocolate",
    "valor": "12.00",
    "unidade": "un.",
    "tipo": "4",
    "disponivel": "1",
    "img": "http://localhost:3333/public/upload/produtos/img-1718893513617-114165896.jpeg"
  }
}
```

Red arrows point to the URL, the 'Body' tab, the 'Form' tab, the 'mensagem' field in the response, and the 'img' file path in the response.

Listando produto inserido

```
{  
  "prd_id": 11,  
  "prd_nome": "Sorvete de chocolate",  
  "prd_valor": "12.00",  
  "prd_unidade": "un.",  
  "prd_img": "http://10.67.22.143:3333/public/upload/produtos/img  
              -1718893513617-114165896.jpeg",  
  "prd_descricao": "3 bolas na tigela"  
}
```