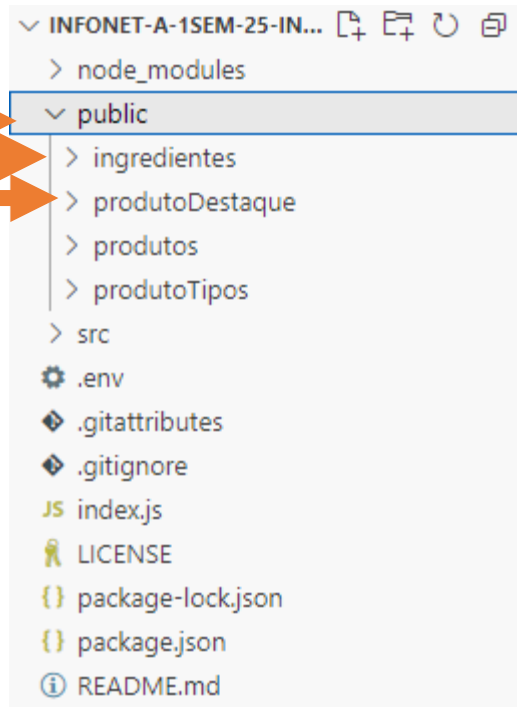


# Trabalhando com imagens

Prof. Me. Ewerton José da Silva

# PREPARANDO PROJETO PARA TRATAR IMAGENS

# Criar pasta publica na raiz do projeto com imagens que possam ser acessadas de forma externa







```
SELECT ptp_id, ptp_nome, ptp_icone FROM produto_tipos;  
SELECT ing_id, ing_nome, ing_img, ing_custo_adicional FROM ingredientes;  
SELECT prd_id, prd_nome, prd_valor, prd_unidade, ptp_id, prd_disponivel, prd_img, prd_destaque, prd_img_destaque, prd_descricao FROM produtos;
```








# Relação entre tabelas e imagens no servidor.






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











ing_id	ing_nome	ing_img	ing_cus
1	Pão	pao.png	0.00
2	Frango	frango.png	7.00
3	Salmão	salmao.png	10.00
4	Alface	alface.png	4.50
5	Rúcula	rucula.png	4.00
6	Tomate	tomate.png	5.25
7	Ervilha	ervilha.png	6.00
8	Milho	milho.png	5.00
9	Pepino	pepino.png	4.50
10	Cebola	cebola.png	4.00
11	Cebola R...	cebolaRox...	4.80
12	Aspargo	aspargo.png	5.90
13	Batata	batata.png	25.40
14	Uva	uva.png	0.00
15	Abacaxi	abacaxi.png	0.00
16	Limão	limao.png	0.00
17	Laranja	laranja.png	0.00
18	Bacon	bacon.png	6.00
19	Couve	couve.png	5.00
20	Carne b...	carneBovin...	8.00
21	Carne su...	carneSuina...	7.00
22	Sal	sal.png	2.00
23	Açúcar	acucar.png	3.00

prd_id	prd_nome	prd_valor	prd_unidade	ptp_id	prd_disponivel	prd_img	prd_destaque	prd_img_destaque	prd_des
1	Lanche de Frango Empanado	31.00	un.	1	1	lancheFrangoEmpanado.jpg	0	NULL	Pão e fra
2	Lanche de Salmão	38.00	un.	1	1	lancheSalmao.jpg	0	NULL	Pão, filé
3	Lanche de Salada	28.00	un.	1	1	lancheVegetariano.jpeg	0	NULL	Pão, alfa
4	Batata frita	25.40	un.	2	1	fritas.jpg	1	promoGenerica.png	Batata de
5	Suco de Abacaxi	12.00	copo	3	1	sucoAbacaxi.png	0	NULL	Abacaxi,
6	Suco de Uva	15.00	copo	3	1	sucoUva.jpeg	0	NULL	Uva, açu
7	Suco de Laranja	6.99	copo	3	1	sucoLaranja.jpg	1	promoSucoLaranja.png	Laranja,
8	Suco de Limão	12.00	copo	3	1	sucoLimao.jpg	0	NULL	Limão, aç
9	Biscoito Amanteigado	12.99	un.	4	0	biscoitoAmanteigado.png	0	NULL	Biscoito s
10	Lanche de Peixe	20.99	un.	1	1	lanchePeixe.jpg	1	promoLanchePeixe.png	Feito com
11	Salada da Casa	16.99	un.	1	1	salada.jpg	1	promoSaladaDaCasa.png	Feito com
12	Sorvete Chocolate	11.00	un.	4	1	sorveteChocolate.jpeg	0	NULL	Refresca
13	Torta de creme de coco	19.00	un.	4	1	tortaCremeCoco.jpg	0	NULL	Torta do
14	Lanche de bacon	34.00	un.	1	1	hamburger-bacon.jpg	0	NULL	Hamburg
15	Combo hamburger + batata	50.00	un.	1	1	lanche.svg	0	NULL	Lanche d

▼ produtoDestaque
 promoGenerica.png
 promoLanchePeixe.png
 promoSaladaDaCasa.png
 promoSucoLaranja.png

▼ produtos
 biscoitoAmanteigado.png
 fritas.jpg
 hamburger-bacon.jpg
 hamburger-batata.jpg
 lancheFrangoEmpanado.jpg
 lanchePeixe.png
 lancheSalmao.png

▼ produtoTipos
 combo.svg
 doce.svg
 lanche.svg
 porcao.svg
 suco.svg

▼ ingredientes
 abacaxi.jpg
 acucar.jpg
 alface.png
 aspargo.png
 bacon.png
 batata.jpg
 carneBovina.jpg
 carneSuina.png
 cebola.jpg
 cebolaRoxa.png

# Instalar a bibliotecas para auxiliar no upload e carregamento das imagens.

- `npm i multer`
- `npm i fs-extra`

# Tornar a pasta public pública.

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



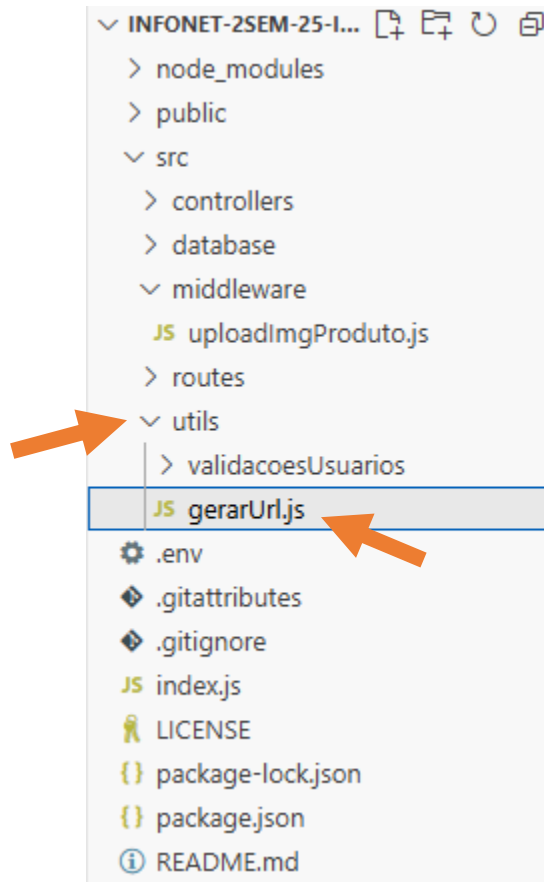
# Definição de URL para compor o caminho da imagem a ser apresentada.

```
🔧 .env
1  #Banco
2  BD_USUARIO=us_aula_node
3  BD_SENHA=123456
4  BD_SERVIDOR=10.67.22.216
5  # BD_SERVIDOR=192.168.15.8
6  BD_PORTA=3306
7  BD_BANCO=bd_aula_node
8
9  # Endereço base da API para o ambiente de desenvolvimento
10 API_BASE_URL=http://localhost:3333
11
12 # Para o acesso externo, inserir o ip da máquina e não localhost Ex:
13 # API_BASE_URL=http://10.67.22.145:3333
14
15 #Email
16 MAIL_HOST=host e-mail
```

# AJUSTES PARA LISTAGEM DE REGISTROS COM IMAGENS

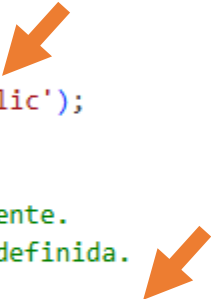


# Criar uma função para automatizar a geração das URLs das imagens



# Código arquivo gerar URL

```
src > utils > JS gerarUrl.js > ...
1  const fse = require('fs-extra');
2  const path = require('path');
3  const { URL } = require('url'); // Módulo nativo do Node.js para trabalhar com URLs
4
5  /**
6   * Caminho físico para a pasta 'public'.
7   */
8  const PUBLIC_ROOT_PATH = path.join(process.cwd(), 'public');
9
10 /**
11  * Lê a URL base da API a partir das variáveis de ambiente.
12  * Fornece um valor padrão caso a variável não esteja definida.
13  */
14 const API_URL = process.env.API_BASE_URL || 'http://localhost:3333';
15
16 /**
17  * Gera uma URL pública e COMPLETA para um recurso (imagem, ícone, etc.).
18  * (...)
19  * @returns {string} A URL completa e formatada (ex: 'http://localhost:3333/upload/produtos/produto-123.jpg').
20  */
21
```

Two orange arrows are present in the image. The first arrow points from the right towards the string 'public' in line 8, which is part of the path.join function. The second arrow points from the right towards the default URL value 'http://localhost:3333' in line 14, which is part of a logical OR assignment.

```
22 function gerarUrl(nomeArquivo, pasta, arquivoPadrao) {
23     const arquivoVerificar = nomeArquivo || arquivoPadrao;
24     const caminhoFisico = path.join(PUBLIC_ROOT_PATH, pasta, arquivoVerificar);
25
26     let caminhoRelativo;
27
28     // Se um nome de arquivo foi fornecido e ele existe...
29     if (nomeArquivo && fse.existsSync(caminhoFisico)) {
30         // ...usa o caminho para esse arquivo.
31         caminhoRelativo = path.join('/public', pasta, nomeArquivo);
32     } else {
33         // ...caso contrário, usa o caminho do arquivo padrão.
34         caminhoRelativo = path.join('/public', pasta, arquivoPadrao);
35     }
36
37     // Garante que o caminho relativo use barras '/'
38     const caminhoRelativoFormatado = caminhoRelativo.replace(/\\/g, '/');
39
40     // Constrói a URL completa de forma segura, evitando barras duplas (//)
41     const urlCompleta = new URL(caminhoRelativoFormatado, API_URL);
42
43     return urlCompleta.href;
44 }
45
46 // Exporte a nova função
47 module.exports = { gerarUrl };
```

Alterando controller relacionado a tabela que tem imagem a ser apresentada.  
Primeiro importa a função criada

```
src > controllers > JS ingredientes.js > ...
```

```
1  const db = require('../database/connection');  
2  const { gerarUrl } = require('../utils/gerarUrl');  
3  
4  module.exports = {  
5    async listarIngredientes(request, response) {
```



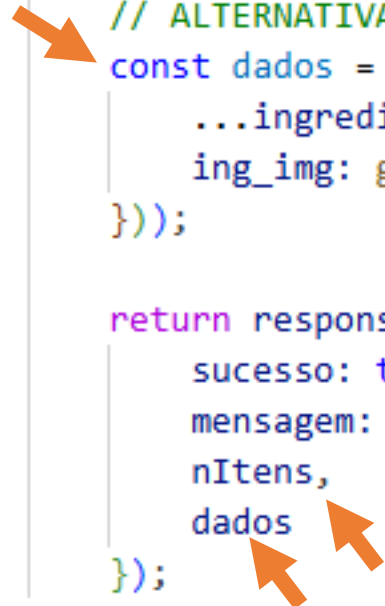
Se existe o ajuste nos nomes dos campos antes da saída dos dados, faça as alterações apontadas abaixo apenas se estiver utilizando o “map” para saída dos dados, se não tiver isso...

```
21      const [rows] = await db.query(sql, values);
22      const nItens = rows.length;
23
24      const dados = rows.map(ingrediente => ({
25          id: ingrediente.ing_id,
26          nome: ingrediente.ing_nome,
27          img: gerarUrl(ingrediente.ing_img, 'ingredientes', 'sem.svg'),
28          custo_adicional: ingrediente.ing_custo_adicional
29      }));
30
31      return response.status(200).json({
32          sucesso: true,
33          mensagem: 'Lista de ingredientes.',
34          nItens,
35          dados
36      });
37  } catch (error) {
```



... Adicione a função das linhas 25 a 28 e altere os dados conforme as linhas 33 e 34.

```
21      const [rows] = await db.query(sql, values);
22      const nItens = rows.length;
23
24      // ALTERNATIVA SEM MEXER COM TODOS OS CAMPOS
25      const dados = rows.map(ingrediente => ({
26          ...ingrediente,
27          ing_img: gerarUrl(ingrediente.ing_img, 'ingredientes', 'sem.jpg')
28      }));
29
30      return response.status(200).json({
31          sucesso: true,
32          mensagem: 'Lista de ingredientes.',
33          nItens,
34          dados
35      });
36      } catch (error) {
37          return response.status(500).json({
```



# Teste de listagem

GET <http://localhost:3333/ingredientes> Send 200 OK 36 ms 2.9 KB

Params Body Auth Headers 3 Scripts Docs

URL PREVIEW

<http://localhost:3333/ingredientes>

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'

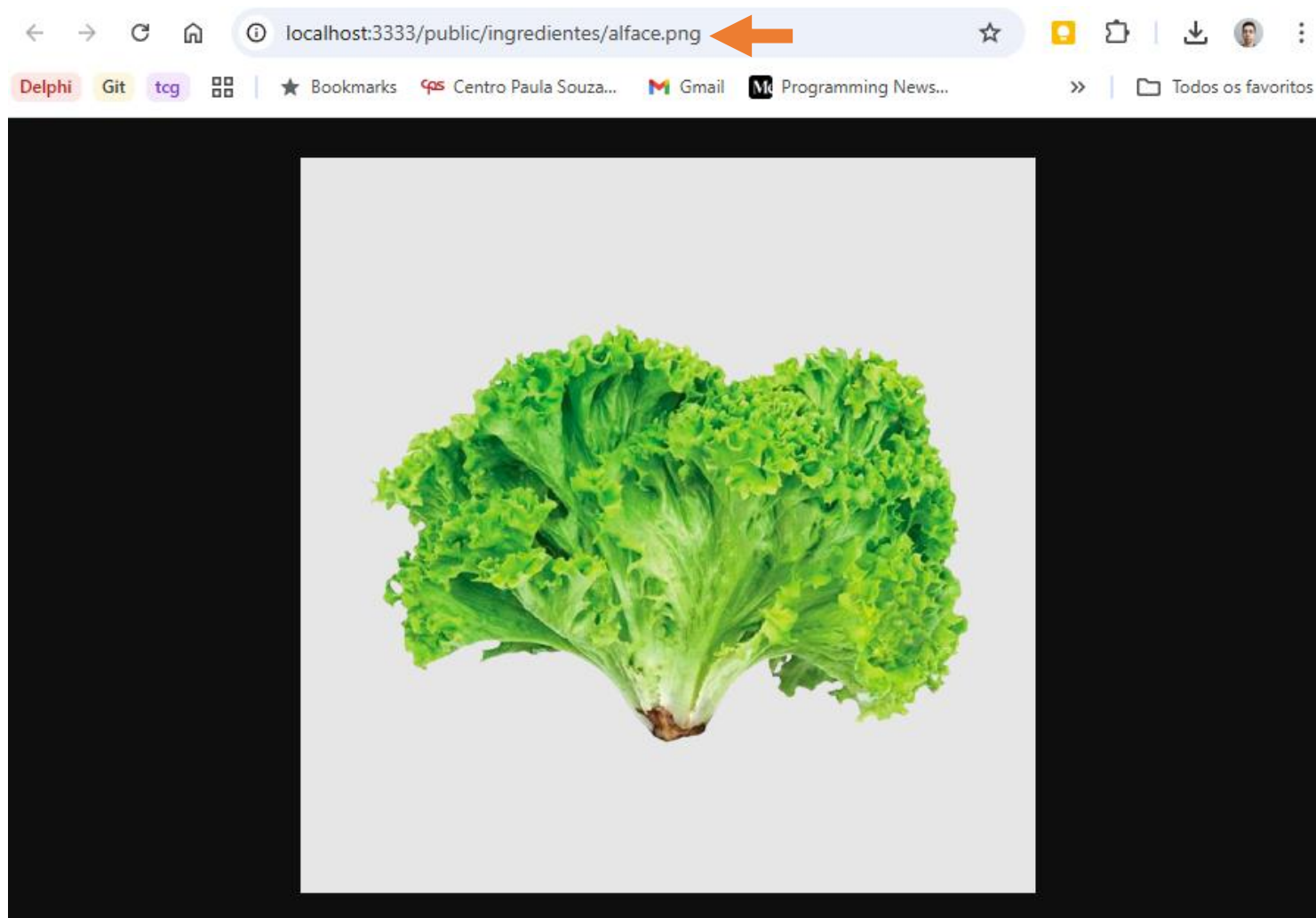
Preview Headers 8 Cookies Tests 0 / 0 → Mock Console

Preview

```
1 {
2   "sucesso": true,
3   "mensagem": "Lista de ingredientes.",
4   "nItens": 26,
5   "dados": [
6     {
7       "id": 1,
8       "nome": "Pão",
9       "img": "http://localhost:3333/public/ingredientes/pao.jpeg",
10      "custo_adicional": "0.00"
11    },
12    {
13      "id": 2,
14      "nome": "Frango",
15      "img": "http://localhost:3333/public/ingredientes/frango.jpg",
16      "custo_adicional": "7.00"
17    },
18    {
19      "id": 3,
20      "nome": "Salmão",
21      "img": "http://localhost:3333/public/ingredientes/salmao.png",
22      "custo_adicional": "10.00"
23    },
24    {
25      "id": 4,
26      "nome": "Alface",
27      "img": "http://localhost:3333/public/ingredientes/alface.png",
28      "custo_adicional": "4.50"
29    },
30    {
31      "id": 5,
32      "nome": "Búfala"
```



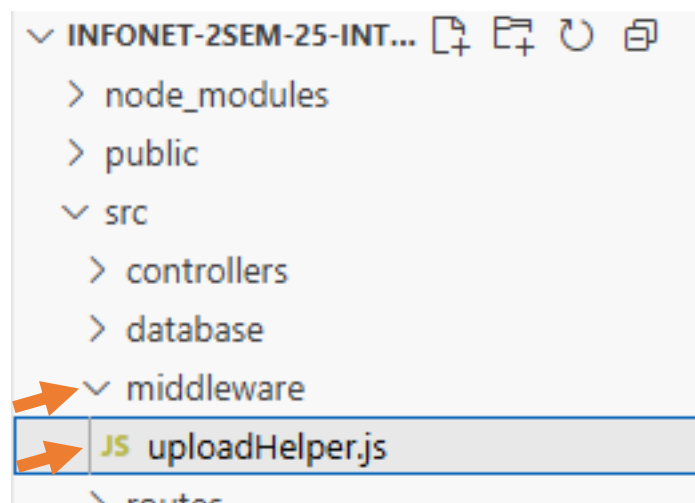
# Visualização da url gerada pela API





UPLOAD

# Upload de imagens



src > middleware > JS uploadHelper.js > ...

```
1  const multer = require('multer');
2  const fs = require('fs');
3
4  // Função que cria a configuração do Multer dinamicamente
5  const uploadImage = (destinationFolder) => {
6    // Validação para garantir que a pasta de destino foi fornecida
7    if (!destinationFolder) {
8      throw new Error("O nome da pasta de destino é obrigatório.");
9    }
10
11    const fullPath = `./public/${destinationFolder}/`;
12
13    // Garante que o diretório de destino exista, se não, cria.
14    if (!fs.existsSync(fullPath)) {
15      fs.mkdirSync(fullPath, { recursive: true });
16    }
17
18    // Configuração do Storage (onde e como salvar)
19    const storage = multer.diskStorage({
20      destination: function (req, file, cb) {
21        cb(null, fullPath);
22      },
23      filename: function (req, file, cb) {
24        const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
25        // Extraí a extensão do arquivo a partir do mimetype
26        const extension = file.mimetype.split('/')[1];
27        cb(null, `${uniqueSuffix}.${extension}`);
28      }
29    });
```

```
31 // Filtro para aceitar apenas certos tipos de imagem
32 const fileFilter = (req, file, cb) => {
33   if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/jpg' || file.mimetype === 'image/png' || file.mimetype === 'image/gif') {
34     cb(null, true);
35   } else {
36     // Rejeita o arquivo com uma mensagem de erro
37     cb(new Error('Formato de imagem não suportado! Use JPEG, JPG, PNG ou GIF.'), false);
38   }
39 };
40
41 // Retorna a instância do Multer configurada
42 return multer({
43   storage: storage,
44   limits: {
45     fileSize: 1024 * 1024 * 5 // 5MB
46   },
47   fileFilter: fileFilter
48 });
49 }
50
51 module.exports = uploadImage;
```

# Ajuste na rota para chamada do upload

src > routes > JS produtos.js > ...

```
1  const express = require('express');
2  const router = express.Router();
3
4  const ProdutosController = require('../controllers/produtos');
5  const IngredientesController = require('../controllers/ingredientes');
6  const ProdutoIngredientesController = require('../controllers/produtoIngredientes');
7
8  const uploadImage = require('../middleware/uploadHelper'); ←
9
10 // Middleware configurado para a pasta 'ingredientes'
11 const uploadIngredientes = uploadImage('ingredientes'); ←
12
```

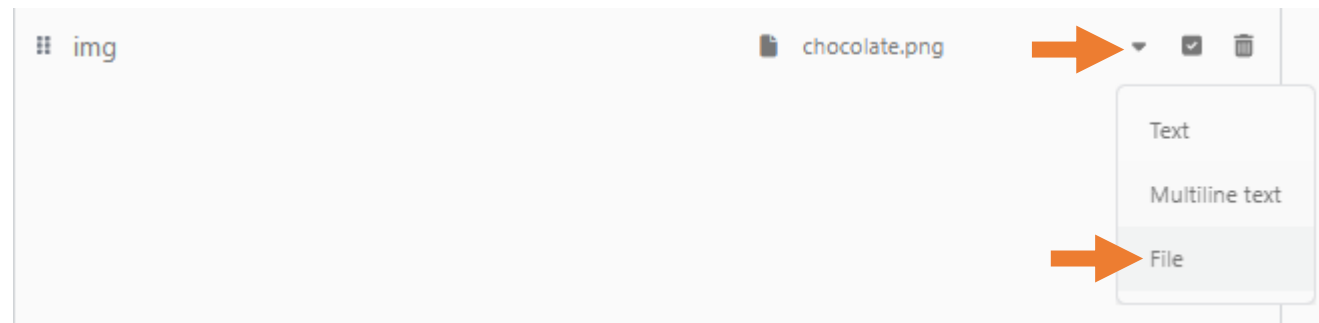
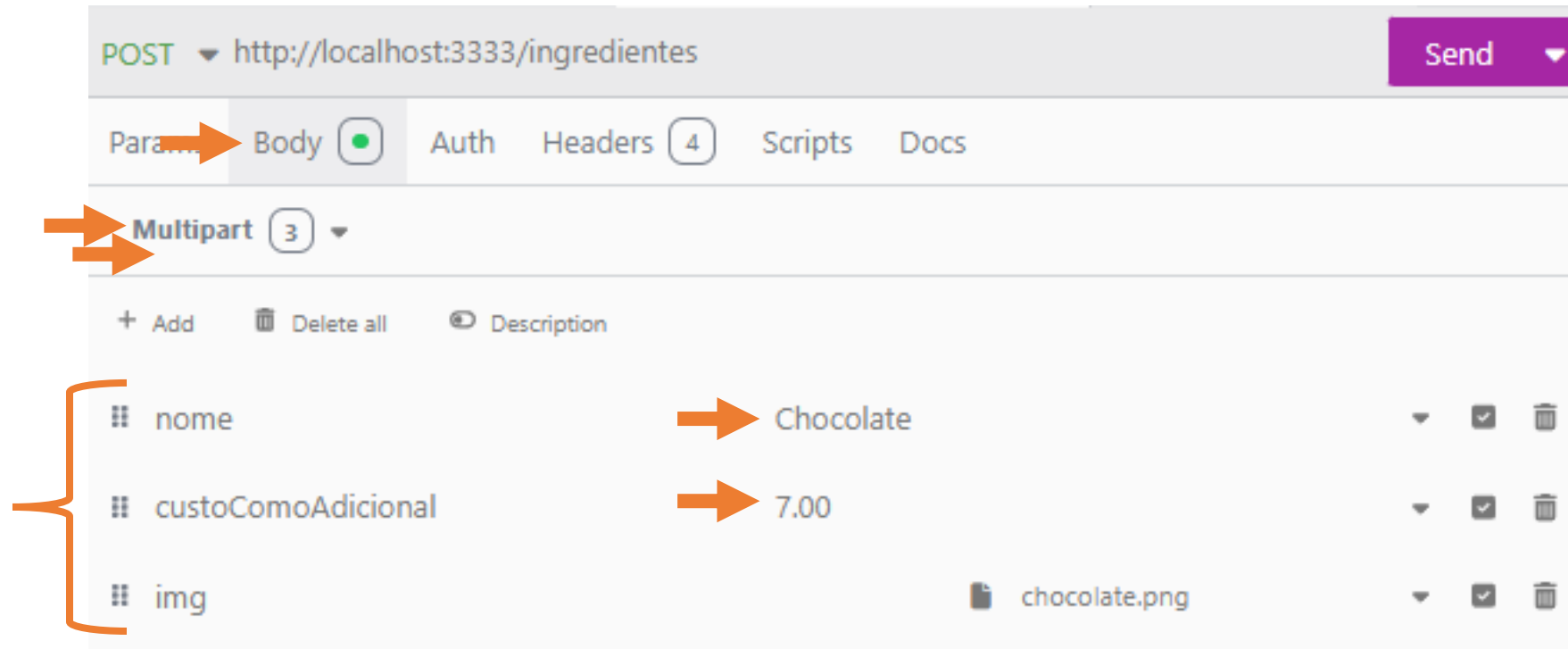
```
20 router.get('/ingredientes', IngredientesController.listarIngredientes);
21 → router.post('/ingredientes', uploadIngredientes.single('img'), IngredientesController.cadastrarIngredientes);
22 router.patch('/ingredientes', IngredientesController.editarIngredientes);
23 router.delete('/ingredientes', IngredientesController.apagarIngredientes);
```

# Editar método de cadastro no controller

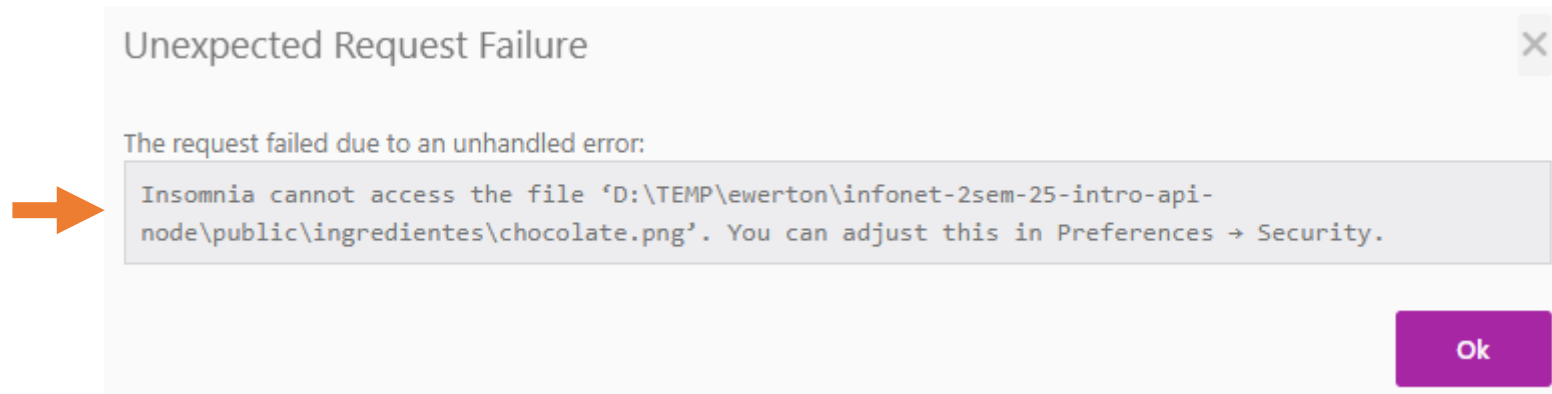
```
51  async cadastrarIngredientes(request, response) {
52      try {
53          const { nome, custoComoAdicional } = request.body;
54          const imagem = request.file;
55
56          const sql = `
57              INSERT INTO ingredientes
58              (ing_nome, ing_img, ing_custo_adicional)
59              VALUES (?, ?, ?);
60          `;
61
62          const values = [nome, imagem.filename, custoComoAdicional];
63
64          const [result] = await db.query(sql, values);
65
66          return response.status(201).json({
67              sucesso: true,
68              mensagem: 'Ingrediente adicionado com sucesso.',
69              dados: { id: result.insertId }
70          });
71      } catch (error) {
72          return response.status(500).json({
73              sucesso: false,
74              mensagem: 'Erro na requisição.',
75              dados: error.message
76          });
77      }
78  },
```



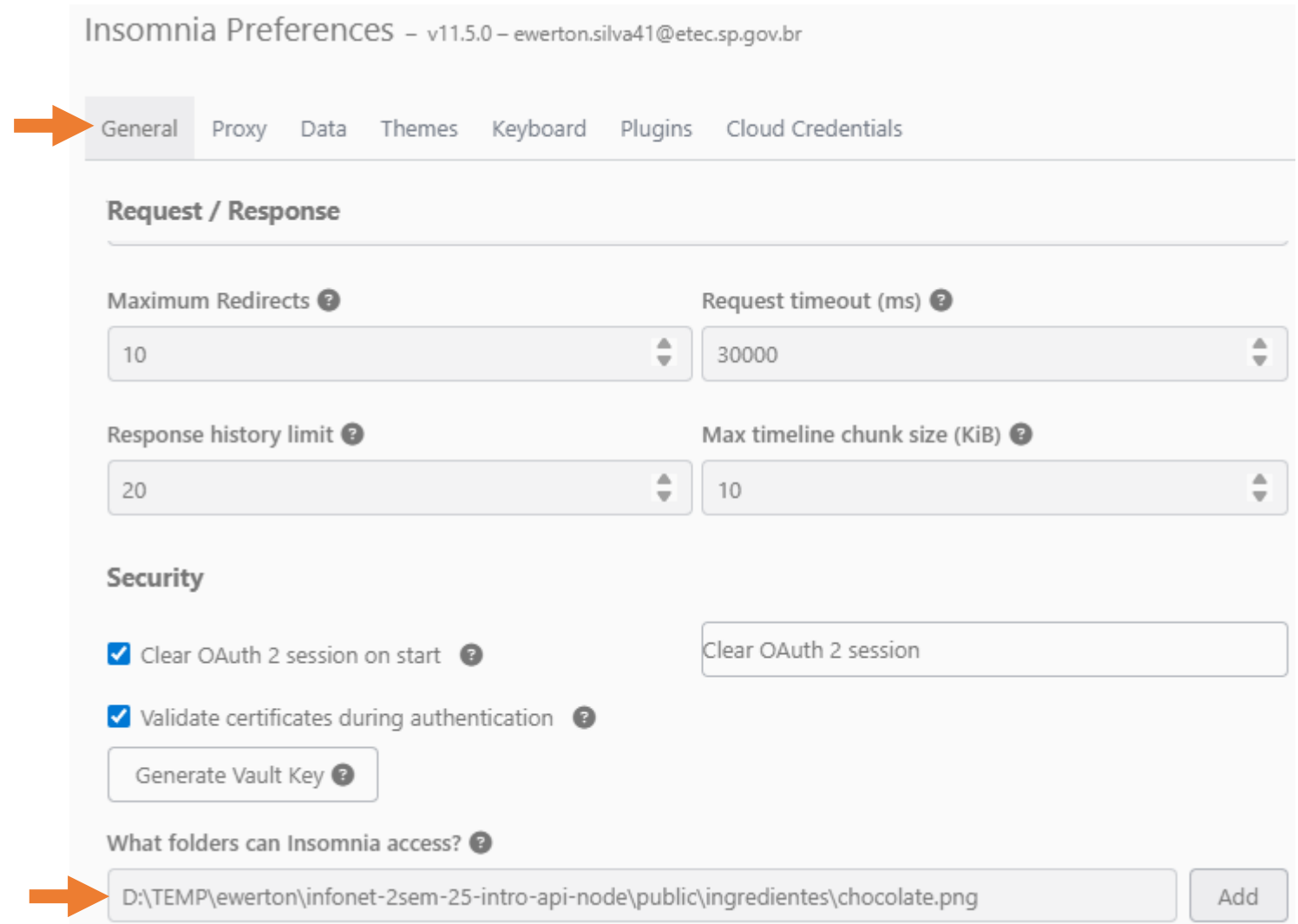
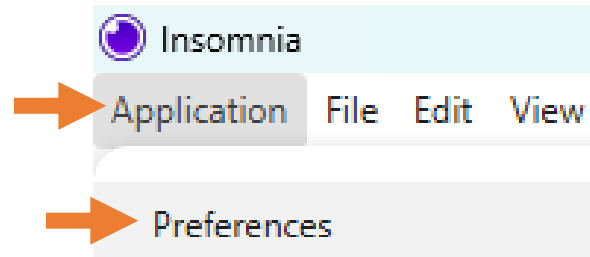
# Teste inserção imagem API



# Caso apareça alguma mensagem de erro, siga as instruções:

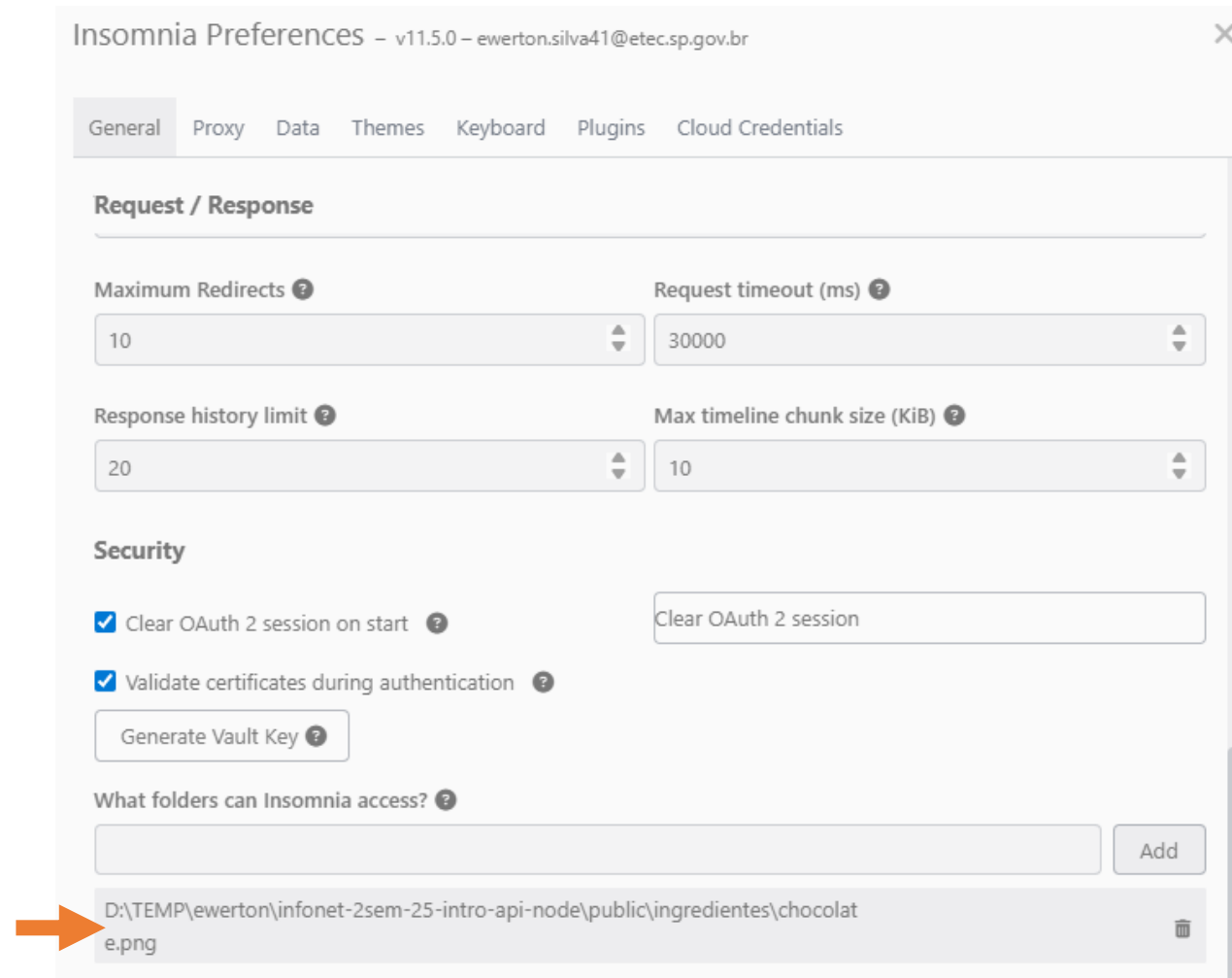


# Cole o link gerado anteriormente:





# Após adicionar a tela deve apontar o caminho adicionado




# Inserção de ingrediente com imagem

POST http://localhost:3333/ingredientes Send 201 Created 262 ms 83 B

Params Body Auth Headers 4 Scripts Docs Preview Headers 8 Cookies Tests 0 / 0 → Mock Console

Multipart 3

+ Add Delete all Description

nome	Chocolate	<span>▼</span> <span>✓</span> <span>🗑️</span>
custoComoAdicional	7.00	<span>▼</span> <span>✓</span> <span>🗑️</span>
img	 chocolate.png	<span>▼</span> <span>✓</span> <span>🗑️</span>

Preview

```
1 {  
2   "sucesso": true,  
3   "mensagem": "Ingrediente adicionado com sucesso.",  
4   "dados": {  
5     "id": 31  
6   }  
7 }
```

# Listagem do item adicionado

```
{  
  "id": 31,  
  "nome": "Chocolate",  
  "img": "http://localhost:3333/public/ingredientes/1757458921523-830637971.png",  
  "custo_adicional": "7.00"  
}
```

