Importar e Exportar Arquivos CSV, JSON e XML

Prof. Me. Hélio Esperidião

CSV - Character-separated values

Os arquivos CSV (do inglês "Character-separated values" ou "valores separados por um delimitador") servem para armazenar dados tabulares (números e texto) em texto simples.

O "texto simples" significa que o arquivo é uma sequência de caracteres puros, sem qualquer informação escondida que o computador tenha que processar.

Registros

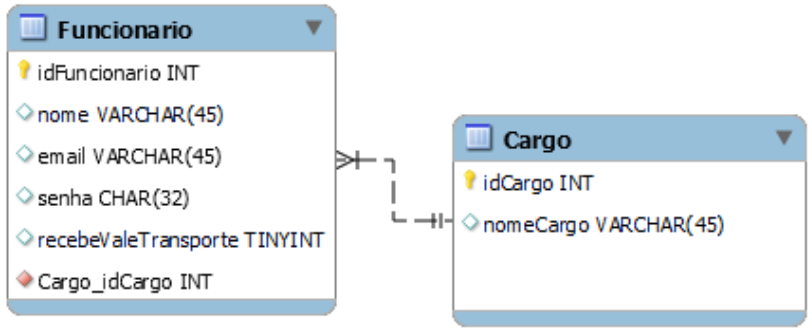
- Um arquivo CSV abriga "registros", separados por quebras de linha (cada "registro" permanece numa linha do arquivo) e cada registro possui um ou mais "campos", separados por um delimitador, os mais comuns sendo a vírgula (","), o ponto e vírgula (";") e o caractere "invisível" que surge ao se pressionar a tecla "tab(\t)".

Exemplo: Estados Brasileiros

- IBGE;Estado;UF;Região;Qtd Mun;Sintaxe
- 11;Rondônia;RO;Região Norte;52;11'RO'
- 12;Acre;AC;Região Norte;22;12'AC'
- 13;Amazonas;AM;Região Norte;62;13'AM'

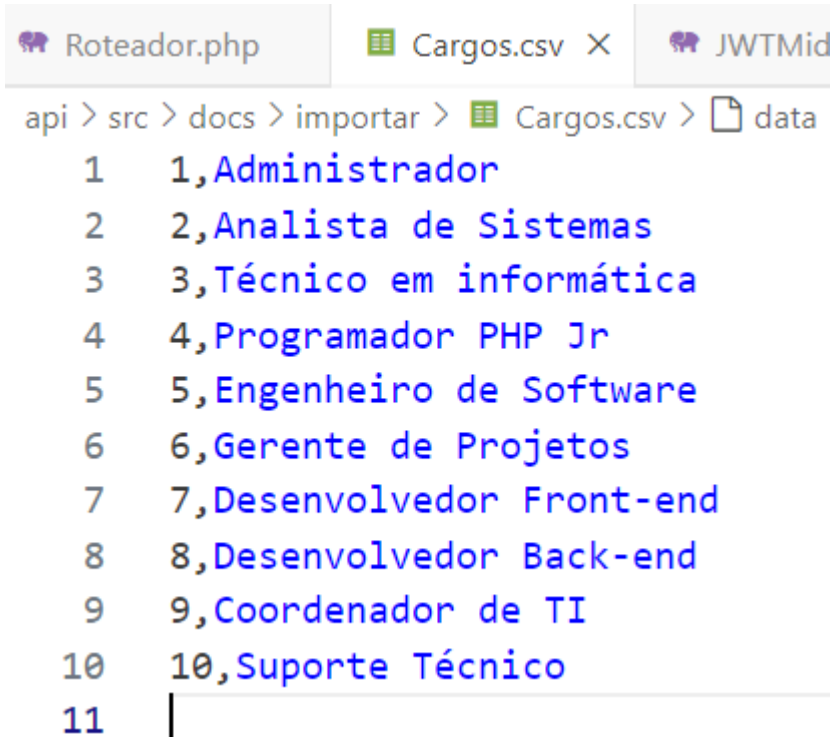
A	B	C	D	E	F
IBGE	Estado	UF	Região	Qtd Mun	Sintaxe
11	Rondônia	RO	Região Norte	52	11'RO'
12	Acre	AC	Região Norte	22	12'AC'
13	Amazonas	AM	Região Norte	62	13'AM'
14	Roraima	RR	Região Norte	15	14'RR'
15	Pará	PA	Região Norte	144	15'PA'
16	Amapá	AP	Região Norte	16	16'AP'
17	Tocantins	TO	Região Norte	139	17'TO'
21	Maranhão	MA	Região Nordeste	217	21'MA'

Exemplo completo CSV



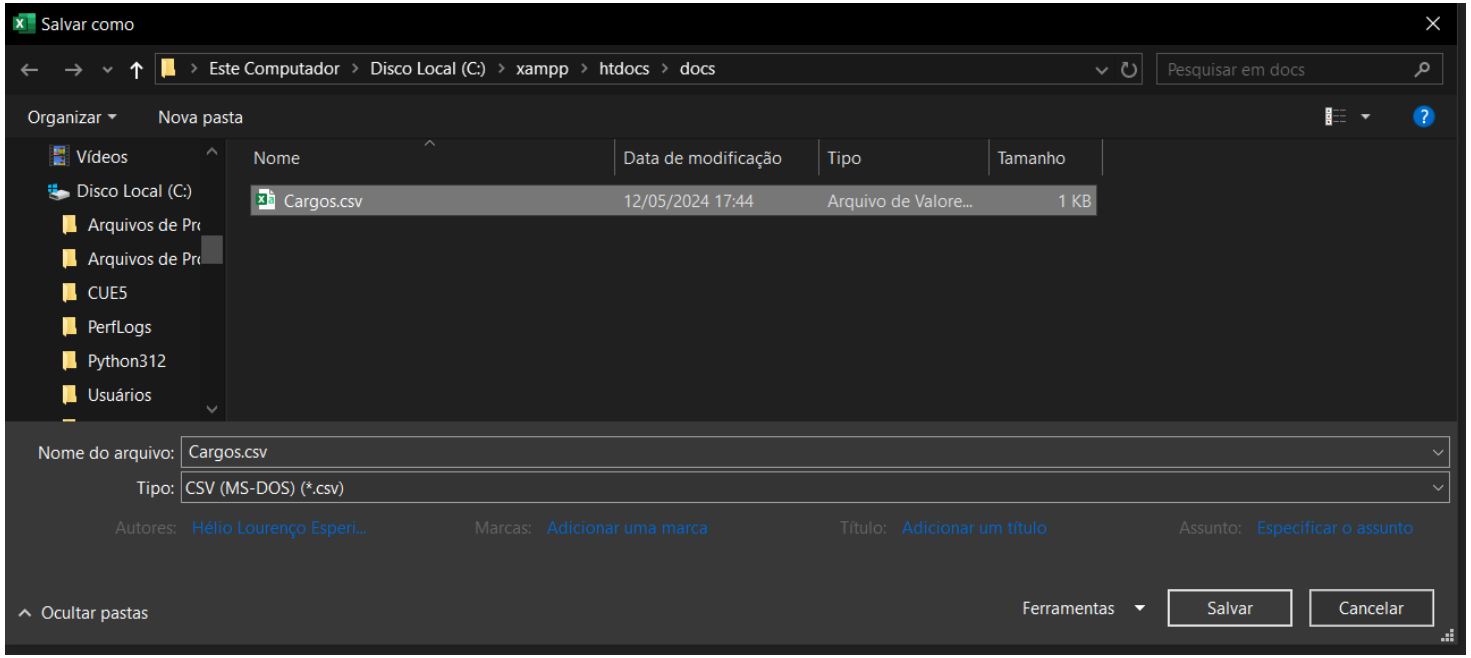
- Os dados serão gravados nas tabelas
- A rota /cargos/csv
  - Recebe um arquivo csv contendo apenas os cargos.

Cargos.csv

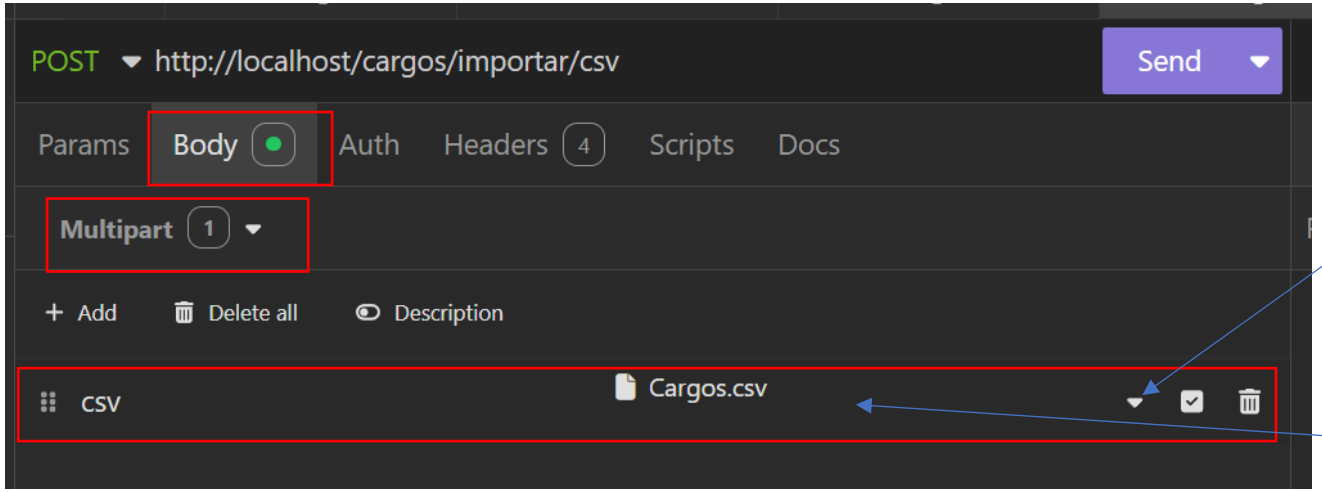

Roteador.php
Cargos.csv
JWTMid
api > src > docs > importar > Cargos.csv > data
1,Administrador
2,Analista de Sistemas
3,Técnico em informática
4,Programador PHP Jr
5,Engenheiro de Software
6,Gerente de Projetos
7,Desenvolvedor Front-end
8,Desenvolvedor Back-end
9,Coordenador de TI
10,Suporte Técnico

Como gerar?

- Qualquer arquivo Excel pode dar origem a um csv
  - Salve uma tabela simples de linhas e colunas apenas.
  - Salve Como o tipo CSV.

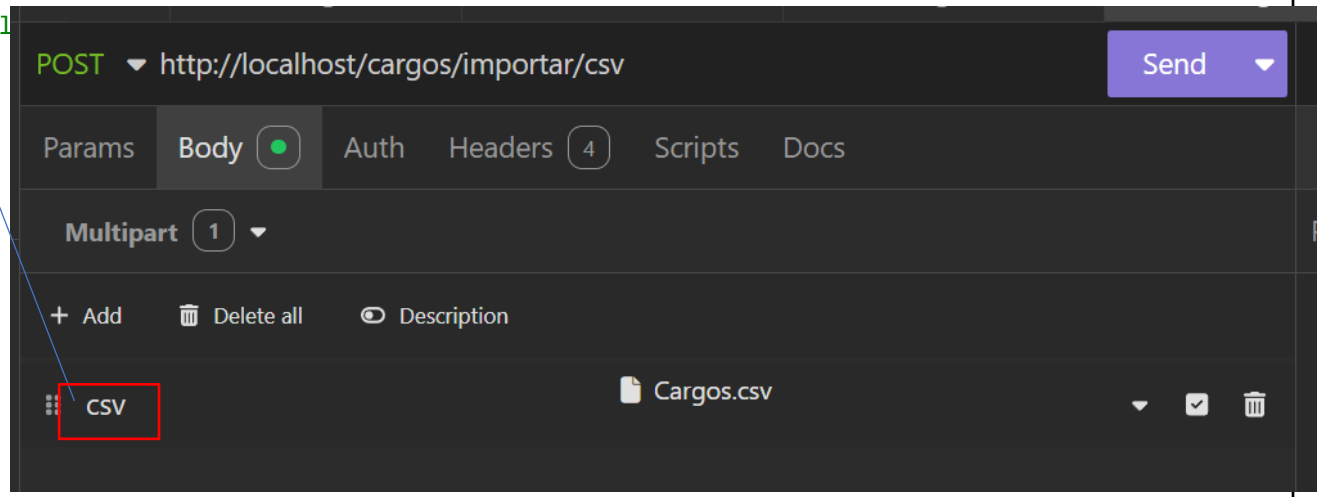


Como enviar um arquivo no insomnia:
Crie uma nova rota post



Crie uma rota para receber o arquivo

```php
$this->router->post(pattern: "/cargos/importar/csv", fn: function (): never {
    try {
        $controle = new CargoControl();
        $controle->importCSV(csvFile: $_FILES['csv']); // Processa e importa os dados do arquivo CSV
        exit();
    } catch (Throwable $throwable) {
                // Caso ocorra um erro durante o processo de exportação (ex: falha ao gerar o JSON),
                // a exceção é capturada e uma resposta de erro é enviada ao cliente.
    $this->sendErrorResponse(
           throwable: $throwable, // Passa a exceção gerada para o tratamento de erro
            message: 'Erro na exportação JSON' // Mensagem expl
     );
          exit();
     }
 });
```



importCSV() - 1/7

```php
<?php
public function importCSV(array $csvFile): never     {
        // Obtém o caminho temporário do arquivo CSV enviado.
        $nomeTemporario = $csvFile['tmp_name'];
        // Verifica se o arquivo foi realmente enviado pelo formulário (evitando qualquer tipo de manipulação externa).
        if (!is_uploaded_file($nomeTemporario)) {
            // Se o arquivo não for válido, envia uma resposta com status 400 (erro de requisição).
            (new Response(
                success: false,
                message: 'Arquivo inválido.', // Mensagem indicando que o arquivo não foi enviado corretamente.
                httpCode: 400
            ))->send();
            exit(); // Finaliza a execução caso o arquivo seja inválido.
        }
•
```

importCSV() - 2/7

```
// Tenta abrir o arquivo CSV no modo de leitura.
        $ponteiroArquivo = fopen($nomeTemporario, "r");
        // Se o arquivo não puder ser aberto, envia uma resposta com status 500 (erro interno).
        if ($ponteiroArquivo === false) {
            (new Response(
                success: false,
                message: 'Não foi possível abrir o arquivo.',
                httpCode: 500
            ))->send();
            exit(); // Finaliza a execução caso o arquivo não possa ser aberto.
        }
```

importCSV() - 3/7

```
// Tenta abrir o arquivo CSV no modo de leitura.
        $ponteiroArquivo = fopen($nomeTemporario, "r");
        // Se o arquivo não puder ser aberto, envia uma resposta com status 500 (erro interno).
        if ($ponteiroArquivo === false) {
            (new Response(
                success: false,
                message: 'Não foi possível abrir o arquivo.',
                httpCode: 500
            ))->send();
            exit(); // Finaliza a execução caso o arquivo não possa ser aberto.
        }
```

importCSV() – 4/7

```
        // Instancia a classe CargoDAO para manipular os cargos no banco de dados.
        $cargoDAO = new CargoDAO();
        // Arrays para armazenar cargos criados e não criados.
        $cargosCriados = [];
        $cargosNaoCriados = [];
```

importCSV() - 5/7

```php
// Lê o arquivo CSV linha por linha.
        while (($linhaArquivo = fgetcsv($ponteiroArquivo, 1000, ",")) !== false) {
            // A função fgetcsv lê uma linha do arquivo e a converte em um array. Cada elemento do array é uma célula do CSV.
            // Agora, é necessário garantir que a codificação dos caracteres esteja correta (UTF-8).
            foreach ($linhaArquivo as &$campo) {
                // Verifica se a codificação do campo não é UTF-8 e converte se necessário.
                if (!mb_detect_encoding($campo, 'UTF-8', true)) {
                    // Converte o campo de ISO-8859-1 para UTF-8 se ele não for UTF-8.
                    $campo = mb_convert_encoding($campo, 'UTF-8', 'ISO-8859-1');
                }
            }
            // Garante que a linha tem pelo menos 2 colunas antes de tentar processá-la.
            if (count($linhaArquivo) < 2) {
                continue; // Pula linhas que não têm os dados necessários.
            }
```

importCSV() – 6/7

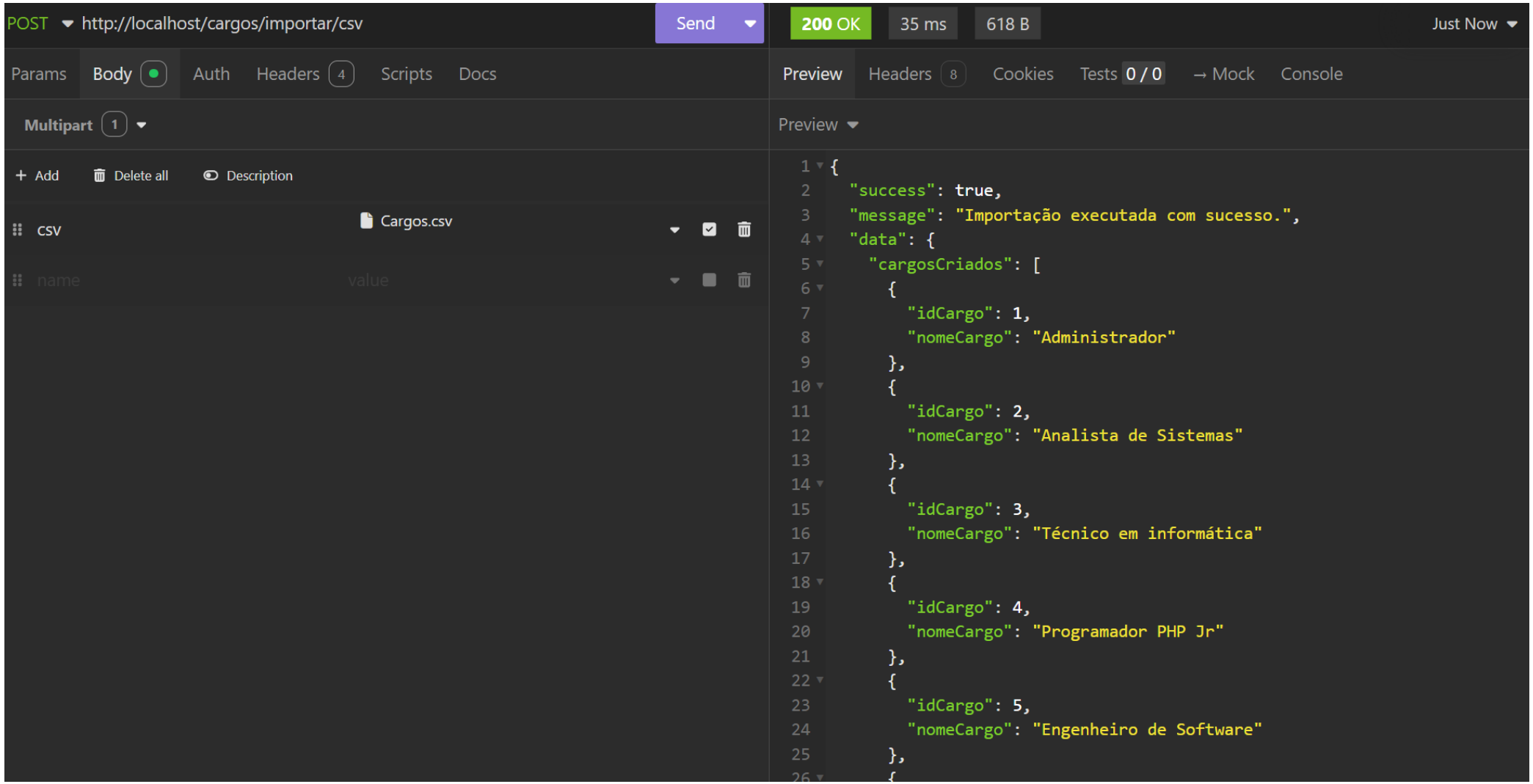
```php
            // Cria o objeto Cargo com os dados lidos do CSV.
            $cargo = new Cargo();
            // Configura o ID e o nome do cargo.
            $cargo->setIdCargo($linhaArquivo[0])
                ->setNomeCargo($linhaArquivo[1]);

                // Tenta inserir o cargo no banco de dados.
            $cargoCriado = $cargoDAO->create($cargo);

                // Verifica se o cargo foi inserido com sucesso.
            if ($cargoCriado == false) {
                // Se falhou, adiciona o cargo à lista de cargos não criados.
                $cargosNaoCriados[] = $cargo;
            } else {
                // Se deu certo, adiciona o cargo à lista de cargos criados.
                $cargosCriados[] = $cargo;
            }
        }//fecha o while
```

CargoDAO.PHP

```php
    public function create(Cargo $cargo): Cargo
     {
         $idCargo = $cargo->getIdCargo();
         if (isset($idCargo)) {
             return $this->createWithId(cargo: $cargo);
         } else {
             return $this->createWithoutId(cargo: $cargo);
         }
     }
```

importCSV() – 7/7

```
        // Fecha o arquivo após processar todas as linhas.
        fclose($ponteiroArquivo);
        // Envia a resposta final com os cargos criados e não criados.
        (new Response(
            success: true,
            message: 'Importação executada com sucesso.', // Mensagem indicando que a importação foi concluída.
            data: [
                "cargosCriados" => $cargosCriados, // Lista de cargos que foram inseridos com sucesso.
                "cargosNaoCriados" => $cargosNaoCriados, // Lista de cargos que falharam ao ser inseridos.
            ],
            httpCode: 200 // Código de sucesso (OK).
        ))->send();

           exit(); // Finaliza a execução após enviar a resposta.
    } //fecha o método
```


POST
http://localhost/cargos/importar/csv
Send
200 OK
35 ms
618 B
Just Now
Params
Body
Auth
Headers 4
Scripts
Docs
Preview
Headers 8
Cookies
Tests 0 / 0
→ Mock
Console
Multipart 1
Preview
+ Add
Delete all
Description
csv
Cargos.csv
name
value
{
"success": true,
"message": "Importação executada com sucesso.",
"data": {
"cargosCriados": [
{
"idCargo": 1,
"nomeCargo": "Administrador"
},
{
"idCargo": 2,
"nomeCargo": "Analista de Sistemas"
},
{
"idCargo": 3,
"nomeCargo": "Técnico em informática"
},
{
"idCargo": 4,
"nomeCargo": "Programador PHP Jr"
},
{
"idCargo": 5,
"nomeCargo": "Engenheiro de Software"
},

formularioUploadCSV.php
Formulário HTML

```
<html>
<head>
   <title>Upload de Arquivos com PHP</title>
</head>
<body>

<form method="post" action="/cargos/importar/csv" enctype="multipart/form-data">
<label>Selecione o arquivo CSV</label>
<input type="file" name="csv" /><br>
<input type="submit" value="Enviar" />
</form>

</body>
</html>
```

```
$controle->importCSV(csvFile: $_FILES['csv']);
```

que é um arquivo JSON?

- JSON (JavaScript Object Notation) é um formato leve de troca de dados.
  - Ele é: Baseado em texto (UTF-8)
  - Fácil de ler e escreverFacilmente processado por linguagens de programação (JavaScript, Python, C#, etc.).
- Cuidados
  - Estrutura rígida: pequenos erros quebram o JSON

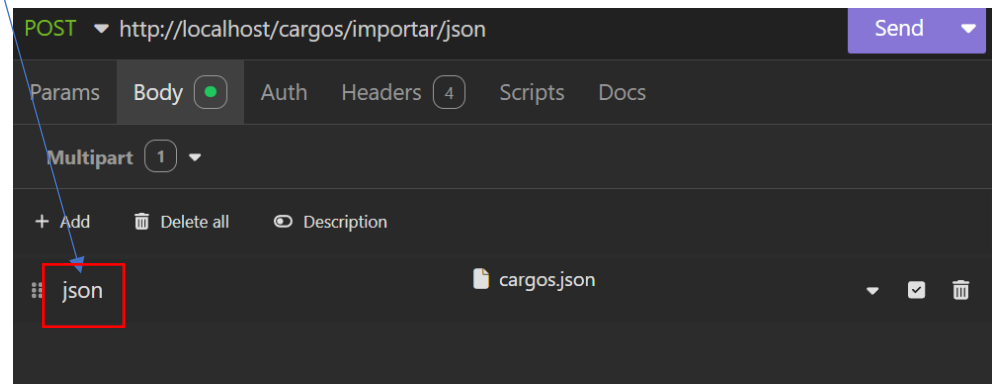
```
{
  "nome": "Maria",
  "idade": 28,
  "ativo": true,
  "skills": ["JavaScript", "Python", "Unity"],
  "endereco": {
    "cidade": "São Paulo",
    "estado": "SP"
  }
}
```

Arquivos .json


```
{
    "cargos": [
        {
            "idCargo": 1,
            "nomeCargo": "Administrador :d)"
        },
        {
            "idCargo": 4,
            "nomeCargo": "Analista de Sistemas Jr"
        },
        {
            "idCargo": 7,
            "nomeCargo": "Engenheiro Jr"
        },
        {
            "idCargo": 2,
            "nomeCargo": "T\u00e9cnico em Inform\u00e1tica Jr"
        },
        {
            "idCargo": 3,
            "nomeCargo": "T\u00e9cnico em Inform\u00e1tica Pleno"
        }
    ]
}
```

/cargos/importar/json

```php
$this->router->post(pattern: "/cargos/importar/json", fn: function (): never {
    try {
         $controle = new CargoControl();
          $controle->importJson(jsonFile: $_FILES['json']); // Processa e importa os dados do arquivo JSON

     catch (Throwable $throwable) {
        $this->sendErrorResponse(
            throwable: $throwable, // Passa a exceção gerada para o tratamento de erro
            message: 'Erro na exportação JSON' // Mensagem explicativa do erro ocorrido
        );

        exit();
    }
});
```

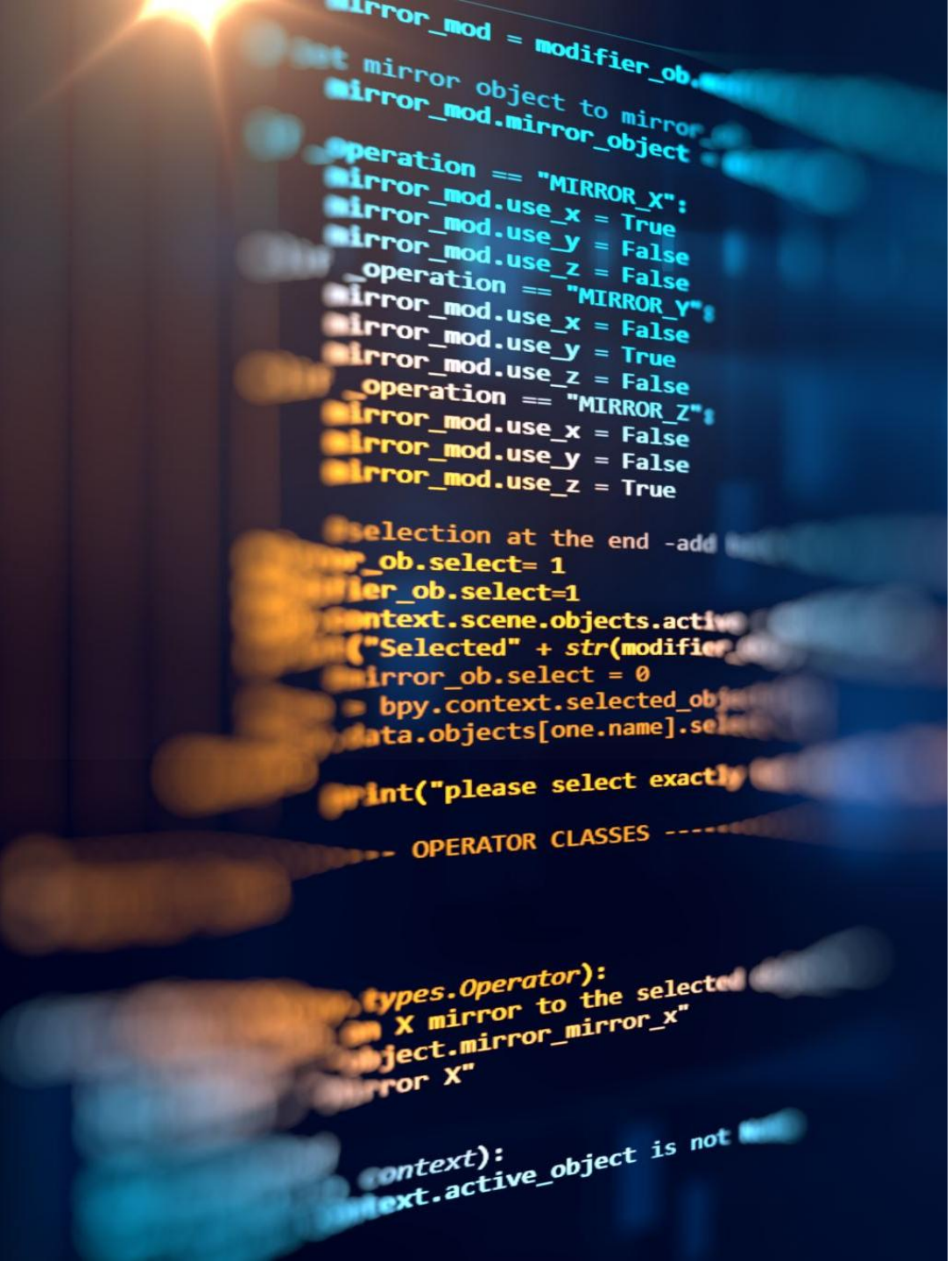


```php
public function importJson(array $jsonFile): never    {
    // Obtém o nome temporário do arquivo JSON enviado pelo formulário HTML
    $nomeTemporario = $jsonFile['tmp_name'];
    // Lê o conteúdo do arquivo JSON
    $conteudoArquivo = file_get_contents($nomeTemporario);
    // Decodifica o JSON para um objeto (não um array associativo)
    $dadosJson = json_decode($conteudoArquivo);
    // Verifica se houve erro ao decodificar o JSON
    if ($dadosJson === null) {
        (new Response(
            success: false,
            message: 'Erro ao decodificar o arquivo JSON',
            httpCode: 400
        ))->send();
        exit();
    }
    if (!isset($dadosJson->cargos)) {
        (new Response(
            success: false,
            message: 'Dados de cargos não encontrados no JSON',
            httpCode: 400
        ))->send();
        exit();
    }
```

importJson(array $jsonFile)1/2

```
    $cargoDAO = new CargoDAO();
    $cargosCriados = [];  // Array para armazenar os cargos criados com sucesso
    $cargosNaoCriados = [];  // Array para armazenar os cargos que não foram criados
    foreach ($dadosJson->cargos as $cargoNode) {
        $cargo = new Cargo();
        $cargo->setIdCargo(idCargo: (int) $cargoNode->idCargo)
            ->setNomeCargo(nomeCargo: (string) $cargoNode->nomeCargo);

        $cargoCriado = $cargoDAO->create(cargo: $cargo);
        if ($cargoCriado == false) {
            $cargosNaoCriados[] = $cargo;
        } else {
            $cargosCriados[] = $cargo;
        }
    }
    (new Response(
        success: true,
        message: 'Importação realizada com sucesso’,
        data: [
            "cargosCriados" => $cargosCriados,
            "cargosNaoCriados" => $cargosNaoCriados,
        ],
        httpCode: 200
    ))->send();
    exit();  // Finaliza a execução após enviar a resposta
}
```

importJson(array $jsonFile) 2/2



O que é XML?

- **XML** (eXtensible Markup Language) é uma **linguagem de marcação** baseada em texto, desenvolvida para **representar e armazenar documentos de forma estruturada**, legível por humanos e por máquinas.

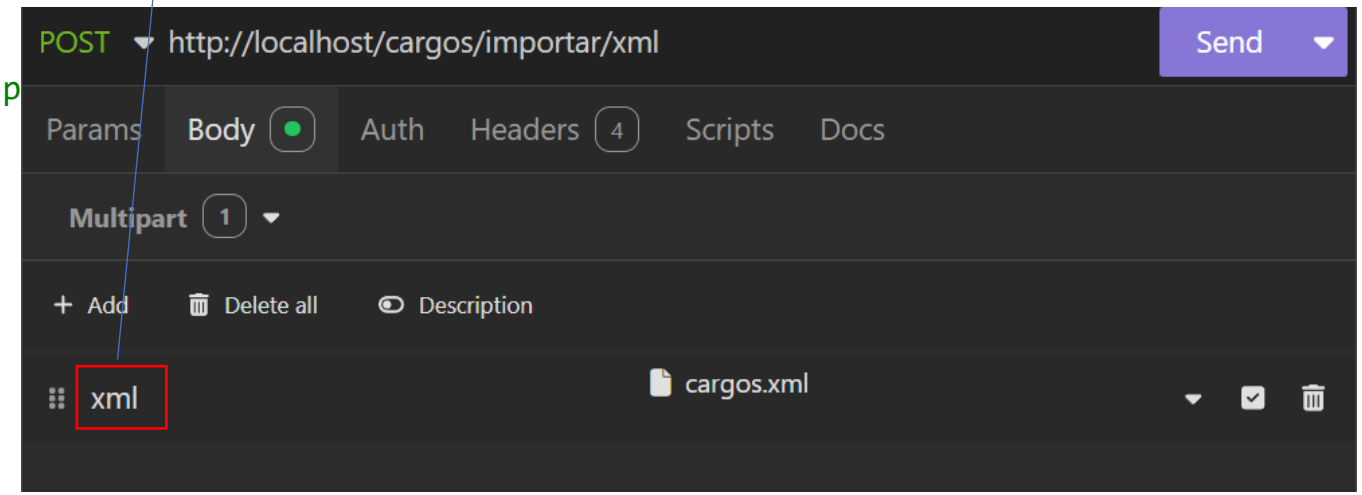
Por que XML é ideal para representar documentos?

- Documentos são mais do que dados
  - Estrutura complexa
  - Hierarquia
  - Metadados (informações sobre os dados)
  - Regras rígidas de validação
  - Conteúdo que pode variar em forma, estilo e profundidade.
- Exemplos:
  - Contratos, Notas fiscais

Arquivos .xml

```
<?xml version="1.0"?>
<cargos>
    <cargo>
        <idCargo>1</idCargo>
        <nomeCargo>Administrador :d)</nomeCargo>
    </cargo>
    <cargo>
        <idCargo>4</idCargo>
        <nomeCargo>Analista de Sistemas Jr</nomeCargo>
    </cargo>
    <cargo>
        <idCargo>7</idCargo>
        <nomeCargo>Engenheiro Jr</nomeCargo>
    </cargo>
    <cargo>
        <idCargo>2</idCargo>
        <nomeCargo>T&#xE9;cnico em Inform&#xE1;tica Jr</nomeCargo>
    </cargo>
    <cargo>
        <idCargo>3</idCargo>
        <nomeCargo>T&#xE9;cnico em Inform&#xE1;tica Pleno</nomeCargo>
    </cargo>
</cargos>
```

```php
$this->router->post(pattern: "/cargos/importar/xml", fn: function (): never {
    try {
        $controle = new CargoControl();
        $controle->importXML(xmlFile: $_FILES['xml']); // Processa e importa os dados do arquivo XML
        exit();
    } catch (Throwable $throwable) {
        // Caso ocorra um erro durante o processo de exportação (ex: falha ao gerar o JSON),
        // a exceção é capturada e uma resposta de erro é enviada ao cliente.
        $this->sendErrorResponse(
            throwable: $throwable, // Passa a exceção gerada para o tratamento de erro
            message: 'Erro na exportação xml' // Mensagem explicativa do erro ocorrido
        );
        // Finaliza a execução do scrip
        exit();
    }
});
```



```php
public function importXML(array $xmlFile): never    {
        // Obtém o nome temporário do arquivo XML enviado pelo formulário HTML
        $nomeTemporario = $xmlFile['tmp_name'];
        // Carrega o XML a partir do arquivo temporário
        $xml = simplexml_load_file(filename: $nomeTemporario);
        // Verifica se o arquivo XML foi carregado corretamente
        if (!$xml) {
            // Caso o XML não seja carregado, envia uma resposta de erro
            (new Response(
                success: false,
                message: 'Erro ao carregar o arquivo XML',
                httpCode: 400
            ))->send();
            exit();  // Finaliza a execução após enviar o erro
        }
```

```
$cargoDAO = new CargoDAO();
$cargosCriados = [];  // Array para armazenar os cargos criados com sucesso
$cargosNaoCriados = [];  // Array para armazenar os cargos que não foram criados
// Loop para processar cada cargo no arquivo XML
foreach ($xml->cargo as $cargoNode) {
    // Cria um novo objeto da classe Cargo e define seus atributos
    $cargo = new Cargo();
    $cargo->setIdCargo(idCargo: (int) $cargoNode->idCargo)  // Define o ID do cargo
        ->setNomeCargo(nomeCargo: (string) $cargoNode->nomeCargo);  // Define o nome do cargo
    // Tenta criar o cargo no banco de dados utilizando o DAO
    $cargoCriado = $cargoDAO->create(cargo: $cargo);
    if ($cargoCriado == false) {
        $cargosNaoCriados[] = $cargo;
    } else {
        $cargosCriados[] = $cargo;
    }
}
(new Response(
    success: true,
    message: 'Importação realizada com sucesso',
    data: [
        "cargosCriados" => $cargosCriados,
        "cargosNaoCriados" => $cargosNaoCriados,
    ],
    httpCode: 200
))->send();

   exit();  // Finaliza a execução após enviar a resposta
}
```

Exportar

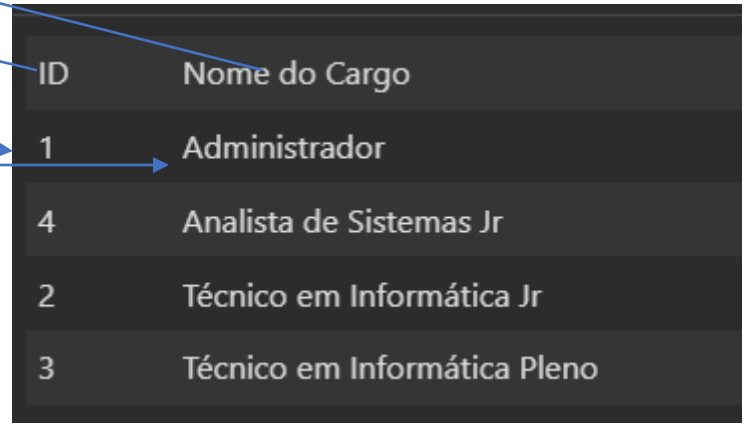
Exportar Dados

Rota /cargos/exportar/csv

```php
$this->router->get(pattern: "/cargos/exportar/csv", fn: function (): never {
            try {
                (new CargoControl())->exportCSV();
                exit();
            } catch (Throwable $throwable) {

                $this->sendErrorResponse(
                    throwable: $throwable,
                    message: 'Erro na exportação CSV'
                );
                exit();
            }

        });
```

Controle : Cargo>>ExportCSV

```php
public function exportCSV(): never
        header(header: 'Content-Type: text/csv; charset=utf-8');
        header(header: 'Content-Disposition: attachment; filename="cargos.csv"');
        $cargoDAO = new CargoDAO();
        $cargos = $cargoDAO->readAll();
        $saida = fopen(filename: 'php://output', mode: 'w'); //cria um ponteiro para escrita de um arquivo
        fputcsv(stream: $saida, fields: ['ID', 'Nome do Cargo']); //escreve os cabeçalhos no arquivo
        foreach ($cargos as $cargo) {
            fputcsv(stream: $saida, fields: [
                $cargo->getIdCargo(), //coluna zero
                $cargo->getNomeCargo() //coluna um
            ]);
        }

    fclose(stream: $saida); // fecha o ponteiro do arquivo.
        exit();
    }
```



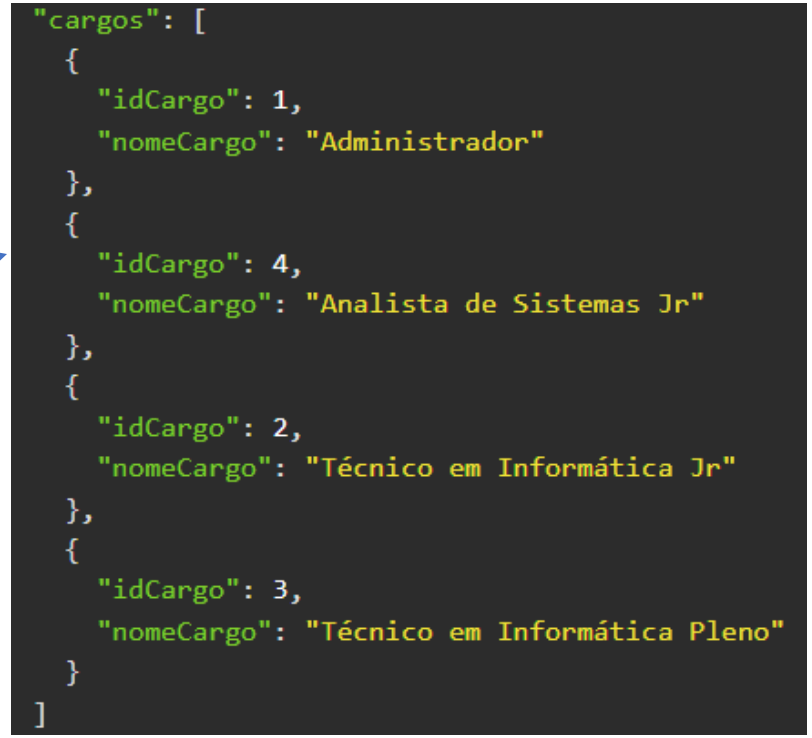
ID	Nome do Cargo
1	Administrador
4	Analista de Sistemas Jr
2	Técnico em Informática Jr
3	Técnico em Informática Pleno

Rota /cargos/exportar/json

```php
$this->router->get(pattern: "/cargos/exportar/json", fn: function (): never {
            try {
                (new CargoControl())->exportJSON(); // Gera o arquivo JSON e envia a resposta ao cliente
                exit();
            } catch (Throwable $throwable) {
                // Caso ocorra um erro durante o processo de exportação (ex: falha ao gerar o JSON),
                $this->sendErrorResponse(
                    throwable: $throwable, // Passa a exceção gerada para o tratamento de erro
                    message: 'Erro na exportação JSON' // Mensagem explicativa do erro ocorrido
                );
                  // Finaliza a execução do script após o erro ser tratado e a resposta enviada
                exit();
            }
        });
```

Controle : Cargo>>ExportJSON

```php
public function exportJSON(): never {
    $cargoDAO = new CargoDAO();
    $cargos = $cargoDAO->readAll();
    // Define os cabeçalhos HTTP apropriados para download de arquivo JSON
    header(header: 'Content-Type: application/json; charset=utf-8');
    header(header: 'Content-Disposition: attachment; filename="cargos.json"');
    // Prepara a resposta com os dados dos cargos
    $resposta = [
        'cargos' => $cargos
    ];
    // Converte os dados para JSON e imprime
    echo json_encode(value: $resposta);
    // Finaliza a execução após envio do arquivo
    exit();
}
```



```json
"cargos": [
  {
    "idCargo": 1,
    "nomeCargo": "Administrador"
  },
  {
    "idCargo": 4,
    "nomeCargo": "Analista de Sistemas Jr"
  },
  {
    "idCargo": 2,
    "nomeCargo": "Técnico em Informática Jr"
  },
  {
    "idCargo": 3,
    "nomeCargo": "Técnico em Informática Pleno"
  }
]
```

Rota /cargos/exportar/xml

```
$this->router->get(pattern: "/cargos/exportar/xml", fn: function (): never {
        try {
            (new CargoControl())->exportXML(); // Gera o arquivo XML e envia a resposta ao cliente
            exit();
        } catch (Throwable $throwable) {
                    $this->sendErrorResponse(
                throwable: $throwable, // Passa a exceção gerada para o tratamento de erro
                message: 'Erro na exportação XML' // Mensagem explicativa do erro ocorrido
            );

               exit();
        }
 });
```

Controle : Cargo>>ExportXML

```php
public function exportXML(): never    {
    $cargoDAO = new CargoDAO();
    $cargos = $cargoDAO->readAll();
       // Define os cabeçalhos HTTP apropriados para download de arquivo XML
    header(header: 'Content-Type: application/xml; charset=utf-8');
    header(header: 'Content-Disposition: attachment; filename="cargos.xml"');
       // Cria um novo objeto SimpleXMLElement para armazenar o conteúdo XML
    $xml = new SimpleXMLElement('<cargos/>');
       // Adiciona os dados dos cargos ao XML
    foreach ($cargos as $cargo) {
        // Cria um elemento <cargo> para cada cargo
        $cargoXML = $xml->addChild(qualifiedName: 'cargo');
           // Adiciona os dados do cargo como elementos dentro de <cargo>
        $cargoXML->addChild(qualifiedName: 'idCargo', value: $cargo->getIdCargo());
        $cargoXML->addChild(qualifiedName: 'nomeCargo', value: $cargo->getNomeCargo());
    }
       // Exibe o conteúdo XML gerado
    echo $xml->asXML();

    exit();
}
```

```xml
<cargo>
  <idCargo>1</idCargo>
  <nomeCargo>Administrador</nomeCargo>
</cargo>
```