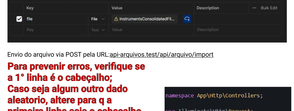
Yan Martins Lourenço Fraga - yanlouifraga@gmail.com

DOCUMENTAÇÃO - API de upload de arquivos / OT

A **API** possui 3 funcionalidades principais:
- Upload de arquivos;
- Histórico de upload;
- Buscas de dados dos arquivos;

UPLOAD



Envio do arquivo via POST pela URL:api-arquivos.test/api/arquivos/import

Para prevenir erros, verifique se a 1° linha é o cabeçalho; Caso seja algum outro dado aleatorio, altere para q a primeira linha seja o cabeçalho.

Importações necessárias do controller do import

```
namespace App\Http\controllers;

use Illuminate\Http\Request;
use App\Imports\ArquivoImport;
use App\Models\Arquivo;
use Maatwebsite\Excel\Facades\Excel;
```

```
$file = $request->file('file'); // Pega o arquivo enviado
$nomeoriginal = $file->getClientOriginalName(); // Pega o nome original do arquivo
$hash = md5_file($file->getRealPath()); // Gera um hash do arquivo para verificar duplicidade
```

Na 1° parte, dentro da function, recebe 3 variaveis:
$file - Recebe o arquivo;
$nomeoriginal - Pra salvar o nome como veio do arquivo, para buscas futuras;
$hash - Um identificador para evitar duplicidade do arquivo no banco de dados;

```
$request->validate([ // Valida o arquivo enviado
    'file' => 'required|file|mimes:xlsx,xls,csv,xlsm,CSV,XLS,XLS',
], [
    'file.required' => 'O arquivo é obrigatório.',
    'file.file' => 'O arquivo deve ser um arquivo válido.',
    'file.mimes' => 'O arquivo deve ser do tipo: xlsx, xls, csv, xlsm, CSV, XLSX, XLS.'
]);
```

VALIDATE para fazer as validações necessárias do arquivo para a importação;

```
if (Arquivo::where('file_hash', $hash)->exists()) {
    return response()->json([
        'status' => 'error',
        'message' => 'Este arquivo já foi enviado.'
    ], 422);
}
```

Verificação de duplicidade de arquivo; Caso ja exista, retorna que o arquivo ja foi enviado

```
{
    "status": "error",
    "message": "Este arquivo já foi enviado."
}
```

```
// Cria o registro de identificação do arquivo na tabela arquivos
$registroArquivo = Arquivo::create([
    'file_hash' => $hash,
    'file_name' => $nomeoriginal
]);
```

Cria as colunas de identificação do arquivo no banco de dados

```
ImportCtcVJob::dispatch($path, $registroArquivo->id);
```

passa o arquivo pra JOB, pra ser importado em 2° plano

```
return response()->json([
    'message' => 'Recebido, o arquivo foi enviado para a importação em 2º plano',
    'arquivo_id' => $registroArquivo->id
]);
```

retorna mensagem de sucesso

```
catch (\Exception $e) {
    // Retorna erro em JSON em qualquer a aplicação
    return response()->json([
        'status' => 'error',
        'message' => $e->getMessage()
    ], 500);
}
```

Em caso de algum outro erro inesperado, ele retorna o erro sem quebrar a aplicação

```
Excel::queueImport(
    new ArquivoImport($this->arquivoId),
    $filePath,
    'local',
    \Maatwebsite\Excel\Excel::CSV,
    ['delimiter' => ';']
);
```

-->

```
return new \Illuminate\Http\JsonResponse([
    'arquivo_id' => $this->arquivoId,
    'status'     => $row['status'] ?? null,
    'tckrsymb'   => $row['tckrsymb'] ?? null,
    'mktem'      => $row['mktem'] ?? null,
    'sctyctgnm'  => $row['sctyctgnm'] ?? null,
    'isin'       => $row['isin'] ?? null,
    'crpnnm'     => $row['crpnnm'] ?? null
];
```

JOB

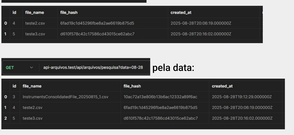
HISTORICO de upload

Quando der um GET na URL: api-arquivos.test/api/arquivos/

	id	file_name	file_hash	created_at
0	1	InstrumentosConsolidadeEf.te_20250815.Lcze	10ac72af3e6f061396ac123332a8f9fac	2025-08-28T00:10:28.000000Z
1	4	teste2.csv	6fa81fdc1d42269fbe8a3ae66318db75d5	2025-08-28T20:06:16.000000Z
2	5	teste3.csv	d61f0ff7fc4317586c643015ccd3a0c7	2025-08-28T20:16:10.000000Z

Ele retorna, todos os arquivos que esta no banco de dados, com o nome original, hash e a data de upload.

```
public function index(Request $request)
    $arquivos = Arquivo::with('linhas')->get();
    return response()->json($arquivos); // retorna todos os arquivos
```



Podendo pesquisar também pelo id: api-arquivos.test/api/arquivos/{id}

id	3
file_name	InstrumentosConsolidadeEf.te_20250815.Lcze
file_hash	10ac72af3e6f061396ac123332a8f9fac
created_at	2025-08-28T10:12:28.000000Z
updated_at	2025-08-28T10:12:28.000000Z
linhas	[ ]

```
public function show($id)
    try {
        return Arquivo::with('linhas')->find($id); // retorna o arquivo pelo ID
    } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
        return response()->json([
            'status' => 'error',
            'message' => 'Nenhum arquivo encontrado.'
        ], 500);
    }
}
```

pelo nome:

	id	file_name	file_hash	created_at
0	4	teste2.csv	6fa81fdc1d42269fbe8a3ae66318db75d5	2025-08-28T20:06:16.000000Z
1	5	teste3.csv	d61f0ff7fc4317586c643015ccd3a0c7	2025-08-28T20:16:10.000000Z

pela data:

	id	file_name	file_hash	created_at
0	1	InstrumentosConsolidadeEf.te_20250815.Lcze	10ac72af3e6f061396ac123332a8f9fac	2025-08-28T10:12:28.000000Z
1	4	teste2.csv	6fa81fdc1d42269fbe8a3ae66318db75d5	2025-08-28T20:06:16.000000Z
2	5	teste3.csv	d61f0ff7fc4317586c643015ccd3a0c7	2025-08-28T20:16:10.000000Z

```
// filtro por data
if (isset($request->data)) {
    $query->where('file_name', 'like', '%' . $request->query('name') . '%');

// filtro por data (created_at)
if ($request->has('data')) {
    $query->where('created_at', 'like', '%' . $request->query('data') . '%');
```

Como os filtros de pesquisa são feitos com o LIKE do MySQL, não precisa pesquisar a data toda, ou o nome exato, caso tenha pelo menos uma parte igual, ele ja vai retornar o arquivo desejado.

Caso o arquivo não seja encontrado ele retorna uma mensagem

```
{
    "status": "vazio",
    "message": "Nenhum arquivo encontrado"
}
```

Buscas de dados

```
// Lista de campos permitidos para busca
$camposPermitidos = [
    'RptDt',
    'rptdt',
    'TckrSymb',
    'tckrsymb',
    'Mktem',
    'mktem',
    'SctyCtgNm',
    'sctyctgnm',
    'ISIN',
    'isin',
    'CrpnNm',
    'crpnnm'
];
```

para prevenir, coloquei campos com maiúsculas e minúsculos, como: RptDt ou rptdt; TckrSymb ou tckrsymb;

```
//  Se não tiver filtro, retorna o ultimo arquivo, paginado
if (empty($filtros)) {

    $arquivoId = $request->input('arquivo_id');

    // Se não for especificado o arquivo, pega o último importado
    if (!$arquivoId) {
        $arquivoId = \App\Models\Arquivo::latest('id')->value('id');
    }

    $linhas = \App\Models\LinhaArquivo::where('arquivo_id', $arquivoId)
        ->paginate(20);
    return response()->json($linhas);
```

Aqui ele vai verificar se foi passado algum filtro, caso não tenha sido passado nenhum filtro, ele vai buscar no ultimo arquivo importado, e retornar paginando 20 linhas por vez

	id	arquivo_id	rptdt	tckrsymb
0	1	1	15/08/2025	003H11
1	2	1	15/08/2025	005H11
2	3	1	15/08/2025	2N0W3

	id	arquivo_id	rptdt	tckrsymb
0	21	1	15/08/2025	A1EU34Q
1	22	1	15/08/2025	A1E3U8K
2	23	1	15/08/2025	A1EX04

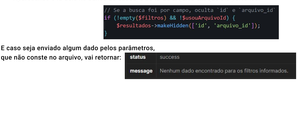
Pode ser feito pesquisas, pelo: arquivo_id; page; e pela colunas (RptDt / TckrSymb / etc...)

```
// Se não for especificado o arquivo, pega o último importado
$arquivoId = $request->input('arquivo_id');
if (!$arquivoId) {
    $arquivoId = \App\Models\Arquivo::latest('id')->value('id');

// Cria a query
$query = \App\Models\LinhaArquivo::where('arquivo_id', $arquivoId);

foreach ($filtros as $campo => $valor) { // percorre os filtros e monta a query
    $query->where(DB::raw('LOWER(CAST(' . $campo . ' AS TEXT))'), 'LIKE', '%'.strtolower($v).'%'); // Busca em caso sensitive

// Retorna todos os colunas (kcl) ordenados
$resultados = $query->get();
```

Caso a pesquisa seja feita so pela coluna, ele vai retornar o dado do ultimo arquivo importado.

	rptdt	tckrsymb	mktem	sctyctgnm	isin	crpnnm
0	15/08/2025	003H11	EQUITY-CASH	PUBLIC	BRBEMCTC0B20	KINEA CO.INVESTIMENTO FDO IMV NOB

porém, caso a pesquisa seja feito em algum arquivo especifico (arquivo_id), o retorno tera alguns dados a mais.



	arquivo_id	rptdt	tckrsymb	mktem	sctyctgnm	isin	crpnnm
0	1	15/08/2025	003H11	EQUITY-CASH	PUBLIC	BRBEMCTC0B20	KINEA CO.INVESTIMENTO FDO IMV NOB

Aparecendo o id dele no banco de dados, e ao ID do arquivo em que ele pertence.

```
// Se a busca não tiver resultados, retorna um erro.
if (!empty($filtros) && $resultados[0]) {
    $resultados->makeHidden(['id', 'arquivo_id']);
```

E caso seja enviado algum dado pelos parâmetros, que não conste no arquivo, ou retornar:

```
{
    "status": "success",
    "message": "Nenhum dado encontrado para os filtros informados."
}
```