

# Boas Práticas e Segurança

- [Entrada de Dados – Boas Práticas e Segurança](#)

# Entrada de Dados – Boas Práticas e Segurança

## Objetivo

Padronizar o tratamento de dados recebidos de usuários, formulários, APIs e URLs, garantindo segurança, integridade e confiabilidade no sistema.

## Por que isso é importante?

- Evita falhas como SQL Injection, XSS e CSRF
- Garante consistência nos dados armazenados
- Evita falhas de sistema por dados malformados
- Reduz risco de vazamentos, invasões e comportamento inesperado

## Fontes comuns de entrada de dados

- Formulários HTML (POST/GET)
- Query strings (URLs)
- Cookies e headers HTTP
- Requisições via API (JSON/XML)
- Arquivos enviados via upload
- Sessões manipuladas via navegador

## Regras gerais para qualquer entrada

1. Nunca confie em dados externos — sempre valide no servidor
2. Sempre trate os dados antes de usar em banco, lógica ou exibição

- 3. Não use dados crus do usuário em queries, HTML ou arquivos

## Tipos de validação e sanitização

### 1. Números

- Use intval() ou floatval() para garantir que o dado é numérico

```
$id = intval($_GET['id']);
```

### 2. Strings

- Use pg\_escape\_string() com conexão ativa antes de usar na query

```
$nome = pg_escape_string($conn, $_POST['nome']);
```

### 3. Datas

- Valide com DateTime::createFromFormat()

```
$data = DateTime::createFromFormat('Y-m-d', $_POST['data']);
```

### 4. Booleanos

- Normalize com filter\_var() e cast:

```
$ativo = filter_var($_POST['ativo'], FILTER_VALIDATE_BOOLEAN);
```

### 5. Emails

- Valide com filter\_var():

```
$email = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);
```

### 6. Uploads

- Valide tipo MIME com `finfo_file()`
- Renomeie o arquivo e armazene na tmp
- Nunca confie apenas na extensão

## Exemplos de tipos MIME por extensão - com variações possíveis

Tipo de Arquivo	Extensão(s)	MIME Types comuns
Texto simples	.txt	text/plain application/octet-stream (alguns sistemas)
Imagem JPEG	.jpg, .jpeg	image/jpeg image/pjpeg (antigo)
Imagem PNG	.png	image/png
PDF	.pdf	application/pdf application/x-pdf application/octet-stream (em alguns navegadores)
DOC (Word 97-2003)	.doc	application/msword application/octet-stream
DOCX (Word novo)	.docx	application/vnd.openxmlformats-officedocument.wordprocessingml.document application/zip
XLS (Excel antigo)	.xls	application/vnd.ms-excel
XLSX (Excel novo)	.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
CSV	.csv	text/csv application/vnd.ms-excel text/plain
ZIP	.zip	application/zip application/x-zip-compressed multipart/x-zip

`application/octet-stream` é genérico - aparece quando o sistema não consegue identificar corretamente o tipo.

Um `.txt` pode retornar `text/plain`, mas também `application/octet-stream` dependendo do upload ou do sistema operacional.

DOCX, XLSX, PPTX são arquivos ZIP internamente, então às vezes o MIME vem como application/zip.

## Código flexível (aceitando múltiplos MIME por tipo)

### Legado:

```
$tiposPermitidos = [  
    ['pdf' => [  
        ['application/pdf',  
        ['application/x-pdf',  
        ['application/octet-stream'  
    ],  
    ['txt' => [  
        ['text/plain',  
        ['application/octet-stream'  
    ],  
    ['jpg' => [  
        ['image/jpeg',  
        ['image/pjpeg'  
    ],  
    ['png' => [  
        ['image/png',  
        ['application/octet-stream'  
    ],  
    ['doc' => [  
        ['application/msword',  
        ['application/octet-stream'  
    ],  
    ['docx' => [  
        ['application/vnd.openxmlformats-officedocument.wordprocessingml.document',  
        ['application/zip',  
        ['application/octet-stream'  
    ],  
];  
  
$extensao = strtolower(pathinfo($_FILES['arquivo']['name'], PATHINFO_EXTENSION));  
  
$mime = finfo_file(finfo_open(FILEINFO_MIME_TYPE), $_FILES['arquivo']['tmp_name']);
```

```
if (!isset($tiposPermitidos[$extensao]) || !in_array($mime, $tiposPermitidos[$extensao])) {  
    die('Tipo de arquivo não permitido.');
```

## std/File.php -> validarUploadArquivo

Este método foi criado para **validar arquivos enviados por upload de forma segura e confiável**.

Ele realiza uma série de verificações importantes:

- Garante que o arquivo foi realmente enviado e está bem formado
- Verifica se a **extensão** está entre as permitidas
- Confirma se o **tipo MIME real** do arquivo corresponde à extensão.
- Impede arquivos maiores do que o tamanho máximo definido
- Bloqueia arquivos com conteúdo malicioso, como código PHP embutido .

```
include(modification("std/File.php"));  
  
$resultado = File::validarUploadArquivo( $_FILES['AArquivo'], array('txt', 'csv', 'docx'), 5 * 1024 * 1024 ); // 5MB  
  
if ($resultado['erro']) {  
    die("Erro: " . $resultado['mensagem']);  
}
```

## Laravel Request

```
class ExemploCompletoRequest extends FormRequest  
{  
    public function authorize()  
    {  
        return true;  
    }  
  
    public function rules()  
    {  
        return [  
            // Texto obrigatório com tamanho máximo  
            'nome' => ['required', 'string', 'max:100'],  
            // E-mail com formato válido  
            'email' => ['required', 'email'],  
            //Inteiro mínimo 18 (ex: idade)
```

```

'idade' => ['nullable', 'integer', 'min:18'],
// Booleano (aceita 1, 0, true, false)
'ativo' => ['required', 'boolean'],
// Data
'data' => ['required', 'date'],
// Upload com restrição de tipo e tamanho (em KB)
'arquivo' => [
    'required',
    'file',
    'max:5120', // 5 MB
    'mimetypes:' .
        'application/pdf,' .
        'image/jpeg,' .
        'image/png,' .
        'text/plain,' .
        'application/msword,' .
        'application/vnd.openxmlformats-officedocument.wordprocessingml.document'
],
'descricao' => ['nullable', 'string'] // Texto opcional
];
}

```

```

public function messages()
{
    return [
        'nome.required' => 'O nome é obrigatório.',
        'email.required' => 'O e-mail é obrigatório.',
        'email.email' => 'Informe um e-mail válido.',
        'idade.integer' => 'A idade deve ser um número inteiro.',
        'idade.min' => 'A idade mínima é 18 anos.',
        'ativo.required' => 'O campo ativo é obrigatório.',
        'ativo.boolean' => 'O campo ativo deve ser verdadeiro ou falso.',
        'data.date' => 'Informe uma data válida.',
        'arquivo.required' => 'Selecione um arquivo.',
        'arquivo.file' => 'O arquivo enviado é inválido.',
        'arquivo.max' => 'O arquivo não pode ter mais que 5MB.',
        'arquivo.mimetypes' => 'Tipo de arquivo não permitido.',
    ];
}
}

```

## Proteção contra injeções e ataques

```
$nome = $_POST['nome'];  
db_query("SELECT * FROM usuarios WHERE nome = '$nome'");
```

```
$nome = pg_escape_string($conn, $_POST['nome']);  
db_query("SELECT * FROM usuarios WHERE nome = '$nome'");
```

## Cuidados comuns

Não exibir mensagens de erro em produção (display\_errors = off)

Nunca armazenar senhas em texto plano

Nunca confiar que um número enviado realmente é um número

Não confiar em campos ocultos no HTML

Não confiar em lógica de permissão baseada só no front-end

Não salve nada que não foi previamente tratado