

Abstract geometric shapes in the top-left corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a pink parallelogram, all overlapping and tilted at various angles.

LARAVEL

AULA 04

Abstract geometric shapes in the top-right corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a pink parallelogram, all overlapping and tilted at various angles.

Abstract geometric shapes in the top-left corner, including a light blue parallelogram, a green parallelogram, a brown parallelogram, and a large magenta parallelogram.

REQUESTS

Uso de Request em Laravel



Laravel fornece todos os dados da solicitação atual (os velhos e conhecidos \$_POST e \$_GET) através do `Illuminate\Http\Request`, um objeto que permite a consulta de informações sobre a solicitação e os dados que podem ser enviados.

Para poder trabalhar com o objeto Request em um controlador, é necessário usar uma injeção de dependências.

```
public function nomeMetodo(Request $request)
{
    // Código
}
```

// Método All - Retorna um array com todos os inputs e seus valores

```
$input = $request->all();
```

// Método Input - Retorna o valor de um só input

```
$input = $request->input('nome');
```

// Método Only - Retorna um array apenas com os inputs solicitados

```
$input = $request->only('nome', 'idade');
```

Abstract geometric shapes in the top-left corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a pink parallelogram, all overlapping and tilted at various angles.

VALIDAÇÕES

Como validar em Laravel?

Abstract geometric shapes in the top-right corner, including a green parallelogram, a blue parallelogram, a pink parallelogram, and an orange parallelogram, all overlapping and tilted at various angles.

Laravel oferece alguns **métodos** muito simples para **validar** os dados recebidos em um pedido HTTP.

Vamos analisar a maneira mais rápida de validar, mantendo a lógica no controlador.

```
public function store(Request $request)
{
    $request->validate([
        'title' => 'required|unique:movies|max:255',
        'body'  => 'required',
    ]);
}
```

Regras de Validação

x required	x numeric	x unique
x sometimes	x integer	x exists
x min - max	x boolean	x mimes - mimetypes
x in - not_in	x array	x email
x between	x string	x url
x confirmed	x date	x alpha - alpha_dash
x before - after	x image	- alpha_num

```
// Mostrar erros de Validação na visualização
```

```
@if (count($errors) > 0)

    <div class="alert alert-danger">

        <ul>

            @foreach ($errors->all() as $error)

                <li>{{ $error }}</li>

            @endforeach

        </ul>

    </div>

@endif
```


HORA DE PRATICAR!



Fazer o exercício 1

The background features abstract geometric shapes in the corners. On the left, there are overlapping shapes in shades of green, blue, orange, and purple. On the right, there are overlapping shapes in shades of green, blue, purple, and orange. The central text is positioned between these two clusters of shapes.

MÉTODOS ELOQUENT

Criar, Atualizar, Eliminar

create ();

Gera um array associativo (chave => valor) e **insere** um novo registro no banco de dados.

```
$fillable = ['nome'];  
  
$usuario = User::create([  
    'nome' => $request->input('nome')  
]);
```

// Para poder utilizar este método, devemos lembrar de colocar o atributo (que neste caso é “nome”) como **\$fillable** dentro do modelo.

save ();

Permite **inserir** um novo registro ou **atualizar** os dados de um registro existente. Para que isso funcione, é importante ter uma instância do modelo criada como mostra o exemplo a seguir.

```
public function store(Request $request)
{
    $usuario = User::create([
        'nome' => $request->input('nome')
    ]);
    $usuario->save();
}
```

save ();

Permite **inserir** um novo registro ou **atualizar** os dados de um registro existente. Para que isso funcione, é importante ter uma instância do modelo criada como mostra o exemplo a seguir.

```
public function store(Request $request)
{
    $usuario = Usuario::find(1);
    $usuario->name = $request->input('name');

    $usuario->save();
}
```

delete ();

Permite **eliminar** datos do banco de dados, contanto que o objeto esteja instanciado.

```
$usuario = Usuario::find(1);
```

```
$usuario->delete();
```



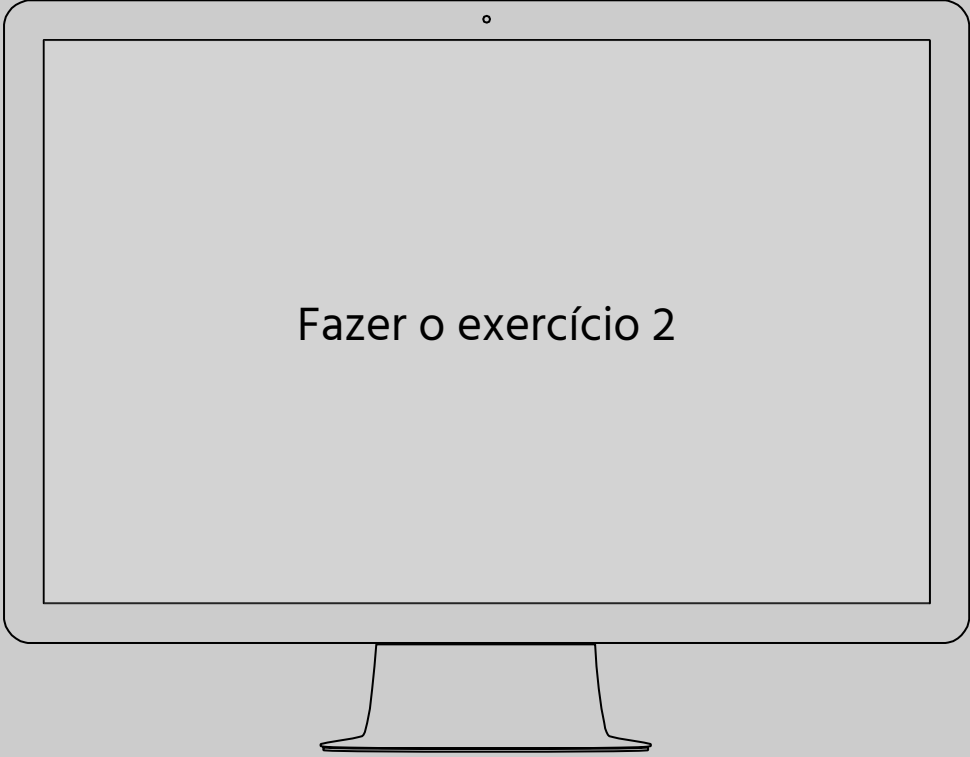
firstOrCreate ();

```
$usuario = Usuario::firstOrCreate([  
    'nome' => $request->input('nome')  
]);
```

updateOrCreate ();

```
$usuario = Usuario::updateOrCreate([  
    'nome' => $request->input('nome')  
]);
```

HORA DE PRATICAR!



Fazer o exercício 2

**ATÉ A
PRÓXIMA!**

