

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

# **LARAVEL**

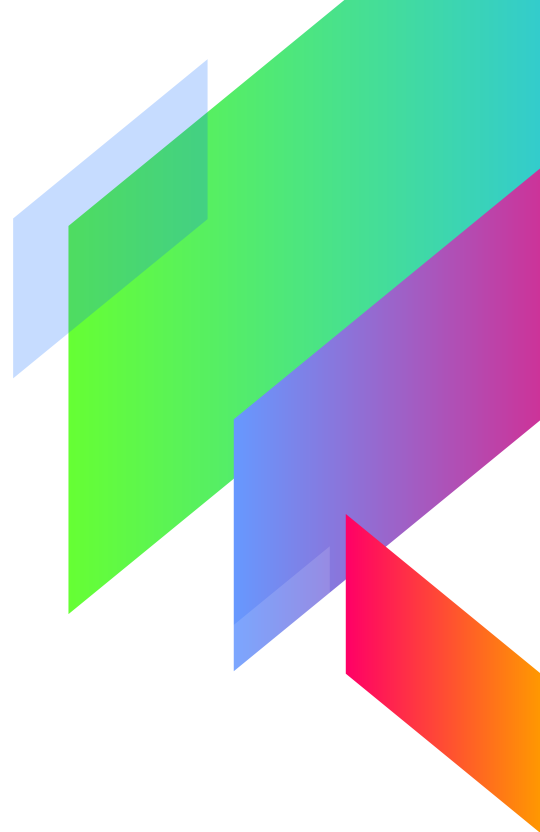
**CLASE 04**

Abstract geometric shapes in the top-right corner, including a green parallelogram, a light blue parallelogram, a purple parallelogram, and a red parallelogram.



# REQUESTS

Uso de Request en Laravel



Laravel nos facilita todos los datos de la solicitud actual (los viejos y conocidos \$\_POST y \$\_GET) a través del `Illuminate\Http\Request`, un objeto sobre el que podremos **consultar información sobre la solicitud** y los datos que pueda estar enviando.

Para poder trabajar con el objeto Request en un controlador lo hacemos a través de inyección de dependencias.

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

// Método All - Nos devuelve un array con todos los inputs y sus valores

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

// Método Input - Nos devuelve el valor de un único input

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

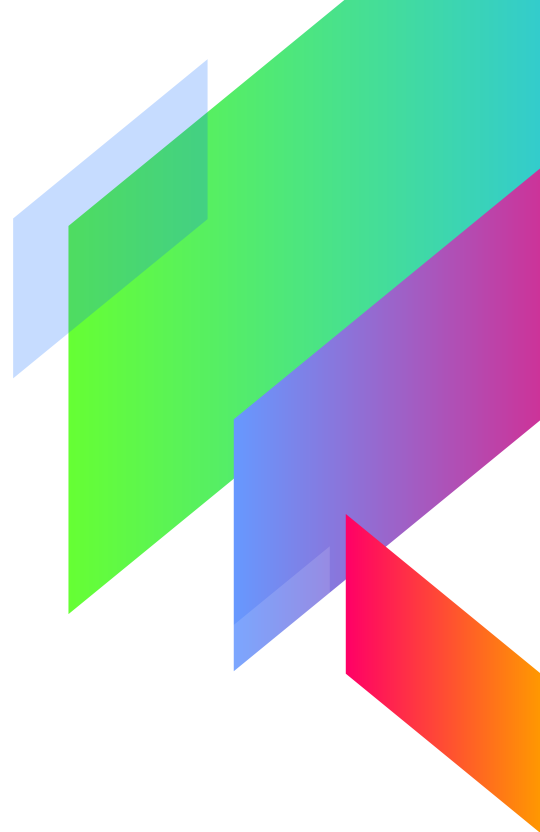
// Método Only - Nos devuelve un array sólo con los inputs solicitados

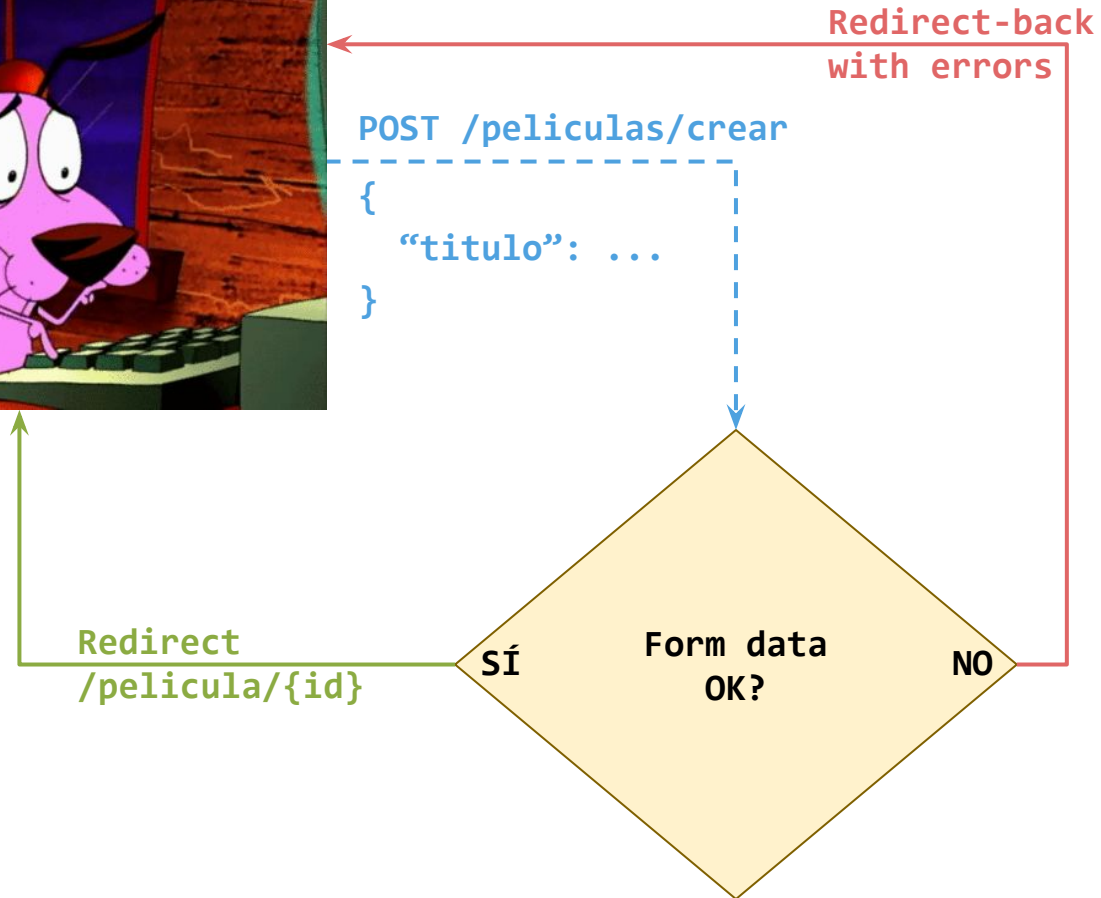
```
$input = $request->only('nombre', 'edad');
```



# VALIDACIONES

¿Cómo validar en Laravel?





Laravel nos proporciona algunos **métodos** muy sencillos para **validar** los datos que recibimos en un pedido HTTP.

Analicemos la manera más rápida de validar, manteniendo la lógica en el controlador.

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

# Reglas de Validación

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



```
// Mostrar errores de Validación en la vista
```

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

    <div class="alert alert-danger">

        <ul>

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

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

            @endforeach

        </ul>

    </div>

@endif
```

**ES MOMENTO DE  
PRACTICAR !**



Abstract geometric shapes in the top-left corner, including a green triangle, a blue parallelogram, a brown trapezoid, and a large purple parallelogram.

# MÉTODOS ELOQUENT

Crear, Actualizar, Eliminar

Abstract geometric shapes in the top-right corner, including a green triangle, a blue parallelogram, a purple parallelogram, and a red triangle.

# Create ();

Genera un array asociativo (clave => valor), e **inserta** un nuevo registro en la base de datos.

```
$usuario = User::create(['nombre' => 'Francisco']);
```

// Para poder utilizar este método debemos recordar colocar el atributo que en este caso es “nombre” como **\$fillable** dentro del modelo.

# Save ();

Nos permite **insertar** un nuevo registro, o, **actualizar** los datos de un registro existente. Para que esto funcione, es importante tener una instancia del modelo creada como vemos en el siguiente ejemplo.

```
public function store(Request $request)
{
    $usuario = new Usuario([
        'name' => $request->input('name'),

    ]);

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

# Save ();

Nos permite **insertar** un nuevo registro, o, **actualizar** los datos de un registro existente. Para que esto funcione, es importante tener una instancia del modelo creada como vemos en el siguiente ejemplo.

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

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

# Delete ();

Nos permite **eliminar** datos de la base de datos, siempre y cuando tengamos instanciado al objeto.

```
$flight = App\Flight::find(1);  
$flight->delete();
```



**ES MOMENTO DE  
PRACTICAR !**





**¡ HASTA LA  
PROXIMA !**

