LARAVEL

CLASE 04

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
}
```

```php
// 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?

Redirect-back
with errors

POST /peliculas/crear

```
{
  “titulo”: ...
}
```

Redirect
/pelicula/{id}

SÍ

Form data
OK?

NO

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

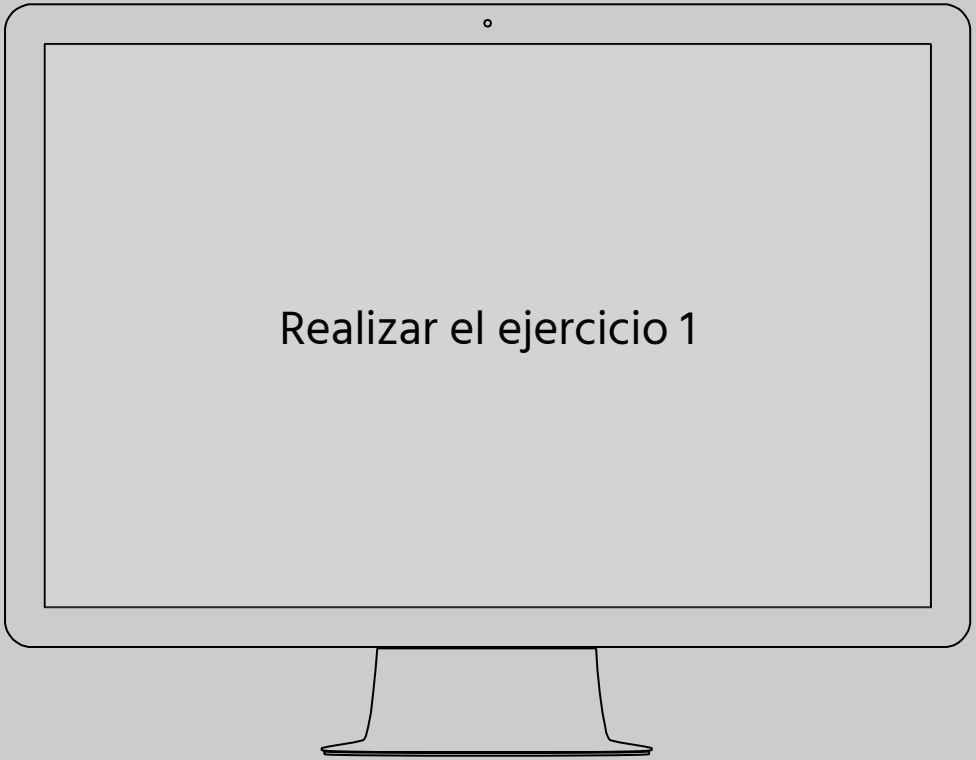
Reglas de Validación

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

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

@if (count($errors))
     <div class="alert alert-danger">
          <ul>
             @foreach ($errors->all() as $error)
                 <li>{{ $error }}</li>
             @endforeach
         </ul>
     </div>
@endif
```

ES MOMENTO DE
PRACTICAR !



MÉTODOS ELOQUENT

Crear, Actualizar, Eliminar

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();
}
```

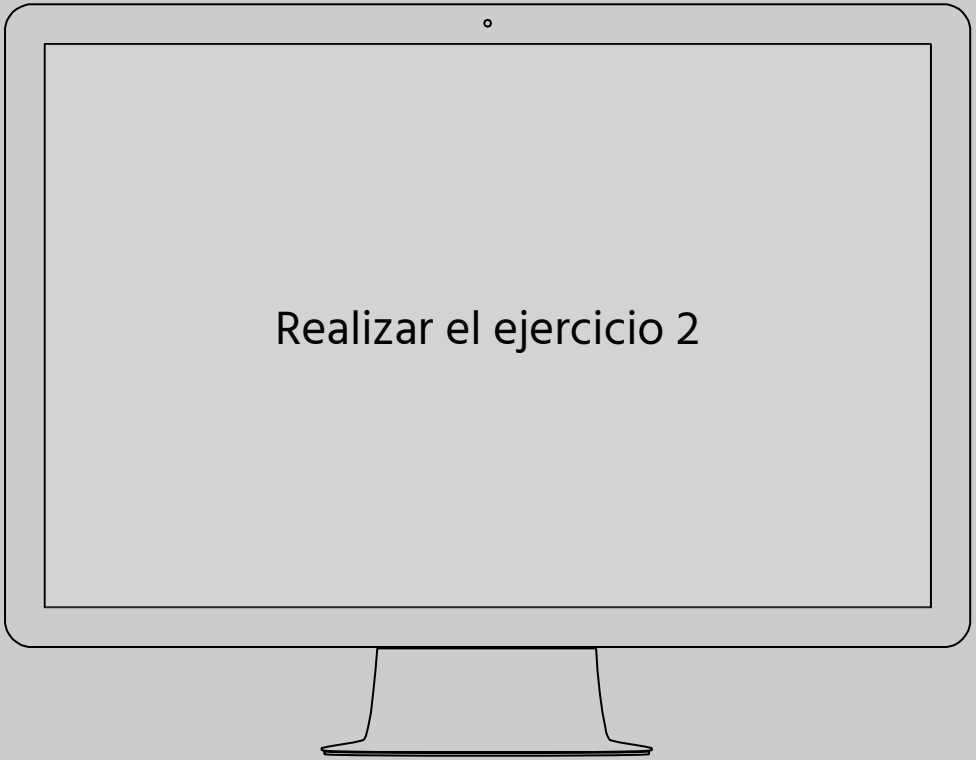
Delete ( );

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 !