La validación de datos de cualquier app es de gran importancia y se deben proveer todos los posibles escenarios y considerar los posibles errores que puede cometer un usuario al interactuar con el sistema.

Básicamente existen 3 formas de validar un formulario, desde el frontend (javascript, jquery, html5),

desde el backend (reglas y funciones de validación) y desde la base de datos, donde se pueden definir los tipos, cantidad de caracteres y otros parámetros que pueden ser configurados para este proceso.

En este ejemplo veremos un poco sobre estos 3 tipos de validación usando Laravel 5 y Eloquent y la validación por defecto de Html5.

Crear nueva base de datos

```
$ mysql -u root -p

$ Enter password :

$ mysql> CREATE DATABASE validation;

$ mysql> exit
```

Validación a nivel de base de datos con Eloquent

Si no conoces mucho sobre el uso de eloquent te recomiendo visitar este tutorial básico de Fluent y Eloquent

Vamos a crear un modelo para registro de clientes haciendo uso de *artisan*

```
$ php artisan make:model client
```

ahora podemos editar la migración de la siguiente forma

```
Schema::create('clients', function(Blueprint $table)
    {
        $table->increments('id');
            $table->string('full_name')->required();
```

```
            $table->string('email')->unique()->required();
            $table->string('address')->required();
            $table->string('phone_number')->nullable();
            $table->enum('type',['empresa','particular'])->default('particular');
            $table->string('register')->nullable();
            $table->timestamps();
        });
```

aprende mas sobre Crear Nuevas Migraciones

required() indica que el registro no puede quedar vacío y que es requerido el valor, *unique()* valida que un valor no pueda repetirse en la tabla, en este caso dos clientes no pueden tener registrado la misma dirección de email, los campos de tipo *enum* se usan para limitar las opciones que pueden ingresar al momento de registrar y el *default()* almacena un valor por defecto en caso de que el usuario no suministre ningún dato.

Por supuesto debemos ejecutar la migración para verificar que todo esté bien

```
$ php artisan migrate
```

editamos el archivo *Client.php* de la siguiente forma

```
<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Client extends Model {

    protected $table = 'clients';

        //Definimos los campos que se pueden llenar con asignación masiva
        protected $fillable = ['full_name', 'email', 'address',
'register','phone_number','type'];

```

```
}
```

Este tipo de validación impide que se registre información que no cuente con los criterios definidos, pero en caso de que el registro falle no ofrece ninguna alerta al usuario, por ello necesitamos usar un método que permita un poco mas de interacción.

Validacion Html5

Iniciaremos creando un formulario (*'form.blade.php'*) de registro para los datos establecidos en la migración usando Form() para facilitar el proceso

```
@extends('app')
@section('content')

<div class="container">
        @if($errors->has())
            <div class="alert alert-warning" role="alert">
               @foreach ($errors->all() as $error)
                  <div>{{ $error }}</div>
              @endforeach
            </div>
        @endif </br>
    <div class="row">
        <div class="col col-md-6 col-md-offset-3"   >
            <div class="panel panel-default">
              <div class="panel-heading"><h3 class="panel-title">Forumulario de registro</h3></div>
              <div class="panel-body">
                {!! Form::open(['route' => 'client', 'method' => 'post']) !!}

                  <div class="form-group">
                      {!! Form::label('full_name', 'Nombre') !!}
                      {!! Form::text('full_name', null, ['class' => 'form-control' , 'required' => 'required']) !!}
```

```
                    </div>
                    <div class="form-group">
                        {!! Form::label('email', 'E-Mail') !!}
                        {!! Form::email('email', null, ['class' => 'form-control' ,
 'required' => 'required']) !!}
                    </div>
                    <div class="form-group">
                        {!! Form::label('address', 'Direccion') !!}
                        {!! Form::text('address', null, ['class' => 'form-control'
, 'required' => 'required']) !!}
                    </div>
                    <div class="form-group">
                        {!! Form::label('phone_number', 'Telefono') !!}
                        {!! Form::text('phone_number', null, ['class' => 'form-
control' ]) !!}
                    </div>
                    <div class="form-group">
                        {!! Form::label('type', 'Tipo') !!}
                        {!! Form::select('type', ['otro' => 'otro', 'empresa' =>
'empresa', 'particular' => 'particular'], null, ['class' => 'form-control' ]) !!}
                    </div>
                    <div class="form-group">
                        {!! Form::label('register', 'Numero de registro') !!}
                        {!! Form::text('register', null, ['class' => 'form-control'
 ]) !!}
                    </div>
                    <div class="form-group">
                        {!! Form::submit('Enviar', ['class' => 'btn btn-success ' ]
 ) !!}
                    </div>
                {!! Form::close() !!}
            </div>
        </div>
    </div>
```

```
    </div>
    @endsection
```

ahora creamos un controlador para clientes llamado *ClientController*

```
$ php artisan make:controller ClientController
```

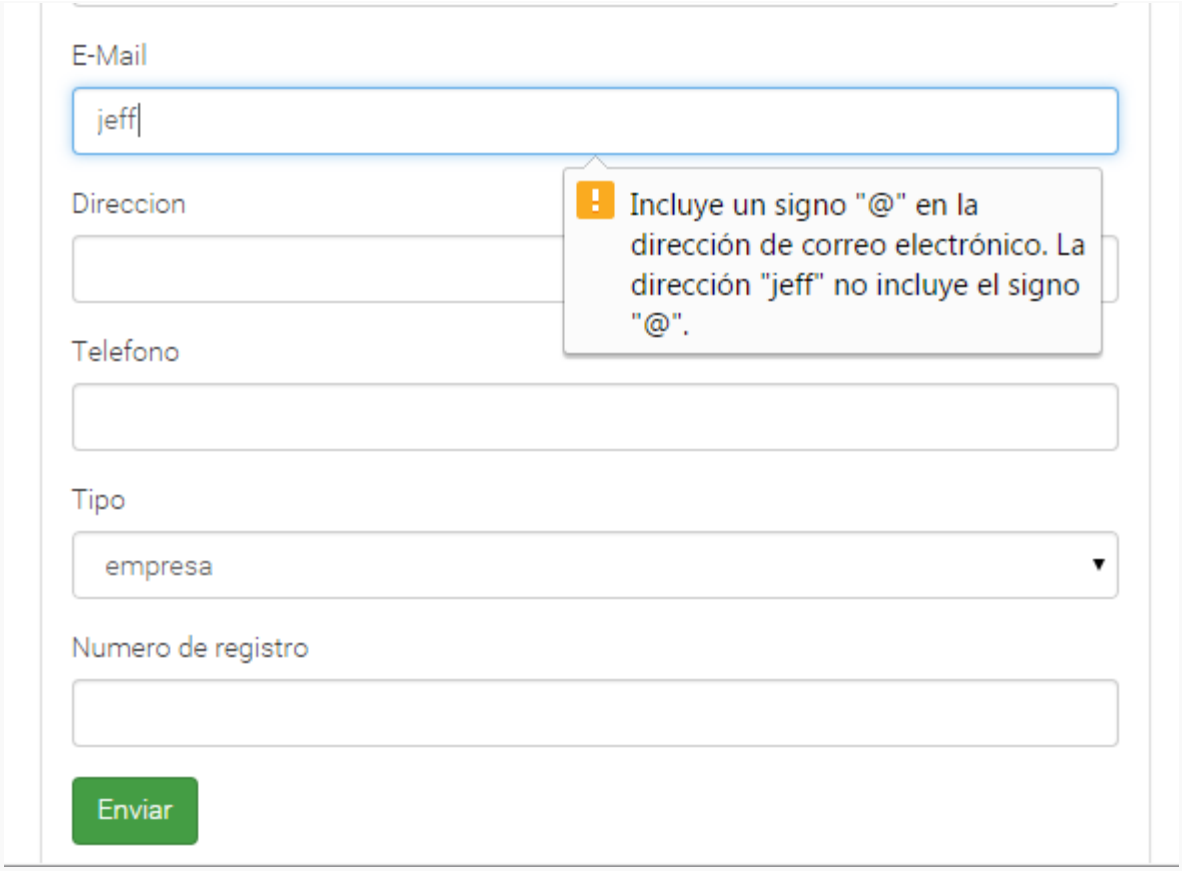
Definimos las rutas de acceso en Laravel

```
Route::get('client', ['as' => 'client', 'uses' => 'ClientController@index']);

Route::post('client', ['as' => 'client', 'uses' => 'ClientController@store']);
```

Creamos la funcion que nos retorna el formulario de registro

```
    public function index()
    {
        return \View::make('form');
    }
```

Si intentamos guardar nos aparecerá un error de validación en pantalla ya que definimos ciertos atributos al crear el formulario como '*required = required*' y para el email no usamos un campo de texto sino uno de tipo email (Html5), por lo cual veremos algo como lo siguiente



Criterios de validación de Laravel

En Laravel podemos hacer uso del método Validator() con el cual podremos establecer ciertas reglas que serán verificadas al momento de enviar los datos, en caso de que la validación falle la aplicación genera los mensajes de error correspondientes para cada campo que podremos mostrar en la vista para indicar al usuario las correcciones que debe realizar.

Vamos a escribir la función que permitirá almacenar los datos de los clientes.

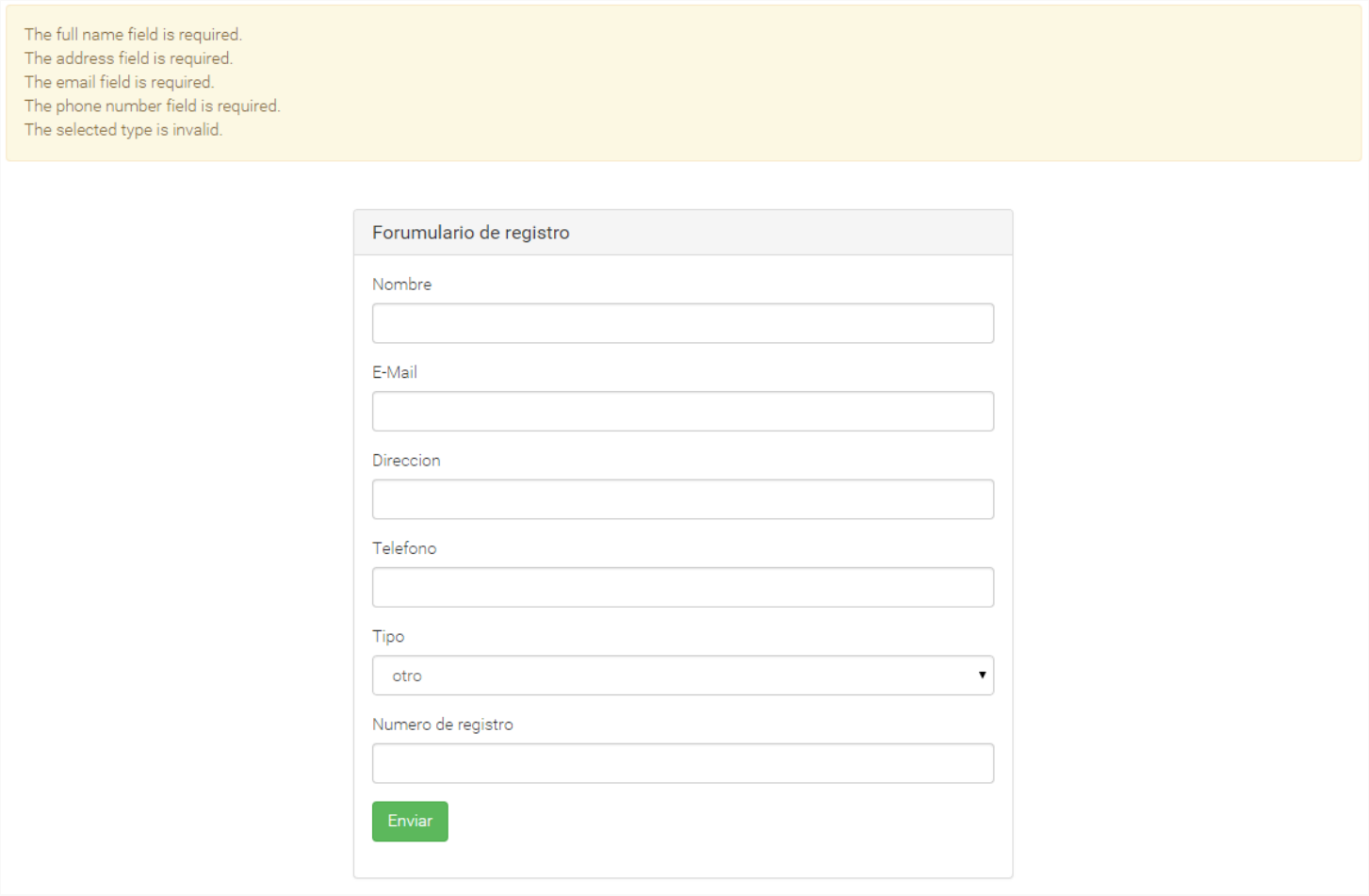
```
1    public function store(Request $request)
2    {
3        $client = new Client;
4
5        $v = \Validator::make($request->all(), [
6
7            'full_name' => 'required',
```

```
            'address' => 'required',
            'email'    => 'required|email|unique:clients',
            'phone_number' => 'required',
            'type' => 'required|in:empresa,particular',
            'register' => 'required_if:type,empresa'
        ]);

        if ($v->fails())
        {
            return redirect()->back()->withInput()->withErrors($v->errors());
        }

        $client->create($request->all());
        $clients = Client::all();
        return \View::make('list', compact('clients'));

    }
```

Perfecto! de esta forma Laravel valida que la información enviada sea la correcta

Para ver los errores que mostrarían las validaciones que colocamos en el controlador, lo puedes hacer desactivando un momento validación HTML5 cambiando esta linea

```
{!! Form::open(['route' => 'client', 'method' => 'post', 'novalidate']) !!}
```




```
'email'     => 'required|email|unique:clients',
```

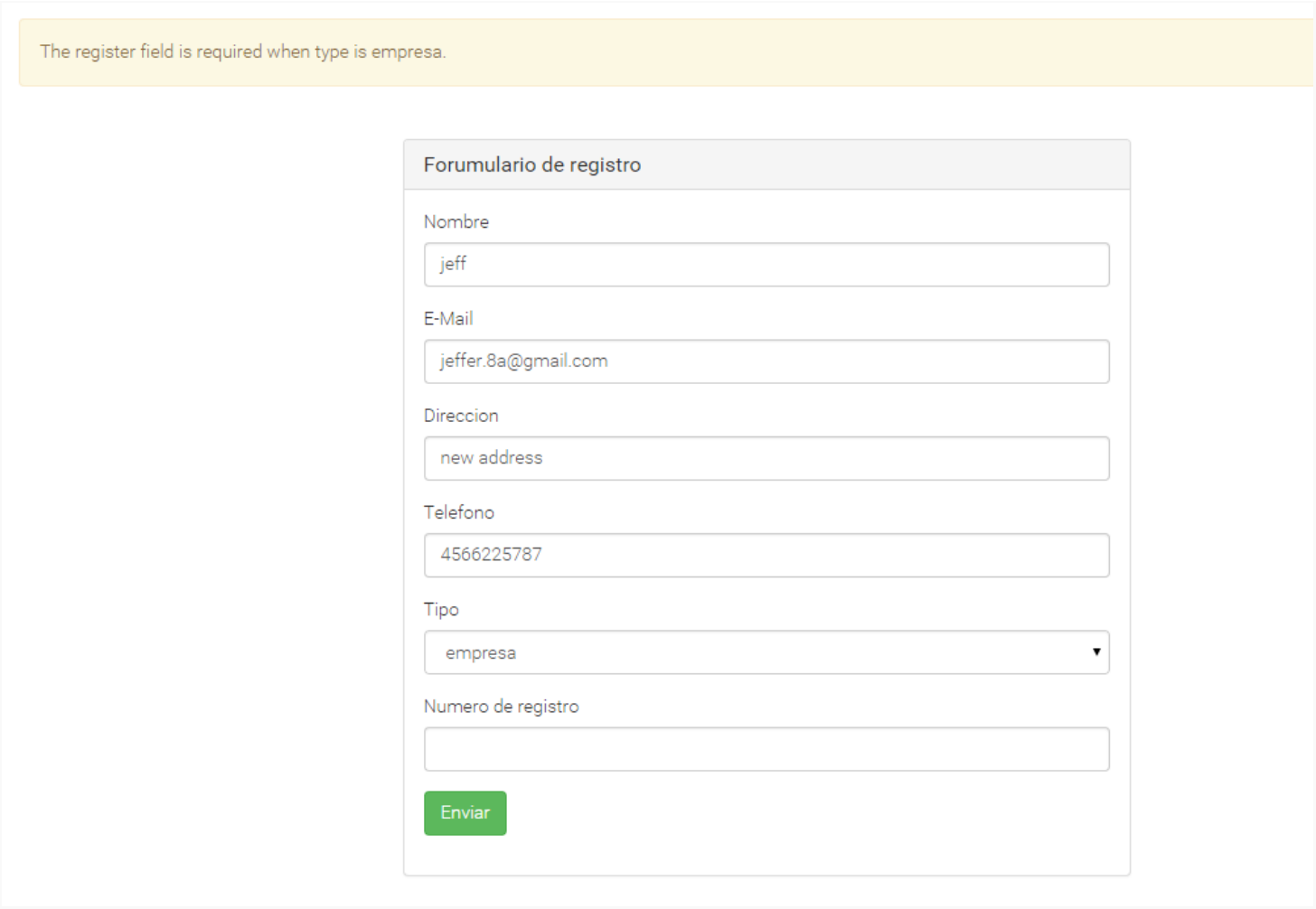
quiere decir que el email es requerido y sera único dentro de la tabla *clients*

```
'type' => 'required|in:empresa,particular',
```

indica que solo admite los valores empresa y particular, por lo cual cuando seleccionamos la opcion "otros" en el formulario nos mostrara un error de validación.

```
'register' => 'required_if:type,empresa'
```

en este caso el campo *register* solo es requerido cuando se haya seleccionado 'empresa' en el campo anterior.



Guardar nuevo registro

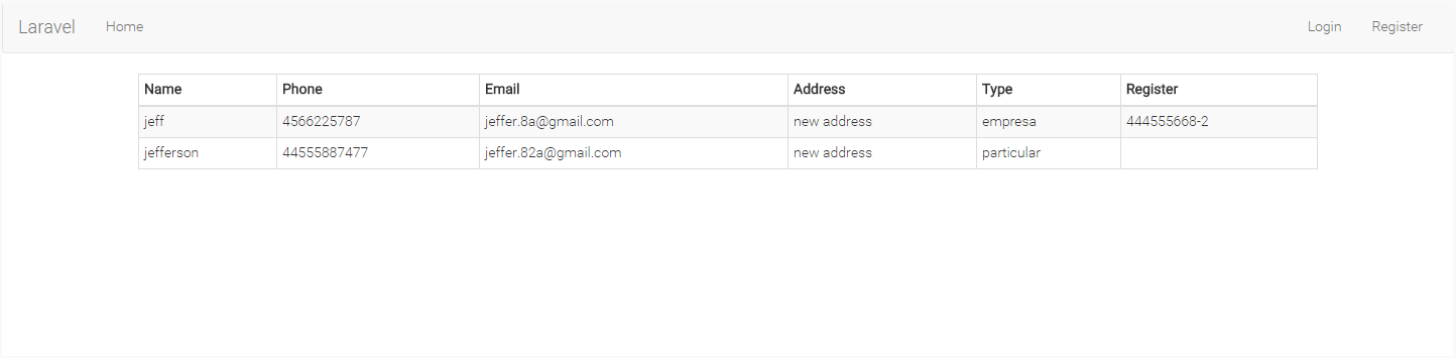
Si no ocurre ningún problema durante la validación el registro podrá ser guardado y redirigir al usuario a la lista de clientes

```
@extends('app')

@section('content')
<div class="container">

    @if (Session::has('deleted'))
      <div class="alert alert-warning" role="alert"> Contacto borrado, si desea deshacer el cambio <a href="{{ route('contact/restore', [Session::get('deleted')]) }}">Click aqui</a> </div>
```

```
    @endif

    @if (Session::has('restored'))
      <div class="alert alert-success" role="alert"> Contacto restaurado</div>
    @endif

    <div class="row">
                <a href="{{ route('client') }}" class="btn btn-xs btn-
primary">Volver</a>
        <table class="table table-condensed table-striped table-bordered">
            <thead>
                <tr>
                  <th>Name</th>
                  <th>Phone</th>
                  <th>Email</th>
                  <th>Address</th>
                  <th>Type</th>
                  <th>Register</th>
                </tr>
            </thead>
            <tbody>
                @foreach($clients as $client)
                <tr>
                    <td>{{ $client->full_name }}</td>
                    <td>{{ $client->phone_number }}</td>
                    <td>{{ $client->email }}</td>
                    <td>{{ $client->address }}</td>
                    <td>{{ $client->type }}</td>
                    <td>{{ $client->register }}</td>
                </tr>
                @endforeach
            </tbody>
        </table>
    </div>
</div>
```

```
@endsection
```

y podremos ver algo así



No olvides dejar tus comentarios y sugerencias.

Recuerda que este tutorial es parte de la serie Laravel desde cero, te invito a que le des un vistazo y aprendas mucho mas sobre este poderoso Framework.