



ISABEL GONZÁLEZ

---

# TIPS LARAVEL

---

2021

---

## Índice

<b>SweetAlert</b>	<b>1</b>
¿Qué es Sweet Alert ?	1
<b>Instalación</b>	<b>1</b>
Composer	1
Instalar las dependencias	1
<b>Probando una redirección</b>	<b>2</b>
Opciones de mensajes	3
<b>Laravel Breeze - Autenticación</b>	<b>4</b>
Instalación	4
<b>Mejorar la seguridad de las contraseñas</b>	<b>4</b>
<b>Testing en Laravel</b>	<b>5</b>
Tipos de pruebas	5
PHPUnit	5
Configurar el entorno de pruebas	5
Configuración del archivo phpunit.xml	5
Configuración del archivo database.php	5
Crear un Test	7
Petición Http Get a la url de Login	7
Ejecutar test	7
<b>Bibliografía</b>	<b>8</b>

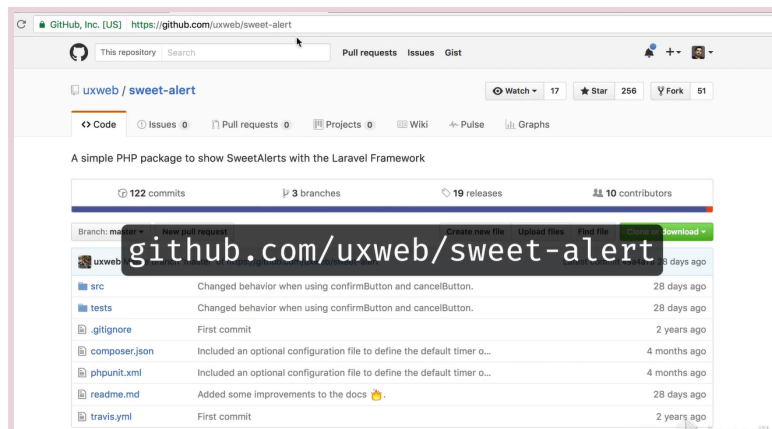
# SweetAlert

## ¿Qué es Sweet Alert ?

Simplemente es una librería de Javascript que permite mejorar la apariencia de los alerts, ofreciendo personalizaciones más atractivas.

## Instalación

Para ello instalaremos la librería



### Installation

Require the package using Composer.

```
composer require uxweb/sweet-alert
```

If using laravel < 5.5 include the service provider and alias within `config/app.php`.

```
'providers' => [
    UxWeb\SweetAlert\SweetAlertServiceProvider::class,
];

'aliases' => [
    'Alert' => UxWeb\SweetAlert\SweetAlert::class,
];
```

## Composer

```
composer require uxweb/sweet-alert
```

## Instalar las dependencias

```
npm install/sweetalert --save-dev
```

## Probando una redirección

En nuestro archivo de routes

```
Route::get('redirect', function(){
    alert()->success('Success Message', 'Optional Title');
    return redirect('/');
});
```

En la vista welcome añadimos el link y la vista que contiene los mensajes de sweet alert

Require sweetalert within your `resources/js/bootstrap.js` file.

```
// ...
require("sweetalert");
// ...
```

Then make sure to include your scripts in your blade layout. Remove the `defer` attribute if your script tag contains it, `defer` will delay the execution of the script which will cause an error as the `sweet::alert` blade template is rendered first by the browser as html.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script>
</head>
<body>
    @include('sweet::alert')
</body>
</html>
```

```
<div class="links">
    <a href="redirect">Redirect</a>
</div>
</div>
</div>
@include('sweet::alert')
</body>
```

incluimos el script

```
</div>
<script src="_/sweetalert/sweetalert.min.js"></script>
@include('sweet::alert')
</body>
```

Añadimos los estilos

```
<!-- Fonts -->
<link href="https://fonts.googleapis.com/css?family=Raleway:100,600" rel="
stylesheet" type="text/css">
<link rel="stylesheet" type="text/css" href="_/sweetalert/sweetalert.min.js">
<!-- Styles -->
```



## Opciones de mensajes

```
SweetAlert::message('Message', 'Optional Title');
```

```
SweetAlert::basic('Basic Message', 'Mandatory Title');
```

```
SweetAlert::info('Info Message', 'Optional Title');
```

```
SweetAlert::success('Success Message', 'Optional Title');
```

```
SweetAlert::error('Error Message', 'Optional Title');
```

```
SweetAlert::warning('Warning Message', 'Optional Title');
```

## Laravel Breeze - Autenticación

### Instalación

```
laravel new breeze
cd breeze
composer require laravel/breeze
php artisan breeze install
php artisan migrate
npm install
npm run dev
```

### Mejorar la seguridad de las contraseñas

```
'password' => ['required',
    Rules\Password::min(1)
        ->letters() // al menos una letra
        ->mixedCase() // al menos una minúscula y una mayúscula
        ->numbers() // al menos un número
        ->symbols() // al menos un símbolo
        ->uncompromised() // No debe estar comprometida
],
```

# Testing en Laravel

## Tipos de pruebas

- Pruebas funcionales
- Pruebas unitarias

## PHPUnit

Laravel cuenta con phpunit integrado.

En el directorio principal del proyecto se encuentra el directorio /tests. Este directorio está separado en dos subdirectorios:

- El directorio Feature: donde escribimos pruebas que emulan peticiones HTTP.
- El directorio Unit: donde escribimos pequeñas partes individuales de la aplicación.

## Configurar el entorno de pruebas

Configuración del archivo phpunit.xml

```
<env name="APP_ENV" value="testing"/>
<env name="DB_CONNECTION" value="sqlite"/>
<env name="DB_DATABASE" value=":memory:"/>
<env name="CACHE_DRIVER" value="array"/>
<env name="SESSION_DRIVER" value="array"/>
<env name="QUEUE_DRIVER" value="sync"/>
```

Configuración del archivo database.php

En el directorio config/database.php.

Debemos dejar la siguiente configuración para el array de 'sqlite'.

```
'sqlite' => [  
  'driver' => 'sqlite',  
  'database' => database_path('database.sqlite'),  
  'prefix' => '',  
],
```



## Crear un Test

```
php artisan make:test LoginTest
```

### Petición Http Get a la url de Login

En el archivo recién creado, LoginTest

```
/** test */  
public function it_visit_page_of_login()  
{  
    $this->get('/login')  
        ->assertStatus(200)  
        ->assertSee('Login');  
}
```

### Ejecutar test

```
vendor/bin/phpunit --filter it_visit_page_of_login
```

## Bibliografía

*Mensajes de sesión con SweetAlert.* (s. f.). Aprendible. Recuperado 26 de mayo de 2021, de <https://aprendible.com/series/laravel-tips/lecciones/video-mensajes-de-sesion-con-sweet-alert-en-laravel-53>

*Testing: Getting Started - Laravel - The PHP Framework For Web Artisans.* (s. f.). Testing. Recuperado 26 de mayo de 2021, de <https://laravel.com/docs/8.x/testing>

*Crear pruebas automatizadas en Laravel con phpunit.* (s. f.). CódigoFacilito. Recuperado 26 de mayo de 2021, de <https://codigofacilito.com/articulos/crear-pruebas-automatizadas-en-laravel-con-phpunit>

*Haciendo Pruebas Automatizadas en Laravel [PARTE I].* (s. f.). Laravel Tip. Recuperado 26 de mayo de 2021, de <https://www.laraveltip.com/haciendo-pruebas-automatizadas-en-laravel/>

*Haciendo Pruebas Automatizadas en Laravel [PARTE II] – Unit Test.* (s. f.). Laravel Tip. Recuperado 26 de mayo de 2021, de <https://www.laraveltip.com/haciendo-pruebas-automatizadas-en-laravel-unit-test/>