

Continuando con las lecciones del [**Curso: Primeros Pasos con Laravel 5.\***](https://styde.net/curso-primeros-pasos-con-laravel-5/) en esta clase vamos a estudiar un poco más las rutas, específicamente las de tipo POST y realizaremos un sencillo ejemplo para probar que funcionan tanto a través de un formulario en el navegador, como a través de las pruebas automatizadas de Laravel.

Repositorio

Mira el código en GitHub: [**actual**](https://github.com/StydeNet/first-steps/tree/3), [**resultado**](https://github.com/StydeNet/first-steps/tree/4), [**comparación**](https://github.com/StydeNet/first-steps/compare/3...4).  
Si no tienes acceso al repositorio, solicítalo a: [**admin@styde.net**](https://styde.net/rutas-para-formularios-en-laravel-metodo-post/)

Notas

* Un servidor web usa el protocolo HTTP para sus comunicaciones, que recibe las peticiones que pueden ser de distintos métodos como pueden ser: GET, POST, PUT, DELETE, PATCH.
* Las rutas de tipo GET son aquellas que solicitan un recurso a través de una ruta al servidor web, cuyo método de petición es GET. En Laravel se pueden expresar así en el archivo /app/Http/routes.php:

Route::get('notes/create', function () {

return '[Create notes]';

});

* Las rutas de tipo POST son aquellas que envían datos, usualmente por medio de formularios, al servidor web y su método de petición es POST. Estas rutas por su parte, en Laravel se pueden escribir:

Route::post('notes', function () {

return 'Creating a note';

});

El formulario creado fue:

<form method="POST">

<input type="hidden" name="\_token" value="{{ csrf\_token() }}"></input>

<textarea></textarea>

<button type="submit">Create note</button>

</form>

* En Laravel   {{ csrf\_field() }}  es equivalente a   <input type="hidden" name="\_token" value="{{ csrf\_token() }}"></input> que es usado para proteger a los formularios de la aplicación de ataques de tipo CSRF (del inglés Cross-site request forgery)
* Si en vez de trabajar con formularios se está trabajando con algún framework de JavaScript como AngularJS, Vue.js, entre otros, donde la aplicación se conecta al servidor por peticiones POST si necesidad de usar un formulario en HTML usamos las pruebas con PHPUnit para comprobar que están funcionando correctamente.
* Para ejecutar las pruebas unitarias   vendor/bin/phpunit .

Es todo por ahora, esperamos que la lección haya sido de tu agrado y que practiques lo aprendido, para cualquier duda escríbenos en la sección de comentarios y por su puesto, no olvides compartir esta lección en las redes sociales.

Material relacionado

* Lección anterior: [**Cómo crear rutas en Laravel 5.\***](https://styde.net/como-crear-rutas-en-laravel-5/)
* [**API REST con Laravel 5.1 – Primeros pasos**](https://styde.net/api-rest-con-laravel-5-1-primeros-pasos/)
* [**PHP Traits en Laravel 5.1**](https://styde.net/php-traits-en-laravel-5-1/)