




27 enero, 2018 [Elivar Largo](#)

CÓMO CREAR UN CRUD EN LARAVEL 5.5 DESDE CERO

Hola que tal, bienvenido a este nuevo artículo en donde aprenderás **cómo crear un crud en Laravel 5.5 desde cero**, este es un artículo que por así decirlo es una continuación del artículo [Cómo instalar y configurar Laravel 5.5](#) y lo que vamos a hacer es básicamente crear las opciones de crear, leer, actualizar y eliminar que son las opciones básicas que tiene una tabla, para esto usaremos Laravel que es un Framework para PHP que como se verá a continuación nos ayuda y nos abstrae la creación de un montón de código simplemente con usar comandos vía consola.

Lista Libros							Añadir Libro
Nombre	Resumen	No. Páginas	Edicion	Autor	Precio	Editar	Eliminar
qqqqqqqqqq	eeeeeeeeeee	23	3	fffffffffff	8.00		
sasasa	fffffffffff	160	2	fffffffffff	5.00		
Libro de PHP	qqqq	160	2	eeeeee	5.00		

« 1 2 »

INSTALACIÓN DE LARAVEL VÍA COMPOSER

Previamente debemos tener instalado XAMPP y en la ubicación **C:\xampp\htdocs** abrimos una ventana de comandos e instalamos usando el siguientes comando:

```
1 composer create-project laravel/laravel=5.5 crud
```

Como te darás cuenta dependiendo la velocidad de tu conexión a internet esto puede o no tardar mucho.

Bien, ahora que ya tenemos creado el proyecto lo primero que vamos a hacer es cambiar la configuración para la base de datos, usuario y clave, si aún no lo has hecho puedes ver el artículo anterior [Cómo instalar y configurar Laravel 5.5](#) donde explico como hacerlo. El nombre de la base de datos le llamaremos del mismo nombre del proyecto **crudlaravel**, posteriormente deberás crear la base de datos en MySQL.



CREACIÓN DE MIGRACIONES

El tema de migraciones en Laravel tiene que ver con el diseño de tablas de nuestra base de datos, y aunque podemos directamente crear nuestra tabla en MySQL crear el controlador y el modelo, lo vamos hacer usando migraciones, **cual es la diferencia?** Pues la diferencia es que usando migraciones diseñas las tablas como si de modelos se tratarán y luego las puedes generar a MySQL con un sólo comando, además puedes llevar la cronología de la creación de tus tablas y si algo salió mal, por ejemplo te olvidaste de crear un campo, fácilmente haces un rollback y deshaces los cambios. Pero bueno para que se entienda mejor vamos al ejemplo.



Udemy Categorías Buscar cursos

Desarrollo Negocios Informática y software Productividad en la oficina Desarrollo personal

Dominando Laravel - De principiante a experto

Aprende a crear aplicaciones robustas y escalables con el framework más popular de PHP, Laravel

★★★★★ 4,5 (549 valoraciones) 2.239 estudiantes inscritos

Creado por Jorge García Fecha de la última actualización: 2/2018 Español

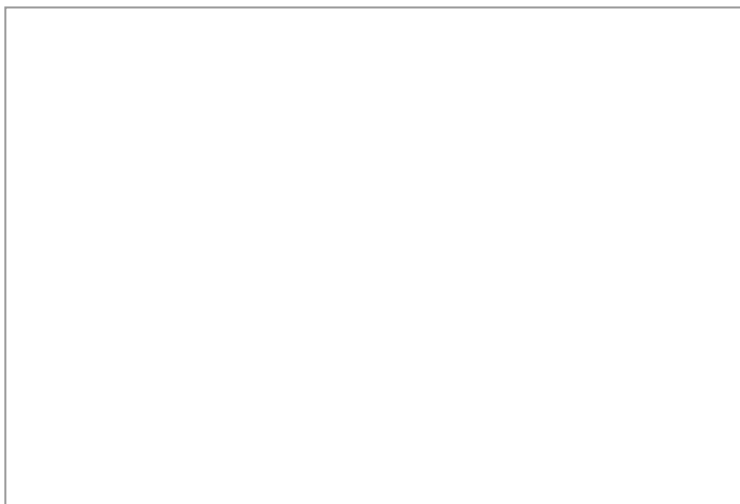
La tabla que vamos a crear para el ejemplo se va llamar Libros y contendrá los campos: nombre, resumen, número de páginas, edición, autor y precio.

Para crear la tabla creamos un archivo de migración, para esto vamos a la ventana de comandos abierta anteriormente y usamos el siguiente comando:

```
1 php artisan make:migration create_libros_table
```

Si te das cuenta ahora se creó un nuevo archivo dentro de la carpeta database->migrations 2018_01_26_035203_create_libros_table.php.

Es una clase que hereda de la clase Migration, en esta clase definimos los campos que va contener la tabla Libros, en esta clase hay dos métodos **down** que se llama cuando ejecutamos un rollback y **up** que es donde crearemos los campos para nuestra tabla, el archivo debe quedar como se muestra a continuación:



```

1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateLibrosTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('libros', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('nombre');
19             $table->string('resumen');
20             $table->integer('npagina');
21             $table->integer('edicion');
22             $table->string('autor');
23             $table->decimal('precio', 5, 2);
24             $table->timestamps();
25         });
26     }
27
28     /**
29      * Reverse the migrations.
30      *
31      * @return void
32      */
33     public function down()
34     {
35         Schema::dropIfExists('libros');
36     }
37 }

```

Ahora lo que nos queda es ejecutar la migración, esto para que se cree nuestra tabla **libros**, para esto ejecutamos el siguiente comando:

```
1 php artisan migrate
```

Lo que nos queda es revisar la tabla libros, para esto puedes usar phpMyadmin o si usas MySQL Workbench puedes ver que efectivamente se creó la tabla libros junto con otras adicionales.

Bien, como te mencionaba en ciertos casos pueda que quieras añadir algún campo a alguna tabla que ya está creada, simplemente puedes hacer un rollback, lo que hace en este caso es eliminar las tablas creadas con las migraciones, la base de datos queda vacía sin ninguna tabla, puedes probar el siguiente comando y verás que pasa:

```
1 php artisan migrate:rollback
```

Si de nuevo revisas la base de datos te darás cuenta que la base de datos volvió a su estado inicial y como este comando era sólo para probar, debes volver a generar el comando migrate para crear la tabla libros.

CREAR EL MODELO

Ahora que ya tenemos la tabla creada, necesitamos un modelo que mapee los campos a la tabla libros y para esto usamos el comando:

```
1 php artisan make:model Libro
```

Si vas a la carpeta app del proyecto vas a encontrar un nuevo archivo llamado Libro.php, como te darás cuenta es una clase vacía y en teoría deberíamos crear las propiedades, los setter y getter, pero con la ayuda de Laravel sólo creamos un array \$fillable y le pasamos los campos que queremos llenar, así de fácil como se muestra a continuación:

```

1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Libro extends Model
8  {
9      //
10     protected $fillable = ['nombre', 'resumen']
11 }
```

CREAR EL CONTROLLER

Laravel es un Framework que usa el patrón MVC (Modelo, Vista, Controlador), por lo que ahora tenemos que crear el controlador con los métodos index, show, update, delete.

Como te habrás dado cuenta hasta ahora hemos generado todo básicamente a través de comandos y bueno esta vez no es la excepción, así que para crear el controlador usamos el siguiente comando:

```
1  php artisan make:controller LibroController --re
```

Si ahora vas a la carpeta app/Http/Controllers puedes ver que se generó un nuevo archivo VideoController.php y como por arte de magia se crearon todos los métodos que necesitamos, a continuación dejo el código para cada uno de los métodos, que como te darás cuenta es corto y sencillo, puesto que el framework prácticamente hace todo por nosotros.

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Libro;
7
8  class LibroController extends Controller
9  {
10     /**
11      * Display a listing of the resource.
12      *
13      * @return \Illuminate\Http\Response
14      */
15     public function index()
16     {
17         //
18         $libros=Libro::orderBy('id','DESC')->paginate(10);
19         return view('Libro.index',compact('libros'));
20     }
21
22     /**
23      * Show the form for creating a new resource.
24      *
25      * @return \Illuminate\Http\Response
26      */
27     public function create()
28     {
29         //
30         return view('Libro.create');
```

```

31     }
32
33     /**
34      * Store a newly created resource in storage.
35      *
36      * @param \Illuminate\Http\Request $request
37      * @return \Illuminate\Http\Response
38      */
39     public function store(Request $request)
40     {
41         //
42         $this->validate($request, [ 'nombre' => 'required' ]);
43         Libro::create($request->all());
44         return redirect()->route('libro.index');
45     }
46
47     /**
48      * Display the specified resource.
49      *
50      * @param int $id
51      * @return \Illuminate\Http\Response
52      */
53     public function show($id)
54     {
55         $libros=Libro::find($id);
56         return view('libro.show',compact('libro'));
57     }
58
59     /**
60      * Show the form for editing the specified resource.
61      *
62      * @param int $id
63      * @return \Illuminate\Http\Response
64      */
65     public function edit($id)
66     {
67         //
68         $libro=Libro::find($id);
69         return view('libro.edit',compact('libro'));
70     }
71
72     /**
73      * Update the specified resource in storage.
74      *
75      * @param \Illuminate\Http\Request $request
76      * @param int $id
77      * @return \Illuminate\Http\Response
78      */
79     public function update(Request $request, $id)
80     {
81         //
82         $this->validate($request, [ 'nombre' => 'required' ]);
83         $libro=Libro::find($id)->update($request->all());
84         return redirect()->route('libro.index');
85     }
86
87     /**
88      * Remove the specified resource from storage.
89      *
90      * @param int $id
91      * @return \Illuminate\Http\Response
92      */
93     public function destroy($id)
94     {
95         //
96         $libro=Libro::find($id)->delete();
97         return redirect()->route('libro.index');
98     }
99

```

```
100 }
```

CREAR LAS RUTAS

Una ruta en Laravel indica a que método del controlador debe direccionar una petición por ejemplo cuando queramos crear un nuevo libro, editar etc., para esto vamos a la carpeta routes y al archivo web.php el cual debe quedar como sigue:

```
1 <?php
2
3 /*
4 |-----
5 | Web Routes
6 |-----
7 |
8 | Here is where you can register web routes for
9 | routes are loaded by the RouteServiceProvider
10 | contains the "web" middleware group. Now crea
11 |
12 */
13
14 Route::resource('libro', 'LibroController');
```

CREAR LAS VISTAS

Finalmente lo que vamos hacer es crear las vistas, para mostrar, editar y eliminar libros, para esto vamos a /resources/views/ y creamos una nueva carpeta llamada layouts y dentro de esta carpeta creamos el archivo layout.blade.php, este archivo tiene la plantilla del proyecto, es importante mencionar que Laravel usa el motor de plantillas blade que junto con HTML permiten crear vistas más simples y limpias.

 Categorías

Buscar cursos 

DesarrolloNegociosInformática y softwareProductividad en la oficinaDesarrollo personal

Crea tu API RESTful con Laravel (5.0 a 5.2)

Desarrolla una API RESTful en PHP con Laravel y desde cero. Domina PHP y RESTful con Laravel

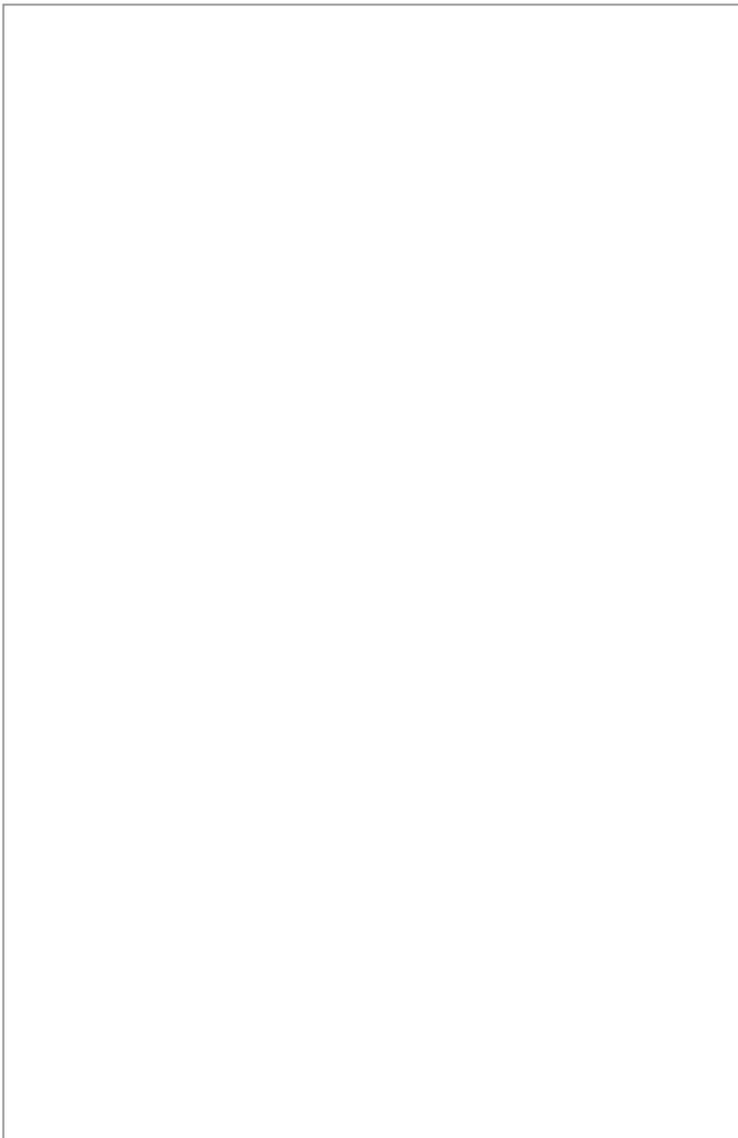
★★★★★ 4,3 (191 valoraciones) 797 estudiantes inscritos

Creado por JuanD MeGon Fecha de la última actualización: 2/2017  Español

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport"
6         content="width=device-width, initial-scale=
7     <link href="https://maxcdn.bootstrapcdn.com
8     <script src="https://maxcdn.bootstrapcdn.co
9 </head>
10 <body>
11
12     <div class="container-fluid" style="margin-
13
14         @yield('content')
15     </div>
16     <style type="text/css">
17         .table {
18             border-top: 2px solid #ccc;
19
20         }
21 </style>
22 </body>
23 </html>
```

ARCHIVO INDEX.BLADE.PHP

Posteriormente en la ruta /resources/views/ creamos una nueva carpeta llamada Libro que va contener las vistas (index, create, show, edit). Dentro de esta carpeta creamos el archivo index.blade.php como se muestra a continuación:




```

1  @extends('layouts.layout')
2  @section('content')
3  <div class="row">
4      <section class="content">
5          <div class="col-md-8 col-md-offset-2">
6              <div class="panel panel-default">
7                  <div class="panel-body">
8                      <div class="pull-left"><h3>Lista Libr
9                      <div class="pull-right">
10                         <div class="btn-group">
11                             <a href="{{ route('libro.create')
12                             </div>
13                         </div>
14                     <div class="table-container">
15                         <table id="mytable" class="table ta
16                         <thead>
17                             <th>Nombre</th>
18                             <th>Resumen</th>
19                             <th>No. Páginas</th>
20                             <th>Edicion</th>
21                             <th>Autor</th>
22                             <th>Precio</th>
23                             <th>Editar</th>
24                             <th>Eliminar</th>
25                         </thead>
26                         <tbody>
27                             @if($libros->count())
28                             @foreach($libros as $libro)
29                                 <tr>
30                                     <td>{{ $libro->nombre }}</td>
31                                     <td>{{ $libro->resumen }}</td>
32                                     <td>{{ $libro->npagina }}</td>
33                                     <td>{{ $libro->edicion }}</td>
34                                     <td>{{ $libro->autor }}</td>
35                                     <td>{{ $libro->precio }}</td>
36                                     <td><a class="btn btn-primary b
37                                     <td>
38                                         <form action="{{ action('Libro
39                                         {{ csrf_field() }}
40                                         <input name="_method" type="
41
42                                         <button class="btn btn-dange
43                                     </td>
44                                 </tr>
45                             @endforeach
46                             @else
47                                 <tr>
48                                     <td colspan="8">No hay registro
49                                 </tr>
50                             @endif
51                         </tbody>
52                     </table>
53                 </div>
54             </div>
55         </div>
56         {{ $libros->links() }}
57     </div>
58 </div>
59 </section>
60
61 @endsection

```

ARCHIVO CREATE.BLADE.PHP

```

1  @extends('layouts.layout')
2  @section('content')
3  <div class="row">
4      <section class="content">

```



```

5      <div class="col-md-8 col-md-offset-2">
6          @if (count($errors) > 0)
7              <div class="alert alert-danger">
8                  <strong>Error!</strong> Revise
9                  <ul>
10                     @foreach ($errors->all() as
11                         <li>{{ $error }}</li>
12                     @endforeach
13                 </ul>
14             </div>
15         @endif
16         @if(Session::has('success'))
17             <div class="alert alert-info">
18                 {{Session::get('success')}}
19             </div>
20         @endif
21
22         <div class="panel panel-default">
23             <div class="panel-heading">
24                 <h3 class="panel-title">Nue
25             </div>
26             <div class="panel-body">
27                 <div class="table-container
28                     <form method="POST" act
29                         {{ csrf_field() }}
30                     <div class="row">
31                         <div class="col
32                             <div class=
33                                 <input
34                             </div>
35                         </div>
36                         <div class="col
37                             <div class=
38                                 <input
39                             </div>
40                         </div>
41                     </div>
42
43                     <div class="form-gr
44                         <textarea name=
45                     </div>
46                     <div class="row">
47                         <div class="col
48                             <div class=
49                                 <input
50                             </div>
51                         </div>
52                         <div class="col
53                             <div class=
54                                 <input
55                             </div>
56                         </div>
57                     </div>
58                     <div class="form-gr
59                         <textarea name=
60                     </div>
61                     <div class="row">
62
63                         <div class="col
64                             <input type
65                             <a href="{{
66                         </div>
67
68                     </div>
69                 </form>
70             </div>
71         </div>
72
73     </div>

```

```

74         </div>
75     </section>
76 @endsection

```

ARCHIVO EDIT.BLADE.PHP

```

1  @extends('layouts.layout')
2  @section('content')
3  <div class="row">
4      <section class="content">
5          <div class="col-md-8 col-md-offset-2">
6              @if (count($errors) > 0)
7                  <div class="alert alert-danger">
8                      <strong>Error!</strong> Revise
9                      <ul>
10                         @foreach ($errors->all() as
11                             <li>{{ $error }}</li>
12                         @endforeach
13                     </ul>
14                 </div>
15             @endif
16             @if(Session::has('success'))
17                 <div class="alert alert-info">
18                     {{Session::get('success')}}
19                 </div>
20             @endif
21
22             <div class="panel panel-default">
23                 <div class="panel-heading">
24                     <h3 class="panel-title">Nue
25                 </div>
26                 <div class="panel-body">
27                     <div class="table-container
28                         <form method="POST" act
29                             {{ csrf_field() }}
30                         <input name="_metho
31                         <div class="row">
32                             <div class="col
33                                 <div class=
34                                     <input
35                                     </div>
36                             </div>
37                             <div class="col
38                                 <div class=
39                                     <input
40                                     </div>
41                             </div>
42                         </div>
43
44                         <div class="form-gr
45                             <textarea name=
46                         </div>
47                         <div class="row">
48                             <div class="col
49                                 <div class=
50                                     <input
51                                     </div>
52                             </div>
53                             <div class="col
54                                 <div class=
55                                     <input
56                                     </div>
57                             </div>
58                         </div>
59                         <div class="form-gr
60                             <textarea name=
61                         </div>
62                     <div class="row">

```

```
63
64         <div class="col
65             <input type
66             <a href="{{
67         </div>
68
69     </div>
70 </form>
71 </div>
72 </div>
73
74 </div>
75 </div>
76 </section>
77 @endsection
```

Y bien como puedes ver que en cuestión de minutos puedes crear un CRUD con Laravel, a continuación dejo el enlace desde el repositorio de Github para que lo descargues [Descarga Crud en Laravel desde Github](#). También puedes continuar revisando **Cómo crear un servicio Api REST y consumirlo desde Android.**

Nota: Para ejecutar el proyecto debes poner en tu navegador localhost/crudlaravel/public/libro y tener levantado el servidor (XAMPP, WAMP etc.)

Nos vemos en un próximo artículo hasta pronto!!



Programación Web Full Stack

Suscríbete ahora y recibe los mejores contenidos sobre Programación Web en tu correo.

Nombre

Email

SUSCRIBIRME

Tus datos estarán protegidos y 100% libre de Spam

ELIVAR LARGO



Full Stack Developer, JavaScript, PHP, Java, Spring, Laravel, Vuejs, Blogger, aprendiendo y compartiendo conocimientos. Cursos de Programación Web en:
<https://programacionfullstack.com/>