



# **UNIVERSIDAD POLITÉCNICA DE TULANCINGO**

**ORTIZ OSORIO CRISTIAN URIEL  
FRANCISCO MELO MARIO ETZAEI**

**U  
P  
T**

**Ingeniería en Sistemas Computacionales**

**Manual de creación de un servicio  
Restful en PHP**



LARAVEL. Es un framework de código abierto de php para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

Para poder crear un proyecto en laravel debemos contar con composer el cual permite descargar las librerías y paquetes necesarios para el desarrollo de un

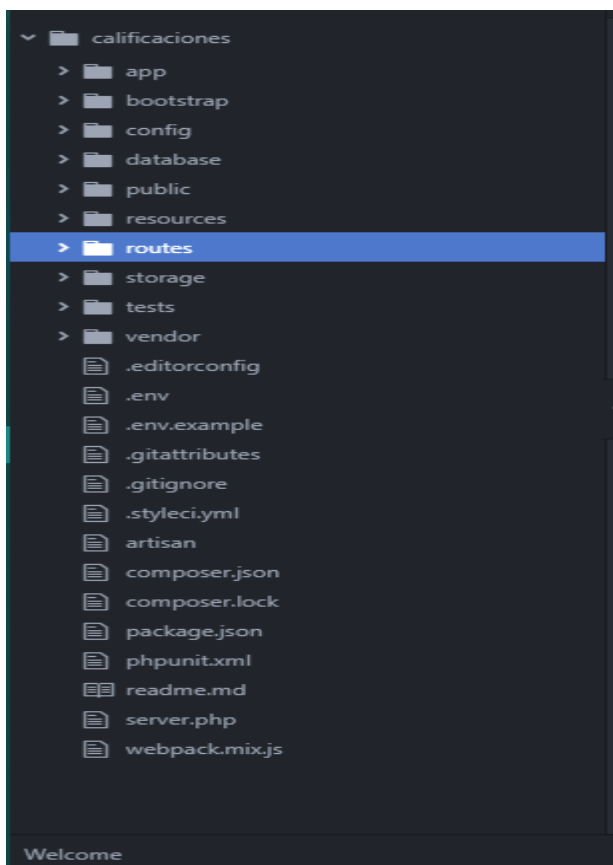
proyecto de laravel, composer está disponible para su descarga desde su página oficial, dejamos un pequeño tutorial para su instalación:

<https://styde.net/instalacion-de-composer-y-laravel-en-windows/>

Después de su instalación vamos a proceder a crear un proyecto con composer en la versión más actualizada de laravel (6.0) con el siguiente comando por consola dentro de la carpeta que queremos que se cree:

```
composer create-project laravel/laravel nameproyect
```

Después de esto composer empezara a descargar paquetes y librerías necesarias para laravel automáticamente y generara el proyecto. Al abrir nos encontramos con los directorios que en principio no sabremos como funcionana en principio los directorios y archivos más importantes que debemos aprender son los siguientes



Los archivos mas importantes son

- .env
- Artisan

El primer archivo artisan es acerca de la configuracion de todo el proyecto, en muchas de las ocasiones el unico cambio que debemos ejercer dentro del archivo es el cambio de la base de datos o de la conexión a la base de datos con una base de datos ya creada.

CarrerController.php	Telemetry Consent	Welcome	2019_11_
<pre>1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:m12wu5GIZNCFHB7OQUjGC9a0os9xGBySrv9cEHqjxjw= 4 APP_DEBUG=true 5 APP_URL=http://localhost 6 7 LOG_CHANNEL=stack 8 9 DB_CONNECTION=mysql 10 DB_HOST=127.0.0.1 11 DB_PORT=3306 12 DB_DATABASE=sistemacon 13 DB_USERNAME=root 14 DB_PASSWORD=Esnafer19 15 16 BROADCAST_DRIVER=log 17 CACHE_DRIVER=file 18 QUEUE_CONNECTION=sync 19 SESSION_DRIVER=file 20 SESSION_LIFETIME=120 21 22 REDIS_HOST=127.0.0.1 23 REDIS_PASSWORD=null 24 REDIS_PORT=6379 25 26 MAIL_DRIVER=smtp 27 MAIL_HOST=smtp.mailtrap.io 28 MAIL_PORT=2525 29 MAIL_USERNAME=null 30 MAIL_PASSWORD=null 31 MAIL_ENCRYPTION=null 32 33 AWS_ACCESS_KEY_ID= 34 AWS_SECRET_ACCESS_KEY= 35 AWS_DEFAULT_REGION=us-east-1</pre>			

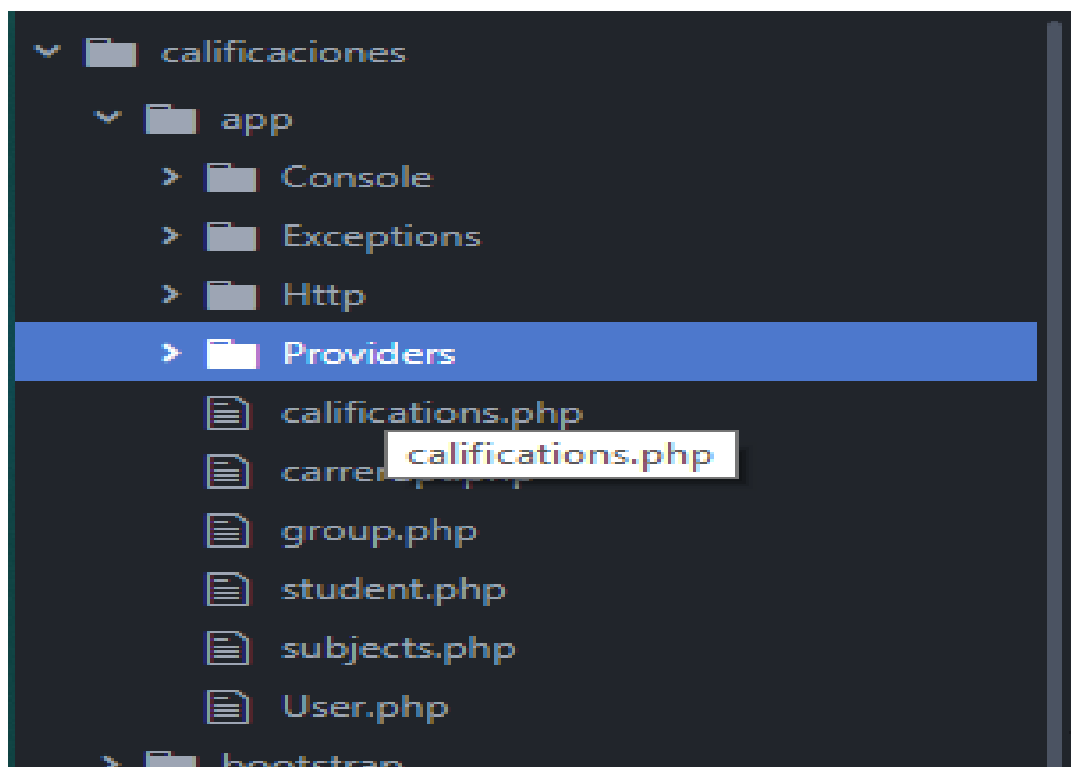
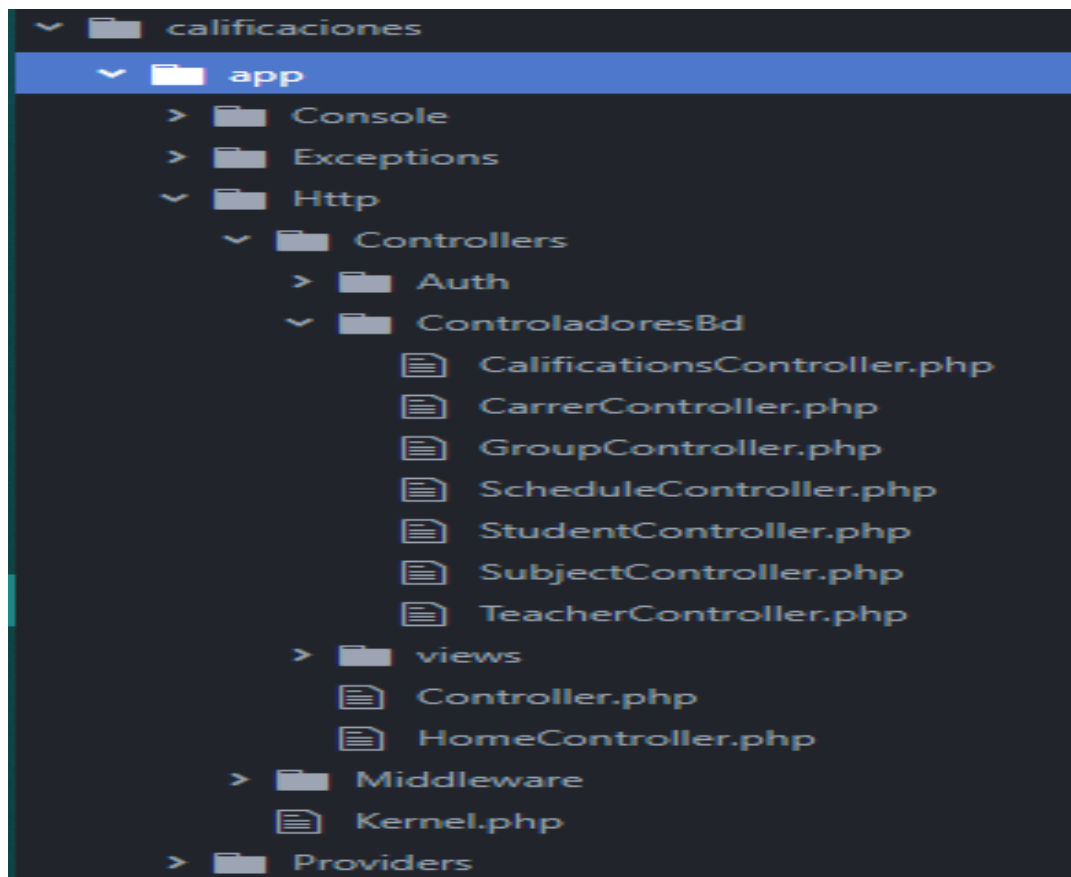
El siguiente archivo artisan contiene las propiedades del comando fundamental de laravel 'artisan' para crear o modificar dentro de la terminal

```
CarrerController.php | Telemetry Consent | Welcome | 2019_11_27_050431... | tableCarrer.
1  #!/usr/bin/env php
2  <?php
3
4  define('LARAVEL_START', microtime(true));
5
6  /*
7  |-----
8  | Register The Auto Loader
9  |-----
10 |
11 | Composer provides a convenient, automatically generated class loader
12 | for our application. We just need to utilize it! We'll require it
13 | into the script here so that we do not have to worry about the
14 | loading of any our classes "manually". Feels great to relax.
15 |
16 |*/
17
18 require __DIR__.'/vendor/autoload.php';
19
20 $app = require_once __DIR__.'/bootstrap/app.php';
21
22 /*
23 |-----
24 | Run The Artisan Application
25 |-----
26 |
27 | When we run the console application, the current CLI command will be
28 | executed in this console and the response sent back to a terminal
29 | or another output device for the developers. Here goes nothing!
30 |
31 |*/
32
33 $kernel = $app->make(Illuminate\Contracts\Console\Kernel::class);
34
35 $status = $kernel->handle(
```

Las carpetas mas importantes son:

- App
  - Http
    - Controllers
- Database
- Resources
- Routes

Laravel es un framework que trabaja estrictamente con el modelo MVC para poder entender un poco dentro de la carpeta App-Http se guardan los controladores y todas las funciones que estan implementadas para el desarrollo del proyecto, de la misma manera App contiene los modelos de cada tabla.



La carpeta es Database en donde se guardan los scripts de la base de datos, gracias al potente framework de Laravel podemos crear tablas a partir de una base de datos existentes dejando de lado el lenguaje sql y partiendo del lenguaje php para el desarrollo de las tablas de la siguiente forma

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateSubjectsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('subjects', function (Blueprint $table) {
17             $table->string('nameSubject',65)->primary();
18             $table->string('nameCarrer',65);
19             $table->integer('quarters');
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('subjects');
31     }
32 }
33
```

ts\_table.php 24:31

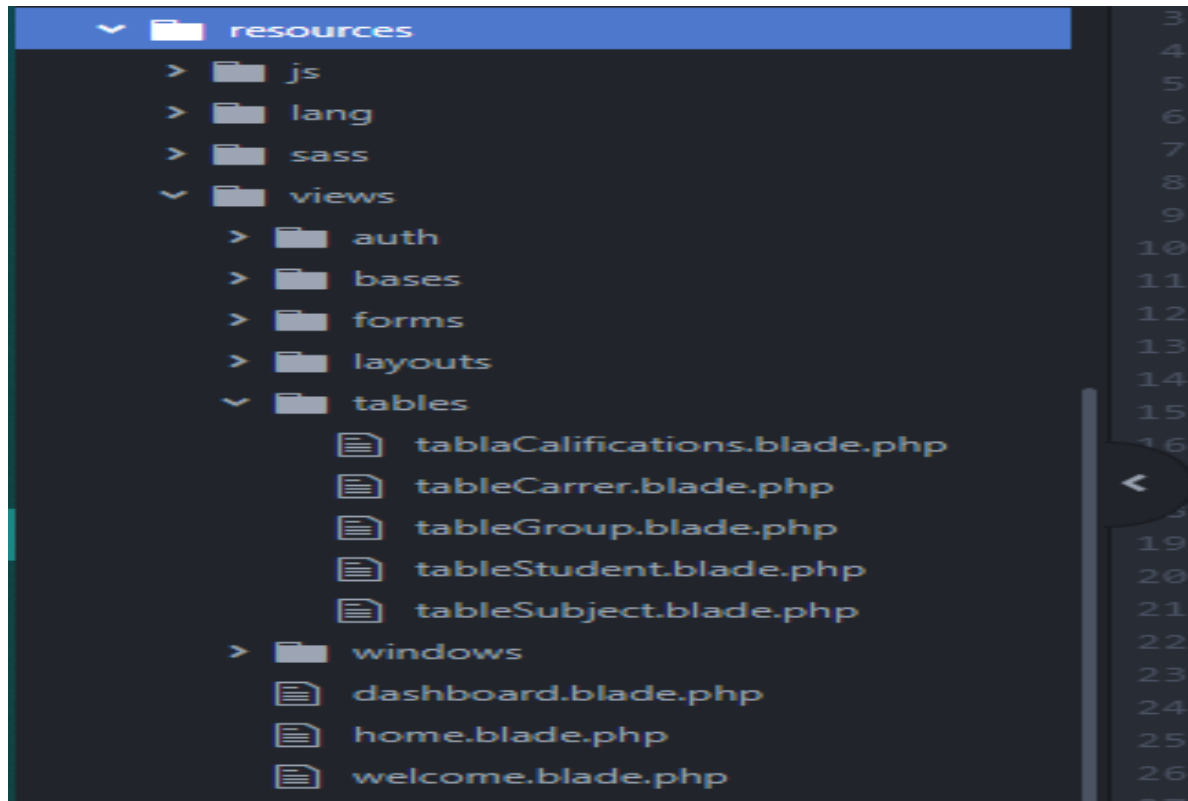
Para poder crear la tabla que se muestra en este script debemos ejecutar el siguiente comando en consola dentro del directorio del proyecto:

```
php artisan migrate
```

Este comando migra los scripts a tablas dentro de la base de datos. En caso de que queramos modificar las tablas debemos de modificar el script y ejecutar el siguiente comando

```
php artisan migrate refresh
```

La carpeta Resources contiene toda la vista del proyectos, archivos como js, css, html, php y Blade.php que es generado gracias al framework, la diferencia entre archivo es su maleabilidad de html con php.

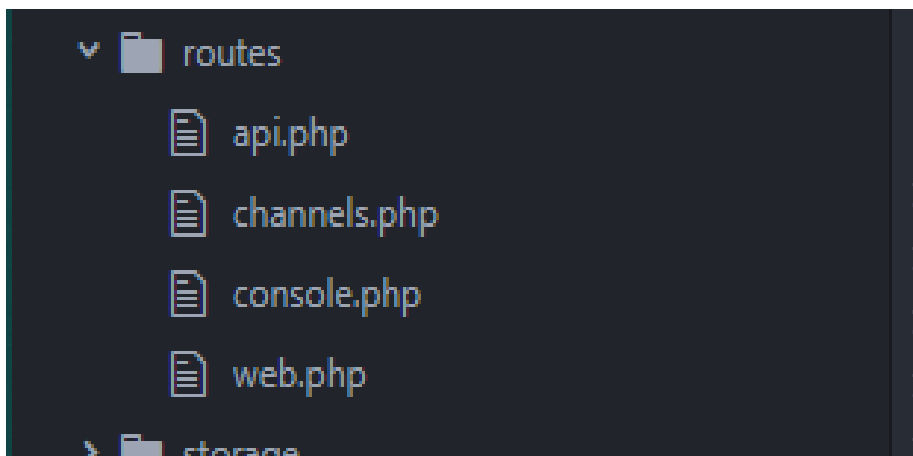


```

1 extends('layouts.app')
2 @section('content')
3 <div class="container">
4     <div class="container col-md-8 col-md-offset-2">
5         <div class="panel panel-default">
6             <div class="panel-heading">
7                 <h2>Alumnos</h2>
8             </div>
9             @if ($student->isEmpty())
10                 <div>No hay Registros</div>
11             @else
12                 <table class="table">
13                     <thead>
14                         <tr>
15                             <th>Número de cuenta</th>
16                             <th>Nombre de estudiante</th>
17                             <th>Apellido paterno</th>
18                             <th>Apellido materno</th>
19                             <th>Nombre carrera</th>
20                             <th>Cuatrimestre</th>
21                             <th>grupo</th>
22                             <th>Modificar</th>
23                             <th>Eliminar</th>
24                         </tr>
25                     </thead>
26                     <tbody>
27                         @foreach($student as $student)
28                             <tr>
29                                 <td>{!! $student->numberAccount !!}</td>
30                                 <td>{!! $student->nameStudent !!}</td>
31                                 <td>{!! $student->lastNameP !!}</td>
32                                 <td>{!! $student->lastNameM !!}</td>
33                                 <td>{!! $student->nameCarrer !!}</td>
34                                 <td>{!! $student->quarters !!}</td>
35

```

Dentro de la carpeta routes es donde toda la magia empieza ya que permite el enrutamiento de todo el proyecto y esto es gracias a diferencia con otros frameworks que yo considero el mejor.



La carpeta contiene 4 archivos, solo se pueden configurar como archivos de ruta 2, los cuales son api.php y web.php. Existen 2 formas de crear rutas:



La primera consta de agregar una función dentro de la ruta sin necesidad de usar un controlador y retornando un valor.

```
13
14 Route::get('/', function () {
15     return view('windows.index');
16     //return view('forms.acceso');
17     //return view('welcome');
18 });
19
```

La segunda consta de una ruta que va hacia un controlador específico para realizar cierta función.

```
20 Auth::routes();
21
22 Route::get('vistaAcceso', 'views\vistas@inicioSesion')->name('vistaAcceso');
23 Route::get('registerStudent', 'views\vistas@registerStudent')->name('registerStudent');
```

Para poder hacer que el proyecto consuma recursos restful es agregando el siguiente prefijo a la ruta

```
Route::get('carrer/{idCarrer}', 'ControladoresBd\CarrerController@getCarrer')->name('getCarrer');
Route::put('carrer/{idCarrer}', 'ControladoresBd\CarrerController@editCarrer')->name('editCarrer');
Route::delete('carrer/{idCarrer}', 'ControladoresBd\CarrerController@deleteCarrer')->name('deleteCarrer');
```

Los prefijos put para modificar, delete para eliminar dirigida a un controlador con una función específica.

Los navegadores solo permiten las peticiones por http post y get, como podemos hacer que un formulario mande peticiones put y delete a través de http, el siguiente formulario muestra un ejemplo en laravel de cómo llevarlo a cabo.

```

<div class="card-header">Grupos</div>

<div class="card-body">
    <form method="POST" action="{{ route('ediGroup',$group->groupA) }}">
        {{ csrf_field() }}
        {{ method_field('PUT') }}

```

```

<td>
    <form action="{{ route('delCarrer',$carrer->idCarrer) }}" method="POST">
        @csrf
        @method('delete')
        <button type="submit" class="btn btn-danger">Eliminar</button>
    </form>
</td>

```

El formulario tiene que llevar el método de POST, pero en el formulario se genera un input hidden con el method deseadado, en este caso los archivos generados con Blade permiten generar componentes html, gracias a esto el método llevara un put o delete dependiendo el caso.

Para poder crear controladores necesitamos el comando:

php artisan make:controller nombre del controlador

Generamos un controlador con una estructura de código php el cual puede tener la siguiente estructura

```

1  <?php
2
3  namespace App\Http\Controllers\ControladoresBd;
4
5  use Illuminate\Http\Request;
6  use App\Http\Controllers\Controller;
7  use App\carreerupt;
8
9  class CarrerController extends Controller
10 {
11
12     public function getAllCarrer(){
13         $carrer=carreerupt::all();
14         return view('tables.tableCarrer')->with('carrer',$carrer);
15     }
16     //Agregar
17     public function addCarrer(Request $request){
18         $carrer=carreerupt::create($request->all());
19         $carrer=carreerupt::all();
20         return view('tables.tableCarrer')->with('carrer',$carrer);
21     }
22
23     public function obtenerCarr($idCarrer){
24         $c=carreerupt::select('nameCarrer')->where('idCarrer','=',$idCarrer)->get();
25         return $c;
26     }
27     //obtener un solo registro
28     public function getCarrer($idCarrer){
29         $carrer=carreerupt::where('idCarrer',$idCarrer)->first();
30         return view('forms.editCarrer')->with('carrer',$carrer);
31     }
32
33     //Modificar
34     public function editCarrer($idCarrer,Request $request){
35

```

El ejemplo recién mostrado es un controlador de base de datos el cuál ocupa un modelo que se ha generado y de él se obtienen las consultas. Para generar un modelo dentro accedemos al directorio del proyecto en la terminal escribimos el comando

php artisan make:model nombre del modelo

Genera un archivo con el nombre dado y su contenido es el siguiente

```

1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class subjects extends Model
8  {
9
10 }

```

Y dentro pondremos la tabla a la que va a hacer referencia y sus respectivos campos.

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class subjects extends Model
8  {
9      protected $table='subjects';
10     public $timestamps = false;
11     protected $fillable=array('nameSubject','nameCarrer','quarters');
12 }
13
```

## Conclusiones

Podemos decir que laravel es un framework muy útil cuando se trata de usar servicios restful, es bastante manejable y fácil de entender, tiene cosas muy interesantes, sobre todo las rutas.

El servicio restful ayuda a los programadores a tener un mejor manejo de los datos y buen retorno con los archivos json.