



# GLOSARIO LARAVEL

Carlos Peñalver Alonso

2DAW



## Contenido

1.	Instalación.....	2
	Primeros pasos.....	2
	Migraciones .....	2
	Modelos.....	3
2.	Introducción a vista y rutas .....	3
	Vistas .....	4
	Rutas.....	4
3.	Inserción de datos .....	5
4.	Listar productos .....	5
5.	Eliminar productos .....	6
6.	Modificar productos .....	6
7.	Refactorizar las vistas Blade .....	7
8.	Protección de la aplicación con autenticación.....	7
9.	Controladores .....	9

# 1. Instalación

## Primeros pasos

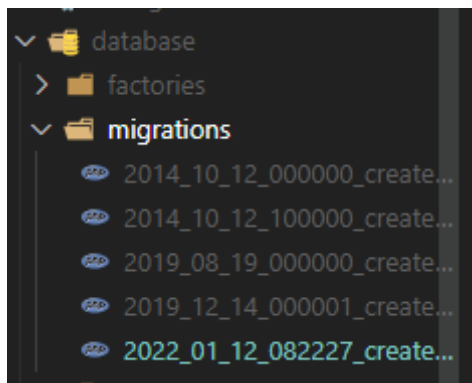
1. Creamos una carpeta donde queramos el proyecto (abriendo un PowerShell) y la nombramos.
2. Descargamos laravel con el siguiente comando "composer create-project laravel/laravel"
3. Abrimos el proyecto en code.
4. Creamos una bbdd nueva en phpmyadmin.
5. Configuramos el archivo '.env'
6. Buscamos las líneas:
  - a. DB\_DATABASE= (aquí añadimos el nombre de la bbdd creada)
  - b. DB\_USERNAME=root
  - c. DB\_PASSWORD=

## Migraciones

Un ejemplo de migración sería: php artisan make:migration create\_products\_table. Su estructura: creamos la migración (make:migration), ponemos create, luego se le pone el nombre de la tabla, en este caso products, y por último lo que es, en nuestro caso una tabla.

Una vez hecho esto debería aparecernos este archivo:

Una vez aquí, buscamos los campos que se van a crear en la tabla (function up) y los editamos:



```
$table->increments('id');  
$table->text('description');  
$table->integer('price');  
$table->timestamps();
```

Para evitar fallos necesitaremos una configuración:

Dentro de la carpeta config>database buscamos la línea que pone 'engine' y cambiamos por 'engine' => 'InnoDB'

Una vez todo configuramos, abrimos PowerShell y ejecutamos la migración: "php artisan migrate".

```
PS C:\wamp64\www\Tienda\laravel> php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (35.83ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (52.23ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (38.83ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (43.10ms)
Migrating: 2022_01_12_082227_create_products_table
Migrated: 2022_01_12_082227_create_products_table (19.59ms)
```

## Modelos

Se debe crear un modelo por tabla y cada modelo tiene que tener el mismo nombre que la tabla pero con la primera letra en mayúscula y en singular: “php artisan make:model Product”.

Todos los modelos se alojan en app>Models.

## 2. Introducción a vista y rutas

Hay 4 tipos de rutas: **GET** (conseguir datos), **POST** (guardar datos), **PUT** (actualizar recursos) y **DELETE** (eliminar recursos).

Las rutas se almacenan en archivos web.php. Por defecto aparece una ruta, que hay que eliminar.

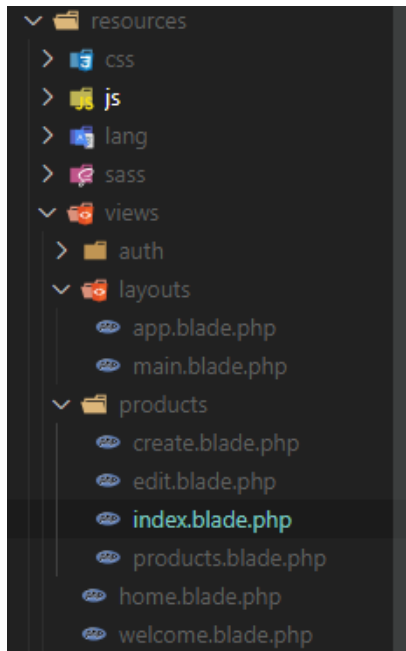
La nueva ruta a crear tiene dos parámetros, el primer parámetro es la url que queremos que ejecute esa ruta y el segundo la función que ejecutará esa ruta.

```
Por ejemplo: Route::get('/', function(){
    return 'Esta es nuestra URL raíz';
});
```

Para probarlo ir a <http://localhost/Tienda/laravel/public>.

## Vistas

Las vistas se alojan en esta carpeta:



Hay que nombrar cada vista de forma que sea:

*NombreDeLaPagina.blade.php*

Dentro se aloja el contenido html a visualizar.

```
('layouts.main')
('contenido')
ss="container"> <br>
class="row">
<div class="col-md-12">
  <div class="card">
    <div class="card-header">
      Listado de productos
      <a href="{{ route('products.create') }}" class="btn btn-success btn-sm float-right"> Crear
    </div>
    <div class="card-body">
      @if(session('info'))
      <div class="alert alert-success">
        {{session('info')}}
      </div>
      @endif
      <table class="table table-hover table-sm">
        <thead>
          <th>Descripcion</th>
          <th>Precio</th>
          <th>Accion</th>
        </thead>
        <tbody>
          @foreach($products as $producto)
          <tr>
            <td>
              {{ $producto->description }}
            </td>
            <td>
              {{ $producto->price }} €
            </td>
            <td>
              <a href="{{ route('products.edit', $producto->id) }}" class="btn btn-primary"> Editar
              <a href="javascript: document.getElementById('delete-{{ $producto->id }}').click()" class="btn btn-danger"> Borrar
            </td>
          </tr>
          <tr>
            <td colspan="3">
              <form action="{{ route('products.destroy', $producto->id) }}" id="delete-{{ $producto->id }}" method="delete" @csrf>
                <input type="submit" value="Borrar" />
              </form>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
</div>
```

## Rutas

Dentro de la página “web.php” se introducen todas las rutas al controller, de manera que se quede mas limpio.

```
use App\Models\Product;
use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;

Route::middleware('auth')->group(function () {
    Route::get('products', 'ProductController@index')->name('products.index');
    Route::get('products/create', 'ProductController@create')->name('products.create');
    Route::post('products', 'ProductController@store')->name('products.store');
    Route::delete('products/{id}', 'ProductController@destroy')->name('products.destroy');
    Route::get('products/{id}/edit', 'ProductController@edit')->name('products.edit');
    Route::put('products/{id}', 'ProductController@update')->name('products.update');
    Route::get('/activity', 'ProductController@updatedActivity');
});

Auth::routes();
```

Los botones de la pagina web se programan de la siguiente manera:

En la url del botón se añade esto:

```
<a href="{{ route('products.edit', $producto->id)}}" ...
```

El nombre de la función que tiene y la variable a la que se refiere. (En este caso el producto a editar).

### 3. Inserción de datos

En el action del formulario tenemos que añadir obligatorio la ruta y después un @csrf, que es el token de seguridad de la aplicación:

```
<form action="{{ route('products.store')}}"  
@csrf
```

Para poder insertar los datos hay que ir al modelo creado de Productos y añadimos el

```
namespace namespace App\Models;
```

Una vez hecho, se hace la programación dentro del controller se hace la ruta en la “web.php”

```
public function store(Request $request) {  
    $newProduct = new Product;  
    $newProduct->description = $request->input('description');  
    $newProduct->price = $request->input('price');  
    $newProduct->save();  
    return redirect()->route('products.index')->with('info',  
'Producto insertado correctamente');  
}
```

### 4. Listar productos

La programación del botón es esta:

Una vez programado el botón, escribimos lo siguiente en la vista para poder mostrarlo dentro de un alert:

```
@if(session('info'))  
<div class="alert alert-success">  
    {{session('info')}}  
</div>  
@endif
```

Después se crea otra ruta para mostrar en la vista:

empieza en el @foreach(\$products as \$producto),

se pone en cada celda de la tabla el producto a visualizar y se cierra el bucle.

También podemos editar el orden en el que se visualizan los productos añadiendo esta línea:

(controller)

```
@foreach($products as $producto)  
<tr>  
    <td>  
        {{ $producto->description }}  
    </td>  
    <td>  
        {{ $producto->price }} €  
    </td>  
    <td>  
        <a href="{{ route('products.edit', $producto->id)}}">Editar</a>  
        <a href="javascript: document.getElementById('{{ $producto->id }}').delete()">Eliminar</a>  
        <form action="{{ route('products.delete', $producto->id)}}" method="post">  
            @method('delete')  
            @csrf  
        </form>  
    </td>  
</tr>  
@endforeach
```

## 5. Eliminar productos

La programación del botón es esta:

```
public function destroy($id) {
    $product = Product::findOrFail($id);
    $product->delete();
    return redirect()->route('products.index')->with('info',
'Producto eliminado correctamente');
}
```

## 6. Modificar productos

Creamos otra vista llamada edit.blade.php donde usaremos la vista create pero editada:

```
<form action="{{ route('products.update',$product->id) }}"
method="POST">
    @method('put')
    @csrf
    <div class="form-group">
        <label for="">Descripcion</label>
        <input type="text" name="description"
value="{{ $product->description }}" class="form-control">
    </div>
    <div class="form-group">
        <label for="">Precio</label>
        <input type="number" name="price"
value="{{ $product->price }}" class="form-control">
    </div>
    <div class="form-group">
        <br>
        <button type="submit" class="btn btn-
primary">Editar</button>
        <a href="{{ route('products.index') }}"
class="btn btn-danger">Cancelar</a>
    </div>
</form>
```

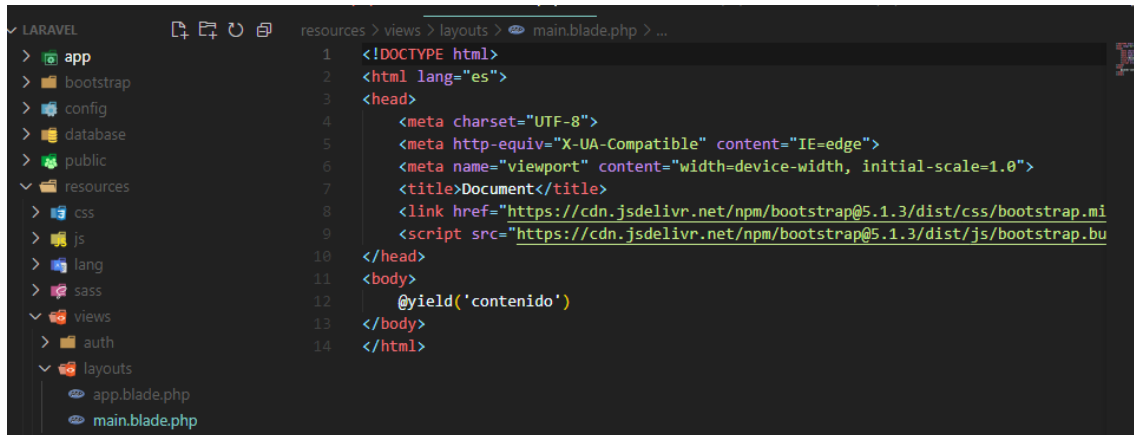
En el formulario utilizamos el \$product -> id para rescatar el id del producto seleccionado y poder modificarlo despues.

En cada input ponemos el valor original con los value="{{ \$product->description }}" o precio, depende del valor que quieras rescatar

La ruta es de tipo put (para actualizar recursos).

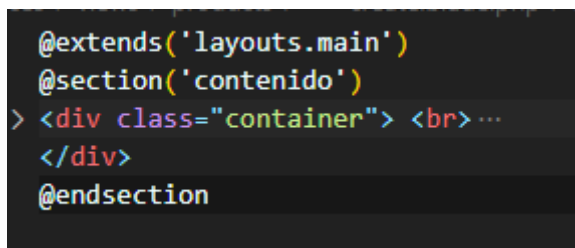
## 7. Refactorizar las vistas Blade

Para poder reutilizar código, creamos en la carpeta de layouts un archivo llamado main.blade.php:



```
resources > views > layouts > main.blade.php > ...
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.mi
9   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bu
10 </head>
11 <body>
12   @yield('contenido')
13 </body>
14 </html>
```

De esta forma, creamos la estructura de nuestra página web. @yield('contenido') es para enlazar el contenido más adelante:



```
@extends('layouts.main')
@section('contenido')
> <div class="container"> <br>...
</div>
@endsection
```

## 8. Protección de la aplicación con autenticación

La imagen de la web anteriormente mostrada ([ver aquí](#)), está contenida dentro de una ruta tipo Middleware para poder protegerlas.

Para ello hay que ejecutar un par de comandos:

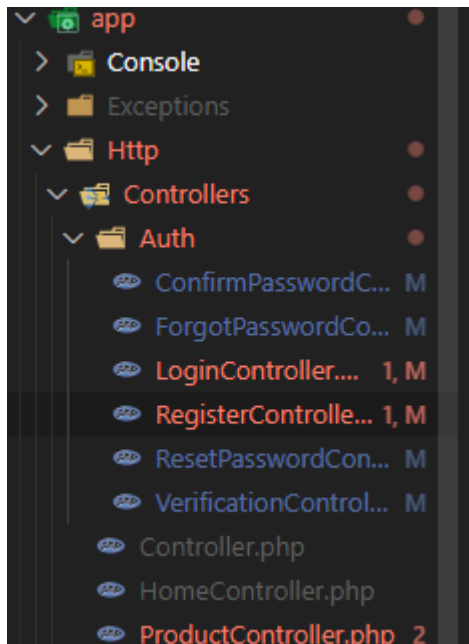
```
composer require laravel/ui || php artisan ui vue --auth
```

Una vez ejecutados, en la página de “web.php” os saldrán dos líneas de código nuevas, pero solo tienes que dejar la de: `Auth::routes();`

Se generará también una vista automáticamente dentro de layouts, le añadimos los enlaces de Bootstrap y listo.



Una vez hecho eso, nos dirigimos a las carpetas del controlador que se encuentran aquí:



En la carpeta de Auth cambiamos las líneas de código que encontremos con el sitio donde tiene que redirigir:

```
protected $redirectTo = '/products';
```

Y más adelante se crea un mensaje de bienvenida en el index:

```
Bienvenido {{auth()->user()->name}}
```

```
<a href="javascript:
document.getElementById('logout').submit()" class="btn btn-danger btn-sm
float-end">Cerrar sesión</a>
    <form id="logout" action="{{route('logout')}}"
method="POST" style="display:none">
        @csrf
    </form>
```

Después escribimos este código dentro del login Controller:

```
public function logout(Request $request)
{
    $this->guard()->logout();

    $request->session()->invalidate();

    $request->session()->regenerateToken();

    if ($response = $this->loggedOut($request)) {
        return $response;
    }

    return $request->wantsJson()
        ? new JsonResponse([], 204)
        : redirect('/login');
}
```

E importamos esta clase:

```
use Illuminate\Http\Request;
```

## 9. Controladores

Se crea un controller con este código: php artisan make:controller ProductController.

Creamos esta ruta dentro del archivo RouteServiceProvider (dentro de app>providers)

```
protected $namespace = 'App\Http\Controllers';
```

Una vez creado, metemos dentro todas las funciones de web.php de manera que web se quede [asi](#) y el controler asi:

```
public function index () {
    $products = DB::table('products')->where('price','>','3')->get();
    return view('products.index', compact('products'));
}
public function create(){
    return view('products.create');
}
public function store(Request $request) {
    $newProduct = new Product;
    $newProduct->description = $request->input('description');
    $newProduct->price = $request->input('price');
    $newProduct->save();
    return redirect()->route('products.index')->with('info',
'Producto insertado correctamente');
}
public function destroy($id) {
    $product = Product::findOrFail($id);
    $product->delete();
    return redirect()->route('products.index')->with('info',
'Producto eliminado correctamente');
}
public function edit($id) {
    $product = Product::findOrFail($id);
    return view('products.edit', compact('product'));
}
public function update(Request $request, $id) {
    $product = Product::findOrFail($id);
    $product->description = $request->input('description');
    $product->price = $request->input('price');
    $product->save();
    return redirect()->route('products.index')->with('info' .
'Producto modificado correctamente');
```