Carlos Peñalver Alonso

2DAW

Glosario Laravel

Contenido

[1. Instalación 2](#_Toc94827914)

[Primeros pasos 2](#_Toc94827915)

[Migraciones 2](#_Toc94827916)

[Modelos 3](#_Toc94827917)

[2. Introducción a vista y rutas 3](#_Toc94827918)

[Vistas 4](#_Toc94827919)

[Rutas 4](#_Toc94827920)

[3. Inserción de datos 5](#_Toc94827921)

[4. Listar productos 5](#_Toc94827922)

[5. Eliminar productos 6](#_Toc94827923)

[6. Modificar productos 6](#_Toc94827924)

[7. Refactorizar las vistas Blade 7](#_Toc94827925)

[8. Protección de la aplicación con autenticación 7](#_Toc94827926)

[9. Controladores 9](#_Toc94827927)

# Instalación

## Primeros pasos

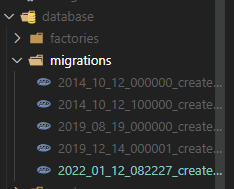
1. Creamos una carpeta donde queramos el proyecto (abriendo un PowerShell) y la nombramos.
2. Descargamos laravel con el siguiente comando “composer create-project laravel/laravel
3. Abrimos el proyecto en code.
4. Creamos una bbdd nueva en phpmyadmin.
5. Configuramos el archivo ‘.env’
6. Buscamos las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

## Migraciones

Un ejemplo de migración sería: php artisan make:migration créate\_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');

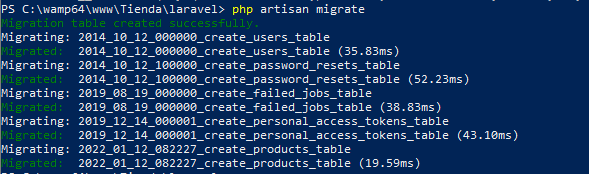
 $table->integer('price');

 $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”.



## 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.

# 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.

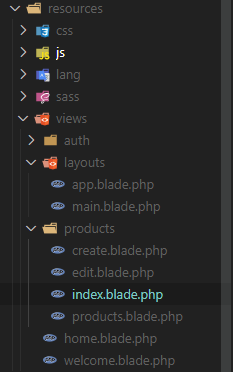
Por ejemplo:  Route::get('/', function(){

*return* 'Esta es nuestra URL raiz';

    });

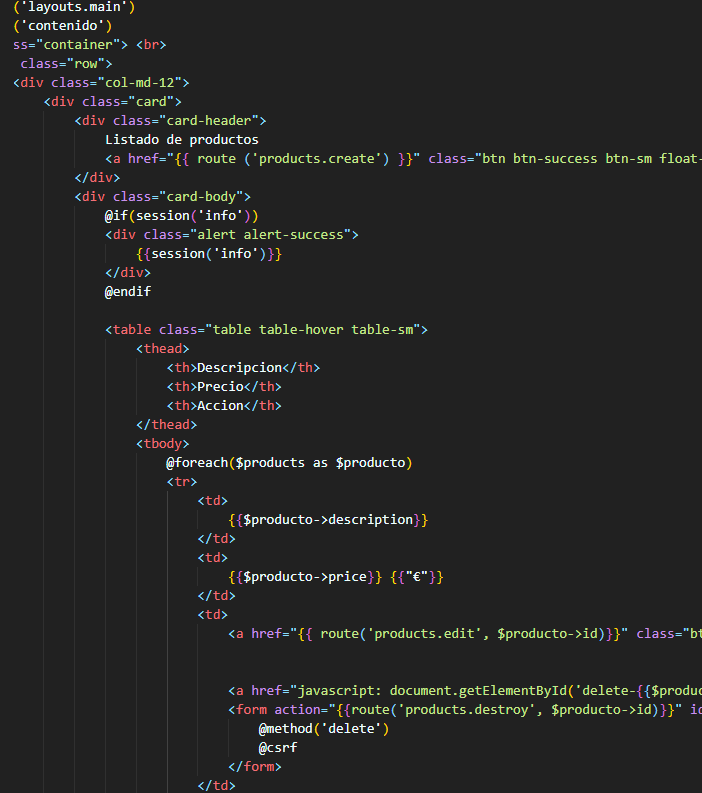
Para probarlo ir a <http://localhost/Tienda/laravel/public>.

## 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.

## 

## Rutas

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

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).

# 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:

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();

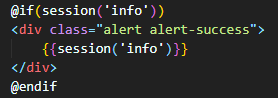
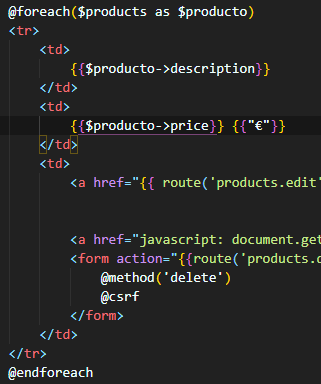
*return* redirect()->route('products.index')->with('info', 'Producto insertado correctamente');

    }

# 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:

 Despues 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)

# 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');

    }

# 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>

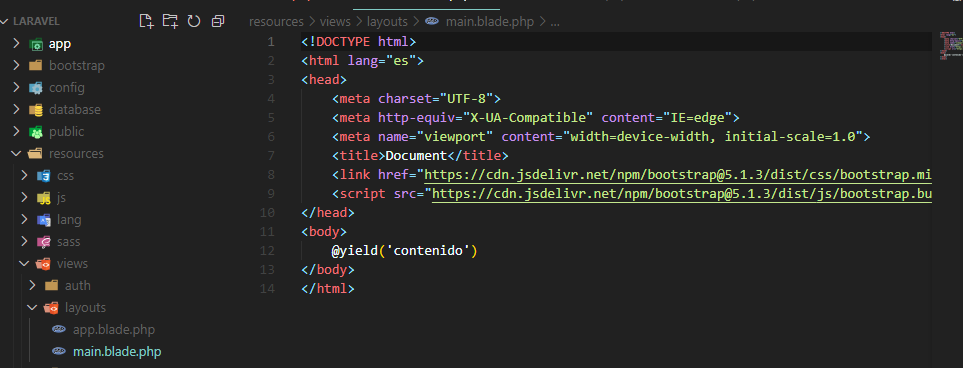
En el formulario utilizamos el $product -> id para rescatar el id del producto seleccionado y poder modificarlo despues.

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

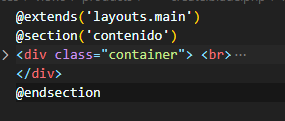
La ruta es de tipo put (para actualizar recursos).

# Refactorizar las vistas Blade

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



De esta forma, creamos la estructura de nuestra página web. @yield(‘contenido’) es para enl azar el contenido más adelante:



# Protección de la aplicación con autenticación

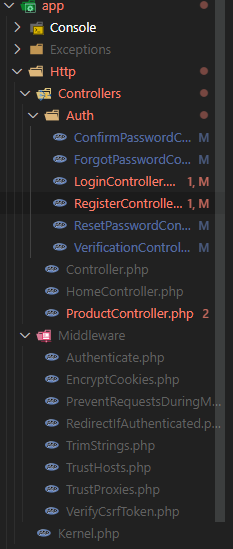
La imagen de la web anteriormente mostrada ([ver aquí](#_Rutas)), está contenida dentro de una ruta tipo Middleware para poder protegerlas.

Para ello hay que ejecutar un par de comandos:

composer require laravel/ui || php artisan ui vue –auth

Una vez ejecutados, en la pagina 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>

Despés 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;

# 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](#_Rutas) 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');