DOCUMENTACIÓN

Creación del proyecto

Para crear el proyecto usamos el comando “composer create-project laravel/laravel ‘nombre’”.

```
PS C:\srv\Laravel> composer create-project laravel/laravel articulo
Installing laravel/laravel (v6.12.0)
```

Una vez creado el proyecto tenemos que crear un usuario y una base de datos para el mismo. En mi caso he creado un usuario y una bdd tienda.

Configuramos el archivo .env con estos datos

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tienda
DB_USERNAME=usuario
DB_PASSWORD=usuario
```

Configuramos el alias.

```
<Directory "C:/srv/Laravel/articulo/public">
        Options Indexes FollowSymlinks MultiViews
        RewriteEngine On
        AllowOverride All
        Require all granted
</Directory>
alias /articulo "C:/srv/Laravel/articulo/public"
```

Una vez echo esto ya tenemos nuestro proyecto laravel funcionando.

Para poder hacer el crud creamos primero el modelo Articulo con las migraciones, factory, seeder y controllador con el comando “php artisan make:model ‘Nombre’ -mrcsf”.

```
PS C:\srv\Laravel\articulo> php artisan make:model Articulo -mrcsf
Model created successfully.
Factory created successfully.
Created Migration: 2020_02_13_112657_create_articulos_table
Seeder created successfully.
Controller created successfully.
```

Ya podemos hacer el crud.

Primeros Pasos

Rutas

Primero hay que crear las rutas, en el archivo web.php

```
Route::resource('articulos','ArticuloController');
```

Plantilla

Creamos una plantilla.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>@yield('titulo')</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstra

</head>
<body style="background-color: darkturquoise">
    <h3 class="text-center">@yield('cabecera')</h3>
    @yield('contenido')
</body>
</html>
```

Modelo

Creamos una variable con los campos que podremos rellenar de articulos.

```
protected $fillable=['nombre','categoria','precio','stock','imagen'];
```

Tabla

En la migración introducimos los datos de la tabla.

```php
Schema::create('articulos', function (Blueprint $table) {
    $table->bigIncrements('id');
    $table->string('nombre');
    $table->string('categoria');
    $table->integer('precio');
    $table->integer('stock');
    $table->string('imagen')->default('img/articulos/default.jpg');
    $table->timestamps();
});
```

Seeder

En DatabaseSeeder ponemos el Seeder de articulos para poder introducir los datos en la bdd.

```php
$this->call(ArticuloSeeder::class);
```

Una vez echo esto podremos empezar con el Crud del proyecto.

Datos

Introducimos algunos artículos.

```php
Articulo::create([
    'nombre'=>'Ordenador',
    'categoria'=>'Electronica',
    'precio'=>'900',
    'stock'=>'10'
]);
```

Hacemos la migración para introducir los datos en la bdd con el comando “php artisan migrate:fresh –seed”

```
PS C:\srv\Laravel\articulo> php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
```

CRUD

Index

En este método devolvemos la vista donde mostraremos todos los artículos.

```
$articulos= Articulo::orderby('id');
return view('articulos.index',compact('articulos'));
```

Crearemos una vista ‘index.blade.php’ para mostrar los datos.

Con este foreach mostramos los datos y creamos los botones para borrar y editar de cada artículo.

```
@foreach ($articulos as $item)
<tr>
    <th scope="row">{{$item->id}}</th>
    <td>{{$item->nombre}}</td>
    <td>{{$item->categoria}}</td>
    <td>{{$item->precio}}</td>
    <td>{{$item->stock}}</td>
    <td>{{$item->imagen}}</td>
    <td>
    <form action="{{route('articulos.destroy',$item)}}" method="POST" name="b">
        @method('DELETE')
        @csrf
        <input type="submit" class="btn btn-danger" value="Borrar">
        <a href="{{route('articulos.edit',$item)}}" class="btn btn-warning">Editar</a>
      </form>
    </td>
</tr>
```

Create

Devolvemos la vista en el método create.

```
return view('articulos.create');
```

Creamos la vista con el formulario para guardar los datos.

Esto es solo crear un formulario por lo que no pondré ninguna captura.

Ahora hacemos el método store.

Ya que no hay mucho que controlar en este ejercicio, tan solo validaré que se envié una foto.

```php
if($request->has('foto')){
    $request->validate([
        'foto'=>['image']
    ]);
    //Nombre de la foto
    $file=$request->file('foto');
    //le asignamos la ruta
    $nombre='articulos/'.time().'_'.$file->getClientOriginalName();
    //Lo guardamos en public
    Storage::disk('public')->put($nombre, \File::get($file));
    //Una vez guardado creamos el nuevo articulo
    $articulo=Articulo::create($request->all());
    $articulo->update(['foto'=>"img/$nombre"]);
}else{
    Articulo::create($request->all());
}
return redirect()->route('articulos.index')->with('mensaje','Articulo creado Correctamente');
```

Para guardar las imágenes en la carpeta public/img/articulos tenemos que modificar el archivo filesystem.php en config.

```php
'public' => [
    'driver' => 'local',
    //'root' => storage_path('app/public'),
    'root' => public_path('img'),
    //'url' => env('APP_URL').'/storage',
    'url' => env('APP_URL').'/img',
    'visibility' => 'public',
],
```

Delete

Comprobamos que la foto que queremos borrar no es default.

```
$foto=$articulo->imagen;
//La foto no es default.jpg por lo que la borramos
if(basename($foto)!="default.jpg"){
    unlink($foto);
}
$articulo->delete();
return redirect()->route('articulos.index')->with('mensaje','Articulo borrado Correctamente');
```

Edit

Este método devuelve una vista donde editaremos el artículo seleccionado.

```
return view('articulos.edit',$articulo);
```

Ahora hacemos la vista con el formulario para editar el artículo.

Una vez hecho el formulario pasamos al método update.

Si se sube una foto, controlo que es una imagen lo que se sube y si lo es cambio la ruta del articulo a esa foto.

```
if($request->has('foto')){
    $request->validate([
        'foto'=>'image'
    ]);

    $file=$request->file('foto');
    $nombre='articulos/'.time().'_'.$file->getClientOriginalName();
    Storage::disk('public')->put($nombre, \File::get($file));
    $articulo->update($request->all());
    $articulo->update(['imagen'=>"img/$nombre"]);
}else{
    $articulo->update($request->all());
}
return redirect()->route('articulos.index')->with('mensaje','Articulo editado correctamente');
```

Show

Este método devuelve una vista.

```
return view('articulos.show',compact('articulo'));
```

Hacemos la vista para mostrar cada artículo.

```
@extends('plantillas.plantilla')
@section('titulo')
    Detalle Articulo
@endsection
@section('cabecera')
    Detalles Articulo Nombre <i><b>{{($articulo->nombre)}}</b></i>

@endsection
@section('contenido')
    <span class="clearfix"></span>
    <div class="card text-white bg-info mt-5 mx-auto" style="max-width: 48rem;">
        <div class="card-header text-center"><b>{{($articulo->id)}}</b></div>
        <div class="card-body" style="font-size: 1.1em">
            <p class="card-text">
            <div class="float-right"><img src="{{asset($articulo->imagen)}}" width="160px" heght="160px" class="rounded-circle"></div>
            <p><b>Nombre:</b> {{$articulo->nombre}}</p>
            <p><b>Categoria:</b> {{$articulo->categoria}}</p>
            <p><b>Precio (€):</b> {{$articulo->precio}}</p>
            <p><b>Stock:</> {{$articulo->stock}}</p>
            </p>
            <a href="{{route('articulos.index')}}" class="float-right btn btn-success">Volver</a>
        </div>
    </div>
```

Por ultimo hacemos el filtro de artículos.

Para ello hacemos un scope para cada filtro.

```
public function scopeCategoria($query,$v){
    if($v=='%'){
        return $query->where('categoria','like',$v);
    }else{
        return $query->where('categoria',$v);
    }

}
```

```php
public function scopePrecio($query,$v){
    if($v=='%'){
        return $query->where('precio','like',$v);
    }
    switch ($v) {
        case '1':
            return $query->where('precio','>','0')->where('precio','<','50');
            break;
        case '2':
            return $query->where('precio','>','50')->where('precio','<','200');
            break;
        case '3':
            return $query->where('precio','>','200')->where('precio','<','500');
            break;
        case '4':
            return $query->where('precio','>','500')->where('precio','<','1000');
            break;
        default:
            # code...
            break;
    }
}
```

Por último modificamos la vista index añadiendo un formulario y editando la paginación.

```blade
<form action="{{route('articulos.index')}}" class="form-inline float-right">
 <select name="precio" class="form-control" onchange="this.form.submit()">
   <option value="%">Todos</option>
   <option value="1">0-50€</option>
   <option value="2">50-200€</option>
   <option value="3">200-500€</option>
   <option value="4">500-1000€</option>
 </select>
 <select name="categoria" class="form-control" onchange="this.form.submit()">
   <option value="%">Todos</option>
   @foreach ($categorias as $item)
     @if ($item==$request->categoria)
     <option value="{{$item}}" selected>{{$item}}</option>
     @else
     <option value="{{$item}}">{{$item}}</option>
     @endif
   @endforeach
 </select>
 <input type="submit" class="btn btn-primary" value="Buscar">
</form>
```

```blade
{{$articulos->appends(Request::except('page'))->links()}}
```