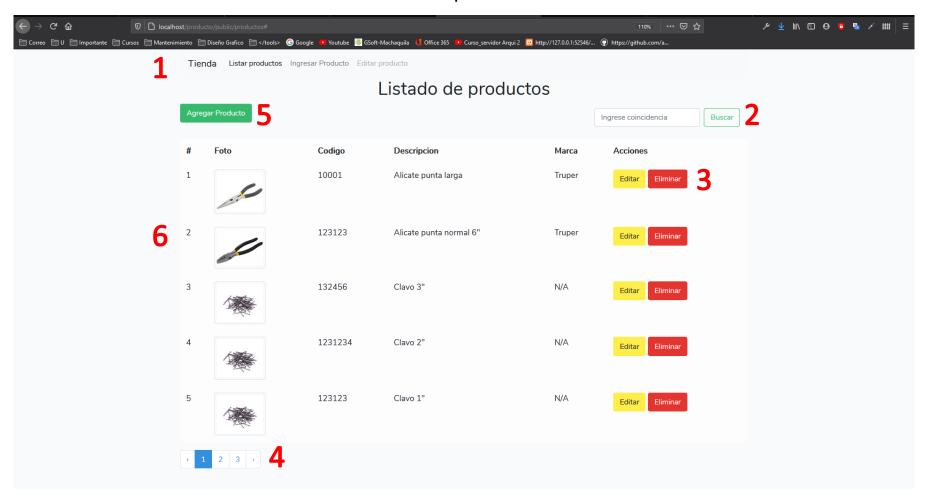
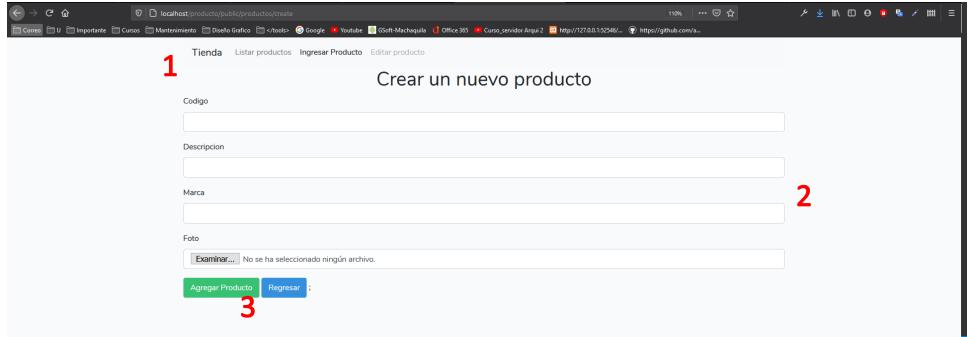
## Listar producto



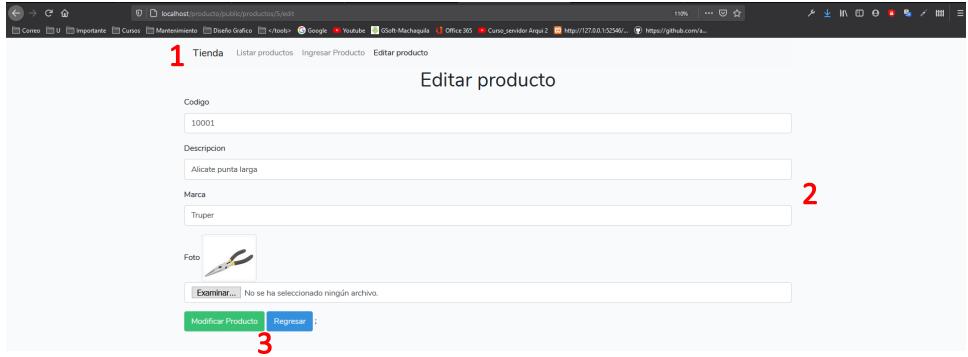
- 1 Barra de navegación, 3 opciones, listar producto, ingresar y editar, esta última bloqueada ya que para editar un producto debe darle en el botón editar de cada ítem.
- **2** Campo para realizar filtro por medio de nombre de producto. Para listar todos no ingresar ningún texto.
- **3** Acciones, eliminar y editar por cada articulo
- 4- Barra de navegación
- **5** Ir a formulario ingreso de datos
- 6 Tabla con información de productos

# Ingresar producto



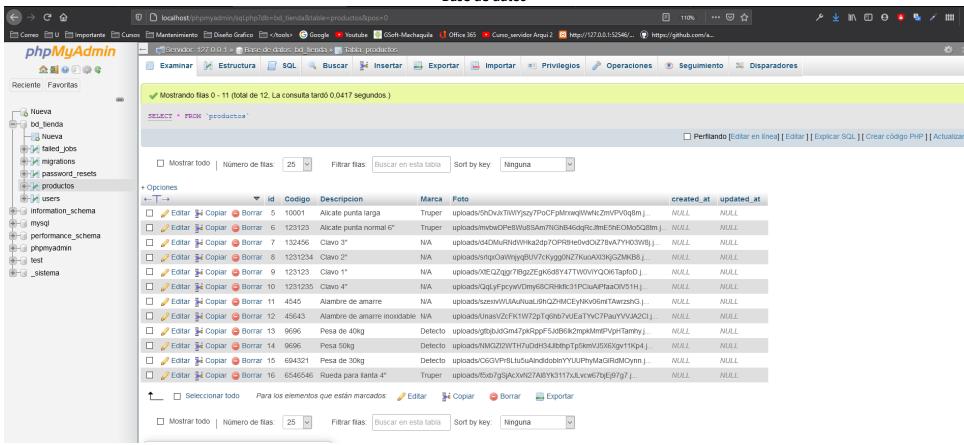
- 1 Barra de navegación, 3 opciones, listar producto, ingresar y editar, esta última bloqueada ya que para editar un producto debe darle en el botón editar de cada ítem.
- **2** Campos a registrar
- **3** Acciones, registrar y regresar

## **Editar producto**



- 1 Barra de navegación, 3 opciones, listar producto, ingresar y editar, esta última bloqueada ya que para editar un producto debe darle en el botón editar de cada ítem.
- **2** Campos a registrar
- **3** Acciones, registrar y regresar

#### Base de datos



## Código Importante

```
<?php
namespace App\Http\Controllers;
use App\Producto;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
class ProductoController extends Controller
     * Display a listing of the resource.
     * @return \Illuminate\Http\Response
    public function index(Request $request)
        $nombre = $request ->get('buscar');
        //dd($nombre);
        $productos['productos']=Producto::where('Descripcion','like',"%$nombre%")-
>paginate(5);
      // $productos['productos']=Producto::paginate(4);
        return view('productos.index',$productos);
     * Show the form for creating a new resource.
     * @return \Illuminate\Http\Response
    public function create()
        return view('productos.create');
     * Store a newly created resource in storage.
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
    public function store(Request $request)
```

```
$campos=[
        'Codigo'=> 'required|string|max:100',
        'Descripcion'=> 'required|string|max:100',
        'Marca'=> 'required|string|max:100',
        'Foto'=> 'required|max:1000|mimes:jpeg,png,jpg',
    ];
    $mensaje=[
        "required"=>'El campo :attribute es requerido'
    1;
    $this->validate($request,$campos,$mensaje);
    $datosProducto = request()->except('_token');
    //return response()->json($datosProducto);
    if($request->hasfile('Foto')){
        $datosProducto['Foto']=$request->file('Foto')->store('uploads','public');
    Producto::insert($datosProducto);
   return redirect('productos')->with('Mensaje','Producto agregado con éxito');
 * Display the specified resource.
 * @param \App\Producto $producto
 * @return \Illuminate\Http\Response
public function show(Producto $producto)
 * Show the form for editing the specified resource.
 * @param \App\Producto $producto
 * @return \Illuminate\Http\Response
public function edit($id)
```

```
$producto = Producto::findOrFail($id);
        return view('productos.edit',compact('producto'));//producto es la misma bariable de
ar arriba
     * Update the specified resource in storage.
     * @param \Illuminate\Http\Request $request
     * @param \App\Producto $producto
     * @return \Illuminate\Http\Response
   public function update(Request $request, $id)
        $campos=[
            'Codigo'=> 'required|string|max:100',
            'Descripcion'=> 'required|string|max:100',
            'Marca'=> 'required|string|max:100',
       ];
        if($request->hasfile('Foto')){
           $campos+=['Foto'=> 'required|max:1000|mimes:jpeg,png,jpg'];
        $mensaje=[
            "required"=>'El campo :attribute es requerido'
        ];
        $this->validate($request,$campos,$mensaje);
        $datosProducto = request()->except(['_token','_method']);
        if($request->hasfile('Foto')){
           $producto = Producto::findOrFail($id);
           Storage::delete('public/'.$producto->Foto);
           $datosProducto['Foto']=$request->file('Foto')->store('uploads','public');
        }
```

```
Producto::where('id','=',$id)->update($datosProducto);
   $producto = Producto::findOrFail($id);
   //return view('productos.edit',compact('producto'));
   return redirect('productos')->with('Mensaje','Producto modificado con éxito');
 * Remove the specified resource from storage.
 * @param \App\Producto $producto
 * @return \Illuminate\Http\Response
public function destroy($id)
    $producto = Producto::findOrFail($id);
    if(Storage::delete('public/'.$producto->Foto)){
        Producto::destroy($id);
           return redirect('productos')->with('Mensaje','Producto Eliminado con éxito');
}
```

Este es el código más importante (Controlador ), ya que este es el controlador de producto, aquí se agrega, se edita y elimina. Lo demás es solo vista CSS + HTLM.

```
vse Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateProductosTable extends Migration
{
    /**
    * Run the migrations.
    *
    * @return void
```

```
Rodolfo Alfredo Martínez Román
```

```
*/
public function up()
{
    Schema::create('productos', function (Blueprint $table) {
        $table->increments('id');
        $table->string('Codigo');
        $table->string('Descripcion');
        $table->string('Marca');
        $table->string('Foto');
        $table->timestamps();
    });
}

/**
    * Reverse the migrations.
    *
        @return void
        */
public function down()
    {
        Schema::dropIfExists('productos');
    }
}
```

### Configuración de base de datos

### Consola

1. Crear proyecto

Composer créate-project -prefer-dist laravel/laravel product

2. Migraciones

Php artisan migrate

3. Rollback para migraciones

artisan migrate:rollback

4. Crear el modelo product

php artisan make:model Producto -mcr

5. Ver lista de rutas disponibles

php artisan route:list

6. Habilitar enlaces para fotografías almacenadas

php artisan storage:link

7. Habilitar bootstrap 4

```
composer require laravel/ui
php artisan ui bootstrap
npm install
npm run dev
```