**MANUAL TÉCNICO**

**CategoryController**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\Category;

use Image;

use Intervention\Image\ImageManager;

class CategoryController extends Controller

{

*/\*\**

*\* @description Esté método busca todas las categorías*

*\* @return regresa todas las categorías*

*\*/*

    public function AllCategory(){

        $categories = Category::all();

        return $categories;

    }

*/\*\**

*\* @description Esté método obtiene todas las categorías*

*\* @return regresa a la vista de todas las categorías*

*\*/*

    public function GetAllCategory(){

        $category = Category::latest()->get();

        return view('backend.category.category\_view',compact('category'));

    } *// End Mehtod*

*/\*\**

*\* @ description Esté método es para entrar a la vista de agregar categorías*

*\* @return regresa a la vista de agregar categorías*

*\*/*

    public function AddCategory(){

        return view('backend.category.category\_add');

      } *// End Mehtod*

*/\*\**

*\* @description Esté método inserta los datos del formulario de categorías a la base de datos*

*\* Tiene una validación donde es obligatorio colocar el nombre de la categoria*

*\* Tiene las variables de imagen para almacenar la imagen*

*\* Image es un método para almacenar la imagen en la carpeta definida y también modifica su tamaño al que queramos*

*\* save\_url guarda la imagen en la ruta donde se encuentra la carpeta de category*

*\* Category insert es para insertar el nombre de la categoría y su imagen*

*\* Notification es para mandar una alerta de que se guardó los datos correctamente*

*\* @return regresa a la vista de todas las categorías*

*\*/*

      public function StoreCategory(Request $request){

        $request->validate([

            'category\_name' => 'required',

        ],[

            'category\_name.required' => 'Input Category Name'

        ]);

        $image = $request->file('category\_image');

        $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

        Image::make($image)->resize(128,128)->save('upload/category/'.$name\_gen);

        $save\_url = 'http://127.0.0.1:8000/upload/category/'.$name\_gen;

        Category::insert([

            'category\_name' => $request->category\_name,

            'category\_image' => $save\_url,

        ]);

        $notification = array(

            'message' => 'Category Inserted Successfully',

            'alert-type' => 'success'

        );

        return redirect()->route('all.category')->with($notification);

    }*// End Mehtod*

*/\*\**

*\* @description busca todas las categorías por medio de la id*

*\* @return regresa a la vista de editar categorías*

*\*/*

    public function EditCategory($id){

        $category = Category::findOrFail($id);

        return view('backend.category.category\_edit',compact('category'));

    } *//End Method*

*/\*\**

*\* Método para actualizar las categorías*

*\* Busca por medio de la id que tiene la categoría*

*\* Tiene la condición de que se subió una nueva imagen, entonces se sube el nombre de la categoría y la imagen nueva*

*\* La imagen nueva se sube en la ruta donde está la carpeta category*

*\* Category findOrFail ayuda a ingresar los nuevos datos por medio de la id y con el método de update*

*\* Ingresa el nuevo nombre de la categoría y la imagen nueva y manda una alerta de que se actualizó correctamente*

*\* Si la condición no se sube una nueva imagen, entonces solo se actualiza el nombre de la categoría*

*\* @return regresa a la vista de todas las categorías*

*\*/*

    public function UpdateCategory(Request $request){

        $category\_id = $request->id;

        if ($request->file('category\_image')) {

        $image = $request->file('category\_image');

        $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

        Image::make($image)->resize(128,128)->save('upload/category/'.$name\_gen);

        $save\_url = 'http://127.0.0.1:8000/upload/category/'.$name\_gen;

        Category::findOrFail($category\_id)->update([

            'category\_name' => $request->category\_name,

            'category\_image' => $save\_url,

        ]);

        $notification = array(

            'message' => 'Category Update With Image Successfully',

            'alert-type' => 'success'

        );

        return redirect()->route('all.category')->with($notification);

        }

        else{

             Category::findOrFail($category\_id)->update([

            'category\_name' => $request->category\_name,

        ]);

        $notification = array(

            'message' => 'Category Updateed Without Image Successfully',

            'alert-type' => 'success'

        );

        return redirect()->route('all.category')->with($notification);

        }

    } *//End Method*

*/\*\**

*\* @description Esté método es para eliminar la categoría por medio de la id*

*\* Después de eliminar la categoría, mostrará una notification de que se eliminó correctamente*

*\* @return regresa a la vista de todas las categorías*

*\*/*

    public function DeleteCategory($id){

        Category::findOrFail($id)->delete();

        $notification = array(

                'message' => 'Category Deleted Successfully',

                'alert-type' => 'success'

            );

            return redirect()->back()->with($notification);

        }*//End Method*

}

**ContactController**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\Contact;

*/\*\**

*\* @description Esté método es para insertar el nombre de usuario, el correo y por ultimo el mensaje*

*\* Date\_default\_timezone\_set es para que en el admin panel se muestre la hora en que envió el mensaje*

*\* Contact insert es para insertar todos los datos requeridos para el formulario de contactos*

*\* @return regresa todos los contactos*

*\*/*

class ContactController extends Controller

{

    public function PostContactDetails(Request $request){

        $name = $request->input('name');

        $email = $request->input('email');

        $message = $request->input('message');

        \date\_default\_timezone\_set("America/Mexico\_City");

        $contact\_time = date("h:i:sa");

        $contact\_date = date("d-m-Y");

        $result = Contact::insert([

            'name' => $name,

            'email' => $email,

            'message' => $message,

            'contact\_date' => $contact\_date,

            'contact\_time' => $contact\_time,

        ]);

        return $result;

    }*// End Method*

*/\*\**

*\* @description Esté método es para obtener todos los contactos ya enviados*

*\* @return regresa la vista de todos los contactos*

*\*/*

    public function GetAllMessage(){

        $message = Contact::latest()->get();

        return view('backend.contact.contact\_all', compact('message'));

    } *// End Method*

*/\*\**

*\* @description Esté método es para eliminar los contactos por medio de la id*

*\* contact findOnfail busca la id para después el método delete elimine el contacto con la id*

*\* notification es para que mande una alerta de que se eliminó el contacto de forma exitosa*

*\* @return regresa a la vista de todos los contactos con la alerta*

*\*/*

    public function DeleteMessage($id){

        Contact::findOrFail($id)->delete();

         $notification = array(

            'message' => 'Message Deleted Successfully',

            'alert-type' => 'success'

        );

        return redirect()->back()->with($notification);

    }*// End Method*

}

**ProductListController**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\ProductList;

use App\Models\Category;

use Image;

class ProductListController extends Controller

{

*/\*\**

*\* @description Esté método es para mostrar todos los productos con remark*

*\* @return regresa todos los productos con remark*

*\*/*

    public function ProductListByRemark(Request $request){

        $remark = $request->remark;

        $productlist = ProductList::where('remark', $remark)->limit(8)->get();

        return $productlist;

    } *// End Method*

*/\* @description El método ProductListByCategory busca todos los productos*

*\* clasificados en cada categoría para regresarlos a la vista con el*

*\* @return $productlist regresa la lista de productos ligados a la categoría seleccionada.*

*\*/*

    public function ProductListByCategory(Request $request){

        $Category = $request->category;

        $productlist = ProductList::where('category', $Category)->get();

        return $productlist;

    } *// End Method*

*/\* @description El método ProductBySearch busca todos los productos que coincidan*

*\* en nombre con la condición de que se escriban 2 o más letras entre el valor ingresado por el usuario y los productos ya existentes.*

*\*  Retorna los resultados por medio de @return $productlist*

*\*/*

    public function ProductBySearch(Request $request){

        $key = $request->key;

        $productlist = ProductList::where('title', 'LIKE', "%{$key}%")->get();

        return $productlist;

    }

*/\* @description El método GetAllProduct obtiene todos los productos y los muestra*

*\* @return regresa a la vista de todos los productos*

*\*/*

    public function GetAllProduct(){

        $products = ProductList::latest()->paginate(10);

        return view('backend.product.product\_all',compact('products'));

    } *// End Method*

*/\* @description El método AddProduct enlaza la vista de añadir productos*

*\* con las categorías registradas.*

*\* @return view devuelve la vista de agregar productos.*

*\*/*

    public function AddProduct(){

        $category = Category::orderBy('category\_name','ASC')->get();

        return view('backend.product.product\_add',compact('category'));

    } *// End Method*

*/\*\**

*\* @description Esté método es para insertar productos*

*\* Image sube la imagen a la carpeta product con un tamaño especifico por medio de la ruta*

*\* notification muestra una alerta cuando el producto se insertó correctamente*

*\* @return regresa a la vista de todos los productos*

*\*/*

    public function StoreProduct(Request $request){

       $image = $request->file('image');

       $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

       Image::make($image)->resize(711,960)->save('upload/product/'.$name\_gen);

       $save\_url = 'http://127.0.0.1:8000/upload/product/'.$name\_gen;

       $product\_id = ProductList::insertGetId([

           'title' => $request->title,

           'price' => $request->price,

           'category' => $request->category,

           'remark' => $request->remark,

           'image' => $save\_url,

       ]);

       $notification = array(

        'message' => 'Product Inserted Successfully',

        'alert-type' => 'success'

    );

    return redirect()->route('all.product')->with($notification);

   } *// End Method*

*/\*\**

*\* @description Esté método es para editar los productos con la vista de editar productos*

*\* Todo está conectado con el producto y la categoría*

*\* @ return regresa a la vista de editar los productos*

*\*/*

   public function EditProduct($id){

    $category = Category::orderBy('category\_name','ASC')->get();

    $product = ProductList::findOrFail($id);

    return view('backend.product.product\_edit',compact('category','product'));

} *// End Method*

*/\*\**

*\* @description Esté método es para editar los productos por medio de la id*

*\* tiene la condición de que si se sube con producto con una imagen entonces también la actualiza*

*\* también actualiza el nombre, precio, categoría, el remark y por último la imagen*

*\* Si no se sube una imagen nueva entonces solo se sube todo lo demás sin la imagen*

*\* notification muestra una alerta de que se actualizó el producto correctamente*

*\* @return regresa a la vista de todos los productos*

*\*/*

public function UpdateProduct(Request $request){

    $product\_id = $request->id;

    if ($request->file('image')) {

        $image = $request->file('image');

        $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

        Image::make($image)->resize(711,960)->save('upload/product/'.$name\_gen);

        $save\_url = 'http://127.0.0.1:8000/upload/product/'.$name\_gen;

        $product\_id = ProductList::findOrFail($product\_id)->update([

            'title' => $request->title,

            'price' => $request->price,

            'category' => $request->category,

            'remark' => $request->remark,

            'image' => $save\_url,

        ]);

    $notification = array(

        'message' => 'Product Update With Image Successfully',

        'alert-type' => 'success'

    );

    return redirect()->route('all.product')->with($notification);

    }

    else{

         ProductList::findOrFail($product\_id)->update([

            'title' => $request->title,

            'price' => $request->price,

            'category' => $request->category,

            'remark' => $request->remark,

    ]);

    $notification = array(

        'message' => 'Product Updateed Without Image Successfully',

        'alert-type' => 'success'

    );

    return redirect()->route('all.product')->with($notification);

    }

} *//End Method*

*/\* @description El método elimina el producto por medio de su id*

*\*@return regresa a la vista de todos los productos*

*\*/*

public function DeleteProduct($id){

    ProductList::findOrFail($id)->delete();

    $notification = array(

            'message' => 'Product Deleted Successfully',

            'alert-type' => 'success'

        );

        return redirect()->back()->with($notification);

    }*//End Method*

}

**SliderController**

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\HomeSlider;

use Image;

class SliderController extends Controller

{

*/\*\**

*\* @description Esté método busca todos los sliders*

*\* @return regresa todos los sliders*

*\*/*

    public function AllSlider(){

        $result = HomeSlider::all();

        return $result;

    }

*/\*\**

*\* @description Esté método obtiene todos los sliders*

*\* @return regresa la vista de todos los sliders*

*\*/*

    public function GetAllSlider(){

        $slider = HomeSlider::latest()->get();

        return view('backend.slider.slider\_view',compact('slider'));

    } *// End Mehtod*

*/\*\**

*\* @description Esté método es para la vista de agregar todos los sliders*

*\* @return regesa a la vista de agregar todos los sliders*

*\*/*

    public function AddSlider(){

         return view('backend.slider.slider\_add');

    }*// End Mehtod*

*/\*\**

*\* @description Esté método es para agregar la imagen para el slider*

*\* EL método solo funciona si se agrega una imagen porque es un dato obligatorio*

*\* la imagen se sube a la carpeta slider mediante su ruta y modifica el tamaño de la imagen con resize*

*\* notification muestra una alerta cuando la imagen se subió exitosamente*

*\* @return regresa a la vista de todos los sliders con la alerta*

*\*/*

    public function StoreSlider(Request $request){

        $request->validate([

           'slider\_image' => 'required',

       ],[

           'slider\_image.required' => 'Upload Slider Image'

       ]);

       $image = $request->file('slider\_image');

       $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

       Image::make($image)->resize(1024,379)->save('upload/slider/'.$name\_gen);

       $save\_url = 'http://127.0.0.1:8000/upload/slider/'.$name\_gen;

       HomeSlider::insert([

           'slider\_image' => $save\_url,

       ]);

       $notification = array(

           'message' => 'Slider Inserted Successfully',

           'alert-type' => 'success'

       );

       return redirect()->route('all.slider')->with($notification);

   }*// End Mehtod*

*/\*\**

*\* @description Esté método es para la vista de editar los sliders*

*\* busca todos los sliders por medio de la id*

*\* @return regresa a la vista de editar los sliders*

*\*/*

   public function EditSlider($id){

    $slider = HomeSlider::findOrFail($id);

    return view('backend.slider.slider\_edit',compact('slider'));

} *// End Mehtod*

*/\*\**

*\* @description Esté método es para editar todos los sliders por medio de la id*

*\* Busca la id correspondiente y muestra los datos del sliders para editarlos*

*\* Image es para subir la nueva imagen con el tamaño ya definido y guardarlo en la carpeta slider con su ruta*

*\* notification es para cuando el slider se modificó exitosamente*

*\* @return regresa a la vista de todos los sliders*

*\*/*

public function UpdateSlider(Request $request){

    $slider\_id = $request->id;

    $image = $request->file('slider\_image');

    $name\_gen = hexdec(uniqid()).'.'.$image->getClientOriginalExtension();

    Image::make($image)->resize(1024,379)->save('upload/slider/'.$name\_gen);

    $save\_url = 'http://127.0.0.1:8000/upload/slider/'.$name\_gen;

    HomeSlider::findOrFail($slider\_id)->update([

        'slider\_image' => $save\_url,

    ]);

    $notification = array(

        'message' => 'Slider Updated Successfully',

        'alert-type' => 'success'

    );

    return redirect()->route('all.slider')->with($notification);

} *// End Mehtod*

*/\*\**

*\* @description Esté método es para eliminar el slider por medio de la id*

*\* Busca la id correspondiente para después eliminarlo*

*\* notification sale la alerta cuando el slider se eliminó correctamente*

*\* @ return regresa a la vista de todas los sliders*

*\*/*

public function DeleteSlider($id){

    HomeSlider::findOrFail($id)->delete();

    $notification = array(

        'message' => 'Slider Deleted Successfully',

        'alert-type' => 'success'

    );

    return redirect()->back()->with($notification);

} *// End Mehtod*

}

**AdminController**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Auth;

use App\Models\User;

use Illuminate\Support\Facades\Hash;

class AdminController extends Controller

{

*/\*\**

*\* @description Esté método es para cerrar sesión del perfil usuarios*

*\* @return regresa a la vista de login al momento de cerrar sesión*

*\*/*

    public function AdminLogout(){

        Auth::logout();

        return Redirect()->route('login');

    } *// end mehtod*

*/\*\**

*\* @description Esté método es para mostrar el usuario que inició sesión*

*\* @return regresa a la vista del perfil del usuario*

*\*/*

    public function UserProfile(){

        $adminData = User::find(1);

        return view('backend.admin.admin\_profile',compact('adminData'));

    } *// end mehtod*

*/\*\**

*\* @description es para editar el usuario existente buscando el id 1*

*\* @ se edita el nombre, correo y por último la foto del usuario*

*\* la foto del usuario se busca y se sube a la carpeta admin\_images*

*\* notification muestra una alerta de que se actualizo correctamente*

*\* @return regresa a la vista del perfil del usuario*

*\*/*

    public function UserProfileStore(Request $request){

        $data = User::find(1);

        $data->name = $request->name;

        $data->email = $request->email;

        if ($request->file('profile\_photo\_path')) {

            $file = $request->file('profile\_photo\_path');

            @unlink(public\_path('upload/admin\_images/'.$data->profile\_photo\_path));

            $filename = date('YmdHi').$file->getClientOriginalName();

            $file->move(public\_path('upload/admin\_images'),$filename);

            $data['profile\_photo\_path'] = $filename;

        }

        $data->save();

        $notification = array(

            'message' => 'User Profile Updated Successfully',

            'alert-type' => 'success'

        );

        return redirect()->route('user.profile')->with($notification);

    } *// end mehtod*

*/\*\**

*\* @decription Esté método es para mostrar la vista de cambiar contraseña*

*\* @ return regresa a la vista de cambiar contraseña*

*\*/*

    public function ChangePassword(){

        return view('backend.admin.change\_password');

    } *// end mehtod*

*/\*\**

*\* @decription Esté método es para mostrar la vista de cambiar contraseña*

*\* @ datos obligatorios la contraseña antigua y confirmar la contraseña*

*\* después ingresar la nueva contraseña encriptada con HASH y cierra sesión*

*\* @return regresa a la vista de cerrar sesión para ingresar con la nueva contraseña*

*\*/*

    public function ChangePasswordUpdate(Request $request){

        $validateData = $request->validate([

            'oldpassword' => 'required',

            'password' => 'required|confirmed'

        ]);

        $hashedPassword = User::find(1)->password;

        if (Hash::check($request->oldpassword,$hashedPassword)) {

            $user = User::find(1);

            $user->password = Hash::make($request->password);

            $user->save();

            Auth::logout();

            return redirect()->route('admin.logout');

        }

        else{

            return redirect()->back();

        }

    } *// end mehtod*

}