Guía Proyecto Tienda Stitch en PHP

Proyecto en PHP con MVC utilizando Bootstrap.

Haremos la misma aplicación que la que se hizo en VB6 en el año 2014-2015

En este documento anotaremos todas las dudas, investigaciones y procesos realizados para la creación del proyecto, como también bibliografía y rutas de directorios donde se encuentre más información útil para este proyecto.

Usaremos de framework Laravel 5.1 de modo de prueba.

Descripción de la aplicación

Aplicación de punto de venta. En ella manejaremos turnos y solo 1 sesión

Tendremos un panel principal en el cual podemos:

Iniciar un turno

Marcará el inicio del turno (Hora – Fecha)

Habilitará entrar en caja.

Cerrar turno

Hora fecha de cierre

Total vendido

ir a la caja

Abrirá Pagina de caja

Ventas

favoritos

editas productos en tiempo de venta.

Administrar almacén

Listar los productos

buscar productos

Ver lista de proveedores

Listar proveedores

Buscar proveedores

Iniciar sesión de administrador

Editar almacén

Editar Proveedores

Cerrar aplicación

Plantillas

Estaran en la carpeta Resource/layouts

La principal será la plantilla main.blade.php

Librerias css: Bootstrap 3.3.5

====================================

- Archivos tocados en Laravel -

====================================

App/User.php

==================

User.php es el model que se usa para manejar los datos de los usuarios,

aquí encontramos los campos que hay en la tabla (name, email, password).

App/Http/routes.php

======================

routes.php guarda las rutas que que van a estar habilitadas en nuestro proyecto.

Route::get('/','FrontController@index');

Llamamos al metodo Route que hace referencia a una ruta get para conseguirla

los parametros son: ('ruta escrita en el navegador','Metodo o funcion que se usa para acceder a ella');

El metodo o funcion en este casa esta en un controlador llamado FrontController y ese controlador contiene

un metodo llamado index el @ hace referencia a que es un metodo.

En routes.php también encontramos otro tipo de ruta

Route::resource('admin','UserController');

Funciona igual que el anterior solo que con resource solo

indicaremos el controlador que vamos a usar ya que dentro

del controlador diremos que vamos a hacer.

app/Http/Controllers/UserController

=====================================

Para crear el controlador de UserControler hemos utilizado el comando dado por

artisan llamado:

cmd> php artisan make:controller UserControler

Esto nos creara un archivo con los metodos basicos para crear registros en la base

datos.

Los controladores tiene la estructura de una clase, en este

controlador hemos programado varios metodos que funcionan

como rutas o que son llamados a la hora de pedir una

una ruta

Enviarnos a la página de crear usuario hemos programado

este metodo

public function create()

{

return view('admin.adminUser');

}

indicamos que es un metodo o función llamado create

que devolverá una vista o view adminUser que esta en

admin.

public function store(UserCreateRequest $request)

{

/\*\*

\* Llamamos al modelo

\*

\*/

User::create([

'name' => $request['name'],

'email' => $request['email'],

'password' => bcrypt($request['password']),

]);

return redirect('/admin/create')->with('message','Usuario creado exitosamente');

}

Con la funcion store pasamos un parametro UserCreateRequest

que hara referencia a los campos que se pueden editar en el modelo

Cuando se inicie la funcion llamara al modelo User aplicará

un metodo de laravel para crear registro que se pasará por un

array con los nombres de los campos editables del modelo y luego

dentro de la variable request los campos del formulario.

una vez insertados los datos devolverá una página mediante

el metodo redirect('ruta de la página') y como opcion se puede

mandar una variable ->('nombre de la variable','contenido de la variable');

app/Http/Controllers/FrontControllers.php

============================================

<?php

namespace AppStitch\Http\Controllers;

use Illuminate\Http\Request;

use AppStitch\Http\Requests;

use AppStitch\Http\Controllers\Controller;

class FrontController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return Response

\*/

}

Estructura de un controlador ^

|

Este controlador contiene todas las rutas que se manejan

a nivel de frontEnd.

public function index()

{

return view('index');

}

Creamos una funcion y decimos que nos devuelva una vista.

app/Html/Request/UserCreateRequest

=======================================

Esta request nos sirve para controlar el acceso a la

comunicacion con la base de datos

aqui tenemos el metodo authorize que lo tenemos que

retornar en true.

Tambien el metodo rules que seran los campos de la base

de datos quese comunicarán con nuestra app.

public function rules()

{

return [

'name' => 'required',

'email' => 'required|unique:users',

'password' => 'required',

];

}

