**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 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**

**Estructura de un controlador**

<?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

\*/

}

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',

];

}

**INSERTAR REGISTROS EN BASE DE DATOS**

**Creando un registro con campo calculado**

**Crear base de datos**

Primero creamos el modelo con el comando: php artisan make:model User –m

Esto hace que se cree el modelo de la tabla con la que nos vamos a conectar y archivo para las migraciones que lo que hace es crear la tabla con la informacion que nosotros le damos automaticamente

Database/migrations:

**public function** up()  
{  
 Schema::*create*(**'almacens'**, **function** (Blueprint $table) {  
  
 $table->increments(**'id'**);  
 $table->string(**'nomProd'**);  
 $table->integer(**'uniProd'**);  
 $table->float(**'precioProd'**);  
 $table->integer(**'ivaProd'**);  
 $table->float(**'netoProd'**);  
 $table->integer(**'idMProd'**);  
 $table->string(**'favProd'**);  
 $table->timestamps();  
  
 });  
}

Con el comando> php artisan migrations

La informacion programada en el archivo de migration se ejecuta en la base de datos y crea la tabla.

**Model**

En los modelos vamos a manejar la tabla de datos con la que nos comunicamos en ella estara identificada la tabla y los campos de esta.

**protected $table** = **'almacens'**;  
  
**protected $fillable** = [**'id'**, **'nomProd'**, **'uniProd'**,**'precioProd'**,**'ivaProd'**,**'netoProd'**,**'idMprod'**,**'favProd'**];

**View**

**Con la sistanxis de blade creamos un formulario**

*{!!Form::label(‘texto’)!!}*

*{!!Form::text(‘nameInput’,null,[‘class’=>’css’])!!}*

*{!! Form::select('favProd', array('pan' => 'Pan', 'bolleria' => 'Bolleria','bebida' =>'Bebida','varios'=>'Varios'), null, ['class'=>'form-control', 'placeholder' => '']) !!}*

{!!Form::*open*(**array**(**'route'**=>**'admin.almacen.store'**))!!}

<**div class="col-md-6"**>  
 <**div class="form-group"**>  
 {!!Form::*label*(**'Código de barras:'**)!!}  
 {!!Form::*text*(**'id'**,**null**,[**'class'**=>**'form-control soloNum'**,**'placeholder'**=>**'Ingresa el código de barras'**])!!}  
 </**div**>  
 <**div class="form-group"**>  
 {!!Form::*label*(**'Nombre del producto:'**)!!}  
 {!!Form::*text*(**'nomProd'**,**null**,[**'class'**=>**'form-control'**,**'placeholder'**=>**'Ingresa el Nombre del producto'**])!!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!!Form::*label*(**'Unidades:'**)!!}  
 {!!Form::*text*(**'uniProd'**,**null**,[**'class'**=>**'form-control soloNum'**,**'placeholder'**=>**'Ingresa el número de unidades'**])!!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!!Form::*label*(**'Precio:'**)!!}  
 {!!Form::*text*(**'precioProd'**,**null**,[**'class'**=>**'form-control soloNum'**,  
 **'placeholder'**=>**'Ingresa el precio del producto'**])!!}  
 </**div**>  
  
</**div**>  
<**div class="col-md-6"**>  
  
 <**div class="form-group"**>  
 {!!Form::*label*(**'IVA:'**)!!}  
 {!!Form::*text*(**'ivaProd'**,**null**,[**'class'**=>**'form-control soloNum'**,**'placeholder'**=>**'Ingresa el IVA del producto'**])!!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!!Form::*label*(**'Código manual:'**)!!}  
 {!!Form::*text*(**'idMProd'**,**null**,[**'class'**=>**'form-control soloNum'**,**'placeholder'**=>**'Ingresa el código manual'**])!!}  
 </**div**>  
 <**div class="form-group"**>  
 {!! Form::*select*(**'favProd'**, **array**(**'pan'** => **'Pan'**, **'bolleria'** => **'Bolleria'**,**'bebida'** =>**'Bebida'**,**'varios'** => **'Varios'**),  
 **null**, [**'class'**=>**'form-control'**, **'placeholder'** => **''**]) !!}  
 </**div**>  
</**div**>

{!!Form::*submit*(**'Registrar'**,[**'class'**=>**'btn btn-primary btn-block'**])!!}  
{!!Form::*close*()!!}

**Request**

En el archivo request podemos hacer que la function authorize haga return true

Y en la function rules hará return a los campos del model que serán obligatorios.

**public function** authorize()  
{  
 **return true**;  
}  
**public function** rules()  
{  
 **return** [  
 *//unique = unico en la tabla* **'id'** => **'required|unique:almacens'**,  
 **'nomProd'** => **'required'**,  
 **'uniProd'** => **'required'**,  
 **'precioProd'** => **'required'**,  
 **'ivaProd'** => **'required'** ];  
}

**Controller**

Para insertar datos en la tabla usamos la funcion store del controlador y de parametro pasamos un request, puede ser el estandar o uno creado.

Para calcular los campos y llamamos por la variable request que nos envian del formulario.

Y con la funcion que nos da laravel insertamos Model::create([‘campoModel’=>$request[‘nameInput’]);

**public function** store(AlmacenCreateRequest $request)  
{  
  
 $precio = *floatval*($request[**'precioProd'**]);  
 $iva = $request[**'ivaProd'**];  
  
 $calNeto = $precio - (($precio \* $iva)/100);  
  
 Almacen::*create*([  
 **'id'** => $request[**'id'**],  
 **'nomProd'** => $request[**'nomProd'**],  
 **'uniProd'** => $request[**'uniProd'**],  
 **'precioProd'** => *floatval*($request[**'precioProd'**]),  
 **'ivaProd'** => $request[**'ivaProd'**],  
 **'netoProd'** => $calNeto,  
 **'idMprod'** => $request[**'idMProd'**],  
 **'favProd'** => $request[**'favProd'**]  
 ]);  
  
 **return** redirect(**'admin/almacen'**)->with(**'message'**,**'Producto creado'**);  
}