Contenido

# Connection of database

Open .env and configure the parameters necessary for the connection of database

# Crud with model eloquent

Configure the file .ev in laravel to connect to the database

Create the models using

php artisan make:model Producto

The models are in app/

The second is you need to put the name of table and quit timestamp

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'producto';

protected $primaryKey = 'idproducto';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

## Relations of model

### One to many

One Tienda has many Productos

class Tienda extends Model

{

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'tienda';

protected $primaryKey = 'idtienda';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

public function productos()

{

return $this->hasMany('App\Producto', 'idtienda');

}

Idtienda is the foreing key

}

//One product belongs to one tienda

class Producto extends Model

{

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'producto';

protected $primaryKey = 'idproducto';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

public function tienda()

{

return $this->belongsTo('App\Tienda','idtienda');

}

'idtienda' is the foreign key

}

### Many to many

One product has many categories and one categories belons to many products

class Producto extends Model

{

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'producto';

protected $primaryKey = 'idproducto';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

public function tienda()

{

return $this->belongsTo('App\Tienda','idtienda');

}

//producto\_categoria is the intermedium table, idproducto and idcategoria are primary and foreign key (Producto and Categoria)

public function categorias()

{

return $this->belongsToMany('App\Categoria','producto\_categoria','idproducto','idcategoria');

}

}

### One to One

It’s the same about was mentioned before

public function config()

{

return $this->hasOne('App\TtrconfigField','ttrfield\_id');

}

public function field()

{

return $this->belongsTo('App\Ttrfield','ttrfield\_id');

}

### hasManyThrough

This situation is when you need to access to another table through another table. For example One User can have many roles. One specific Form is allows for some roles. In this case The user need to move to roles before to get in Form. So

User->roles->form

So

public function formularios()

{

return $this->hasManyThrough(

'App\ttrform\_ttrcargo', 'App\smbd\_etl\_extract\_ttrcargo',

'cedula', 'idttrcargo', 'cedula'

);

}

ttrform\_ttrcargo is the final table and smbd\_etl\_extract\_ttrcargo' is the intermedium table

cedula is the foreign key of the intermedium table and idttrcargo is the foreign key of the final table. The third parameter cedula is the primary key of the first table in this case User.

Whit this you can use something like this

@if(count(Auth::User()->formularios()->get())) //means if the user has formularios

### Joins with model

First in database go to config/database.php and change

'strict' => true to 'strict' => false, this is to allow the group by because if you do not change this you will get an error.

The second thing is to put the join like this example

$formularios = Ttrform::join('ttrform\_ttrcargo','ttrform\_ttrcargo.idttrform','=','ttrform.idttrform')

->join('ttrfieldsf','ttrfieldsf.ttrform\_id','=','ttrform.idttrform')

->join('ttrvalues','ttrvalues.idttrfieldsf','=','ttrfieldsf.idttrfieldsf')

->join('smbd\_etl\_extract\_ttrcargo','smbd\_etl\_extract\_ttrcargo.idttrcargo','=','ttrform\_ttrcargo.idttrcargo')

->where('smbd\_etl\_extract\_ttrcargo.cedula','=',Auth::User()->cedula)

->where('ttrform.active','=',1)

->where('ttrfieldsf.active','=',1)

->where('ttrvalues.active','=',1)

->groupBy('ttrform.idttrform')

->get();

### One important thing.

In controller you have

public function edit($id)

{

$formulario = Ttrform::where('idttrform','=',$id)->first();

$cargos = Cargo::all();

return view('formulario.edit',array('formulario'=>$formulario,'cargos'=>$cargos));

}

In view

@foreach ($formulario->fields as $ttrfield)

after

@foreach ($ttrfield->values as $valor)

In the models is of formulario

public function fields()

{

return $this->hasMany('App\Ttrfield','ttrform\_id');

}

In fields is

public function values()

{

return $this->hasMany('App\Ttrvalue','idttrfieldsf');

}

public function formulario()

{

return $this->belongsTo('App\Ttrform','ttrform\_id');

}

And in values is

public function field()

{

return $this->belongsTo('App\Ttrfield','idttrfieldsf');

}

#### When you have relations you must put the foreign . For example.

One form can have many fields

public function fields()

{

return $this->hasMany('App\Ttrfield','ttrform\_id');

}

Look the foreign key is 'ttrform\_id'. It’s the second parameter

and one field belongs to formulario

public function formulario()

{

return $this->belongsTo('App\Ttrform','ttrform\_id');

}

Look that 'ttrform\_id' is the foreign key of fields

One filed can have many values

public function values()

{

return $this->hasMany('App\Ttrvalue','idttrfieldsf');

}

One value belongs to a field

public function field()

{

return $this->belongsTo('App\Ttrfield','idttrfieldsf');

}

So

@foreach ($formulario->fields as $ttrfield)

@foreach ($ttrfield->values as $valor)

This is the way to call one table that is related with another table.

When you need to call this put

This is other thing

$ttrfield->config->required

Without pharentesis

## Controller

First execute this line in cmd

php artisan make:controller ProductoController --resource

This will create the controller

The second is put this routes/web.php

Route::resource('productos', 'ProductoController');

This will create these default routes as you can see

|  |  |  |  |
| --- | --- | --- | --- |
| GET | /photos | index | photos.index |
| GET | /photos/create | create | photos.create |
| POST | /photos | store | photos.store |
| GET | /photos/{photo} | show | photos.show |
| GET | /photos/{photo}/edit | edit | photos.edit |
| PUT/PATCH | /photos/{photo} | update | photos.update |
| DELETE | /photos/{photo} | destroy | photos.destroy |

Note: If you need to add more route , you must add the route before the resource

Route::get('formularios/formularios\_usuario', 'FormularioController@formularios\_usuario');

Route::resource('formularios', 'FormularioController');

The next is the ProductoController

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Producto as Producto;

use App\Tienda as Tienda;

use App\Categoria as Categoria;

class ProductoController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

### public function index()

{

//

$productos = Producto::all();

return view('producto.index',['productos'=>$productos]);

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

### public function create()

{

//

$tiendas = Tienda::all();

$categorias = Categoria::all();

return view('producto.create',['tiendas'=>$tiendas,'categorias'=>$categorias]);

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

### public function store(Request $request)

{

$producto = new Producto;

$producto->nombre = $request->nombre;

$producto->precio = $request->precio;

$producto->idtienda = $request->tienda;

$producto->save();

$categorias = $request->categoria;

$producto->categorias()->attach($categorias);

//despues de guardar retorne a index

$productos = Producto::all();

return redirect('productos');

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

### public function show($id)

{

$producto = Producto::where('idproducto','=',$id)->first();

return view('producto.delete',['producto'=>$producto]);

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

### public function edit($id)

{

$producto = Producto::where('idproducto','=',$id)->first();

$tiendas = Tienda::all();

$categorias = Categoria::all();

return view('producto.edit',['producto'=>$producto,'tiendas'=>$tiendas,'categorias'=>$categorias]);

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

### public function update(Request $request, $id)

{

$producto = Producto::where('idproducto','=',$id)->first();

$producto->nombre = $request->nombre;

$producto->precio = $request->precio;

$producto->idtienda = $request->tienda;

$producto->save();

$categorias = $request->categoria;

$producto->categorias()->detach($producto->categorias);

$producto->categorias()->attach($categorias);

//despues de guardar retorne a index

return redirect('productos');

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

### public function destroy($id)

{

$producto = Producto::where('idproducto','=',$id)->first();

$producto->categorias()->detach();

$producto->delete();

//despues de guardar retorne a index

$productos = Producto::all();

return redirect('productos');

}

}

## Views

Create a folder with the name product and create these files

Index.blade.php, create.blade.php, update.blade.php

### Layout

Create a folder with the name layouts and within views and create template.blade.php into layouts

<html>

<head>

<title>App Name - @yield('title')</title>

</head>

<body>

@section('sidebar')

This is the master sidebar.

@show

<div class="container">

@yield('content')

</div>

</body>

</html>

#### Include scripts in template

Put in your layout

@yield('footer\_scripts')

And in the template put

@section('footer\_scripts')

<script src="{{ asset('js/form.js?v=1') }}"></script>

<script src="{{ asset('js/formulario\_ajax.js') }}"></script>

@stop

### Index view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<a href="{{ url('productos/create') }}">Crear producto</a>

<table>

<th>Nombre</th>

<th>Precio</th>

<th>Tienda</th>

<th>Categorias</th>

<th>Acciones</th>

@foreach ($productos as $producto)

<tr>

<td>{{ $producto->nombre }}</td>

<td>{{ $producto->precio }}</td>

<td>{{ $producto->tienda->nombre }}</td>

<td>

@foreach ($producto->categorias as $categoria)

{{ $categoria->nombre }},

@endforeach

</td>

<td>

<a href="{{ url('productos/'.$producto->idproducto.'/edit') }}">Editar</a>

<a href="{{ url('productos/'.$producto->idproducto) }}">Eliminar</a>

</td>

</tr>

@endforeach

</table>

@endsection

### Create view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<form action="{{ url('productos') }}" method="POST">

<div>

<label>Nombre</label>

<input type="text" name="nombre" value="{{ old('nombre') }}">

</div>

<div>

<label>Precio</label>

<input type="text" name="precio" value="{{ old('precio') }}">

</div>

<div>

<label>Tienda</label>

<select name="tienda">

<option value="">Seleccione la tienda</option>

@foreach ($tiendas as $tienda)

<option value="{{ $tienda->idtienda }}">{{ $tienda->nombre }}</option>

@endforeach

</select>

</div>

<div>

<label>Categorias</label>

<select name="categoria[]" multiple>

<option value="">Seleccione la tienda</option>

@foreach ($categorias as $categorias)

<option value="{{ $categorias->idcategoria }}">{{ $categorias->nombre }}</option>

@endforeach

</select>

</div>

<div>

<input type="submit" name="">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

</form>

@endsection

### Edit view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<form action="{{ url('/productos/'.$producto->idproducto) }}" method="POST">

<div>

<label>Nombre</label>

<input type="text" name="nombre" value="{{ $producto->nombre }}">

</div>

<div>

<label>Precio</label>

<input type="text" name="precio" value="{{ $producto->precio }}">

</div>

<div>

<label>Tienda</label>

<select name="tienda">

<option value="">Seleccione la tienda</option>

@foreach ($tiendas as $tienda)

<option @if($tienda->idtienda == $producto->idtienda) selected @endif value = "{{ $tienda->idtienda }}">{{ $tienda->nombre }}</option>

@endforeach

</select>

</div>

<div>

<label>Categorias</label>

<select name="categoria[]" multiple>

<option value="">Seleccione la tienda</option>

@foreach ($categorias as $categoria)

<option value="{{ $categoria->idcategoria }}"

@foreach ($producto->categorias as $categoria\_producto)

@if($categoria\_producto->idcategoria == $categoria->idcategoria) selected @endif

@endforeach

>{{ $categoria->nombre }}</option>

@endforeach

</select>

</div>

<div>

<input type="submit" name="">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

<input name="\_method" type="hidden" value="PATCH">

</form>

@endsection

### Delete view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<form action="{{ url('/productos/'.$producto->idproducto) }}" method="POST">

<div>

<h1>Esta seguro de borrar este producto?</h1>

</div>

<div>

<h2>{{ $producto->nombre }}</h2>

</div>

<div>

<input type="submit" name="" value="borrar">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

<input name="\_method" type="hidden" value="DELETE">

</form>

@endsection

# Validation

public function store(Request $request)

{

$this->validate($request, [

'title' => 'required|unique:posts|max:255',

'body' => 'required',

]);

// The blog post is valid, store in database...

}

## Validate unique in update

'nombre' => 'required|unique:cr\_modules,name,'.$id.',id\_module|max:20',

In this case is necessary to validate a unique name and with id\_module means the id of the table.

## Validation in controller

$validator = Validator::make($request->all(), [

'nombre\_formulario' => 'required',

'formulario\_activo' => 'required',

'cargos[]' => 'required',

]);

if ($validator->fails())

{

$errors = $validator->errors();

foreach ($errors->all() as $message) {

echo "<li>".$message."</li>";

}

return;

}

In template put

@if (count($errors) > 0)

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

## Validation of multiple select

'cargos' => 'required|array|min:1',

## Validation of many input arrays

This will validate all the names that contains name=nombre\_campo[]

'nombre\_campo.\*' => 'required',

## Custom error messsages

$messages = [

'required' => 'El :attribute es requerido.',

];

$validator = Validator::make($request->all(), [

'nombre\_formulario' => 'required',

'formulario\_activo' => 'required',

'cargos' => 'required|array|min:1',

'nombre\_campo.\*' => 'required',

'tipo\_campo.\*' => 'required',

'campo\_activo.\*' => 'required',

'campo\_requerido.\*' => 'required',

],$messages);

# Pagination

$users = App\User::paginate(15);

In the view

<div class="container">

@foreach ($users as $user)

{{ $user->name }}

@endforeach

</div>

{{ $users->links() }}

# Security

The tables that has been used are

Important . It’s required the field password and remember\_token in the user table in order to use the security laravel

But if you want to disable remember token just put in the model user

public function getRememberTokenName()

{

return null; // not supported

}

One user can has many roles.

## Tables

|  |  |
| --- | --- |
| usuarios | CREATE TABLE `usuarios` (  `idusuario` int(11) NOT NULL AUTO\_INCREMENT,  `nombre` varchar(100) NOT NULL,  `email` varchar(255) NOT NULL,  `telefono` varchar(100) DEFAULT NULL,  `password` varchar(255) NOT NULL,  `remember\_token` varchar(255) DEFAULT NULL,  PRIMARY KEY (`idusuario`) ) |

CREATE TABLE `roles` (  
 `idrol` int(11) NOT NULL AUTO\_INCREMENT,  
 `nombre` varchar(100) NOT NULL,  
 PRIMARY KEY (`idrol`)  
)

CREATE TABLE `usuarios\_roles` (  
 `idusuario` int(11) NOT NULL,  
 `idrol` int(11) NOT NULL,  
 PRIMARY KEY (`idusuario`,`idrol`),  
 KEY `idrol` (`idrol`),  
 CONSTRAINT `usuarios\_roles\_ibfk\_1` FOREIGN KEY (`idusuario`) REFERENCES `usuarios` (`idusuario`),  
 CONSTRAINT `usuarios\_roles\_ibfk\_2` FOREIGN KEY (`idrol`) REFERENCES `roles` (`idrol`)  
)

## Models

### User model

<?php

namespace App;

use Illuminate\Notifications\Notifiable;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable

{

use Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'name', 'email', 'password',

];

/\*\*

\* The attributes that should be hidden for arrays.

\*

\* @var array

\*/

protected $hidden = [

'password', 'remember\_token',

];

protected $table = 'usuarios';

protected $primaryKey = 'idusuario';

public $timestamps = false;

public function roles()

{

return $this->belongsToMany('App\Rol','usuarios\_roles','idusuario','idrol');

}

}

### Rol Model

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Rol extends Model

{

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'roles';

protected $primaryKey = 'idrol';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

}

## LogueoController

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Auth;

use Illuminate\Http\Request;

class LogueoController extends Controller

{

/\*\*

\* Handle an authentication attempt.

\*

\* @return Response

\*/

public function authenticate(Request $request)

{

if (Auth::attempt(['email' => $request->usuario, 'password' => $request->clave ],$request->recuerdame)) {

// Authentication passed...

return redirect()->intended('usuarios');

}

else

{

echo "Credenciales incorrectas";

}

}

public function index()

{

return view('login.index');

}

public function cerrar\_sesion()

{

Auth::logout();

return view('login.index');

}

}

### Route

In routes/web.php put

Route::post('autenticar', 'LogueoController@authenticate');

Route::get('cerrar\_sesion', 'LogueoController@cerrar\_sesion');

Route::get('ingreso', 'LogueoController@index');

### Usuario Controller

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\User;

use App\Rol;

use Illuminate\Support\Facades\Auth;

class UsuarioController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

if((!Auth::viaRemember() && !Auth::check()) || (!Auth::User()->roles()->where('nombre' , 'admin')->exists() &&

!Auth::User()->roles()->where('nombre' , 'ver')->exists()))

{

return view('invalido');

}

$usuarios = User::all();

return view('usuario.index',['usuarios'=>$usuarios]);

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

if(!Auth::check() || (!Auth::User()->roles()->where('nombre' , 'admin')->exists() &&

!Auth::User()->roles()->where('nombre' , 'insertar')->exists()))

{

return view('invalido');

}

$roles = Rol::all();

return view('usuario.create',['roles'=>$roles]);

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

if(!Auth::check() || Auth::User()->roles()->whereNotIn('nombre' , ['admin','insertar']))

{

return view('invalido');

}

$usuario = new User;

$usuario->nombre = $request->nombre;

$usuario->email = $request->email;

$usuario->telefono = $request->telefono;

$usuario->password = bcrypt($request->clave);

$usuario->save();

$roles = $request->rol;

$usuario->roles()->attach($roles);

return redirect('usuarios');

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

//

}

}

## Views

### Login view

The login is in the folder Login

@extends('layouts.template1')

@section('content')

<form action="{{ url('autenticar') }}" method="post">

<div>

<label>Usuario</label>

<input type="text" name="usuario">

</div>

<div>

<label>Clave</label>

<input type="password" name="clave">

</div>

<div>

<input type="checkbox" name="recuerdame">Recuerdame

</div>

<div>

<input type="submit" name="">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

</form>

@endsection

### Usuario

#### Index view

@extends('layouts.template1')

@section('content')

@if((Auth::User()->roles()->where('nombre' , 'admin')->exists() ||

Auth::User()->roles()->where('nombre' , 'insertar')->exists()))

<a href="{{ url('usuarios/create') }}">Crear Usuario</a>

@endif

<table>

<th>Nombre</th>

<th>Email</th>

<th>Telefono</th>

<th>Roles</th>

<th>Acciones</th>

@foreach ($usuarios as $usuario)

<tr>

<td>{{ $usuario->nombre }}</td>

<td>{{ $usuario->email }}</td>

<td>{{ $usuario->telefono }}</td>

<td>

@foreach ($usuario->roles as $rol)

{{ $rol->nombre }},

@endforeach

</td>

<td>

<a href="{{ url('usuarios/'.$usuario->idusuario.'/edit') }}">Editar</a>

<a href="{{ url('usuarios/'.$usuario->idusuario) }}">Eliminar</a>

</td>

</tr>

@endforeach

</table>

@endsection

#### Create view

@extends('layouts.template1')

@section('content')

<form action="{{ url('usuarios') }}" method="POST">

<div>

<label>Nombre</label>

<input type="text" name="nombre" value="{{ old('nombre') }}">

</div>

<div>

<label>Email</label>

<input type="text" name="email" value="{{ old('email') }}">

</div>

<div>

<label>Telefono</label>

<input type="text" name="telefono" value="{{ old('telefono') }}">

</div>

<div>

<label>Clave</label>

<input type="password" name="clave" value="{{ old('clave') }}">

</div>

<div>

<label>Roles</label>

@foreach ($roles as $rol)

<input type="checkbox" name="rol[]" value="{{ $rol->idrol }}">{{ $rol->nombre }}

@endforeach

</div>

<div>

<input type="submit" name="">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

</form>

@endsection

## The most important thing

To log in the user is necessary to have

public function authenticate(Request $request)

{

if (Auth::attempt(['email' => $request->usuario, 'password' => $request->clave ],$request->recuerdame)) {

// Authentication passed...

return redirect()->intended('usuarios');

}

else

{

echo "Credenciales incorrectas";

}

}

The log out is

Auth::logout();

$request->recuerdame is a checkbox just to remember the user credentials that are store in cookies.

The permissions are

if((!Auth::viaRemember() && !Auth::check()) || (!Auth::User()->roles()->where('nombre' , 'admin')->exists() &&

!Auth::User()->roles()->where('nombre' , 'ver')->exists()))

This part (!Auth::viaRemember() && !Auth::check()) means if the user does not have remember and is not auth

## Middleware

In model of user add

/\*\*

\* Checks if the user belongs to role.'

\*

\* @param string roleSlug

\* return boolean

\*/

public function hasRole(string $roleSlug)

{

$roles = $roleSlug;

$rolesArray = explode(';',$roles);

$roles = $this->roles()->whereIn('nombre', $rolesArray)->count() > 0;

return $roles;

}

Use the command

php artisan make:middleware CheckRole

The file will be created in App/http/Middleware/CheckRole.php

The code is

<?php

namespace App\Http\Middleware;

use Closure;

class CheckRole

{

public function handle($request, Closure $next, $role)

{

if (auth()->check() && auth()->user()->hasRole ($role)) {

return $next($request);

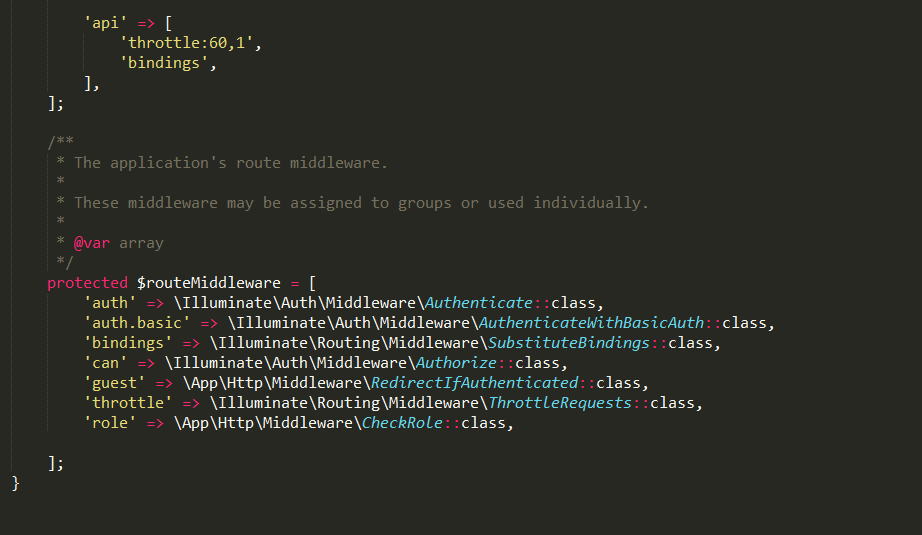
}

return redirect('/');

}

}

You may also register the middleware in app/kernel.php



Adding 'role' => \App\Http\Middleware\CheckRole::class,

Finally choose the web.php and add a middleware

Route::get('usuarios', 'UsuarioController@index')->middleware('role:ver;insertar;admin');

The only roles that have access to this route are ver,inserter,admin. Remember to separate the roles by using ;

In template would be

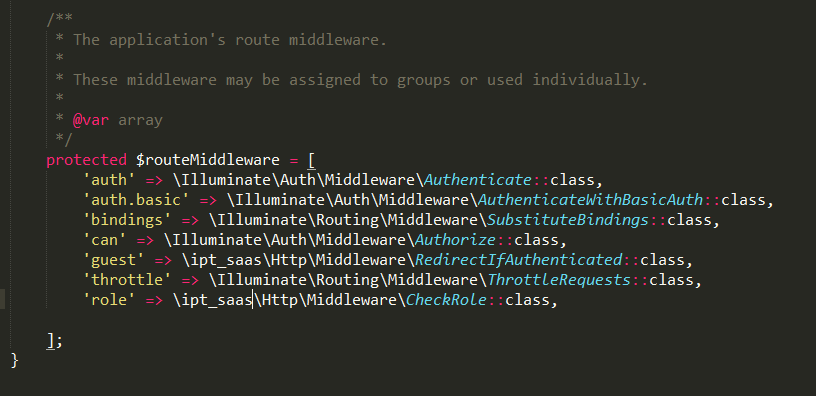
@if(auth()->user()->hasRole('rol-crear;admin'))

<a href="{{ url('usuarios/create') }}">Crear Usuario</a>

@endif

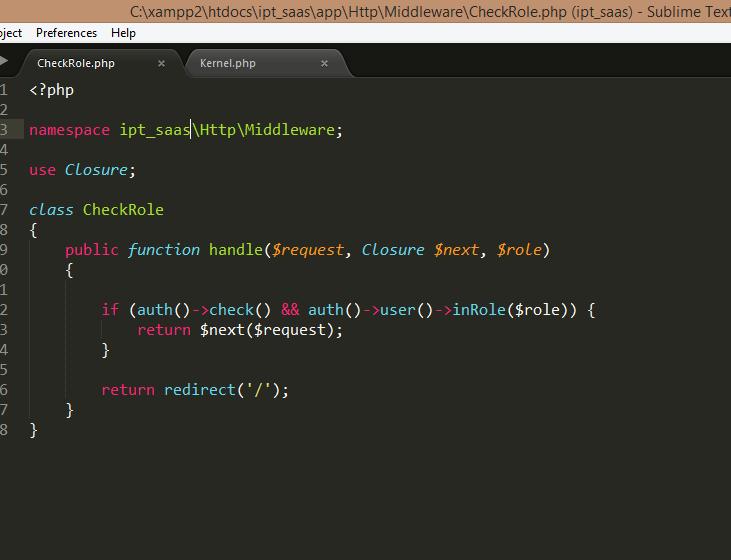
IMPORTANT!!!

If you change the name of your application you must put the correct name.



As you can see there is not route like this: App\Http/Middleware The correct route is \itp\_saas\Http\Middleware

And change the name space. In this case ipt\_saas.



You do not need to put nothing in the controller. This will handles for you.

# Where with relations

$admins = User::whereHas('roles', function ($query) {

$query->where('name', '=', 'admin');

})->get();

Important: **whereHas** Contais just one record meanwhile ‘**with**’ contains the record and other records(all the records including the currently record)

For example

$userByPlanByOrganization = User\_by\_plan\_by\_organization::whereHas('user', function ($query) use ($admin) {

$query->where('id\_user', '=', $admin->id\_user);

But if you need to get related record just put

$organizations = Organization::with('plans\_by\_organization')->whereHas('plans\_by\_organization.plan', function ($query) {

$query->where('name', '=', 'admin');

})->get();

})->first(); // this will get just one record. If you use **with** it will get all the record include admin and superadmin …

$users = App\User::with(['posts' => function ($query) {

$query->where('title', 'like', '%first%');

}])->get();

Other example

$formulario = Ttrform::with(['fields' => function ($query) {

$query->where('active', '=', '1');

}])

->where('idttrform','=',$id)

->first();

## Advance where

Select the client with payment and status of payment payed.

Table client,pagos and estado\_pagos

Cliente::with(['pagos' => function ($query) {

$query->whereHas('estado', function ($query) {

$query->where('nombre\_pago', 'pagado');

});

},

'pagos.estado' => function ($query) {

$query->where('nombre\_pago', 'pagado');

}])

->whereHas('pagos.estado', function ($query) {

$query->where('nombre\_pago', 'pagado');

})

->get();

# Crud with query builder

## Create the controller

First execute this line in cmd

php artisan make:controller ProductoController --resource

This will create the controller

The second is put this routes/web.php

Route::resource('productos', 'ProductoController');

This will create these default routes as you can see

|  |  |  |  |
| --- | --- | --- | --- |
| GET | /photos | index | photos.index |
| GET | /photos/create | create | photos.create |
| POST | /photos | store | photos.store |
| GET | /photos/{photo} | show | photos.show |
| GET | /photos/{photo}/edit | edit | photos.edit |
| PUT/PATCH | /photos/{photo} | update | photos.update |
| DELETE | /photos/{photo} | destroy | photos.destroy |

## Creating the views

Create a folder with the name product and create these files

Index.blade.php, create.blade.php, update.blade.php

### Layout

Create a folder with the name layouts and within views and create template.blade.php into layouts

<html>

<head>

<title>App Name - @yield('title')</title>

</head>

<body>

@section('sidebar')

This is the master sidebar.

@show

<div class="container">

@yield('content')

</div>

</body>

</html>

### Index view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<a href="{{ url('productos/create') }}">Crear producto</a>

<table>

<th>Nombre</th>

<th>Precio</th>

<th>Categoria</th>

<th>Acciones</th>

@foreach ($productos as $producto)

<tr>

<td>{{ $producto->nombre }}</td>

<td>{{ $producto->precio }}</td>

<td>{{ $producto->cnombre }}</td>

<td>

<a href="{{ url('productos/'.$producto->id.'/edit') }}">Editar</a>

<a style="cursor:pointer;" data-t="{{ url('productos/'.$producto->id) }}" data-token="{{ csrf\_token() }}" onclick="eliminar(this)">Eliminar</a>

</td>

</tr>

@endforeach

</table>

@endsection

### Create view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<form action="{{ url('productos') }}" method="POST">

<div>

<label>Nombre del producto</label>

<input type="text" name="nombre" required>

</div>

<div>

<label>precio</label>

<input type="text" name="precio" required>

</div>

<div>

<label>Categoria</label>

<select name="categoria" required>

<option value="">Seleccione.....</option>

@foreach ($categorias as $categoria)

<option value = "{{ $categoria->id }}">{{ $categoria->nombre }}</option>

@endforeach

</select>

</div>

<div>

<input type="submit" name="Crear">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

</form>

@endsection

@if (count($errors) > 0)

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

### Edit view

@extends('layouts.template1')

@section('title', 'Page Title')

@section('sidebar')

@parent

<p>This is appended to the master sidebar.</p>

@endsection

@section('content')

<form action="{{ url('/productos/'.$producto->id) }}" method="POST">

<div>

<label>Nombre del producto</label>

<input type="text" name="nombre" value="{{ $producto->nombre }}" required>

</div>

<div>

<label>precio</label>

<input type="text" name="precio" value="{{ $producto->precio }}" required>

</div>

<div>

<label>Categoria</label>

<select name="categoria" required>

<option value="">Seleccione.....</option>

@foreach ($categorias as $categoria)

<option @if($categoria->id == $producto->idcategoria) selected @endif value = "{{ $categoria->id }}">{{ $categoria->nombre }}</option>

@endforeach

</select>

</div>

<div>

<input type="submit" value="Actualizar">

</div>

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

<input name="\_method" type="hidden" value="PATCH">

</form>

@endsection

@if (count($errors) > 0)

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

## Creating index in controller

Put the scripts that are required

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

In controller index put this

public function index()

{

//

$productos = DB::table('productos')

->join('categorias', 'productos.idcategoria', '=', 'categorias.id')

->select('productos.\*', 'categorias.nombre as cnombre')

->get();

return view('producto.index', ['productos' => $productos]);

}

## Creating insert in controller

public function create()

{

//

$categorias = DB::table('categorias')->get();

return view('producto.create', ['categorias' => $categorias]);

}

## Creating store in controller

public function store(Request $request)

{

//

$nombre = $request->nombre;

$precio = $request->precio;

$idcategoria = $request->categoria;

$this->validate($request, [

'nombre' => 'required',

'precio' => 'required',

'categoria' => 'required',

]);

// si no hay errores en validate se inserta el registro

DB::table('productos')->insert(

['nombre' => $nombre, 'precio' => $precio , 'idcategoria'=> $idcategoria]

);

$productos = DB::table('productos')

->join('categorias', 'productos.idcategoria', '=', 'categorias.id')

->select('productos.\*', 'categorias.nombre as cnombre')

->get();

return view('producto.index', ['productos' => $productos]);

}

## Creating edit in controller

public function edit($id)

{

//

$categorias = DB::table('categorias')->get();

$producto = DB::table('productos')->where('id', $id)->first();

return view('producto.edit', ['producto' => $producto, 'categorias' => $categorias]);

}

## Creating update in controller

public function update(Request $request, $id)

{

//

$nombre = $request->nombre;

$precio = $request->precio;

$idcategoria = $request->categoria;

$this->validate($request, [

'nombre' => 'required',

'precio' => 'required',

'categoria' => 'required',

]);

DB::table('productos')

->where('id', $id)

->update(['nombre' => $nombre,'precio' =>$precio,'idcategoria'=>$idcategoria]);

return view('producto.edit', ['producto' => $producto, 'categorias' => $categorias]);

}

## Creating destroy in controller

public function destroy($id)

{

//

DB::table('productos')->where('id', '=', $id)->delete();

}

### Script

Copy and paste jquery in public/js/jquery.js

Create a script called eliminar.js and within that put

function eliminar(eliminar)

{

$.ajax({

method : 'delete',

url : $(eliminar).attr('data-t'),

data : {

\_token : $(eliminar).data('token')

},

success: function(result) {

// Do something with the result

alert("Registro eliminado");

$(eliminar).parent().parent().remove();

}

});

}

In href of index view there is

<a style="cursor:pointer;" data-t="{{ url('productos/'.$producto->id) }}" data-token="{{ csrf\_token() }}" onclick="eliminar(this)">Eliminar</a>

# Create a rute of specific method of controller

Route::get('photos/popular', 'PhotoController@method');

# Create flash message

In controller put

\Session::flash('flash\_message','Registro guardado');

In teamplate put

@if(session('flash\_message'))

{{session('flash\_message')}}

@endif

# Old values in form

This is just when you send the form and you don’t want that the records will reset

<input type="text" name="name" value="{{ old('name') }}" class="form-control">

THE TYPE SELECT is in this way

<select name="tipo\_documento" required>

<option value="">Seleccione...</option>

<option value="1" @if (old('tipo\_documento') == '1') selected="selected" @endif>Cedula</option>

<option value="2" @if (old('tipo\_documento') == '2') selected="selected" @endif>Rut</option>

<option value="3" @if (old('tipo\_documento') == '3') selected="selected" @endif>Cedula extranjeria</option>

</select>

Remember when is foreach

<select name="curso">

<option value="">Seleccione el curso que desea cursar</option>

@foreach ($cursos as $curso)

<option value="{{ $curso->id }}" @if (old('curso') == $curso->id ) selected="selected" @endif>{{ $curso->fullname }}</option>

@endforeach

</select>

# Query more advance

In your controller put

use ipt\_saas\Models\Sa\Organization;

use ipt\_saas\Models\Sa\Plan\_by\_organization;

use Illuminate\Support\Facades\DB;

the query is

$organizations = Organization::join('sa\_plans\_by\_organization','sa\_plans\_by\_organization.id\_organization','=','sa\_organizations.id\_organization')

->select(DB::raw('\*,sum(sa\_plans\_by\_organization.users\_amount) as total, sa\_organizations.id\_organization'))

->join('sa\_plans','sa\_plans.id\_plan','=','sa\_plans\_by\_organization.id\_plan')

->where('sa\_plans.name','!=','admin')

->groupBy('sa\_organizations.id\_organization')

->get();

In the view is this

@foreach( $organizations as $organization )

<tr>

<td>{{ $organization->business\_name }}</td>

<td>

{{ $organization->total\_users\_amount }}

</td>

<td>0</td>

<td class="btn-tb-right">

<a href="{{ url('super-admin/organizaciones/'.$organization->id\_organization) }}" class="btn btn-white waves-effect waves-light btn-sm">Ver</a></td>

</tr>

@endforeach

To use group By without problem change config/database.php. change 'strict' => false, for 'strict' => true,. This is in the section mysql

## Build the json

$ObjetoCliente = [];

$clientes = Cliente::All();

//$clientes2 = Cliente::with('pagos')->get();

foreach ($clientes as $iCliente => $cliente) {

$ObjetoCliente["cliente"][] = $cliente;

$pagos = Pago:: join('estado\_pagos','pagos.idestado','=','estado\_pagos.idestado')

->where('idcliente',$cliente->idcliente)

->where('estado\_pagos.nombre\_pago','=','no valido')

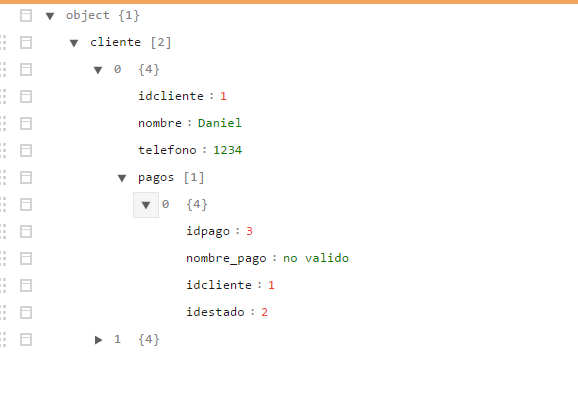
->get();

// $ObjetoCliente[] = $pagos;

$ObjetoCliente["cliente"][$iCliente]["pagos"]= $pagos;

}

return json\_encode($ObjetoCliente);



# Migrations

First execute the command. This will create a table in database called migrations.

php artisan migrate:install // this is no necesary

The next command execute rollback. This is for revert the last change that was executed

php artisan migrate:rollback

For create a migrations put

php artisan make:migration name\_of\_table

The code above will create a new file calle name\_of\_table. This is a new migration.

It’s better to put

php artisan make:migration create\_users\_table --create=users

This will create the file but with code.

This code

php artisan migrate

Will create the tables in the database.

## Foreign key

The first code below with unsigned means that you will make a relation using the field id\_usuario

$table->integer('id\_usuario')->unsigned();

This will create the relation. Imagine alter table… add foreing key references.

$table->foreign('id\_usuario')->references('id')->on('users');

## Drop columns

Create a new migrations with

php artisan make:migration create\_users\_table

Put

public function up()

{

Schema::table('users', function (Blueprint $table) {

$table->dropColumn('status');

});

}

public function down()

{

Schema::table('users', function (Blueprint $table) {

$table->boolean('status')->default(true);

});

}

This will delete the column status from the table users. The function down will revert the changes (rollback). In other words it will create the status because it was deleted.

## Reset

The command below will delete all the tables from database

php artisan migrate:reset

## Refresh

This command will delete all the tables of database and it will create it again

php artisan migrate:refresh

## Adding columns

The table method on the Schema facade may be used to update existing tables. Like the createmethod, the table method accepts two arguments: the name of the table and a Closure that receives a Blueprint instance you may use to add columns to the table:

Schema::table('users', function (Blueprint $table) {

$table->string('email');

});

## Dropping columns

Schema::table('users', function (Blueprint $table) {

$table->dropColumn('email');

});

# Seeding

This will generate all the records that you need on database.

First execute this command

php artisan make:seeder UsersTableSeeder

This will create the file into seed folder.

In the file that was created put

use ipt\_saas\Models\Cr\User;// thiss is no necessary if you use db:table in order to create the table

In public function run put

$users = [

[

'username' => 'sadmin',

'password' => Hash::make("1234"),

'name' => 'Juan',

'second\_name' => 'Carlos',

'lastname' => 'Sanchez',

'email' => 'halirio@gmail.com',

'created\_at' => '2017-06-03',

'id\_user\_status' => 1,

],

[

'username' => 'admin',

'password' => Hash::make("1234"),

'name' => 'Daniel',

'second\_name' => 'Polo',

'lastname' => 'Lopera',

'email' => 'jonathan@globoavans.com',

'created\_at' => '2017-06-03',

'id\_user\_status' => 1,

],

];

DB::table('cr\_users')->truncate();

foreach($roles as $rol) {

DB::table('cr\_roles\_by\_user')->insert($rol);

}

Or use

foreach($users as $user) {

$user = User::create($user);

}

Next open the file databaseSeeder.php in seeds/DatabaseSeeder.php

In the public function run put

Eloquent::unguard();

//disable foreign key check for this connection before running seeders

DB::statement('SET FOREIGN\_KEY\_CHECKS = 0;');

$this->call(RolesTableSeeder::class);

$this->call(UserStatusTableSeeder::class);

$this->call(UsersTableSeeder::class);

DB::statement('SET FOREIGN\_KEY\_CHECKS = 1;');

Finally execute

Phpartisan db:seed

# Testing

First execute this command

composer require laravel/dusk

In App\Providers put into register

use Laravel\Dusk\DuskServiceProvider;

/\*\*

\* Register any application services.

\*

\* @return void

\*/

public function register()

{

if ($this->app->environment('local', 'testing')) {

$this->app->register(DuskServiceProvider::class);

}

}

Next, run the dusk:install Artisan command:

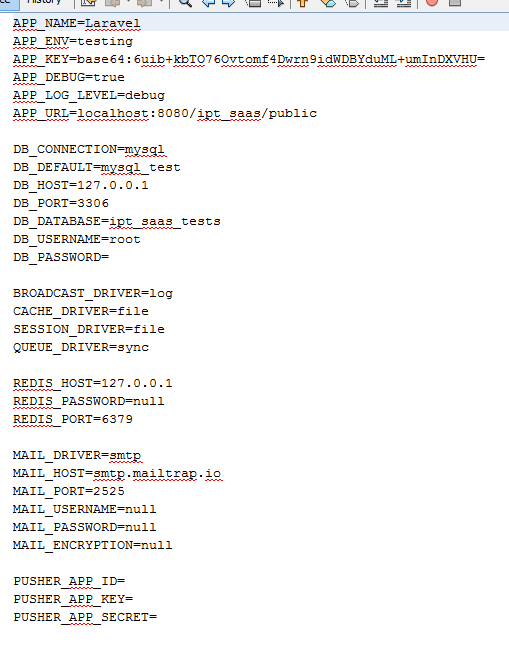
php artisan dusk:install

## Configure env

Frist create this file with this name that is the copy of .env

.env.dusk

Make sure to have this



Change APP\_ENV = testing because in phpunit.xml there is this

<env name="APP\_ENV" value="testing"/>

In config/database.php put this into connections and after mysql

'mysql\_tests' => [

'driver' => 'mysql',

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '3306'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'unix\_socket' => env('DB\_SOCKET', ''),

'charset' => 'utf8mb4',

'collation' => 'utf8mb4\_unicode\_ci',

'prefix' => '',

'strict' => true,

'engine' => null,

],

Look that in .env.dusk there is a name called DB\_DEFAULT=mysql\_test and the database is DB\_DATABASE=ipt\_saas\_tests

## Generating a File test

Put as you like but remember that the last word must be Test

php artisan dusk:make TestExampleTest

The file will be created in Tests\Browser as TestExampleTest

public function testExample()

{

$this->browse(function (Browser $browser) {

$browser->visit('/ingreso')

->type('usuario','santiago5020g@hotmail.com')

->type('clave','1234')

->press('input[type=submit]')

->assertPathIs('/laravel/seguridad/public/usuarios')

->clickLink('Crear Usuario')

->assertPathIs('/laravel/seguridad/public/usuarios/create')

->type('nombre','Rodrigo sa')

->type('email','prueba2@hotmail.com')

->type('telefono','301020')

->type('clave','1234')

->check('div input[name="rol[]"]','1')

->press('input[type=submit]')

->assertPathIs('/laravel/seguridad/public/usuarios')

->assertSee('prueba1@hotmail.com');

});

}

Click in a element

->click("td a");

To run the test put

php artisan dusk