1) Crear el proyecto Laravel (moviles)

composer create-project --prefer-dist laravel/laravel nombre-proyecto-laravel

2) Elegir y descargar la plantilla

Copiar las carpetas js,img,css a las respectivas carpetas dentro de la carpeta public del proyecto laravel

Crear en resources/views/layouts las plantillas master privada y public del proyecto laravel basandose en la plantilla descargada. Se modificarán las rutas de las imágenes , js, enlaces, etc. Ejemplo

<script src="{{ asset('js/app.js') }}" defer></script>

<img src="{{asset('img/h4-slide.png')}}" alt="Slide">

<li class="nav-item">

<a class="nav-link" href="{{ url('webs') }}">

<span data-feather="shopping-cart"></span>

Webs

</a>

</li>

y crearemos las partes variables de la plantilla (titulo,contenido,css,js,etc) : Ejemplo. Para el contenido.

@yield('content')

Crear las carpetas publicas y privadas en donde irán las opciones que heredan de la página maestra

@extends('layouts.public')

@section('titulo')

Bienvenido/a

@endsection

@section('seleccionado1')

selected

@endsection

@section('css')

<!-- insert los css propios de la página -->

@endsection

@section('content')

<!-- insert the page content here -->

<div class="slider-area">

<!-- Slider -->

<div class="block-slider block-slider4">

<ul class="" id="bxslider-home4">

<li>

<img src="{{asset('img/h4-slide.png')}}" alt="Slide">

<div class="caption-group">

<h2 class="caption title">

iPhone <span class="primary">6 <strong>Plus</strong></span>

………

<!-- End main content area -->

@endsection

@section('js')

<!-- insert los js propios de la página -->

@endsection

3) Crear la opción de autenticacion.

$ php artisan make:auth

Cambiar el contenido de la clase ‘AppServiceProvider’ (App/Providers)

use Illuminate\Support\Facades\Schema;

public function boot()

{

//

Schema::defaultStringLength(191);

}

4) Crear roles de usuario y los seeders

Documentación: <https://medium.com/@cvallejo/autenticaci%C3%B3n-de-usuarios-y-roles-en-laravel-5-5-97ab59552d91>

$ php artisan make:model Role -m

Editar la clase **CreateRolesTable** en la carpeta de migrations:

public function up()  
{  
 Schema**::**create('roles', function (Blueprint $table) {  
 $table**->**increments('id');  
 **$table->string('name');  
 $table->string('description');**  
 $table**->**timestamps();  
 });  
}

5) Crear una nueva migración para la tabla dinámica *role\_user*:

$ php artisan make:migration create\_role\_user\_table

Editar la clase **CreateRoleUserTable** en la carpeta database/migrations:

public function up()  
{  
 Schema::create('role\_user', function (Blueprint $table) {  
 $table->increments('id');  
 **$table->integer('role\_id')->unsigned();  
 $table->integer('user\_id')->unsigned();**  
 $table->timestamps();   
 });  
}

Ahora hay que generar una relación ***many-to-many*** entre el ***User*** y el ***Role.***

Abrir el modelo ***User.php*** y agregar el siguiente método:

public function roles()  
{  
 return $this  
 ->belongsToMany('App\Role')  
 ->withTimestamps();  
}

Hacer lo mismo con el modelo ***Role.php***:

public function users()  
{  
 return $this  
 ->belongsToMany('App\User')  
 ->withTimestamps();  
}

Es momento de crear algunos seeders y agregar roles y usuarios en la base de datos:

$ php artisan make:seeder RoleTableSeeder

$ php artisan make:seeder UserTableSeeder

Editar la clase ***RoleTableSeeder*** (se encuentra dentro de la carpeta: database/seeds/) agregando el siguiente código al método ***run***:

<?php

use Illuminate\Database\Seeder;

use App\Role;

class RoleTableSeeder extends Seeder{

public function run(){

$role = new Role();

$role->name = 'superadmin';

$role->description = 'SuperAdministrador';

$role->save();

$role = new Role();

$role->name = 'admin';

$role->description = 'Administrator';

$role->save();

$role = new Role();

$role->name = 'user';

$role->description = 'User';

$role->save();

}

}

Hacer lo mismo con la clase ***UserTableSeeder***:

<?php

use Illuminate\Database\Seeder;

use App\User;

use App\Role;

class UserTableSeeder extends Seeder

{

public function run()

{

$role\_user = Role::where('name', 'user')->first();

$role\_admin = Role::where('name', 'admin')->first();

$role\_superadmin = Role::where('name', 'superadmin')->first();

$user = new User();

$user->name = 'User';

$user->email = 'user@example.com';

$user->email\_verified\_at = now();

$user->password = bcrypt('secret');

$user->remember\_token = Str::random(10);

$user->save();

$user->roles()->attach($role\_user);

$user = new User();

$user->name = 'Admin';

$user->email = 'admin@example.com';

$user->email\_verified\_at = now();

$user->password = bcrypt('secret');

$user->remember\_token = Str::random(10);

$user->save();

$user->roles()->attach($role\_admin);

}

}

Editar la clase ***DatabaseSeeder*** (ubicada en la carpeta: database/seeds/) agregándole el siguiente código al método ***run***:

public function run()  
{  
 // La creación de datos de roles debe ejecutarse primero  
 **$this->call(RoleTableSeeder::class);**

// Los usuarios necesitarán los roles previamente generados  
 **$this->call(UserTableSeeder::class);**  
}

Ya queda poco… está casi todo listo. Ahora es tiempo de correr las migraciones y generar el contenido:

*Hay que asegurarse de tener definidas las variables de entorno en nuestro archivo****.env****relativas a la base de datos que utilizaremos*

$ php artisan migrate:refresh --seed

Abrir el modelo **User.php** y agregar estos tres pequeños métodos:

public function authorizeRoles($roles)  
{  
 if ($this->hasAnyRole($roles)) {  
 return true;  
 }  
 abort(401, 'Esta acción no está autorizada.');  
}

public function hasAnyRole($roles)  
{  
 if (is\_array($roles)) {  
 foreach ($roles as $role) {  
 if ($this->hasRole($role)) {  
 return true;  
 }  
 }  
 } else {  
 if ($this->hasRole($roles)) {  
 return true;  
 }  
 }  
 return false;  
}

public function hasRole($role)  
{  
 if ($this->roles()->where('name', $role)->first()) {  
 return true;  
 }  
 return false;  
}

Abrir el archivo app/Http/Controllers/Auth/***RegisterController.php*** y cambiar el método ***create()*** para definir por defecto el Role para los nuevos Users:

use App\Role;

class RegisterController ...

protected function create(array $data)  
{  
 $user = User::create([  
 'name' => $data['name'],  
 'email' => $data['email'],  
 'password' => bcrypt($data['password']),  
 ]);

$user  
 ->roles()  
 ->attach(Role::where('name', 'user')->first());

return $user;  
}

Por último el paso final. Ahora, todo lo que se necesita es llamar al método authorizeRoles() dentro de la acciones del controlador y pasar el array con los roles de usuario y el nivel de acceso que se desee.

class HomeController extends Controller  
{  
 public function \_\_construct()  
 {  
 $this->middleware('auth');  
 }

public function index(**Request $request**)  
 {  
 $request->user()->authorizeRoles(['user', 'admin']);

return view('home');  
 }

/\*  
 *public function someAdminStuff(Request $request)  
 {  
 $request->user()->authorizeRoles(‘admin’);*

*return view(‘some.view’);  
 }*  
 \*/  
}

Opcional:

Define parámetros de selección para la vista Home (resources/views/):

@if(Auth::user()->hasRole('admin'))  
 <div>Acceso como administrador</div>  
@else  
 <div>Acceso usuario</div>  
@endif

5.-GitHub

- Una vez creado el proyecto podríamos subir el proyecto a GITHUB a través de la herramienta cliente Github Desktop. Los pasos a seguir serían: crear el repositorio github vinculado a una carpeta. Esto se hace creando un nuevo repositario. Por ejemplo ‘Moviles’ y luego el path ‘C:\xampp\htdocs’. Luego habrá que publicar.

Si estaríamos trabajando de forma colaborativa, un usuario cualquiera podría descargarse el proyecto, clonando el proyecto del servidor.

En laravel deberíamos regenerar los archivos del framework desde la carpeta del proyecto lanzando el comando:

**composer install**

Miramos si nos falta el fichero .ENV responsable de las conexiones. En caso de no tenerlo lo copiamos de algún otro proyecto.

Probar.

-