

LARAVEL

Este tutorial muestra como agregar y mostrar notas usando laravel y postgresql.

El tutorial se divide en 7 secciones:

I. Configuración

II. Configuración para la BD

III. Creación de modelos y tablas

IV. Controladores

V. Rutas

VI. Sistema de plantillas y layout con Bootstrap

VII. Paginación, cargo de registros con seeders y model factory

I. CONFIGURACIÓN

1. Verificar que el archivo **sites-available** y **.htaccess** tengan la siguiente configuración :

```
seciti@seciti-hp: /etc/apache2/sites-available
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
seciti@seciti-hp: /var/www/lara... x seciti@seciti-hp: /var/www/lara... x seciti@seciti-hp: /etc/apache2/... x
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/
<Directory />
    Options FollowSymLinks
    AllowOverride All
</Directory>
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>

#DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

"000-default.conf" 44L, 1676C 1,1 Comienzo
```

```
seciti@seciti-hp: /var/www/laravel/public x seciti@seciti-hp: /etc/apache2/sites-available
<IfModule mod_rewrite.c>
    <IfModule mod_negotiation.c>
        #Options MultiViews
        Options Indexes FollowSymLinks MultiViews
    </IfModule>

    RewriteEngine On
    #RewriteBase /laravel
    # Redirect Trailing Slashes If Not A Folder...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^(.*)/$ /$1 [L,R=301]

    # Handle Front Controller...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]

    # Handle Authorization Header
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

II. CONFIGURACIÓN PARA LA BASE DE DATOS

1. Dentro del archivo **database.php** en el directorio **config** configurar el driver de la conexión, por defecto vendrá con mysql.

```
'default' => env('DB_CONNECTION', 'pgsql')
```

2. Configurar el archivo **.env** que se encuentra en la raíz del proyecto.

```
DB_CONNECTION=pgsql
DB_HOST=localhost
DB_PORT=5432
DB_DATABASE=laraveldb
DB_USERNAME=jazz
DB_PASSWORD=2012600221
```

3. Después de configurar , abrir la terminal para realizar la migraciones de las 3 tablas que se crearan por default con el siguiente comando:

```
php artisan migrate
```

```
seciti@seciti-hp:/var/www/laravel$ php artisan migrate
Migration table created successfully.
Migrated: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_100000_create_password_resets_table
seciti@seciti-hp:/var/www/laravel$
```

III. CREACIÓN DE LOS MODELOS Y LAS TABLAS

1. Desde consola ejecutar el siguiente comando para crear el modelo “note”:

```
php artisan make:model Note
```

```
php artisan make:model: Category
```

2. Crear la tabla “notes” para el modelo “note” :

```
php artisan mak:mig create_notes:table --create= notes
```

3. Crear la tabla “categories” para el modelo “category” :

```
php artisan mak:mig create_categories:table --create= categories
```

*Nota: Todas las migraciones que se crean se guardan en **database/migrations**

4. Revisar la BD para verificar que se ha creado la tabla **notes**;

```
laravel> \d notes
```

Column	Type	Table	Modifiers
id	integer	"public.notes"	not null default nextval('notes_id_seq'::regclass)
created_at	timestamp(0) without time zone		
updated_at	timestamp(0) without time zone		

Indexes:

"notes_pkey" PRIMARY KEY, btree (id)

5. Agregar el campo “note” en la tabla “notes” desde el archivo de migración que corresponde con la creación de la tabla :

```
public function up()
{
    Schema::create('notes', function (Blueprint $table) {
        $table->increments('id');

        $table->mediumText('note');
        $table->timestamps();
    });
}
```

6. Agregar el campo “name” en la tabla “categories” desde el archivo de migración que corresponde con la creación de la tabla :

```
public function up()
{
    Schema::create('categories', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name',30);

        $table->timestamps();
    });
}
```

7. Realizar un rollback y volver a migrar las tablas para tener los cambios

```
php artisan migrate:rollback
```

```
php artisan migrate
```

IV. CONTROLADORES

Los controladores, los cuales son una capa en nuestra aplicación que nos permite comunicar las rutas con los modelos y demás clases y servicios para dar una respuesta al usuario.

1. Desde la consola crear un controlador para la clase Notes con el comando:

```
php artisan make:controller nombre_controlador
```

```
seciti@seciti-hp:/var/www/laravel$ php artisan make:controller NotesController
Controller created successfully.
seciti@seciti-hp:/var/www/laravel$
```

2. Una vez creado el controlador , abrimos el archivo ubicado en App/Http/Controllers/NotesController y se agregan las siguientes funciones:

```
class NotesController extends Controller
{
    public function index()
    {
        $notes = Note::paginate(8);
        return view('notes/list', compact('notes'));
    }

    //Metodo get -> mostrar formulario
    public function create()
    {
        return view('notes/create');
    }

    //Metodo post -> procesar formulario
    public function store()
    {
        // Validar formulario

        $this->validate(request(), [
            'note' => ['required' , 'max:100']
        ]);
        //Regresa el array de datos enviado por el formulario mediante helper request
        $data = request()->all();

        //Insertar en la bd
        Note::create($data);

        return redirect()->to('notes');
        #return request()->get('note');
        #return request()->only(['note']);

        //Regresa el array de datos enviado por el formulario mediante de Facades
        #return request::all();
    }
    public function show($note)
    {
        //Cargar una nueva nota
        $note = Note::findOrFail($note);

        //return $note->note;
        return view('notes/details', compact('note'));
    }
}
```

index : Permite listar las notas de la bd, mostrando 8 notas en cada página.
create: Muestra página del formulario para crear una nota.
store: Valida que la nota tenga maximo 100 caracteres o que no sea nula y si la validación se cumple guarda la nota.
show: Muestra una nota en particular.

V. RUTAS

1. Para crear rutas se agregan en el archivo `/app/Http/routes.php`

`Route::get('nombre_url' , 'controlador@funcion')`

- 1er. Muestra el titulo de laravel
- 2da. Muestra la lista de todas las notas
- 3ra. Vista para crear una nueva nota (Método get)
- 4ta. Valida y guarda la nota (Método post)
- 5ta. Muestra una nota en particular

```
<?php

/*
|-----
| Application Routes
|-----
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::get('/', function () {
    return view('welcome');
});

use App\Note ;
Route::get('notes', 'NotesController@index');

Route::get('notes/create', 'NotesController@create');|
Route::post('notes', 'NotesController@store');

Route::get('notes/{note}', 'NotesController@show') -> where('note', '[0-9]+');
```

*Guardar cambios

2. Desde consola se puede listar todos los controladores que se tienen en el proyecto con el comando :

```
php artisan route:list
```

VI. SISTEMA DE PLANTILLAS Y LAYOUT CON BOOSTSTRAP 3

1. Crear una plantilla en la carpeta resources/view

Para usar Bootstrap: copiamos cualquier plantilla y sustituimos los CDN del CSS y JS

<http://getbootstrap.com/getting-started/#examples>

Bootstrap CDN

The folks over at [MaxCDN](#) graciously provide CDN support for Bootstrap's CSS and JavaScript. Just use these [Bootstrap CDN](#) links.

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/
/css/bootstrap.min.css" integrity="sha384-
BVYiISIFeK1dGmJRAkycuHAHRg320mUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css" integrity="sha384-
rHyoN1iRsVX4nD0JutlInGas1CJuC7uwjduW9SVrLvRYooPp2bWYgmJQIXw1/Sp" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHj0MaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNICPD7Txa"
crossorigin="anonymous"></script>
```

Copy

Directivas de Blade para crear layouts:

- **@yield** :directiva para indicar una sección en una plantilla que podemos extender en otras vistas.
- **@extends** : directiva para establecer que dicha vista extiende de otra vista (una plantilla), es decir que usa el contenido de esa vista.
- **@section** : directiva para usar una sección definida por medio de **@yield** en una plantilla.

2. Una vez que se ha creado el layout, se quita el contenido dentro de la clase “container” y se agrega la siguiente directiva **@yield('nombre-sección')**


```

<div class="container">
  <div class="starter-template">
    @yield('content')
  </div>
</div><!-- /.container -->

```

3. Crear la carpeta notes dentro de resources/views y agregar 3 vistas:

1. create.blade.php : Vista para el formulario
2. list.blade.php : Vista para listar las notas
3. details.blade.php : Muestra una nota en particular

4. Vista para el formulario

```

@extends('layout')

@section('content')

    <h2> Formulario </h2>

    <div class="container">
        @include('partials/errors')
        <div>
            <form class="form-horizontal" role="form" method="POST" action="{{ url('notes') }}">
                <div class="form-group">
                    <h3>Formulario</h3>
                    {{ csrf_field() }}
                    <!--input type="hidden" name="_token" value="{{ csrf_token() }}"-->
                    <textarea class="form-control" name="note">{{ old('note') }}</textarea><br>
                    <button type="submit" class="btn btn-info" id="enviar-nota">Submit</button>
                </div>
            </form>
        </div>
    </div>
@endsection

```

{{ csrf_field() }} : Es usado para proteger a los formularios de la aplicación de ataques de tipo CSRF (del inglés Cross-site request forgery)

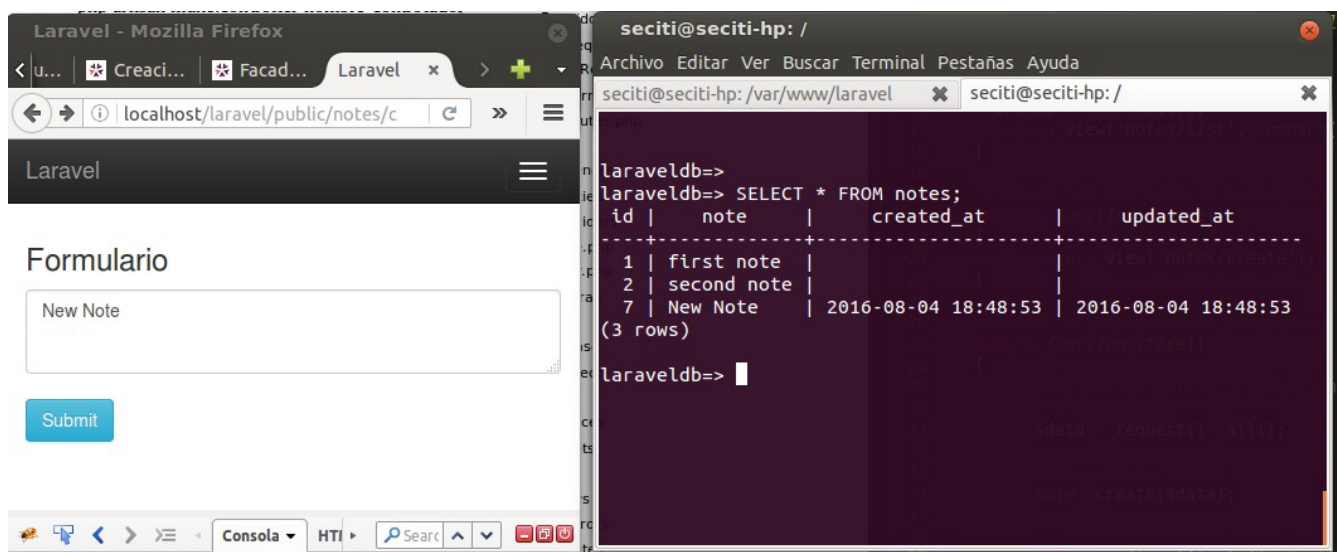
{{old ('note') }} : Al no cumplir la validación permite que el texto ingresado se conserve.

@include('partial/errors') : Archivo que muestra los errores si la validación no se cumple.

5. Crear una carpeta partials, crear el archivo errors.blade.php y agregar el siguiente código :

```
@if(! $errors->isEmpty())
    <div class="alert alert-danger">
        <p><strong>No es posible agregar la nota debido a los siguientes errores :</strong></p>
        <ul>
            @foreach($errors->all() as $error)
                <li> {{ $error }} </li>
            @endforeach
        </ul>
    </div>
@endif
```

6. Probar los validación desde la web, si la nota tiene máximo 100 caracteres se agrega exitosamente

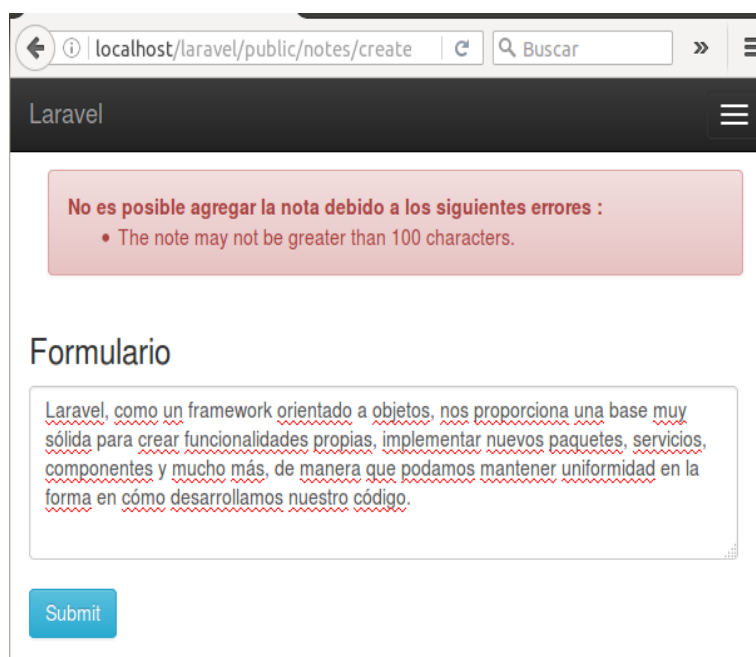


The screenshot shows a web browser window with the URL `localhost/laravel/public/notes/c` and a terminal window. The browser displays a form titled "Formulario" with a text input labeled "New Note" and a "Submit" button. The terminal window shows the command `laravel> SELECT * FROM notes;` and the resulting table:

id	note	created_at	updated_at
1	first note		
2	second note		
7	New Note	2016-08-04 18:48:53	2016-08-04 18:48:53

(3 rows)

De lo contrario mostrara el siguiente error:



The screenshot shows a web browser window with the URL `localhost/laravel/public/notes/create`. The browser displays an error message in a red box:

No es posible agregar la nota debido a los siguientes errores :

- The note may not be greater than 100 characters.

Below the error message is a form titled "Formulario" with a text input containing the text "Laravel, como un framework orientado a objetos, nos proporciona una base muy sólida para crear funcionalidades propias, implementar nuevos paquetes, servicios, componentes y mucho más, de manera que podamos mantener uniformidad en la forma en cómo desarrollamos nuestro código." and a "Submit" button.

7. Vista para mostrar las notas

```
@extends('layout')

@section('content')
<h2 class="title">Notas</h2>
<div class="container">
  <h2 class="title">Notas</h2>
  <div>
    <ul class="list-group">
      @foreach ($notes as $note)

        <li class="list-group-item">
          <span class="label label-info"> {{ $note->category->name }}</span><br>
          {{ substr ($note->note,0,120)}}...
          <a href="{{ url('notes/'.$note->id)}}" class="small">Ver nota</a>
        </li>
      @endforeach

    </ul>
    {!! $notes->render() !!}
  </div>
</div>

@endsection
```

Además del código html para la vista, se agrega un **foreach** que nos permitirá mostrar todas las notas que se tiene en la base de datos

{{ \$note->category->name }} ---> Muestra la categoría a la que pertenece

{{ substr (\$note->note,0,120)}} --->Permite cortar la nota

Dentro del controlador “NotesController” se crea la función “index” que permite traer todas las notas de la bd y realizar la paginación de tal manera que haya 8 notas en cada página.

```
public function index()
{
    $notes = Note::paginate(8);
    return view('notes/list', compact('notes'));
}
```

8. Vista para mostrar una nota en particular

```
@extends('layout')

@section('content')

    <h2> Notas </h2>
    <div class="container">
        <div>
            <h2> Notas </h2>
            {{ $note->note}}

            <div>
                <br>Categoria: <span class="label label-info"> {{ $note->category->name }}</span>
            </div>
        </div>
    </div>
@endsection
```

En el controlador “NotesController” se crea la función “show” que permite mostrar una nota en particular :

```
public function show($note)
{
    //Cargar una nueva nota
    $note = Note::findOrFail($note);

    //return $note->note;
    return view('notes/details',compact('note'));
}
```

VII. PAGINACIÓN, CARGO DE REGISTROS CON SEEDERS Y MODEL FACTORY

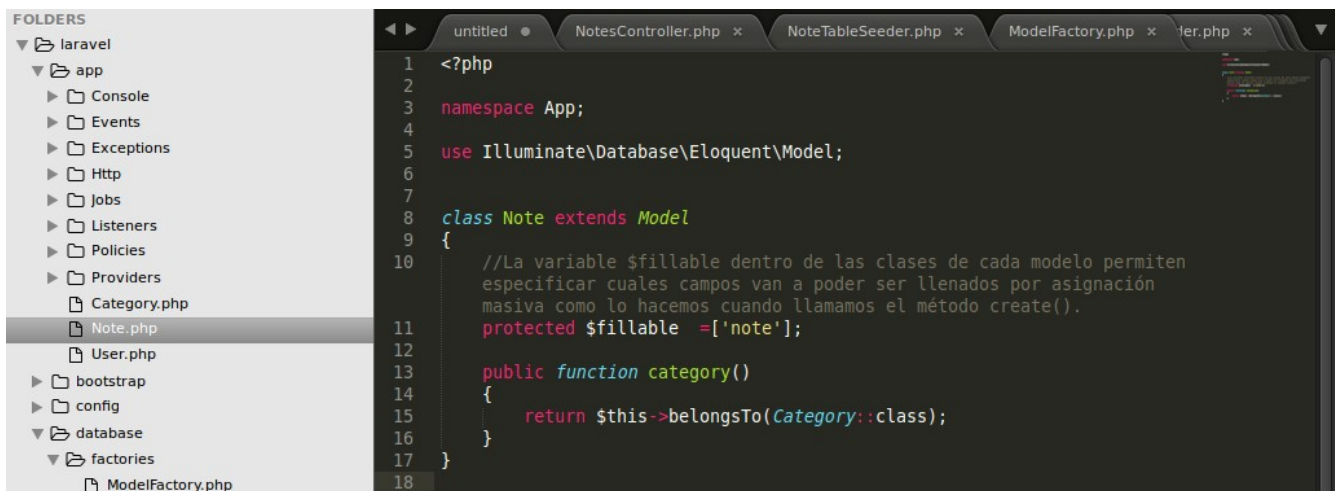
1. Para realizar la paginación se configura la vista y el controlador:

`{!! $notes->render()!!}` -->Dentro la vista “list.blade.php” después de cerrar la lista

`$notes = Note::paginate();` ->Dentro de la función index en el controlador Notes

MODELOS :RELACIONES

2. Para crear la relación **una Nota pertenece a una Categoría**, en el modelo Note creamos la función **category()** de la siguiente manera:



3. Para hacer la relación inversa **una categoría tiene muchas notas**, en el modelo Category creamos el método notes() con lo siguiente:

```
<?php

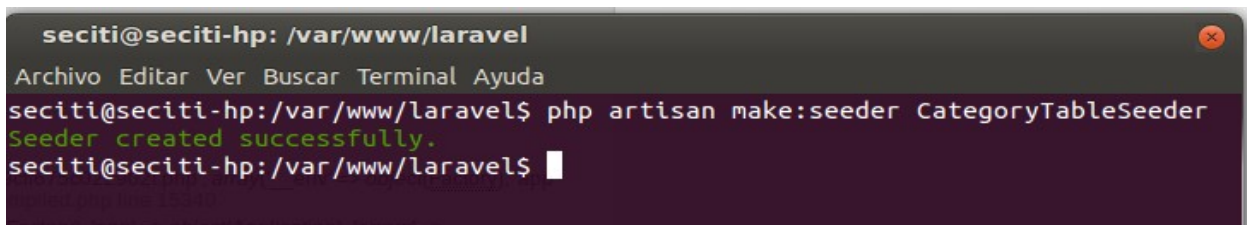
namespace App;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    public function notes()
    {
        return $this->hasMany(Note::class);
    }
}
```

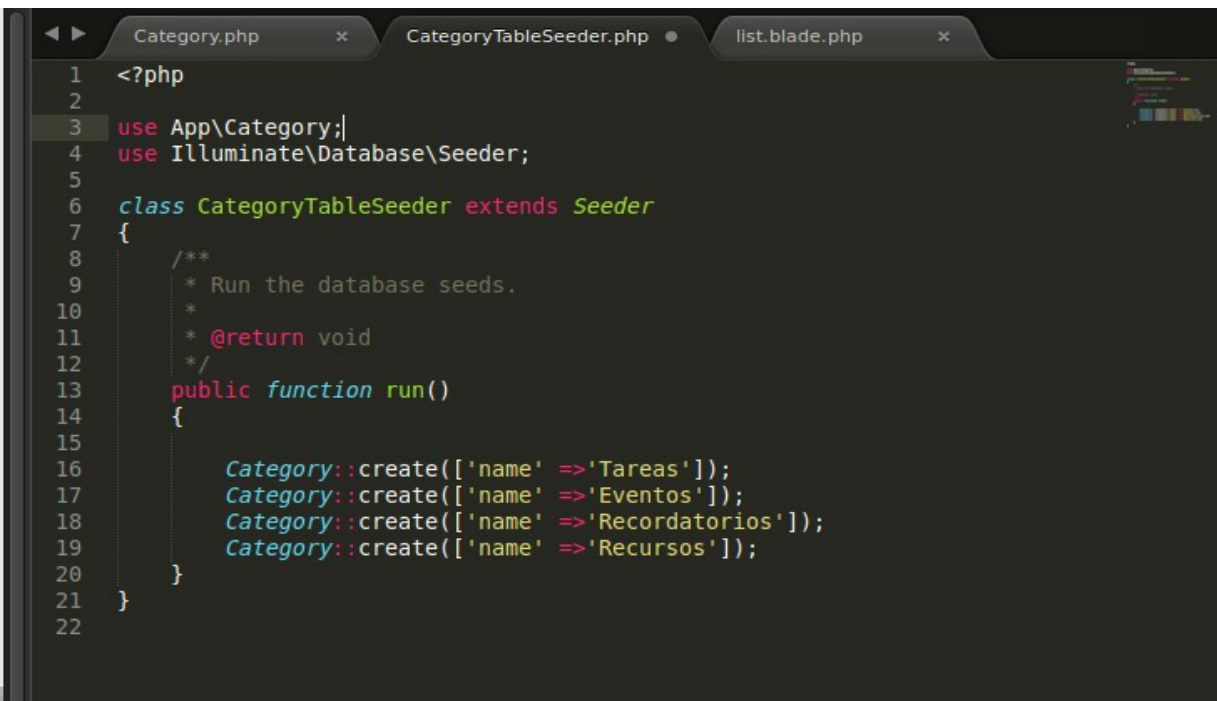
SEEDER

4. Crear un Seeder para Categorías el cual permite crear datos y cargarlos en la bd.



```
seciti@seciti-hp: /var/www/laravel
Archivo Editar Ver Buscar Terminal Ayuda
seciti@seciti-hp:/var/www/laravel$ php artisan make:seeder CategoryTableSeeder
Seeder created successfully.
seciti@seciti-hp:/var/www/laravel$
```

5. Agregar las siguientes categorías



```
Category.php x CategoryTableSeeder.php • list.blade.php x
1 <?php
2
3 use App\Category;
4 use Illuminate\Database\Seeder;
5
6 class CategoryTableSeeder extends Seeder
7 {
8     /**
9      * Run the database seeds.
10     *
11     * @return void
12     */
13     public function run()
14     {
15
16         Category::create(['name' => 'Tareas']);
17         Category::create(['name' => 'Eventos']);
18         Category::create(['name' => 'Recordatorios']);
19         Category::create(['name' => 'Recursos']);
20     }
21 }
22
```

6. Crear un Seeder para notes:

php artisan make:seeder NotesTableSeeder

```
routes.php x NotesController.php x NoteTableSeeder.php ●
<?php
use Illuminate\Database\Seeder;
use App\Note;
use App\Category;
class NoteTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $categories = Category::all();
        $notes = factory(Note::class)->times(20)->make();

        foreach ($notes as $note) {
            $category = $categories->random();
            //Relacion con notas
            $category->notes()->save($note);
        }
    }
}
```

7. Crear un factory para NoteTableSeeder en el archivo database/factories/ModelFactory

```
$factory->define(App\Note::class, function (Faker\Generator $faker) {
    return [
        'note' => $faker->paragraph
    ];
});
```

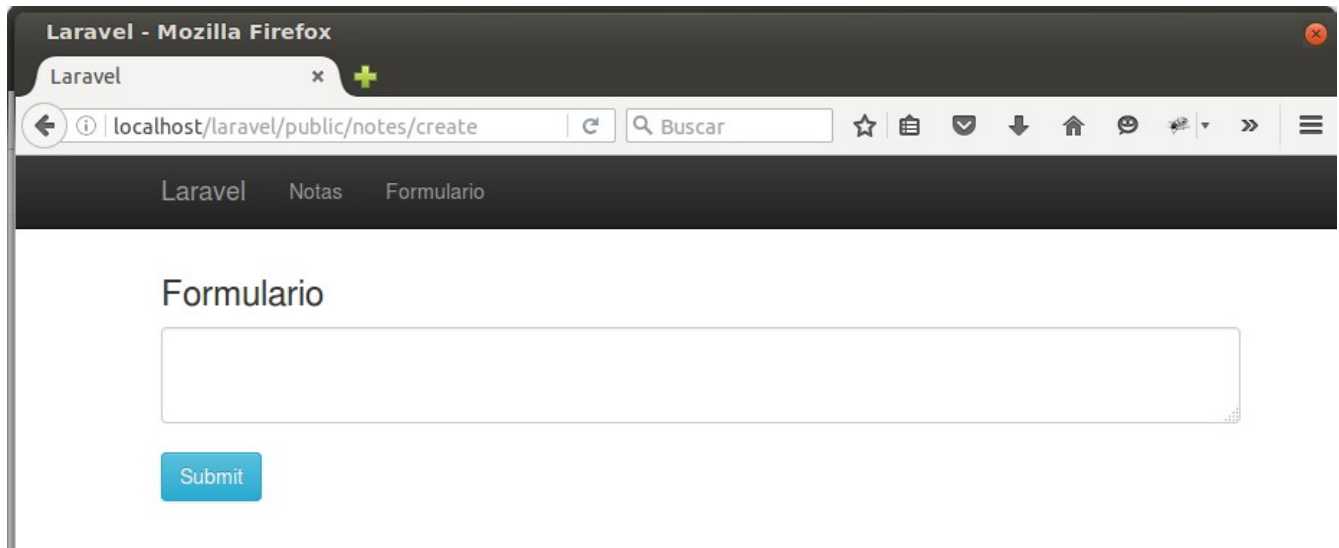
8. Despues de crear NoteTableSeeder, CategoryTableSeeder se deben agregar en el archivo database/seeds/DatabaseSeeder :

```
public function run()
{
    // $this->call(UsersTableSeeder::class);
    $this->call(CategoryTableSeeder::class);
    $this->call(NoteTableSeeder::class);
}
```

9. Desde consola ejecutar el siguiente comando para guardar cambios

```
php artisan migrate:refresh --seed
```

VISTAS



Notas

Recursos

Ut necessitatibus pariatur aperiam nam. Adipisci et dolorum voluptas quis nesciunt odio dolore et.... [Ver nota](#)

Recordatorios

Velit veritatis soluta sunt temporibus. Harum sequi alias nihil. Id pariatur est accusamus eius odit cupiditate.... [Ver nota](#)

Tareas

Voluptatem dolorem veritatis ipsum corporis labore architecto. Quae quam commodi sed ipsa possimus cupiditate. Dolor cor... [Ver nota](#)

Tareas

Est aliquid dolores alias tenetur voluptates sit. Qui reiciendis reiciendis nesciunt molestiae velit asperiores. Exceptu... [Ver nota](#)

Recordatorios

Voluptatem ullam doloribus aut aut enim. Quia et repellendus voluptas recusandae similique natus quo veritatis. Voluptas... [Ver nota](#)

Recordatorios

Corporis molestias deserunt laborum rerum. Repellat tempora quia quo beatae eligendi sed corporis earum. Voluptas et et ... [Ver nota](#)

Recordatorios

Voluptatibus aut fugit rerum odit natus. Illum quasi amet qui et.... [Ver nota](#)

Recursos

Delectus et autem omnis incidunt. Vel praesentium tenetur dignissimos reprehenderit. Dicta suscipit pariatur blanditiis ... [Ver nota](#)

