



Análisis y desarrollo de software.

ID: 2556456



www.sena.edu.co

@SENAComunica

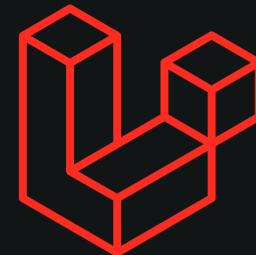
Instructor

Diego Fernando Calderón Silva
Correo: dfcalderon@sena.edu.co

Migraciones



Migraciones En Laravel



Las Migraciones se les conoce como el control de versiones de tu base de datos; de esta forma se puede crear la base de datos y compartir el diseño con el equipo de trabajo.

Si deseas agregar nuevas tablas o columnas a una tabla existente, puedes hacerlo con una nueva migración; si el resultado no fue el deseado, puedes revertir esa migración.

Comandos básicos

`Php artisan migrate`

`Php artisan migrate:rollback`

`Php artisan migrate:rollback --step=1`

`php artisan make:migration agregar_imagen_user`

Migraciones



Corriendo nuestra primera migración.

Las migraciones siempre estarán alojadas en el directorio database.

Laravel por defecto ya tiene unas migraciones creadas, entre ellas destaco a "users"

```
✓ database
  > factories
✓ migrations
  ↗ 2014_10_12_000000_create_users_table.php
  ↗ 2014_10_12_100000_create_password_reset_tokens_table.php
  ↗ 2019_08_19_000000_create_failed_jobs_table.php
  ↗ 2019_12_14_000001_create_personal_access_tokens_table.php
  > seeders
```

Migraciones



Corriendo nuestra primera migración.

Es hora de correr la primera migración, para esto vamos a ir a la consola y estando dentro de nuestro proyecto le daremos el comando:

```
php artisan migrate
```

Y empezara con las migraciones. Cabe resaltar que el servicio de MySql debe estar activo y la bd debe estar configurada en .env

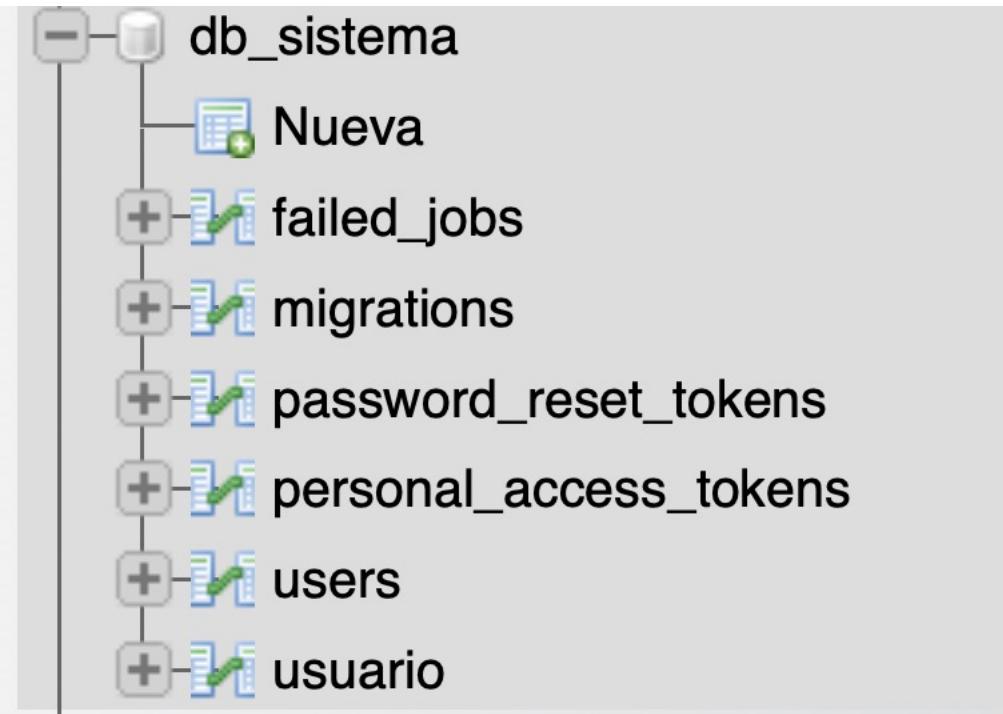
```
[ingdiegoc@MacBook-Pro-de-Diego devstagram % php artisan migrate
INFO  Running migrations.

2014_10_12_00000_create_users_table ..... 39ms DONE
2014_10_12_100000_create_password_reset_tokens_table .. 19ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 13ms DONE
2019_12_14_000001_create_personal_access_tokens_table .. 21ms DONE
ingdiegoc@MacBook-Pro-de-Diego devstagram %
```

Migraciones

Corriendo nuestra primera migración.

Si revisamos nuestra BD en nuestro gestor de preferencia, nos encontraremos que se han creado las tablas que trae integradas laravel por defecto



Migraciones



Corriendo nuestra primera migración.

Ahora, si nos vamos a revisar las migraciones de users, nos encontraremos con que los campos creados son los mismos que estamos validando desde la vista, agregando unos timestamp por defecto junto con el id.

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
}
```

Migraciones



Corriendo nuestra primera migración.

Recordemos que cuando intentamos ingresar un email en su respectivo campo y no es estaba en el formato esperado, laravel nos devolvía un error...

¿Qué tal si volvemos a probar?

E-MAIL

E-mail

El formato del email no es válido.

PASSWORD

Migraciones



Corriendo nuestra primera migración.

Esto sucede debido a que en nuestro controlador estamos dando la orden de que email haga parte de la migración de la tabla users

```
public function store(Request $request) {
    //dd($request->get('name'));
    $this->validate($request, [
        'name'=>'required | max:30',
        'username'=>'required | unique:users|min:3|max:30',
        'email' =>'required | unique:users|email|max:60',
        'password' =>'required'
    ]);
}
```

Migraciones



Corriendo nuestra primera migración.

Terminemos nuestra validación de password, y miremos que pasa si cambio el name de confirmación de password.

Finalmente, usando el comando dd, veamos un mensaje que diga: Creando usuario.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class RegisterController extends Controller
{
    public function index() {
        return view('auth.register');
    }

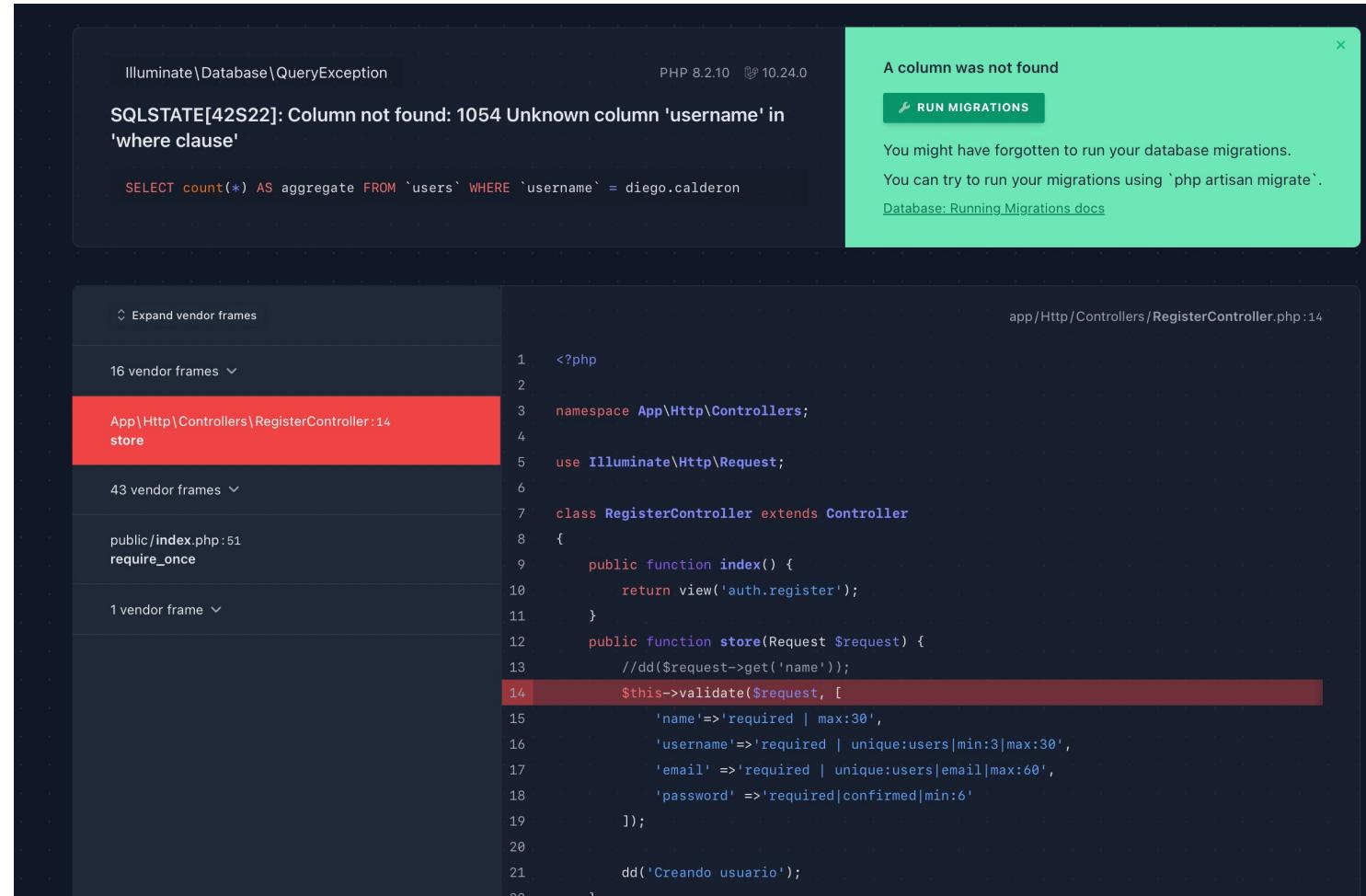
    public function store(Request $request) {
        //dd($request->get('name'));
        $this->validate($request, [
            'name'=>'required | max:30',
            'username'=>'required | unique:users|min:3|max:30',
            'email' =>'required | unique:users|email|max:60',
            'password' =>'required|confirmed|min:6'
        ]);

        dd('Creando usuario');
    }
}
```

Migraciones

Corriendo nuestra primera migración.

Ahora, si lleno todos los datos de mi formulario, y le doy a crear. Evidencio que laravel me indica que el campo username no está creado.



The screenshot shows a developer environment with a terminal window and a code editor.

Terminal Output:

```
Illuminate\Database\QueryException
SQLSTATE[42S22]: Column not found: 1054 Unknown column 'username' in
'where clause'

SELECT count(*) AS aggregate FROM `users` WHERE `username` = diego.calderon
```

Message Box:

A column was not found

Code Editor (app/Http/Controllers/RegisterController.php:14):

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class RegisterController extends Controller
8 {
9     public function index() {
10         return view('auth.register');
11     }
12     public function store(Request $request) {
13         //dd($request->get('name'));
14         $this->validate($request, [
15             'name'=>'required | max:30',
16             'username'=>'required | unique:users|min:3|max:30',
17             'email' =>'required | unique:users@email|max:60',
18             'password' =>'required|confirmed|min:6'
19         ]);
20
21         dd('Creando usuario');
22     }
}
```

Corriendo nuestra primera migración.

Vamos a añadir la migración para username.

Para esto; analicemos primero un poco la pantalla de error que nos da laravel, ahí encontramos que, en la parte oscura, avisa el código de error que se está intentando ejecutar, junto con la cláusula exacta, mientras que, en el lado de color, se evidencia que laravel nos está advirtiendo que tal vez no creamos la migración.



The screenshot shows a terminal window with two distinct sections. On the left, a dark background displays a stack trace and a database query. On the right, a light green background displays an error message from Laravel's error reporting system.

Terminal Output (Left):

```
Illuminate\Database\QueryException
SQLSTATE[42S22]: Column not found: 1054 Unknown column 'username' in
'where clause'
SELECT count(*) AS aggregate FROM `users` WHERE `username` = diego.calderon
```

Laravel Error Message (Right):

A column was not found

[RUN MIGRATIONS](#)

You might have forgotten to run your database migrations.
You can try to run your migrations using `php artisan migrate`.
[Database: Running Migrations docs](#)

Migraciones



Corriendo nuestra primera migración.

Recordemos que las migraciones son un control de versiones de las cosas que vamos haciendo en nuestro proyecto, y es por esta razón que debemos crear una nueva migración para la columna de username así:

```
php artisan make:migration add_username_to_users_table
```

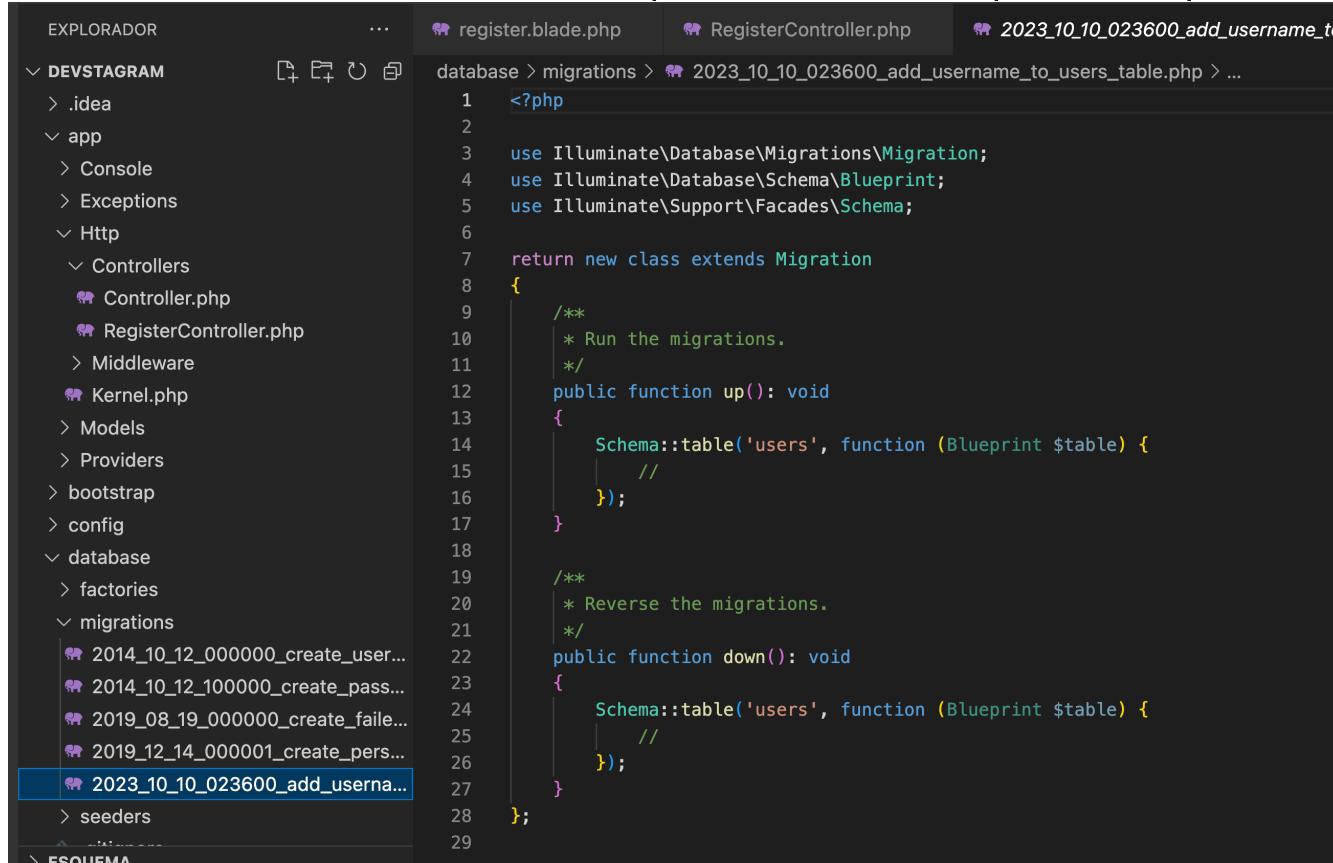
```
[ingdiegoc@MacBook-Pro-de-Diego devstagram % php artisan make:migration add_username_to_users_table
INFO Migration [database/migrations/2023_10_10_023600_add_username_to_users_table.php] created successfully.
```

Si analizamos un poco el comando, notaremos que después de add, se encuentra el nombre del campo que queremos añadir, y seguidamente en "to_users_table" nos hacemos referencia a la tabla en donde queremos crear el nuevo campo

Migraciones

Corriendo nuestra primera migración.

Siguiendo con el análisis, en nuestro directorio de migraciones nos encontramos una nueva migración en donde se destacan los métodos up y down, el up corresponde a migrate mientras que el down al rollback



```
EXPLORADOR ... register.blade.php RegisterController.php 2023_10_10_023600_add_username_to_
DEVSTAGRAM .idea app
> Console
> Exceptions
< Http
  < Controllers
    Controller.php
    RegisterController.php
  > Middleware
    Kernel.php
  > Models
  > Providers
  > bootstrap
  > config
  < database
    > factories
    < migrations
      2014_10_12_000000_create_user...
      2014_10_12_100000_create_pass...
      2019_08_19_000000_create_fai...
      2019_12_14_000001_create_pers...
      2023_10_10_023600_add_username...
    > seeders
  > SCHEMA
register.blade.php
RegisterController.php
2023_10_10_023600_add_username_to_
database > migrations > 2023_10_10_023600_add_username_to_users_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9
10     /**
11      * Run the migrations.
12     */
13     public function up(): void
14     {
15         Schema::table('users', function (Blueprint $table) {
16             //
17         });
18
19     /**
20      * Reverse the migrations.
21     */
22     public function down(): void
23     {
24         Schema::table('users', function (Blueprint $table) {
25             //
26         });
27     }
28 };
29
```

Migraciones



Corriendo nuestra primera migración.

Y dentro del método up, le indicaremos a nuestra migración que campo se va a crear así:

```
public function up(): void
{
    Schema::table('users', function (Blueprint $table) {
        $table->string('username');
    });
}
```

Y así mismo en el down

```
public function down(): void
{
    Schema::table('users', function (Blueprint $table) {
        $table->dropColumn(['username']);
    });
}
```

Migraciones



Corriendo nuestra primera migración.

Vuelva a llenar los campos del formulario y verifique lo que sucede

Migraciones



Corriendo nuestra primera migración.

Evidenciamos nuevamente el error de no haber creado el username, por eso debemos crear la migración de nuevo con el siguiente comando:

```
php artisan migrate
```

Esto correrá la migración, y si abrimos cualquiera de nuestros gestores de base de datos, encontraremos una nueva columna llamada username

#	Nombre	Tipo	Cotejamiento	A
1	id	bigint(20)		U↑
2	name	varchar(255)	utf8mb4_unicode_ci	
3	email	varchar(255)	utf8mb4_unicode_ci	
4	email_verified_at	timestamp		
5	password	varchar(255)	utf8mb4_unicode_ci	
6	remember_token	varchar(100)	utf8mb4_unicode_ci	
7	created_at	timestamp		
8	updated_at	timestamp		
9	username	varchar(255)	utf8mb4_unicode_ci	

Migraciones



Corriendo nuestra primera migración.

Vuelva a llenar los campos del formulario y verifique lo que sucede

Migraciones



Eloquent ORM



Laravel incluye su propio ORM (Object Relacional Mapper) que hace muy sencillo interactuar con tu base de datos.

En Eloquent cada tabla, tiene su propio modelo; ese modelo interactúa únicamente con esa tabla y tiene las funciones necesarias para crear registros, obtenerlos, actualizarlos y eliminarlas.

La forma en que se crea un modelo usando este ORM es:

```
php artisan make:model Cliente
```

Migraciones



Eloquent ORM



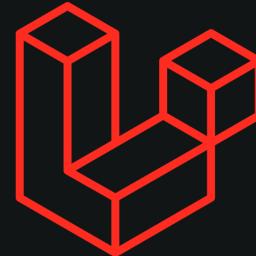
Cuando creas el Modelo Cliente (según el comando enviado en la diapositiva anterior), Eloquent asume que la tabla se va a llamar clientes Si el Modelo se llama Producto; Eloquent espera una tabla llamada productos Puede ser un problema llamar tu modelo Proveedor, porque Eloquent espera la tabla llamada provedors, pero se puede reescribir en el modelo.

El único modelo que viene creado por defecto en laravel es:

User.php

Migraciones

Eloquent ORM



Vamos a empezar a trabajar con nuestro modelo, para lograr hacer consultas a nuestra base de datos, para esto nos vamos al controlador y creamos la siguiente algoritmia:
En la parte alta del código importaremos el modelo asi:

```
use App\Models\User;
```

Luego, dentro del método store después de la validación, usamos un método estatico que se llama create y es el equivalente a INSERT INTO pero usando un modelo en laravel:

```
User::create([
    'name' => $request->name,
    'username' => $request->username,
    'email' => $request->email,
    'password' => $request->password
]);
```

Migraciones



Corriendo nuestra primera migración.

Vuelva a llenar los campos del formulario y verifique lo que sucede

Migraciones



Corriendo nuestra primera migración.

Illuminate\Database\QueryException

PHP 8.2.10 10.24.0

SQLSTATE[HY000]: General error: 1364 Field 'username' doesn't have a default value

```
INSERT INTO
`users` (
`name`,
`email`,
`password`,
`updated_at`,
`created_at`
)
VALUES
(
diego,
diego@info.com,
$ 2y $ 10 $ dCc.EkjYL3HT6T5rD09WOeK1Qldisg2 / NB0Q3QQKKXLxHiRB6d9KG,
2023-10-10 03:08:05,
2023-10-10 03:08:05
)
```

Insertar el usuario con el modelo

Cuando empezamos a trabajar laravel, nos encontramos en los formularios que es un framework enfocado en la seguridad y muestra de ellos fue que detecto un simple ataque cruzado, en esta ocasión está pasando algo similar ya que se crea según la migración todos los campos menos el campo que nosotros registramos, por esto debemos decirle a laravel que nosotros ingresamos un campo, esto para que nosotros seamos conscientes de que datos se ingresan y que datos no.

Para corregir esto iremos a app/Models/User.php y buscaremos la propiedad \$fillable; esta propiedad son los datos que espera laravel que el usuario le dé:

```
protected $fillable = [
    'name',
    'email',
    'password',
];
```

Insertar el usuario con el modelo

Si nos fijamos, no existe username, por lo que lo agregaremos asi:

```
protected $fillable = [
    'name',
    'email',
    'password',
    'username'
];
```

Insertar el usuario con el modelo

Vuelva a llenar los campos del formulario y verifique lo que sucede

Migraciones



Insertar el usuario con el modelo

Aparentemente no pasa nada, pero si vamos a nuestro administrador de base datos, nos encontramos con:

id	name	email	email_verified_at	password	remember_token	created_at	updated_at	username
1	diego	diego@info.com	NULL	\$2y\$10\$ZQuete20HUFuqGlume/Ms.T8.U/0mvsyfLxumb4xNVL... NULL		2023-10-10 03:14:32	2023-10-10 03:14:32	diegokld

Si no aparece tu password con hash, debes agregarlo en el controlador:

```
User::create([
    'name' => $request->name,
    'username' => $request->username,
    'email' => $request->email,
    'password' => Hash::make($request->password)
]);
```

Migraciones

Insertar el usuario con el modelo

Si ingresamos nuevamente un dato, pero esta vez alternamos en mayúsculas y minúsculas veremos que así mismo se guarda en la bd, es por esto que debemos preparar los datos para insertarlos, para esto en nuestro controlador agregaremos la línea

```
use Illuminate\Support\Str;
```

Que nos permite agregar los helper y finalmente escribiremos nuestro código así:

```
User::create([
    'name' => $request->name,
    'username' => Str::lower($request->username),
    'email' => $request->email,
    'password' => Hash::make($request->password)
]);
```

Insertar el usuario con el modelo

Vuelva a llenar los campos del formulario y verifique lo que sucede

Insertar el usuario con el modelo

Revise las diferencias que se enmarcan en la bd cuando se usa lower o slug

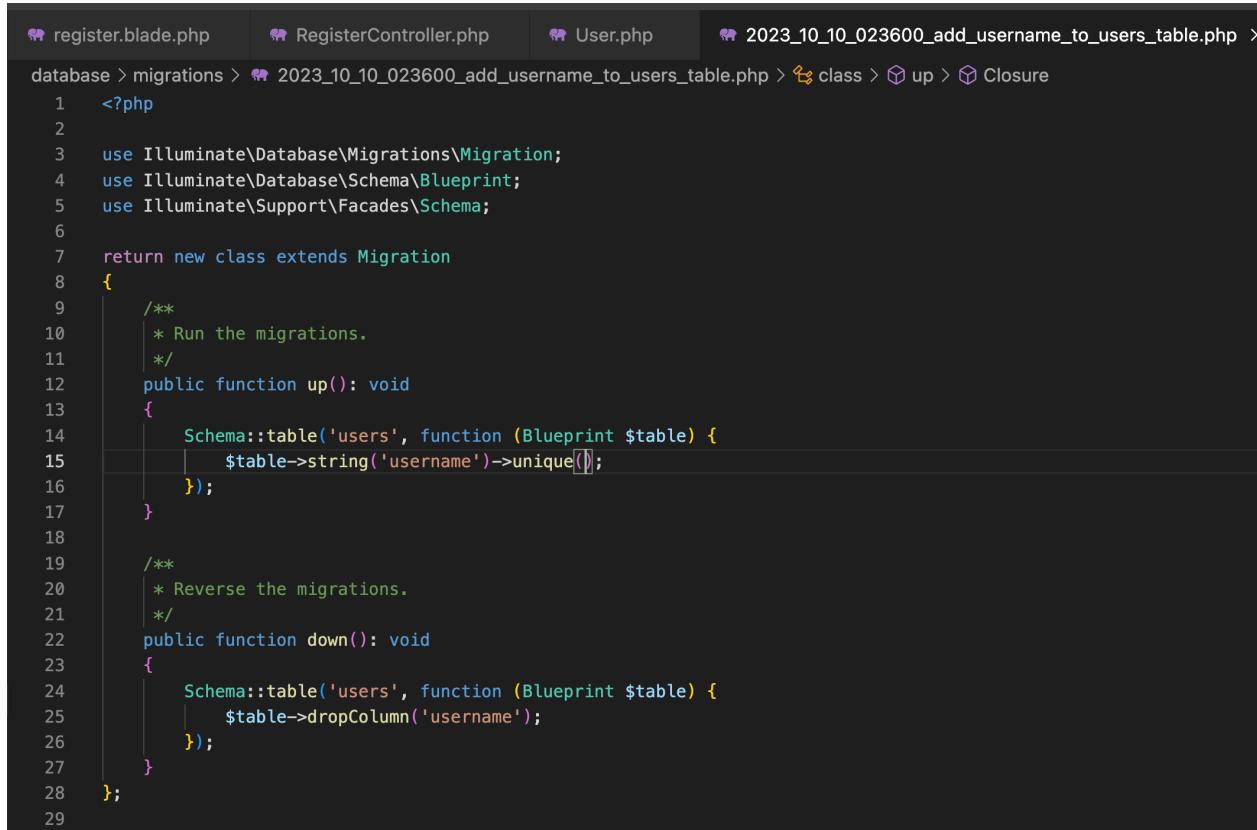
```
User::create([
    'name' => $request->name,
    'username' => Str::lower($request->username),
    'email' => $request->email,
    'password' => Hash::make($request->password)
]);
```

```
User::create([
    'name' => $request->name,
    'username' => Str::slug($request->username),
    启示 'email' => $request->email,
    'password' => Hash::make($request->password)
]);
```

Migraciones

Insertar el usuario con el modelo

Para corregir esto, nos dirigimos al script de la migración de username y en el método up mas específicamente en donde se usa el método string apuntaremos a unique asi:



A screenshot of a code editor showing a PHP migration file named `2023_10_10_023600_add_username_to_users_table.php`. The file contains the following code:

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::table('users', function (Blueprint $table) {
15             $table->string('username')->unique();
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('users', function (Blueprint $table) {
25             $table->dropColumn('username');
26         });
27     }
28 };
29
```

Migraciones



Redireccionar usuario al inicio

Finalmente, luego de crear y listar los datos para insertarlos a la bd, vamos a crear algunos controladores nuevos llamados: Postcontroller y LoginController

```
[ingdiegoc@MacBook-Pro-de-Diego devstagram % php artisan make:controller PostController
    INFO Controller [app/Http/Controllers/PostController.php] created successfully.

[ingdiegoc@MacBook-Pro-de-Diego devstagram % php artisan make:controller LoginController
    INFO Controller [app/Http/Controllers/LoginController.php] created successfully.
```

Y luego en web.php:

```
<?php

use App\Http\Controllers\PostController;
use App\Http\Controllers\RegisterController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

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

Route::get('/register', [RegisterController::class, 'index'])->name('register');
Route::post('/register', [RegisterController::class, 'store']);

Route::get('/muro', [PostController::class, 'index'])->name('post.index');
```

Redireccionar usuario al inicio

Y luego en web.php:

```
<?php

use App\Http\Controllers\PostController;
use App\Http\Controllers\RegisterController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/
Route::get('/', function () {
    return view('principal');
});
Route::get('/register', [RegisterController::class, 'index'])->name('register');
Route::post('/register', [RegisterController::class, 'store']);

Route::get('/muro', [PostController::class, 'index'])->name('post.index');
```

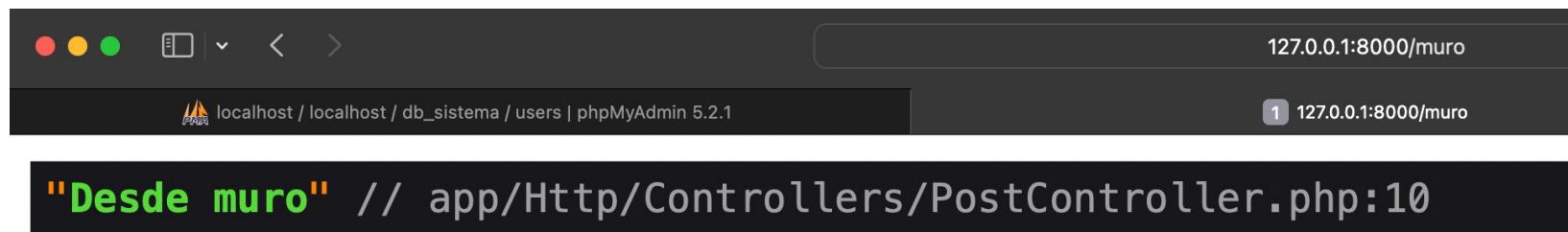
Migraciones

Redireccionar usuario al inicio

Ahora nos dirigimos a PostController:

```
class PostController extends Controller
{
    public function index(){
        dd('Desde muro');
    }
}
```

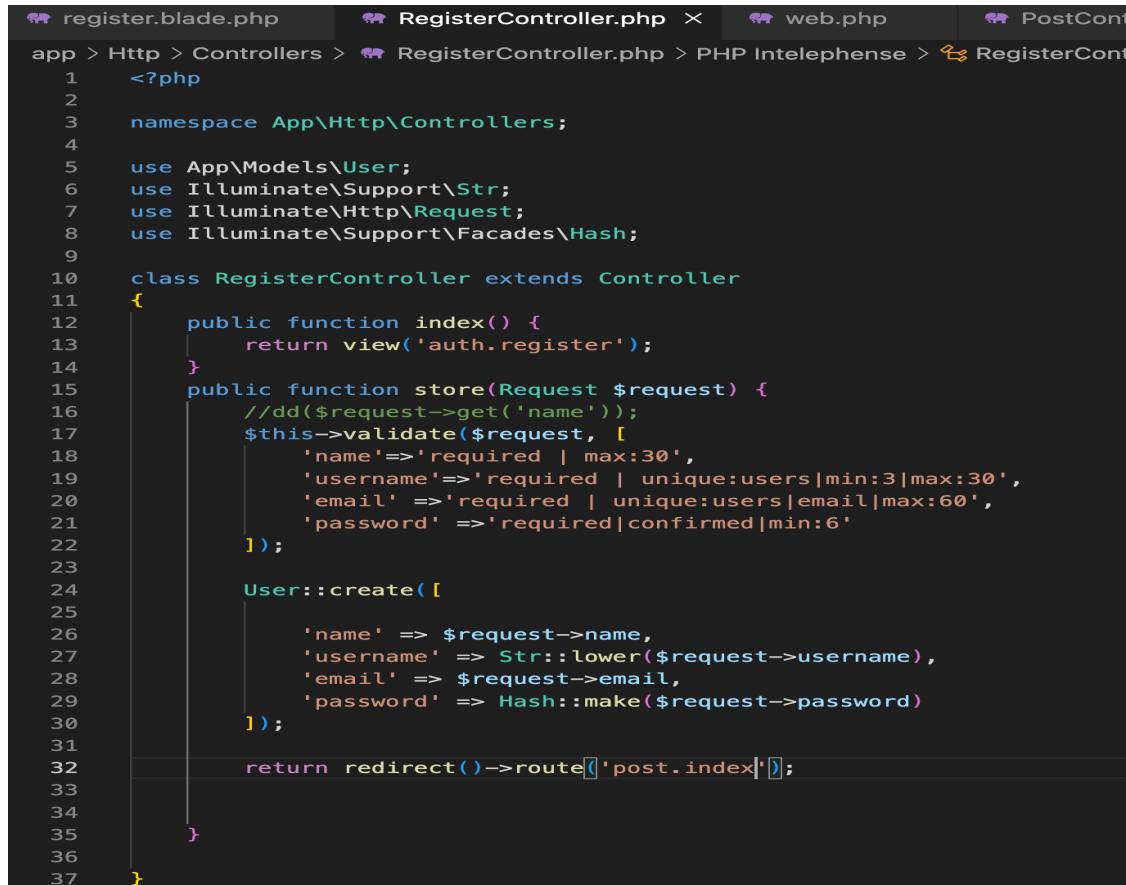
Y si en la ruta del navegador escribimos /muro:



Migraciones

Redireccionar usuario al inicio

Seguidamente nos dirigimos al controlador RegisterController para dar la directriz de redireccionamiento haciendo uso de los helper así:



```
register.blade.php RegisterController.php × web.php PostController.php
app > Http > Controllers > RegisterController.php > PHP Intelephense > RegisterController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\User;
6  use Illuminate\Support\Str;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Hash;
9
10 class RegisterController extends Controller
11 {
12     public function index() {
13         return view('auth.register');
14     }
15     public function store(Request $request) {
16         //dd($request->get('name'));
17         $this->validate($request, [
18             'name'=>'required | max:30',
19             'username'=>'required | unique:users|min:3|max:30',
20             'email' =>'required | unique:users@email|max:60',
21             'password' =>'required|confirmed|min:6'
22         ]);
23
24         User::create([
25             'name' => $request->name,
26             'username' => Str::lower($request->username),
27             'email' => $request->email,
28             'password' => Hash::make($request->password)
29         ]);
30
31         return redirect()->route('post.index');
32
33
34
35     }
36
37 }
```

Redireccionar usuario al inicio

Vuelva a llenar los campos del formulario y verifique lo que sucede



G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682



www.sena.edu.co