LFTS - Forms and Authentication

45. Build a Register User Page

> Como indica el titulo vamo a crear un registro para usuarios y que estos se guarden en la base de datos

```php
use App\Http\Controllers\PostController;
use App\Http\Controllers\RegisterController;

Route::get('/', [PostController::class, 'index'])->name('home');

Route::get('/posts/{post:slug}', [PostController::class, 'show'])->name('post');

Route::get('/register', [RegisterController::class, 'create']);
Route::post('/register', [RegisterController::class, 'store']);
```

resources > views > register > create.blade.php > x-layout

```blade
<x-layout>

    <section class="px-6 py-8">

        <main class="max-w-lg mx-auto mt-10 bg-gray-100 border border-gray-200 p-6 rounded-xl">
            <h1 class="text-center font-bold text-xl">Register</h1>
            <form method="POST" action="/register" class="mt-10">
                @csrf
                <div class="mb-6">
                    <label for="name" class="block mb-2 uppercase font-bold text-xs text-gray-700">Name</label>
                    <input type="text" name="name" id="name" class="border border-gray-400 p-2 w-full" required>
                </div>
                <div class="mb-6">
                    <label for="username" class="block mb-2 uppercase font-bold text-xs text-gray-700">Username</label>
                    <input type="text" name="username" id="username" class="border border-gray-400 p-2 w-full" required>
                </div>
                <div class="mb-6">
                    <label for="email" class="block mb-2 uppercase font-bold text-xs text-gray-700">Email</label>
                    <input type="email" name="email" id="email" class="border border-gray-400 p-2 w-full" required>
                </div>
                <div class="mb-6">
                    <label for="password" class="block mb-2 uppercase font-bold text-xs text-gray-700">Password</label>
                    <input type="password" name="password" id="password" class="border border-gray-400 p-2 w-full" required>
                </div>
                <div class="mb-6">
                    <button type="submit" class="bg-blue-400 text-white rounded py-2 px-4 hover:bg-blue-500">Submit</button>
                </div>
            </form>
        </main>

    </section>

</x-layout>
```

```php
     */

    protected $guarded = [];

    /**
```

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;

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

    public function store()
    {
        $attributes = request()->validate([
            'name' => 'required|max:255',
            'username' => 'required|max:255|min:3',
            'email' => 'required|email|max:255',
            'password' => 'required|min:8|max:255',
        ]);

        User::create($attributes);
        return redirect('/');

    }
}
```

46. Automatic Password Hashing With Mutators

> En este episodio nos enseñan a crear contraseñas encriptadas de forma automatica

```php
public function setPasswordAttribute($password)
{
    $this->attributes['password'] = bcrypt($password);
}
```

```php
    public function store()
    {
        $attributes = request()->validate([
            'name' => 'required|max:255',
            'username' => 'required|max:255|min:3',
            'email' => 'required|email|max:255',
            'password' => 'required|min:8|max:255',
        ]);

        //$attributes['password'] = bcrypt($attributes['password']);

        User::create($attributes);
        return redirect('/');

    }
```

47. Failed Validation and Old Input Data

> En este episodio haces un poco de control de errores y validaciones

```blade
<form method="POST" action="/register" class="mt-10">
    @csrf
    <div class="mb-6">
        <label for="name" class="block mb-2 uppercase font-bold text-xs text-gray-700">Name</label>
        <input type="text" name="name" id="name" class="border border-gray-400 p-2 w-full" value="{{
        @error('name')
            <p class="text-red-500 text-xs mt-1">{{$message}}</p>
        @enderror
    </div>
    <div class="mb-6">
        <label for="username" class="block mb-2 uppercase font-bold text-xs text-gray-700">Username<
        <input type="text" name="username" id="username" class="border border-gray-400 p-2 w-full" v
        @error('username')
            <p class="text-red-500 text-xs mt-1">{{$message}}</p>
        @enderror
    </div>
    <div class="mb-6">
        <label for="email" class="block mb-2 uppercase font-bold text-xs text-gray-700">Email</label
        <input type="email" name="email" id="email" class="border border-gray-400 p-2 w-full" value=
        @error('email')
            <p class="text-red-500 text-xs mt-1">{{$message}}</p>
        @enderror
    </div>
    <div class="mb-6">
        <label for="password" class="block mb-2 uppercase font-bold text-xs text-gray-700">Password<
        <input type="password" name="password" id="password" class="border border-gray-400 p-2 w-ful
        @error('password')
            <p class="text-red-500 text-xs mt-1">{{$message}}</p>
        @enderror
    </div>
    <div class="mb-6">
```

```php
    public function store()
    {
        $attributes = request()->validate([
            'name' => 'required|max:255',
            'username' => 'required|max:255|min:3',
            'username' => ['required', 'min:3', 'max:255', Rule::unique('users', 'username')],
            'email' => 'required|email|max:255|unique:users,email',
        ]);
```

48. Show a Success Flash Message

Ahora vamos a crear un mensage que valide que nuestro usuario fue creado correctamente, obviamente nosotros en este momento tenemos que imnediatamente depues de que el usuario sea registrado pase a la pagina principal, y eso a nosotros nos sirve como validacion, pero al usuario claramente no, aparte dudo que este sea el estado final, falta el login

```php
public function store()
{
    $attributes = request()->validate([
        'name' => 'required|max:255',
        'username' => ['required', 'min:3', 'max:255', Rule::unique('users', 'username')],
        'email' => 'required|email|max:255|unique:users,email',
        'password' => 'required|min:8|max:255',
    ]);
```

resources > views > components > flash.blade.php > ...

```blade
@if (session()->has('success'))
    <div x-data="{show:true}" x-init="setTimeout(()=>show=false, 4000)" x-show="show"
        class="fixed bg-blue-500 text-white py-2 px-4 rounded-xl bottom-3 right-3 text-sm">
        <p>{{session('success')}}</p>
    </div>
@endif
```

```blade
        </footer>
    </section>

    <x-flash />
</body>
```

49.Login and Logout

En este episodio creamos la logica para logear y desloguear de nuestra pagina

```php
Route::get('/register', [RegisterController::class, 'create'])->widdleware('guest');
Route::post('/register', [RegisterController::class, 'store'])->widdleware('guest');
Route::post('/logout', [SessionsController::class, 'destroy']);
```

```blade
<div class="mt-8 md:mt-0 flex items-center">
    @auth
        <span class="text-xs font-bold uppercase">Welcome, {{auth()->user()->name}}!</span>
        <form action="/logout" method="POST" class="text-xs font-semibold text-blue-500 ml-6">
            @csrf
            <button type="submit">Log Out</button>
        </form>
    @else
        <a href="/register" class="text-xs font-bold uppercase">Register</a>
        <a href="/login" class="ml-4 text-xs font-bold uppercase">Login</a>
    @endauth
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class SessionsController extends Controller
{
    public function destroy()
    {
        auth()->logout();
        return redirect('/')->with('success', 'Goodbye!');
    }
}
```

```php
$user = User::create($attributes);
auth()->login($user);
return redirect('/')->with('success', 'Your account has been created');
```

50. Build the Log In Page

> En este episodio vamos a crear todo lo que nos falta para iniciar sesion, esto obviamente seria una vista para el login y la validacion de credenciales

![Visual Studio Code](./images/route%2050.PNGodificacion del controlador de registros")

```blade
<x-layout>

    <section class="px-6 py-8">

        <main class="max-w-lg mx-auto mt-10 bg-gray-100 border border-gray-200 p-6 rounded-xl">
            <h1 class="text-center font-bold text-xl">Log In</h1>
            <form method="POST" action="/login" class="mt-10">
                @csrf
                <div class="mb-6">
                    <label for="email" class="block mb-2 uppercase font-bold text-xs text-gray-700">Email</label>
                    <input type="email" name="email" id="email" class="border border-gray-400 p-2 w-full" value="{{old('email')}}" required>
                    @error('email')
                        <p class="text-red-500 text-xs mt-1">{{$message}}</p>
                    @enderror
                </div>
                <div class="mb-6">
                    <label for="password" class="block mb-2 uppercase font-bold text-xs text-gray-700">Password</label>
                    <input type="password" name="password" id="password" class="border border-gray-400 p-2 w-full" required>
                    @error('password')
                        <p class="text-red-500 text-xs mt-1">{{$message}}</p>
                    @enderror
                </div>
                <div class="mb-6">
                    <button type="submit" class="bg-blue-400 text-white rounded py-2 px-4 hover:bg-blue-500">Submit</button>
                </div>

                {{-- @if ($errors->any())
                    <ul>
                        @foreach ($errors->all() as $error)
                            <li class="text-red-500 text-xs">{{$error}}</li>
                        @endforeach
                    </ul>
                @endif --}}
            </form>
        </main>

    </section>

</x-layout>
```

```php
    public function store()
    {
        $attributes = request()->validate([
            'email' => 'required|email|exists:users,email',
            'password' => 'required',
        ]);

        if(auth()->attempt($attributes)){
            return redirect('/')->with('success', 'Welcome back!');
        }
;
        return back()->withInput()->withErrors(['password' => 'Your provided credentials are incorrect']);

    }

```

```php
Route::get('/login', [SessionsController::class, 'create'])->widdleware('guest');
Route::post('/login', [SessionsController::class, 'store'])->widdleware('guest');

Route::post('/logout', [SessionsController::class, 'destroy'])->widdleware('auth');
```

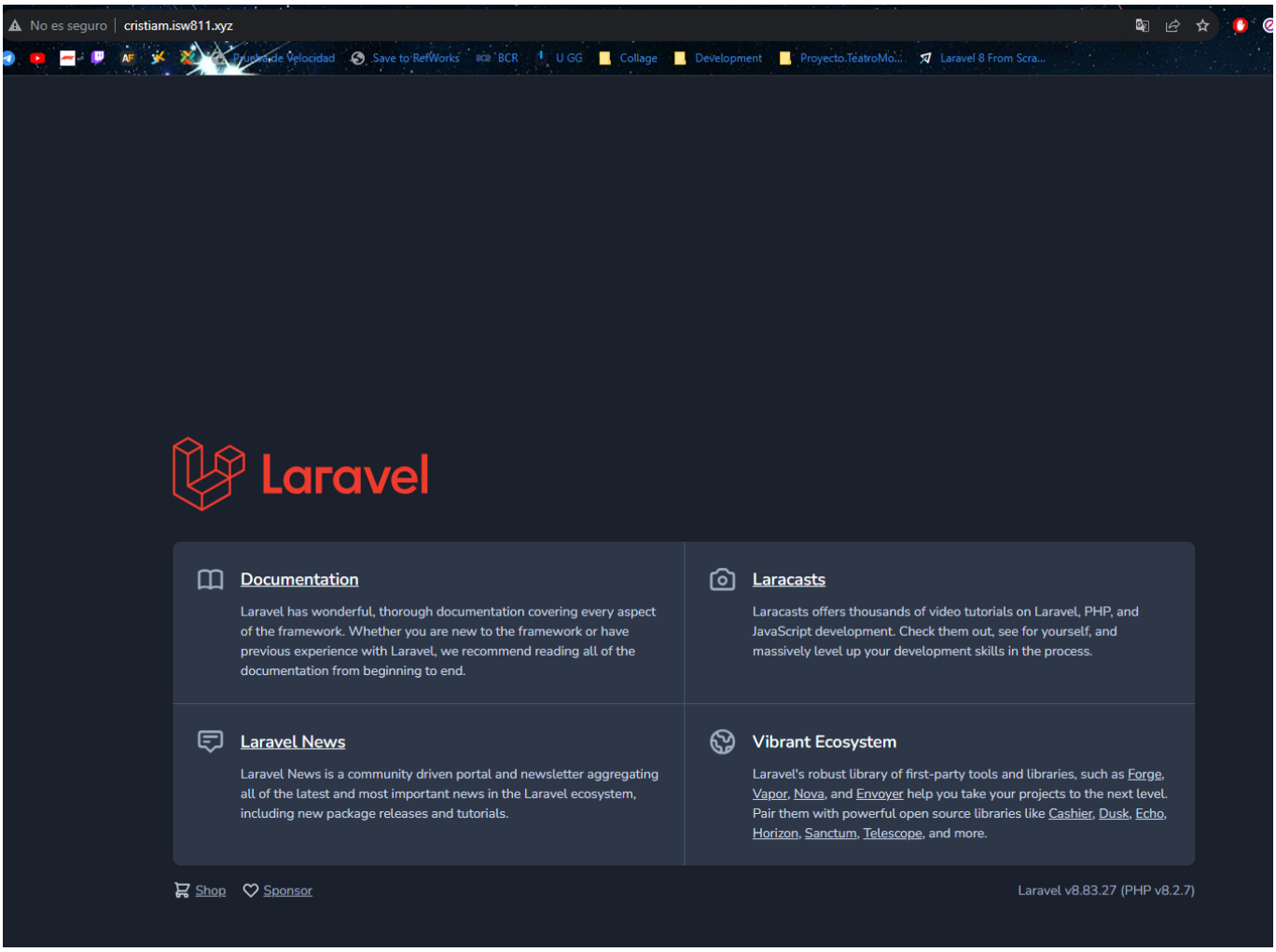
51. Laravel Breeze Quick Peek

> Para este capitulo hay crear un nuevo proyecto y para conveniencia personal voy a utilizar uno de los sitios que creamos al princio del curso para evitar crear otro vhost y demas pasos

```php
public function store()
{
    $attributes = request()->validate([
        'email' => 'required|email|exists:users,email',
        'password' => 'required',
    ]);

    if (!auth()->attempt($attributes)) {
        return back()->withInput()->withErrors(['password' => 'Your provided credentials are incorrect']);
    };

    session()->regenerate();

    return redirect('/')->with('success', 'Welcome back!');
}
```

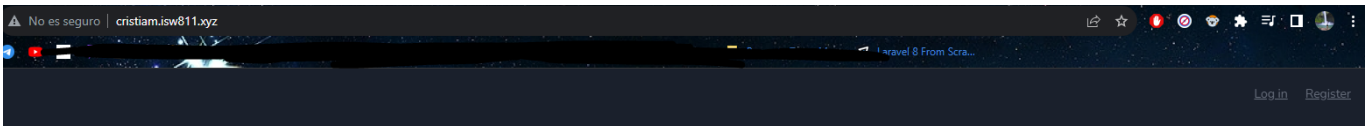
proyecto nuevo

para esto vamos a usar el host virtual que habiamos creados en la primera clase llamado "crisitiam.isw811.xyz" y le vamos a meter laravel



ahora vamos con laravel breeze

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install
npm run dev
```


No es seguro | cristiam.isw811.xyz
Laravel 8 From Scra...
Log in
Register