DOCUMENTACIÓN GTV 2024

¿Qué es GTV?

GTV es una aplicación turística diseñada para cualquier persona que desee, de manera sencilla, descubrir y disfrutar de lo mejor de una ciudad, ya sea que la visite por primera vez o no.

Basada en experiencias previas de otros viajeros y en sus propias visitas anteriores, GTV ofrece un registro mapeado y de acceso rápido a los puntos más destacados de la ciudad. Estos lugares, recomendados por diversos usuarios, son considerados esenciales para visitar durante su estancia.

La aplicación no se limita solo a los típicos monumentos o construcciones impresionantes, sino que también destaca opciones más simples pero igualmente valiosas, como una excelente cafetería para descansar después de un largo viaje o un jardín vibrante para pasear y despejar la mente.

Configuración del proyecto

Dependencias Requeridas:

1. **PHP**: ^8.2.0
    - Versión mínima de PHP requerida para ejecutar el proyecto. PHP 8.2 ofrece mejoras en rendimiento, nuevas características del lenguaje y correcciones de seguridad.
2. **consoletvs/charts**: 6.*
    - Biblioteca para generar gráficos y visualizaciones de datos. Es muy útil para presentar estadísticas y datos de manera visualmente atractiva y comprensible.
3. **enqueue/amqp-bunny**: ^0.10.9
    - Biblioteca para la integración de RabbitMQ usando el cliente Bunny. Facilita la implementación de colas de mensajes para una comunicación eficiente entre diferentes partes del sistema.
4. **fedeisas/laravel-mail-css-inliner**: dev-master
    - Inliner de CSS para emails en Laravel. Permite que los estilos CSS sean incorporados directamente en los emails, asegurando que se vean correctamente en la mayoría de los clientes de correo.
5. **guzzlehttp/guzzle**: ^7.0.1
    - Cliente HTTP para realizar peticiones HTTP. Guzzle es una herramienta potente y flexible que permite enviar solicitudes HTTP de manera sencilla, manejar respuestas, y gestionar cookies y sesiones.
6. **laravel/framework**: ^10.0
    - Framework principal de Laravel. Laravel 10 proporciona una base robusta para el desarrollo de aplicaciones web modernas, con características como enrutamiento, ORM (Eloquent), migraciones, y más.
7. **laravel/jetstream**: ^2.8
    - Paquete de autenticación y gestión de usuarios para Laravel. Jetstream incluye funcionalidades como inicio de sesión, registro, verificación de email, recuperación de contraseñas, y gestión de perfiles de usuario.
8. **laravel/legacy-factories**: ^1.3
    - Soporte para las fábricas de modelos legadas en Laravel. Permite continuar utilizando las fábricas de modelos antiguas en proyectos que han sido actualizados a versiones más nuevas de Laravel.

9. **laravel/sanctum**: ^3.2
    - Sistema de autenticación para APIs en Laravel. Sanctum proporciona una manera sencilla de autenticar usuarios de aplicaciones SPA (Single Page Application), aplicaciones móviles y APIs simples.
10. **laravel/tinker**: ^2.0
    - Herramienta interactiva de consola para Laravel. Tinker permite interactuar con la aplicación desde la línea de comandos, facilitando la ejecución de tareas como pruebas de modelos y consultas a la base de datos.
11. **laravel/ui**: ^4.0
    - Interfaz de usuario para autenticación y scaffolding básico en Laravel. Incluye plantillas básicas y lógica para la autenticación de usuarios.
12. **livewire/livewire**: ^2.5
    - Biblioteca para construir interfaces dinámicas con Laravel. Livewire permite crear componentes interactivos que se actualizan en tiempo real sin necesidad de escribir mucho JavaScript.
13. **simplesoftwareio/simple-qrcode**: ^4.2
    - Generador de códigos QR. Facilita la creación de códigos QR, los cuales pueden ser utilizados para compartir información de forma rápida y eficiente.
14. **spatie/laravel-permission**: ^5
    - Gestión de permisos y roles para Laravel. Este paquete permite asignar roles y permisos a los usuarios de manera sencilla, y controlar el acceso a diferentes partes de la aplicación.
15. **vladimir-yuldashev/laravel-queue-rabbitmq**: 13.3.5
    - Integración de colas de trabajo con RabbitMQ para Laravel. Facilita la gestión de tareas en segundo plano, mejorando la escalabilidad y eficiencia de la aplicación.

Dependencias de Desarrollo:

1. **fakerphp/faker**: ^1.19
    - Biblioteca para generar datos ficticios para pruebas. Muy útil para poblar la base de datos con datos de prueba realistas durante el desarrollo y las pruebas.
2. **mockery/mockery**: ^1.0
    - Biblioteca para crear objetos simulados (mocks) para pruebas unitarias. Ayuda a aislar y probar componentes individuales del código.
3. **nunomaduro/collision**: ^6.1
    - Mejora la experiencia de depuración en consola para Laravel. Proporciona una interfaz clara y comprensible para los errores y excepciones durante el desarrollo.
4. **phpunit/phpunit**: ^9.0

    - Marco de pruebas unitarias para PHP. PHPUnit es una herramienta esencial para escribir y ejecutar pruebas automatizadas, asegurando la calidad y estabilidad del código.
5. **spatie/laravel-ignition**: ^2.0
    - Herramienta de depuración y manejo de excepciones para Laravel. Laravel Ignition mejora la forma en que se presentan y manejan las excepciones, proporcionando información detallada y sugerencias para solucionar errores.

Variables de entorno

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=cookie
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
```

```
MAIL_HOST=sandbox.smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=3500ad23594896
MAIL_PASSWORD=478d0793c002e7
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=no-reply@demomailtrap.com
MAIL_FROM_NAME=GTV


AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Librería Leaflet

¿Qué es Leaflet?

Leaflet es una librería JavaScript que permite crear mapas interactivos de una forma muy sencilla.

Mailtrap

Mailtrap es un servicio en línea utilizado para capturar correos electrónicos enviados desde una aplicación en un entorno de desarrollo. Esto permite a los desarrolladores probar y depurar la funcionalidad de correo electrónico sin enviar correos electrónicos reales a los usuarios.

Documentación del código

Middleware

Middleware AdminOrTeacherOrStudentMiddleware

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class AdminOrTeacherOrStudentMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next):
Response
    {
        if (auth()->user()->hasAnyRole(['Administrador',
'Profesor', 'Alumno'])) {
            return $next($request);
        }
```

```
        return redirect()->route('welcome');     }
}

Route::group(['middleware' => 'admin_or_teacher_or_student'],
function () {
   Route::get('points-of-interest',
ShowPoint::class)->name('points.index');
   Route::get('videos', ListVideos::class)->name('videos.index');
   Route::get('photographies',
Photographies::class)->name('photographies.index');
});
```

Modelos

Modelo Photography

El modelo `Photography` está bien definido para gestionar la información de fotografías, incluyendo su asociación con puntos de interés, áreas temáticas y usuarios que las crean y actualizan. Estas relaciones facilitan una estructura de datos coherente y permiten consultas eficientes sobre las fotografías en el contexto de su aplicación turística.

```php
class Photography extends Model
{
   use HasFactory;

   protected $guarded = ['id', 'created_at', 'updated_at'];

   public function pointOfinterest()
   {
       return $this->belongsTo(PointOfInterest::class);
   }

   public function thematicArea()
   {
       return $this->belongsTo(ThematicArea::class);
   }
```

```
    public function creator()
    {
        return $this->belongsTo(User::class);
    }

    public function updater()
    {
        return $this->belongsTo(User::class);
    }
}
```

Modelo Place

El modelo `Place` está diseñado para gestionar la información sobre lugares, incluyendo su jerarquía, y su asociación con usuarios que los crean y actualizan. También maneja la relación con puntos de interés, permitiendo una estructura de datos organizada y consultas eficientes sobre los lugares y sus características asociadas en el contexto de su aplicación turística.

```
class Place extends Model
{
    use HasFactory;

    protected $guarded = [];

    public function place()
    {
        return $this->belongsTo(Place::class, 'place_id');
    }

    public function creator()
    {
        return $this->belongsTo(User::class, 'creator');
```

```
    }

    public function updater()
    {
        return $this->belongsTo(User::class, 'updater');
    }

    public function pointsOfInterest()
    {
        return $this->hasMany(PointOfInterest::class);
    }
}
```

Modelo PointOfInterest

El modelo `PointOfInterest` está diseñado para gestionar la información sobre los puntos de interés, incluyendo su relación con usuarios, áreas temáticas, fotografías, videos, lugares y visitas. También incluye métodos personalizados para manejar la creación y sincronización de datos, así como para gestionar eventos del ciclo de vida del modelo. Esta configuración permite una gestión robusta y eficiente de los puntos de interés de la aplicación.

```
class PointOfInterest extends Model
{
    use HasFactory;
```

```
    protected $guarded = [];
    protected $dates = ['created_at','updated_at','creation_date',
'last_update_date'];

    public function creator()
    {
        return $this->belongsTo(User::class, 'creator');
    }

    public function updater()
    {
        return $this->belongsTo(User::class, 'updater');
    }

    public function thematicAreas()
    {
        return
$this->belongsToMany(ThematicArea::class)->withPivot('point_of_int
erest_id', 'title', 'description');
    }

    public function photographies()
    {
        return $this->hasMany(Photography::class);
    }

    public function place()
    {
        return $this->belongsTo(Place::class);
    }

    public function videos()
    {
        return $this->hasMany(Video::class);
    }

    public function visits()
    {
        return $this->hasMany(Visit::class);
    }
```

```
    public static function create(array $attributes = [])
    {
        $attributes['creation_date']=Carbon::now();
        $attributes['creator']= auth()->user()->id;

        $pointOfInterest = static::query()->create($attributes);

        return $pointOfInterest;
    }

    public function syncthematicAreas($thematicAreas, $title,
$description)
    {
        $this->thematicAreas()->detach();

        if(!$this->existThematicAreaId($thematicAreas)) {
            $this->thematicAreas()->attach($thematicAreas, [
                'title' => $title,
                'description' => $description,
            ]);
        }

        return
$this->thematicAreas()->updateExistingPivot($thematicAreas, [
            'title' => $title,
            'description' => $description,
        ]);
    }

    public function existThematicAreaId($id)
    {
        return $this->thematicAreas()
            ->where('thematic_area_id', '=', $id)
            ->exists();
    }

    public static function boot()
    {
        parent::boot();

        static::updating(function($pointsofinterest) {
```

```
            $pointsofinterest->last_update_date = Carbon::now();
            $pointsofinterest->updater = auth()->user()->id;
        });

        static::deleting(function($pointOfInterest){
            $pointOfInterest->thematicAreas()->detach();
            $pointOfInterest->photographies()->each(function($p) {
                $p->point_of_interest_id = null;
                $p->save();
            });

            $pointOfInterest->visits()->each(function($v) {
                $v->point_of_interest_id = null;
                $v->save();
            });

            $pointOfInterest->videos()->each(function($v) {
                $v->point_of_interest_id = null;
                $v->save();
            });
        });
    }

    public function scopeAllowed($query)
    {
        if(auth()->user()->can('view', $this)) {
            return $query;
        }else{
            if (auth()->user()->hasRole('Profesor')){
                return $query->where('creator',
auth()->id())->orWhere('updater', auth()->id());
            }
            abort(403);
        }
    }
    public static function countNewPointsOfInterest()
    {
        return
(int)count(PointOfInterest::whereDate('creation_date',
Carbon::today())->get());
    }
```

```
    public static function datesForGrafic(){
        return
PointOfInterest::query()->where('deleted_at','=',null)->whereDate(
'creation_date','>=',
Carbon::now()->subDays(7))->get()->groupBy(function($date) {
            return
Carbon::parse($date->creation_date)->format('d-m-Y' );
        });
    }
}
```

Modelo ThematicArea

El modelo `ThematicArea` está diseñado para gestionar la información sobre áreas temáticas, incluyendo su relación con puntos de interés, fotografías, videos y usuarios. Estas relaciones permiten una estructura de datos coherente y consultas eficientes sobre las áreas temáticas y sus características asociadas en el contexto de su aplicación turística.

```
class ThematicArea extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function pointsOfInterest(){
        return
$this->belongsToMany(PointOfInterest::class)->withPivot('thematic_
area_id', 'title', 'description');
    }

    public function photographies()
    {
        return $this->hasMany(Photography::class);
    }
```

```
    public function videos()
    {
        return $this->hasMany(Video::class);
    }

    public function users()
    {
        return
$this->belongsToMany(User::class)->withPivot(['date', 'active']);
    }
}
```

Modelo User

El modelo `User` está diseñado para gestionar la información de los usuarios y sus interacciones con otros modelos dentro de la aplicación. Incluye funcionalidades avanzadas como autenticación de dos factores, notificaciones y gestión de roles, además de las relaciones con lugares, fotografías, puntos de interés, áreas temáticas y videos, proporcionando una estructura robusta y flexible para la gestión de usuarios de la aplicación.

```
class User extends Authenticatable

{

    use HasRoles, HasApiTokens, HasFactory, HasProfilePhoto,
Notifiable, TwoFactorAuthenticatable;



    protected $guarded = [];

    protected $hidden = [

        'password',
```

```
        'remember_token',
        'two_factor_recovery_codes',
        'two_factor_secret',
    ];
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    protected $appends = [
        'profile_photo_url',
    ];

    public function places()
    {
        return $this->hasMany(Place::class, 'creator');
    }

    public function photographies()
    {
        return $this->hasMany(Photography::class, 'creator');
    }

    public function pointsOfInterest()
    {
```

```
        return $this->hasMany(PointOfInterest::class, 'creator');
    }

    public function thematicAreas()
    {
        return
$this->belongsToMany(ThematicArea::class)->withPivot(['date',
'active']);
    }

    public function videos()
    {
        return $this->hasMany(Video::class, 'creator');
    }
}
```

Modelo Video

El modelo `Video` gestiona la información de los videos en tu aplicación, incluyendo relaciones con elementos de video, usuarios, puntos de interés y áreas temáticas. Esta configuración proporciona una estructura sólida para manejar y consultar videos dentro del contexto de la aplicación.

```php
class Video extends Model
{
    use HasFactory;

    protected $guarded = [];

    public function videoItems()
    {
        return $this->hasMany(VideoItem::class);
    }

    public function creator()
    {
        return $this->belongsTo(User::class, 'creator');
    }

    public function updater()
    {
        return $this->belongsTo(User::class, 'updater');
    }

    public function pointOfInterest()
    {
        return $this->belongsTo(PointOfInterest::class);
    }

    public function thematicArea()
    {
        return $this->belongsTo(ThematicArea::class);
    }
}
```

Modelo VideoItem

El modelo `VideoItem` gestiona la información de los elementos de video en tu aplicación y está diseñado para mantener una relación con los videos correspondientes. Esta estructura

proporciona una manera eficiente de organizar y consultar los elementos de video asociados a cada video de la aplicación.

```php
class VideoItem extends Model
{
    use HasFactory;

    protected $guarded = [];

    public function video()
    {
        return $this->belongsTo(Video::class);
    }
}
```

El modelo `VideoItem` gestiona la información de los elementos de video en tu aplicación y está diseñado para mantener una relación con los videos correspondientes. Esta estructura proporciona una manera eficiente de organizar y consultar los elementos de video asociados a cada video de la aplicación.

Modelo Visit

El modelo `Visit` gestiona las visitas a puntos de interés en tu aplicación, permitiendo la creación, recuperación y manipulación de datos relacionados con estas visitas. Además, incluye métodos personalizados para obtener datos estadísticos sobre las visitas, lo que proporciona una funcionalidad útil para análisis y visualización de la aplicación.

```php
class Visit extends Model
{
```

```
    use HasFactory, SoftDeletes;

    protected $guarded = [];
    protected $dates = ['hour'];

    public static function create(array $attributes = [])
    {
        $visit = static::query()->create($attributes);

        return $visit;
    }

    public function pointOfInterest()
    {
        return $this->belongsTo(PointOfInterest::class);
    }

    public function setHourAttribute($hour)
    {
        $this->attributes['hour'] = $hour ?
Carbon::parse($hour)->format('Y-m-d H:i:s') : null;
    }

    public static function DatesForGrafic()
    {
        return Visit::query()->whereDate('created_at','>=',
Carbon::now()->subDays(7))->get()
            ->groupBy(
                function($date) {
                    return
Carbon::parse($date->created_at)->format('d-m-Y' );
                });
    }

    public static function getPointsOfInterestMostVisit(){
        return
Visit::query()->get()->groupBy('point_of_interest_id')->take(5)->s
ort();
    }
}
```

Vistas

Dashboard

Esta vista define la estructura de una página de dashboard que incluye un encabezado con el título "Dashboard" y un área de contenido principal que contiene una caja estilizada que muestra el contenido del componente `x-jet-welcome`. Este componente `x-jet-welcome` es un componente predefinido de Jetstream que usualmente proporciona una bienvenida y alguna información básica o enlaces útiles para el usuario.

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            {{ __('Dashboard') }}
        </h2>
    </x-slot>

    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-xl
sm:rounded-lg">
                <x-jet-welcome />
            </div>
        </div>
    </div>
</x-app-layout>
```

Navigation-menu

Esta vista organiza y estructura un sistema de navegación completo que es adaptativo a diferentes tamaños de pantalla y maneja tanto la navegación general de la aplicación como la configuración específica del usuario y equipos.

- Configuración Inicial:
- `x-data="{ open: false }"`: Usa Alpine.js para manejar el estado de la navegación responsiva, específicamente para abrir y cerrar el menú.
- Contenedor Principal:
- `class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8"`: Define el contenedor principal con un ancho máximo y margenes horizontales automáticos para centrarlo.
- Secciones de la Barra de Navegación:
- Logo:
    - `shrink-0 flex items-center`: Contiene el logo que enlaza a la ruta del dashboard.
- Enlaces de Navegación:
    - `hidden space-x-8 sm:-my-px sm:ml-10 sm:flex`: Muestra enlaces de navegación (ej. Dashboard) solo en pantallas grandes (ocultos en móviles).
- Menú del Usuario y Equipos:
- Dropdown de Equipos:
    - Si la aplicación tiene características de equipos habilitadas (`Laravel\Jetstream\Jetstream::hasTeamFeatures()`), se muestra un dropdown para gestionar y cambiar entre equipos.
- Dropdown de Configuración del Usuario:
    - Muestra el nombre del usuario o su foto de perfil, y contiene enlaces a la configuración del perfil, tokens API (si están habilitados) y la opción para cerrar sesión.
- Menú Hamburguesa para Móviles:
- `sm:hidden`: Botón que alterna el estado del menú responsivo (abrir/cerrar) en pantallas pequeñas.
- Menú de Navegación Responsivo:
- Se muestra solo en pantallas pequeñas (`sm:hidden`).
- Contiene enlaces a las mismas opciones de navegación, configuración de perfil y gestión de equipos que en la versión de escritorio.
- Formularios y Enlaces Adicionales:
- Formulario para cerrar sesión.
- Opciones para gestionar el equipo y cambiar entre equipos (si las características de equipos están habilitadas).

```
<nav x-data="{ open: false }" class="bg-white border-b
border-gray-100">
    <!-- Primary Navigation Menu -->
    <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
```

```
        <div class="flex justify-between h-16">
            <div class="flex">
                <!-- Logo -->
                <div class="shrink-0 flex items-center">
                    <a href="{{ route('dashboard') }}">
                        <x-jet-application-mark class="block h-9 w-auto"
/>
                    </a>
                </div>

                <!-- Navigation Links -->
                <div class="hidden space-x-8 sm:-my-px sm:ml-10
sm:flex">
                    <x-jet-nav-link href="{{ route('dashboard') }}"
:active="request()->routeIs('dashboard')">
                        {{ __('Dashboard') }}
                    </x-jet-nav-link>
                </div>
            </div>

            <div class="hidden sm:flex sm:items-center sm:ml-6">
                <!-- Teams Dropdown -->
                @if (Laravel\Jetstream\Jetstream::hasTeamFeatures())
                    <div class="ml-3 relative">
                        <x-jet-dropdown align="right" width="60">
                            <x-slot name="trigger">
                                <span class="inline-flex rounded-md">
                                    <button type="button"
class="inline-flex items-center px-3 py-2 border border-transparent
text-sm leading-4 font-medium rounded-md text-gray-500 bg-white
hover:bg-gray-50 hover:text-gray-700 focus:outline-none
focus:bg-gray-50 active:bg-gray-50 transition">
                                        {{
Auth::user()->currentTeam->name }}

                                        <svg class="ml-2 -mr-0.5 h-4
w-4" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"
fill="currentColor">
```

```
                                        <path fill-rule="evenodd"
d="M10 3a1 1 0 01.707.293l3 3a1 1 0 01-1.414 1.414L10 5.414 7.707
7.707a1 1 0 01-1.414-1.414l3-3A1 1 0 0110 3zm-3.707 9.293a1 1 0 011.414
0L10 14.586l2.293-2.293a1 1 0 011.414 1.414l-3 3a1 1 0 01-1.414
0l-3-3a1 1 0 010-1.414z" clip-rule="evenodd" />
                                    </svg>
                                </button>
                            </span>
                        </x-slot>

                        <x-slot name="content">
                            <div class="w-60">
                                <!-- Team Management -->
                                <div class="block px-4 py-2 text-xs
text-gray-400">
                                    {{ __('Manage Team') }}
                                </div>

                                <!-- Team Settings -->
                                <x-jet-dropdown-link href="{{
route('teams.show', Auth::user()->currentTeam->id) }}">
                                    {{ __('Team Settings') }}
                                </x-jet-dropdown-link>

                                @can('create',
Laravel\Jetstream\Jetstream::newTeamModel())
                                    <x-jet-dropdown-link href="{{
route('teams.create') }}">
                                        {{ __('Create New Team') }}
                                    </x-jet-dropdown-link>
                                @endcan

                                <div class="border-t
border-gray-100"></div>

                                <!-- Team Switcher -->
                                <div class="block px-4 py-2 text-xs
text-gray-400">
```

```
                                    {{ __('Switch Teams') }}
                                </div>

                                @foreach (Auth::user()->allTeams()
as $team)
                                    <x-jet-switchable-team
:team="$team" />
                                @endforeach
                            </div>
                        </x-slot>
                    </x-jet-dropdown>
                </div>
            @endif

            <!-- Settings Dropdown -->
            <div class="ml-3 relative">
                <x-jet-dropdown align="right" width="48">
                    <x-slot name="trigger">
                        @if
(Laravel\Jetstream\Jetstream::managesProfilePhotos())
                            <button class="flex text-sm border-2
border-transparent rounded-full focus:outline-none
focus:border-gray-300 transition">
                                <img class="h-8 w-8 rounded-full
object-cover" src="{{ Auth::user()->profile_photo_url }}" alt="{{
Auth::user()->name }}" />
                            </button>
                        @else
                            <span class="inline-flex rounded-md">
                                <button type="button"
class="inline-flex items-center px-3 py-2 border border-transparent
text-sm leading-4 font-medium rounded-md text-gray-500 bg-white
hover:text-gray-700 focus:outline-none transition">
                                    {{ Auth::user()->name }}

                                    <svg class="ml-2 -mr-0.5 h-4
w-4" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"
fill="currentColor">
```

```
                                        <path fill-rule="evenodd"
d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414
1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z" clip-rule="evenodd"
/>
                                    </svg>
                                </button>
                            </span>
                        @endif
                    </x-slot>

                    <x-slot name="content">
                        <!-- Account Management -->
                        <div class="block px-4 py-2 text-xs
text-gray-400">
                            {{ auth()->user()->roles->first()->name
}}
                        </div>

                        <x-jet-dropdown-link href="{{
route('profile.show') }}">
                            {{ trans('Profile') }}
                        </x-jet-dropdown-link>

                        @if
(Laravel\Jetstream\Jetstream::hasApiFeatures())
                            <x-jet-dropdown-link href="{{
route('api-tokens.index') }}">
                                {{ __('API Tokens') }}
                            </x-jet-dropdown-link>
                        @endif

                        <div class="border-t border-gray-100"></div>

                        <!-- Authentication -->
                        <form method="POST" action="{{
route('logout') }}" x-data>
                            @csrf
```

```
                                <x-jet-dropdown-link href="{{
route('logout') }}"

@click.prevent="$root.submit();">
                                    {{ __('Log Out') }}
                                </x-jet-dropdown-link>
                            </form>
                        </x-slot>
                    </x-jet-dropdown>
                </div>
            </div>

            <!-- Hamburger -->
            <div class="-mr-2 flex items-center sm:hidden">
                <button @click="open = ! open" class="inline-flex
items-center justify-center p-2 rounded-md text-gray-400
hover:text-gray-500 hover:bg-gray-100 focus:outline-none
focus:bg-gray-100 focus:text-gray-500 transition">
                    <svg class="h-6 w-6" stroke="currentColor"
fill="none" viewBox="0 0 24 24">
                        <path :class="{'hidden': open, 'inline-flex': !
open }" class="inline-flex" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
                        <path :class="{'hidden': ! open, 'inline-flex':
open }" class="hidden" stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M6 18L18 6M6 6l12 12" />
                    </svg>
                </button>
            </div>
        </div>
    </div>

    <!-- Responsive Navigation Menu -->
    <div :class="{'block': open, 'hidden': ! open}" class="hidden
sm:hidden">
        <div class="pt-2 pb-3 space-y-1">
            <x-jet-responsive-nav-link href="{{ route('dashboard') }}"
:active="request()->routeIs('dashboard')">
```

```
                {{ __('Dashboard') }}
            </x-jet-responsive-nav-link>
        </div>

        <!-- Responsive Settings Options -->
        <div class="pt-4 pb-1 border-t border-gray-200">
            <div class="flex flex-col sm:flex-row items-center
sm:items-start px-4">
                @if
(Laravel\Jetstream\Jetstream::managesProfilePhotos())
                    <img class="h-10 w-10 rounded-full object-cover"
src="{{ Auth::user()->profile_photo_url }}" alt="{{ Auth::user()->name
}}" />
                @endif

                <div class="sm:w-1/2 mt-3 sm:mt-0 flex flex-col
items-center sm:items-start">
                    <div class="font-medium text-base text-gray-800">{{
Auth::user()->name }}</div>
                    <div class="font-medium text-sm text-gray-500">{{
Auth::user()->email }}</div>
                </div>
            </div>

            <div class="mt-4 space-y-4">
                <!-- Account Management -->
                <x-jet-responsive-nav-link href="{{
route('profile.show') }}" :active="request()->routeIs('profile.show')">
                    {{ __('Profile') }}
                </x-jet-responsive-nav-link>

                @if (Laravel\Jetstream\Jetstream::hasApiFeatures())
                    <x-jet-responsive-nav-link href="{{
route('api-tokens.index') }}"
:active="request()->routeIs('api-tokens.index')">
                        {{ __('API Tokens') }}
                    </x-jet-responsive-nav-link>
                @endif
```

```
            </div>
        </div>
    </div>

                <!-- Authentication -->
                <form method="POST" action="{{ route('logout') }}"
x-data>
                    @csrf

                    <x-jet-responsive-nav-link href="{{ route('logout')
}}"
                                               @click.prevent="$root.submit();">
                        {{ __('Log Out') }}
                    </x-jet-responsive-nav-link>
                </form>

                <!-- Team Management -->
                @if (Laravel\Jetstream\Jetstream::hasTeamFeatures())
                    <div class="border-t border-gray-200"></div>

                    <div class="block px-4 py-2 text-xs text-gray-400">
                        {{ __('Manage Team') }}
                    </div>

                    <!-- Team Settings -->
                    <x-jet-responsive-nav-link href="{{
route('teams.show', Auth::user()->currentTeam->id) }}"
:active="request()->routeIs('teams.show')">
                        {{ __('Team Settings') }}
                    </x-jet-responsive-nav-link>

                    @can('create',
Laravel\Jetstream\Jetstream::newTeamModel())
                        <x-jet-responsive-nav-link href="{{
route('teams.create') }}" :active="request()->routeIs('teams.create')">
                            {{ __('Create New Team') }}
                        </x-jet-responsive-nav-link>
                    @endcan
```

```
                    <div class="border-t border-gray-200"></div>

                    <!-- Team Switcher -->
                    <div class="block px-4 py-2 text-xs text-gray-400">
                        {{ __('Switch Teams') }}
                    </div>

                    @foreach (Auth::user()->allTeams() as $team)
                        <x-jet-switchable-team :team="$team"
component="jet-responsive-nav-link" />
                    @endforeach
                @endif
            </div>
        </div>
    </div>
</nav>
```

Navigation

Esta vista proporciona una barra de navegación responsiva y dinámica que se adapta según el tamaño de la pantalla.

1. Contenedor Principal:

- `bg-blue-500 border-gray-200 px-2 sm:px-4 py-2.5 dark:bg-gray-800`: Define el estilo de fondo, borde y espaciado de la barra de navegación.
- `container flex flex-wrap justify-between items-center mx-auto`: Configura el contenedor para alinear su contenido y distribuir los elementos de manera flexible.

2. Logo:

- `hidden md:block`: Oculta el logo en pantallas pequeñas y lo muestra en pantallas medianas y grandes.

- El enlace redirige a la página de bienvenida.

3. Botón de Menú para Móviles:

- `id="menu-toggle"`: Define el botón que activa/desactiva el menú en pantallas pequeñas.
- Muestra un ícono de hamburguesa utilizando un SVG.

4. Menú de Navegación para Móviles:

- `w-full md:flex md:w-auto hidden`: Se oculta inicialmente y se muestra solo en pantallas medianas y grandes.
- Contiene enlaces de navegación que se adaptan según los roles del usuario, con estilos específicos para cada enlace dependiendo de si es Administrador, Profesor o Alumno.

5. Foto de Perfil:

- `hidden md:block ml-8 relative mt-0 flex items-center`: Oculta en pantallas pequeñas y muestra la foto de perfil en pantallas medianas y grandes.
- Muestra un dropdown con opciones de gestión de cuenta (perfil y cerrar sesión).
- Solo se muestra si el usuario está autenticado.

6. Contador de Correos No Leídos:

- Solo visible para el rol de Administrador.
- Utiliza un componente Livewire para mostrar el contador de correos no leídos.

7. Script de JavaScript:

- Maneja la visibilidad del menú en pantallas pequeñas y grandes.
- `menuToggle.addEventListener('click', function () {...})`: Alterna la visibilidad del menú móvil.
- `function updateLayout() {...}`: Ajusta la visibilidad de los elementos según el tamaño de la ventana.
- `window.addEventListener('resize', updateLayout)`: Recalcula la disposición de los elementos al redimensionar la ventana.

```
<nav class="bg-blue-500 border-gray-200 px-2 sm:px-4 py-2.5
dark:bg-gray-800">
```

```
   <div class="container flex flex-wrap justify-between items-center
mx-auto">
       <!-- Logo -->
       <div class="hidden md:block" id="logo">
           <a href="{{ route('welcome') }}" class="self-center text-2xl
font-semibold whitespace-nowrap text-white">GTV</a>
       </div>
       <!-- Mobile menu button -->
       <button id="menu-toggle" class="md:hidden flex items-center
text-white focus:outline-none">
           <svg class="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">
               <path class="inline-flex" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4
18h16"></path>
           </svg>
       </button>

       <!-- Mobile menu -->
       <div class="w-full md:flex md:w-auto hidden" id="mobile-menu">
           <ul class="flex flex-col md:border-4
md:border-r-indigo-500/50 mt-4 md:flex-row md:space-x-6 md:mt-0
md:text-md md:font-bold">
               @role('Administrador')
               <li>
                   <a href="{{ route('users.index') }}" class="block
py-2 pr-4 pl-3 text-white border-b border-gray-100 hover:bg-gray-50
md:hover:bg-transparent md:border-0 md:hover:text-gray-200 md:p-0
dark:text-gray-400 md:dark:hover:text-white dark:hover:bg-gray-700
dark:hover:text-white md:dark:hover:bg-transparent
dark:border-gray-700">Usuarios</a>
               </li>
               <li>
                   <a href="{{ route('video-items.index') }}"
class="block py-2 pr-4 pl-3 text-white border-b border-gray-100
hover:bg-gray-50 md:hover:bg-transparent md:border-0
md:hover:text-gray-200 md:p-0 dark:text-gray-400
```

```
md:dark:hover:text-white dark:hover:bg-gray-700 dark:hover:text-white
md:dark:hover:bg-transparent dark:border-gray-700">Vídeo Items</a>
                </li>
                @endrole
                @hasanyrole('Administrador|Profesor')
                <li>
                    <a href="{{ route('thematic-areas.index') }}"
class="block py-2 pr-4 pl-3 text-white border-b border-gray-100
hover:bg-gray-50 md:hover:bg-transparent md:border-0
md:hover:text-gray-200 md:p-0 dark:text-gray-400
md:dark:hover:text-white dark:hover:bg-gray-700 dark:hover:text-white
md:dark:hover:bg-transparent dark:border-gray-700">Áreas temáticas</a>
                </li>
                <li>
                    <a href="{{ route('visit.index') }}" class="block
py-2 pr-4 pl-3 text-white border-b border-gray-100 hover:bg-gray-50
md:hover:bg-transparent md:border-0 md:hover:text-gray-200 md:p-0
dark:text-gray-400 md:dark:hover:text-white dark:hover:bg-gray-700
dark:hover:text-white md:dark:hover:bg-transparent
dark:border-gray-700">Visitas</a>
                </li>
                @endhasanyrole
                @hasanyrole('Administrador|Profesor|Alumno')
                <li>
                    <a href="{{ route('points.index') }}" class="block
py-2 pr-4 pl-3 text-white border-b border-gray-100 hover:bg-gray-50
md:hover:bg-transparent md:border-0 md:hover:text-gray-200 md:p-0
dark:text-gray-400 md:dark:hover:text-white dark:hover:bg-gray-700
dark:hover:text-white md:dark:hover:bg-transparent
dark:border-gray-700">Puntos de Interés</a>
                </li>
                <li>
                    <a href="{{ route('places.index') }}" class="block
py-2 pr-4 pl-3 text-white border-b border-gray-100 hover:bg-gray-50
md:hover:bg-transparent md:border-0 md:hover:text-gray-200 md:p-0
dark:text-gray-400 md:dark:hover:text-white dark:hover:bg-gray-700
dark:hover:text-white md:dark:hover:bg-transparent
dark:border-gray-700">Lugares</a>
```

```
                </li>
                <li>
                    <a href="{{ route('videos.index') }}" class="block
py-2 pr-4 pl-3 text-white border-b border-gray-100 hover:bg-gray-50
md:hover:bg-transparent md:border-0 md:hover:text-gray-200 md:p-0
dark:text-gray-400 md:dark:hover:text-white dark:hover:bg-gray-700
dark:hover:text-white md:dark:hover:bg-transparent
dark:border-gray-700">Vídeos</a>
                </li>
                <li>
                    <a href="{{ route('photographies.index') }}"
class="block py-2 pr-4 pl-3 text-white border-b border-gray-100
hover:bg-gray-50 md:hover:bg-transparent md:border-0
md:hover:text-gray-200 md:p-0 dark:text-gray-400
md:dark:hover:text-white dark:hover:bg-gray-700 dark:hover:text-white
md:dark:hover:bg-transparent dark:border-gray-700">Foto</a>
                </li>
                @endhasanyrole
            </ul>
        </div>

        <!-- Profile photo -->
        <div class="hidden md:block ml-8 relative mt-0 flex
items-center" id="profile-photo">
            @auth
                <x-jet-dropdown align="right" width="48">
                    <x-slot name="trigger">
                        <button class="flex text-sm border-2
border-transparent rounded-full focus:outline-none
focus:border-gray-300 transition">
                            <img class="h-10 w-10 md:h-8 md:w-8 lg:w-4
lg:h-4 rounded-full object-cover" src="{{
Auth::user()->profile_photo_url }}" alt="{{ Auth::user()->login }}" />
                        </button>
                    </x-slot>

                    <x-slot name="content">
                        <!-- Account Management -->
```

```
                        <div class="block px-4 py-2 text-sm
text-gray-400">
                            @if (auth()->user()->roles->first())
                                {{ auth()->user()->roles->first()->name
}}
                            @endif
                        </div>

                        <x-jet-dropdown-link href="{{
route('profile.show') }}">
                            {{ __('Profile') }}
                        </x-jet-dropdown-link>

                        <div class="border-t border-gray-100"></div>

                        <!-- Authentication -->
                        <form method="POST" action="{{ route('logout')
}}">
                            @csrf

                            <x-jet-dropdown-link href="{{
route('logout') }}"
onclick="event.preventDefault();this.closest('form').submit();">
                                {{ __('Log Out') }}
                            </x-jet-dropdown-link>
                        </form>
                    </x-slot>
                </x-jet-dropdown>
            @endauth
            @role('Administrador')
                <div id="inbox" class="space-x-4">
                @livewire('admin.unread-email-counter')
                </div>
            @endrole
        </div>

    </div>
</nav>
```

```
<script>
   document.addEventListener('DOMContentLoaded', function() {
       var menu = document.getElementById('mobile-menu');
       var logo = document.getElementById('logo');
       var profilePhoto = document.getElementById('profile-photo');
       var menuToggle = document.getElementById('menu-toggle');
       var inbox = document.getElementById('inbox');

       menuToggle.addEventListener('click', function () {
           menu.classList.toggle('hidden');
           if (menu.classList.contains('hidden')) {
               logo.classList.remove('hidden');
               profilePhoto.classList.remove('hidden');
               inbox.classList.remove('hidden');
           } else {
               logo.classList.add('hidden');
               profilePhoto.classList.add('hidden');
               inbox.classList.remove('hidden');
           }
       });

       function updateLayout() {
           if (window.innerWidth < 1024) {
               menuToggle.classList.remove('hidden');
               menu.classList.add('hidden');
               logo.classList.add('hidden');
               profilePhoto.classList.add('hidden');
               inbox.classList.add('hidden');
           } else {
               menuToggle.classList.add('hidden');
               logo.classList.remove('hidden');
               profilePhoto.classList.remove('hidden');
               menu.classList.remove('hidden');
               inbox.classList.remove('hidden');
           }
       }
```

```
        updateLayout();

        window.addEventListener('resize', updateLayout);

    });
</script>
```

Policy

Esta vista muestra una página de política a los usuarios no autenticados.

1. **Layout de Invitado (`<x-guest-layout>`)**:
    - Este código utiliza un layout específico para usuarios no autenticados, lo que implica que la página se mostrará cuando un usuario visite la aplicación sin iniciar sesión.
2. **Contenido de la Página**:
    - `bg-gray-100`: Establece un fondo de color gris claro para la página.
    - `min-h-screen flex flex-col items-center pt-6 sm:pt-0`: Configura la altura mínima de la pantalla y alinea verticalmente los elementos en el centro.
    - `<x-jet-authentication-card-logo />`: Inserta el logo de autenticación, presumiblemente proporcionado por el paquete Jetstream o una personalización similar.
    - `w-full sm:max-w-2xl mt-6 p-6 bg-white shadow-md overflow-hidden sm:rounded-lg prose`: Define un contenedor con un ancho máximo en pantallas pequeñas y medianas, con márgenes y relleno para separar el contenido. El contenido se representa en un fondo blanco con una sombra y es redondeado en las esquinas en pantallas pequeñas y medianas. El contenido de la política se inserta utilizando la variable `$policy`, que probablemente contenga contenido HTML.

```
<x-guest-layout>
    <div class="pt-4 bg-gray-100">
```

```
        <div class="min-h-screen flex flex-col items-center pt-6
sm:pt-0">
            <div>
                <x-jet-authentication-card-logo />
            </div>

            <div class="w-full sm:max-w-2xl mt-6 p-6 bg-white shadow-md
overflow-hidden sm:rounded-lg prose">
                {!! $policy !!}
            </div>
        </div>
    </div>
</x-guest-layout>
```

Terms

Esta vista muestra una página de términos y condiciones a los usuarios no autenticados.

1. **Layout de Invitado (`<x-guest-layout>`)**:
    - Al igual que en el fragmento anterior, se utiliza un layout específico para usuarios no autenticados.
2. **Contenido de la Página**:
    - `bg-gray-100`: Establece un fondo de color gris claro para la página.
    - `min-h-screen flex flex-col items-center pt-6 sm:pt-0`: Configura la altura mínima de la pantalla y alinea verticalmente los elementos en el centro.
    - `<x-jet-authentication-card-logo />`: Inserta el logo de autenticación, que puede ser un componente proporcionado por el paquete Jetstream u otro componente personalizado.
    - `w-full sm:max-w-2xl mt-6 p-6 bg-white shadow-md overflow-hidden sm:rounded-lg prose`: Define un contenedor con un ancho máximo en pantallas pequeñas y medianas, con márgenes y relleno para separar el contenido. El contenido se representa en un fondo blanco con una sombra y es redondeado en las esquinas en pantallas pequeñas y

medianas. Los términos y condiciones se insertan utilizando la variable `$terms`, que probablemente contenga contenido HTML.

```
<x-guest-layout>
    <div class="pt-4 bg-gray-100">
        <div class="min-h-screen flex flex-col items-center pt-6
sm:pt-0">
            <div>
                <x-jet-authentication-card-logo />
            </div>

            <div class="w-full sm:max-w-2xl mt-6 p-6 bg-white shadow-md
overflow-hidden sm:rounded-lg prose">
                {!! $terms !!}
            </div>
        </div>
    </div>
</x-guest-layout>
```

API

Api-token-manager

Esta vista proporciona una interfaz para que los usuarios creen, administren y eliminen tokens de API, así como para definir los permisos asociados a esos tokens.

1.Crear Token de API:

- Se utiliza un formulario de Jetstream (`<x-jet-form-section>`) para crear un nuevo token de API.
- Los usuarios pueden ingresar un nombre para el token y seleccionar los permisos requeridos, si están disponibles.

- Al hacer clic en el botón "Create", se envía una solicitud para crear el token.

2.Administrar Tokens de API Existente:

- Si el usuario tiene tokens de API existentes, se muestran en una sección separada.
- Cada token se muestra con su nombre y la opción para eliminarlo.
- También se proporciona la opción para administrar los permisos del token, si están disponibles.

3.Modales:

- Se utilizan modales de Jetstream (`<x-jet-dialog-modal>`) para mostrar el valor del token y para administrar los permisos del token.
- En el modal de valor del token, se muestra el token generado y se ofrece la opción de copiarlo.
- En el modal de permisos del token, se muestran los permisos disponibles y se permite al usuario seleccionar los que desea asignar al token.

4.Confirmación de Eliminación:

- Cuando un usuario intenta eliminar un token de API, se muestra un modal de confirmación para confirmar la acción.

```
<div>
    <!-- Generate API Token -->
    <x-jet-form-section submit="createApiToken">
        <x-slot name="title">
            {{ __('Create API Token') }}
        </x-slot>

        <x-slot name="description">
            {{ __('API tokens allow third-party services to authenticate
with our application on your behalf.') }}
        </x-slot>

        <x-slot name="form">
            <!-- Token Name -->
            <div class="col-span-6 sm:col-span-4">
```

```
                <x-jet-label for="name" value="{{ __('Token Name') }}"
/>
                <x-jet-input id="name" type="text" class="mt-1 block
w-full" wire:model.defer="createApiTokenForm.name" autofocus />
                <x-jet-input-error for="name" class="mt-2" />
            </div>

            <!-- Token Permissions -->
            @if (Laravel\Jetstream\Jetstream::hasPermissions())
                <div class="col-span-6">
                    <x-jet-label for="permissions" value="{{
__('Permissions') }}" />

                    <div class="mt-2 grid grid-cols-1 md:grid-cols-2
gap-4">
                        @foreach
(Laravel\Jetstream\Jetstream::$permissions as $permission)
                            <label class="flex items-center">
                                <x-jet-checkbox
wire:model.defer="createApiTokenForm.permissions"
:value="$permission"/>
                                <span class="ml-2 text-sm
text-gray-600">{{ $permission }}</span>
                            </label>
                        @endforeach
                    </div>
                </div>
            @endif
        </x-slot>

        <x-slot name="actions">
            <x-jet-action-message class="mr-3" on="created">
                {{ __('Created.') }}
            </x-jet-action-message>

            <x-jet-button>
                {{ __('Create') }}
            </x-jet-button>
```

```
        </x-slot>
    </x-jet-form-section>

    @if ($this->user->tokens->isNotEmpty())
        <x-jet-section-border />

        <!-- Manage API Tokens -->
        <div class="mt-10 sm:mt-0">
            <x-jet-action-section>
                <x-slot name="title">
                    {{ __('Manage API Tokens') }}
                </x-slot>

                <x-slot name="description">
                    {{ __('You may delete any of your existing tokens if
they are no longer needed.') }}
                </x-slot>

                <!-- API Token List -->
                <x-slot name="content">
                    <div class="space-y-6">
                        @foreach ($this->user->tokens->sortBy('name') as
$token)
                            <div class="flex items-center
justify-between">
                                <div>
                                    {{ $token->name }}
                                </div>

                                <div class="flex items-center">
                                    @if ($token->last_used_at)
                                        <div class="text-sm
text-gray-400">
                                            {{ __('Last used') }} {{
$token->last_used_at->diffForHumans() }}
                                        </div>
                                    @endif
```

```
                                    @if
(Laravel\Jetstream\Jetstream::hasPermissions())
                                        <button class="cursor-pointer
ml-6 text-sm text-gray-400 underline"
wire:click="manageApiTokenPermissions({{ $token->id }})">
                                            {{ __('Permissions') }}
                                        </button>
                                    @endif

                                    <button class="cursor-pointer ml-6
text-sm text-red-500" wire:click="confirmApiTokenDeletion({{ $token->id
}})">
                                        {{ __('Delete') }}
                                    </button>
                                </div>
                            </div>
                        @endforeach
                    </div>
                </x-slot>
            </x-jet-action-section>
        </div>
    @endif

    <!-- Token Value Modal -->
    <x-jet-dialog-modal wire:model="displayingToken">
        <x-slot name="title">
            {{ __('API Token') }}
        </x-slot>

        <x-slot name="content">
            <div>
                {{ __('Please copy your new API token. For your
security, it won\'t be shown again.') }}
            </div>

            <x-jet-input x-ref="plaintextToken" type="text" readonly
:value="$plainTextToken"
```

```
                class="mt-4 bg-gray-100 px-4 py-2 rounded font-mono
text-sm text-gray-500 w-full"
                autofocus autocomplete="off" autocorrect="off"
autocapitalize="off" spellcheck="false"
                @showing-token-modal.window="setTimeout(() =>
$refs.plaintextToken.select(), 250)"
            />
        </x-slot>

        <x-slot name="footer">
            <x-jet-secondary-button wire:click="$set('displayingToken',
false)" wire:loading.attr="disabled">
                {{ __('Close') }}
            </x-jet-secondary-button>
        </x-slot>
    </x-jet-dialog-modal>

    <!-- API Token Permissions Modal -->
    <x-jet-dialog-modal wire:model="managingApiTokenPermissions">
        <x-slot name="title">
            {{ __('API Token Permissions') }}
        </x-slot>

        <x-slot name="content">
            <div class="grid grid-cols-1 md:grid-cols-2 gap-4">
                @foreach (Laravel\Jetstream\Jetstream::$permissions as
$permission)
                    <label class="flex items-center">
                        <x-jet-checkbox
wire:model.defer="updateApiTokenForm.permissions"
:value="$permission"/>
                        <span class="ml-2 text-sm text-gray-600">{{
$permission }}</span>
                    </label>
                @endforeach
            </div>
        </x-slot>
```

```
        <x-slot name="footer">
            <x-jet-secondary-button
wire:click="$set('managingApiTokenPermissions', false)"
wire:loading.attr="disabled">
                {{ __('Cancel') }}
            </x-jet-secondary-button>

            <x-jet-button class="ml-3" wire:click="updateApiToken"
wire:loading.attr="disabled">
                {{ __('Save') }}
            </x-jet-button>
        </x-slot>
    </x-jet-dialog-modal>

    <!-- Delete Token Confirmation Modal -->
    <x-jet-confirmation-modal wire:model="confirmingApiTokenDeletion">
        <x-slot name="title">
            {{ __('Delete API Token') }}
        </x-slot>

        <x-slot name="content">
            {{ __('Are you sure you would like to delete this API
token?') }}
        </x-slot>

        <x-slot name="footer">
            <x-jet-secondary-button
wire:click="$toggle('confirmingApiTokenDeletion')"
wire:loading.attr="disabled">
                {{ __('Cancel') }}
            </x-jet-secondary-button>

            <x-jet-danger-button class="ml-3"
wire:click="deleteApiToken" wire:loading.attr="disabled">
                {{ __('Delete') }}
            </x-jet-danger-button>
        </x-slot>
    </x-jet-confirmation-modal>
```

```
</div>
```

Index

Esta vista proporciona una página dentro de la aplicación web donde los usuarios pueden gestionar sus tokens de API

1.Layout de la Aplicación (`<x-app-layout>`):

- Este código está dentro de un layout de la aplicación, lo que implica que tiene una estructura común a todas las páginas de la aplicación.
- Define una ranura (`<x-slot>`) llamada "header" que contiene el título "API Tokens".

2.Contenido de la Página:

- Dentro del layout, hay un contenedor `<div>` que contiene el componente Livewire llamado "api-token-manager".
- Este componente probablemente se encarga de mostrar y manejar la lógica para la gestión de tokens de API.

3.Estilos y Diseño:

- Se utilizan clases de Tailwind CSS para definir el diseño y los estilos de la página.
- Se especifica un ancho máximo, un relleno vertical y horizontal, y un tamaño de fuente para el contenido.
- Se utiliza un esquema de columnas responsivo para adaptar el diseño en diferentes tamaños de pantalla.

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            {{ __('API Tokens') }}
        </h2>
    </x-slot>

    <div>
```

```
        <div class="max-w-7xl mx-auto py-10 sm:px-6 lg:px-8">
            @livewire('api.api-token-manager')
        </div>
    </div>
</x-app-layout>
```

Auth

Confirm-password

Esta página solicita al usuario que confirme su contraseña antes de acceder a un área segura de la aplicación, como una medida de seguridad adicional.

1.Tarjeta de Autenticación de Jetstream (`<x-jet-authentication-card>`):

- Contiene el formulario de confirmación de contraseña.
- Utiliza el logo predeterminado de Jetstream (`<x-jet-authentication-card-logo>`).

2.Mensaje de Advertencia:

- Muestra un mensaje de texto para informar al usuario que esta es un área segura de la aplicación y que debe confirmar su contraseña antes de continuar.

3.Errores de Validación:

- Si hay errores de validación en el formulario, se muestran aquí utilizando el componente `<x-jet-validation-errors>`.

4.Formulario de Confirmación de Contraseña:

- Utiliza el método POST para enviar los datos del formulario a la ruta `password.confirm`.
- Contiene un campo para ingresar la contraseña del usuario.

- Presenta un botón "Confirm" (`<x-jet-button>`) para enviar el formulario.

5.Requerimientos de Seguridad:

- El campo de contraseña es obligatorio (`required`) y tiene el enfoque automático (`autofocus`), lo que significa que el cursor del usuario se colocará automáticamente en ese campo cuando se cargue la página.

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div class="mb-4 text-sm text-gray-600">
            {{ __('This is a secure area of the application. Please
confirm your password before continuing.') }}
        </div>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.confirm') }}">
            @csrf

            <div>
                <x-jet-label for="password" value="{{ __('Password') }}"
/>
                <x-jet-input id="password" class="block mt-1 w-full"
type="password" name="password" required
autocomplete="current-password" autofocus />
            </div>

            <div class="flex justify-end mt-4">
                <x-jet-button class="ml-4">
                    {{ __('Confirm') }}
                </x-jet-button>
            </div>
```

```
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Forgot-password

Esta página muestra un mensaje indica que puede solicitar el cambio de su contraseña a traves de su correo electronico.

1.Tarjeta de Autenticación de Jetstream (`<x-jet-authentication-card>`):

- Contiene el formulario para solicitar un enlace de restablecimiento de contraseña.
- Utiliza el logo predeterminado de Jetstream (`<x-jet-authentication-card-logo>`).

2.Mensaje Informativo:

- Muestra un mensaje explicativo para indicar al usuario que olvidó su contraseña y que puede solicitar un enlace de restablecimiento proporcionando su dirección de correo electrónico.

3.Estado de Sesión:

- Si existe un mensaje de sesión, como "status", se muestra aquí en verde.

4.Errores de Validación:

- Si hay errores de validación en el formulario, se muestran utilizando el componente `<x-jet-validation-errors>`.

5.Formulario de Solicitud de Enlace de Restablecimiento de Contraseña:

- Utiliza el método POST para enviar los datos del formulario a la ruta `password.email`.
- Contiene un campo para ingresar la dirección de correo electrónico del usuario.
- Presenta un botón "Email Password Reset Link" (`<x-jet-button>`) para enviar el formulario.

6.Requerimientos de Seguridad:

- El campo de correo electrónico es obligatorio (`required`) y tiene el enfoque automático (`autofocus`), lo que significa que el cursor del usuario se colocará automáticamente en ese campo cuando se cargue la página.
- Utiliza la directiva `old('email')` para recordar el valor anterior del campo de correo electrónico si hay errores de validación.

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div class="mb-4 text-sm text-gray-600">
            {{ __('Forgot your password? No problem. Just let us know your email address and we will email you a password reset link that will allow you to choose a new one.') }}
        </div>

        @if (session('status'))
            <div class="mb-4 font-medium text-sm text-green-600">
                {{ session('status') }}
            </div>
        @endif

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.email') }}">
            @csrf

            <div class="block">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autofocus />
            </div>
```

```
            <div class="flex items-center justify-end mt-4">
                <x-jet-button>
                    {{ __('Email Password Reset Link') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Login

Esta vista representa una página de inicio de sesión en una aplicación web.

1.Tarjeta de Autenticación de Jetstream (`<x-jet-authentication-card>`):

- Contiene el formulario de inicio de sesión.
- Utiliza un título personalizado "GTV" dentro de `<x-slot name="logo">`.

2.Errores de Validación:

- Si hay errores de validación en el formulario, se muestran utilizando el componente `<x-jet-validation-errors>`.

3.Estado de Sesión:

- Si existe un mensaje de sesión, como "status", se muestra aquí en verde.

4.Formulario de Inicio de Sesión:

- Utiliza el método POST para enviar los datos del formulario a la ruta `login`.
- Contiene campos para ingresar el correo electrónico (`email`) y la contraseña (`password`) del usuario.
- Ambos campos son obligatorios (`required`).
- El campo de correo electrónico tiene el enfoque automático (`autofocus`), lo que significa que el cursor del usuario se colocará automáticamente en ese campo cuando se cargue la página.
- Tiene una opción "Remember me" para recordar la sesión del usuario.

- Contiene un enlace "Forgot your password?" (`<a>`) que redirige a la página de restablecimiento de contraseña si está habilitada en la aplicación.
- También tiene un enlace "Register" que lleva a la página de registro (`register`).
- Presenta un botón "Log in" (`<x-jet-button>`) para enviar el formulario.

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <span class="text-4xl font-bold whitespace-nowrap text-gray-800">GTV</span>
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        @if (session('status'))
            <div class="mb-4 font-medium text-sm text-green-600">
                {{ session('status') }}
            </div>
        @endif

        <form method="POST" action="{{ route('login') }}">
            @csrf

            <div>
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email"
:value="old('email')" required autofocus />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}" />
                <x-jet-input id="password" class="block mt-1 w-full" type="password"
name="password" required autocomplete="current-password" />
            </div>
```

```
        <div class="block mt-4">
            <label for="remember_me" class="flex items-center">
                <x-jet-checkbox id="remember_me" name="remember" />
                <span class="ml-2 text-sm text-gray-600">{{ __('Remember me') }}</span>
            </label>
        </div>

        <div class="block mt-4 flex items-center">
            <span class="text-sm text-gray-600">{{ __('You are not logged in') }}</span>
            <a href="{{ route('register') }}" class="ml-2 text-sm text-blue-600 underline">{{
__('Register') }}</a>
        </div>

        <div class="flex items-center justify-end mt-4">
            @if (Route::has('password.request'))
                <a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{
route('password.request') }}">
                    {{ __('Forgot your password?') }}
                </a>
            @endif

            <x-jet-button class="ml-4">
                {{ __('Log in') }}
            </x-jet-button>
        </div>
    </form>
  </x-jet-authentication-card>
</x-guest-layout>
```

Register

Esta vista representa una página de registro en una aplicación web.

1.Formulario de Registro:

- Permite a los usuarios crear una nueva cuenta proporcionando su nombre, correo electrónico y contraseña.
- Los campos obligatorios incluyen nombre (`name`), correo electrónico (`email`), contraseña (`password`) y confirmación de contraseña (`password_confirmation`).
- Incluye un checkbox para que los usuarios acepten los términos de servicio y la política de privacidad, si esta función está habilitada.

2.Gestión de Errores:

- Si hay errores de validación en el formulario (por ejemplo, campos faltantes o contraseña no coincidente), se muestran mensajes de error.

3.Estado de Sesión:

- Si hay un mensaje de estado (por ejemplo, después de una acción exitosa como el registro), se muestra en la parte superior.

4.Interfaz de Usuario Amigable:

- Utiliza un diseño limpio y sencillo para una fácil comprensión y navegación por parte del usuario.
- Ofrece retroalimentación visual para resaltar los errores y mensajes de estado.

5.Integración con Jetstream:

- Utiliza componentes de Jetstream para la interfaz de usuario, lo que facilita la personalización y la integración con Laravel.
- Proporciona enlaces para iniciar sesión si el usuario ya tiene una cuenta.

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('register') }}">
```

```
            @csrf

            <div>
                <x-jet-label for="name" value="{{ __('Name') }}" />
                <x-jet-input id="name" class="block mt-1 w-full"
type="text" name="name" :value="old('name')" required autofocus
autocomplete="name" />
            </div>

            <div class="mt-4">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full"
type="email" name="email" :value="old('email')" required />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}"
/>
                <x-jet-input id="password" class="block mt-1 w-full"
type="password" name="password" required autocomplete="new-password" />
            </div>

            <div class="mt-4">
                <x-jet-label for="password_confirmation" value="{{
__('Confirm Password') }}" />
                <x-jet-input id="password_confirmation" class="block
mt-1 w-full" type="password" name="password_confirmation" required
autocomplete="new-password" />
            </div>

            @if
(Laravel\Jetstream\Jetstream::hasTermsAndPrivacyPolicyFeature())
                <div class="mt-4">
                    <x-jet-label for="terms">
                        <div class="flex items-center">
                            <x-jet-checkbox name="terms" id="terms"/>

                            <div class="ml-2">
```

```
                                {!! __('I agree to the :terms_of_service
and :privacy_policy', [
                                            'terms_of_service' => '<a
target="_blank" href="'.route('terms.show').'" class="underline text-sm
text-gray-600 hover:text-gray-900">'.__('Terms of Service').'</a>',
                                            'privacy_policy' => '<a
target="_blank" href="'.route('policy.show').'" class="underline
text-sm text-gray-600 hover:text-gray-900">'.__('Privacy
Policy').'</a>',
                                ]) !!}
                            </div>
                        </div>
                    </x-jet-label>
                </div>
            @endif

            <div class="flex items-center justify-end mt-4">
                <a class="underline text-sm text-gray-600
hover:text-gray-900" href="{{ route('login') }}">
                    {{ __('Already registered?') }}
                </a>

                <x-jet-button class="ml-4">
                    {{ __('Register') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Reset-password

Esta vista representa una página de restablecimiento de contraseña en una aplicación web.

1.Formulario de Restablecimiento de Contraseña:

- Permite a los usuarios restablecer su contraseña proporcionando su correo electrónico y una nueva contraseña.
- Los campos obligatorios incluyen correo electrónico (`email`), contraseña (`password`) y confirmación de contraseña (`password_confirmation`).
- El token necesario para el restablecimiento de contraseña se pasa como un campo oculto en el formulario.

2.Gestión de Errores:

- Si hay errores de validación en el formulario (por ejemplo, campos faltantes o contraseñas no coincidentes), se muestran mensajes de error.

3.Estado de Sesión:

- Si hay un mensaje de estado (por ejemplo, después de un restablecimiento de contraseña exitoso), se muestra en la parte superior.

4.Interfaz de Usuario Amigable:

- Utiliza un diseño limpio y sencillo para una fácil comprensión y navegación por parte del usuario.
- Ofrece retroalimentación visual para resaltar los errores y mensajes de estado.

5.Integración con Jetstream:

- Utiliza componentes de Jetstream para la interfaz de usuario, lo que facilita la personalización y la integración con Laravel.
- Utiliza el componente `x-jet-validation-errors` para mostrar errores de validación.

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.update') }}">
```

```
            @csrf

            <input type="hidden" name="token" value="{{
$request->route('token') }}">

            <div class="block">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full"
type="email" name="email" :value="old('email', $request->email)"
required autofocus />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}"
/>
                <x-jet-input id="password" class="block mt-1 w-full"
type="password" name="password" required autocomplete="new-password" />
            </div>

            <div class="mt-4">
                <x-jet-label for="password_confirmation" value="{{
__('Confirm Password') }}" />
                <x-jet-input id="password_confirmation" class="block
mt-1 w-full" type="password" name="password_confirmation" required
autocomplete="new-password" />
            </div>

            <div class="flex items-center justify-end mt-4">
                <x-jet-button>
                    {{ __('Reset Password') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Two-factor-challenge

Esta vista representa una página de inicio de sesión que utiliza autenticación de dos factores.

1.Selección entre Código de Autenticación y Código de Recuperación:

- Al cargar la página, se muestra un mensaje solicitando al usuario que confirme el acceso a su cuenta ingresando el código de autenticación proporcionado por su aplicación de autenticación.
- También hay un enlace "Use a recovery code" que permite al usuario cambiar al uso de un código de recuperación en lugar de un código de autenticación.
- Cuando el usuario hace clic en el enlace "Use a recovery code", se muestra un mensaje diferente que solicita al usuario que ingrese uno de sus códigos de recuperación de emergencia.

2.Formulario de Inicio de Sesión:

- El formulario permite al usuario ingresar el código de autenticación o el código de recuperación, según la selección realizada.
- Utiliza el componente `x-jet-validation-errors` para mostrar errores de validación, como códigos incorrectos o campos faltantes.

3.Conmutación entre Modos de Autenticación:

- Los botones "Use a recovery code" y "Use an authentication code" permiten al usuario alternar entre el uso de un código de autenticación y un código de recuperación, respectivamente.
- Al cambiar entre los modos, el enfoque del cursor se ajusta automáticamente al campo correspondiente.

4.Interfaz de Usuario Amigable:

- Utiliza un diseño limpio y sencillo para una fácil comprensión y navegación por parte del usuario.
- Ofrece retroalimentación visual para resaltar el modo de autenticación actual y cualquier error de validación.

```
<x-guest-layout>
```

```
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div x-data="{ recovery: false }">
            <div class="mb-4 text-sm text-gray-600" x-show="! recovery">
                {{ __('Please confirm access to your account by entering
the authentication code provided by your authenticator application.')
}}
            </div>

            <div class="mb-4 text-sm text-gray-600" x-show="recovery">
                {{ __('Please confirm access to your account by entering
one of your emergency recovery codes.') }}
            </div>

            <x-jet-validation-errors class="mb-4" />

            <form method="POST" action="{{ route('two-factor.login')
}}">
                @csrf

                <div class="mt-4" x-show="! recovery">
                    <x-jet-label for="code" value="{{ __('Code') }}" />
                    <x-jet-input id="code" class="block mt-1 w-full"
type="text" inputmode="numeric" name="code" autofocus x-ref="code"
autocomplete="one-time-code" />
                </div>

                <div class="mt-4" x-show="recovery">
                    <x-jet-label for="recovery_code" value="{{
__('Recovery Code') }}" />
                    <x-jet-input id="recovery_code" class="block mt-1
w-full" type="text" name="recovery_code" x-ref="recovery_code"
autocomplete="one-time-code" />
                </div>
```

```
                <div class="flex items-center justify-end mt-4">
                    <button type="button" class="text-sm text-gray-600
hover:text-gray-900 underline cursor-pointer"
                                        x-show="! recovery"
                                        x-on:click="
                                            recovery = true;
                                            $nextTick(() => {
$refs.recovery_code.focus() })
                                        ">
                        {{ __('Use a recovery code') }}
                    </button>

                    <button type="button" class="text-sm text-gray-600
hover:text-gray-900 underline cursor-pointer"
                                        x-show="recovery"
                                        x-on:click="
                                            recovery = false;
                                            $nextTick(() => {
$refs.code.focus() })
                                        ">
                        {{ __('Use an authentication code') }}
                    </button>

                    <x-jet-button class="ml-4">
                        {{ __('Log in') }}
                    </x-jet-button>
                </div>
            </form>
        </div>
    </x-jet-authentication-card>
</x-guest-layout>
```

Verify-email

Esta página solicita al usuario que verifique su dirección de correo electrónico antes de continuar.

1.Mensaje de Verificación de Correo Electrónico:

- Muestra un mensaje solicitando al usuario que verifique su dirección de correo electrónico haciendo clic en el enlace enviado por correo electrónico.
- Si el usuario no recibió el correo electrónico de verificación, se le ofrece la opción de volver a enviar el correo electrónico de verificación.

2.Notificación de Envío Exitoso:

- Si se ha enviado con éxito un nuevo enlace de verificación al correo electrónico del usuario, se muestra un mensaje de confirmación.

3.Formulario para Reenviar Correo Electrónico de Verificación:

- Contiene un formulario que permite al usuario enviar nuevamente el correo electrónico de verificación.
- Utiliza un botón "Resend Verification Email" para enviar el formulario.

4.Enlaces Adicionales:

- Proporciona enlaces adicionales para que el usuario edite su perfil o cierre sesión si lo desea.
- El enlace "Edit Profile" redirige al usuario a la página de edición de perfil.
- El botón "Log Out" permite al usuario cerrar sesión.

5.Diseño Responsivo y Amigable:

- Utiliza un diseño claro y sencillo para facilitar la comprensión y navegación del usuario.
- Los elementos están distribuidos en un diseño limpio y ordenado para una fácil interacción.

```
<x-guest-layout>
    <x-jet-authentication-card>
```

```
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div class="mb-4 text-sm text-gray-600">
            {{ __('Before continuing, could you verify your email address by clicking on the link we just emailed to you? If you didn\'t receive the email, we will gladly send you another.') }}
        </div>

        @if (session('status') == 'verification-link-sent')
            <div class="mb-4 font-medium text-sm text-green-600">
                {{ __('A new verification link has been sent to the email address you provided in your profile settings.') }}
            </div>
        @endif

        <div class="mt-4 flex items-center justify-between">
            <form method="POST" action="{{ route('verification.send') }}">
                @csrf

                <div>
                    <x-jet-button type="submit">
                        {{ __('Resend Verification Email') }}
                    </x-jet-button>
                </div>
            </form>

            <div>
                <a
                    href="{{ route('profile.show') }}"
                    class="underline text-sm text-gray-600 hover:text-gray-900"
                >
                    {{ __('Edit Profile') }}</a>

                <form method="POST" action="{{ route('logout') }}" class="inline">
                    @csrf
```

```
                <button type="submit" class="underline text-sm text-gray-600
hover:text-gray-900 ml-2">
                    {{ __('Log Out') }}
                </button>
            </form>
        </div>
    </div>
  </x-jet-authentication-card>
</x-guest-layout>
```

Components

Button-link

Este componente proporciona un enlace estilizado y reutilizable que puede adaptarse dinámicamente al color especificado por el usuario.

1.Componente de Enlace Reutilizable:

- Este código define un componente de enlace reutilizable que puede ser utilizado en cualquier parte de una aplicación Laravel o Blade.

2.Propiedad de Color:

- El componente acepta una propiedad llamada "color", que por defecto es "blue". Esto permite al usuario personalizar el color del enlace según sus necesidades.

3.Estilos Dinámicos:

- Los estilos del enlace se generan dinámicamente utilizando clases de tailwindcss y el color proporcionado como prop.

- El color del texto y el fondo del enlace se establecen utilizando la clase "text-white" y "bg-$color-700" respectivamente, donde "$color" es el valor proporcionado como prop.
- Se aplican estilos adicionales de interacción como hover y focus utilizando clases dinámicas basadas en el color proporcionado.
- Los estilos para la versión oscura del tema también se gestionan dinámicamente.

4.Contenido del Enlace:

- El contenido del enlace se pasa como un slot utilizando la variable $slot.
- Esto permite que el componente sea flexible y pueda contener cualquier contenido dentro del enlace.

```
@props(['color' => 'blue'])
<a {{ $attributes->merge(['class' => "text-white bg-$color-700 hover:bg-$color-800
focus:ring-4 focus:ring-$color-300 font-medium rounded-lg text-sm px-5 py-2.5
dark:bg-$color-600 dark:hover:bg-$color-700 focus:outline-none
dark:focus:ring-$color-800"]) }}>
  {{ $slot }}
</a>
```

Button

1.Componente de Botón Reutilizable:

- Este código define un componente de botón reutilizable que puede ser utilizado en cualquier parte de una aplicación Laravel o Blade.

2.Propiedad de Color:

- El componente acepta una propiedad llamada "color", que por defecto es "blue". Esto permite al usuario personalizar el color del botón según sus necesidades.

3.Estilos Dinámicos:

- Los estilos del botón se generan dinámicamente utilizando clases de tailwindcss y el color proporcionado como prop.
- El color del texto y el fondo del botón se establecen utilizando la clase "text-white" y "bg-$color-700" respectivamente, donde "$color" es el valor proporcionado como prop.
- Se aplican estilos adicionales de interacción como hover y focus utilizando clases dinámicas basadas en el color proporcionado.
- Los estilos para la versión oscura del tema también se gestionan dinámicamente.

4.Contenido del Botón:

- El contenido del botón se pasa como un slot utilizando la variable $slot.
- Esto permite que el componente sea flexible y pueda contener cualquier contenido dentro del botón.

```
@props(['color' => 'blue'])
<button type="button" {{ $attributes->merge(['class' => "text-white bg-$color-700
hover:bg-$color-800 focus:ring-4 focus:ring-$color-300 font-medium rounded-lg text-sm px-5
py-2.5 dark:bg-$color-600 dark:hover:bg-$color-700 focus:outline-none
dark:focus:ring-$color-800"]) }}>
  {{ $slot }}
</button>
```

Table

1.Componente de Tabla Reutilizable:

- Este componente define una tabla HTML con estilos específicos que pueden ser utilizados en cualquier parte de una aplicación Laravel o Blade.

2.Uso del Componente:

- El uso del componente se proporciona como un comentario al comienzo del código. Proporciona ejemplos de cómo usar el componente para definir la cabecera de la tabla, una fila en el cuerpo de la tabla y una fila con acciones en el cuerpo de la tabla.

3.Personalización de la Tabla:

- El componente acepta dos parámetros llamados `$thead` y `$tbody`, que se utilizan para personalizar dinámicamente tanto la cabecera como el cuerpo de la tabla.
- La cabecera de la tabla se define dentro del elemento `<thead>` y se pasa como contenido a través de la variable `$thead`.
- El cuerpo de la tabla se define dentro del elemento `<tbody>` y se pasa como contenido a través de la variable `$tbody`.

4.Estilos y Clases de Tailwind CSS:

- La tabla y sus elementos tienen estilos definidos utilizando clases de tailwindcss para manejar la alineación, el espaciado, los colores y otros aspectos visuales.

Los estilos de luz y oscuro se gestionan dinámicamente utilizando las clases `dark:bg-xxx` y `dark:text-xxx` para el tema oscuro.

```
{{--
```

```
Use:

   Table head element:

                  <th scope="col" class="px-6 py-3">

                      Product name
```

```
                    </th>


    Table head action element:

                    <th scope="col" class="px-6 py-3">

                        <span class="sr-only">Actions</span>

                    </th>


    Table body row:

                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">

                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">

                            Apple MacBook Pro 17"

                        </td>

                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap text-right">

                            <a href="#" class="font-medium text-blue-600
dark:text-blue-500 hover:underline">Edit</a>

                        </td>

                    </tr>


    Table body actions td inside a row:

                    <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap text-right">
```

```
                        <a href="#" class="font-medium text-blue-600
dark:text-blue-500 hover:underline">Edit</a>

                    </td>

--}}



<div>

   <div class="relative overflow-x-auto shadow-md sm:rounded-lg">

       <table class="w-full text-sm text-left text-gray-500
dark:text-gray-400">

           <thead class="text-xs text-gray-700 uppercase bg-gray-50
dark:bg-gray-700 dark:text-gray-400">

               <tr>

                   {{ $thead }}

               </tr>

           </thead>

           <tbody>

               {{ $tbody }}

           </tbody>

       </table>

   </div>

</div>
```

Email

Proposal-rejected

Este archivo HTML genera una página web simple con un mensaje de rechazo

1.Estructura HTML básica:

- Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`.
- La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.

2.Metaetiquetas:

- Define las metaetiquetas para especificar la codificación de caracteres (`charset`) y la escala inicial (`viewport`) para dispositivos móviles.

3.Enlaces de Recursos Externos:

- Importa la fuente `Nunito` de Google Fonts y el archivo CSS de `tailwindcss`.
- También enlaza el archivo CSS de `flowbite`, una biblioteca de componentes CSS.

4.Clases de Tailwind CSS:

- Utiliza clases de Tailwind CSS para aplicar estilos a los elementos HTML.
- La clase `font-sans` se utiliza para establecer la familia de fuentes sans-serif.
- La clase `antialiased` se utiliza para mejorar la apariencia del texto en pantallas de alta resolución.

5.Contenido de la Página:

- El contenido de la página está dentro del elemento `<main>` con una clase `p-20` para el espaciado.
- Contiene un contenedor `<div>` con clases de Tailwind CSS para darle estilo al mensaje de rechazo de propuesta.

- Incluye un saludo personalizado utilizando el nombre del usuario (`$userName`) pasado como variable.
- Proporciona un mensaje de rechazo de propuesta y una nota para contactar al equipo de gestión si es necesario.

6.Script JavaScript:

- Se incluye un script de `flowbite.js` al final del documento para agregar funcionalidades dinámicas, como interacciones y animaciones, proporcionadas por la biblioteca `flowbite`.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">

        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">
        <script src="https://cdn.tailwindcss.com"></script>
        <link rel="stylesheet"
href="https://unpkg.com/flowbite@1.4.7/dist/flowbite.min.css" />
    </head>
    <body class="font-sans antialiased">
        <div class="min-h-screen">
            <!-- Page Content -->
```

```
            <main class="p-20">

                <div class="mt-4 p-4 max-w-lg text-center space-y-4
bg-white rounded-lg border shadow-md sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">

                    <h5 class="text-4xl font-bold whitespace-nowrap
text-gray-800">GTV</h5>

                    <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 font-bold">Rejected Proposal</p>

                    <div class="space-y-1">

                    <h1 class="text-2xl font-bold text-gray-800
mb-4">Hello, {{ $userName }}</h1>

                    <p class="text-gray-600 mb-4">We regret to inform
you that your proposal has been rejected.</p>

                    <p class="text-gray-600 mb-4">If you have any
questions, please do not hesitate to contact us.</p>

                    <p class="text-gray-600">Greetings,<br>The
management team</p>

                </div>

                </div>

            </main>

        </div>



        <script
src="https://unpkg.com/flowbite@1.4.7/dist/flowbite.js"></script>

    </body>

</html>
```

Teacher-Request

Este archivo HTML genera una página web simple con un mensaje de notificación sobre una nueva solicitud y un botón para redirigir al usuario a una bandeja de entrada específica

1.Estructura HTML básica:

- Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`.
- La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.

2.Metaetiquetas:

- Define las metaetiquetas para especificar la codificación de caracteres (`charset`) y la escala inicial (`viewport`) para dispositivos móviles.

3.Enlaces de Recursos Externos:

- Importa la fuente `Nunito` de Google Fonts y el archivo CSS de `tailwindcss`.
- También enlaza el archivo CSS de `flowbite`, una biblioteca de componentes CSS.

4.Clases de Tailwind CSS:

- Utiliza clases de Tailwind CSS para aplicar estilos a los elementos HTML.
- La clase `font-sans` se utiliza para establecer la familia de fuentes sans-serif.
- La clase `antialiased` se utiliza para mejorar la apariencia del texto en pantallas de alta resolución.

5.Contenido de la Página:

- El contenido de la página está dentro del elemento `<main>` con una clase `p-20` para el espaciado.
- Contiene un contenedor `<div>` con clases de Tailwind CSS para darle estilo al mensaje de notificación y al botón de redirección.
- El mensaje de notificación informa al usuario sobre una nueva solicitud.

- Se incluye un botón que redirige al usuario a una bandeja de entrada específica, utilizando la función `route()` para generar la URL.

6.Script JavaScript:

- Se incluye un script de `flowbite.js` al final del documento para agregar funcionalidades dinámicas proporcionadas por la biblioteca `flowbite`.

```
<!DOCTYPE html>
```

```
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

    <head>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width,
initial-scale=1">



        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">

        <script src="https://cdn.tailwindcss.com"></script>

        <link rel="stylesheet"
href="https://unpkg.com/flowbite@1.4.7/dist/flowbite.min.css" />

    </head>

    <body class="font-sans antialiased">

        <div class="min-h-screen">

            <!-- Page Content -->
```

```
            <main class="p-20">

                <div class="mt-4 p-4 max-w-lg text-center space-y-4
bg-white rounded-lg border shadow-md sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">

                    <h5 class="text-4xl font-bold whitespace-nowrap
text-gray-800">GTV</h5>

                    <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 font-bold">You have a new request</p>

                    <div>

                        <a href="{{ route('admin.emails') }}"
class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4
focus:ring-blue-300 font-medium rounded-lg text-sm px-4 py-2
dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800">

                            Go to tray

                        </a>

                    </div>

                </div>

            </main>

        </div>



        <script
src="https://unpkg.com/flowbite@1.4.7/dist/flowbite.js"></script>

    </body>

</html>
```

User created

Esta vista HTML genera una página web que muestra información sobre un nuevo registro de usuario.

1.**Estructura HTML básica**:

- Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`.
- La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.

2.Metaetiquetas:

- Define las metaetiquetas para especificar la codificación de caracteres (`charset`) y la escala inicial (`viewport`) para dispositivos móviles.
- También incluye un título (`title`) que indica "Usuario registrado".

3.Enlaces de Recursos Externos:

- Importa la fuente `Nunito` de Google Fonts y el archivo CSS de `tailwindcss`.
- Además, enlaza el archivo CSS de `flowbite`, una biblioteca de componentes CSS.

4.Clases de Tailwind CSS:

- Utiliza clases de Tailwind CSS para aplicar estilos a los elementos HTML.
- La clase `font-sans` se utiliza para establecer la familia de fuentes sans-serif.
- La clase `antialiased` se utiliza para mejorar la apariencia del texto en pantallas de alta resolución.

5.Contenido de la Página:

- El contenido de la página está dentro del elemento `<main>` con una clase `p-20` para el espaciado.

- Contiene un contenedor `<div>` con clases de Tailwind CSS para darle estilo al mensaje de notificación y al icono de usuario.
- El mensaje informa al usuario sobre el nuevo registro.
- Se incluye el nombre, correo electrónico y rol del usuario registrado.
- Se proporciona un enlace que redirige al usuario a la página de usuarios, utilizando la función `route()` para generar la URL.

6.Script JavaScript:

- Se incluye un script de `flowbite.js` al final del documento para agregar funcionalidades dinámicas proporcionadas por la biblioteca `flowbite`.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <title>Usuario registrado</title>

        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">
        <script src="https://cdn.tailwindcss.com"></script>
        <link rel="stylesheet"
href="https://unpkg.com/flowbite@1.4.7/dist/flowbite.min.css" />
    </head>
    <body class="font-sans antialiased">
        <div class="min-h-screen">
```

```
            <!-- Page Content -->

            <main class="p-20">

                <div class="mt-4 p-4 max-w-lg text-center space-y-4
bg-white rounded-lg border shadow-md sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">

                    <h5 class="text-4xl font-bold whitespace-nowrap
text-gray-800">GTV</h5>

                    <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 font-bold">Nuevo registro de usuario</p>

                    <div class="relative w-36 h-36 overflow-hidden
bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">

                        <svg class="w-36 h-36 text-gray-400"
fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>

                    </div>

                    <div class="space-y-1">

                        <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400">Nombre: <span class="text-gray-500">{{ $user->name
}}</span></p>

                        <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400">Email: <span class="text-gray-500">{{ $user->email
}}</span></p>

                        <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400">Rol: <span class="text-gray-500">{{
$user->roles->first()->name }}</span></p>

                    </div>
```

```
                    <div>

                        <a href="{{ route('users.index') }}"
class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4
focus:ring-blue-300 font-medium rounded-lg text-sm px-4 py-2
dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800">

                            Ir a usuarios

                        </a>

                    </div>

                </div>

            </main>

        </div>



        <script
src="https://unpkg.com/flowbite@1.4.7/dist/flowbite.js"></script>

    </body>

</html>
```

User-to-promoted-to-teacher

Este archivo HTML genera una página web que muestra un anuncio de promoción para un usuario.

1. **Estructura HTML básica**:Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`.La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.
2. **Metaetiquetas**:Define las metaetiquetas para especificar la codificación de caracteres (`charset`) y la escala inicial (`viewport`) para dispositivos móviles.También incluye un enlace para cargar la fuente `Nunito` desde Google Fonts.
3. **Enlaces de Recursos Externos**:Importa la biblioteca `tailwindcss` desde un CDN y el archivo CSS de `flowbite`.
4. **Clases de Tailwind CSS**:Utiliza clases de Tailwind CSS para aplicar estilos a los elementos HTML.La clase `font-sans` se utiliza para establecer la familia de fuentes sans-serif.La clase `antialiased` se utiliza para mejorar la apariencia del texto en pantallas de alta resolución.
5. **Contenido de la Página**:El contenido de la página está dentro del elemento `<main>` con una clase `p-20` para el espaciado.Contiene un contenedor `<div>` con clases de Tailwind CSS para darle estilo al mensaje de anuncio de promoción.Se incluye el nombre del usuario que ha sido promovido.Se proporciona un mensaje de felicitación y expectativas para el usuario promovido.Se incluye un mensaje de despedida del equipo de administración.
6. **Script JavaScript**:Se incluye un script de `flowbite.js` al final del documento para agregar funcionalidades dinámicas proporcionadas por la biblioteca `flowbite`.

```
<!DOCTYPE html>
```

```
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

   <head>

       <meta charset="utf-8">

       <meta name="viewport" content="width=device-width,
initial-scale=1">



       <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">
```

```
        <script src="https://cdn.tailwindcss.com"></script>

        <link rel="stylesheet"
href="https://unpkg.com/flowbite@1.4.7/dist/flowbite.min.css" />

    </head>

    <body class="font-sans antialiased">

        <div class="min-h-screen">

            <!-- Page Content -->

            <main class="p-20">

                <div class="mt-4 p-4 max-w-lg text-center space-y-4
bg-white rounded-lg border shadow-md sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">

                    <h5 class="text-4xl font-bold whitespace-nowrap
text-gray-800">GTV</h5>

                    <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 font-bold">Promotion Announcement</p>

                    <div class="space-y-1">

                        <h1 class="text-4xl font-bold text-gray-800 mb-6
dark:text-white">Congratulations, {{ $user->name }}!</h1>

                        <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 mb-6">You have been promoted to a Teacher. We are
excited to see what you will bring to our community.</p>

                        <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400">Best regards,<br>The Administration Team</p>

                    </div>

                </div>

            </main>
```

```
        </div>



        <script
src="https://unpkg.com/flowbite@1.4.7/dist/flowbite.js"></script>

    </body>

</html>
```

User-register

Este archivo HTML genera una página web que muestra un mensaje de bienvenida a un nuevo usuario después de registrarse

1. **Estructura HTML básica**: Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`. La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`. Se define el título de la página como "GTV".
2. **Metaetiquetas**: Define las metaetiquetas para especificar la codificación de caracteres (`charset`) y la escala inicial (`viewport`) para dispositivos móviles.También incluye un enlace para cargar la fuente `Nunito` desde Google Fonts.
3. **Enlaces de Recursos Externos**: Importa la biblioteca `tailwindcss` desde un CDN y el archivo CSS de `flowbite`.
4. **Clases de Tailwind CSS**: Utiliza clases de Tailwind CSS para aplicar estilos a los elementos HTML.La clase `font-sans` se utiliza para establecer la familia de fuentes sans-serif.La clase `antialiased` se utiliza para mejorar la apariencia del texto en pantallas de alta resolución.
5. **Contenido de la Página**: El contenido de la página está dentro del elemento `<main>` con una clase `p-20` para el espaciado.Contiene un contenedor `<div>` con clases de Tailwind CSS para dar estilo al mensaje de bienvenida.Se incluye el nombre del usuario que ha sido registrado.Se proporciona un mensaje de bienvenida

al usuario, informándole que su cuenta ha sido creada exitosamente.Se incluye un enlace para dirigir al usuario a la página principal del sitio web.

6. **Script JavaScript**: Se incluye un script de `flowbite.js` al final del documento para agregar funcionalidades dinámicas proporcionadas por la biblioteca `flowbite`.

```
<!DOCTYPE html>
```

```
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

    <head>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width,
initial-scale=1">

        <title>GTV</title>



        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">

        <script src="https://cdn.tailwindcss.com"></script>

        <link rel="stylesheet"
href="https://unpkg.com/flowbite@1.4.7/dist/flowbite.min.css" />

    </head>

    <body class="font-sans antialiased">

        <div class="min-h-screen">

            <!-- Page Content -->

            <main class="p-20">
```

```
                <div class="mt-4 p-4 max-w-lg text-center space-y-4
bg-white rounded-lg border shadow-md sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">

                    <h5 class="text-4xl font-bold whitespace-nowrap
text-gray-800">GTV</h5>

                    <p class="text-base text-gray-700 sm:text-lg
dark:text-gray-400 font-bold">  <h2>¡Bienvenido, {{ $name }}!</h2></p>

                    <div class="relative w-36 h-36 overflow-hidden
bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">

                        <svg class="w-36 h-36 text-gray-400"
fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>

                    </div>

                    <div class="space-y-1">

                <p>Estamos encantados de tenerte con nosotros.</p>

                <p>Tu cuenta ha sido creada exitosamente.</p>

                    </div>

                        <hr>

                    <div>

                        <p>Para ver más detalles, por favor visita
nuestro sitio web:</p>

                        <br>

                        <a href="{{ route('users.index') }}"
class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4
focus:ring-blue-300 font-medium rounded-lg text-sm px-4 py-2
```

```
dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800">

                            Visitar la pagina

                        </a>

                    </div>

                </div>

            </main>

        </div>



        <script
src="https://unpkg.com/flowbite@1.4.7/dist/flowbite.js"></script>

    </body>

</html>
```

Layouts

App

Esta vista es una plantilla base para las páginas web de este proyecto.

1. **Estructura HTML básica**:Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`.La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.

2. **Metaetiquetas**: Define las metaetiquetas para especificar la codificación de caracteres (`charset`), la escala inicial (`viewport`) para dispositivos móviles y el token CSRF para protección contra falsificación de solicitudes entre sitios.
3. **Título de la Página**: Utiliza el título de la aplicación definido en el archivo de configuración de la aplicación (`config('app.name')`) o el valor predeterminado "Laravel".
4. **Fuentes y Estilos**: Importa la fuente `Nunito` desde Google Fonts. Enlaza la hoja de estilos de la aplicación a través de Laravel Mix usando la función `mix()`.
5. **Componentes de Livewire**: Incluye los estilos y scripts de Livewire para la funcionalidad de los componentes de Livewire.
6. **Script JavaScript**: Carga el archivo JavaScript de la aplicación a través de Laravel Mix usando la función `mix()`, diferido para mejorar el rendimiento de la página.
7. **Contenido de la Página**: Muestra un banner de Jetstream si está presente. Incluye la barra de navegación de la aplicación, que probablemente esté definida en otro archivo. Contiene el contenido principal de la página, que se inyecta dinámicamente utilizando la directiva de Blade `{{ $slot }}`.
8. **Bloques de Contenido Dinámico**: Utiliza las directivas `@stack` y `@push` para incluir scripts y estilos adicionales que pueden ser necesarios para páginas específicas.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'Laravel') }}</title>

    <!-- Fonts -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">

    <!-- Styles -->
    <link rel="stylesheet" href="{{ mix('css/app.css') }}">

    @livewireStyles
```

```
    <!-- Scriptss -->
    <script src="{{ mix('js/app.js') }}" defer></script>
</head>
<body class="font-sans antialiased">
<x-jet-banner />

<div class="min-h-screen bg-gray-200">
    @include('navigation')

    <!-- Page Content -->
    <main class="container mx-auto px-2 md:px-20 py-16">
        {{ $slot }}
    </main>
</div>

        @stack('modals')

        @livewireScripts

        @stack('scripts')
    </body>
</html>
```

Guest

Esta vista es una plantilla que proporciona la estructura básica y los recursos necesarios (fuentes, estilos y scripts).

1. **Estructura HTML básica**: Define un documento HTML básico con etiquetas `<html>`, `<head>` y `<body>`. La configuración regional (`lang`) se establece dinámicamente utilizando la función `app()->getLocale()`.

2. **Metaetiquetas**: Define las metaetiquetas para especificar la codificación de caracteres (`charset`), la escala inicial (`viewport`) para dispositivos móviles y el token CSRF para protección contra falsificación de solicitudes entre sitios.
3. **Título de la Página**: Utiliza el título de la aplicación definido en el archivo de configuración de la aplicación (`config('app.name')`) o el valor predeterminado "Laravel".
4. **Fuentes y Estilos**: Importa la fuente `Nunito` desde Google Fonts. Enlaza la hoja de estilos de la aplicación a través de Laravel Mix usando la función `mix()`.
5. **Scripts JavaScript**: Carga el archivo JavaScript de la aplicación a través de Laravel Mix usando la función `mix()`, diferido para mejorar el rendimiento de la página.
6. **Contenido de la Página**: Define un contenedor `<div>` con clases de estilo de texto (`font-sans text-gray-900 antialiased`). El contenido de la página se inserta dinámicamente utilizando la variable `$slot`, que contiene el contenido proporcionado por las vistas Blade que utilizan esta plantilla.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <meta name="csrf-token" content="{{ csrf_token() }}">

        <title>{{ config('app.name', 'Laravel') }}</title>

        <!-- Fonts -->
        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&
display=swap">

        <!-- Styles -->
        <link rel="stylesheet" href="{{ mix('css/app.css') }}">

        <!-- Scripts -->
        <script src="{{ mix('js/app.js') }}" defer></script>
    </head>
    <body>
        <div class="font-sans text-gray-900 antialiased">
```

```
            {{ $slot }}
        </div>
    </body>
</html>
```

Livewire

Welcome

Esta vista proporciona una interfaz de usuario dinámica que muestra diferentes conjuntos de datos dependiendo del rol del usuario que ha iniciado sesión, permitiendo una experiencia personalizada según el tipo de usuario.

1. **Bienvenida y Cabecera**: Muestra un mensaje de bienvenida con el nombre del usuario que ha iniciado sesión.
2. **Tabla de Últimos Usuarios Registrados (para Administradores)**: Muestra una tabla con información sobre los últimos usuarios registrados en el sistema. Incluye columnas para la foto de perfil, ID, nombre y correo electrónico de cada usuario.
3. **Tabla de Últimas Visitas Registradas (para Administradores y Profesores)**: Muestra una tabla con información sobre las últimas visitas registradas en el sistema. Incluye columnas para el ID, sistema operativo, y punto de interés visitado.
4. **Tabla de Últimas Áreas Temáticas Creadas (para Administradores y Profesores)**: Muestra una tabla con información sobre las últimas áreas temáticas creadas en el sistema. Incluye columnas para el ID, nombre y descripción de cada área temática.
5. **Tabla de Últimos Puntos de Interés Creados (para Administradores, Profesores y Alumnos)**: Muestra una tabla con información sobre los últimos puntos de interés creados en el sistema. Incluye columnas para el ID, distancia, y nombre del lugar asociado a cada punto de interés.

6. **Mapa de Puntos de Interés (para Administradores y GTVisor)**: Muestra un mapa interactivo con marcadores para cada punto de interés registrado en el sistema. Utiliza la biblioteca Leaflet para renderizar el mapa y los marcadores.
7. **Tabla de Últimos Lugares Creados (para Administradores y Profesores)**: Muestra una tabla con información sobre los últimos lugares creados en el sistema. Incluye columnas para el ID, nombre y descripción de cada lugar.
8. **Tabla de Últimos Videos Subidos (para Administradores, Profesores y Alumnos)**: Muestra una tabla con información sobre los últimos videos subidos al sistema. Incluye columnas para el ID, descripción y área temática asociada a cada video.

```
<div>
  <h1 class="text-2xl font-semibold text-gray-700">Bienvenido <span class="text-blue-500 font-bold">{{ auth()->user()->name }}</span></h1>

  <div class="mt-6 space-y-10">
    @role('Administrador')

    {{--TABLA USUARIOS   ID-NOMBRE-EMAIL--}}
    <div class="relative overflow-x-auto shadow-md sm:rounded-lg">
      <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
        <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700 dark:text-gray-400">
          <tr>
            <th scope="col" class="px-6 py-3" colspan="3">
              ÚLTIMOS USUARIOS REGISTRADOS
            </th>
            <th scope="col" class="px-6 py-3 text-right" colspan="2">
              TOTAL - {{$countusers}}
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td class="px-6 py-3">
              FOTO
            </td>
```

```
                <td class="px-6 py-3">
                    ID
                </td>
                <td class="px-6 py-3">
                    NOMBRE
                </td>
                <td class="px-6 py-3">
                    EMAIL
                </td>
            </tr>
            @foreach($users as $user)
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        @if($user->profile_photo_path)
                            <img class="w-10 h-10 rounded-full" src="{{ $user->profile_photo_path
}}" alt="User avatar">
                        @else
                            <div class="relative w-10 h-10 overflow-hidden bg-gray-100 rounded-full
dark:bg-gray-600 mx-auto">
                                <svg class="absolute w-12 h-12 text-gray-400 -left-1"
fill="currentColor" viewBox="0 0 20 20" xmlns="http://www.w3.org/2000/svg"><path
fill-rule="evenodd" d="M10 9a3 3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>
                            </div>
                        @endif
                    </td>
                    <td class="px-6 py-4">
                        {{ $user->id }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $user->name }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $user->email }}
                    </td>
```

```
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showUsers('{{ $user->id }}')" class="text-orange-500
hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>
    @endrole

    @hasanyrole('Administrador|Profesor')
    {{--TABLA VISITAS   ID-SSOO-PUNTO DE INTERÉS--}}
    <div class="relative overflow-x-auto shadow-md sm:rounded-lg">
        <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
            <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
            <tr>
                <th scope="col" class="px-6 py-3" colspan="2">
                    ÚLTIMAS VISITAS REGISTRADAS
                </th>
                <th scope="col" class="px-6 py-3 text-right" colspan="2">
                    TOTAL - {{$countvisits}}
                </th>
            </tr>
            </thead>
            <tbody>
            <tr>
                <td class="px-6 py-3">
                    ID
                </td>
                <td class="px-6 py-3">
                    SSOO
                </td>
```

```
                <td class="px-6 py-3">
                    PUNTO DE INTERÉS
                </td>
            </tr>
            @foreach($visits as $visit)
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        <div>{{ $visit->id}}</div>
                    </td>
                    <td class="px-6 py-4">
                        <div>{{ $visit->ssoo }}</div>
                    </td>
                    <td class="px-6 py-4">
                        <div >{{ $visit->point_of_interest_id }}</div>
                    </td>
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showVisits('{{ $visit->id }}')" class="text-orange-500
hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>


    {{--TABLA ÁREAS TEMÁTICAS   ID-NOMBRE-DESCRIPCIÓN--}}
    <div class="relative overflow-x-auto shadow-md sm:rounded-lg">
        <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
            <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
            <tr>
                <th scope="col" class="px-6 py-3" colspan="2">
```

```
                ÚLTIMAS ÁREAS TEMÁTICAS CREADAS
            </th>
            <th scope="col" class="px-6 py-3 text-right" colspan="2">
                TOTAL - {{$countareas}}
            </th>
        </tr>
        </thead>
        <tbody>
        <tr>
            <td class="px-6 py-3">
                ID
            </td>
            <td class="px-6 py-3">
                NOMBRE
            </td>
            <td class="px-6 py-3">
                DESCRIPCIÓN
            </td>
        </tr>
        @foreach($thematicAreas as $thematicArea)
            <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                <td class="px-6 py-4">
                    {{ $thematicArea->id }}
                </td>
                <td class="px-6 py-4">
                    {{ $thematicArea->name }}
                </td>
                <td class="px-6 py-4">
                    {{ $thematicArea->description }}
                </td>
                <td class="px-6 py-4 text-right">
                    <button wire:click="showAreas('{{ $thematicArea->id }}')"
class="text-orange-500 hover:text-orange-400 hover:underline ml-2 font-semibold">
                        Ver más
                    </button>
```

```
                </td>
            </tr>
        @endforeach
        </tbody>
    </table>
</div>

@endhasanyrole

@hasanyrole('Administrador|Profesor|Alumno')
{{--TABLA PUNTOS DE INTERÉS   ID-DISTANCIA-SITIO--}}
<div class="relative overflow-x-auto shadow-md sm:rounded-lg">
    <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
        <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
        <tr>
            <th scope="col" class="px-6 py-3" colspan="2">
                ÚLTIMOS PUNTOS DE INTERÉS CREADOS
            </th>
            <th scope="col" class="px-6 py-3 text-right" colspan="2">
                TOTAL - {{$countpoints}}
            </th>
        </tr>
        </thead>
        <tbody>
        <tr>
            <td class="px-6 py-3">
                ID
            </td>
            <td class="px-6 py-3">
                DISTANCIA
            </td>
            <td class="px-6 py-3">
                SITIO
            </td>
        </tr>
```

```
            @foreach($points as $point)
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        {{$point->id}}
                    </td>
                    <td class="px-6 py-4">
                        {{$point->distance}}
                    </td>
                    <td class="px-6 py-4">
                        {{$point->place->name}}
                    </td>
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showPoints('{{ $point->id }}')" class="text-orange-500
hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>
    @endhasanyrole

    @role('Administrador|GTVisor')
    <div style="width: 100%; height: 70vh;">
        <div id="map" style="width: 100%; height: 100%;"wire:ignore></div>
    </div>

    <div class="flex items-center mb-6">
        <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css">
    </div>

    @push('scripts')
        <!-- Script para inicializar el mapa Leaflet -->
```

```
        <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
        <script>
            // Obtener las coordenadas del primer punto para centrar el mapa
            var firstPoint = {!! json_encode($points->first()) !!};
            var initialLat = firstPoint.latitude;
            var initialLng = firstPoint.longitude;

            // Definir las coordenadas máximas y mínimas para limitar el mapa
            var southWest = L.latLng(-90, -180);
            var northEast = L.latLng(90, 180);
            var bounds = L.latLngBounds(southWest, northEast);

            // // Inicializar el mapa Leaflet centrado en la primera coordenada y con límites
máximos
            var map = L.map('map', {
                minZoom: 1.5, // Establece el zoom mínimo permitido
                maxZoom: 18, // Establece el zoom máximo permitido
                maxBounds: bounds // Establece los límites máximos del mapa
            }).setView([initialLat, initialLng], 10); // Se ajustó el nivel de zoom a 10

            // Agregar la capa de basemaps al mapa ya que con esta si que aparece en
español
            L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
                attribution: '&copy; <a href="https://carto.com/"></a> ',
            }).addTo(map);

            // Iterar sobre los puntos y agregar marcadores
            @foreach($points as $point)
            L.marker([{{ $point->latitude }}, {{ $point->longitude }}])
                .addTo(map)
                .bindPopup(
                    '<img src="{{ $point->photographies->first()->route ?? '' }}" alt="Fotografía del
punto de interés">'+
                    '<br><b>Nombre:</b> {{ $point->name }}'+
                    '<br><b>Descripción del lugar:</b> {{ $point->place->description }}'+
                    '<br><b>Latitud:</b> {{ $point->latitude }}'+
```

```
                '<br><b>Longitud:</b> {{ $point->longitude }}'
            );
        @endforeach
    </script>
@endpush
@endrole

@hasanyrole('Administrador|Profesor')

{{--TABLA LUGARES   ID-NOMBRE-DESCRIPCION--}}
<div class="relative overflow-x-auto shadow-md sm:rounded-lg">
    <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
        <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
        <tr>
            <th scope="col" class="px-6 py-3" colspan="2">
                ÚLTIMOS LUGARES CREADOS
            </th>
            <th scope="col" class="px-6 py-3 text-right" colspan="2">
                TOTAL - {{$countplaces}}
            </th>
        </tr>
        </thead>
        <tbody>
        <tr>
            <td class="px-6 py-3">
                ID
            </td>
            <td class="px-6 py-3">
                NOMBRE
            </td>
            <td class="px-6 py-3">
                DESCRIPCIÓN
            </td>
        </tr>
        @foreach($places as $place)
```

```
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        {{ $place->id }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $place->name }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $place->description }}
                    </td>
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showPlaces('{{ $place->id }}')" class="text-orange-500
hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>

    @endhasanyrole

    @hasanyrole('Administrador|Profesor|Alumno')

    {{--TABLA VIDEOS   ID-DESCRIPCIÓN-AREA--}}
    <div class="relative overflow-x-auto shadow-md sm:rounded-lg">
        <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
            <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
            <tr>
                <th scope="col" class="px-6 py-3" colspan="2">
                    ÚLTIMOS VIDEOS SUBIDOS
                </th>
```

```
                <th scope="col" class="px-6 py-3 text-right" colspan="2">
                    TOTAL - {{$countvideos}}
                </th>
            </tr>
            </thead>
            <tbody>
            <tr>
                <td class="px-6 py-3">
                    ID
                </td>
                <td class="px-6 py-3">
                    DESCRIPCIÓN
                </td>
                <td class="px-6 py-3">
                    ÁREA TEMÁTICA
                </td>
            </tr>
            @foreach($videos as $video)
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        {{ $video->id }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $video->description }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $video->thematicArea->name }}
                    </td>
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showVideos('{{ $video->id }}')" class="text-orange-500
hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
```

```
            @endforeach
            </tbody>
        </table>
    </div>

    {{--TABLA FOTOGRAFÍAS   ID-PUNTO INTERÉS-ÁREA TEMÁTICA--}}
    <div class="relative overflow-x-auto shadow-md sm:rounded-lg">
        <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
            <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700
dark:text-gray-400">
            <tr>
                <th scope="col" class="px-6 py-3" colspan="2">
                    ÚLTIMAS FOTOGRAFÍAS SUBIDAS
                </th>
                <th scope="col" class="px-6 py-3 text-right" colspan="2">
                    TOTAL - {{$countphotographies}}
                </th>
            </tr>
            </thead>
            <tbody>
            <tr>
                <td class="px-6 py-3">
                    ID
                </td>
                <td class="px-6 py-3">
                    PUNTO INTERÉS
                </td>
                <td class="px-6 py-3">
                    ÁREA TEMÁTICA
                </td>
            </tr>
            @foreach($photographies as $photography)
                <tr class="bg-white border-b dark:bg-gray-800 dark:border-gray-700
hover:bg-gray-50 dark:hover:bg-gray-600">
                    <td class="px-6 py-4">
                        {{ $photography->id }}
```

```
                    </td>
                    <td class="px-6 py-4">
                        {{ $photography->point_of_interest_id }}
                    </td>
                    <td class="px-6 py-4">
                        {{ $photography->thematic_area_id }}
                    </td>
                    <td class="px-6 py-4 text-right">
                        <button wire:click="showPhotographies('{{ $photography->id }}')"
class="text-orange-500 hover:text-orange-400 hover:underline ml-2 font-semibold">
                            Ver más
                        </button>
                    </td>
                </tr>
            @endforeach
            </tbody>
        </table>
    </div>
</div>
@endhasanyrole


{{-- Modal showVideos --}}
<x-jet-dialog-modal wire:model="detailsModalVideos.open">
    <x-slot name="title">
        <span class="text-2xl">Detalles del vídeo #{{ $detailsModalVideos['id'] }}</span>
    </x-slot>

    <x-slot name="content">
        <div class="space-y-3">
            @if($detailsModalVideos['route'] !== null)
                @livewire('admin.video.video-preview', ['route' => $detailsModalVideos['route']])
            @endif
            <div>
                <x-jet-label>
                    Descripción: {{ $detailsModalVideos['description']}}
```

```
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Ruta: {{ $detailsModalVideos['route'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Orden: {{ $detailsModalVideos['order'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Punto de interés: {{ $detailsModalVideos['pointOfInterest'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    @if( ! empty($detailsModalVideos['thematicAreaId']))
                        Área temática: {{ $detailsModalVideos['thematicAreaName'] }} ({{
$detailsModalVideos['thematicAreaId'] }})
                    @else
                        Área temática: <span class="text-red-600">Sin área temática</span>
                    @endif
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Creador: {{ $detailsModalVideos['creatorName'] }} ({{
$detailsModalVideos['creatorId'] }})
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Actualizador:
```

```
                    @if($detailsModalVideos['updaterName'])
                        {{ $detailsModalVideos['updaterName'] }} ({{
$detailsModalVideos['updaterId'] }})
                    @else
                        {{ 'ninguno' }}
                    @endif
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de creación: {{ $detailsModalVideos['createdAt'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de actualización: {{ $detailsModalVideos['updatedAt'] }}
                </x-jet-label>
            </div>
        </div>
    </x-slot>

    <x-slot name="footer">
        <x-button wire:click="$set('detailsModalVideos.open', false)">
            Cerrar
        </x-button>
    </x-slot>
</x-jet-dialog-modal>

{{-- Modal showTemathicAreas --}}
<x-jet-dialog-modal wire:model="detailsModalAreas.open">
    <x-slot name="title">
        <span class="text-2xl">Detalle del área temática #{{ $detailsModalAreas['id']
}}</span>
    </x-slot>

    <x-slot name="content">
```

```
        <div class="space-y-6">
            <div class="mb-4">
                <x-jet-label>
                    Nombre: {{ $detailsModalAreas['name'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    Descripción: {{ $detailsModalAreas['description'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    Fecha de creación: {{ $detailsModalAreas['createdAt'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                @if($detailsModalAreas['updatedAt'] == NULL)
                    <x-jet-label>
                        Fecha de actualización: No se ha actualizado
                    </x-jet-label>
                @else
                <x-jet-label>
                    Fecha de actualización: {{ $detailsModalAreas['updatedAt'] }}
                </x-jet-label>
                @endif
            </div>
        </div>
    </x-slot>

    <x-slot name="footer">
        <x-button wire:click="$toggle('detailsModalAreas.open')">
            Cerrar
```

```
        </x-button>
    </x-slot>
</x-jet-dialog-modal>

{{-- Modal showUsers --}}
<x-jet-dialog-modal wire:model="detailsModalUsers.open">
    <x-slot name="title">
        <span class="text-2xl">Detalles del usuario #{{ $detailsModalUsers['id'] }}</span>
    </x-slot>

    <x-slot name="content">
        <div class="space-y-3">
            <div class="my-8">
                @if($detailsModalUsers['avatar'])
                    <img class="w-36 h-36 rounded-full mx-auto" src="{{
$detailsModalUsers['avatar'] }}" alt="User avatar">
                @else
                    <div class="relative w-36 h-36 overflow-hidden bg-gray-100 rounded-full
dark:bg-gray-600 mx-auto">
                        <svg class="w-36 h-36 text-gray-400" fill="currentColor" viewBox="0 0 20
20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3 3 0 100-6 3
3 0 000 6zm-7 9a7 7 0 1114 0H3z" clip-rule="evenodd"></path></svg>
                    </div>
                @endif
            </div>
            <div>
                <x-jet-label>
                    Nombre: {{ $detailsModalUsers['name']}}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Email: {{ $detailsModalUsers['email'] }}
                </x-jet-label>
            </div>
            <div>
```

```
                <x-jet-label>
                    Rol: {{ $detailsModalUsers['rol'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de creación: {{ $detailsModalUsers['createdAt'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de actualización: {{ $detailsModalUsers['updatedAt'] }}
                </x-jet-label>
            </div>
        </div>
    </x-slot>

    <x-slot name="footer">
        <x-button wire:click="$set('detailsModalUsers.open', false)">
            Cerrar
        </x-button>
    </x-slot>
</x-jet-dialog-modal>

{{-- Modal showVisits --}}
<x-jet-dialog-modal wire:model="detailsModalVisits.open">
    <x-slot name="title">
        <span class="text-2xl">Detalles de la Visita #{{ $detailsModalVisits['id'] }}</span>
    </x-slot>

    <x-slot name="content">
        <div class="space-y-3">
            <div>
                <x-jet-label>
                    Hora: {{ $detailsModalVisits['hour']}}
                </x-jet-label>
```

```
            </div>
            <div>
                <x-jet-label>
                    Id Dispositivo: {{ $detailsModalVisits['deviceid'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Version de la App: {{ $detailsModalVisits['appversion'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Agente: {{ $detailsModalVisits['useragent'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Sistema Operativo: {{ $detailsModalVisits['ssoo'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Version Sistema Operativo: {{ $detailsModalVisits['ssooversion'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Punto de Interest:

{!!QrCode::size(100)->generate(json_encode($detailsModalVisits['point_of_interest_id'],
JSON_PRETTY_PRINT)) !!}
                </x-jet-label>
            </div>
        </div>
    </x-slot>
```

```
    <x-slot name="footer">
        <x-button wire:click="$set('detailsModalVisits.open', false)">
            Cerrar
        </x-button>
    </x-slot>
</x-jet-dialog-modal>

{{-- Modal showPoints --}}
<x-jet-dialog-modal wire:model="detailsModalPoints.open">
    <x-slot name="title">
        <span class="text-2xl">Detalles del Punto de Interes #{{ $detailsModalPoints['id']
}}</span>
    </x-slot>

    <x-slot name="content">
        <div class="space-y-3">
            <div>
                <x-jet-label>
                    Distancia: {{ $detailsModalPoints['distance']}}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Latitud: {{ $detailsModalPoints['latitude'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Longitud: {{ $detailsModalPoints['longitude'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Sitio: {{ $detailsModalPoints['placeName'] }} ({{ $detailsModalPoints['placeId']
}})
```

```
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Creador: {{ $detailsModalPoints['creatorName'] }} ({{
$detailsModalPoints['creatorId'] }})
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Actualizador:
                    @if($detailsModalPoints['updaterName'])
                        {{ $detailsModalPoints['updaterName'] }} ({{ $detailsModalPoints['updaterId']
}})
                    @else
                        {{ 'ninguno' }}
                    @endif
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de creación: {{ $detailsModalPoints['createdAt'] }}
                </x-jet-label>
            </div>
            <div>
                <x-jet-label>
                    Fecha de actualización: {{ $detailsModalPoints['updatedAt'] }}
                </x-jet-label>
            </div>
        </div>
    </x-slot>

    <x-slot name="footer">
        <x-button wire:click="$set('detailsModalPoints.open', false)">
            Cerrar
        </x-button>
```

```
    </x-slot>
  </x-jet-dialog-modal>

  {{-- Modal showPlaces --}}
  <x-jet-dialog-modal wire:model="detailsModalPlaces.open">
    <x-slot name="title">
      <span class="text-2xl">Detalles del lugar #{{ $detailsModalPlaces['id'] }}</span>
    </x-slot>

    <x-slot name="content">
      <div class="space-y-3">
        <div>
          <x-jet-label>
            ID: {{ $detailsModalPlaces['id']}}
          </x-jet-label>
        </div>
        <div>
          <x-jet-label>
            Nombre: {{ $detailsModalPlaces['name']}}
          </x-jet-label>
        </div>
        <div>
          <x-jet-label>
            Descripción: {{ $detailsModalPlaces['description']}}
          </x-jet-label>
        </div>
        <div>
          <x-jet-label>
            Creador: {{ $detailsModalPlaces['creatorName'] }} ({{
$detailsModalPlaces['creatorId'] }})
          </x-jet-label>
        </div>
        <div>
          <x-jet-label>
            Actualizador:
            @if($detailsModalPlaces['updaterName'])
```

```
                    {{ $detailsModalPlaces['updaterName'] }} ({{
$detailsModalPlaces['updaterId'] }})
                @else
                    {{ 'ninguno' }}
                @endif
            </x-jet-label>
        </div>
        <div>
            <x-jet-label>
                Fecha de creación: {{ $detailsModalPlaces['createdAt'] }}
            </x-jet-label>
        </div>
        <div>
            <x-jet-label>
                Última actualización: {{ $detailsModalPlaces['updatedAt'] }}
            </x-jet-label>
        </div>
    </div>
</x-slot>

<x-slot name="footer">
    <x-button wire:click="$set('detailsModalPlaces.open', false)">
        Cerrar
    </x-button>
</x-slot>
</x-jet-dialog-modal>

{{-- Modal showPhotographies--}}
<x-jet-dialog-modal wire:model="detailsModalPhotographies.open">
<x-slot name="title">
    <span class="text-2xl">Detalle de la fotografía #{{ $detailsModalPhotographies['id']
}}</span>
</x-slot>

<x-slot name="content">
    <div class="space-y-6">
```

```
            <div class="mb-4">
                <a class="max-w-xs" href="{{ $detailsModalPhotographies['route'] }}"
target="_blank">
                    <img src="{{ $detailsModalPhotographies['route'] }}">
                </a>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    Ruta: {{ $detailsModalPhotographies['route'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    Orden: {{ $detailsModalPhotographies['order'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    Punto de interes: {{ $detailsModalPhotographies['pointOfInterestId'] }}
                </x-jet-label>
            </div>

            <div class="mb-4">
                <x-jet-label>
                    @if( ! empty($detailsModalPhotographies['thematicAreaId']))
                        Área temática: {{ $detailsModalPhotographies['thematicAreaName'] }} (ID:
{{ $detailsModalPhotographies['thematicAreaId'] }})
                    @else
                        Área temática: <span class="text-red-600">Sin área temática</span>
                    @endif
                </x-jet-label>
            </div>
```

```
            <div class="mb-4">
                <x-jet-label>
                    Creador: {{ $detailsModalPhotographies['creatorName'] }} (ID: {{
$detailsModalPhotographies['creatorId'] }})
                </x-jet-label>
            </div>

            @if( ! is_null($detailsModalPhotographies['updaterId']))
                <div class="mb-4">
                    <x-jet-label>
                        Actualizador: {{ $detailsModalPhotographies['updaterName'] }} (ID: {{
$detailsModalPhotographies['updaterId'] }})
                    </x-jet-label>
                </div>
            @endif

            <div class="mb-4">
                <x-jet-label>
                    Fecha de creación: {{ $detailsModalPhotographies['createdAt'] }}
                </x-jet-label>
            </div>

            @if( ! is_null($detailsModalPhotographies['updaterId']))
                <div class="mb-4">
                    <x-jet-label>
                        Fecha de actualización: {{ $detailsModalPhotographies['updatedAt'] }}
                    </x-jet-label>
                </div>
            @endif
        </div>
    </x-slot>

    <x-slot name="footer">
        <x-button wire:click="$toggle('detailsModalPhotographies.open')">
            Cerrar
        </x-button>
```

```
    </x-slot>
  </x-jet-dialog-modal>
</div>
```

(Aquí se crea un contenedor HTML para el mapa, dentro de este hay otro elemento con el ID map, que actúa como el contenedor del mapa en sí.)

```
<div style="width: 100%; height: 80vh;">
   <div id="map" style="width: 100%; height: 100%;"wire:ignore></div>
</div>
```

(Este bloque de código agrega un enlace a la hoja de estilos de Leaflet para aplicar estilos al mapa)

```
<div class="flex items-center mb-6">
   <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css">
</div>
```

(Inicialización del mapa)

```
@push('scripts')
   <!-- Script para inicializar el mapa Leaflet -->
   <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
   <script>
```

(Se obtienen las coordenadas del primer punto para cuando se cargue el mapa por primera vez se centre en dichas coordenadas)

```
        var firstPoint = {!! json_encode($points->first()) !!};
        var initialLat = firstPoint.latitude;
        var initialLng = firstPoint.longitude;
```

(Se definen las coordenadas máximas y mínimas para limitar el mapa)

```
        var southWest = L.latLng(-90, -180);
        var northEast = L.latLng(90, 180);
        var bounds = L.latLngBounds(southWest, northEast);
```

(Se inicializa el mapa con un zoom máximo y un zoom mínimo y también con las coordenadas explicadas anteriormente)

```
        var map = L.map('map', {
            minZoom: 1.5, // Establece el zoom mínimo permitido
            maxZoom: 18, // Establece el zoom máximo permitido
            maxBounds: bounds // Establece los límites máximos del mapa
        }).setView([initialLat, initialLng], 10); // Se ajustó el nivel de zoom a 10
```

(Se agrega una capa de azulejos (tiles) de OpenStreetMap al mapa)

```
        L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
            attribution: '&copy; <a href="https://carto.com/"></a> ',
        }).addTo(map);
```

(Coge los puntos de interés que hay en la BDD (recuperados en la clase del componente) y los añade al mapa)

```
        @foreach($points as $point)
        L.marker([{{ $point->latitude }}, {{ $point->longitude }}])
            .addTo(map)
            .bindPopup(
           '<img src="{{ $point->photographies->first()->route ?? '' }}"
        alt="Fotografía del punto de interés">'+
           '<br><b>Nombre:</b> {{ $point->name }}'+
           '<br><b>Descripción del lugar:</b> {{ $point->place->description }}'+
           '<br><b>Latitud:</b> {{ $point->latitude }}'+
           '<br><b>Longitud:</b> {{ $point->longitude }}'
        );

        @endforeach
    </script>
@endpush
```

Admin

Unread-email-counter

Esta vista proporciona una manera rápida de visualizar notificaciones de correos electrónicos entrantes y acceder a ellos a través de un menú desplegable en la interfaz de usuario.

1. **Icono de bandeja de entrada**: Se muestra un icono de bandeja de entrada con una imagen predeterminada. Si hay correos electrónicos no leídos, se muestra un contador de notificaciones en forma de un círculo rojo en la esquina superior derecha del icono.
2. **Menú desplegable de correos electrónicos**: Al hacer clic en el icono de la bandeja de entrada, se abre un menú desplegable que muestra una lista de correos electrónicos entrantes. Cada elemento de la lista muestra el asunto del correo electrónico, la fecha y hora de creación y un fragmento del cuerpo del correo electrónico (limitado a 50 caracteres). Si no hay correos electrónicos entrantes, se muestra un mensaje indicando que no hay peticiones.

3. **Enlace para ver más correos electrónicos**: Si hay correos electrónicos no leídos, se muestra un botón al final del menú desplegable que permite al usuario acceder a una página específica (probablemente llamada "admin.emails") para ver más correos electrónicos y administrarlos.

```
<div class="ml-8 bottom-5">
  <x-jet-dropdown width="96">
    <x-slot name="trigger">
      <span class="relative inline-block cursor-pointer">
        <img class="w-8 h-8 sm:w-10 sm:h-10"
src="https://img.icons8.com/material-outlined/24/inbox.png" alt="inbox"/>
        @if($emailCount > 0)
          <span class="absolute top-0 right-0 inline-flex items-center justify-center w-4 h-4
text-xs sm:text-sm font-bold text-red-100 bg-red-600 rounded-full">
            {{ $emailCount }}
          </span>
        @endif
      </span>
    </x-slot>

    <x-slot name="content">
      <ul>
        @forelse($emails as $email)
          <li class="flex p-2 border-b border-gray-200">
            <article class="flex-1">
              <h1 class="font-bold">{{ $email->subject }}</h1>
              <p class="">{{ $email->created_at->format('d-m-Y H:i') }}</p>
              <p class="text-gray-600">{{ Str::limit($email->body, 50) }}</p>
            </article>
          </li>
        @empty
          <li class="py-6 px-4">
            <p class="text-center text-gray-700">
              No tiene peticiones
            </p>
          </li>
```

```
            @endforelse
        </ul>
        @if($emailCount > 0)
            <div class="px-3 py-2">
                <a href="{{ route('admin.emails') }}" class="text-white bg-blue-700
hover:bg-blue-800 focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-4
py-2 dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none dark:focus:ring-blue-800">
                    Ver peticiones
                </a>
            </div>
        @endif
    </x-slot>
  </x-jet-dropdown>
</div>
```

Email

Email-manager

Esta vista presenta una lista de emails con opciones de búsqueda y filtrado, así como la capacidad de ver detalles de un email en un modal.

1. **Encabezado HTML**: Define el encabezado del documento HTML, incluyendo las metaetiquetas necesarias, el título de la página y los enlaces a fuentes y estilos.

2. **Cuerpo HTML**: Contiene el contenido principal de la página. Aquí hay una descripción detallada de cada sección:
3. **Título**: Muestra el título de la página como "Listado de Emails".
4. **Filtros de Búsqueda**: Presenta un formulario de búsqueda con un menú desplegable para seleccionar el criterio de búsqueda (ID, cuerpo del mensaje, fecha de creación, fecha de actualización), un campo de entrada de texto y un botón para eliminar los filtros aplicados.
5. **Lista de Emails**: Muestra una tabla con la lista de emails. La tabla tiene columnas para el ID, del remitente (`from`) y las acciones.
6. **Paginación**: Si hay más de una página de resultados, se muestra la paginación.
7. **Modal**: Muestra un modal cuando se hace clic en un email en la lista. El modal muestra información detallada del email seleccionado, como el asunto, el nombre del remitente, el email del remitente y el rol del remitente. También incluye botones para ir a una página de usuarios (`Ir a usuarios`) y para cerrar el modal (`Cerrar`).
8. **Scripts JavaScript (al final)**: Incluye el script JavaScript de la aplicación para mejorar la interactividad y la funcionalidad de la página.

```
<div>
```

```
    <div>
        <div class="flex items-center mb-6">
            <h1 class="text-2xl font-semibold text-gray-700">Listado de
Emails</h1>
        </div>

        <div class="mb-3">
            <div class="inline">
                <select class="text-black bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                font-medium rounded-lg text-sm py-1.5 dark:bg-blue-600
dark:hover:bg-blue-700
                focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                    <option value="id">ID</option>
                    <option value="body">CUERPO DEL MENSAJE</option>
                    <option value="created_at">FECHA DE
CREACIÓN</option>
                    <option value="updated_at">FECHA DE
ACTUALIZACIÓN</option>
```

```
                </select>
            </div>

            <x-jet-input class="py-1 border-black" type="text"
wire:model="search" placeholder="Buscar ..."></x-jet-input>

            <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
        </div>

        @if(count($emails))
            <x-table>
                <x-slot name="thead">
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                        ID
                        @if($sortField === 'id' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up"></i>
                        @elseif($sortField === 'id' && $sortDirection
=== 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('from')">
                        From
                        @if($sortField === 'from' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up"></i>
                        @elseif($sortField === 'from' && $sortDirection
=== 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3">
                        Acciones
                    </th>
```

```
                </x-slot>

                <x-slot name="tbody">
                    @foreach($emails as $email)
                        <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                {{ $email->id }}
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                {{ $email->from }}
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap flex gap-4">
                                <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $email->id }}')">
                                    <i class="fa-solid fa-eye"></i>
                                </span>
                                <span class="font-medium text-red-600
cursor-pointer" wire:click="delete('{{ $email->id }}')">
                                    <i class="fa-solid fa-trash"></i>
                                </span>
                            </td>
                        </tr>
                    @endforeach
                </x-slot>
            </x-table>
        @else
            <div>
                <strong>No se encontraron registros</strong>
            </div>
        @endif

        @if($emails->hasPages())
            <div class="px-6 py-3">
```

```
                {{ $emails->links() }}
            </div>
        @endif

        <!-- Modal -->
        @if($showModal)
            <div x-data="{ showModal: @entangle('showModal') }">
                <div class="fixed inset-0 flex items-center
justify-center z-50 bg-gray-900 bg-opacity-50 transition-opacity
ease-out duration-300" x-show="showModal">
                    <div class="bg-white p-6 rounded shadow-lg transform
transition-all ease-in-out duration-300"
                        x-show="showModal"
                        x-transition:enter="transition ease-out
duration-300"
                        x-transition:enter-start="opacity-0 scale-90"
                        x-transition:enter-end="opacity-100 scale-100"
                        x-transition:leave="transition ease-in
duration-200"
                        x-transition:leave-start="opacity-100 scale-100"
                        x-transition:leave-end="opacity-0 scale-90">

                        <div>
                            <!-- Page Content -->
                            <main>
                                <div class="mt-4 p-4 max-w-lg
text-center space-y-4 bg-white sm:p-8 dark:bg-gray-800
dark:border-gray-700 mx-auto">
                                    <h5 class="text-4xl font-bold
whitespace-nowrap text-gray-800">GTV</h5>
                                    <p class="text-base text-gray-700
sm:text-lg dark:text-gray-400 font-bold">{{ $email->subject }}</p>
                                    <div class="relative w-36 h-36
overflow-hidden bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">
                                        <svg class="w-36 h-36
text-gray-400" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
```

```
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>
                                    </div>
                                    <div class="space-y-1">
                                        <p class="text-base
text-gray-700 sm:text-lg dark:text-gray-400">Nombre: <span
class="text-gray-500">{{ $email->user->name }}</span></p>
                                        <p class="text-base
text-gray-700 sm:text-lg dark:text-gray-400">Email: <span
class="text-gray-500">{{ $email->user->email }}</span></p>
                                        <p class="text-base
text-gray-700 sm:text-lg dark:text-gray-400">Role: <span
class="text-gray-500">{{ $email->user->roles->first()->name
}}</span></p>
                                    </div>

                                    <div>
                                        <a href="{{ route('users.index')
}}" class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4
focus:ring-blue-300 font-medium rounded-lg text-sm px-4 py-2
dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800">
                                            Ir a usuarios
                                        </a>
                                        <button wire:click="closeModal"
class="text-white bg-blue-500 hover:bg-blue-800 focus:ring-4
focus:ring-blue-300 font-medium rounded-lg text-sm px-4 py-2
dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800">Cerrar</button>
                                    </div>
                                </div>
                            </main>
                        </div>
                    </div>
                </div>
            </div>
        @endif
    </div>
```

```
</div>
```

Photography

Photographies

Esta vista permite listar, buscar, filtrar, ver, crear, editar y eliminar fotografías, proporcionando una interfaz de usuario completa para administrar este recurso en la aplicación

1. **Encabezado HTML y Estilos**: Define el encabezado del documento HTML, incluyendo las metaetiquetas necesarias y los estilos CSS. También incluye enlaces a los estilos de Tailwind CSS y a las fuentes.
2. **Cuerpo HTML**: Contiene el contenido principal de la página. Aquí hay una descripción detallada de cada sección:
3. **Título y Botón de Añadir**: Muestra el título de la página como "Listado de fotografías" y un botón para añadir nuevas fotografías.
4. **Filtros de Búsqueda**: Presenta un formulario de búsqueda con un menú desplegable para seleccionar el criterio de búsqueda, un campo de entrada de texto y un botón para eliminar los filtros aplicados. Los criterios de búsqueda incluyen ID, punto de interés, área temática, creador, actualizador, fecha de creación y fecha de actualización, dependiendo de los roles del usuario.
5. **Lista de Fotografías**: Muestra una tabla con la lista de fotografías. La tabla tiene columnas para el ID, la ruta, el orden, el punto de interés, el área temática, el creador, el actualizador, la fecha de creación, la fecha de actualización y las acciones (ver, editar, eliminar).
6. **Paginación**: Si hay más de una página de resultados, se muestra la paginación.
7. **Modales**: Se utilizan modales para mostrar detalles de una fotografía, crear una nueva fotografía y editar una fotografía existente.
8. **Scripts JavaScript (al final)**: Incluye scripts JavaScript para mejorar la interactividad y la funcionalidad de la página. Se utilizan eventos Livewire para manejar la creación, actualización y eliminación de fotografías, así como para mostrar mensajes de confirmación al realizar estas acciones.

```
<div>
    <div>
        <div class="flex items-center mb-6">
            <h1 class="text-2xl font-semibold text-gray-700">Listado de
fotografías</h1>

            <button wire:click="$toggle('createForm.open')"
type="button" class="text-white bg-blue-700
                hover:bg-blue-800 focus:ring-4 focus:ring-blue-300
font-medium rounded-lg text-sm
                px-5 py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700
focus:outline-none
                dark:focus:ring-blue-800 ml-auto">
                Añadir
            </button>
        </div>

        <div class="mb-3">
            <div class="inline">
                <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                    <option value="id">ID</option>
                    <option value="point_of_interest_id">PUNTO DE
INTERÉS</option>
                    @hasanyrole('Administrador|Profesor')
                        <option value="thematic_area_id">ÁREA
TEMÁTICA</option>
                        <option value="creator">CREADOR</option>
                        <option value="Updater">ACTUALIZADOR</option>
                    @endhasanyrole
                    <option value="created_at">FECHA DE
CREACIÓN</option>
```

```
                    <option value="updated_at">FECHA DE
ACTUALIZACIÓN</option>
                </select>
            </div>

            <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                placeholder="Buscar ..."></x-jet-input>

            <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
        </div>

        @if(count($photographies))
            <x-table>
                <x-slot name="thead">
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                        ID
                        @if($sortField === 'id' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'id' && $sortDirection
=== 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3">
                        Ruta
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('order')">
                        Orden
                        @if($sortField === 'order' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'order' && $sortDirection
=== 'desc')
```

```
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('point_of_interest_id')">
                    Punto de interés
                    @if($sortField === 'point_of_interest_id' &&
$sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'point_of_interest_id' &&
$sortDirection === 'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('thematic_area_id')">
                    Área temática
                    @if($sortField === 'thematic_area_id' &&
$sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'thematic_area_id' &&
$sortDirection === 'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('creator')">
                    Creador
                    @if($sortField === 'creator' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'creator' &&
$sortDirection === 'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updater')">
```

```
                        Actualizador
                        @if($sortField === 'updater' && $sortDirection
=== 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'updater' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                        Fecha creación
                        @if($sortField === 'created_at' &&
$sortDirection === 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updated_at')">
                        Fecha actualización
                        @if($sortField === 'updated_at' &&
$sortDirection === 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'updated_at' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3">
                        Acciones
                    </th>
                </x-slot>

                <x-slot name="tbody">
                    @foreach($photographies as $photography)
```

```
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $photography->id }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            <a class="max-w-xs" href="{{
$photography->route }}" target="_blank">
                                <img src="{{ $photography->route
}}">
                            </a>
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $photography->order }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            @if( !
empty($photography->point_of_interest_id))
                                {{
$photography->point_of_interest_id }}
                            @else
                                <span
class="text-red-600">Ninguno</span>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            @if( !
is_null($photography->thematic_area_id) && !
empty($photography->point_of_interest_id))
                                {{
\App\Models\ThematicArea::find($photography->thematic_area_id)->name }}
```

```
                                    (ID: {{
$photography->thematic_area_id }})
                                @else
                                    <p class="text-red-600">Ninguna</p>
                                @endif
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                {{
\App\Models\User::find($photography->creator)->name }}
                                @role('Administrador')
                                    (ID: {{
\App\Models\User::find($photography->creator)->id }})
                                @endrole
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                @if($photography->updater)
                                    {{
\App\Models\User::find($photography->updater)->name }}
                                    @role('Administrador')
                                    (ID: {{
\App\Models\User::find($photography->updater)->id }})
                                    @endrole
                                @else
                                    <p class="text-center"> N/A</p>
                                @endif
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                {{ $photography->created_at }}
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                @if($photography->updater)
                                    {{ $photography->updated_at }}
                                @else
                                    <p class="text-center"> N/A</p>
```

```
                                @endif
                            </td>
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                                <span class="font-medium text-blue-600
cursor-pointer mr-3"
                                      wire:click="show('{{
$photography->id }}')">
                                    <i class="fa-solid fa-eye"></i>
                                </span>
                                <span class="font-medium text-yellow-400
cursor-pointer mr-3"
                                      wire:click="edit('{{
$photography->id }}')">
                                    <i class="fa-solid fa-pencil"></i>
                                </span>
                                <span class="font-medium text-red-500
cursor-pointer"

wire:click="$emit('deletePhotography', '{{ $photography->id }}')">
                                    <i class="fa-solid fa-trash"></i>
                                </span>
                            </td>
                        </tr>
                    @endforeach
                </x-slot>
            </x-table>

            @if($photographies->hasPages())
                <div class="mt-6">
                    {{ $photographies->links() }}
                </div>
            @endif
        @else
            <p class="mt-4">No se han encontrado resultados</p>
        @endif
    </div>
```

```
    <x-jet-dialog-modal wire:model="showModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalle de la fotografía #{{
$showModal['id'] }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div class="mb-4">
                    <a class="max-w-xs" href="{{ $showModal['route'] }}"
target="_blank">
                        <img src="{{ $showModal['route'] }}">
                    </a>
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Ruta: {{ $showModal['route'] }}
                    </x-jet-label>
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Orden: {{ $showModal['order'] }}
                    </x-jet-label>
                </div>

                @if( ! empty($showModal['pointOfInterestId']))
                    <div class="mb-4">
                        <x-jet-label>
                            Punto de interés: {{
$showModal['pointOfInterestId'] }}
                        </x-jet-label>
                    </div>

                    <div class="mb-4">
                        <x-jet-label>
                            @if( ! empty($showModal['thematicAreaId']))
```

```
                                Área temática: {{
$showModal['thematicAreaName'] }} (ID: {{ $showModal['thematicAreaId']
}})
                            @else
                                Área temática: <span
class="text-red-600">Ninguna</span>
                            @endif
                        </x-jet-label>
                    </div>
                @else
                    <div class="mb-4">
                        <x-jet-label>
                            Punto de interés: <span
class="text-red-600">Ninguno</span>
                        </x-jet-label>
                    </div>
                    <div class="mb-4">
                        <x-jet-label>
                            Área temática: <span
class="text-red-600">Ninguna</span>
                        </x-jet-label>
                    </div>
                @endif

                <div class="mb-4">
                    <x-jet-label>
                        Creador: {{ $showModal['creatorName'] }} (ID: {{
$showModal['creatorId'] }})
                    </x-jet-label>
                </div>

                @if( ! is_null($showModal['updaterId']))
                    <div class="mb-4">
                        <x-jet-label>
                            Actualizador: {{ $showModal['updaterName']
}} (ID: {{ $showModal['updaterId'] }})
                        </x-jet-label>
                    </div>
```

```
                @endif

                <div class="mb-4">
                    <x-jet-label>
                        Fecha de creación: {{ $showModal['createdAt'] }}
                    </x-jet-label>
                </div>

                @if( ! is_null($showModal['updaterId']))
                    <div class="mb-4">
                        <x-jet-label>
                            Fecha de actualización: {{
$showModal['updatedAt'] }}
                        </x-jet-label>
                    </div>
                @endif
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$toggle('showModal.open')">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    <x-jet-dialog-modal wire:model="createForm.open">
        <x-slot name="title">
            <span class="text-2xl">Crear fotografía</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">

                @if ($this->createForm['route'])
                    Preview
                    <img src="{{
$this->createForm['route']->temporaryUrl() }}">
```

```
                @endif

                <div class="mb-6">
                    <label for="Fotografía" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">
                        Fotografía
                    </label>

                    <input type="file" wire:model="createForm.route"
class="block w-full text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 cursor-pointer dark:text-gray-400 focus:outline-none
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
mt-1"></input>

                    <x-jet-input-error for="editForm.route" class="mt-2"
/>
                </div>
                <div class="mb-6">
                    <label for="pointsOfInterest" class="block mb-2
text-sm font-medium text-gray-900 dark:text-gray-400">
                        Punto de interés
                    </label>
                    <select id="pointsOfInterest" class="bg-gray-50
border border-gray-300 text-gray-900 text-sm rounded-lg
                focus:ring-blue-500 focus:border-blue-500 block w-full
p-2.5 dark:bg-gray-700 dark:border-gray-600
                dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500"
                            wire:model="createForm.pointOfInterestId">
                        <option>Seleccione un punto de interés</option>
                        @foreach ($pointsOfInterest as $pointOfInterest)
                            <option value="{{ $pointOfInterest->id}}">{{
$pointOfInterest->id }}</option>
                        @endforeach
                    </select>
                    @error('createForm.pointOfInterestId') <span
class="text-red-600">{{ $message }}</span> @enderror
                </div>
```

```
                <div class="mb-6">
                    <label for="thematicAreas" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-400">
                        Área temática
                    </label>
                    <select id="thematicAreas" class="bg-gray-50 border
border-gray-300 text-gray-900 text-sm rounded-lg
                focus:ring-blue-500 focus:border-blue-500 block w-full
p-2.5 dark:bg-gray-700 dark:border-gray-600
                dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500"
                            wire:model="createForm.thematicAreaId">
                        <option>Seleccione un area de temática</option>
                        @if( ! is_null($thematicAreas))
                            @foreach ($thematicAreas as $thematicArea)
                                <option value="{{ $thematicArea->id
}}">{{ $thematicArea->name }}</option>
                            @endforeach
                        @endif
                    </select>
                    @error('createForm.thematicAreaId') <span
class="text-red-600">{{ $message }}</span> @enderror
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="save">
                Crear
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    <x-jet-dialog-modal wire:model="editModal.open">
        <x-slot name="title">
            <span class="text-2xl">Actualizar fotografía #{{
$this->editModal['id'] }}</span>
```

```
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div class="mb-4">
                    @if ($this->editForm['route'])
                        Preview:
                        <img src="{{
$this->editForm['route']->temporaryUrl() }}">
                    @else
                        <a class="max-w-xs" href="{{
$this->editModal['route'] }}" target="_blank">
                            <img src="{{ $this->editModal['route'] }}">
                        </a>
                    @endif
                </div>
                <div class="mb-4">
                    <x-jet-label>
                        Cambiar fotografía
                    </x-jet-label>

                    <x-jet-input class="w-full"  type="file"
wire:model="editForm.route"></x-jet-input>

                    <x-jet-input-error for="editForm.route" class="mt-2"
/>
                </div>
                <div class="mb-4">
                    <x-jet-label>
                        Punto de interés
                    </x-jet-label>

                    <select wire:model="editForm.pointOfInterestId"
class="bg-gray-50 border border-gray-300
                            text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full
                            p-2.5 dark:bg-gray-700 dark:border-gray-600
dark:placeholder-gray-400 dark:text-white
```

```
                            dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1">
                        @foreach($pointsOfInterest as $pointOfInterest)
                            <option value="{{ $pointOfInterest->id
}}">{{ $pointOfInterest->id }}</option>
                        @endforeach
                    </select>

                    <x-jet-input-error for="editForm.pointOfInterest"
class="mt-2" />
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Área temática
                    </x-jet-label>

                    <select wire:model="editForm.thematicAreaId"
class="bg-gray-50 border border-gray-300
                            text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full
                            p-2.5 dark:bg-gray-700 dark:border-gray-600
dark:placeholder-gray-400 dark:text-white
                            dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1">
                        <option value="">Seleccione un area de
temática</option>
                        @if( ! is_null($thematicAreas))
                            @foreach($thematicAreas as $thematicArea)
                                <option value="{{ $thematicArea->id
}}">{{ $thematicArea->name }}</option>
                            @endforeach
                        @endif
                    </select>

                    <x-jet-input-error for="editForm.thematicAreaId"
class="mt-2" />
                </div>
```

```
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button style="margin-right: 10px;" wire:click="update({{
$editModal['id'] }})">
                Actualizar
            </x-button>
        </x-slot>
Photographies    </x-jet-dialog-modal>

    @push('scripts')
        <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>

        <script>
            Livewire.on('photographyCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'La fotografía ha sido creada.',
                    'success'
                )
            });
        </script>

        <script>
            Livewire.on('photographyUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'La fotografía ha sido actualizada.',
                    'success'
                )
            });
        </script>

        <script>
            Livewire.on('deletePhotography', photographyId => {
                Swal.fire({
                    title: '¿Quieres eliminar esta fotografía?',
```

```
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {

Livewire.emitTo('admin.photography.photographies', 'delete',
photographyId)
                        Swal.fire(
                            '¡Hecho!',
                            'La fotografía ha sido eliminada.',
                            'success'
                        )
                    }
                })
            });
        </script>
    @endpush
</div>
```

Places

Create-place

Este modal proporciona una interfaz para que los usuarios ingresen el nombre y la descripción de un nuevo lugar, y luego lo guarden en la base de datos. Una vez que se guarda con éxito, se muestra un mensaje de confirmación al usuario.

1. **Modal de Diálogo**: Utiliza el componente x-jet-dialog-modal de Laravel Jetstream para crear un modal de diálogo. Este modal se muestra cuando la variable de Livewire createForm.open está activa (true), lo que indica que se está creando un nuevo lugar.
2. **Título del Modal**: Define el título del modal como "Añadir lugar".
3. **Contenido del Modal**: Contiene un formulario para ingresar el nombre y la descripción del lugar. Aquí hay una descripción de los campos:
4. **Nombre**: Un campo de entrada de texto (<input>) para ingresar el nombre del lugar. El valor de este campo está vinculado a la variable de Livewire createForm.name.
5. **Descripción**: Un área de texto (<textarea>) para ingresar la descripción del lugar. El valor de este campo está vinculado a la variable de Livewire createForm.description.
6. Botones de Acción del Modal: Define los botones de acción del modal en la sección del footer.
7. **Crear**: Un botón de color azul (<x-button color="blue">) que llama a la función save de Livewire cuando se hace clic. Esta función se encarga de guardar el nuevo lugar en la base de datos.
8. **Scripts JavaScript (al final)**: Incluye un script JavaScript para mostrar un mensaje de confirmación utilizando SweetAlert2 cuando se ha creado exitosamente un nuevo lugar. El evento Livewire placeCreated se activa cuando se completa con éxito la creación del lugar.

```
<div>
    <x-jet-dialog-modal wire:model="createForm.open">
        <x-slot name="title">
```

```
            <span class="text-2xl">Añadir lugar</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div>
                    <x-jet-label>
                        Nombre
                    </x-jet-label>
                    <x-jet-input type="text" class="block p-2.5 w-full
text-sm text-gray-900 bg-gray-50 rounded-lg border border-gray-300
focus:ring-blue-500 focus:border-blue-500 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500 mt-1"
                                wire:model="createForm.name" />
                    <x-jet-input-error for="createForm.name"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Descripción
                    </x-jet-label>
                    <textarea wire:model="createForm.description"
rows="4" class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50
rounded-lg border border-gray-300 focus:ring-blue-500
focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600
dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1"></textarea>
                    <x-jet-input-error for="createForm.description"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button color="blue"
                      wire:click="save">
                Crear
```

```
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('placeCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El lugar ha sido creado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

Edit-place

Este modal proporciona una interfaz para que los usuarios editen el nombre y la descripción de un lugar existente, y luego actualicen la información en la base de datos. Una vez que se actualiza con éxito, se muestra un mensaje de confirmación al usuario.

1. **Modal de Diálogo**: Utiliza el componente x-jet-dialog-modal de Laravel Jetstream para crear un modal de diálogo. Este modal se muestra cuando la variable de Livewire editForm.open está activa (true), lo que indica que se está editando un lugar existente.
2. **Título del Modal**: Define el título del modal como "Editar lugar #{{ $placeId }}". Aquí, $placeId es la variable que contiene el ID del lugar que se está editando.
3. **Contenido del Modal**: Contiene un formulario para editar el nombre y la descripción del lugar. Aquí hay una descripción de los campos:
4. **Nombre**: Un campo de entrada de texto (<input>) para editar el nombre del lugar. El valor de este campo está vinculado a la variable de Livewire editForm.name.
5. **Descripción**: Un área de texto (<textarea>) para editar la descripción del lugar. El valor de este campo está vinculado a la variable de Livewire editForm.description.

6. **Botones de Acción del Modal**: Define los botones de acción del modal en la sección del footer.
7. **Actualizar**: Un botón de color azul (<x-button color="blue">) que llama a la función update de Livewire cuando se hace clic. Esta función se encarga de actualizar el lugar en la base de datos. El ID del lugar se pasa como argumento a la función update.
8. **Scripts JavaScript (al final)**: Incluye un script JavaScript para mostrar un mensaje de confirmación utilizando SweetAlert2 cuando se ha actualizado exitosamente un lugar. El evento Livewire placeUpdated se activa cuando se completa con éxito la actualización del lugar.

```
<div>
    <x-jet-dialog-modal wire:model="editForm.open">
        <x-slot name="title">
            <span class="text-2xl">Editar lugar #{{ $placeId }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div>
                    <x-jet-label>
                        Nombre
                    </x-jet-label>
                    <x-jet-input type="text" class="block p-2.5 w-full
text-sm text-gray-900 bg-gray-50 rounded-lg border border-gray-300
focus:ring-blue-500 focus:border-blue-500 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500 mt-1"
                                    wire:model="editForm.name" />
                    <x-jet-input-error for="editForm.name" class="mt-2"
/>
                </div>
                <div>
                    <x-jet-label>
                        Descripción
                    </x-jet-label>
                    <textarea wire:model="editForm.description" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
```

```
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></textarea>
                    <x-jet-input-error for="editForm.description"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button color="blue"
                      wire:click="update('{{ $placeId }}')">
                Actualizar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('placeUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El lugar ha sido actualizado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

List-places

Esta vista proporciona una interfaz completa para que los usuarios administren los lugares de la aplicación, incluida la creación, visualización, edición y eliminación de lugares, así como opciones de búsqueda y ordenamiento.

1. **Título y Botón de Agregar**: Muestra un título "Listado de lugares" y un botón "Añadir" que abre un modal para crear un nuevo lugar. El modal se activa cuando se hace clic en el botón y emite un evento para abrir el modal de creación.
2. **Barra de Búsqueda y Filtros**: Permite al usuario buscar lugares por diferentes criterios (ID, nombre, descripción, etc.). También hay un botón "Eliminar filtros" para restablecer los filtros de búsqueda.
3. **Modal de Creación de Lugar**: Un modal que se muestra cuando se hace clic en el botón "Añadir" mencionado anteriormente. Permite al usuario ingresar los detalles del nuevo lugar.
4. **Listado de Lugares**: Muestra una tabla con los lugares disponibles. Los lugares se muestran en filas, y cada fila contiene detalles como ID, nombre, descripción, creador, actualizador, fecha de creación, etc. Hay también botones para ver detalles, editar y eliminar cada lugar.
5. **Ordenamiento de Columnas**: Permite al usuario ordenar los lugares por columnas haciendo clic en el encabezado de la columna respectiva. Se muestra una flecha ascendente o descendente según el orden actual.
6. **Paginación**: Si hay más de una página de lugares, se muestra la paginación al final de la tabla.
7. **Modal de Detalles del Lugar**: Un modal que muestra detalles adicionales de un lugar específico cuando se hace clic en el botón "Ver detalles" en la fila correspondiente. Muestra información como ID, nombre, descripción, creador, actualizador, fecha de creación, fecha de actualización, etc.
8. **Scripts JavaScript (al final)**: Incluye un script JavaScript que muestra una ventana emergente de confirmación cuando se intenta eliminar un lugar. Si el usuario confirma la eliminación, se emite un evento Livewire para eliminar el lugar y se muestra un mensaje de éxito.

```
<div>
    <div class="flex items-center mb-6">
```

```
        <h1 class="text-2xl font-semibold text-gray-700">Listado de
lugares</h1>

        <button type="button"
                class="text-white bg-blue-700 hover:bg-blue-800
focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-5
py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800 ml-auto"
                wire:click="$emitTo('admin.places.create-place',
'openCreationModal')">
            Añadir
        </button>
    </div>

    <div class="mb-3">
        <div class="inline">
            <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                <option value="id">ID</option>
                <option value="name">NOMBRE</option>
                <option value="description">DESCRIPCIÓN</option>
                <option value="creator">CREADOR</option>
                <option value="updater">ACTUALIZADOR</option>
                <option value="created_at">FECHA DE CREACIÓN</option>
            </select>
        </div>

        <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                     placeholder="Buscar ..."></x-jet-input>

        <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
```

```
    </div>

    @livewire('admin.places.create-place')

    @if(count($places))
        @livewire('admin.places.edit-place')

        <x-table>
            <x-slot name="thead">
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                    ID
                    @if($sortField === 'id' && $sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'id' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('name')">
                    Nombre
                    @if($sortField === 'name' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'name' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('description')">
                    Descripción
                    @if($sortField === 'description' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'description' &&
$sortDirection === 'desc')
```

```
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('creator')">
                    Creador
                    @if($sortField === 'creator' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'creator' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updater')">
                    Actualizador
                    @if($sortField === 'updater' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'updater' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    <span class="sr-only">Actions</span>
```

```
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($places as $place)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $place->id }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $place->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $place->description }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{
\App\Models\User::find($place->creator)->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{
\App\Models\User::find($place->updater)->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $place->created_at }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap flex gap-4">
                            <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $place->id }}')">
```

```
                                <i class="fa-solid fa-eye"></i>
                            </span>
                            <span class="font-medium text-yellow-400
cursor-pointer"

wire:click="$emitTo('admin.places.edit-place', 'openEditModal', '{{
$place->id }}')">
                                <i class="fa-solid fa-pencil"></i>
                            </span>
                            <span class="font-medium text-red-500
cursor-pointer"
                                  wire:click="$emit('deletePlace', '{{
$place->id }}')">
                                <i class="fa-solid fa-trash"></i>
                            </span>
                        </td>
                    </tr>
                @endforeach
            </x-slot>
        </x-table>

        @if($places->hasPages())
            <div class="mt-6">
                {{ $places->links() }}
            </div>
        @endif
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif

    {{-- Modal show --}}
    <x-jet-dialog-modal wire:model="detailsModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalles del lugar #{{
$detailsModal['id'] }}</span>
        </x-slot>

        <x-slot name="content">
```

```
            <div class="space-y-3">
                <div>
                    <x-jet-label>
                        ID: {{ $detailsModal['id']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Nombre: {{ $detailsModal['name']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Descripción: {{ $detailsModal['description']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Creador: {{ $detailsModal['creatorName'] }} ({{
$detailsModal['creatorId'] }})
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Actualizador:
                        @if($detailsModal['updaterName'])
                            {{ $detailsModal['updaterName'] }} ({{
$detailsModal['updaterId'] }})
                        @else
                            {{ 'ninguno' }}
                        @endif
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de creación: {{ $detailsModal['createdAt']
}}
                    </x-jet-label>
```

```
                </div>
                <div>
                    <x-jet-label>
                        Última actualización: {{
$detailsModal['updatedAt'] }}
                    </x-jet-label>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$set('detailsModal.open', false)">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('deletePlace', placeId => {
                Swal.fire({
                    title: '¿Quieres eliminar este lugar?',
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {
                        Livewire.emitTo('admin.places.list-places',
'delete', placeId)
                        Swal.fire(
                            '¡Hecho!',
                            'El lugar ha sido eliminado.',
                            'success'
                        )
```

```
                    }
                })
            });
        </script>
    @endpush
</div>
```

Point

Create-point

Este modal es parte de una interfaz de usuario que permite a los usuarios añadir nuevos puntos de interés en la aplicación. Incluye validación en tiempo real, selección dinámica de lugares, y notificaciones al usuario sobre el éxito de la operación.

1. **Título del Modal**: El título del modal es "Añadir Punto de Interes".
2. **Contenido del Modal**:
3. **Campo Nombre**: Un campo de entrada para el nombre del punto de interés. Está vinculado al modelo de Livewire `createForm.name` y muestra errores de validación si los hay.
4. **Campo Distancia**: Un campo de entrada para la distancia del punto de interés, vinculado a `createForm.distance` y muestra errores de validación.
5. **Campo Longitud**: Un campo de entrada para la longitud del punto de interés, vinculado a `createForm.longitude` y muestra errores de validación.
6. **Campo Latitud**: Un campo de entrada para la latitud del punto de interés, vinculado a `createForm.latitude` y muestra errores de validación.
7. **Campo Lugar**: Un campo de selección para asociar el punto de interés a un lugar específico. Los lugares se cargan dinámicamente de la variable `places` y se vincula a `createForm.place`.
8. **Pie de Modal**:
9. **Botón Crear**: Un botón que, al hacer clic, llama al método `save` en el componente de Livewire para guardar el nuevo punto de interés.
10. **Scripts JavaScript**:

11. **Evento PointCreated**: Un script que escucha el evento `PointCreated` de Livewire. Cuando se emite este evento, se muestra una alerta de éxito usando SweetAlert para notificar al usuario que el punto de interés ha sido creado exitosamente.

```
<div>
    <x-jet-dialog-modal wire:model="createForm.open">
        <x-slot name="title">
            <span class="text-2xl">Añadir Punto de Interes</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div>
                    <x-jet-label>
                        Nombre
                    </x-jet-label>
                    <input wire:model="createForm.name" class="block
p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                    <x-jet-input-error for="createForm.name"
class="mt-2" />

                    <x-jet-label>
                        Distancia
                    </x-jet-label>
                    <input wire:model="createForm.distance" class="block
p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                    <x-jet-input-error for="createForm.distance"
class="mt-2" />
```

```
                </div>
                <div>
                    <x-jet-label>
                        Longitud
                    </x-jet-label>
                    <input wire:model="createForm.longitude" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></input>
                    <x-jet-input-error for="createForm.longitude"
class="mt-2" />
                </div>

                <div>
                    <x-jet-label>
                        Latitud
                    </x-jet-label>
                    <input wire:model="createForm.latitude" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></input>
                    <x-jet-input-error for="createForm.latitude"
class="mt-2" />
                </div>

                <div>
                    <x-jet-label>
                        Lugar
                    </x-jet-label>
                    <select wire:model="createForm.place"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
```

```
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
uno</option>
                        @foreach($places as $place)
                            <option value="{{ $place->id }}">{{
$place->name }}</option>
                        @endforeach
                    </select>
                    <x-jet-input-error for="createForm.place"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button color="blue" wire:click="save">
                Crear
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('PointCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El Punto ha sido creado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

Edit-point

Este modal es parte de una interfaz de usuario que permite a los usuarios editar puntos de interés existentes en la aplicación. Incluye validación en tiempo real, selección dinámica de lugares, y notificaciones al usuario sobre el éxito de la operación.

Título del Modal: El título del modal es "Editar punto de interés #{{ $pointId }}", donde `{{ $pointId }}` se reemplaza dinámicamente con el ID del punto de interés que se está editando.
Contenido del Modal:

1. **Campo Nombre**: Un campo de entrada para el nombre del punto de interés, vinculado al modelo de Livewire `editForm.name` y muestra errores de validación si los hay.
2. **Campo Distancia**: Un campo de entrada para la distancia del punto de interés, vinculado a `editForm.distance` y muestra errores de validación.
3. **Campo Longitud**: Un campo de entrada para la longitud del punto de interés, vinculado a `editForm.longitude` y muestra errores de validación.
4. **Campo Latitud**: Un campo de entrada para la latitud del punto de interés, vinculado a `editForm.latitude` y muestra errores de validación.
5. **Campo Sitio**: Un campo de selección para asociar el punto de interés a un lugar específico, cargando dinámicamente los lugares desde la variable `places`, vinculado a `editForm.place`.
6. **Pie del Modal**:
7. **Botón Actualizar**: Un botón que, al hacer clic, llama al método `update('{{ $pointId }}')` en el componente de Livewire para guardar los cambios en el punto de interés.
8. **Scripts JavaScript**:
9. **Evento pointUpdated**: Un script que escucha el evento `pointUpdated` de Livewire. Cuando se emite este evento, se muestra una alerta de éxito usando SweetAlert para notificar al usuario que el punto de interés ha sido actualizado exitosamente.

```
<div>
    <x-jet-dialog-modal wire:model="editForm.open">
        <x-slot name="title">
```

```
            <span class="text-2xl">Editar punto de interés #{{ $pointId
}}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div>
                    <x-jet-label>
                        Nombre
                    </x-jet-label>
                    <input wire:model="editForm.name" class="block p-2.5
w-full text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                    <x-jet-input-error for="editForm.name" class="mt-2"
/>

                    <x-jet-label>
                        Distancia
                    </x-jet-label>
                    <input wire:model="editForm.distance" class="block
p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                    <x-jet-input-error for="editForm.distance"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Longitud
                    </x-jet-label>
                    <input wire:model="editForm.longitude" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
```

```
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></input>
                    <x-jet-input-error for="editForm.longitude"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Latitud
                    </x-jet-label>
                    <input wire:model="editForm.latitude" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></input>
                    <x-jet-input-error for="editForm.latitude"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Sitio
                    </x-jet-label>
                    <select wire:model="editForm.place"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
uno</option>
                        @foreach($places as $place)
                            <option value="{{ $place->id }}">{{
$place->name }}</option>
                        @endforeach
                    </select>
                    <x-jet-input-error for="editForm.place" class="mt-2"
/>
```

```
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button color="blue" wire:click="update('{{ $pointId
}}')">
                Actualizar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('pointUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El punto ha sido actualizado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

Show-point

Este componente de interfaz de usuario es parte de un sistema administrativo para gestionar puntos de interés.

1. **Encabezado**: Título "Listado de puntos de interés". Botón "Añadir" que abre un modal para crear un nuevo punto de interés mediante un evento de Livewire.
2. **Filtros y búsqueda**: Menú desplegable para seleccionar la columna por la que se desea filtrar (ID, distancia, latitud, longitud, sitio, creador, actualizador, fecha de

creación, fecha de actualización). Campo de texto para introducir términos de búsqueda. Botón "Eliminar filtros" para restablecer los filtros aplicados.

3. **Tabla de puntos de interés**: Si hay puntos de interés, se muestran en una tabla con las siguientes columnas: QR, ID, Nombre, Distancia, Latitud, Longitud, Sitio, Creador, Actualizador, Fecha de creación, y Acciones. Cada columna es ordenable al hacer clic en su encabezado. En cada fila, los datos de los puntos de interés se muestran junto con acciones disponibles: Icono de lápiz para editar el punto (abre un modal de edición). Icono de basura para eliminar el punto (abre un cuadro de diálogo de confirmación). Icono de globo para ver la ubicación en Google Maps.
4. **Modal de detalles**: Modal para mostrar detalles específicos de un punto de interés seleccionado, incluyendo nombre, distancia, latitud, longitud, sitio, creador, actualizador, fecha de creación y de actualización. Botón para cerrar el modal.
5. **Paginación**: Si la lista de puntos de interés es larga, se proporciona paginación.
6. **Script de confirmación**:Script para mostrar un cuadro de diálogo de confirmación al intentar eliminar un punto de interés, usando SweetAlert.

```
<div>
    <div class="flex items-center mb-6">
        <h1 class="text-2xl font-semibold text-gray-700">Listado de
puntos de interés</h1>

        <button type="button"
                class="ml-12 text-white bg-blue-700 hover:bg-blue-800
focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-5
py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800 ml-auto"
                wire:click="$emitTo('admin.point.create-point',
'openCreationModal')">
            Añadir
        </button>
    </div>

    <div class="mb-3">
        <div class="inline">
            <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
```

```
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                <option value="id">ID</option>
                <option value="distance">DISTANCIA</option>
                <option value="latitude">LATITUD</option>
                <option value="longitude">LONGITUD</option>
                <option value="place_id">SITIO</option>
                <option value="creator">CREADOR</option>
                <option value="updater">ACTUALIZADOR</option>
                <option value="created_at">FECHA DE CREACIÓN</option>
                <option value="updated_at">FECHA DE
ACTUALIZACIÓN</option>
            </select>
        </div>

        <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                     placeholder="Buscar ..."></x-jet-input>

        <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
    </div>

    @livewire('admin.point.create-point')

    @if(count($points))
        @livewire('admin.point.edit-point')

        <x-table>
            <x-slot name="thead">
                <th scope="col" class="px-6 py-3">
                    QR
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                    ID
                    @if($sortField === 'id' && $sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
```

```
                        @elseif($sortField === 'id' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    Nombre
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('distance')">
                    Distancia
                    @if($sortField === 'distance' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'distance' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('latitude')">
                    Latitud
                    @if($sortField === 'latitude' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'latitude' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('longitude')">
                    Longitud
                    @if($sortField === 'longitude' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'longitude' &&
$sortDirection === 'desc')
```

```
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('place_id')">
                    Sitio
                    @if($sortField === 'place_id' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'place_id' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('creator')">
                    Creador
                    @if($sortField === 'creator' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'creator' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updater')">
                    Actualizador
                    @if($sortField === 'updater' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'updater' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
```

```
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    <span class="sr-only">Actions</span>
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($points as $point)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">

{!!QrCode::size(100)->generate("https://www.google.com/maps?q={$point->
latitude},{$point->longitude}") !!}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->id}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->name}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->distance}}
                        </td>
```

```
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->latitude}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->longitude}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->place->name}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">

{{\App\Models\User::find($point->creator)->name}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if($point->updater)

{{\App\Models\User::find($point->updater)->name}}
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{$point->created_at}}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap flex gap-4 mt-10">
                            <span class="font-medium text-yellow-400
cursor-pointer"

wire:click="$emitTo('admin.point.edit-point', 'openEditModal',
'{{$point->id}}')">
                                <i class="fa-solid fa-pencil"></i>
                            </span>
```

```
                            <span class="font-medium text-red-500
cursor-pointer"
                                  wire:click="$emit('deletePoint',
'{{$point->id}}')">
                                <i class="fa-solid fa-trash"></i>
                            </span>
                            <span class="font-medium text-blue-500
cursor-pointer">
                                <a
href="https://www.google.com/maps?q={{$point->latitude}},{{$point->long
itude}}">
                                    <i class="fas fa-globe"></i>
                                </a>
                            </span>
                        </td>
                    </tr>
                @endforeach
            </x-slot>
        </x-table>

        @if($points->hasPages())
            <div class="mt-6">
                {{ $points->links() }}
            </div>
        @endif
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif

    {{-- Modal show --}}
    <x-jet-dialog-modal wire:model="detailsModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalles del Punto de Interes #{{
$detailsModal['id'] }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-3">
```

```
                <div>
                    <x-jet-label>
                        Nombre: {{ $detailsModal['name']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Distancia: {{ $detailsModal['distance']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Latitud: {{ $detailsModal['latitude'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Longitud: {{ $detailsModal['longitude'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Sitio: {{ $detailsModal['placeName'] }} ({{
$detailsModal['placeId'] }})
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Creador: {{ $detailsModal['creatorName'] }} ({{
$detailsModal['creatorId'] }})
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Actualizador:
                        @if($detailsModal['updaterName'])
                            {{ $detailsModal['updaterName'] }} ({{
$detailsModal['updaterId'] }})
```

```
                        @else
                            {{ 'ninguno' }}
                        @endif
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de creación: {{ $detailsModal['createdAt']
}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de actualización: {{
$detailsModal['updatedAt'] }}
                    </x-jet-label>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$set('detailsModal.open', false)">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('deletePoint', pointId => {
                Swal.fire({
                    title: '¿Quieres eliminar este punto?',
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
```

```
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {
                        Livewire.emitTo('admin.point.show-point',
'delete', pointId)
                        Swal.fire(
                            '¡Hecho!',
                            'El punto ha sido borrado.',
                            'success'
                        )
                    }
                })
            });
        </script>
    @endpush
</div>
```

Thematic-area

Thematic-areas

Este componente de interfaz proporciona todas las herramientas necesarias para la gestión completa de áreas temáticas, incluyendo creación, visualización, edición, eliminación y filtrado de áreas temáticas.

1. **Título y Botón de Añadir**: Un encabezado que dice "Listado de áreas temáticas". Un botón "Añadir" que abre el formulario de creación de un área temática.
2. **Filtros de Búsqueda**: Un selector para elegir el criterio de búsqueda (ID, nombre, descripción, fecha de creación, fecha de actualización). Un campo de entrada de texto para realizar la búsqueda. Un botón para eliminar los filtros de búsqueda.
3. **Tabla de Listado**:Una tabla que muestra las áreas temáticas con columnas para ID, nombre, descripción, fecha de creación y fecha de actualización. Las columnas son ordenables, es decir, se puede hacer clic en los encabezados para ordenar los datos ascendente o descendentemente. Acciones disponibles por cada área temática: ver detalles, editar y eliminar.

4. **Modales**:
    a. **Modal de Detalle**: Muestra información detallada de un área temática específica.
    b. **Modal de Creación**: Formulario para crear una nueva área temática, con campos para nombre y descripción.
    c. **Modal de Edición**: Formulario para actualizar una área temática existente, con campos para nombre y descripción.
5. **Manejo de Eventos con SweetAlert2**: Al crear, actualizar o eliminar un área temática, se muestra una alerta de éxito. Al intentar eliminar un área temática, se muestra una alerta de confirmación para asegurar que el usuario desea proceder con la acción.
6. **Script para Manejo de Eventos**: El código usa `Livewire` para manejar eventos y la comunicación entre los componentes de la interfaz y el backend. Las alertas y confirmaciones son manejadas mediante `SweetAlert2`.

```
<div>
    <div>
        <div class="flex items-center mb-6">
            <h1 class="text-2xl font-semibold text-gray-700">Listado de
áreas temáticas</h1>

            <button wire:click="$toggle('createForm.open')"
type="button" class="text-white bg-blue-700
                hover:bg-blue-800 focus:ring-4 focus:ring-blue-300
font-medium rounded-lg text-sm
                px-5 py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700
focus:outline-none
                dark:focus:ring-blue-800 ml-auto">
                Añadir
            </button>
        </div>

        <div class="mb-3">
            <div class="inline">
                <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
```

```
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                    <option value="id">ID</option>
                    <option value="name">NOMBRE</option>
                    <option value="description">DESCRIPCIÓN</option>
                    <option value="created_at">FECHA DE
CREACIÓN</option>
                    <option value="updated_at">FECHA DE
ACTUALIZACIÓN</option>
                </select>
            </div>

            <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                placeholder="Buscar ..."></x-jet-input>

            <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>

        </div>

        @if(count($thematicAreas))
            <x-table>
                <x-slot name="thead">
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                        ID
                        @if($sortField === 'id' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'id' && $sortDirection
=== 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('name')">
                        Nombre
```

```
                        @if($sortField === 'name' && $sortDirection ===
'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'name' && $sortDirection
=== 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('description')">
                        Descripción
                        @if($sortField === 'description' &&
$sortDirection === 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'description' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                        Fecha creación
                        @if($sortField === 'created_at' &&
$sortDirection === 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                        @endif
                    </th>
                    <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updated_at')">
                        Fecha actualización
                        @if($sortField === 'updated_at' &&
$sortDirection === 'asc')
                            <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'updated_at' &&
$sortDirection === 'desc')
```

```
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    Acciones
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($thematicAreas as $thematicArea)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $thematicArea->id }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $thematicArea->name }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $thematicArea->description }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">
                            {{ $thematicArea->created_at }}
                        </td>
                        <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap">

                            @if($thematicArea->updated_at)
                                {{ $thematicArea->updated_at }}
                            @else
                                <p class="text-center"> N/A</p>
                            @endif
                        </td>
```

```
                            <td class="px-6 py-4 font-medium
text-gray-900 dark:text-white whitespace-nowrap flex gap-4">
                                <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $thematicArea->id }}')">
                                    <i class="fa-solid fa-eye"></i>
                                </span>
                                    <span class="font-medium
text-yellow-400 cursor-pointer"
                                            wire:click="edit('{{
$thematicArea->id }}')">
                                    <i class="fa-solid fa-pencil"></i>
                                </span>
                                    <span class="font-medium
text-red-500 cursor-pointer"

wire:click="$emit('deleteThematicArea', '{{ $thematicArea->id }}')">
                                    <i class="fa-solid fa-trash"></i>
                                </span>
                            </td>
                        </tr>
                    @endforeach
                </x-slot>
            </x-table>

            @if($thematicAreas->hasPages())
                <div class="mt-6">
                    {{ $thematicAreas->links() }}
                </div>
            @endif
        @else
            <p class="mt-4">No se han encontrado resultados</p>
        @endif
    </div>

    <x-jet-dialog-modal wire:model="showModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalle del área temática #{{
$showModal['id'] }}</span>
```

```
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div class="mb-4">
                    <x-jet-label>
                        Nombre: {{ $showModal['name'] }}
                    </x-jet-label>
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Descripción: {{ $showModal['description'] }}
                    </x-jet-label>
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Fecha de creación: {{ $showModal['createdAt'] }}
                    </x-jet-label>
                </div>

                <div class="mb-4">
                    <x-jet-label>
                        Fecha de actualización: {{
$showModal['updatedAt'] }}
                    </x-jet-label>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$toggle('showModal.open')">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>
```

```
    <x-jet-dialog-modal wire:model="createForm.open">
        <x-slot name="title">
            <span class="text-2xl">Crear área temática</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div class="mb-4">
                    <x-jet-label>
                        Nombre
                    </x-jet-label>

                    <input wire:model="createForm.name" type="text"
id="name" minlength="1" class="bg-gray-50 border border-gray-300
text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>

                    <x-jet-input-error for="createForm.name"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Descripción
                    </x-jet-label>

                    <textarea wire:model="createForm.description"
rows="4" class="block p-2.5 w-full
                    text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500
                    focus:border-blue-500 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400
                    dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1"></textarea>

                    <x-jet-input-error for="createForm.description"
class="mt-2" />
```

```
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button style="margin-right: 10px;" wire:click="save">
                Crear
            </x-button>
            <x-button wire:click="$toggle('createForm.open')">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    <x-jet-dialog-modal wire:model="editModal.open">
        <x-slot name="title">
            <span class="text-2xl">Actualizar área temática</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                <div class="mb-4">
                    <x-jet-label>
                        Nombre
                    </x-jet-label>

                    <input wire:model="editForm.name" type="text"
id="name" minlength="1" class="bg-gray-50 border border-gray-300
text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>

                    <x-jet-input-error for="editForm.name" class="mt-2"
/>
                </div>
                <div>
                    <x-jet-label>
```

```
                        Descripción
                    </x-jet-label>

                    <textarea wire:model="editForm.description" rows="4"
class="block p-2.5 w-full
                    text-sm text-gray-900 bg-gray-50 rounded-lg border
border-gray-300 focus:ring-blue-500
                    focus:border-blue-500 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400
                    dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1"></textarea>

                    <x-jet-input-error for="editForm.description"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button style="margin-right: 10px;" wire:click="update({{
$editModal['id'] }})">
                Actualizar
            </x-button>
            <x-button wire:click="$toggle('editModal.open')">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>

        <script>
            Livewire.on('thematicAreaCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El área temática ha sido creada.',
                    'success'
```

```
                )
            });
        </script>

        <script>
            Livewire.on('thematicAreaUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El área temática ha sido actualizada.',
                    'success'
                )
            });
        </script>

        <script>
            Livewire.on('deleteThematicArea', thematicAreaId => {
                Swal.fire({
                    title: '¿Quieres eliminar esta área temática?',
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {

Livewire.emitTo('admin.thematic-area.thematic-areas', 'delete',
thematicAreaId)
                        Swal.fire(
                            '¡Hecho!',
                            'El área temática ha sido eliminada.',
                            'success'
                        )
                    }
                })
            });
```

```
        </script>
    @endpush
</div>
```

User

Create-user

Esta vista proporciona una interfaz para agregar un nuevo usuario, valida los datos ingresados y muestra una notificación cuando se completa con éxito la creación del usuario.

1. Muestra un modal con el título "Añadir usuario" y varios campos para ingresar información del usuario.
2. Permite cargar un avatar del usuario mediante un campo de entrada de tipo archivo.
3. Solicita al usuario ingresar su nombre, rol, email y contraseña, junto con la confirmación del email y la contraseña.
4. Muestra mensajes de error si alguno de los campos no se completa correctamente.
5. Cuando se hace clic en el botón "Crear", se activa el método `save` en Livewire, el cual guarda la información del usuario.
6. Después de que se crea exitosamente un usuario, se muestra una notificación utilizando SweetAlert2 con el mensaje "El usuario ha sido creado".

```
<div>
    <x-jet-dialog-modal wire:model="createForm.open">
```

```
        <x-slot name="title">
            <span class="text-2xl">Añadir usuario</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                @if($avatarTemporaryUrl)
                    <img class="w-36 h-36 rounded-full mx-auto" src="{{
$avatarTemporaryUrl }}" alt="User avatar">
                @endif

                <div>
                    <x-jet-label>
                        Avatar
                    </x-jet-label>
                    <input wire:model="createForm.avatar" type="file"
accept=".png, .jpg" class="block w-full text-sm text-gray-900
bg-gray-50 rounded-lg border border-gray-300 cursor-pointer
dark:text-gray-400 focus:outline-none dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 mt-1">
                    <p wire:loading
wire:target="createForm.avatar">Subiendo...</p>
                    <x-jet-input-error for="createForm.avatar"
class="mt-2" />
                </div>

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
                        <label for="name" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Nombre</label>
                        <input wire:model="createForm.name" type="text"
id="name" minlength="1" class="bg-gray-50 border border-gray-300
text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>
                        <x-jet-input-error for="createForm.name"
class="mt-2" />
```

```
                    </div>
                    <div>
                        <label for="role" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Role</label>
                        <select wire:model="createForm.role"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                            <option value="" selected disabled>Elige
uno</option>
                            @foreach($roles as $role)
                                <option value="{{ $role->id }}">{{
$role->name }}</option>
                            @endforeach
                        </select>
                        <x-jet-input-error for="createForm.role"
class="mt-2" />
                    </div>
                </div>

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
                        <label for="email" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Email</label>
                        <input wire:model="createForm.email"
type="email" id="email" minlength="1" maxlength="45" class="bg-gray-50
border border-gray-300 text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                        <x-jet-input-error for="createForm.email"
class="mt-2" />
                    </div>
                    <div>
```

```
                        <label for="email_confirmation" class="block
mb-2 text-sm font-medium text-gray-900 dark:text-gray-300">Confirmar
email</label>
                        <input
wire:model="createForm.email_confirmation" type="email"
id="email_confirmation" minlength="1" maxlength="45" class="bg-gray-50
border border-gray-300 text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                    </div>
                </div>

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
                        <label for="password" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Contraseña</label>
                        <input wire:model="createForm.password"
type="password" id="password" minlength="8" maxlength="45"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                        <x-jet-input-error for="createForm.password"
class="mt-2" />
                    </div>
                    <div>
                        <label for="password_confirmation" class="block
mb-2 text-sm font-medium text-gray-900 dark:text-gray-300">Confirmar
contraseña</label>
                        <input
wire:model="createForm.password_confirmation" type="password"
id="password_confirmation" minlength="8" maxlength="45"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
```

```
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                    </div>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="save">
                Crear
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('userCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El usuario ha sido creado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

Edit-user

Este código proporciona una interfaz de usuario para editar la información de un usuario, incluida la capacidad de actualizar los detalles y recibir una notificación visual cuando se completa la acción.

1. **Componente Modal de Edición de Usuario**: Utiliza el componente `x-jet-dialog-modal` de Laravel Livewire para mostrar un modal de edición de usuario. El modal se activa o desactiva según el estado de `editForm.open`.
2. **Contenido del Modal**: Muestra la información del usuario, como su nombre, rol, correo electrónico y avatar. Si el usuario tiene un avatar, se muestra la imagen; de lo contrario, se muestra un ícono predeterminado. Se incluyen campos de entrada para editar el nombre, el rol, el correo electrónico y la contraseña del usuario.
3. **Selección de Rol**: Permite al usuario seleccionar el rol del usuario a partir de opciones proporcionadas en un menú desplegable. Las opciones de rol se obtienen de una variable `$roles`, que se espera que contenga información sobre los roles disponibles.
4. **Actualización de Usuario**: Cuando se hace clic en el botón "Actualizar", se llama al método `update` de Livewire, pasando el ID del usuario como parámetro. Se espera que este método procese los cambios realizados por el usuario y actualice la información del usuario en el sistema.
5. **Notificación de Actualización**: Utiliza el evento Livewire `userUpdated` para mostrar una notificación al usuario después de que se haya actualizado la información del usuario. Se utiliza SweetAlert2 (`Swal.fire`) para mostrar un mensaje de éxito cuando se completa la actualización.
6. **Scripts y Estilos Adicionales**: Se incluye un bloque `@push('scripts')` que contiene un script JavaScript para manejar el evento `userUpdated`.Este script probablemente está diseñado para mostrar una notificación visual después de que se haya actualizado el usuario.

```
<div>
    <x-jet-dialog-modal wire:model="editForm.open">
        <x-slot name="title">
            <span class="text-2xl">Editar usuario #{{ $userId }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                @if($avatarRoute)
```

```
                    <img class="w-36 h-36 rounded-full mx-auto" src="{{
$avatarRoute }}" alt="User avatar">
                @else
                    <div class="relative w-36 h-36 overflow-hidden
bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">
                        <svg class="w-36 h-36 text-gray-400"
fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>
                    </div>
                @endif

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
                        <label for="name" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Nombre</label>
                        <input wire:model="editForm.name" type="text"
id="name" minlength="1" class="bg-gray-50 border border-gray-300
text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>
                        <x-jet-input-error for="editForm.name"
class="mt-2" />
                    </div>
                    <div>
                        <label for="role" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Role</label>
                        <select wire:model="editForm.role"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                            <option value="" disabled>Elige uno</option>
                            @foreach($roles as $role)
```

```
                                <option value="{{ $role->id }}">{{
$role->name }}</option>
                            @endforeach
                        </select>
                        <x-jet-input-error for="editForm.role"
class="mt-2" />
                    </div>
                </div>

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
                        <label for="email" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Email</label>
                        <input wire:model="editForm.email" type="email"
id="email" minlength="1" maxlength="45" class="bg-gray-50 border
border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>
                        <x-jet-input-error for="editForm.email"
class="mt-2" />
                    </div>
                    <div>
                        <label for="email_confirmation" class="block
mb-2 text-sm font-medium text-gray-900 dark:text-gray-300">Confirmar
email</label>
                        <input wire:model="editForm.email_confirmation"
value="{{ $editForm['email'] }}" type="email" id="email_confirmation"
minlength="1" maxlength="45" class="bg-gray-50 border border-gray-300
text-gray-900 text-sm rounded-lg focus:ring-blue-500
focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700
dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500" required>
                    </div>
                </div>

                <div class="grid gap-6 mb-6 lg:grid-cols-2">
                    <div>
```

```
                        <label for="password" class="block mb-2 text-sm
font-medium text-gray-900 dark:text-gray-300">Contraseña</label>
                        <input wire:model="editForm.password"
type="text" id="password" minlength="8" maxlength="45"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                        <x-jet-input-error for="editForm.password"
class="mt-2" />
                    </div>
                    <div>
                        <label for="password_confirmation" class="block
mb-2 text-sm font-medium text-gray-900 dark:text-gray-300">Confirmar
contraseña</label>
                        <input
wire:model="editForm.password_confirmation" type="text"
id="password_confirmation" minlength="8" maxlength="45"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
required>
                    </div>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="update('{{ $userId }}')">
                Actualizar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
```

```
            Livewire.on('userUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El usuario ha sido actualizado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

List-users

Esta vista proporciona una interfaz completa para administrar usuarios en un sistema web, incluida la capacidad de agregar, ver, editar, eliminar y buscar usuarios, así como ordenarlos y mostrar detalles específicos de cada usuario.

1. **Encabezado y Botón de Añadir Usuario**: Muestra un título "Listado de usuarios" y un botón "Añadir" para permitir la creación de nuevos usuarios. El botón "Añadir" solo se muestra si el usuario actual tiene el rol de "Administrador".
2. **Barra de Búsqueda y Filtros**: Permite al usuario buscar usuarios por diferentes criterios, como ID, nombre, fecha de creación o fecha de actualización. Se proporciona un campo de entrada y un menú desplegable para seleccionar el criterio de búsqueda. Un botón "Eliminar filtros" restablece los filtros de búsqueda a sus valores predeterminados.
3. **Creación de Usuario**: Incluye un componente de creación de usuario `admin.user.create-user` que se muestra cuando se hace clic en el botón "Añadir".
4. **Tabla de Usuarios**:Muestra una tabla con la información de los usuarios, incluidos campos como ID, nombre, correo electrónico, rol, fecha de creación y fecha de actualización. Los usuarios se muestran en filas, y cada fila incluye iconos para ver, editar y eliminar el usuario correspondiente. Los nombres de las columnas son

clicables y permiten ordenar la tabla ascendente o descendentemente según el campo seleccionado.

5. **Detalles del Usuario**: Al hacer clic en el icono de "ver", se muestra un modal con los detalles del usuario seleccionado, como su nombre, correo electrónico, rol y fechas de creación y actualización.
6. **Paginación**: Si hay más usuarios de los que se pueden mostrar en una sola página, se incluye un sistema de paginación para navegar entre las diferentes páginas de resultados.
7. **Eliminación de Usuario**: Utiliza SweetAlert2 (`Swal.fire`) para confirmar la eliminación de un usuario. Al confirmar la eliminación, se emite un evento Livewire `deleteUser` para procesar la eliminación del usuario. Se muestra una notificación de éxito después de que el usuario ha sido eliminado.
8.

```
    <div>
    <div class="flex items-center mb-6">
        <h1 class="text-2xl font-semibold text-gray-700">Listado de
usuarios</h1>

        @hasanyrole('Administrador')
        <button type="button"
                class="text-white bg-blue-700 hover:bg-blue-800
focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-5
py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800 ml-auto"
                wire:click="$emitTo('admin.user.create-user',
'openCreationModal')">
            Añadir
        </button>
        @endhasanyrole
    </div>

    <div class="mb-3">
        <div class="inline">
            <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
```

```
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                <option value="id">ID</option>
                <option value="name">NOMBRE</option>
                <option value="created_at">FECHA DE CREACIÓN</option>
                <option value="updated_at">FECHA DE
ACTUALIZACIÓN</option>
            </select>
        </div>

        <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                     placeholder="Buscar ..."></x-jet-input>

        <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
    </div>

    @livewire('admin.user.create-user')

    @if(count($users))
        @livewire('admin.user.edit-user')

        <x-table>
            <x-slot name="thead">
                <th scope="col" class="px-6 py-3"></th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                    ID
                    @if($sortField === 'id' && $sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'id' && $sortDirection ===
'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('name')">
```

```
                    Nombre
                    @if($sortField === 'name' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'name' && $sortDirection ===
'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('email')">
                    Email
                    @if($sortField === 'email' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'email' && $sortDirection ===
'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    Rol
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updated_at')">
                    Fecha actualización
```

```
                    @if($sortField === 'updated_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                    @elseif($sortField === 'updated_at' &&
$sortDirection === 'desc')
                        <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    <span class="sr-only">Actions</span>
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($users as $user)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if($user->profile_photo_path)
                                <img class="w-10 h-10 rounded-full"
src="{{ $user->profile_photo_path }}" alt="User avatar">
                            @else
                                <div class="relative w-10 h-10
overflow-hidden bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">
                                    <svg class="absolute w-12 h-12
text-gray-400 -left-1" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>
                                </div>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->id }}
                        </td>
```

```
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->email }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->roles->first()->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->created_at }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $user->updated_at }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap flex gap-4 mt-2">
                            <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $user->id }}')">
                                <i class="fa-solid fa-eye"></i>
                            </span>
                            <span class="font-medium text-yellow-400
cursor-pointer"

wire:click="$emitTo('admin.user.edit-user', 'openEditModal', '{{
$user->id }}')">
                                <i class="fa-solid fa-pencil"></i>
                            </span>
                            <span class="font-medium text-red-500
cursor-pointer"
                                  wire:click="$emit('deleteUser', '{{
$user->id }}')">
                                <i class="fa-solid fa-trash"></i>
```

```
                            </span>
                        </td>
                    </tr>
                @endforeach
            </x-slot>
        </x-table>

        @if($users->hasPages())
            <div class="mt-6">
                {{ $users->links() }}
            </div>
        @endif
    {{-- Modal show --}}
    <x-jet-dialog-modal wire:model="detailsModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalles del usuario #{{
$detailsModal['id'] }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-3">
                <div class="my-8">
                    @if($detailsModal['avatar'])
                        <img class="w-36 h-36 rounded-full mx-auto"
src="{{ $detailsModal['avatar'] }}" alt="User avatar">
                    @else
                        <div class="relative w-36 h-36 overflow-hidden
bg-gray-100 rounded-full dark:bg-gray-600 mx-auto">
                            <svg class="w-36 h-36 text-gray-400"
fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M10 9a3
3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z"
clip-rule="evenodd"></path></svg>
                        </div>
                    @endif
                </div>
                <div>
                    <x-jet-label>
```

```
                        Nombre: {{ $detailsModal['name']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Email: {{ $detailsModal['email'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Rol: {{ $user->roles->first()->name }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de creación: {{ $detailsModal['createdAt']
}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de actualización: {{
$detailsModal['updatedAt'] }}
                    </x-jet-label>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$set('detailsModal.open', false)">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif
```

```
    @push('scripts')
        <script>
            Livewire.on('deleteUser', userId => {
                Swal.fire({
                    title: '¿Quieres eliminar este usuario?',
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {
                        Livewire.emitTo('admin.user.list-users',
'delete', userId)
                        Swal.fire(
                            '¡Hecho!',
                            'El usuario ha sido eliminado.',
                            'success'
                        )
                    }
                })
            });
        </script>
    @endpush
</div>
```

Video

Create-video

Esta vista proporciona un formulario interactivo para que los administradores del sistema puedan agregar nuevos vídeos, ingresando información relevante como el archivo de vídeo, detalles sobre el contenido del vídeo y su categorización. Una vez que se crea el vídeo, se muestra una notificación al usuario para confirmar el éxito de la operación.

1. **Componente Modal de Creación de Vídeo**: Utiliza el componente `x-jet-dialog-modal` de Laravel Livewire para mostrar un modal de creación de vídeo. El modal se activa o desactiva según el estado de `createForm.open`.
2. **Contenido del Modal**: Muestra campos para ingresar información sobre el vídeo, como el archivo de vídeo, el punto de interés, el área temática y una descripción.
3. **Previsualización del Vídeo**: Si hay una URL temporal de vídeo disponible (`$videoTemporaryUrl`), muestra una previsualización del vídeo utilizando el componente `admin.video.video-preview`.
4. **Subida de Archivo de Vídeo**: Permite al usuario seleccionar un archivo de vídeo utilizando un campo de entrada de tipo "file". Muestra un mensaje "Subiendo..." mientras el archivo se está cargando.
5. **Selección de Punto de Interés y Área Temática**: Utiliza menús desplegables para permitir al usuario seleccionar el punto de interés y el área temática asociados con el vídeo. Las opciones para estos menús se obtienen de las variables `$pointsOfInterest` y `$thematicAreas`, respectivamente.
6. **Descripción del Vídeo**: Permite al usuario ingresar una descripción del vídeo utilizando un campo de texto multilinea.
7. **Botón de Crear Vídeo**: Al hacer clic en el botón "Crear", se llama al método `save` de Livewire para procesar la creación del vídeo.
8. **Notificación de Creación de Vídeo**: Utiliza el evento Livewire `videoCreated` para mostrar una notificación al usuario después de que se ha creado el vídeo. Se utiliza SweetAlert2 (`Swal.fire`) para mostrar un mensaje de éxito.

9. **Scripts Adicionales**: Se incluye un bloque `@push('scripts')` que contiene un script JavaScript para manejar el evento `videoCreated`.

```
<div>
    <x-jet-dialog-modal wire:model="createForm.open">
        <x-slot name="title">
            <span class="text-2xl">Añadir vídeo</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                @if($videoTemporaryUrl)
                    @livewire('admin.video.video-preview', ['route' =>
$videoTemporaryUrl])
                @endif
                <div>
                    <x-jet-label>
                        Archivo
                    </x-jet-label>
                    <input wire:model="createForm.file" type="file"
accept="video/mp4" class="block w-full text-sm text-gray-900 bg-gray-50
rounded-lg border border-gray-300 cursor-pointer dark:text-gray-400
focus:outline-none dark:bg-gray-700 dark:border-gray-600
dark:placeholder-gray-400 mt-1">
                    <p wire:loading
wire:target="createForm.file">Subiendo...</p>
                    <x-jet-input-error for="createForm.file"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Punto de interés
                    </x-jet-label>
                    <select wire:model="createForm.pointOfInterest"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
```

```
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
uno</option>
                        @foreach($pointsOfInterest as $pointOfInterest)
                            <option value="{{ $pointOfInterest->id
}}">{{ $pointOfInterest->id }}</option>
                        @endforeach
                    </select>
                    <x-jet-input-error for="createForm.pointOfInterest"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Área temática
                    </x-jet-label>
                    <select wire:model="createForm.thematicArea"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
una</option>
                        @foreach($thematicAreas as $thematicArea)
                            <option value="{{ $thematicArea->id }}">{{
$thematicArea->name }}</option>
                        @endforeach
                    </select>
                    <x-jet-input-error for="createForm.thematicArea"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Descripción
                    </x-jet-label>
                    <textarea wire:model="createForm.description"
rows="4" class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50
```

```
rounded-lg border border-gray-300 focus:ring-blue-500
focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600
dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500
dark:focus:border-blue-500 mt-1"></textarea>
                    <x-jet-input-error for="createForm.description"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="save">
                Crear
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('videoCreated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El vídeo ha sido creado.',
                    'success'
                )
            });
        </script>
    @endpush
</div>
```

Edit-video

Esta vista proporciona un formulario interactivo para que los administradores del sistema puedan editar la información de un vídeo existente, incluidos detalles como el punto de

interés, el área temática y la descripción. Una vez que se actualiza el vídeo, se muestra una notificación al usuario para confirmar el éxito de la operación.

1. **Componente Modal de Edición de Vídeo**: Utiliza el componente `x-jet-dialog-modal` de Laravel Livewire para mostrar un modal de edición de vídeo. El modal se activa o desactiva según el estado de `editForm.open`.
2. **Contenido del Modal**: Muestra campos para editar información sobre el vídeo, como el punto de interés, el área temática y la descripción. También incluye una previsualización del vídeo utilizando el componente `admin.video.video-preview`.
3. **Selección de Punto de Interés y Área Temática**: Utiliza menús desplegables para permitir al usuario seleccionar el punto de interés y el área temática asociados con el vídeo. Las opciones para estos menús se obtienen de las variables `$pointsOfInterest` y `$thematicAreas`, respectivamente. Las opciones seleccionadas se establecen según los valores actuales almacenados en el formulario de edición (`editForm`).
4. **Descripción del Vídeo**: Permite al usuario editar la descripción del vídeo utilizando un campo de texto multilinea.
5. **Botón de Actualizar Vídeo**: Al hacer clic en el botón "Actualizar", se llama al método `update` de Livewire para procesar la actualización del vídeo.
6. **Notificación de Actualización de Vídeo**:Utiliza el evento Livewire `videoUpdated` para mostrar una notificación al usuario después de que se ha actualizado el vídeo. Se utiliza SweetAlert2 (`Swal.fire`) para mostrar un mensaje de éxito.
7. **Scripts Adicionales**: Se incluye un bloque `@push('scripts')` que contiene un script JavaScript para manejar el evento `videoUpdated`.

```
<div>
    <x-jet-dialog-modal wire:model="editForm.open">
        <x-slot name="title">
            <span class="text-2xl">Editar vídeo #{{ $videoId }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-6">
                @livewire('admin.video.video-preview', ['route' =>
$videoRoute])
                <div>
                    <x-jet-label>
```

```
                        Punto de interés
                    </x-jet-label>
                    <select wire:model="editForm.pointOfInterest"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
uno</option>
                        @foreach($pointsOfInterest as $pointOfInterest)
                            <option value="{{ $pointOfInterest->id }}"
@if($pointOfInterest->id === $editForm['pointOfInterest']) {{
'selected' }} @endif>
                                {{ $pointOfInterest->id }}
                            </option>
                        @endforeach
                    </select>
                    <x-jet-input-error for="editForm.pointOfInterest"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Área temática
                    </x-jet-label>
                    <select wire:model="editForm.thematicArea"
class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1">
                        <option value="" selected disabled>Elige
una</option>
                        @if( ! is_null($thematicAreas))
                            @foreach($thematicAreas as $thematicArea)
                                <option value="{{ $thematicArea->id }}"
@if($thematicArea->id === $editForm['thematicArea']) {{ 'selected' }}
@endif>
```

```
                                    {{ $thematicArea->name }}
                                </option>
                            @endforeach
                        @endif
                    </select>
                    <x-jet-input-error for="editForm.thematicArea"
class="mt-2" />
                </div>
                <div>
                    <x-jet-label>
                        Descripción
                    </x-jet-label>
                    <textarea wire:model="editForm.description" rows="4"
class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg
border border-gray-300 focus:ring-blue-500 focus:border-blue-500
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500
mt-1"></textarea>
                    <x-jet-input-error for="editForm.description"
class="mt-2" />
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="update('{{ $videoId }}')">
                Actualizar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('videoUpdated', () => {
                Swal.fire(
                    '¡Hecho!',
                    'El vídeo ha sido actualizado.',
                    'success'
```

```
                )
            });
        </script>
    @endpush
</div>
```

List-videos

Esta vista proporciona una interfaz completa para administrar vídeos, incluida la capacidad de agregar, buscar, filtrar, ver detalles, editar y eliminar vídeos.

1. **Encabezado de la Lista de Vídeos**: Muestra el título "Listado de vídeos" junto con un botón "Añadir" para agregar nuevos vídeos. Al hacer clic en el botón "Añadir", se activa un modal para crear un nuevo vídeo.
2. **Filtros de Búsqueda y Eliminación**: Permite al usuario filtrar la lista de vídeos por diferentes criterios, como ID, descripción, punto de interés, etc. También incluye un campo de búsqueda para buscar vídeos por texto. El botón "Eliminar filtros" restablece los filtros de búsqueda y muestra todos los vídeos nuevamente.
3. **Lista de Vídeos**: Muestra una tabla con los vídeos que coinciden con los criterios de búsqueda y filtrado. Cada fila de la tabla muestra información sobre un vídeo, como ID, descripción, punto de interés, etc. Se proporcionan flechas ascendentes y descendentes junto a los encabezados de columna para indicar el orden ascendente o descendente. Se incluyen botones de acción para ver detalles, editar y eliminar cada vídeo.
4. **Modal de Detalles del Vídeo**: Se utiliza un modal para mostrar los detalles completos de un vídeo cuando se hace clic en el botón de "ver detalles". Muestra información detallada sobre el vídeo, como descripción, punto de interés, área temática, creador, etc. Incluye un botón para cerrar el modal y volver a la lista de vídeos.
5. **Scripts Adicionales**: Se incluye un script JavaScript que maneja el evento de eliminación de vídeo. Cuando un usuario intenta eliminar un vídeo, se muestra un cuadro de confirmación (usando SweetAlert2) y, si el usuario confirma, se emite un evento Livewire para eliminar el vídeo seleccionado.

```
<div>
    <div class="flex items-center mb-6">
```

```
        <h1 class="text-2xl font-semibold text-gray-700">Listado de
vídeos</h1>

        <button type="button"
                class="text-white bg-blue-700 hover:bg-blue-800
focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-5
py-2.5 dark:bg-blue-600 dark:hover:bg-blue-700 focus:outline-none
dark:focus:ring-blue-800 ml-auto"
                wire:click="$emitTo('admin.video.create-video',
'openCreationModal')">
            Añadir
        </button>
    </div>

    <div class="mb-3">
        <div class="inline">
            <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                <option value="id">ID</option>
                <option value="description">DESCRIPCIÓN</option>
                <option value="point_of_interest_id">PUNTO DE
INTERÉS</option>
                <option value="thematic_area_id">ÁREA TEMÁTICA</option>
                <option value="creator">CREADOR</option>
                <option value="updater">ACTUALIZADOR</option>
                <option value="created_at">FECHA DE CREACIÓN</option>
            </select>
        </div>

        <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                     placeholder="Buscar ..."></x-jet-input>
```

```
        <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
    </div>

    @livewire('admin.video.create-video')

    @if(count($videos))
        @livewire('admin.video.edit-video')

        <x-table>
            <x-slot name="thead">
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                    ID
                    @if($sortField === 'id' && $sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'id' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('description')">
                    Descripción
                    @if($sortField === 'description' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'description' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('point_of_interest_id')">
                    Punto de interés
                    @if($sortField === 'point_of_interest_id' &&
$sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
```

```
                            @elseif($sortField ===
'point_of_interest_id' && $sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    Orden
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('thematic_area_id')">
                    Área temática
                    @if($sortField === 'thematic_area_id' &&
$sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'thematic_area_id' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('creator')">
                    Creador
                    @if($sortField === 'creator' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'creator' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('updater')">
                    Actualizador
                    @if($sortField === 'updater' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'updater' &&
$sortDirection === 'desc')
```

```
                            <i class="fa-solid fa-arrow-down"></i>

                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    <span class="sr-only">Actions</span>
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($videos as $video)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $video->id }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $video->description }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if( ! is_null($video->pointOfInterest))
                                {{ $video->pointOfInterest->id }}
                            @else
```

```
                                <span
class="text-red-600">Ninguno</span>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $video->order }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if( ! is_null($video->thematic_area_id) &&
! is_null($video->pointOfInterest))
                                {{ $video->thematicArea->name }}
                            @else
                               <span class="text-red-600">Ninguna</span>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{
\App\Models\User::find($video->creator)->name }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if($video->updater)
                                {{
\App\Models\User::find($video->updater)->name }}
                            @else
                                <p class="text-center"> N/A</p>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $video->created_at }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap flex gap-4">
```

```
                            <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $video->id }}')">
                                <i class="fa-solid fa-eye"></i>
                            </span>
                            <span class="font-medium text-yellow-400
cursor-pointer"

wire:click="$emitTo('admin.video.edit-video', 'openEditModal', '{{
$video->id }}')">
                                <i class="fa-solid fa-pencil"></i>
                            </span>
                            <span class="font-medium text-red-500
cursor-pointer"
                               wire:click="$emit('deleteVideo', '{{
$video->id }}')">
                                <i class="fa-solid fa-trash"></i>
                            </span>
                        </td>
                    </tr>
                @endforeach
            </x-slot>
        </x-table>

        @if($videos->hasPages())
            <div class="mt-6">
                {{ $videos->links() }}
            </div>
        @endif
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif

    {{-- Modal show --}}
    <x-jet-dialog-modal wire:model="detailsModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalles del vídeo #{{
$detailsModal['id'] }}</span>
        </x-slot>
```

```
        <x-slot name="content">
            <div class="space-y-3">
                @if($detailsModal['route'] !== null)
                    @livewire('admin.video.video-preview', ['route' =>
$detailsModal['route']])
                @endif
                <div>
                    <x-jet-label>
                        Descripción: {{ $detailsModal['description']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Ruta: {{ $detailsModal['route'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Orden: {{ $detailsModal['order'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        @if( ! empty($detailsModal['pointOfInterest']))
                            Punto de interés: {{
$detailsModal['pointOfInterest'] }}
                        @else
                            Punto de interés: <span
class="text-red-600">Ninguno</span>
                        @endif
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        @if( ! empty($detailsModal['thematicAreaId']) &&
! empty($detailsModal['pointOfInterest']))
```

```
                            Área temática: {{
$detailsModal['thematicAreaName'] }} ({{
$detailsModal['thematicAreaId'] }})
                        @else
                            Área temática: <span
class="text-red-600">Ninguna</span>
                        @endif
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Creador: {{ $detailsModal['creatorName'] }} ({{
$detailsModal['creatorId'] }})
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Actualizador:
                        @if($detailsModal['updaterName'])
                            {{ $detailsModal['updaterName'] }} ({{
$detailsModal['updaterId'] }})
                        @else
                            {{ 'ninguno' }}
                        @endif
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de creación: {{ $detailsModal['createdAt']
}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Fecha de actualización: {{
$detailsModal['updatedAt'] }}
                    </x-jet-label>
                </div>
```

```
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$set('detailsModal.open', false)">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>

    @push('scripts')
        <script>
            Livewire.on('deleteVideo', videoId => {
                Swal.fire({
                    title: '¿Quieres eliminar este vídeo?',
                    text: 'Esta operación es irreversible',
                    icon: 'warning',
                    showCancelButton: true,
                    cancelButtonText: "Cancelar",
                    confirmButtonColor: '#d33',
                    cancelButtonColor: '#3085d6',
                    confirmButtonText: 'Eliminar'
                }).then((result) => {
                    if (result.isConfirmed) {
                        Livewire.emitTo('admin.video.list-videos',
'delete', videoId)
                        Swal.fire(
                            '¡Hecho!',
                            'El vídeo ha sido eliminado.',
                            'success'
                        )
                    }
                })
            });
        </script>
    @endpush
</div>
```

Video-preview

Esta vista crea un reproductor de vídeo HTML5 que carga y reproduce un vídeo proporcionado dinámicamente a través de la variable `$route`. Los controles estándar del reproductor permiten a los usuarios reproducir, pausar y controlar el vídeo.

```
<div class="my-10">
    <video class="mx-auto w-4/5" controls>
        <source src="{{ $route }}" type="video/mp4">
        Your browser does not support the video tag.
    </video>
</div>
```

Video-ítem

List-video-items

Esta vista proporciona una interfaz para visualizar y explorar los metadatos de vídeos con capacidades de búsqueda, filtrado, ordenamiento y paginación.

1. **Filtrado y búsqueda**: Hay un campo de entrada de texto (`<x-jet-input>`) que permite al usuario ingresar un término de búsqueda. Además, hay un menú desplegable (`<select>`) que permite al usuario seleccionar el campo por el cual desea buscar.
2. **Eliminación de filtros**: El botón "Eliminar filtros" (`<x-jet-button>`) restablece los filtros de búsqueda.

3. **Ordenamiento**: Cada columna de la tabla es clickable (`<th>`) y está conectada a una función de ordenamiento en vivo que ordena los resultados basándose en el campo correspondiente.
4. **Visualización de resultados**: Los resultados se muestran en una tabla (`<x-table>`) donde se presentan los metadatos de cada vídeo, como ID, ID del vídeo, nombre del vídeo, calidad, formato, orientación y fecha de creación.
5. **Paginación**: Si hay múltiples resultados, se muestra una sección de paginación al final para navegar a través de las páginas de resultados.

```
<div>
   <h1 class="text-2xl font-semibold text-gray-700 mb-6">Metadatos de
vídeos</h1>

   <div class="mb-3">
       <div class="inline">
           <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                   font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
                   focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
               <option value="id">ID</option>
               <option value="video_id">ID VIDEO</option>
               <option value="quality">CALIDAD</option>
               <option value="format">FORMATO</option>
               <option value="orientation">ORIENTACIÓN</option>
               <option value="created_at">FECHA DE CREACIÓN</option>
           </select>
       </div>

       <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                    placeholder="Buscar ..."></x-jet-input>

       <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
   </div>
```

```
   @if(count($videoItems))
       <x-table>
           <x-slot name="thead">
               <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                   ID
                   @if($sortField === 'id' && $sortDirection === 'asc')
                       <i class="fa-solid fa-arrow-up">
                           @elseif($sortField === 'id' &&
$sortDirection === 'desc')
                               <i class="fa-solid fa-arrow-down"></i>
                   @endif
               </th>
               <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('video_id')">
                   ID del vídeo
                   @if($sortField === 'video_id' && $sortDirection ===
'asc')
                       <i class="fa-solid fa-arrow-up">
                           @elseif($sortField === 'video_id' &&
$sortDirection === 'desc')
                               <i class="fa-solid fa-arrow-down"></i>
                   @endif
               </th>
               <th scope="col" class="px-6 py-3">
                   Nombre del vídeo
               </th>
               <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('quality')">
                   Calidad
                   @if($sortField === 'quality' && $sortDirection ===
'asc')
                       <i class="fa-solid fa-arrow-up">
                           @elseif($sortField === 'quality' &&
$sortDirection === 'desc')
                               <i class="fa-solid fa-arrow-down"></i>
                   @endif
               </th>
```

```
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('format')">
                    Formato
                    @if($sortField === 'format' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'format' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('orientation')">
                    Orientación
                    @if($sortField === 'orientation' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'orientation' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($videoItems as $videoItem)
```

```
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->id }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->video_id }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->video->description }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->quality }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->format }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->orientation }}
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            {{ $videoItem->created_at }}
                        </td>
                    </tr>
                @endforeach
            </x-slot>
        </x-table>

        @if($videoItems->hasPages())
            <div class="mt-6">
```

```
                {{ $videoItems->links() }}
            </div>
        @endif
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif
</div>
```

Visit

Show-visits

Esta vista proporciona una interfaz para administrar y visualizar visitas registradas, con capacidades de búsqueda, filtrado, ordenamiento, paginación y visualización detallada a través de un modal.

1. **Título y búsqueda**: Muestra un título "Listado de visitas" y proporciona un campo de búsqueda similar al anterior, con la capacidad de filtrar por diferentes campos como ID, Device ID, SSOO (Sistema Operativo), Versión de SSOO, Punto de Interés y Fecha de Creación.
2. **Eliminación de filtros**: Al igual que en el código anterior, hay un botón "Eliminar filtros" que permite al usuario restablecer los filtros de búsqueda.
3. **Visualización de resultados**: Si hay visitas encontradas, se muestran en una tabla similar a la del código anterior, donde se muestran los detalles de cada visita, como ID, Device ID, SSOO, Versión de SSOO, Punto de Interés y Fecha de Creación.
4. **Paginación**: También se incluye la paginación si hay múltiples resultados.
5. **Modal de detalles**: Al hacer clic en el icono de ojo en una fila de la tabla, se abre un modal que muestra detalles adicionales de la visita, como la Hora, ID del Dispositivo, Versión de la Aplicación, Agente del Usuario, Sistema Operativo, Versión del Sistema Operativo y el Punto de Interés. Además, parece que se genera un código QR para representar el Punto de Interés.

```
<div>
    <div class="flex items-center mb-6">
        <h1 class="text-2xl font-semibold text-gray-700">Listado de
visitas</h1>
    </div>

    <div class="mb-3">
        <div class="inline">
            <select class="text-black  bg-blue-100 hover:bg-grey-200
focus:ring-4 focus:ring-blue-300
                    font-medium rounded-lg text-sm py-1.5
dark:bg-blue-600 dark:hover:bg-blue-700
                    focus:outline-none dark:focus:ring-blue-800 ml-auto"
wire:model="searchColumn">
                <option value="id">ID</option>
                <option value="deviceid">DEVICE ID</option>
                <option value="ssoo">SSOO</option>
                <option value="ssooversion">SSOO VERSION</option>
                <option value="point_of_interest_id">PUNTO DE
INTERÉS</option>
                <option value="created_at">FECHA DE CREACIÓN</option>
            </select>
        </div>

        <x-jet-input class="py-1 border-black" type="text"
wire:model="search"
                     placeholder="Buscar ..."></x-jet-input>

        <x-jet-button wire:click="resetFilters">Eliminar
filtros</x-jet-button>
    </div>

    @if(count($visits))
        <x-table>
            <x-slot name="thead">
```

```
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('id')">
                    ID
                    @if($sortField === 'id' && $sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'id' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('deviceid')">
                    Device ID
                    @if($sortField === 'deviceid' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'deviceid' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('ssoo')">
                    SSOO
                    @if($sortField === 'ssoo' && $sortDirection ===
'asc')
                        <i class="fa-solid fa-arrow-up">
                            @elseif($sortField === 'ssoo' &&
$sortDirection === 'desc')
                                <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('ssooversion')">
                    SSOO Version
                    @if($sortField === 'ssooversion' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
```

```
                        @elseif($sortField === 'ssooversion' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('point_of_interest_id')">
                    Punto de Interes
                    @if($sortField === 'point_of_interest_id' &&
$sortDirection === 'asc')
                        <i class="fa-solid fa-arrow-up">
                        @elseif($sortField ===
'point_of_interest_id' && $sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3 cursor-pointer"
wire:click="sort('created_at')">
                    Fecha creación
                    @if($sortField === 'created_at' && $sortDirection
=== 'asc')
                        <i class="fa-solid fa-arrow-up">
                        @elseif($sortField === 'created_at' &&
$sortDirection === 'desc')
                            <i class="fa-solid fa-arrow-down"></i>
                    @endif
                </th>
                <th scope="col" class="px-6 py-3">
                    <span class="sr-only">Actions</span>
                </th>
            </x-slot>

            <x-slot name="tbody">
                @foreach($visits as $visit)
                    <tr class="border-b dark:bg-gray-800
dark:border-gray-700 odd:bg-white even:bg-gray-50 odd:dark:bg-gray-800
even:dark:bg-gray-700">
```

```
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            <div>{{ $visit->id}}</div>
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            <div>{{ $visit->deviceid }}</div>
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            <div>{{ $visit->ssoo }}</div>
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            <div>{{ $visit->ssooversion }}</div>
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            @if( ! empty($visit->point_of_interest_id))
                                <div >{{ $visit->point_of_interest_id
}}</div>
                            @else
                                <span
class="text-red-600">Ninguno</span>
                            @endif
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap">
                            <div>{{ $visit->created_at }}</div>
                        </td>
                        <td class="px-6 py-4 font-medium text-gray-900
dark:text-white whitespace-nowrap flex gap-4">
                            <span class="font-medium text-blue-600
cursor-pointer" wire:click="show('{{ $visit->id }}')">
                                <i class="fa-solid fa-eye"></i>
                            </span>
                        </td>
                    </tr>
```

```
                @endforeach
            </x-slot>
        </x-table>

        @if($visits->hasPages())
            <div class="mt-6">
                {{ $visits->links() }}
            </div>
        @endif
    @else
        <p class="mt-4">No se han encontrado resultados</p>
    @endif

    {{-- Modal show --}}
    <x-jet-dialog-modal wire:model="detailsModal.open">
        <x-slot name="title">
            <span class="text-2xl">Detalles de la Visita #{{
$detailsModal['id'] }}</span>
        </x-slot>

        <x-slot name="content">
            <div class="space-y-3">
                <div>
                    <x-jet-label>
                        Hora: {{ $detailsModal['hour']}}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Id Dispositivo: {{ $detailsModal['deviceid'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Version de la App: {{
$detailsModal['appversion'] }}
                    </x-jet-label>
                </div>
```

```
                <div>
                    <x-jet-label>
                        Agente: {{ $detailsModal['useragent'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Sistema Operativo: {{ $detailsModal['ssoo'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Version Sistema Operativo: {{
$detailsModal['ssooversion'] }}
                    </x-jet-label>
                </div>
                <div>
                    <x-jet-label>
                        Punto de Interest:

{!!QrCode::size(100)->generate(json_encode($detailsModal['point_of_inte
rest_id'], JSON_PRETTY_PRINT)) !!}
                    </x-jet-label>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-button wire:click="$set('detailsModal.open', false)">
                Cerrar
            </x-button>
        </x-slot>
    </x-jet-dialog-modal>
</div>
```

Profile

Create-gtvisor-user

Este componente de interfaz de usuario permite al usuario cambiar su rol a "GTVisor" con una confirmación adicional a través de un modal.

1. **Slot "title"**: Muestra el título de la sección. En este caso, parece que el título se establece dinámicamente utilizando una función de traducción para obtener el texto "GTVisor user".
2. **Slot "description"**: Proporciona una descripción breve de lo que sucederá cuando el usuario haga clic en el botón para cambiar su rol. También parece ser traducido dinámicamente.
3. **Slot "content"**: Aquí se muestra el contenido principal de la sección. Primero, hay un texto explicativo de lo que sucederá cuando el usuario haga clic en el botón de cambio. Luego, hay un botón de acción (`<x-jet-button>`) que activa la función `confirmChangeUser`. Este botón está deshabilitado mientras se está cargando (`wire:loading.attr="disabled"`).
4. **Change User Confirmation Modal**: Este es un modal (una ventana emergente) que se abre cuando el usuario hace clic en el botón principal para cambiar su rol. Dentro del modal, hay un mensaje de confirmación preguntando si el usuario está seguro de querer cambiar su rol a "GTVisor". Hay dos botones en el pie de página del modal: uno para cancelar la acción y otro para confirmar el cambio de rol. Ambos botones también están configurados para deshabilitarse mientras se está cargando (`wire:loading.attr="disabled"`).

```
<x-jet-action-section>
    <x-slot name="title">
        {{ __('GTVisor user') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Change to a GTVisor user.') }}
    </x-slot>

    <x-slot name="content">
        <div class="max-w-xl text-sm text-gray-600">
            {{ __('Once you click this button, you will become a GTVisor
user and will have access to the map menu. To revert, go back to the
profile and you will have a button to return to your original role.')
}}
        </div>

        <div class="mt-5">
            <x-jet-button wire:click="confirmChangeUser"
wire:loading.attr="disabled">
                {{ __('Change') }}
            </x-jet-button>
        </div>

        <!-- Change User Confirmation Modal -->
        <x-jet-dialog-modal wire:model="confirmingUserDeletion">
            <x-slot name="title">
                {{ __('Change to GTVisor') }}
            </x-slot>

            <x-slot name="content">
                {{ __('Are you sure you want to change your role to
GTVisor?') }}
            </x-slot>

            <x-slot name="footer">
```

```
                <x-jet-secondary-button
wire:click="$toggle('confirmingUserDeletion')"
wire:loading.attr="disabled">
                    {{ __('Cancel') }}
                </x-jet-secondary-button>

                <x-jet-button class="ml-3" wire:click="changeUser"
wire:loading.attr="disabled">
                    {{ __('Change to GTVisor') }}
                </x-jet-button>
            </x-slot>
        </x-jet-dialog-modal>
    </x-slot>
</x-jet-action-section>
```

Delete-user-form

Este componente de interfaz de usuario permite al usuario eliminar su cuenta de forma permanente, con una confirmación adicional a través de un modal y la necesidad de ingresar su contraseña para confirmar la acción.

1. **Slot "title"**: Muestra el título de la sección, que en este caso es "Delete Account".
2. **Slot "description"**: Proporciona una descripción breve de lo que sucederá cuando el usuario elimine su cuenta, que es "Permanently delete your account".
3. **Slot "content"**: Contiene el contenido principal de la sección. Primero, hay un texto explicativo que advierte al usuario que una vez que elimine su cuenta, todos sus recursos y datos serán borrados permanentemente. Se recomienda al usuario descargar cualquier dato o información que desee conservar antes de eliminar la cuenta. Luego, hay un botón de acción rojo (`<x-jet-danger-button>`) que activa la función `confirmUserDeletion`. Este botón también está deshabilitado mientras se está cargando (`wire:loading.attr="disabled"`).

4. **Delete User Confirmation Modal**: Similar al primer componente, este es un modal que se abre cuando el usuario hace clic en el botón principal para eliminar su cuenta. El modal contiene un mensaje de confirmación preguntando si el usuario está seguro de querer eliminar su cuenta y una advertencia sobre la eliminación permanente de recursos y datos. También hay un campo de entrada de contraseña donde el usuario debe ingresar su contraseña para confirmar la eliminación de la cuenta. El modal tiene dos botones en el pie de página: uno para cancelar la acción y otro para confirmar la eliminación de la cuenta. Ambos botones también están configurados para deshabilitarse mientras se está cargando (`wire:loading.attr="disabled"`).

```
<x-jet-action-section>
    <x-slot name="title">
        {{ __('Delete Account') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Permanently delete your account.') }}
    </x-slot>

    <x-slot name="content">
        <div class="max-w-xl text-sm text-gray-600">
            {{ __('Once your account is deleted, all of its resources
and data will be permanently deleted. Before deleting your account,
please download any data or information that you wish to retain.') }}
        </div>

        <div class="mt-5">
            <x-jet-danger-button wire:click="confirmUserDeletion"
wire:loading.attr="disabled">
                {{ __('Delete Account') }}
            </x-jet-danger-button>
        </div>

        <!-- Delete User Confirmation Modal -->
        <x-jet-dialog-modal wire:model="confirmingUserDeletion">
            <x-slot name="title">
                {{ __('Delete Account') }}
```

```
            </x-slot>

            <x-slot name="content">
                {{ __('Are you sure you want to delete your account?
Once your account is deleted, all of its resources and data will be
permanently deleted. Please enter your password to confirm you would
like to permanently delete your account.') }}

                <div class="mt-4" x-data="{}"
x-on:confirming-delete-user.window="setTimeout(() =>
$refs.password.focus(), 250)">
                    <x-jet-input type="password" class="mt-1 block
w-3/4"
                                 placeholder="{{ __('Password') }}"
                                 x-ref="password"
                                 wire:model.defer="password"
                                 wire:keydown.enter="deleteUser" />

                    <x-jet-input-error for="password" class="mt-2" />
                </div>
            </x-slot>

            <x-slot name="footer">
                <x-jet-secondary-button
wire:click="$toggle('confirmingUserDeletion')"
wire:loading.attr="disabled">
                    {{ __('Cancel') }}
                </x-jet-secondary-button>

                <x-jet-danger-button class="ml-3"
wire:click="deleteUser" wire:loading.attr="disabled">
                    {{ __('Delete Account') }}
                </x-jet-danger-button>
            </x-slot>
        </x-jet-dialog-modal>
    </x-slot>
</x-jet-action-section>
```

Logout-other-browser-sessions-form

Este componente permite al usuario administrar y cerrar sesiones activas en otros dispositivos y navegadores, con una confirmación adicional a través de un modal y la necesidad de ingresar su contraseña para confirmar la acción.

1. **Slot "title"**: Muestra el título de la sección, que es "Browser Sessions".
2. **Slot "description"**: Proporciona una descripción breve de lo que hace esta sección, que es "Manage and log out your active sessions on other browsers and devices".
3. **Slot "content"**: Contiene el contenido principal de la sección. Comienza con un texto explicativo que informa al usuario que puede cerrar la sesión en todos los demás dispositivos y navegadores si es necesario. A continuación, se muestra una lista de sesiones activas en otros dispositivos y navegadores, si las hay. Cada sesión se muestra con un icono que representa el tipo de dispositivo (escritorio o móvil) y detalles como el sistema operativo y el navegador, la dirección IP y la hora de la última actividad. Además, hay un botón para cerrar la sesión en otros dispositivos y navegadores, que activa la función `confirmLogout`. También hay un mensaje de acción que se muestra cuando se completa el cierre de sesión en otros dispositivos.
4. **Log Out Other Devices Confirmation Modal**: Este es un modal que se abre cuando el usuario hace clic en el botón para cerrar la sesión en otros dispositivos. El modal solicita al usuario que ingrese su contraseña para confirmar la acción. El modal tiene dos botones en el pie de página: uno para cancelar la acción y otro para confirmar el cierre de sesión en otros dispositivos.

```
<x-jet-action-section>
    <x-slot name="title">
        {{ __('Browser Sessions') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Manage and log out your active sessions on other browsers and devices.') }}
    </x-slot>
```

```
    <x-slot name="content">
        <div class="max-w-xl text-sm text-gray-600">
            {{ __('If necessary, you may log out of all of your other
browser sessions across all of your devices. Some of your recent
sessions are listed below; however, this list may not be exhaustive. If
you feel your account has been compromised, you should also update your
password.') }}
        </div>

        @if (count($this->sessions) > 0)
            <div class="mt-5 space-y-6">
                <!-- Other Browser Sessions -->
                @foreach ($this->sessions as $session)
                    <div class="flex items-center">
                        <div>
                            @if ($session->agent->isDesktop())
                                <svg fill="none" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24"
stroke="currentColor" class="w-8 h-8 text-gray-500">
                                    <path d="M9.75 17L9 20l-1
1h8l-1-1-.75-3M3 13h18M5 17h14a2 2 0 002-2V5a2 2 0 00-2-2H5a2 2 0 00-2
2v10a2 2 0 002 2z"></path>
                                </svg>
                            @else
                                <svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" stroke-width="2" stroke="currentColor" fill="none"
stroke-linecap="round" stroke-linejoin="round" class="w-8 h-8
text-gray-500">
                                    <path d="M0 0h24v24H0z"
stroke="none"></path><rect x="7" y="4" width="10" height="16"
rx="1"></rect><path d="M11 5h2M12 17v.01"></path>
                                </svg>
                            @endif
                        </div>

                        <div class="ml-3">
                            <div class="text-sm text-gray-600">
```

```
                                {{ $session->agent->platform() ?
$session->agent->platform() : 'Unknown' }} - {{
$session->agent->browser() ? $session->agent->browser() : 'Unknown' }}
                            </div>

                            <div>
                                <div class="text-xs text-gray-500">
                                    {{ $session->ip_address }},

                                    @if ($session->is_current_device)
                                        <span class="text-green-500
font-semibold">{{ __('This device') }}</span>
                                    @else
                                        {{ __('Last active') }} {{
$session->last_active }}
                                    @endif
                                </div>
                            </div>
                        </div>
                    </div>
                @endforeach
            </div>
        @endif

        <div class="flex items-center mt-5">
            <x-jet-button wire:click="confirmLogout"
wire:loading.attr="disabled">
                {{ __('Log Out Other Browser Sessions') }}
            </x-jet-button>

            <x-jet-action-message class="ml-3" on="loggedOut">
                {{ __('Done.') }}
            </x-jet-action-message>
        </div>

        <!-- Log Out Other Devices Confirmation Modal -->
        <x-jet-dialog-modal wire:model="confirmingLogout">
            <x-slot name="title">
```

```
                {{ __('Log Out Other Browser Sessions') }}
            </x-slot>

            <x-slot name="content">
                {{ __('Please enter your password to confirm you would
like to log out of your other browser sessions across all of your
devices.') }}

                <div class="mt-4" x-data="{}"
x-on:confirming-logout-other-browser-sessions.window="setTimeout(() =>
$refs.password.focus(), 250)">
                    <x-jet-input type="password" class="mt-1 block
w-3/4"
                                 placeholder="{{ __('Password') }}"
                                 x-ref="password"
                                 wire:model.defer="password"

wire:keydown.enter="logoutOtherBrowserSessions" />

                    <x-jet-input-error for="password" class="mt-2" />
                </div>
            </x-slot>

            <x-slot name="footer">
                <x-jet-secondary-button
wire:click="$toggle('confirmingLogout')" wire:loading.attr="disabled">
                    {{ __('Cancel') }}
                </x-jet-secondary-button>

                <x-jet-button class="ml-3"
                              wire:click="logoutOtherBrowserSessions"
                              wire:loading.attr="disabled">
                    {{ __('Log Out Other Browser Sessions') }}
                </x-jet-button>
            </x-slot>
        </x-jet-dialog-modal>
    </x-slot>
</x-jet-action-section>
```

Revert-gtvisor-user

Este componente permite al usuario revertir su rol al rol anterior, con una confirmación adicional a través de un modal y la necesidad de confirmar la acción.

1. **Slot "title"**: Muestra el título de la sección, que es "Revert to User".
2. **Slot "description"**: Proporciona una descripción breve de lo que hace esta sección, que es "Revert your role to the previous user role".
3. **Slot "content"**: Contiene el contenido principal de la sección. Comienza con un texto explicativo que informa al usuario que puede volver a su rol anterior haciendo clic en un botón. A continuación, hay un botón para revertir el rol, que activa la función `confirmRevertUser`. Este botón también está deshabilitado mientras se está cargando (`wire:loading.attr="disabled"`).
4. **Revert Role Confirmation Modal**: Este es un modal que se abre cuando el usuario hace clic en el botón para revertir su rol. El modal solicita al usuario que confirme si está seguro de revertir su rol anterior. El modal tiene dos botones en el pie de página: uno para cancelar la acción y otro para confirmar la reversión del rol.

```
    <x-jet-action-section>
        <x-slot name="title">
            {{ __('Revert to User') }}
        </x-slot>

        <x-slot name="description">
            {{ __('Revert your role to the previous user role.') }}
        </x-slot>

        <x-slot name="content">
            <div class="max-w-xl text-sm text-gray-600">
                {{ __('If you want to go back to your previous role, click
this button.') }}
            </div>

            <div class="mt-5">
```

```
            <x-jet-button wire:click="confirmRevertUser"
wire:loading.attr="disabled">
                {{ __('Revert') }}
            </x-jet-button>
        </div>

        <x-jet-dialog-modal wire:model="confirmingUserReversion">
            <x-slot name="title">
                {{ __('Revert Role') }}
            </x-slot>

            <x-slot name="content">
                {{ __('Are you sure you want to revert to your previous
role?') }}
            </x-slot>

            <x-slot name="footer">
                <x-jet-secondary-button
wire:click="$toggle('confirmingUserReversion')"
wire:loading.attr="disabled">
                    {{ __('Cancel') }}
                </x-jet-secondary-button>

                <x-jet-button class="ml-3" wire:click="revertUser"
wire:loading.attr="disabled">
                    {{ __('Revert') }}
                </x-jet-button>
            </x-slot>
        </x-jet-dialog-modal>
    </x-slot>
</x-jet-action-section>
```

Show

Este código define la estructura y el contenido de una página de perfil de usuario en una aplicación web, incluidos formularios para actualizar información de perfil, contraseñas, autenticación de dos factores, cerrar sesiones en otros dispositivos, eliminar la cuenta y realizar cambios en los roles del usuario.

1. **x-app-layout**: Este componente probablemente define el diseño general de la aplicación. Dentro de este componente, se definen varias secciones de contenido.
2. **x-slot "header"**: Esta sección define el encabezado de la página, que parece ser el título "Profile".
3. **Contenido principal**: Este contenido principal está contenido dentro de un `<div>` y está destinado a mostrar información y formularios relacionados con el perfil del usuario.
4. **Update Profile Information Form**: Si la característica de actualización de información de perfil está habilitada, se muestra un formulario para actualizar la información del perfil.
5. **Update Password Form**: Si la característica de actualización de contraseñas está habilitada, se muestra un formulario para actualizar la contraseña del usuario.
6. **Two Factor Authentication Form**: Si el usuario puede gestionar la autenticación de dos factores, se muestra un formulario para gestionar esta configuración.
7. **Logout Other Browser Sessions Form**: Se muestra un formulario para cerrar la sesión en otras sesiones de navegador activas.
8. **Delete User Form**: Si la aplicación tiene características de eliminación de cuenta, se muestra un formulario para eliminar la cuenta del usuario.
9. **Revert GTVisor User Form**: Si el usuario actual tiene el rol de "GTVisor" y su rol anterior no es nulo, se muestra un formulario para revertir al rol anterior. Si no tiene el rol "GTVisor", se muestra un formulario para crear un usuario con el rol "GTVisor".
10. **Teacher Request Form**: Si el usuario tiene el rol de "Alumno", se muestra un formulario para enviar una solicitud de profesor.
11. **x-jet-section-border**: Este componente parece ser un borde de sección visual que separa visualmente cada sección de contenido.

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            {{ __('Profile') }}
        </h2>
    </x-slot>

    <div>
        <div class="max-w-7xl mx-auto py-10 sm:px-6 lg:px-8">
```

```
            @if
(Laravel\Fortify\Features::canUpdateProfileInformation())
                @livewire('profile.update-profile-information-form')

                <x-jet-section-border />
            @endif

            @if
(Laravel\Fortify\Features::enabled(Laravel\Fortify\Features::updatePass
words()))
                <div class="mt-10 sm:mt-0">
                    @livewire('profile.update-password-form')
                </div>

                <x-jet-section-border />
            @endif

            @if
(Laravel\Fortify\Features::canManageTwoFactorAuthentication())
                <div class="mt-10 sm:mt-0">
                    @livewire('profile.two-factor-authentication-form')
                </div>

                <x-jet-section-border />
            @endif

            <div class="mt-10 sm:mt-0">
                @livewire('profile.logout-other-browser-sessions-form')
            </div>

            @if
(Laravel\Jetstream\Jetstream::hasAccountDeletionFeatures())
                <x-jet-section-border />

                <div class="mt-10 sm:mt-0">
                    @livewire('profile.delete-user-form')
                </div>
            @endif
```

```
            <x-jet-section-border />

            <div class="mt-10 sm:mt-0">
                @if(auth()->user()->hasRole('GTVisor') &&
(auth()->user()->toArray()['previous_role'] == null))
                @elseif(auth()->user()->hasRole('GTVisor'))
                    @livewire('profile.revert-gtvisor-user')
                @else
                    @livewire('profile.create-gtvisor-user')
                @endif
            </div>
                <x-jet-section-border />

                @role('Alumno')

                <div class="mt-10 sm:mt-0">
                    @livewire('profile.teacher-request')
                </div>

                @endrole


        </div>
    </div>
</x-app-layout>
```

Teacher-request

Este componente permite al usuario enviar una solicitud a los administradores para cambiar su rol a profesor, con una confirmación adicional a través de un modal que requiere la confirmación del usuario antes de enviar el correo electrónico.

1. **Slot "title"**: Muestra el título de la sección, que es "Fetch Teacher".
2. **Slot "description"**: Proporciona una descripción breve de lo que hace esta sección, que es "Application for the Position of Teacher".
3. **Slot "content"**: Contiene el contenido principal de la sección. Comienza con un texto explicativo que informa al usuario que al hacer clic en el botón, enviará una solicitud a los administradores para obtener acceso como profesor. A continuación, hay un botón que activa la función `confirmSendEmail`. Este botón también está deshabilitado mientras se está cargando (`wire:loading.attr="disabled"`).
4. **Send Email Confirmation Modal**: Este es un modal que se abre cuando el usuario hace clic en el botón para enviar la solicitud. El modal solicita al usuario que confirme si está seguro de querer enviar un correo electrónico para cambiar su rol a profesor. El modal tiene dos botones en el pie de página: uno para cancelar la acción y otro para confirmar el envío del correo electrónico.
5. **Slot "title"**: Título del modal, que es "Petition to admin email".
6. **Slot "content"**: Contenido del modal, que pregunta si el usuario está seguro de querer enviar un correo electrónico para cambiar su rol a profesor.
7. **Slot "footer"**: Contiene dos botones:
8. Un botón secundario para cancelar la acción, que cierra el modal al hacer clic.
9. Un botón principal que confirma la acción de enviar el correo electrónico, que llama a la función `sendEmail`.

```
<x-jet-action-section>
    <x-slot name="title">
        {{ __('Fetch Teacher') }}
    </x-slot>

    <x-slot name="description">
        {{ __(' Application for the Position of Teacher') }}
    </x-slot>

    <x-slot name="content">
        <div class="max-w-xl text-sm text-gray-600">
            {{ __('By clicking, you send a request to our
administrators, who will grant you access as a teacher.') }}
        </div>

        <div class="mt-5">
```

```
            <x-jet-button wire:click="confirmSendEmail"
wire:loading.attr="disabled">
                {{ __('Petition') }}
            </x-jet-button>
        </div>

        <!-- Change User Confirmation Modal -->
        <!-- Change User Confirmation Modal -->
        <x-jet-dialog-modal wire:model="SendEmailToAdmin">
            <x-slot name="title">
                {{ __('Petition to admin email') }}
            </x-slot>

            <x-slot name="content">
                {{ __('Are you sure you want to send a email to change
your role to Teacher?') }}
            </x-slot>

            <x-slot name="footer">
                <x-jet-secondary-button
wire:click="$toggle('SendEmailToAdmin')" wire:loading.attr="disabled">
                    {{ __('Cancel') }}
                </x-jet-secondary-button>

                <x-jet-button class="ml-3" wire:click="sendEmail"
wire:loading.attr="disabled">
                    {{ __('Send email') }}
                </x-jet-button>
            </x-slot>
        </x-jet-dialog-modal>
    </x-slot>
</x-jet-action-section>
```

Two-factor-authentication-form

Este componente permite al usuario habilitar, confirmar, ver códigos de recuperación, regenerar códigos de recuperación y deshabilitar la autenticación de dos factores en su cuenta, proporcionando una capa adicional de seguridad.

1. **Slot "title"**: Muestra el título de la sección, que es "Two Factor Authentication".
2. **Slot "description"**: Proporciona una descripción breve de lo que hace esta sección, que es "Add additional security to your account using two factor authentication".
3. **Slot "content"**: Contiene el contenido principal de la sección. Aquí está el desglose detallado:
4. **Estado de la Autenticación de Dos Factores**: Si 2FA está habilitado, muestra "You have enabled two factor authentication". Si se está mostrando la confirmación, muestra "Finish enabling two factor authentication". Si 2FA no está habilitado, muestra "You have not enabled two factor authentication".
5. **Descripción de 2FA**: Explica cómo funciona 2FA y cómo se utiliza con la aplicación Google Authenticator en el teléfono del usuario.
6. **Habilitación de 2FA**: Si 2FA está habilitado, muestra información adicional según el estado de la configuración:
7. **Código QR**: Si se está mostrando el código QR, proporciona instrucciones para escanear el código QR o ingresar la clave de configuración en la aplicación de autenticación.
8. **Código de Configuración**: Muestra la clave de configuración que se debe ingresar en la aplicación de autenticación.
9. **Confirmación del Código**: Si se está mostrando la confirmación, proporciona un campo para ingresar el código OTP generado por la aplicación de autenticación.
10. **Códigos de Recuperación**: Si se están mostrando los códigos de recuperación, proporciona instrucciones y muestra los códigos de recuperación que se deben almacenar de manera segura.
11. **Habilitar 2FA**: Si 2FA no está habilitado, muestra un botón para habilitar 2FA.
12. **Regenerar Códigos de Recuperación**: Si se están mostrando los códigos de recuperación, muestra un botón para regenerar los códigos.
13. **Mostrar Códigos de Recuperación**: Si 2FA está habilitado pero no se están mostrando los códigos de recuperación, muestra un botón para mostrar los códigos.
14. **Confirmar 2FA**: Si se está mostrando la confirmación, muestra un botón para confirmar la habilitación de 2FA.
15. **Deshabilitar 2FA**: Muestra un botón para deshabilitar 2FA.

```
<x-jet-action-section>
    <x-slot name="title">
        {{ __('Two Factor Authentication') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Add additional security to your account using two factor
authentication.') }}
    </x-slot>

    <x-slot name="content">
        <h3 class="text-lg font-medium text-gray-900">
            @if ($this->enabled)
                @if ($showingConfirmation)
                    {{ __('Finish enabling two factor authentication.')
}}
                @else
                    {{ __('You have enabled two factor authentication.')
}}
                @endif
            @else
                {{ __('You have not enabled two factor authentication.')
}}
            @endif
        </h3>

        <div class="mt-3 max-w-xl text-sm text-gray-600">
            <p>
                {{ __('When two factor authentication is enabled, you
will be prompted for a secure, random token during authentication. You
may retrieve this token from your phone\'s Google Authenticator
application.') }}
            </p>
        </div>

        @if ($this->enabled)
```

```
            @if ($showingQrCode)
                <div class="mt-4 max-w-xl text-sm text-gray-600">
                    <p class="font-semibold">
                        @if ($showingConfirmation)
                            {{ __('To finish enabling two factor
authentication, scan the following QR code using your phone\'s
authenticator application or enter the setup key and provide the
generated OTP code.') }}
                        @else
                            {{ __('Two factor authentication is now
enabled. Scan the following QR code using your phone\'s authenticator
application or enter the setup key.') }}
                        @endif
                    </p>
                </div>

                <div class="mt-4">
                    {!! $this->user->twoFactorQrCodeSvg() !!}
                </div>

                <div class="mt-4 max-w-xl text-sm text-gray-600">
                    <p class="font-semibold">
                        {{ __('Setup Key') }}: {{
decrypt($this->user->two_factor_secret) }}
                    </p>
                </div>

                @if ($showingConfirmation)
                    <div class="mt-4">
                        <x-jet-label for="code" value="{{ __('Code') }}"
/>

                        <x-jet-input id="code" type="text" name="code"
class="block mt-1 w-1/2" inputmode="numeric" autofocus
autocomplete="one-time-code"
                            wire:model.defer="code"

wire:keydown.enter="confirmTwoFactorAuthentication" />
```

```
                    <x-jet-input-error for="code" class="mt-2" />
                </div>
            @endif
        @endif

        @if ($showingRecoveryCodes)
            <div class="mt-4 max-w-xl text-sm text-gray-600">
                <p class="font-semibold">
                    {{ __('Store these recovery codes in a secure
password manager. They can be used to recover access to your account if
your two factor authentication device is lost.') }}
                </p>
            </div>

            <div class="grid gap-1 max-w-xl mt-4 px-4 py-4 font-mono
text-sm bg-gray-100 rounded-lg">
                @foreach
(json_decode(decrypt($this->user->two_factor_recovery_codes), true) as
$code)
                    <div>{{ $code }}</div>
                @endforeach
            </div>
        @endif
    @endif

    <div class="mt-5">
        @if (! $this->enabled)
            <x-jet-confirms-password
wire:then="enableTwoFactorAuthentication">
                <x-jet-button type="button"
wire:loading.attr="disabled">
                    {{ __('Enable') }}
                </x-jet-button>
            </x-jet-confirms-password>
        @else
            @if ($showingRecoveryCodes)
```

```
                    <x-jet-confirms-password
wire:then="regenerateRecoveryCodes">
                        <x-jet-secondary-button class="mr-3">
                            {{ __('Regenerate Recovery Codes') }}
                        </x-jet-secondary-button>
                    </x-jet-confirms-password>
                @elseif ($showingConfirmation)
                    <x-jet-confirms-password
wire:then="confirmTwoFactorAuthentication">
                        <x-jet-button type="button" class="mr-3"
wire:loading.attr="disabled">
                            {{ __('Confirm') }}
                        </x-jet-button>
                    </x-jet-confirms-password>
                @else
                    <x-jet-confirms-password
wire:then="showRecoveryCodes">
                        <x-jet-secondary-button class="mr-3">
                            {{ __('Show Recovery Codes') }}
                        </x-jet-secondary-button>
                    </x-jet-confirms-password>
                @endif

                @if ($showingConfirmation)
                    <x-jet-confirms-password
wire:then="disableTwoFactorAuthentication">
                        <x-jet-secondary-button
wire:loading.attr="disabled">
                            {{ __('Cancel') }}
                        </x-jet-secondary-button>
                    </x-jet-confirms-password>
                @else
                    <x-jet-confirms-password
wire:then="disableTwoFactorAuthentication">
                        <x-jet-danger-button
wire:loading.attr="disabled">
                            {{ __('Disable') }}
                        </x-jet-danger-button>
```

```
                    </x-jet-confirms-password>
                @endif

            @endif
        </div>
    </x-slot>
</x-jet-action-section>
```

Update-password-form

Este formulario permite a los usuarios actualizar su contraseña actual a una nueva, asegurando que ingresen la nueva contraseña dos veces para confirmarla y validando que la contraseña actual sea correcta.

1. **Título**: El título del formulario se define como "Update Password" (Actualizar contraseña).
2. **Descripción**: La descripción del formulario sugiere usar una contraseña larga y aleatoria para mantener la cuenta segura.
3. **Campo para la Contraseña Actual**: Incluye una etiqueta y un campo de entrada para la contraseña actual (`current_password`).
4. **Campo para la Nueva Contraseña**: Incluye una etiqueta y un campo de entrada para la nueva contraseña (`password`).
5. **Campo para Confirmar la Nueva Contraseña**: Incluye una etiqueta y un campo de entrada para confirmar la nueva contraseña (`password_confirmation`). Cada campo tiene validación de errores correspondiente que se muestra debajo del campo si hay algún problema.
6. **Mensaje de Acción**: Muestra un mensaje de confirmación "Saved." cuando la contraseña se ha guardado correctamente.
7. **Botón de Guardar**: Botón para enviar el formulario y guardar los cambios.

```
<x-jet-form-section submit="updatePassword">
```

```
    <x-slot name="title">
        {{ __('Update Password') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Ensure your account is using a long, random password to
stay secure.') }}
    </x-slot>

    <x-slot name="form">
        <div class="col-span-6 sm:col-span-4">
            <x-jet-label for="current_password" value="{{ __('Current
Password') }}" />
            <x-jet-input id="current_password" type="password"
class="mt-1 block w-full" wire:model.defer="state.current_password"
autocomplete="current-password" />
            <x-jet-input-error for="current_password" class="mt-2" />
        </div>

        <div class="col-span-6 sm:col-span-4">
            <x-jet-label for="password" value="{{ __('New Password') }}"
/>
            <x-jet-input id="password" type="password" class="mt-1 block
w-full" wire:model.defer="state.password" autocomplete="new-password"
/>
            <x-jet-input-error for="password" class="mt-2" />
        </div>

        <div class="col-span-6 sm:col-span-4">
            <x-jet-label for="password_confirmation" value="{{
__('Confirm Password') }}" />
            <x-jet-input id="password_confirmation" type="password"
class="mt-1 block w-full"
wire:model.defer="state.password_confirmation"
autocomplete="new-password" />
            <x-jet-input-error for="password_confirmation" class="mt-2"
/>
        </div>
```

```
    </x-slot>

    <x-slot name="actions">
        <x-jet-action-message class="mr-3" on="saved">
            {{ __('Saved.') }}
        </x-jet-action-message>

        <x-jet-button>
            {{ __('Save') }}
        </x-jet-button>
    </x-slot>
</x-jet-form-section>
```

Update-profile-information-form

Este formulario permite a los usuarios actualizar su nombre, dirección de correo electrónico y foto de perfil, manejando las verificaciones necesarias para el correo electrónico y proporcionando vistas previas y eliminación de la foto de perfil.

1. **Título**: El título del formulario es "Profile Information" (Información del Perfil).
2. **Descripción**: La descripción del formulario indica que se puede actualizar la información del perfil y la dirección de correo electrónico de la cuenta.
3. **Foto de Perfil**: Si la aplicación gestiona fotos de perfil, se incluye un campo para cargar una nueva foto. Un input oculto para seleccionar un archivo de foto. Al cambiar el archivo, se actualizan las variables `photoName` y `photoPreview` para mostrar una vista previa de la nueva foto. Se muestra la foto de perfil actual si no hay una nueva foto seleccionada. Se muestra una vista previa de la nueva foto si se ha seleccionado una. Un botón para seleccionar una nueva foto. Un botón para eliminar la foto de perfil actual si existe. Manejo de errores de entrada para la foto.
4. **Nombre**: Un campo de entrada para el nombre del usuario (`name`), con su correspondiente etiqueta y manejo de errores.
5. **Correo Electrónico**: Un campo de entrada para la dirección de correo electrónico del usuario (`email`), con su correspondiente etiqueta y manejo de errores. Si el

correo electrónico no está verificado, se muestra un mensaje indicando esto y un botón para reenviar el correo de verificación. Un mensaje de confirmación que se muestra cuando se ha enviado un nuevo enlace de verificación.

6. **Mensaje de Acción**: Muestra un mensaje de confirmación "Saved." cuando la información del perfil se ha guardado correctamente.
7. **Botón de Guardar**: Botón para enviar el formulario y guardar los cambios. Se desactiva mientras se está cargando una nueva foto.

```
<x-jet-form-section submit="updateProfileInformation">
    <x-slot name="title">
        {{ __('Profile Information') }}
    </x-slot>

    <x-slot name="description">
        {{ __('Update your account\'s profile information and email
address.') }}
    </x-slot>

    <x-slot name="form">
        <!-- Profile Photo -->
        @if (Laravel\Jetstream\Jetstream::managesProfilePhotos())
            <div x-data="{photoName: null, photoPreview: null}"
class="col-span-6 sm:col-span-4">
                <!-- Profile Photo File Input -->
                <input type="file" class="hidden"
                            wire:model="photo"
                            x-ref="photo"
                            x-on:change="
                                    photoName =
$refs.photo.files[0].name;

                                    const reader = new FileReader();
                                    reader.onload = (e) => {
                                        photoPreview = e.target.result;
                                    };

reader.readAsDataURL($refs.photo.files[0]);
                            " />
```

```
                <x-jet-label for="photo" value="{{ __('Photo') }}" />

                <!-- Current Profile Photo -->
                <div class="mt-2" x-show="! photoPreview">
                    <img src="{{ $this->user->profile_photo_url }}"
alt="{{ $this->user->name }}" class="rounded-full h-20 w-20
object-cover">
                </div>

                <!-- New Profile Photo Preview -->
                <div class="mt-2" x-show="photoPreview" style="display:
none;">
                    <span class="block rounded-full w-20 h-20 bg-cover
bg-no-repeat bg-center"
                          x-bind:style="'background-image: url(\'' +
photoPreview + '\');'">
                    </span>
                </div>

                <x-jet-secondary-button class="mt-2 mr-2" type="button"
x-on:click.prevent="$refs.photo.click()">
                    {{ __('Select A New Photo') }}
                </x-jet-secondary-button>

                @if ($this->user->profile_photo_path)
                    <x-jet-secondary-button type="button" class="mt-2"
wire:click="deleteProfilePhoto">
                        {{ __('Remove Photo') }}
                    </x-jet-secondary-button>
                @endif

                <x-jet-input-error for="photo" class="mt-2" />
            </div>
        @endif

        <!-- Name -->
        <div class="col-span-6 sm:col-span-4">
```

```
            <x-jet-label for="name" value="{{ __('Name') }}" />
            <x-jet-input id="name" type="text" class="mt-1 block w-full"
wire:model.defer="state.name" autocomplete="name" />
            <x-jet-input-error for="name" class="mt-2" />
        </div>

        <!-- Email -->
        <div class="col-span-6 sm:col-span-4">
            <x-jet-label for="email" value="{{ __('Email') }}" />
            <x-jet-input id="email" type="email" class="mt-1 block
w-full" wire:model.defer="state.email" />
            <x-jet-input-error for="email" class="mt-2" />

            @unless ($this->user->hasVerifiedEmail())
                <p class="text-sm mt-2">
                    {{ __('Your email address is unverified.') }}

                    <button type="button" class="underline text-sm
text-gray-600 hover:text-gray-900"
wire:click.prevent="sendEmailVerification">
                        {{ __('Click here to re-send the verification
email.') }}
                    </button>
                </p>

                @if ($this->verificationLinkSent)
                    <p v-show="verificationLinkSent" class="mt-2
font-medium text-sm text-green-600">
                        {{ __('A new verification link has been sent to
your email address.') }}
                    </p>
                @endif
            @endif
        </div>
    </x-slot>

    <x-slot name="actions">
        <x-jet-action-message class="mr-3" on="saved">
```

```
            {{ __('Saved.') }}
        </x-jet-action-message>

        <x-jet-button wire:loading.attr="disabled" wire:target="photo">
            {{ __('Save') }}
        </x-jet-button>
    </x-slot>
</x-jet-form-section>
```

Controladores

Photography

El controlador `Photographies` gestiona las fotografías en la aplicación, proporcionando funcionalidades para crear, editar, eliminar y visualizarlas. Destaca por:

- Paginación y carga de archivos.
- Atributos públicos para gestionar el estado de la página.
- Reglas de validación para los campos del formulario.
- Métodos de inicialización y obtención de datos.
- Métodos de creación y actualización de fotografías.
- Métodos de visualización y edición de detalles.
- Método para eliminar fotografías.
- Ordenamiento y filtros personalizables.
- Método de renderizado para gestionar la paginación de resultados.

```
class Photographies extends Component
```

```
{
    use WithPagination;
    use WithFileUploads;

    public $pointsOfInterest;
    public $thematicAreas;

    public $listeners = ['delete'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';

    protected $queryString = ['search'];

    public $createForm = [
        'open' => false,
        'route' => null,
        'order' => '',
        'pointOfInterestId' => '',
        'thematicAreaId' => '',
    ];

    public $editForm = [
        'route' => null,
        'order' => '',
        'pointOfInterestId' => '',
        'thematicAreaId' => '',
    ];

    protected $rules = [
        'createForm.route' => 'image|max:5120',
        'createForm.pointOfInterestId' => 'required|integer',
        'createForm.thematicAreaId' => 'required|integer',
    ];

    protected $validationAttributes = [
        'createForm.route' => 'fotografía',
        'createForm.pointOfInterestId' => 'punto de interés',
```

```
        'createForm.thematicAreaId' => 'área temática',

        'editForm.route' => 'fotografía',
        'editForm.pointOfInterestId' => 'punto de interés',
        'editForm.thematicAreaId' => 'área temática',
    ];

    public $showModal = [
        'open' => false,
        'id' => null,
        'route' => null,
        'order' => null,
        'pointOfInterestId' => null,
        'thematicAreaId' => null,
        'thematicAreaName' => null,
        'creatorId' => null,
        'creatorName' => null,
        'updaterId' => null,
        'updaterName' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public $editModal = [
        'open' => false,
        'id' => null,
        'route' => null,
        'order' => null,
        'pointOfInterestId' => null,
        'thematicAreaId' => null,
        'thematicAreaName' => null,
        'creatorId' => null,
        'creatorName' => null,
        'updaterId' => null,
        'updaterName' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public function mount()
    {
```

```
        $this->getPointsOfInterest();
        $this->getThematicAreas();
    }

    public function getPointsOfInterest()
    {
        $this->pointsOfInterest = PointOfInterest::all();
    }

    public function getThematicAreas()
    {
        $this->thematicAreas = ThematicArea::all();
    }

    public function updatedCreateFormPointOfInterestId()
    {
        $this->createForm['thematicAreaId'] = '';
        $this->thematicAreas =
PointOfInterest::find($this->createForm['pointOfInterestId'])->the
maticAreas;
    }

    public function updatedEditFormPointOfInterestId()
    {
        $this->editForm['thematicAreaId'] = '';
        $this->thematicAreas =
PointOfInterest::find($this->editForm['pointOfInterestId'])->thema
ticAreas;
    }

    public function save()
    {
        $this->validate();

        $this->createForm['route']->storeAs('public/photos',
$this->createForm['route']->getFilename());

        $order = Photography::where('point_of_interest_id',
$this->createForm['pointOfInterestId'])->count();

        $photography = Photography::create([
```

```
            'route' => 'storage/photos/' .
$this->createForm['route']->getFilename(),
            'order' =>  $order + 1,
            'point_of_interest_id' =>
$this->createForm['pointOfInterestId'],
            'thematic_area_id' =>
$this->createForm['thematicAreaId'],
            'creator' => auth()->user()->id,
            'updater' => null,
            'updated_at' => null,
        ]);

        ProcessPhotography::dispatch($photography);

        $this->reset('createForm');

        $this->emit('photographyCreated');
    }

    public function update(Photography $photography)
    {
        $this->validate([
            'editForm.route' => 'max:5120',
            'editForm.pointOfInterestId' => 'required|integer',
            'editForm.thematicAreaId' => 'required|integer',
        ]);

        if (! is_null($this->editForm['route'])) {
            $this->editForm['route']->storeAs('public/photos',
$this->editForm['route']->getFilename());

            $photography['route'] = 'storage/photos/' .
$this->editForm['route']->getFilename();
        }

        $order = Photography::where('point_of_interest_id',
$this->editForm['pointOfInterestId'])->count();

        $photography['order'] = $order +1;
        $photography['point_of_interest_id'] =
$this->editForm['pointOfInterestId'];
```

```
        $photography['thematic_area_id'] =
$this->editForm['thematicAreaId'];
        $photography['updater'] = auth()->user()->id;

        $photography->update();

        Log::info('Photography with ID ' . $photography->id . ' was
updated ' . $photography);

        $this->reset(['editForm']);
        $this->reset(['editModal']);

        $this->emit('photographyUpdated');
    }

    public function show(Photography $photography)
    {
        if ( ! is_null( User::find($photography->updater) )) {
            $this->showModal['updaterId'] =
User::find($photography->updater)->id;
            $this->showModal['updaterName'] =
User::find($photography->updater)->name;
        } else {
            $this->showModal['updaterId'] = null;
            $this->showModal['updaterName'] = null;
        }

        $this->showModal['id'] = $photography->id;

        $this->showModal['route'] = $photography->route;
        $this->showModal['order'] = $photography->order;
        $this->showModal['pointOfInterestId'] =
$photography['point_of_interest_id'];
        $this->showModal['thematicAreaId'] =
$photography->thematicArea->id ?? '';
        $this->showModal['thematicAreaName'] =
$photography->thematicArea->name ?? '';

        $this->showModal['creatorId'] =
User::find($photography->creator)->id;
```

```
        $this->showModal['creatorName'] =
User::find($photography->creator)->name;

        $this->showModal['createdAt'] = $photography->created_at;
        $this->showModal['updatedAt'] = $photography->updated_at;

        $this->showModal['open'] = true;
    }

    public function edit(Photography $photography)
    {
        $this->reset(['editForm']);

        if ( ! is_null($photography['point_of_interest_id'])) {
            $this->thematicAreas =
PointOfInterest::find($photography['point_of_interest_id'])->thema
ticAreas;
        } else {
            $this->thematicAreas = null;
        }

        $this->editForm['pointOfInterestId'] =
$photography['point_of_interest_id'] ?? '';
        $this->editForm['thematicAreaId'] =
$photography->thematicArea->id ?? '';

        $this->editModal['id'] = $photography->id;
        $this->editModal['route'] = $photography->route;

        $this->editModal['open'] = true;
    }

    public function delete(Photography $photography)
    {
        $photography->delete();

        Log::info('Photography with ID ' . $photography->id . ' was
deleted ' . $photography);
    }

    public function sort($field)
```

```
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }

    public function render()
    {
        $this->reset(['page']);

        if (auth()->user()->hasRole('Alumno')) {
            $photographies = Photography::where('creator',
auth()->user()->id)
                ->where($this->searchColumn, 'like', '%'.
$this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        } else {
            $photographies =
Photography::where($this->searchColumn, 'like', '%'. $this->search
.'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        }
```

```
        return view('livewire.admin.photography.photographies',
compact('photographies'));
    }
}
```

Places

CreatePlace

El controlador CreatePlace se encarga de la creación de nuevos lugares en la aplicación. Sus puntos principales son:

Define atributos y reglas de validación para el formulario de creación.
Implementa un método para abrir el modal de creación.
Incluye un método save() para validar y guardar los datos del nuevo lugar.
Dispara un evento después de la creación del lugar.
Renderiza la vista del formulario de creación de lugares.

```
class CreatePlace extends Component
{
    protected $listeners = ['openCreationModal'];

    public $createForm = [
        'open' => false,
        'name' => '',
        'description' => '',
    ];

    protected $rules = [
        'createForm.name' => 'required',
        'createForm.description' => 'required|string|max:2000',
    ];
```

```
    protected $validationAttributes = [
        'createForm.name' => 'nombre',
        'createForm.description' => 'descripción',
    ];

    public function openCreationModal()
    {
        $this->createForm['open'] = true;
    }

    public function save()
    {
        $this->validate();

        $place = Place::create([
            'name' => $this->createForm['name'],
            'description' => $this->createForm['description'],
            'creator' => auth()->user()->id,
            'updater' => auth()->user()->id,
        ]);

        ProcessPlace::dispatch($place);

        $this->reset('createForm');
        $this->emit('placeCreated');
        $this->emitTo('admin.places.list-places', 'render');
    }


    public function render()
    {
        return view('livewire.admin.places.create-place');
    }
}
```

EditPlace

El controlador `EditPlace` administra la edición de lugares en la aplicación. Sus puntos principales incluyen:

- Utiliza atributos para almacenar el ID del lugar y los datos del formulario de edición.
- Define reglas de validación para los campos del formulario.
- Implementa un método para abrir el modal de edición y cargar los datos del lugar seleccionado.
- Incluye un método `update()` para validar y actualizar los datos del lugar.
- Dispara eventos después de la actualización del lugar.
- Renderiza la vista del formulario de edición de lugares.

```
class EditPlace extends Component
{
    public $placeId;

    protected $listeners = ['openEditModal'];

    public $editForm = [
        'open' => false,
        'name' => '',
        'description' => '',
    ];

    protected $rules = [
        'editForm.name' => 'required',
        'editForm.description' => 'required|string|max:2000',
    ];

    protected $validationAttributes = [
        'editForm.name' => 'nombre',
```

```
        'editForm.description' => 'descripción',
    ];

    public function openEditModal(Place $place)
    {
        $this->reset(['editForm']);

        $this->placeId = $place->id;
        $this->editForm['name'] = $place->name;
        $this->editForm['description'] = $place->description;

        $this->editForm['open'] = true;
    }

    public function update(Place $place)
    {
        $this->validate();

        $place->update([
            'name' => $this->editForm['name'],
            'description' => $this->editForm['description'],
            'updater' => auth()->user()->id,
        ]);

        Log::info('Place with ID ' . $place->id . ' was updated ' .
$place);

        $this->editForm['open'] = false;
        $this->reset(['editForm']);

        $this->emitTo('admin.places.list-places', 'render');
        $this->emit('placeUpdated');
    }

    public function render()
    {
        return view('livewire.admin.places.edit-place');
    }
}
```

ListPlaces

El controlador `ListPlaces` se encarga de mostrar una lista paginada de lugares en la aplicación. Sus puntos principales son:

- Utiliza el trait `WithPagination` para habilitar la paginación de resultados.
- Define atributos públicos para gestionar el estado de la búsqueda y el ordenamiento de la lista.
- Implementa métodos para mostrar detalles, eliminar lugares y gestionar el ordenamiento.
- Renderiza la vista de la lista de lugares según los criterios definidos por el usuario.

```
class ListPlaces extends Component
{
    use WithPagination;

    public $listeners = ['delete', 'render'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';

    protected $queryString = ['search'];

    public $detailsModal = [
        'open' => false,
        'id' => null,
        'name' => null,
        'description' => null,
        'creatorName' => null,
        'creatorId' => null,
        'updaterName' => null,
        'updaterId' => null,
```

```
        'deletedAt' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public function show(Place $place)
    {
        $this->detailsModal['open'] = true;
        $this->detailsModal['id'] = $place->id;
        $this->detailsModal['name'] = $place->name;
        $this->detailsModal['description'] = $place->description;
        $this->detailsModal['creatorName'] =
User::find($place->creator)->name;
        $this->detailsModal['creatorId'] = $place->creator;
        $this->detailsModal['updaterName'] = $place->updater ?
User::find($place->updater)->name : null;
        $this->detailsModal['updaterId'] = $place->updater;
        $this->detailsModal['createdAt'] = $place->created_at;
        $this->detailsModal['updatedAt'] = $place->updated_at;
    }

    public function delete(Place $place)
    {
        $place->delete();

        Log::info('Place with ID ' . $place->id . ' was deleted ' .
$place);
    }

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }
```

```
    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }

    public function render()
    {
        if (auth()->user()->hasRole('Alumno')) {
            $places = Place::where('creator', auth()->user()->id)
                ->where($this->searchColumn, 'like', '%'.
$this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        } else {
            $places = Place::where($this->searchColumn, 'like',
'%'. $this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        }

        return view('livewire.admin.places.list-places',
compact('places'));
    }
}
```

Point

CreatePoint

El controlador `CreatePoint` se encarga de crear nuevos puntos de interés en la aplicación. Sus puntos principales son:

- Define atributos para capturar el nombre, la distancia, la latitud, la longitud y el lugar del nuevo punto de interés.
- Utiliza una lista de lugares para seleccionar el lugar asociado al punto de interés.
- Escucha el evento `openCreationModal` para abrir el modal de creación.
- Implementa un método `getPlaces` para obtener la lista de lugares disponibles.
- Implementa un método `save` para validar los datos del formulario y crear un nuevo punto de interés.
- Despacha un trabajo de procesamiento después de crear el punto de interés.
- Reinicia el formulario después de crear el punto de interés y emite un evento para indicar que se ha creado un nuevo punto de interés.
- Renderiza la vista para crear un nuevo punto de interés.

```
class CreatePoint extends Component
{
    public $distance, $latitude, $longitude;
    public $places = [];

    protected $listeners = ['openCreationModal'];

    public $createForm = [
        'open' => false,
        'name' => '',
        'distance' => '',
        'latitude' => '',
        'longitude' => '',
        'place' => '',
    ];

    protected $rules = [
        'createForm.name' => 'required',
```

```
        'createForm.distance' => 'required|numeric',
        'createForm.latitude' => 'required|numeric',
        'createForm.longitude' => 'required||numeric',
        'createForm.place' => 'required|exists:places,id',
    ];

    protected $validationAttributes = [
        'createForm.name' => 'nombre',
        'createForm.distance' => 'distancia',
        'createForm.latitude' => 'latitud',
        'createForm.longitude' => 'longitud',
        'createForm.place' => 'sitio',
    ];

    public function openCreationModal()
    {
        $this->createForm['open'] = true;
        $this->getPlaces();
    }

    public function getPlaces()
    {
        $this->place = Place::all();
    }

    public function mount()
    {
        $this->places = Place::all();
    }

    public function save()
    {
        $this->validate();

        $pointOfInterest = PointOfInterest::create([
            'name' => $this->createForm['name'],
            'distance' => $this->createForm['distance'],
            'latitude' => $this->createForm['latitude'],
            'longitude' => $this->createForm['longitude'],
            'place_id' => $this->createForm['place'],
            'creator' => auth()->user()->id,
```

```
            'updater' => null,
        ]);

        ProcessPointOfInterest::dispatch($pointOfInterest);

        $this->reset('createForm');
        $this->emit('PointCreated');
        $this->emitTo('admin.point.show-point', 'render');
    }

    public function render()
    {
        return view('livewire.admin.point.create-point');
    }
}
```

EditPoint

El controlador `EditPoint` se encarga de editar puntos de interés existentes en la aplicación. Sus puntos principales son:

- Define atributos para capturar la distancia, la latitud, la longitud y el ID del punto de interés a editar, así como una lista de lugares disponibles.
- Escucha el evento `openEditModal` para abrir el modal de edición y cargar los datos del punto de interés seleccionado.
- Implementa un método `getPlaces` para obtener la lista de lugares disponibles.
- Implementa un método `update` para validar los datos del formulario y actualizar el punto de interés.
- Después de la actualización, registra un mensaje de registro y emite eventos para indicar que el punto de interés ha sido actualizado.
- Renderiza la vista para editar un punto de interés.

```
class EditPoint extends Component
{
```

```php
    public $distance, $latitude, $longitude, $pointId;
    public $places = [];

    protected $listeners = ['openEditModal'];

    public $editForm = [
        'open' => false,
        'name' => '',
        'distance' => '',
        'latitude' => '',
        'longitude' => '',
        'place' => '',
    ];

    protected $rules = [
        'editForm.name' => 'required',
        'editForm.distance' => 'required|numeric',
        'editForm.latitude' => 'required|numeric',
        'editForm.longitude' => 'required|numeric',
        'editForm.place' => 'required|exists:places,id',
    ];

    protected $validationAttributes = [
        'editForm.name' => 'nombre',
        'editForm.distance' => 'distancia',
        'editForm.latitude' => 'latitud',
        'editForm.longitude' => 'longitud',
        'editForm.place' => 'sitio',
    ];

    public function openEditModal(PointOfInterest $point)
    {
        $this->reset(['editForm']);

        $this->pointId = $point->id;
        $this->editForm['name'] = $point->name ;
        $this->editForm['distance'] = $point->distance ;
        $this->editForm['latitude'] = $point->latitude;
        $this->editForm['longitude'] = $point->longitude;
        $this->editForm['place'] = $point->place->id;
```

```
        $this->getPlaces();

        $this->editForm['open'] = true;
    }

    public function getPlaces()
    {
        $this->places = Place::all();
    }

    public function update(PointOfInterest $point)
    {
        $this->validate();

        $point->update([
            'updater' => auth()->user()->id,
            'name' => $this->editForm['name'],
            'distance' => $this->editForm['distance'],
            'latitude' => $this->editForm['latitude'],
            'longitude' => $this->editForm['longitude'],
            'place_id' => $this->editForm['place'],
        ]);

        Log::info('Point of interest with ID ' . $point->id . ' was
updated ' . $point);

        $this->editForm['open'] = false;
        $this->reset(['editForm']);

        $this->emitTo('admin.point.show-point', 'render');
        $this->emit('pointUpdated');
    }

    public function render()
    {
        return view('livewire.admin.point.edit-point');
    }
}
```

ShowPoint

El controlador `ShowPoint` se encarga de mostrar los detalles de los puntos de interés en la aplicación. Sus puntos principales son:

- Define atributos para capturar la búsqueda, el campo de búsqueda, el campo y la dirección de ordenamiento, y los detalles del modal.
- Escucha los eventos `delete` y `render` para permitir la eliminación y el renderizado de los puntos de interés.
- Implementa métodos para mostrar los detalles de un punto de interés seleccionado y para eliminar un punto de interés.
- Implementa métodos para ordenar los resultados y restablecer los filtros de búsqueda.
- Renderiza la vista para mostrar los puntos de interés.

```
class ShowPoint extends Component
{
    use WithPagination;

    public $listeners = ['delete', 'render'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';

    protected $queryString = ['search'];

    public $detailsModal = [
        'open' => false,
        'id' => null,
        'name' => null,
        'distance' => null,
        'latitude' => null,
```

```
        'longitude' => null,
        'placeId' => null,
        'placeName' => null,
        'creatorName' => null,
        'creatorId' => null,
        'updaterName' => null,
        'updaterId' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public function show(PointOfInterest $point)
    {
        $this->detailsModal['open'] = true;
        $this->detailsModal['id'] = $point->id;
        $this->detailsModal['name'] = $point->name;
        $this->detailsModal['distance'] = $point->distance;
        $this->detailsModal['latitude'] = $point->latitude;
        $this->detailsModal['longitude'] = $point->longitude;
        $this->detailsModal['placeId'] = $point->place;
        $this->detailsModal['placeName'] = $point->place->name;
        $this->detailsModal['creatorName'] =
User::find($point->creator)->name;
        $this->detailsModal['creatorId'] = $point->creator;
        $this->detailsModal['updaterName'] = $point->updater ?
User::find($point->updater)->name : null;;
        $this->detailsModal['updaterId'] = $point->updater;
        $this->detailsModal['createdAt'] = $point->created_at;
        $this->detailsModal['updatedAt'] = $point->updated_at;
    }

    public function delete(PointOfInterest $pointOfInterest)
    {
        $pointOfInterest->delete();

        Log::info('Point with ID ' . $pointOfInterest->id . ' was
deleted ' . $pointOfInterest);
    }

    public function sort($field)
    {
```

```
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }

    public function render()
    {
        if (auth()->user()->hasRole('Alumno')) {
            $points = PointOfInterest::where($this->searchColumn,
'like', '%'. $this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        } else {
            $points = PointOfInterest::where($this->searchColumn,
'like', '%'. $this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        }

        return view('livewire.admin.point.show-point',
compact('points'));
    }
}
```

ThematicArea

Utiliza la paginación para mostrar las áreas temáticas.
Escucha el evento `delete`.
Define propiedades para la búsqueda, el campo de búsqueda, el campo y la dirección de ordenamiento, y formularios para la creación y edición de áreas temáticas.
Define reglas de validación para los formularios de creación y edición.
Define atributos para mostrar los detalles de un área temática y para editarla.
Implementa métodos para guardar, actualizar y eliminar áreas temáticas.
Implementa métodos para ordenar los resultados y restablecer los filtros de búsqueda.
Define métodos para actualizar la búsqueda y renderizar las áreas temáticas según el rol del usuario autenticado.

```php
class ThematicAreas extends Component
{
    use WithPagination;

    public $listeners = ['delete'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'asc';

    protected $queryString = ['search'];

    public $createForm = [
        'open' => false,
        'name' => '',
        'description' => '',
    ];
    public $editForm = [
        'name' => '',
        'description' => '',
    ];

    protected $rules = [
        'createForm.name' => 'required|max:45',
```

```
        'createForm.description' => 'required|max:2000',
    ];

    protected $validationAttributes = [
        'createForm.name' => 'nombre',
        'createForm.description' => 'descripción',

        'editForm.name' => 'nombre',
        'editForm.description' => 'descripción',
    ];

    public $showModal = [
        'open' => false,
        'id' => null,
        'name' => null,
        'description' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public $editModal = [
        'open' => false,
        'id' => null,
    ];

    public function show(ThematicArea $thematicArea)
    {
        $this->showModal['id'] = $thematicArea->id;

        $this->showModal['name'] = $thematicArea->name;
        $this->showModal['description'] =
$thematicArea->description;
        $this->showModal['createdAt'] = $thematicArea->created_at;
        $this->showModal['updatedAt'] = $thematicArea->updated_at;

        $this->showModal['open'] = true;
    }

    public function edit(ThematicArea $thematicArea)
    {
        $this->editModal['id'] = $thematicArea->id;
```

```
        $this->editForm['name'] = $thematicArea->name;
        $this->editForm['description'] =
$thematicArea->description;

        $this->editModal['createdAt'] = $thematicArea->created_at;
        $this->editModal['updatedAt'] = $thematicArea->updated_at;

        $this->editModal['open'] = true;
    }

    public function save()
    {
        $this->validate();

        $thematicArea = ThematicArea::create([
            'name' => $this->createForm['name'],
            'description' => $this->createForm['description'],
            'updated_at' => null,
        ]);

        ProcessThematicArea::dispatch($thematicArea);

        $this->reset('createForm');

        $this->emit('thematicAreaCreated');
    }

    public function update(ThematicArea $thematicArea)
    {
        $this->validate([
            'editForm.name' => 'max:45',
            'editForm.description' => 'max:2000',
        ]);

        $thematicArea['name'] = $this->editForm['name'];
        $thematicArea['description'] =
$this->editForm['description'];

        $thematicArea->update();
```

```
        Log::info('Thematic area with ID ' . $thematicArea->id . '
was updated ' . $thematicArea);

        $this->editModal['open'] = false;
        $this->reset(['editForm']);

        $this->emit('thematicAreaUpdated');
    }

    public function delete(ThematicArea $thematicArea)
    {
        $thematicArea->pointsOfInterest()->detach();

        $thematicArea->delete();

        Log::info('Thematic area with ID ' . $thematicArea->id . '
was deleted ' . $thematicArea);
    }

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }
```

```
    public function render()
    {
        if (auth()->user()->hasRole('Administrador')
            || auth()->user()->hasRole('Profesor')) {

            $thematicAreas =
ThematicArea::where($this->searchColumn, 'like', '%' .
$this->search . '%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);

            return
view('livewire.admin.thematic-area.thematic-areas',
compact('thematicAreas'));
        }
    }
}
```

User

CreateUser

El controlador `CreateUser` se encarga de crear nuevos usuarios en la aplicación. Aquí están los puntos principales:

1. **Atributos y Listeners**: Define los atributos necesarios para el formulario de creación de usuario, así como los listeners para eventos específicos, como `openCreationModal`.
2. **Validación de Datos**: Utiliza reglas de validación para garantizar que los datos ingresados por el usuario sean válidos antes de crear un nuevo usuario.
3. **Carga de Archivos**: Utiliza el trait `WithFileUploads` para manejar la carga de archivos, en este caso, para cargar avatares de usuario.
4. **Método `save`**: Implementa el método `save` para validar los datos del formulario, crear un nuevo usuario en la base de datos, asignarle un rol y enviar una notificación de correo electrónico.
5. **Renderizado de la Vista**: El método `render` se encarga de renderizar la vista correspondiente al formulario de creación de usuario.

```
class CreateUser extends Component
{
    use WithFileUploads;

    public $roles, $avatarTemporaryUrl;

    protected $listeners = ['openCreationModal'];

    public $createForm = [
        'open' => false,
        'avatar' => null,
        'name' => '',
        'email' => '',
        'password' => '',
        'role' => '',
    ];

    protected $rules = [
        'createForm.avatar' => '',
        'createForm.name' => 'required|string',
        'createForm.email' =>
'required|confirmed|string|max:45|unique:users,email',
        'createForm.password' =>
'required|confirmed|string|min:8|max:500',
        'createForm.role' => 'required|exists:roles,id',
    ];

    protected $validationAttributes = [
        'createForm.avatar' => 'avatar',
        'createForm.name' => 'nombre',
        'createForm.email' => 'email',
        'createForm.password' => 'contraseña',
        'createForm.role' => 'rol',
    ];

    public function mount()
    {
        $this->roles = Role::all();
```

```
    }

    public function openCreationModal()
    {
        $this->createForm['open'] = true;
    }

    public function updatedCreateFormAvatar()
    {
        $this->avatarTemporaryUrl =
$this->createForm['avatar']->temporaryUrl();
    }

    public function save()
    {
        $this->validate();

        if (null !== $this->createForm['avatar']) {
            $avatarRoute =
$this->createForm['avatar']->store('public/user-avatars');
        }

        $user = User::create([
            'name' => $this->createForm['name'],
            'email' => $this->createForm['email'],
            'password'=> \bcrypt($this->createForm['password']),
            'profile_photo_path' => $avatarRoute ?? null,
        ]);

        $role = Role::findById($this->createForm['role']);
        $user->assignRole($role);

        ProcessUser::dispatch($user);
        Mail::to('admin@mail.com')->send(new UserCreated($user));

        $this->reset('createForm');
        $this->emit('userCreated');
        $this->emitTo('admin.user.list-users', 'render');
    }

    public function render()
```

```
    {
        return view('livewire.admin.user.create-user');
    }
}
```

EditUser

El controlador `EditUser` se encarga de editar usuarios existentes en la aplicación. Aquí están los puntos principales:

1. **Atributos y Listeners**: Define los atributos necesarios para el formulario de edición de usuario, así como los listeners para eventos específicos, como `openEditModal`.
2. **Validación de Datos**: Utiliza reglas de validación para garantizar que los datos ingresados por el usuario sean válidos antes de editar un usuario existente. Además, personaliza las reglas de validación para asegurarse de que el correo electrónico sea único, excluyendo el correo electrónico del usuario actual.
3. **Carga de Roles**: En el método `mount`, carga todos los roles disponibles en la aplicación para permitir al usuario seleccionar uno al editar.
4. **Método `openEditModal`**: Prepara el formulario de edición con los datos del usuario seleccionado, incluyendo su nombre, correo electrónico, contraseña (encriptada), y rol. También determina la ruta del avatar del usuario, si existe.
5. **Método `update`**: Valida los datos del formulario de edición, actualiza el usuario en la base de datos con los nuevos datos proporcionados, y actualiza sus roles. Luego, emite eventos para notificar que el usuario ha sido actualizado y que se debe renderizar la lista de usuarios.
6. **Renderizado de la Vista**: El método `render` se encarga de renderizar la vista correspondiente al formulario de edición de usuario.

```
class EditUser extends Component
{
    public $roles, $userId, $avatarRoute;

    protected $listeners = ['openEditModal'];
```

```
    public $editForm = [
        'open' => false,
        'name' => '',
        'email' => '',
        'password' => '',
        'role' => '',
    ];

    protected $rules = [
        'editForm.name' => 'required|string',
        'editForm.email' =>
'required|confirmed|string|max:45|unique:users,email',
        'editForm.email_confirmation' => '',
        'editForm.password' =>
'required|confirmed|string|min:8|max:500',
        'editForm.password_confirmation' => '',
        'editForm.role' => 'required|exists:roles,id',
    ];

    protected $validationAttributes = [
        'editForm.name' => 'nombre',
        'editForm.email' => 'email',
        'editForm.password' => 'contraseña',
        'editForm.role' => 'rol',
    ];

    public function mount()
    {
        $this->roles = Role::all();
    }

    public function openEditModal(User $user)
    {
        $this->reset(['editForm']);
        $this->reset(['avatarRoute']);

        $this->userId = $user->id;
        if ($user->profile_photo_path) {
            $this->avatarRoute =
Storage::url($user->profile_photo_path);
```

```
        }
        $this->editForm['name'] = $user->name;
        $this->editForm['email'] = $user->email;
        $this->editForm['email_confirmation'] = $user->email;
        $this->editForm['password'] = $user->password;
        $this->editForm['password_confirmation'] = $user->password;
        $this->editForm['role'] = $user->roles->first()->id;
        $this->editForm['open'] = true;
    }

    public function update(User $user)
    {
        $this->rules['editForm.email'] =
'required|confirmed|string|max:45|unique:users,email,' .
$this->userId;

        $this->validate();

        $isUpdated = $user->update([
            'name' => $this->editForm['name'],
            'email' => $this->editForm['email'],
            'password'=> \bcrypt($this->editForm['password']),
        ]);

        $role = Role::findById($this->editForm['role']);
        $user->syncRoles($role);

        if ($isUpdated) {
            Log::info('User with ID ' . auth()->user()->id . '
edited the following user ' . $user);
        }

        $this->editForm['open'] = false;
        $this->reset(['editForm']);
        $this->emitTo('admin.user.list-users', 'render');
        $this->emit('userUpdated');
    }

    public function render()
    {
        return view('livewire.admin.user.edit-user');
```

```
    }
}
```

ListUser

El controlador `ListUsers` se encarga de mostrar la lista de usuarios en la aplicación. Aquí están sus principales características:

1. **Atributos y Listeners**: Define atributos para capturar la búsqueda, el campo de búsqueda, el campo y la dirección de ordenamiento, y los detalles del modal. Además, escucha los eventos `delete` y `render` para permitir la eliminación y el renderizado de los usuarios.
2. **Método `show`**: Prepara los detalles del usuario seleccionado para mostrarlos en el modal. Incluye información como avatar, nombre, correo electrónico, contraseña, fecha de verificación de correo electrónico, fecha de creación y fecha de actualización del usuario.
3. **Método `delete`**: Elimina el usuario seleccionado y su avatar asociado, si existe. Luego, registra esta acción en el registro de eventos.
4. **Métodos para Ordenar y Restablecer Filtros**: Implementa métodos para ordenar los resultados de la lista de usuarios y para restablecer los filtros de búsqueda y ordenamiento.
5. **Método `render`**: Consulta la base de datos para obtener los usuarios que coinciden con los criterios de búsqueda y los ordena según el campo y la dirección de ordenamiento. Luego, devuelve la vista para mostrar la lista de usuarios paginada.

```
class ListUsers extends Component
{
    use WithPagination;

    public $listeners = ['delete', 'render'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';
```

```
    protected $queryString = ['search'];

    public $detailsModal = [
        'open' => false,
        'avatar' => null,
        'id' => null,
        'name' => '',
        'email' => '',
        'password' => '',
        'emailVerifiedAt' => '',
        'createdAt' => '',
        'updatedAt' => '',
    ];

    public function show(User $user)
    {
        $this->reset(['detailsModal']);

        $this->detailsModal['open'] = true;
        if ($user->profile_photo_path) {
            $this->detailsModal['avatar'] =
Storage::url($user->profile_photo_path);
        }
        $this->detailsModal['id'] = $user->id;
        $this->detailsModal['name'] = $user->name;
        $this->detailsModal['email'] = $user->email;
        $this->detailsModal['password'] = $user->password;
        $this->detailsModal['emailVerifiedAt'] =
$user->email_verified_at;
        $this->detailsModal['createdAt'] = $user->created_at;
        $this->detailsModal['updatedAt'] = $user->updated_at;
    }

    public function delete(User $user)
    {
        if(! is_null($user->profile_photo_path) &&
Storage::exists($user->profile_photo_path)) {
            Storage::delete($user->profile_photo_path);
        }

        $isDeleted = $user->delete();
```

```
        if ($isDeleted) {
            Log::alert('User with ID ' . auth()->user()->id . '
removed an user ' . $user);
        }
    }

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }

    public function render()
    {
        $users = User::where('email', '<>', auth()->user()->email)
            ->where($this->searchColumn, 'like', '%'. $this->search
.'%')
            ->orderBy($this->sortField, $this->sortDirection)
            ->paginate(10);

        return view('livewire.admin.user.list-users',
compact('users'));
    }
```

```
}
```

Video

CreateVideo

El controlador `CreateVideo` se encarga de crear nuevos videos asociados a puntos de interés y áreas temáticas en la aplicación. Aquí están sus principales características:

1. **Atributos y Listeners**: Define atributos para capturar la lista de puntos de interés, áreas temáticas, el orden del video y la URL temporal del video. Además, escucha el evento `openCreationModal` para abrir el modal de creación.
2. **Método `openCreationModal`**: Prepara el modal de creación de video estableciendo el atributo `open` en true y obteniendo la lista de puntos de interés disponibles.
3. **Métodos `getPointsOfInterest` y `getThematicAreas`**: Obtienen los puntos de interés y áreas temáticas disponibles según la selección del usuario.
4. **Método `updatedCreateFormFile`**: Actualiza la URL temporal del video cuando se selecciona un archivo.

5. **Métodos `updatedCreateFormPointOfInterest` y `setVideoOrder`**: Actualizan el orden del video y obtienen las áreas temáticas disponibles según el punto de interés seleccionado.
6. **Método `save`**: Valida los datos del formulario, guarda el video en la carpeta de almacenamiento, crea una nueva instancia de Video y VideoItem en la base de datos, y despacha eventos para procesar el video creado. Finalmente, reinicia los atributos del formulario y emite eventos para notificar que el video ha sido creado y solicitar el renderizado de la lista de videos.
7. **Método `render`**: Devuelve la vista para mostrar el formulario de creación de video.

```
class CreateVideo extends Component
{
    use WithFileUploads;

    public $pointsOfInterest = [], $thematicAreas = [], $order = 1;
    public $videoTemporaryUrl;

    protected $listeners = ['openCreationModal'];

    public $createForm = [
        'open' => false,
        'file' => null,
        'pointOfInterest' => '',
        'thematicArea' => '',
        'description' => '',
    ];

    protected $rules = [
        'createForm.file' => 'required',
        'createForm.pointOfInterest' => 'required',
        'createForm.thematicArea' =>
'required|exists:thematic_areas,id',
        'createForm.description' => 'required|string|max:2000',
    ];

    protected $validationAttributes = [
```

```
        'createForm.file' => 'archivo',
        'createForm.pointOfInterest' => 'punto de interés',
        'createForm.thematicArea' => 'área temática',
        'createForm.description' => 'descripción',
    ];

    public function openCreationModal()
    {
        $this->createForm['open'] = true;
        $this->getPointsOfInterest();
    }

    public function getPointsOfInterest()
    {
        $this->pointsOfInterest = PointOfInterest::all();
    }

    public function getThematicAreas()
    {
        $selectedPointOfInterest =
PointOfInterest::find($this->createForm['pointOfInterest']);
        $this->thematicAreas =
$selectedPointOfInterest->thematicAreas;
    }

    public function updatedCreateFormFile()
    {
        $this->videoTemporaryUrl =
$this->createForm['file']->temporaryUrl();
    }

    public function updatedCreateFormPointOfInterest()
    {
        $this->reset('order');
        $this->createForm['thematicArea'] = '';
        $this->setVideoOrder();
        $this->getThematicAreas();
    }

    public function setVideoOrder()
    {
```

```
        $videos =
PointOfInterest::find($this->createForm['pointOfInterest'])->video
s;

        if (count($videos) > 0) {
            $this->order = \count($videos) + 1;
        }
    }

    public function save()
    {
        $this->validate();

        $fileRoute =
$this->createForm['file']->store('public/videos');

        $video = Video::create([
            'route' => $fileRoute,
            'point_of_interest_id' =>
$this->createForm['pointOfInterest'],
            'order'=> $this->order,
            'creator' => auth()->user()->id,
            'updater' => null,
            'thematic_area_id' =>
$this->createForm['thematicArea'],
            'description' => $this->createForm['description'],
        ]);

        ProcessVideo::dispatch($video);

        $videoItem = VideoItem::create([
            'video_id' => $video->id,
        ]);

        ProcessVideoItem::dispatch($videoItem);

        $this->reset('videoTemporaryUrl');
        $this->reset('createForm');
        $this->emit('videoCreated');
        $this->emitTo('admin.video.list-videos', 'render');
    }
```

```
    public function render()
    {
        return view('livewire.admin.video.create-video');
    }
}
```

EditVideo

El controlador `EditVideo` se encarga de editar los detalles de un video existente en la aplicación. Aquí están sus características principales:

1. **Atributos y Listeners**: Define atributos para capturar la lista de puntos de interés, áreas temáticas y el ID y la ruta del video a editar. Además, escucha el evento `openEditModal` para abrir el modal de edición.
2. **Método `openEditModal`**: Prepara el modal de edición del video estableciendo los valores iniciales de los campos del formulario con los detalles del video seleccionado. Obtiene la lista de puntos de interés y áreas temáticas disponibles.
3. **Métodos `getPointsOfInterest` y `getThematicAreas`**: Obtienen los puntos de interés y áreas temáticas disponibles según la selección del usuario.
4. **Método `updatedEditFormPointOfInterest`**: Actualiza las áreas temáticas disponibles según el punto de interés seleccionado en el formulario de edición.
5. **Método `update`**: Valida los datos del formulario, actualiza los detalles del video en la base de datos y emite eventos para notificar que el video ha sido actualizado y solicitar el renderizado de la lista de videos.
6. **Método `render`**: Devuelve la vista para mostrar el formulario de edición de video.

```
class EditVideo extends Component
{
    public $pointsOfInterest = [], $thematicAreas = [];
    public $videoId, $videoRoute;

    protected $listeners = ['openEditModal'];

    public $editForm = [
        'open' => false,
```

```
        'pointOfInterest' => '',
        'thematicArea' => '',
        'description' => '',
    ];

    protected $rules = [
        'editForm.pointOfInterest' => 'required',
        'editForm.thematicArea' =>
'required|exists:thematic_areas,id',
        'editForm.description' => 'required|string|max:2000',
    ];

    protected $validationAttributes = [
        'editForm.pointOfInterest' => 'punto de interés',
        'editForm.thematicArea' => 'área temática',
        'editForm.description' => 'descripción',
    ];

    public function openEditModal(Video $video)
    {
        $this->reset(['editForm']);

        $this->videoId = $video->id;
        $this->videoRoute = Storage::url($video->route);
        $this->editForm['pointOfInterest'] =
$video->pointOfInterest->id ?? '';
        $this->editForm['thematicArea'] = $video->thematicArea->id
?? '';
        $this->editForm['description'] = $video->description;

        $this->getPointsOfInterest();
        $this->getThematicAreas();

        $this->editForm['open'] = true;
    }

    public function getPointsOfInterest()
    {
        $this->pointsOfInterest = PointOfInterest::all();
    }
```

```
    public function getThematicAreas()
    {
        if ( ! empty($this->thematicAreas)) {
            $selectedPointOfInterest =
PointOfInterest::find($this->editForm['pointOfInterest']);
        } else {
            $selectedPointOfInterest = PointOfInterest::first();
        }
        $this->thematicAreas =
$selectedPointOfInterest->thematicAreas;
    }

    public function updatedEditFormPointOfInterest()
    {
        $this->editForm['thematicArea'] = '';
        $this->getThematicAreas();
    }

    public function update(Video $video)
    {
        $this->validate();

        $video->update([
            'updater' => auth()->user()->id,
            'point_of_interest_id' =>
$this->editForm['pointOfInterest'],
            'thematic_area_id' => $this->editForm['thematicArea'],
            'description' => $this->editForm['description'],
        ]);

        $this->editForm['open'] = false;
        $this->reset(['editForm']);

        $this->emitTo('admin.video.list-videos', 'render');
        $this->emit('videoUpdated');
    }

    public function render()
    {
        return view('livewire.admin.video.edit-video');
    }
```

```
}
```

ListVideo

El controlador `ListVideos` se encarga de mostrar la lista de videos en la aplicación. Aquí están sus características principales:

1. **Atributos y Listeners**: Define atributos para capturar la búsqueda, el campo de búsqueda, el campo y la dirección de ordenamiento, y los detalles del modal. Escucha los eventos `delete` y `render` para permitir la eliminación y el renderizado de los videos.
2. **Método `show`**: Prepara y muestra los detalles de un video seleccionado en el modal correspondiente, incluyendo su descripción, ruta, orden, punto de interés asociado, área temática, creador, actualizador y fechas de creación y actualización.
3. **Método `delete`**: Elimina el video seleccionado, eliminando también su archivo de ruta asociado si existe en el almacenamiento.
4. **Método `sort`**: Ordena la lista de videos según el campo y la dirección de ordenamiento especificados.
5. **Métodos `resetFilters` y `updatingSearch`**: Restablecen los filtros de búsqueda y la página actual cuando se actualiza la búsqueda.
6. **Método `render`**: Renderiza la vista para mostrar la lista de videos, filtrados y ordenados según las preferencias del usuario, paginados para facilitar la navegación.

```
class ListVideos extends Component
{
    use WithPagination;

    public $listeners = ['delete', 'render'];

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';
```

```
    protected $queryString = ['search'];

    public $detailsModal = [
        'open' => false,
        'id' => null,
        'description' => null,
        'route' => null,
        'order' => null,
        'pointOfInterest' => null,
        'thematicAreaName' => null,
        'thematicAreaId' => null,
        'creatorName' => null,
        'creatorId' => null,
        'updaterName' => null,
        'updaterId' => null,
        'createdAt' => null,
        'updatedAt' => null,
    ];

    public function show(Video $video)
    {
        $this->detailsModal['open'] = true;
        $this->detailsModal['id'] = $video->id;
        $this->detailsModal['description'] = $video->description;
        $this->detailsModal['route'] = Storage::url($video->route);
        $this->detailsModal['order'] = $video->order;
        $this->detailsModal['pointOfInterest'] =
$video->pointOfInterest->id ?? '';
        $this->detailsModal['thematicAreaName'] =
$video->thematicArea->name ?? '';
        $this->detailsModal['thematicAreaId'] =
$video->thematicArea->id ?? '';
        $this->detailsModal['creatorName'] =
User::find($video->creator)->name;
        $this->detailsModal['creatorId'] = $video->creator;
        $this->detailsModal['updaterName'] = $video->updater ?
User::find($video->updater)->name : null;;
        $this->detailsModal['updaterId'] = $video->updater;
        $this->detailsModal['createdAt'] = $video->created_at;
        $this->detailsModal['updatedAt'] = $video->updated_at;
    }
```

```
    public function delete(Video $video)
    {
        if(Storage::exists($video->route)) {
            Storage::delete($video->route);
        }

        $video->delete();
    }

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
        $this->resetPage();
    }

    public function render()
    {
        if (auth()->user()->hasRole('Alumno')) {
            $videos = Video::where($this->searchColumn, 'like',
'%'. $this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        } else {
```

```
            $videos = Video::where($this->searchColumn, 'like',
'%'. $this->search .'%')
                ->orderBy($this->sortField, $this->sortDirection)
                ->paginate(10);
        }

        return view('livewire.admin.video.list-videos',
compact('videos'));
    }
}
```

VideoPreview

El controlador `VideoPreview` se encarga de mostrar una vista previa de un video en la aplicación. Aquí están sus características principales:

1. **Atributos y Listeners**: Define un atributo `$route` para almacenar la ruta del video que se mostrará en la vista previa. Además, escucha el evento `render`, aunque no se utiliza en este contexto.
2. **Método `render`**: Renderiza la vista `livewire.admin.video.video-preview`, que mostrará la vista previa del video utilizando la ruta almacenada en el atributo `$route`.

```php
class VideoPreview extends Component
{
    public $route = '';
    protected $listeners = ['render'];

    public function render()
    {
        return view('livewire.admin.video.video-preview');
    }
}
```

VideoItem

El controlador `ListVideoItems` se encarga de mostrar una lista paginada de elementos de video en la aplicación. Aquí están sus características principales:

1. **Atributos y métodos de ordenamiento y filtrado**: Define atributos para capturar la búsqueda, el campo de búsqueda, el campo y la dirección de ordenamiento. Implementa métodos para ordenar los resultados y restablecer los filtros de búsqueda.

2. **Método `render`**: Recupera los elementos de video de la base de datos según los filtros y el ordenamiento especificados. Luego, los pasa a la vista `livewire.admin.video-item.list-video-items` para su renderizado, junto con la paginación para mostrar los resultados de forma adecuada.

```
class ListVideoItems extends Component
{
    use WithPagination;

    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';

    protected $queryString = ['search'];

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
```

```
        $this->resetPage();
    }

    public function render()
    {
        $videoItems = VideoItem::where($this->searchColumn, 'like',
'%'. $this->search .'%')
            ->orderBy($this->sortField, $this->sortDirection)
            ->paginate(10);

        return view('livewire.admin.video-item.list-video-items',
compact('videoItems'));
    }
}
```

Visit

EditVisits

El controlador `EditVisits` se encarga de editar las visitas registradas en la aplicación. Aquí están sus principales características:

1. **Atributos y métodos de edición**: Define atributos para capturar el identificador del dispositivo (`deviceId`), el identificador de la visita (`visitId`) y una lista de puntos de interés disponibles (`pointsOfInterest`). Escucha el evento `openEditModal` para abrir el modal de edición y ejecutar la lógica correspondiente.
2. **Formulario de edición y validación**: Utiliza un formulario de edición (`editForm`) con campos para el agente del usuario (`useragent`), el sistema operativo (`ssoo`) y el punto de interés visitado (`pointOfInterest`). Define reglas de validación para cada campo del formulario.
3. **Método `openEditModal`**: Recibe una visita como parámetro y carga sus datos en el formulario de edición. También obtiene la lista de puntos de interés disponibles para mostrar en un menú desplegable.
4. **Método `update`**: Valida los datos del formulario de edición y luego actualiza la visita en la base de datos con los nuevos valores proporcionados. Finalmente, emite eventos para solicitar el renderizado de la lista de visitas y notificar sobre la actualización exitosa.
5. **Método `render`**: Devuelve la vista correspondiente para mostrar el formulario de edición.

```
class EditVisits extends Component
{
    public $deviceId, $visitId;
    public $pointsOfInterest = [];

    protected $listeners = ['openEditModal'];

    public $editForm = [
        'open' => false,
        'useragent' => '',
        'ssoo' => '',
        'pointOfInterest' => ''
    ];

    protected $rules = [
```

```
        'editForm.pointOfInterest' =>
'required|exist_point_of_interest',
        'editForm.useragent' => 'required|max:200',
        'editForm.ssoo' => 'required',
    ];

    protected $validationAttributes = [
        'editForm.useragent' => 'Agente',
        'editForm.ssoo' => 'ssoo',
        'editForm.description' => 'descripción',
    ];

    public function openEditModal(Visit $visit)
    {
        $this->reset(['editForm']);

        $this->visitId = $visit->id;
        $this->editForm['pointOfInterest'] =
$visit->pointOfInterest->id;
        $this->editForm['ssoo'] = $visit->ssoo;
        $this->editForm['useragent'] = $visit->useragent;

        $this->getPointsOfInterest();

        $this->editForm['open'] = true;
    }

    public function getPointsOfInterest()
    {
        $this->pointsOfInterest = PointOfInterest::all();
    }

    public function update(Visit $vist)
    {
        $this->validate();

        $vist->update([
            'updater' => auth()->user()->id,
            'point_of_interest_id' =>
$this->editForm['pointOfInterest'],
            'ssoo' => $this->editForm['ssoo'],
```

```
            'useragent' => $this->editForm['useragent'],
        ]);

        $this->editForm['open'] = false;
        $this->reset(['editForm']);

        $this->emitTo('admin.visit.show-visits', 'render');
        $this->emit('visitUpdated');
    }

    public function render()
    {
        return view('livewire.admin.visit.edit-visits');
    }
}
```

ShowVisits

El controlador `ShowVisits` se encarga de mostrar las visitas registradas en la aplicación. Aquí tienes un resumen de sus características principales:

1. **Atributos y métodos de visualización**: Define atributos para capturar el nombre del punto de interés (`pointName`), la búsqueda (`search`), la columna de búsqueda (`searchColumn`), y los detalles de la visita en un modal (`detailsModal`). Escucha los eventos `delete` y `render` para permitir la eliminación y el renderizado de las visitas.
2. **Método `show`**: Recibe una visita como parámetro y muestra sus detalles en el modal de detalles. También obtiene el nombre del punto de interés asociado a la visita para mostrarlo correctamente.
3. **Método `getPointName`**: Recibe el identificador de un punto de interés y devuelve su nombre correspondiente.
4. **Método `delete`**: Elimina la visita seleccionada.
5. **Métodos de ordenamiento y reinicio de filtros**: Define métodos para ordenar los resultados y restablecer los filtros de búsqueda.
6. **Método `render`**: Obtiene las visitas de la base de datos, las ordena según los criterios especificados y las devuelve en formato paginado para su visualización en la vista correspondiente.

```php
class ShowVisits extends Component
{
    use WithPagination;

    protected $listeners = ['delete', 'render'];

    public $pointName;
    public $search;
    public $searchColumn = 'id';

    public $sortField = 'id';
    public $sortDirection = 'desc';

    protected $queryString = ['search'];


    public $detailsModal = [
        'open' => false,
        'id' => null,
        'hour' => null,
        'deviceid' => null,
        'appversion' => null,
        'useragent' => null,
        'ssoo' => null,
        'ssooversion' => null,
        'latitude' => null,
        'longitude' => null,
        'point_of_interest_id' => null,
    ];

    public function show(Visit $visit)
    {
        $this->pointName =
$this->getPointName($visit->point_of_interest_id);

        $this->detailsModal['open'] = true;
        $this->detailsModal['id'] = $visit->id;
        $this->detailsModal['hour'] = $visit->hour;
        $this->detailsModal['deviceid'] = $visit->deviceid;
        $this->detailsModal['appversion'] = $visit->appversion;
```

```
        $this->detailsModal['useragent'] = $visit->useragent;
        $this->detailsModal['ssoo'] = $visit->ssoo;
        $this->detailsModal['ssooversion'] = $visit->ssooversion;
        $this->detailsModal['latitude'] = $visit->latitude;
        $this->detailsModal['longitude'] = $visit->longitude;
        $this->detailsModal['point_of_interest_id'] =
$this->pointName;
        $this->detailsModal['createdAt'] = $visit->created_at;

    }

    public function getPointName($pointId)
    {
        return PointOfInterest::find($pointId);
    }

    public function delete(Visit $visit)
    {
        $visit->delete();
    }

    public function sort($field)
    {
        if ($this->sortField === $field && $this->sortDirection !==
'desc') {
            $this->sortDirection = 'desc';
        } else {
            $this->sortDirection = 'asc';
        }

        $this->sortField = $field;
    }

    public function resetFilters()
    {
        $this->reset(['search', 'sortField', 'sortDirection']);
        $this->resetPage();
    }

    public function updatingSearch()
    {
```

```
        $this->resetPage();
    }

    public function render()
    {
        $visits = Visit::where($this->searchColumn, 'like', '%'.
$this->search .'%')
            ->orderBy($this->sortField, $this->sortDirection)
            ->paginate(10);

        return view('livewire.admin.visit.show-visits',
compact('visits'));
    }
}
```

Archivos de configuración de la aplicación

Mail.php

Configuración del servidor de envío de emails al servicio de mensajeria Mailtrap.

```
<?php

return [

    /*

|--------------------------------------------------------------------------
----------
    | Mail Driver
```

```
|--------------------------------------------------------------------
----------
    |
    | Laravel supports both SMTP and PHP's "mail" function as
drivers for the
    | sending of e-mail. You may specify which one you're using
throughout
    | your application here. By default, Laravel is setup for SMTP
mail.
    |
    | Supported: "smtp", "sendmail", "mailgun", "ses",
    |            "postmark", "log", "array"
    |
    */

    'driver' => env('MAIL_DRIVER', 'smtp'),

    /*

|--------------------------------------------------------------------
----------
    | SMTP Host Address

|--------------------------------------------------------------------
----------
    |
    | Here you may provide the host address of the SMTP server
used by your
    | applications. A default option is provided that is
compatible with
    | the Mailgun mail service which will provide reliable
deliveries.
    |
    */

    'host' => env('MAIL_HOST', 'smtp.mailtrap.io'),

    /*
```

```
    |--------------------------------------------------------------------------
    | SMTP Host Port
    |--------------------------------------------------------------------------
    |
    | This is the SMTP port used by your application to deliver e-mails to
    | users of the application. Like the host we have set this value to
    | stay compatible with the Mailgun e-mail application by default.
    |
    */

    'port' => env('MAIL_PORT', 587),

    /*
    |--------------------------------------------------------------------------
    | Global "From" Address
    |--------------------------------------------------------------------------
    |
    | You may wish for all e-mails sent by your application to be sent from
    | the same address. Here, you may specify a name and address that is
    | used globally for all e-mails that are sent by your application.
    |
    */

    'from' => [
        'address' => env('MAIL_FROM_ADDRESS', 'no-reply@gtv.com'),
        'name' => env('MAIL_FROM_NAME', 'GTV'),
    ],
```

```
    /*
    |--------------------------------------------------------------------------
    | E-Mail Encryption Protocol
    |--------------------------------------------------------------------------
    |
    | Here you may specify the encryption protocol that should be used when
    | the application send e-mail messages. A sensible default using the
    | transport layer security protocol should provide great security.
    |
    */

    'encryption' => env('MAIL_ENCRYPTION', 'tls'),

    /*
    |--------------------------------------------------------------------------
    | SMTP Server Username
    |--------------------------------------------------------------------------
    |
    | If your SMTP server requires a username for authentication, you should
    | set it here. This will get used to authenticate with your server on
    | connection. You may also set the "password" value below this one.
    |
    */

    'username' => env('MAIL_USERNAME'),
```

```
    'password' => env('MAIL_PASSWORD'),

    /*
    |--------------------------------------------------------------------------
    | Sendmail System Path
    |--------------------------------------------------------------------------
    |
    | When using the "sendmail" driver to send e-mails, we will need to know
    | the path to where Sendmail lives on this server. A default path has
    | been provided here, which will work well on most of your systems.
    |
    */

    'sendmail' => '/usr/sbin/sendmail -bs',

    /*
    |--------------------------------------------------------------------------
    | Markdown Mail Settings
    |--------------------------------------------------------------------------
    |
    | If you are using Markdown based email rendering, you may configure your
    | theme and component paths here, allowing you to customize the design
    | of the emails. Or, you may simply stick with the Laravel defaults!
    |
    */

    'markdown' => [
```

```
        'theme' => 'default',

        'paths' => [
            resource_path('views/vendor/mail'),
        ],
    ],

    /*
|--------------------------------------------------------------------------
    | Log Channel
|--------------------------------------------------------------------------
    |
    | If you are using the "log" driver, you may specify the
logging channel
    | if you prefer to keep mail messages separate from other log
entries
    | for simpler reading. Otherwise, the default channel will be
used.
    |
    */

    'log_channel' => env('MAIL_LOG_CHANNEL'),

];
```

Tests

Test para comprobar que accedes a la página del GTVisor y funciona.

```
    public function testMapPageWithGTVisor()
    {
        // Ejecuta el seeder para crear los usuarios, incluido GTVisor
        $this->seed(UserSeeder::class);
```

```
        // Encuentra el usuario GTVisor creado por el seeder
        $gtvisor = User::where('email', 'gtvisor@mail.com')->first();

        // Autentica como el usuario GTVisor
        $this->actingAs($gtvisor);

        // Realiza una solicitud GET a la página
        $response = $this->get('/');

        // Asegura que la respuesta tenga un código de estado 200 (OK)
        $response->assertStatus(200);
    }
```

Test para crear un punto y comprobar que se ve en el mapa de la página desde el usuario GTVisor.

```
public function testGTVisorPointIsCreatedAndVisibleOnMap()
    {
        // Ejecutar el seeder para crear los usuarios y el lugar
        $this->seed(UserSeeder::class);
        $this->seed(PlaceSeeder::class);

        // Buscar el usuario GTVisor y el lugar creado por los seeders
        $gtvisor = User::where('email', 'gtvisor@mail.com')->first();
        $place = Place::first();

        // Autenticar como el usuario GTVisor
        $this->actingAs($gtvisor);

        // Asegurar que no hay puntos de interés en la base de datos
inicialmente
        $this->assertDatabaseCount('point_of_interests', 0);

        // Simular la interacción del usuario con el componente Livewire
para crear un nuevo punto de interés
        Livewire::test(CreatePoint::class)
            ->set('createForm.name', 'Prueba')
            ->set('createForm.distance', '99')
            ->set('createForm.latitude', '10')
```

```
            ->set('createForm.longitude', '15')
            ->set('createForm.place', $place->id)
            ->call('save');

        // Verificar que se haya creado el punto de interés en la base
de datos
        $this->assertDatabaseCount('point_of_interests', 1);

        // Obtener el último punto de interés creado en la base de datos
        $point = PointOfInterest::latest()->first();

        // Realizar una solicitud HTTP a la página del mapa
        $response = $this->get('/');

        // Verificar que la página cargue correctamente
        $response->assertStatus(200);

        // Verificar que el marcador del punto de interés creado esté
presente en el HTML generado por la página
        $response->assertSee($point->latitude);
        $response->assertSee($point->longitude);
        $response->assertSee($point->name);
    }
```

Test para crear un punto y comprobar que se ve en el mapa de la página desde el usuario Admin.

```
  public function testAdminPointIsCreatedAndVisibleOnMap()
    {
        // Crear un usuario administrador y un lugar
        $adminUser = $this->createAdmin();
        $place = $this->createPlace();

        // Actuar como el usuario administrador
        $this->actingAs($adminUser);

        // Asegurar que no hay puntos de interés en la base de datos
inicialmente
        $this->assertDatabaseCount('point_of_interests', 0);
```

```
        // Simular la interacción del usuario con el componente Livewire
para crear un nuevo punto de interés
        Livewire::test(CreatePoint::class)
            ->set('createForm.name', 'Prueba')
            ->set('createForm.distance', '99')
            ->set('createForm.latitude', '10')
            ->set('createForm.longitude', '15')
            ->set('createForm.place', $place->id)
            ->call('save');

        // Verificar que se haya creado el punto de interés en la base
de datos
        $this->assertDatabaseCount('point_of_interests', 1);

        // Obtener el último punto de interés creado en la base de datos
        $point = PointOfInterest::latest()->first();

        // Realizar una solicitud HTTP a la página del mapa
        $response = $this->get('/');

        // Verificar que la página cargue correctamente
        $response->assertStatus(200);

        // Verificar que el marcador del punto de interés creado esté
presente en el HTML generado por la página
        $response->assertSee($point->latitude);
        $response->assertSee($point->longitude);
        $response->assertSee($point->name);
    }
```

Cambiar a rol gtvisor

En este test comprobamos la funcionalidad de los controladores que permiten a partir del rol actual del usuario ya sea admin, estudiante o teacher. poder cambiar rápidamente a rol -> GTVisor el cual permite visualizar contenido específico para él, además de poder volver al rol anterior si se quisiera.

```php
public function can_change_user_to_gtv_user_and_rollback()
{
    Role::create(['name' => 'GTVisor']);
    $user = User::factory()->create();
    $role = Role::create(['name' => 'Alumno']);

    $user->assignRole($role);

    Livewire::test(CreateUser::class)
        ->set('createForm.name', $user->name)
        ->set('createForm.email', $user->email)
        ->set('createForm.password', $user->password)
        ->set('createForm.role', $role->id)
        ->call('save');

    $this->assertDatabaseHas('users', [
        'name' => $user->name,
        'email' => $user->email,
    ]);

    $response = $this->actingAs($user)->get('/user/profile');

    $response->assertSee('Change to a GTVisor user.');

    // cambiamos de alumno -> gtvisor
    Livewire::test(CreateGtvisorUser::class)
        ->call('confirmChangeUser')
        ->assertSet('confirmingUserDeletion', true)
        ->set('confirmingUserDeletion', false)
        ->call('changeUser');

    $response2 = $this->actingAs($user)->get('/user/profile');
    $response2->assertSee('Revert to User');

    // regresamos de gtvisor -> alumno
    Livewire::test(RevertGtvisorUser::class)
        ->call('confirmRevertUser')
        ->assertSet('confirmingUserReversion', true)
        ->set('confirmingUserReversion', false)
```

```
        ->call('revertUser');

    $response3 = $this->actingAs($user)->get('/user/profile');
    $response3->assertSee('Change to a GTVisor user.');
}
```

Trait

```php
public function createStudent(array $attributes = [])
{
    $studentRole = Role::firstOrCreate(['name' => 'Alumno']);

    $defaultAttributes = [
        'name' => 'Student',
        'email' => 'student' . uniqid() . '@mail.com',
        'password' => bcrypt('password'),
    ];

    $userAttributes = array_merge($defaultAttributes, $attributes);

    $user = User::factory()->create($userAttributes);
    $user->assignRole($studentRole);
    return $user;
}
```

El método createStudent dentro del trait TestHelpers en Laravel se utiliza para crear usuarios con el rol de 'Alumno'. Este método es útil durante las pruebas para generar usuarios con atributos específicos de manera rápida y consistente.

Este metodo acepta un array de atributos como argumento opcional y luego lo mergea con el array de atributos por defecto.

```php
$studentRole = Role::firstOrCreate(['name' => 'Alumno']);
```

firstOrCreate se hace para asegurar que el rol 'Alumno' exista antes de asignarlo al usuario. Esto es importante porque permite que el método createStudent pueda ser llamado múltiples veces sin causar problemas de duplicación o errores relacionados con la inexistencia del rol. De esta forma el rol va a ser único y no se va a crear varias veces el mismo rol si usamos en un mismo test varias veces el metodo createStudent().

UpdateUserTest

La clase UpdateUserTest se utiliza para verificar la funcionalidad de actualización de usuarios en la aplicación a través de varios escenarios de prueba. Esta clase utiliza el trait TestHelpers para crear usuarios con roles específicos, permitiendo que las pruebas sean más eficientes y evitando la duplicación de código.

- Propósito de la Clase: la clase UpdateUserTest está diseñada para asegurarse de que las características de edición de usuarios funcionen correctamente en una aplicación Laravel utilizando Livewire. Las pruebas incluidas en esta clase validan la capacidad de actualizar la información de un usuario y aseguran que se cumplan todas las reglas de validación relevantes.

1. Creación de Usuarios:
    - Usuario Existente ($existingUser): Se crea un usuario estudiante utilizando el método createStudent del trait TestHelpers. Este usuario ya existe en el sistema y su correo electrónico será utilizado para probar la unicidad.
    - Usuario Administrador ($adminUser): Se crea un usuario administrador utilizando el método createAdmin del mismo trait. Este usuario tendrá permisos para realizar la actualización.
    - Usuario a Editar ($user): Se crea otro usuario estudiante que será el objeto de la actualización.
2. Autenticación del Usuario Administrador:
    - Se autentica como el usuario administrador utilizando Livewire::actingAs($adminUser). Esto asegura que las acciones siguientes se realicen con los permisos adecuados.
3. Prueba del Componente Livewire:
    - Inicialización del Componente:
        - Se inicializa el componente EditUser llamando a Livewire::test(EditUser::class).
    - Apertura del Modal de Edición:
        - Se llama al método openEditModal pasando el usuario que se desea editar ($user).
    - Configuración de Campos del Formulario:
        - Se establece el campo de correo electrónico (editForm.email) del formulario con el correo del existingUser.
        - Se confirma el correo electrónico (editForm.email_confirmation) con el mismo valor.
    - Intento de Actualización:

        - Se llama al método update para intentar guardar los cambios en el usuario.
4. Verificación de Errores:
    - Validación de Unicidad:
        - Se verifica que el formulario tiene errores relacionados con el campo email, específicamente que el correo debe ser único (assertHasErrors(['editForm.email' => 'unique'])).

Importancia de la Prueba

- Integridad de los Datos: Asegura que no haya duplicados de correos electrónicos en la base de datos, lo cual es crucial para la integridad y consistencia de los datos del usuario.
- Validación de Reglas de Negocio: Verifica que las reglas de negocio relacionadas con la unicidad de los correos electrónicos se apliquen correctamente.
- Seguridad: Ayuda a prevenir problemas de seguridad y acceso no autorizado, ya que los correos electrónicos duplicados pueden llevar a problemas de autenticación y autorización.

Uso del Trait TestHelpers

El trait TestHelpers proporciona métodos para crear usuarios con roles específicos de manera eficiente:

- createStudent: Crea usuarios estudiantes con atributos predeterminados y únicos (gracias al uso de uniqid() para los correos electrónicos).
- createAdmin: Crea un usuario administrador, asegurando que las pruebas se realicen con los permisos necesarios.

```
public function password_length_must_not_exceed_500_characters()

{

    $adminUser = $this->createAdmin();

    $user = $this->createStudent();



    Livewire::actingAs($adminUser)

        ->test(EditUser::class)

        ->call('openEditModal', $user)
```

```
        ->set('editForm.password', str_repeat('a', 501))

        ->call('update', $user)

        ->assertHasErrors(['editForm.password' => 'max']);
}
```

En este test que comprueba que no se exceda los 500 caracteres al actualizar la contraseña, podemos ver que se usa el metodo str_repeat para repetir el caracter “a” 501 veces.

ListUsersTest

La clase ListUsersTest contiene una serie de pruebas para verificar la funcionalidad del componente Livewire ListUsers, que se utiliza en la interfaz de administración para listar y gestionar usuarios. Estas pruebas comprueban que los usuarios se visualizan correctamente, se pueden buscar y ordenar de diferentes maneras, y que los filtros se comportan adecuadamente.

Propósito de la Clase

El propósito de esta clase es asegurarse de que el componente ListUsers funcione correctamente, permitiendo al administrador ver, buscar, ordenar y filtrar usuarios en la aplicación. Esto se verifica mediante una serie de pruebas automatizadas que simulan interacciones del usuario y verifican los resultados esperados.

```
public function it_see_the_users()
{
    $adminUser = $this->createAdmin();
    $teacherRole = Role::create(['name' => 'Profesor']);

    $user1 = $this->createStudent();
    $user2 = $this->createStudent();

    $user2->assignRole($teacherRole);

    $this->assertDatabaseCount('users',3);

    Livewire::actingAs($adminUser)
        ->test(ListUsers::class)
        ->call('show', $user1)
```

```
        ->assertSee($user1->name)
        ->assertSee($user1->email)
        ->assertSee($user1->roles->first()->name)
        ->assertSee($user1->created_at->toDateString())
        ->assertSee($user1->updated_at->toDateString())
        ->call('show', $user2)
        ->assertSee($user2->name)
        ->assertSee($user2->email)
        ->assertSee($user2->roles->first()->name)
        ->assertSee($user2->created_at->toDateString())
        ->assertSee($user2->updated_at->toDateString());
}
```

En este test se comprueba que se ven en la lista los usuarios, para ello creamos un administrador para que vea la lista y dos usuarios con distinto rol y como siempre usando el trait, llamamos al método show de ListUsers y una cosa interesante es que para ver la fecha en la lista usamos el método toDateString() que usa para convertir una instancia de Carbon (que representa una fecha y hora) a una cadena de texto que contiene sólo la fecha en formato 'YYYY-MM-DD'. Esto es útil para verificar que las fechas created_at y updated_at de los usuarios se muestran correctamente en la interfaz.

```
public function it_searches_users_by_id()
{
    $adminUser = $this->createAdmin();
    $user1 = $this->createStudent();
    $user2 = $this->createStudent();
    $user3 = $this->createStudent();

    $this->assertDatabaseCount('users',4);

    Livewire::actingAs($adminUser)
        ->test(ListUsers::class)
        ->set('searchColumn', 'id')
        ->set('search', $user2->id)
        ->call('render')
        ->assertSee($user2->email)
        ->assertDontSee($user1->email)
        ->assertDontSee($user3->email);
}
```

Aqui para buscar por “id” se crean 3 usuarios y se llama al método searchColumn de ListUsers para buscar por “id” y se llama al método search para poner el id del user2 y se

llama a render para recargar la vista y hacer que se apliquen los cambios de la búsqueda y comprobar que efectivamente se ve el email del user 2 pero no los de los usuarios 1 y 3.

```
public function it_searches_users_by_created_at()
{
    $adminUser = $this->createAdmin();
    $user1 = $this->createStudent(['created_at' =>
now()->subDays(2)]);
    $user2 = $this->createStudent(['created_at' =>
now()->subDays(1)]);
    $user3 = $this->createStudent(['created_at' => now()]);

    Livewire::actingAs($adminUser)
        ->test(ListUsers::class)
        ->set('searchColumn', 'created_at')
        ->set('search', $user1->created_at->toDateString())
        ->call('render')
        ->assertSee($user1->email)
        ->assertDontSee($user2->email)
        ->assertDontSee($user3->email);
}
```

El test it_searches_users_by_created_at verifica que el componente ListUsers puede buscar y filtrar usuarios por la fecha de creación (created_at).
$user1 = $this->createStudent(['created_at' => now()->subDays(2)]);

- Se crea un usuario con la fecha de creación establecida a hace 2 días. now()->subDays(2) utiliza la biblioteca Carbon para obtener la fecha y hora actuales y restarle 2 días.

```
public function it_searches_users_by_name_and_sorts_by_email_asc()
{
    $adminUser = $this->createAdmin();
    $user1 = $this->createStudent(['name' => 'Paco', 'email' =>
'b_student@mail.com']);
    $user2 = $this->createStudent(['name' => 'Pablo', 'email' =>
'a_student@mail.com']);
    $user3 = $this->createStudent();
```

```
    $this->assertDatabaseCount('users', 4);

    Livewire::actingAs($adminUser)
        ->test(ListUsers::class)
        ->set('searchColumn', 'name')
        ->set('search', 'Pa')
        ->call('sort', 'email')
        ->call('render')
        ->assertSeeInOrder([$user2->email, $user1->email])
        ->assertDontSee($user3->email);
}
```

Este es un test doble de filtros que busca el usuario por nombre y lo ordena por email ascendente, por eso se le asigna un email que empieza por la letra “b” al primer usuario y al segundo por la letra “a”, y a parte hay que crear a los usuarios con un nombre que comparta el principio.

```
public function
it_searches_users_by_updated_at_and_sorts_by_email_desc()
{
    $adminUser = $this->createAdmin();
    $user1 = $this->createStudent(['email'=>'b_student@mail.com',
'updated_at' => now()->subDays(1)]);
    $user2 = $this->createStudent(['email'=>'a_student@mail.com',
'updated_at' => now()->subDays(1)]);
    $user3 = $this->createStudent(['updated_at' => now()]);

    $this->assertDatabaseCount('users', 4);

    Livewire::actingAs($adminUser)
        ->test(ListUsers::class)
        ->set('searchColumn', 'updated_at')
        ->set('search', now()->subDays(1)->toDateString())
        ->call('sort', 'email')
        ->call('sort', 'email')
        ->call('render')
        ->assertSeeInOrder([$user1->email, $user2->email])
        ->assertDontSee($user3->email);
}
```

En este test de dos filtros a la vez, al ordenar descendentemente hay que llamar dos veces al método sort de ListUsers

1. Primera Llamada a sort('email'):
    - Si sortField no es email, se establece sortField a email y sortDirection a asc.
2. Segunda Llamada a sort('email'):
    - Dado que sortField ya es email y sortDirection es asc (resultado de la primera llamada), la segunda llamada cambia sortDirection a desc.

```
public function sort($field)
{
    if ($this->sortField === $field && $this->sortDirection !==
'desc') {
        $this->sortDirection = 'desc';
    } else {
        $this->sortDirection = 'asc';
    }

    $this->sortField = $field;
}
```

CreateUserTest

Esta clase extiende TestCase e incluye varios métodos de prueba que verifican diferentes aspectos de la creación de usuarios, asegurando que se cumplan las validaciones y que la funcionalidad opere correctamente.
Propósito
El propósito principal de esta clase de prueba es asegurar que la funcionalidad de creación de usuarios opere como se espera. Esto incluye verificar la creación de usuarios con atributos válidos, validar errores en campos obligatorios y restricciones específicas, y asegurar que ciertos criterios de negocio se cumplan, como la unicidad del correo electrónico y las validaciones de contraseñas.

```
public function
the_password_must_not_exceed_maximum_length_creating_a_user()
{
    $adminUser = $this->createAdmin();

    $studentRole = \Spatie\Permission\Models\Role::create(['name' =>
'Alumno']);
```

```
    $longPassword = str_repeat('a', 501); // Contraseña de más de 500
caracteres

    Livewire::actingAs($adminUser)
        ->test(CreateUser::class)
        ->set('createForm.open', true)
        ->set('createForm.name', 'john')
        ->set('createForm.email', 'john@mail.com')
        ->set('createForm.email_confirmation', 'john@mail.com')
        ->set('createForm.password', $longPassword)
        ->set('createForm.password_confirmation', $longPassword)
        ->set('createForm.role', $studentRole->id)
        ->call('save')
        ->assertHasErrors(['createForm.password' => 'max']);

    $this->assertDatabaseCount('users', 1); // Solo el administrador
}
```

En esta clase hay que crear siempre un rol antes para que no de error ya que tiene que existir un rol al crear un usuario ya que el rol es obligatorio.

```
public function the_role_must_exist_creating_a_user()
{
    $adminUser = $this->createAdmin();

    $invalidRoleId = 999; // Suponiendo que este ID no existe en la
base de datos

    Livewire::actingAs($adminUser)
        ->test(CreateUser::class)
        ->set('createForm.open', true)
        ->set('createForm.name', 'john')
        ->set('createForm.email', 'john@mail.com')
        ->set('createForm.email_confirmation', 'john@mail.com')
        ->set('createForm.password', 'password')
        ->set('createForm.password_confirmation', 'password')
        ->set('createForm.role', $invalidRoleId)
        ->call('save')
        ->assertHasErrors(['createForm.role' => 'exists']);

    $this->assertDatabaseCount('users', 1); // Solo el administrador
}
```

La biblioteca Spatie Laravel Permission es una biblioteca popular que facilita la gestión de roles y permisos en aplicaciones Laravel. Proporciona una manera sencilla de asignar roles y permisos a los usuarios y de verificar los permisos dentro de tu aplicación.
En el test, el rol se crea utilizando la clase Role proporcionada por la biblioteca Spatie Laravel Permission. Esta biblioteca debe estar instalada en tu proyecto Laravel. Puedes instalarla mediante Composer con el siguiente comando:

composer require spatie/laravel-permission

- Instalación y Configuración de Spatie Laravel Permission

Para utilizar Spatie Laravel Permission, necesitas seguir estos pasos:

Instalación:
composer require spatie/laravel-permission

Publicación de la Configuración:
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

Ejecución de Migraciones:

php artisan migrate

Configuración del Modelo de Usuario:
Añadir el trait HasRoles al modelo User:
php
Copiar código
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    use HasRoles;

    // ...
}

```php
public function the_avatar_must_be_an_image()
{
    $adminUser = $this->createAdmin();
```

```
    $studentRole = \Spatie\Permission\Models\Role::create(['name' =>
'Alumno']);

    $invalidAvatar = UploadedFile::fake()->create('document.pdf',
100, 'application/pdf');

    Livewire::actingAs($adminUser)
        ->test(CreateUser::class)
        ->set('createForm.open', true)
        ->set('createForm.name', 'John Doe')
        ->set('createForm.email', 'john@example.com')
        ->set('createForm.email_confirmation', 'john@example.com')
        ->set('createForm.password', 'password')
        ->set('createForm.password_confirmation', 'password')
        ->set('createForm.role', $studentRole->id)
        ->set('createForm.avatar', $invalidAvatar)
        ->call('save')
        ->assertHasErrors(['createForm.avatar' => 'image']);

    $this->assertDatabaseCount('users', 1); // Solo el administrador
}
```

UploadedFile::fake():

- Laravel proporciona esta funcionalidad para crear archivos subidos falsos. Esto es útil para pruebas, ya que permite simular la subida de archivos sin necesidad de cargar archivos reales.
- La clase UploadedFile es parte de Laravel y se encuentra en el namespace Illuminate\Http.

create('document.pdf', 100, 'application/pdf'):

- 'document.pdf': Es el nombre del archivo falso que se está creando. En este caso, se nombra el archivo como document.pdf.
- 100: Es el tamaño del archivo en kilobytes. Aquí, se crea un archivo falso con un tamaño de 100 KB.
- 'application/pdf': Es el tipo MIME del archivo. En este caso, se especifica que el archivo es un PDF (application/pdf).

```
/** @test */
public function the_avatar_must_not_exceed_maximum_size()
```

```
{
    $adminUser = $this->createAdmin();

    $studentRole = \Spatie\Permission\Models\Role::create(['name' =>
'Alumno']);

    $largeAvatar =
UploadedFile::fake()->create('large-avatar.jpg')->size(2048); // 2MB
file

    Livewire::actingAs($adminUser)
        ->test(CreateUser::class)
        ->set('createForm.open', true)
        ->set('createForm.name', 'John Doe')
        ->set('createForm.email', 'john@example.com')
        ->set('createForm.email_confirmation', 'john@example.com')
        ->set('createForm.password', 'password')
        ->set('createForm.password_confirmation', 'password')
        ->set('createForm.role', $studentRole->id)
        ->set('createForm.avatar', $largeAvatar)
        ->call('save')
        ->assertHasErrors(['createForm.avatar' => 'max']);

    $this->assertDatabaseCount('users', 1); // Solo el administrador
}
```

$largeAvatar = UploadedFile::fake()->create('large-avatar.jpg')->size(2048);

1. UploadedFile::fake():
    - Laravel proporciona esta funcionalidad para crear archivos subidos falsos. Es una herramienta útil para pruebas, ya que permite simular la subida de archivos sin necesidad de interactuar con el sistema de archivos real.
2. create('large-avatar.jpg'):
    - 'large-avatar.jpg': Es el nombre del archivo falso que se está creando. En este caso, el archivo se llama large-avatar.jpg.
    - El método create genera un archivo falso con el nombre especificado.
3. ->size(2048):
    - 2048: Especifica el tamaño del archivo en kilobytes (KB). Aquí, se crea un archivo de imagen falso con un tamaño de 2048 KB, lo que equivale a 2 MB.
    - El método size se encadena para establecer el tamaño del archivo falso.

```php
/** @test */
public function can_create_user_with_valid_avatar()
{
    $adminUser = $this->createAdmin();

    $studentRole = \Spatie\Permission\Models\Role::create(['name' =>
'Alumno']);

    $validAvatar = UploadedFile::fake()->image('avatar.jpg', 100,
100)->size(512); // 512KB file

    Livewire::actingAs($adminUser)
        ->test(CreateUser::class)
        ->set('createForm.open', true)
        ->set('createForm.name', 'John Doe')
        ->set('createForm.email', 'john@example.com')
        ->set('createForm.email_confirmation', 'john@example.com')
        ->set('createForm.password', 'password')
        ->set('createForm.password_confirmation', 'password')
        ->set('createForm.role', $studentRole->id)
        ->set('createForm.avatar', $validAvatar)
        ->call('save');

    $this->assertDatabaseHas('users', [
        'name' => 'John Doe',
        'email' => 'john@example.com',
    ]);
}
```

```
$validAvatar = UploadedFile::fake()->image('avatar.jpg', 100, 100)->size(512);
```

1. UploadedFile::fake():
    - Laravel proporciona esta funcionalidad para crear archivos subidos falsos. Esto es útil para pruebas, ya que permite simular la subida de archivos sin necesidad de interactuar con el sistema de archivos real.
    - La clase UploadedFile es parte de Laravel y se encuentra en el namespace Illuminate\Http.
2. image('avatar.jpg', 100, 100):
    - 'avatar.jpg': Es el nombre del archivo falso que se está creando. En este caso, el archivo se llama avatar.jpg.

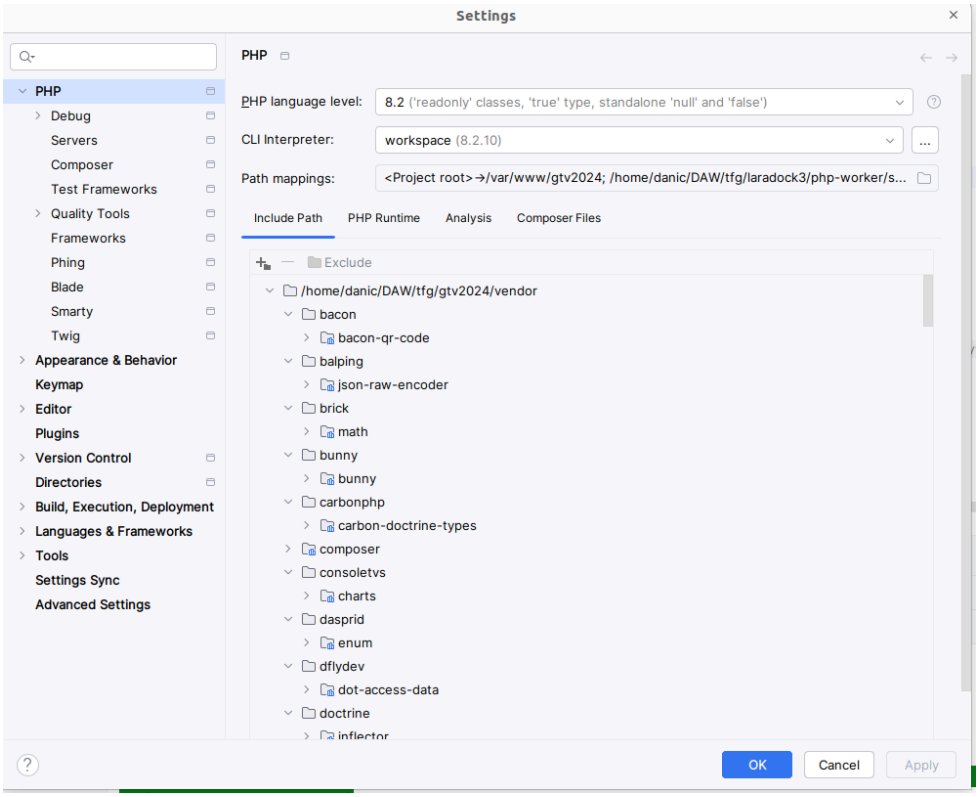
- 100, 100: Especifica las dimensiones de la imagen en píxeles (ancho y alto). Aquí, se crea una imagen de 100x100 píxeles.
- El método image genera un archivo de imagen falso con las dimensiones especificadas.

3. ->size(512):
   - 512: Especifica el tamaño del archivo en kilobytes (KB). Aquí, se crea un archivo de imagen falso con un tamaño de 512 KB.
   - El método size se encadena para establecer el tamaño del archivo falso.

Lanzar test con Phpstorm

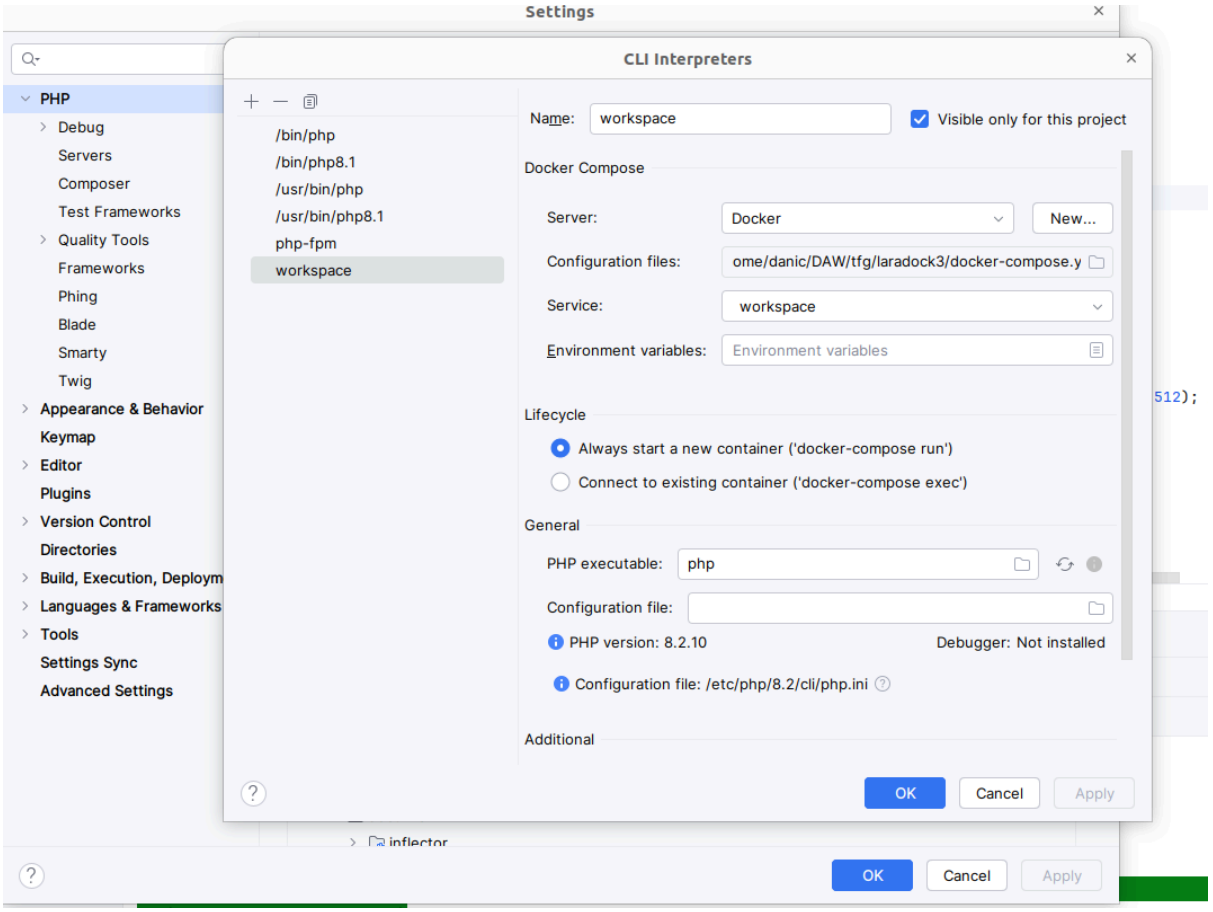
1- Primero hay que crear un CLI Interpreter configurado para usar el contenedor workspace:
Esto permite ejecutar scripts PHP directamente desde PhpStorm dentro del contenedor Docker configurado, esto facilita la configuración y ejecución de pruebas unitarias con PHPUnit. Al usar el CLI interpreter configurado para Docker, PhpStorm puede ejecutar las pruebas dentro del contenedor, asegurando que se utilice el mismo entorno que el de desarrollo.

Ve a File -> Settings -> PHP.
Haz clic en los tres puntos al lado de CLI Interpreter.



Haz clic en el botón + y selecciona From Docker, Vagrant, VM, Remote....

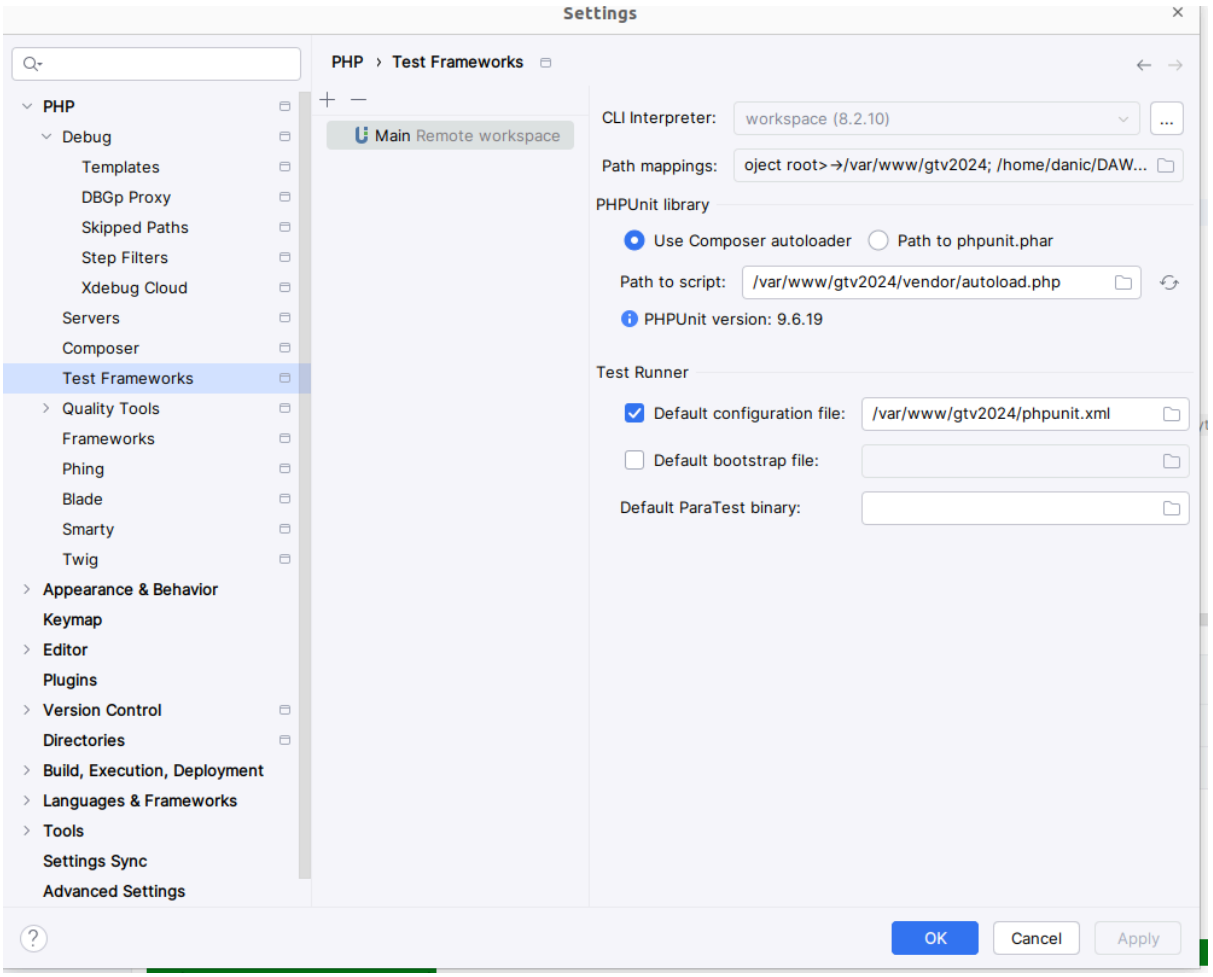
Selecciona Docker Compose.



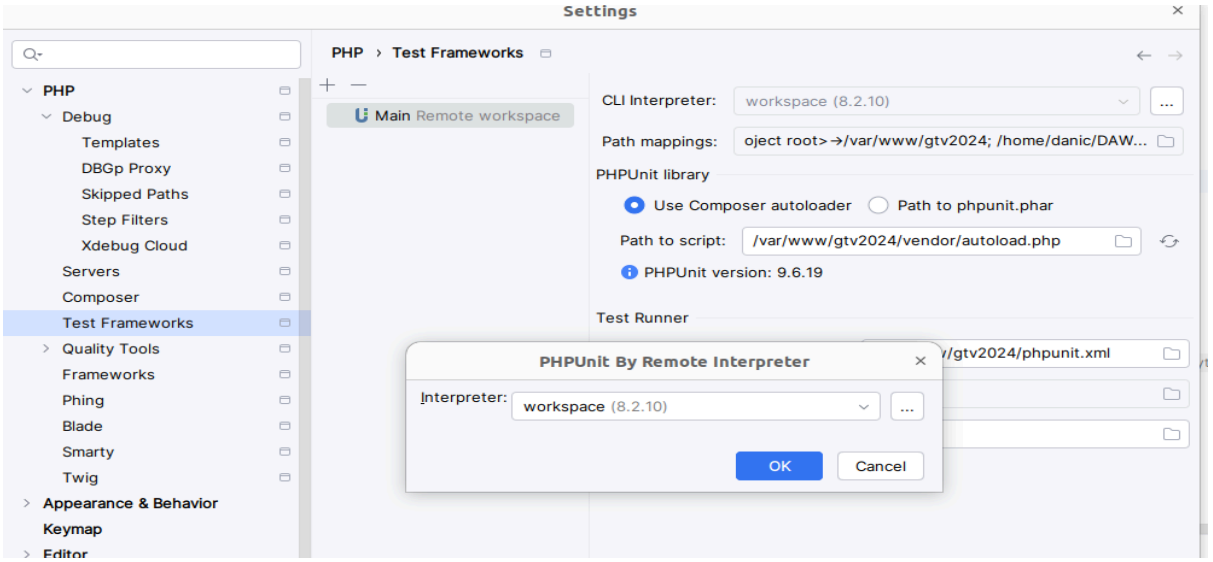
En Configuration files, selecciona tu archivo docker-compose.yml ubicado en /path/to/your/project/laradock.
En Service, selecciona workspace.
Asegúrate de que la ruta del intérprete de PHP sea correcta

2-Configurar Path Mappings:

En File -> Settings -> PHP->Test Frameworks hay que crear el mapeado de la ruta local y remoto.

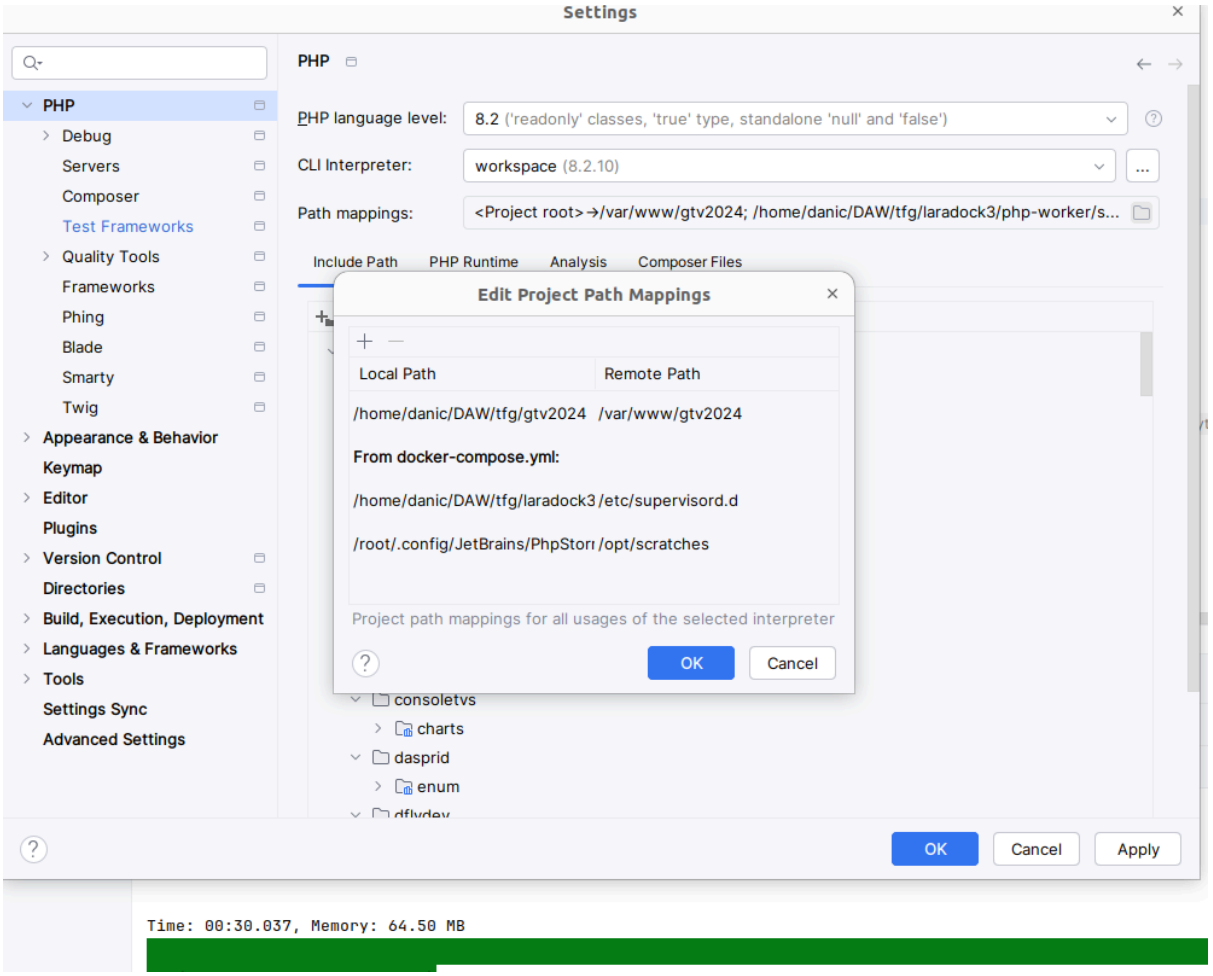


Dar al boton de “+” y elegir la opción PHPUnit By Remote Interpreter y tenemos que elegir el interprete que hemos creado anteriormente.



Una vez creada seleccionar User Composer autoloader y asegurar de que esté configurado para usar el intérprete Docker configurado previamente. En Path to script ponemos que apunte correctamente al archivo vendor/autoload.php

Para asegurar que el mapeo de la ruta es correcto nos vamos a file->Settings->PHP y en Path mappings le damos a la carpeta y podremos definir la ruta local y la remota que apunta al proyecto en el workspace.



Finalmente dando click derecho en la carpeta test del proyecto y darle a ‘run tests (PhpUnit)’ ya se lanzarán todos los tests, se puede también dar click derecho a la clase de test que se quiera y lanzar solo esos.

Base de datos

Factories

OldPointsOfInteresFactory

```
$factory->define(PointOfInterest::class, function (Faker $faker) {
    return [
        'name' => $faker->city,
        'distance' => $faker->randomNumber(2),
        'latitude' => $this->faker->latitude(36, 43),
        'longitude' => $this->faker->longitude(-9, 3),
        'creator'=>
$faker->randomElement(User::all()->pluck('id')->toArray()),
        'updater' =>
$faker->randomElement(User::all()->pluck('id')->toArray()),

'place_id'=>$faker->randomElement(Place::all()->pluck('id')->toArr
ay()),
    ];
});
```

OldVisitFactory.php

```
$factory->define(Visit::class, function (Faker $faker) {
    return [
        'hour' => $faker->dateTime(),
        'deviceid' => $faker->uuid,
        'appversion' => $faker->numberBetween(1, 10),
        'useragent' => $faker->word,
        'ssoo' => $faker->word,
        'ssooversion' => $faker->numberBetween(1, 10),
        'latitude' => $faker->latitude,
        'longitude' => $faker->longitude,
        'point_of_interest_id' =>
$faker->randomElement(PointOfInterest::all()->pluck('id')->toArray
())
    ];
});
```

PhotographyFactory

```
$factory->define(Photography::class, function (Faker $faker) {
    return [
        'route' =>
'https://images.pexels.com/photos/303383/pexels-photo-303383.jpeg?
auto=compress&cs=tinysrgb&w=600',
        'order' => $faker->randomDigit,
        'point_of_interest_id' =>
$faker->randomElement(PointOfInterest::all()->pluck('id')->toArray
()),
        'thematic_area_id' =>
$faker->randomElement(ThematicArea::all()->pluck('id')->toArray())
,
        'creator' =>
$faker->randomElement(User::all()->pluck('id')->toArray()),
        'updater' => null,
        'updated_at' => null,
```

```
    ];
});
```

PlaceFactory

```
$factory->define(Place::class, function (Faker $faker) {
    return [
        'name'=> $faker->city,
        'description' => $faker->sentence(5),
        'place_id'=>
$faker->randomElement(Place::all()->pluck('id')->toArray()),
        'creator' =>
$faker->randomElement(User::all()->pluck('id')->toArray()),
        'updater' =>
$faker->randomElement(User::all()->pluck('id')->toArray()),
    ];
});
```

PointOfInteresFactory

```
    public function definition()
    {
        return [
            'qr' => Str::random(35),
            'distance' => $this->faker->randomNumber(2),
            'latitude' => $this->faker->latitude(3, 20),
```

```
            'longitude' => $this->faker->longitude(3, 20),
            'creator'=>
$this->faker->randomElement(User::all()->pluck('id')->toArray()),
            'updater' =>
$this->faker->randomElement(User::all()->pluck('id')->toArray()),

'place_id'=>$this->faker->randomElement(Place::all()->pluck('id')-
>toArray()),
        ];
    }
}
```

RoleFactorty

```
    public function definition()
    {
        return [
            'name' => $this->faker->name(),
            'guard_name' => $this->faker->name(),
        ];
    }
}
```

ThematicAreaFactory

```
$factory->define(\App\Models\ThematicArea::class, function (Faker
$faker) {
    return [
        'name'=> $faker->word,
        'description'=> $faker->sentence(2),
        'updated_at' => null,
    ];
});
```

UserFactory

```
class UserFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = User::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'name' => $this->faker->name(),
            'email' => $this->faker->unique()->safeEmail(),
            'email_verified_at' => null,
            'password' =>
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', //
password
```

```
            'remember_token' => Str::random(10),
        ];
    }

    /**
     * Indicate that the model's email address should be
unverified.
     *
     * @return \Illuminate\Database\Eloquent\Factories\Factory
     */
    public function unverified()
    {
        return $this->state(function (array $attributes) {
            return [
                'email_verified_at' => null,
            ];
        });
    }

    /**
     * Indicate that the user should have a personal team.
     *
     * @return $this
     */
    public function withPersonalTeam()
    {
        if (! Features::hasTeamFeatures()) {
            return $this->state([]);
        }

        return $this->has(
            Team::factory()
                ->state(function (array $attributes, User $user) {
                    return ['name' => $user->name.'\'s Team',
'user_id' => $user->id, 'personal_team' => true];
                }),
            'ownedTeams'
        );
    }
}
```

VideoFactory

```
$factory->define(Video::class, function (Faker $faker) {
    $pointOfInterest =
$faker->randomElement(PointOfInterest::all()->pluck('id')->toArray
());
    $thematicAreas =
PointOfInterest::find($pointOfInterest)->thematicAreas->pluck('id'
)->toArray();
    $description = $faker->sentence(5);

    $foundVideos = PointOfInterest::find($pointOfInterest)->videos;

    \count($foundVideos) > 0
        ? $order = \count($foundVideos) + 1
        : $order = 1;

    return [
        'route' => 'videos/' . $faker->uuid() . '.mp4',
        'point_of_interest_id' => $pointOfInterest,
        'order' => $order,
        'creator' =>
$faker->randomElement(User::all()->pluck('id')->toArray()),
        'updater' => null,
```

```
        'thematic_area_id' =>
$faker->randomElement($thematicAreas),
        'description' => $description,
    ];
});
```

VideoItemFactory

```
$factory->define(VideoItem::class, function (Faker $faker) {
    $qualities = array('360p', '480p', '720p', '1080p', '4K');
    $formats = array('avi', 'mp4', 'ogg');
    $orientations = array('horizontal', 'vertical');
    return [
        'quality' => $faker->randomElement($qualities),
        'format' => $faker->randomElement($formats),
        'orientation' => $faker->randomElement($orientations),
    ];
});
```

VisitFactory

```
    public function definition()
```

```
    {
        return [
            'hour' => $this->faker->dateTime(),
            'deviceid' => $this->faker->uuid,
            'appversion' => $this->faker->numberBetween(1, 10),
            'useragent' => $this->faker->word,
            'ssoo' => $this->faker->word,
            'ssooversion' => $this->faker->numberBetween(1, 10),
            'latitude' => $this->faker->latitude,
            'longitude' => $this->faker->longitude,
            'point_of_interest_id' =>
$this->faker->randomElement(PointOfInterest::all()->pluck('id')->t
oArray())
        ];
    }
}
```

Migraciones

CreateFailedJobsTable

```
public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
```

```
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
}
```

CreateThematicAreasTable

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

            $table->string('name',45)->nullable();
            $table->string('description',245)->nullable();

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

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('thematic_areas');
    }
}
```

CreatePlacesTable

```
public function up()
    {
        Schema::create('places', function (Blueprint $table) {
            $table->id();
            $table->string('name', 30);
            $table->string('description', 2000);

$table->foreignId('place_id')->nullable()->references('id')->on('p
laces')->onDelete('cascade');

$table->foreignId('creator')->references('id')->on('users')->onDel
ete('cascade');

$table->foreignId('updater')->nullable()->references('id')->on('us
ers')->onDelete('cascade');

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

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('places');
    }
}
```

CreatePointOfInterestsTable

```
public function up()
    {
        Schema::create('point_of_interests', function (Blueprint
$table) {
            $table->id();
            $table->string('name',45);
            $table->integer('distance')->nullable();
            $table->decimal('latitude',10,8)->nullable();
            $table->decimal('longitude',11,8)->nullable();

$table->foreignId('place_id')->references('id')->on('places')->onD
elete('cascade');

$table->foreignId('creator')->references('id')->on('users')->onDel
ete('cascade');

$table->foreignId('updater')->nullable()->references('id')->on('us
ers')->onDelete('cascade');
            $table->date('last_update_date')->nullable();
            $table->date('creation_date')->nullable();

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

    /**
     * Reverse the migrations.
     *
     * @return void
```

```
     */
    public function down()
    {
        Schema::dropIfExists('point_of_interests');
    }
}
```

CreatePointOfInterestThematicAreaTable

```
public function up()
{
    Schema::create('point_of_interest_thematic_area', function
(Blueprint $table) {

        $table->string('title',145);
        $table->string('description',2000)->nullable();
        $table->integer('code_id')->nullable();

        $table->foreignId('thematic_area_id')->references('id')
            ->on('thematic_areas')->onDelete('cascade');

        $table->foreignId('point_of_interest_id')->references('id')
            ->on('point_of_interests')->onDelete('cascade');
    });
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
```

```
    Schema::dropIfExists('point_of_interest_thematic_area');
}
```

CreatePhotographiesTable

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

            $table->string('route', 245);
            $table->integer('order');


$table->foreignId('point_of_interest_id')->nullable()->references(
'id')

->on('point_of_interests')->onDelete('cascade')->onDelete('set
null');


$table->foreignId('thematic_area_id')->nullable()->references('id'
)
                ->on('thematic_areas')->onDelete('set null');

            $table->foreignId('creator')->references('id')
                ->on('users')->onDelete('cascade');


$table->foreignId('updater')->nullable()->references('id')
                ->on('users')->onDelete('cascade');
```

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

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('photographies');
    }
}
```

CreateVisitsTable

```
public function up()
    {
        Schema::create('visits', function (Blueprint $table) {
            $table->id();
            $table->dateTime('hour');
            $table->string('deviceid',85);
            $table->string('appversion',45);
            $table->string('useragent',95);
            $table->string('ssoo',45);
            $table->string('ssooversion',45);
            $table->decimal('latitude',10,8);
            $table->decimal('longitude',11,8);
```

```
$table->foreignId('point_of_interest_id')->nullable()->references(
'id')->on('point_of_interests')->onDelete('cascade');

            $table->timestamps();
            $table->softDeletes();

        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('visits');

    }
}
```

CreateVideosTable

```
public function up()
    {
        Schema::create('videos', function (Blueprint $table) {
            $table->id();
            $table->string('route', 245);

$table->foreignId('point_of_interest_id')->nullable()->references(
'id')->on('point_of_interests')->onDelete('cascade');
            $table->integer('order');
```

```
$table->foreignId('creator')->references('id')->on('users')->onDel
ete('cascade');

$table->foreignId('updater')->nullable()->references('id')->on('us
ers')->onDelete('cascade');

$table->foreignId('thematic_area_id')->nullable()->references('id'
)->on('thematic_areas')->onDelete('set null');
            $table->string('description', 2000);

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

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('videos');
    }
}
```

CreateVideoItemsTable

```
public function up()
    {
        Schema::create('video_items', function (Blueprint $table) {
```

```
            $table->id();

$table->foreignId('video_id')->references('id')->on('videos')->onD
elete('cascade');
            $table->string('quality', 45)->nullable();
            $table->string('format', 45)->nullable();
            $table->string('orientation', 45)->nullable();

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

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('video_items');
    }
}
```

CreatePermissionTable

```
public function up()
    {
        $tableNames = config('permission.table_names');
        $columnNames = config('permission.column_names');

        Schema::create($tableNames['permissions'], function
(Blueprint $table) {
```

```
            $table->bigIncrements('id');
            $table->string('name');
            $table->string('guard_name');
            $table->timestamps();
        });

        Schema::create($tableNames['roles'], function (Blueprint
$table) {
            $table->bigIncrements('id');
            $table->string('name');
            $table->string('display_name')->nullable();
            $table->string('guard_name');
            $table->timestamps();
        });

        Schema::create($tableNames['model_has_permissions'],
function (Blueprint $table) use ($tableNames, $columnNames) {
            $table->unsignedBigInteger('permission_id');

            $table->string('model_type');

$table->unsignedBigInteger($columnNames['model_morph_key']);
            $table->index([$columnNames['model_morph_key'],
'model_type'], 'model_has_permissions_model_id_model_type_index');

            $table->foreign('permission_id')
                ->references('id')
                ->on($tableNames['permissions'])
                ->onDelete('cascade');

            $table->primary(['permission_id',
$columnNames['model_morph_key'], 'model_type'],

'model_has_permissions_permission_model_type_primary');
        });

        Schema::create($tableNames['model_has_roles'], function
(Blueprint $table) use ($tableNames, $columnNames) {
            $table->unsignedBigInteger('role_id');

            $table->string('model_type');
```

```
$table->unsignedBigInteger($columnNames['model_morph_key']);
            $table->index([$columnNames['model_morph_key'],
'model_type'], 'model_has_roles_model_id_model_type_index');

            $table->foreign('role_id')
                ->references('id')
                ->on($tableNames['roles'])
                ->onDelete('cascade');

            $table->primary(['role_id',
$columnNames['model_morph_key'], 'model_type'],
                    'model_has_roles_role_model_type_primary');
        });

        Schema::create($tableNames['role_has_permissions'],
function (Blueprint $table) use ($tableNames) {
            $table->unsignedBigInteger('permission_id');
            $table->unsignedBigInteger('role_id');

            $table->foreign('permission_id')
                ->references('id')
                ->on($tableNames['permissions'])
                ->onDelete('cascade');

            $table->foreign('role_id')
                ->references('id')
                ->on($tableNames['roles'])
                ->onDelete('cascade');

            $table->primary(['permission_id', 'role_id'],
'role_has_permissions_permission_id_role_id_primary');
        });

        app('cache')
            ->store(config('permission.cache.store') != 'default' ?
config('permission.cache.store') : null)
            ->forget(config('permission.cache.key'));
    }

    /**
```

```
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        $tableNames = config('permission.table_names');

        Schema::drop($tableNames['role_has_permissions']);
        Schema::drop($tableNames['model_has_roles']);
        Schema::drop($tableNames['model_has_permissions']);
        Schema::drop($tableNames['roles']);
        Schema::drop($tableNames['permissions']);
    }
}
```

CreateThematicAreaUserTable

```
public function up()
    {
        Schema::create('thematic_area_user', function (Blueprint
$table) {
```

```
            $table->primary(['thematic_area_id', 'user_id',
'date']);

            $table->date('date');
            $table->boolean('active');

            $table->foreignId('thematic_area_id')->references('id')
                ->on('thematic_areas')->onDelete('cascade');

            $table->foreignId('user_id')->references('id')
                ->on('users')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('thematic_area_user');
    }
}
```

CreateSessionTable

```
public function up()
```

```
    {
        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->text('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('sessions');
    }
};
```

AddPreviousRoleToUsersTable

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

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

```
    }
};
```

KeepEmailForAdminTable

```
public function up()
    {
        Schema::create('email_for_admin', function (Blueprint
$table) {
            $table->id();
            $table->text('body')->comment('Cuerpo del mensaje');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('email_for_admin');
    }
};
```

AddFromInEmailForAdmin

```
    public function up()
```

```
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
            $table->string('from')->comment('Destinatario del
correo');
        });
    }

    public function down()
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
            $table->dropColumn('from');
        });
    }
};
```

AddSubjectFromEmailForAdmin

```
    public function up()
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
            $table->string('subject')->comment('peticion');
        });
    }

    public function down()
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
            $table->dropColumn('subject');
        });
```

```
    }
};
```

AddRequestedTeacherRoleRoleToUserTable

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

$table->boolean('requested_teacher_role')->default(false);
        });
    }

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

AddUserIdToEmailFromAdmin

```
    public function up()
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
```

```
$table->foreignId('user_id')->constrained()->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('email_for_admin', function (Blueprint
$table) {
            $table->dropForeign(['user_id']);
            $table->dropColumn('user_id');
        });
    }
}
```

Añadimos una nueva columna a la tabla usuarios para que guarde el rol anterior al cambio de usuario gtvisor.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
```

```
            $table->string('previous_role')->nullable();
        });
    }

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

Seeders

DatabaseSeeder

```
public function run()
    {
        Storage::disk('public')->deleteDirectory('videos');
        Storage::disk('public')->makeDirectory('videos');

        $this->call(ThematicAreaSeeder::class);
        $this->call(UserSeeder::class);
        $this->call(PlaceSeeder::class);
        $this->call(PointOfInterestSeeder::class);
        $this->call(PhotographySeeder::class);
        $this->call(VisitSeeder::class);
        $this->call(VideoSeeder::class);
        $this->call(VideoItemSeeder::class);
```

```
    }
}
```

PhotographySeeder

```
public function run()
    {
        Storage::disk('public')->deleteDirectory('photos');
        Storage::disk('public')->makeDirectory('photos');

        factory(Photography::class, 30)->create();
    }
}
```

PlaceSeeder

```
 public function run()
    {
        factory(Place::class,10)->create();
    }
```

PointOfInterestSeeder

```
public function run()
    {
        $pointsOfInterest = factory(PointOfInterest::class,
20)->make();
        $pointsOfInterest->each(function($pointOfInterest) {
            $faker = \Faker\Factory::create();
            $pointOfInterest->save();
            $thematicAreas=
ThematicArea::all()->pluck('id')->toArray();

$pointOfInterest->thematicAreas()->attach(Arr::random($thematicAre
as, 2),
                [
                    'title' => $faker->sentence,
                    'description' => $faker->text,
                ]);
        });
    }
}
```

ThematicAreaSeeder

```
public function run()
{
    factory(ThematicArea::class,10)->create();
}
```

UserSeeder

```
public function run()
{
    Storage::disk('public')->deleteDirectory('user-avatars');

    $adminRole = Role::create(['name' => 'Administrador']);
    $teacherRole = Role::create(['name' => 'Profesor']);
    $studentRole = Role::create(['name' => 'Alumno']);
    $gtvisorRole = Role::create(['name' => 'GTVisor']);

    $admin = User::create([
        'name' => 'Admin',
        'email' => 'admin@mail.com',
        'password' =>
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', //
password
    ]);
    $admin->assignRole($adminRole);

    $teacher = User::create([
        'name' => 'Teacher',
        'email' => 'teacher@mail.com',
        'password' =>
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', //
password
    ]);
    $teacher->assignRole($teacherRole);

    $student = User::create([
        'name' => 'Student',
        'email' => 'student@mail.com',
        'password' =>
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', //
password
    ]);
    $student->assignRole($studentRole);

    $gtvisor = User::create([
        'name' => 'GTVisor',
```

```
        'email' => 'gtvisor@mail.com',
        'password' =>
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', //
password
    ]);
    $gtvisor->assignRole($gtvisorRole);
}
```

VideoSeeder

```
public function run()
{
    Storage::disk('public')->deleteDirectory('videos');

    factory(Video::class,10)->create()->each(function (Video
$video) {
        factory(VideoItem::class)->create([
            'video_id' => $video->id,
        ]);
    });
}
```

VisitSeeder

```
public function run()
{
    factory(Visit::class, 30)->create();
}
```

Vistas

Responsive

Codigo utilizado para hacer la aplicacion responsive.

Botón:

```
<button id="menu-toggle" class="md:hidden flex items-center
text-white focus:outline-none">
    <svg class="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">
        <path class="inline-flex" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4
18h16"></path>
    </svg>
</button>
```

Lógica que sigue el botón:

```
document.addEventListener('DOMContentLoaded', function() {
    var menu = document.getElementById('mobile-menu');
    var logo = document.getElementById('logo');
    var profilePhoto = document.getElementById('profile-photo');
```

```
    var menuToggle = document.getElementById('menu-toggle');

    function updateLayout() {
        if (window.innerWidth < 1024) {
            menuToggle.classList.remove('hidden');
            menu.classList.add('hidden');
            logo.classList.add('hidden');
            profilePhoto.classList.add('hidden');
        } else {
            menuToggle.classList.add('hidden');
            logo.classList.remove('hidden');
        public function run()
     {
          $pointsOfInterest = factory(PointOfInterest::class,
20)->make();
          $pointsOfInterest->each(function($pointOfInterest) {
              $faker = \Faker\Factory::create();
              $pointOfInterest->save();
              $thematicAreas=
ThematicArea::all()->pluck('id')->toArray();

$pointOfInterest->thematicAreas()->attach(Arr::random($thematicAre
as, 2),
                  [
                      'title' => $faker->sentence,
                      'description' => $faker->text,
                  ]);
          });
     }
}    profilePhoto.classList.remove('hidden');
            menu.classList.remove('hidden');
        }
    }

    updateLayout();

    window.addEventListener('resize', updateLayout);

    menuToggle.addEventListener('click', function () {
        menu.classList.toggle('hidden');
        if (menu.classList.contains('hidden')) {
            logo.classList.remove('hidden');
            profilePhoto.classList.remove('hidden');
```

```
        } else {
            logo.classList.add('hidden');
            profilePhoto.classList.add('hidden');
        }
    });
});
```

Cambio de usuario a GTVisor

Cambios/Arreglos varios a la aplicación

Registro de Usuarios

Habilitar el registro de un usuario cuyo rol por defecto será el de alumno

```
public function create(array $input)
{
    Validator::make($input, [
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => $this->passwordRules(),
        'terms' => Jetstream::hasTermsAndPrivacyPolicyFeature() ?
['accepted', 'required'] : '',
    ])->validate();

    $user = User::create([
        'name' => $input['name'],
        'email' => $input['email'],
        'password' => Hash::make($input['password']),
    ]);

    // Asignar el rol de "student" al usuario
    $user->syncRoles(['Alumno']);

    return $user;
}
```

En la vista login.blade.php añadimos el link para registrar al usuario si este no tiene cuenta

```
<div class="block mt-4 flex items-center">
   <span class="text-sm text-gray-600">{{ __('You are not logged in')
}}</span>
   <a href="{{ route('register') }}" class="ml-2 text-sm text-blue-600
underline">{{ __('Register') }}</a>
</div>
```

Dependencias faltantes

Añadidas dependencias faltantes para los iconos y las alertas

```
"devDependencies": {
   "@fortawesome/fontawesome-free": "^6.5.2",
   "@tailwindcss/forms": "^0.4.0",
   "@tailwindcss/typography": "^0.5.0",
   "alpinejs": "^3.0.6",
   "axios": "^0.21",
   "laravel-mix": "^6.0.41",
   "less": "^4.1.2",
   "less-loader": "^10.2.0",
   "lodash": "^4.17.19",
   "postcss": "^8.1.14",
   "postcss-import": "^14.0.1",
   "resolve-url-loader": "^5.0.0",
   "sass": "^1.49.8",
   "sass-loader": "^12.6.0",
   "sweetalert2": "^11.11.0",
   "tailwindcss": "^3.0.0"
},
```