

Projecte GLOBAL: VideosApp

2n SPRINT



Professor: Jordi Vega

Assignatures: M7, M8 i M9

Nom: Manuel Steven

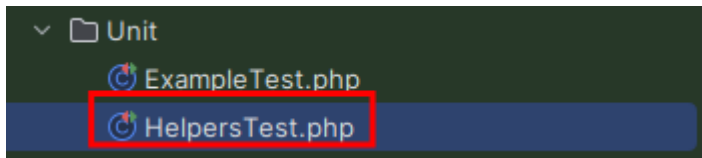
Cognoms: Romero Vargas

Índex

Corregir els errors del 1r sprint.	3
Descomentari linies de db_connection i db_database	5
Crear la migració de videos	5
Crear controlador de Videos	6
Crear model de Videos	7
Crear helper de videos	10
Afegir els seeders	11
Crear Layout de videos	12
Crear la ruta del show	14
Crearem la vista del show	14
Afegir test de creació de videos	16
Crear VideosTest al directori tests/Unit	17
Crear VideosTest al directori tests/Feature	20
Afegir guia del projecte	22
Instal·lació i execució Larastan	23
Correcció dels errors trobats	24
Error 1	24
Error 2	24
Error 3	25
Error 4	26
Error 5	26
Error 6	27
Error 6	27
Error 8	28
Error 9	29
Error 10	31
Error 11	32
Errors 12 i 13	33
Errors finals	34

Corregir els errors del 1r sprint.

El primer error que tenia era que el test **HelpersTest** no es trobava en el directori **Unit**, en el meu cas el tenia en el directori **Features**, així que com a primer pas l'he mogut al directori **Unit**.



Un altre petit error que podem trobar en el nostre test, es que no testejava suficients camps, així alguns més com el `current_team_id`, la contrasenya, el `user_id`, el personal team.

```
public function test_can_create_default_user()
{
    // Llamamos al helper para crear el usuario por defecto
    $user = UserHelper::createDefaultUser();

    // Verificamos que el usuario se haya creado en la base de datos
    $this->assertDatabaseHas( table: 'users', [
        'name' => config( key: 'users.default_user.name'),
        'email' => config( key: 'users.default_user.email'),
        'current_team_id' => $user->current_team_id, // Usamos el equipo asignado dinámicamente
    ]);

    // Verificamos que la contraseña almacenada sea válida
    $this->assertTrue(Hash::check(
        config( key: 'users.default_user.password'),
        $user->password
    ));

    // Verificamos que el equipo personal se haya creado correctamente
    $this->assertDatabaseHas( table: 'teams', [
        'name' => $user->name . "'s Team",
        'user_id' => $user->id,
        'personal_team' => true,
    ]);
}
```

Seguirem el mateix procediment en el test del usuari professor.

```
Steven *
public function test_can_create_default_teacher()
{
    // Llamamos al helper para crear el profesor por defecto
    $teacher = UserHelper::createDefaultTeacher();

    // Verificamos que el profesor se haya creado en la base de datos
    $this->assertDatabaseHas( table: 'users', [
        'name' => config( key: 'users.default_teacher.name'),
        'email' => config( key: 'users.default_teacher.email'),
        'current_team_id' => $teacher->current_team_id, // Usamos el equipo asignado dinámicamente
    ]);

    // Verificamos que la contraseña almacenada sea válida
    $this->assertTrue(Hash::check(
        config( key: 'users.default_teacher.password'),
        $teacher->password
    ));

    // Verificamos que el equipo personal se haya creado correctamente
    $this->assertDatabaseHas( table: 'teams', [
        'name' => $teacher->name . "'s Team",
        'user_id' => $teacher->id,
        'personal_team' => true,
    ]);
}
```

Afegirem el `current_team_id` al `config/users.php` per a que ens agafi la verificació correctament.

```
php users.php x
1 <?php
2
3 return [
4     'default_user' => [
5         'name' => env( key: 'DEFAULT_USER_NAME', default: 'Robin'),
6         'email' => env( key: 'DEFAULT_USER_EMAIL', default: 'robin@iesebre.com'),
7         'password' => env( key: 'DEFAULT_USER_PASSWORD', default: 'RobinHood'),
8         'current_team_id' => env( key: 'DEFAULT_USER_CURRENT_TEAM_ID', default: 1),
9     ],
10    'default_teacher' => [
11        'name' => env( key: 'DEFAULT_TEACHER_NAME', default: 'Batman'),
12        'email' => env( key: 'DEFAULT_TEACHER_EMAIL', default: 'batman@iesebre.com'),
13        'password' => env( key: 'DEFAULT_TEACHER_PASSWORD', default: 'TheDarkKnight'),
14        'current_team_id' => env( key: 'DEFAULT_TEACHER_CURRENT_TEAM_ID', default: 2),
15    ],
16 ];
```

Descomentari línies de db_connection i db_database

Ens haurem de dirigir al fitxer **phpunit.xml** i descomentar les següents línies per tal de que els testos utilitzi una base de dades temporal i no afecti a la base de dades real.

```
<env name="DB_CONNECTION" value="sqlite"/>
<env name="DB_DATABASE" value=":memory:"/>
```

Crear la migració de videos

Utilitzarem l'ordre **php artisan make:migration** per a crear una migració per a la nostra taula de videos.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan make:migration create_videos_table --create=videos
INFO Migration [database/migrations/2025_01_16_145657_create_videos_table.php] created successfully.
```

Un cop creada la migració haurem d'establir els diferents camps demanats en l'enunciat en el format corresponent. (series_id per ara no es una clau foranea ja que la taula de series la crearem en el següent sprint)

```
public function up()
{
    Schema::create( table: 'videos', function (Blueprint $table) {
        $table->id(); // id autoincremental
        $table->string( column: 'title'); // titulo del video
        $table->text( column: 'description'); // descripción, opcional
        $table->string( column: 'url'); // URL del video
        $table->timestamp( column: 'published_at')->nullable(); // fecha de publicación, opcional
        $table->string( column: 'previous')->nullable(); // referencia al video anterior
        $table->string( column: 'next')->nullable(); // referencia al video siguiente
        $table->unsignedBigInteger( column: 'series_id'); // referencia a la serie, sin clave foránea
        $table->timestamps(); // created_at y updated_at

        // Clave foránea será agregada más adelante
        // $table->foreign('series_id')->references('id')->on('series')->cascadeOnDelete();
    });
}
```

Crear controlador de Videos

Per a crear el controlador **VideosController** utilitzarem l'ordre **php artisan make:controller**.

Controlador de videos, VideosController amb les funcions testedBy i el show.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan make:controller VideosController  
  
INFO Controller [app/Http/Controllers/VideosController.php] created successfully.
```

Dins del controlador crearem la funció de show que ens trobarà el video segons la seva id, y en cas de que la trobe la mostrarà en l'arxiu **show.blade.php** que es trobarà en el directori **videos** dins de **resources/views**, si el video no existeix retorna un error 404.

```
public function show($id)  
{  
    // Busca el vídeo per ID  
    $video = Video::find($id);  
  
    // Si el vídeo no existeix, retorna un error 404  
    if (!$video) {  
        abort( code: 404, message: 'Video not found');  
    }  
  
    // Retorna la vista del vídeo amb la informació  
    return view( view: 'videos.show', compact( var_name: 'video') );  
}
```

En el tested by verifico si el video existeix també, sino existeix retorno a l'usuari a la pàgina 404.

```
public function testedBy($id)  
{  
    // Busca el video por ID  
    $video = Video::find($id);  
  
    // Si el video no existe, devuelve un error 404  
    if (!$video) {  
        return response()->json(['error' => 'Video not found'], status: 404);  
    }  
}
```

Crear model de Videos

Crearem el model de videos en l'ordre **php artisan make:model**

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan make:model Video
```

```
INFO Model [app/Models/Video.php] created successfully.
```

En el model primer que tot afegirem els camps que podes ser assignats.

```
class Video extends Model
{
    use HasFactory;

    // Campos que pueden ser asignados de forma masiva
    no usages
    protected $fillable = [
        'title',          // Título del video
        'description',     // Descripción del video
        'url',             // URL del video
        'previous',
        'next',
        'series_id', // Asegúrate de incluir este campo
        'published_at',   // Fecha de publicación
    ];
}
```

En el camp de `published_at` crearem una funció **`getFormattedPublishedAtAttribute()`** que ens retornara la data formatejada, de manera que el resultat serà. per exemple: "13 de gener de 2025"

```
no usages
protected $dates = ['published_at'];

/**
 * Devuelve la fecha publicada formateada (por ejemplo, "13 de enero de 2025").
 *
 * @return string|null
 */
no usages  ⬆ Steven
public function getFormattedPublishedAtAttribute(): ?string
{
    if ($this->published_at) {
        return Carbon::parse($this->published_at)->isoFormat('format: 'D [de] MMMM [de] YYYY');
    }
    return null;
}
```

Crearem també una funció anomenada **`getFormattedForHumansPublishedAtAttribute()`**, que ens retornara la data publicada en un format llegible, per exemple (fa 2 hores)

```
/**
 * Devuelve la fecha publicada en un formato legible (por ejemplo, "hace 2 horas").
 *
 * @return string|null
 */
no usages  ⬆ Steven
public function getFormattedForHumansPublishedAtAttribute(): ?string
{
    if ($this->published_at) {
        return Carbon::parse($this->published_at)->diffForHumans();
    }
    return null;
}
```


Per finalitzar crearem una funció anomenada **getPublishedAtTimestampAttribute()** que ens retorna el Unix timestamp de la data publicada.

```
/**
 * Devuelve el Unix timestamp de la fecha publicada.
 *
 * @return int|null
 */
no usages  👤 Steven
public function getPublishedAtTimestampAttribute(): ?int
{
    if ($this->published_at) {
        return Carbon::parse($this->published_at)->timestamp;
    }
    return null;
}
```

Crear helper de videos

En el directori **config** crearem un arxiu videos.php on tindrem la informació dels diferents videos que utilitzarem més endavant.

```
php videos.php x
1  <?php
2
3  return [
4      'video_1' => [
5          'title' => 'Gol de Barou Shouei',
6          'description' => 'El gol del rey del campo.',
7          'url' => 'https://www.youtube.com/embed/3PHjyf4EF9o',
8          'previous' => null,
9          'next' => null,
10         'series_id' => 1, // Valor manual predeterminado
11     ],
12     'video_2' => [
13         'title' => 'Gol de Nagi Seishirou',
14         'description' => 'El gol del genio del control.',
15         'url' => 'https://www.youtube.com/embed/qTsJNWgIgNk',
16         'previous' => null,
17         'next' => null,
18         'series_id' => 1, // Valor manual predeterminado
19     ],
20     'video_3' => [
21         'title' => '1r gol de Shidou Ryousei',
```

Crearem el VideoHelper y dins una funció per a crear els diferents videos, això ens ajudarà en el procés dels tests on verificarem que els videos es puguin crear correctament i en el seeder on tindrem els videos per defecte a crear en la nostra aplicació.

```
2 usages  Steven *
public static function createDefaultVideos()
{
    $videos = [];

    $videos[] = Video::create([
        'title' => config( key: 'videos.video_1.title'),
        'description' => config( key: 'videos.video_1.description'),
        'url' => config( key: 'videos.video_1.url'),
        'previous' => config( key: 'videos.video_1.previous'),
        'next' => config( key: 'videos.video_1.next'),
        'series_id' => 1, // Valor asignado manualmente
        'published_at' => now(), // Fecha actual
    ]);

    $videos[] = Video::create([
        'title' => config( key: 'videos.video_2.title'),
        'description' => config( key: 'videos.video_2.description'),
        'url' => config( key: 'videos.video_2.url'),
```

Afegir els seeders

En el projecte anterior en el **DatabaseSeeder.php** ja havíem fer la crida al helper que crea els usuaris per defecte, ara farem la crida a el helper de vídeos per a crear els vídeos predeterminats.

```
DatabaseSeeder.php x
1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Helpers\UserHelper;
6  use App\Helpers\VideoHelper;
7  use App\Models\User;
8  // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
9  use Illuminate\Database\Seeder;
10
11  class DatabaseSeeder extends Seeder
12  {
13      /**
14       * Seed the application's database.
15       */
16      public function run(): void
17      {
18          // User::factory(10)->withPersonalTeam()->create();
19
20          // Crear el usuario por defecto
21          UserHelper::createDefaultUser();
22
23          // Crear el profesor por defecto
24          UserHelper::createDefaultTeacher();
25
26          // Crear el video por defecto
27          VideoHelper::createDefaultVideo();
28      }
29  }
```

Crear Layout de videos

Crearem un component anomenat **VideosAppLayout** en l'ordre **php artisan make:component**.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan make:component VideosAppLayout

[INFO] Component [app/View/Components/VideosAppLayout.php] created successfully.

[INFO] View [resources/views/components/videos-app-layout.blade.php] created successfully.
```

Dins de **app/View/components** tindrem el layout que ens dirigira a la vista de **videos-app-layout** que es troba dins del directori **components**

```
© VideosAppLayout.php x
9      class VideosAppLayout extends Component
12          * Create a new component instance.
13          */
          no usages
14      public function __construct()
15      {
16          //
17      }
18
19      /**
20       * Get the view / contents that represent the component.
21       */
          no usages
22      public function render(): View|Closure|string
23      {
24          return view('components/videos-app-layout');
25      }
26  }
```

En el layout tindrem l'estructura bàsica de la nostra pàgina de videos, y fent us d'un slot, mostrarem en la vista show el contingut de cada video.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{{ $title ?? 'Videos App' }}</title>
  @vite('resources/css/app.css') {{-- Tailwind CSS --}}
</head>
<body class="flex flex-col min-h-screen bg-gray-300 text-gray-800 font-sans">

  <!-- Main Content -->
  <main class="flex-grow container mx-auto px-4 py-6">
    {{ $slot }}
  </main>
</body>
</html>
```

Crear la ruta del show

En **routes/web.php** haurem d'afegir la ruta al show de les vistes de videos, de manera que podrem anar visualitzant cada video segons la seva id.

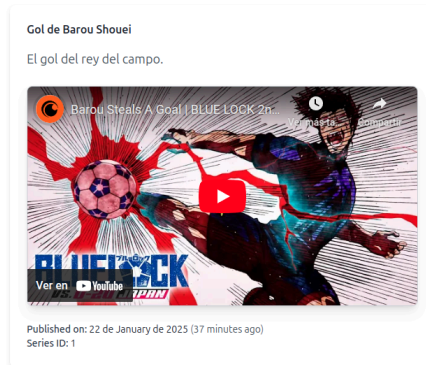
```
php web.php x
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\VideosController;
5
6
7 Route::get(uri: '/', function () {
8     return view(view: 'welcome');
9 });
10
11 Route::get(uri: 'videos/{id}', [VideosController::class, 'show'])->name(name: 'videos.show');
```

Crearem la vista del show

Crearem la vista del show de videos, de manera que podrem veure la informació del nostre video (títol, descripció, data de publicació formatada, series id) i el mateix video mitjançant un iframe. Per finalitzar afegirem uns estils bàsics en tailwindcss, tot i que no tindrà res especial ja que en els següents sprints ja ens centrarem en pulir els estils.

```
show.blade.php x
1 <x-videos-app-layout title="Video: {{ $video->title }}">
2     <div class="bg-white shadow-md rounded-lg p-6">
3         <h1 class="text-3xl font-semibold text-gray-800 mb-4">{{ $video->title }}</h1>
4         <p class="text-gray-600 text-lg mb-6">{{ $video->description }}</p>
5
6         <div class="flex justify-center mb-6">
7             <iframe
8                 src="{{ $video->url }}"
9                 class="rounded-lg shadow-lg border border-gray-300"
10                width="560"
11                height="315"
12                frameborder="0"
13                allow="autoplay; encrypted-media"
14                allowfullscreen>
15            </iframe>
16        </div>
17
18        <div class="text-sm text-gray-500">
19            <p>
20                <span class="font-medium text-gray-700">Published on:</span>
21                <span class="text-gray-800">{{ $video->formatted_published_at }}</span>
22                (<span class="text-gray-600">{{ $video->formatted_for_humans_published_at }}</span>)
23            </p>
24            <p>
25                <span class="font-medium text-gray-700">Series ID:</span>
26                <span class="text-gray-800">{{ $video->series_id }}</span>
27            </p>
28        </div>
29    </div>
30 </x-videos-app-layout>
31
```

Podrem veure com al crear un video, podem accedir correctament segons la seva id, i podem veure com en el iframe es mostra correctament.



Afegir test de creació de videos

Dins del nostre HelpersTest farem la crida a el helper de videos que hem creat anteriorment.

```
HelpersTest.php x
1  <?php
2
3  namespace Tests\Unit;
4
5  use App\Helpers\UserHelper;
6  use App\Helpers\VideoHelper;
7  use Illuminate\Foundation\Testing\RefreshDatabase;
8  use Illuminate\Support\Facades\Hash;
9  use Tests\TestCase;
10
```

I crearem un petit test que crearà un video y verificarà que es creen els diferents camps, com el títol, descripció, url, series id i la data de publicació.

```
public function test_can_create_default_video()
{
    // Llamamos al helper para crear los videos por defecto
    $videos = VideoHelper::createDefaultVideos();

    // Seleccionamos el primer video creado
    $video = $videos->first();

    // Verificamos que el video se haya creado correctamente en la base de datos
    $this->assertDatabaseHas('videos', [
        'title' => config('videos.video_1.title'),
        'description' => config('videos.video_1.description'),
        'url' => config('videos.video_1.url'),
        'series_id' => 1, // Valor esperado manualmente
    ]);

    // Verificamos el formato personalizado de la fecha publicada
    $formattedDate = $video->formatted_published_at;
    $this->assertNotNull($formattedDate);
    $this->assertEquals(now()->isoFormat('D [de] MMMM [de] YYYY'), $formattedDate);
}
```


Crear VideosTest al directori tests/Unit

Crearem un test dins de VideosTest que comprovi que la formatació dels videos és correcta, l'anomenarem `can_get_formatted_published_at_date()`.

```
Steven *
public function test_can_get_formatted_published_at_date()
{
    // Crear un video amb una data de publicació
    $video = Video::factory()->create([
        'published_at' => Carbon::parse('2025-01-17 14:00:00'),
    ]);

    // Comprovar que el format és correcte
    $this->assertEquals('17 de enero de 2025', $video->formatted_published_at);
}
```

Al executar el test en primera instancia podrem veure que ens retorna un error, i es que al haver ficat la data en espanyol, realment no coincideix en el format definit pel test.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan test --filter=VideosTest

FAIL Tests\Unit\VideosTest
x can get formatted published at date

FAILED Tests\Unit\VideosTest > can get formatted published at date
Failed asserting that two strings are equal.
- '17 de enero de 2025'
+ '17 de January de 2025'

at tests/Unit/VideosTest.php:24
20 |         'published_at' => Carbon::parse('2025-01-17 14:00:00'),
21 |     ];
22 |
23 |     // Comprovar que el format és correcte
→ 24 |     $this->assertEquals('17 de enero de 2025', $video->formatted_published_at);
25 | }
26 |
27 | // public function test_can_get_formatted_published_at_date()
28 | // {

1 tests/Unit/VideosTest.php:24

Tests: 1 failed (1 assertions)
Duration: 0.16s
```

Podriem simplement canviar el test a anglès, o també tenim l'opció de establir l'idioma de Carbon a espanyol.

```
Steven
public function test_can_get_formatted_published_at_date()
{
    // Configurar idioma a espanyol
    Carbon::setLocale(locale: 'es');

    // Crear un vídeo amb una data de publicació
    $video = Video::factory()->create([
        'published_at' => Carbon::parse(time: '2025-01-17 14:00:00'),
    ]);

    // Comprovar que el format és correcte
    $this->assertEquals(expected: '17 de enero de 2025', $video->formatted_published_at);
}
```

Podrem veure com realment el test es realitza sense problemes.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan test --filter=VideosTest

PASS Tests\Unit\VideosTest
✓ can get formatted published at date

Tests: 1 passed (1 assertions)
Duration: 0.16s
```

Crearem un test **test_can_get_formatted_published_at_date_when_not_published()** que provi que quan un vídeo no té data de publicació (`published_at` és `null`), el mètode **getFormattedPublishedAtAttribute** retorna `null`.

```
public function test_can_get_formatted_published_at_date_when_not_published()
{
    // Crear un vídeo sense data de publicació
    $video = Video::factory()->create([
        'published_at' => null,
    ]);

    // Comprovar que la propietat retorna null
    $this->assertNull($video->formatted_published_at);
}
```

Al model `Video`, ja hem manejat correctament el cas quan `published_at` és `null`, retornant `null`. Així que el test ja hauria de passar amb el codi actual.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan test --filter=VideosTest

PASS Tests\Unit\VideosTest
✓ can get formatted published at date when not published

Tests: 1 passed (1 assertions)
Duration: 0.16s
```

Crear VideosTest al directori tests/Feature

Crearem un test anomenat **users_can_view_videos()** que verifica que un cop creat el video, les persones puguin visualitzar-lo.

El test fa una petició HTTP per accedir a un vídeo a través de la ruta. Comprovarem que la resposta té un estat 200 (OK) i que la vista es carrega correctament.

```
public function test_users_can_view_videos()
{
    $video = Video::factory()->create();

    $response = $this->get(route( name: 'videos.show', $video->id));

    $response->assertStatus( status: 200);
    $response->assertViewIs( value: 'videos.show');
    $response->assertViewHas('video', $video);
}
```

El controlador **VideosController@show** ja està implementat correctament i torna la vista amb el vídeo. Per tant, el test hauria de passar sense problemes.

```
Calumnat@steven:~/Documents/VideosAppSteven$ php artisan test --filter=VideosTest

PASS Tests\Feature\VideosTest
✓ users can view videos

Tests:      1 passed (3 assertions)
Duration: 0.17s
```

També en un test que anomenarem **users_cannot_view_not_existing_videos()** que verificarà que si no hi han vídeos o, en cas de que hagues intentes accedir a un vídeo inexistent, ens apareixerà l'error 404

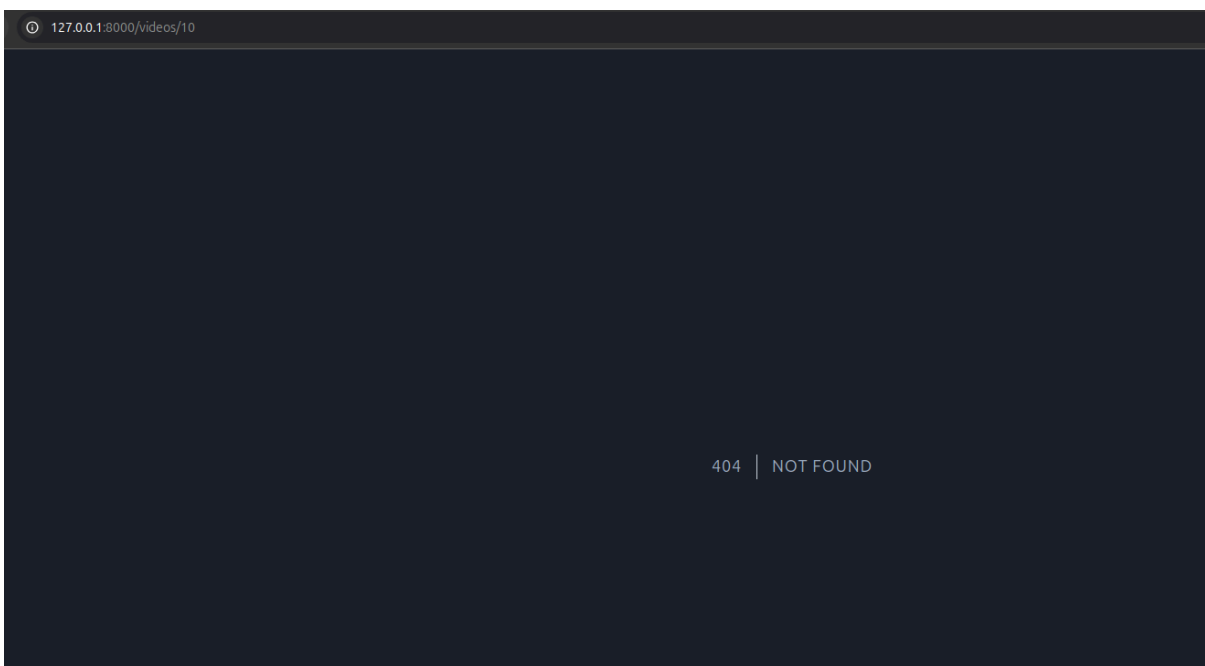
Intentem accedir a un vídeo amb un ID que no existeix (9999 en aquest cas).

```
Steven *  
public function test_users_cannot_view_not_existing_videos()  
{  
    $response = $this->get(route( name: 'videos.show', parameters: 9999));  
  
    $response->assertStatus( status: 404);  
}
```

I en el nostre cas el controlador **VideosController@show** ja torna un error 404 quan el vídeo no es troba a la base de dades. Aquest comportament ja està implementat, per la qual cosa el test hauria de passar sense problemes.

```
alumnat@steven:~/Documents/VideosAppSteven$ php artisan test --filter=VideosTest  
  
PASS Tests\Feature\VideosTest  
✓ users cannot view not existing videos  
  
Tests: 1 passed (1 assertions)  
Duration: 0.16s
```

Podrem veure com realment al accedir a una pàgina ens retorna el error **404 | NOT FOUND**



Afegir guia del projecte

Afegirem a **resources/markdown/terms** una petita guia sobre que tracta el projecte i que hem fet als dos sprints.

```
M+ terms.md x
1  # Guia del projecte VideosAppSteven
2
3  **Benvinguts al projecte VideosAppSteven**, una aplicació desenvolupada per gestionar i visualitzar videos utilitzant e
4
5  ---
6
7  ## **Objectiu del projecte**
8  VideosAppSteven és una plataforma que permet gestionar videos amb funcionalitats bàsiques com visualitzar la informació
9
10 ---
11
12 ## **Desenvolupament del 1r Sprint**
13
14 ### **1. Creació del projecte**
15 El projecte es va iniciar amb el nom VideosAppSteven, utilitzant les opcions següents:
16 - Jetstream amb Livewire.
17 - Suport per a teams.
18 - Base de dades en SQLite.
19 - Tests amb PHPUnit.
20
21 ### **2. Creació dels Helpers**
22 S'han creat helpers a la carpeta app per facilitar la gestió d'usuaris per defecte. També s'han definit credencials d
23
24 ### **3. Tests de Helpers
```

Instal·lació i execució Larastan

Per a instal·lar larastan en el nostre projecte executarem l'ordre **composer require --dev "larastan/larastan3.0"**

```
alumnat@steven:~/Documents/VideosAppSteven$ composer require --dev "larastan/larastan:^3.0"
./composer.json has been updated
Running composer update larastan/larastan
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

laravel/fortify ..... DONE
laravel/jetstream ..... DONE
laravel/pail ..... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
livewire/livewire ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE

83 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

Found 2 security vulnerability advisories affecting 2 packages.
Run "composer audit" for a full list of advisories.
```

Crearem un arxiu **phpstan.neon** que inclurà les extensions de **larastan** i **carbon**.

```
alumnat@steven:~/Documents/VideosAppSteven$ cat phpstan.neon
includes:
    - vendor/larastan/larastan/extension.neon
    - vendor/nesbot/carbon/extension.neon

parameters:
    paths:
        - app/
        - routes/
        - tests/
    # Level 10 is the highest level
    level: 5
```

Per a executar un anàlisi, executarem l'ordre **./vendor/bin/phpstan analyse**

```
alumnat@steven:~/Documents/VideosAppSteven$ ./vendor/bin/phpstan analyse
Note: Using configuration file /home/alumnat/Documents/VideosAppSteven/phpstan.neon.
24/24 [████████████████████] 100%
```

Correcció dels errors trobats

Error 1

Podrem veure com ens troba un error en el directori **Actions/Jetstream/DeleteUser.php**, en el qual troba una propietat indefinida en el projecte.

```
Line Actions/Jetstream/DeleteUser.php
40 Access to an undefined property App\Models\User::$ownedTeams.
    property.notFound
    Learn more: https://phpstan.org/blog/solving-phpstan-access-to-undefined-property
```

Haurem de definir-la manualment en un comentari, per a que larastan la trobi.

```
/**
 * @property-read \Illuminate\Database\Eloquent\Collection|\App\Models\Team[] $ownedTeams
 */
```

Error 2

En el segon error el trobem en el **Actions/Jetstream/InviteTeamMember.php**, error en el qual hi ha un parametre que no identifica o no troba correctament.

```
Line Actions/Jetstream/InviteTeamMember.php
37 Parameter #1 $invitation of class Laravel\Jetstream\Mail\TeamInvitation constructor expects Laravel\Jetstream\TeamInvitation, Illuminate\Database\Eloquent\Model given.
    argument.type
```

Per a solucionar-ho haurem de afegir un comentari per a que larastan trobe la variable **invitation**

```
© InviteTeamMember.php x
20 class InviteTeamMember implements InvitesTeamMembers
25     public function invite(User $user, Team $team, string $email, ?string $role
30
31         InvitingTeamMember::dispatch($team, $email, $role);
32
33         /** @var TeamInvitationModel $invitation */
34         $invitation = $team->teamInvitations()->create([
35             'email' => $email,
36             'role' => $role,
37         ]);
38
39         Mail::to($email)->send(new TeamInvitation($invitation));
40     }
41
```


Error 3

Ens trobarà un error a l'acció **RemoveTeamMember**, on ens dirà que ha trobat una propietat indefinida.

```
-----  
Line   Actions/Jetstream/RemoveTeamMember.php  
-----  
45     Access to an undefined property Illuminate\Database\Eloquent\Model::$id.  
       property.notFound  
       💡 Learn more: https://phpstan.org/blog/solving-phpstan-access-to-undefined-property  
-----
```

Per a solucionar aquest error haurèm de definir la propietat **id** (i altres necessaries) tant en el model de **user.php** com en el **team.php**

```
© User.php x  
1  <?php  
2  
3  namespace App\Models;  
4  
5  // use Illuminate\Contracts\Auth\MustVerifyEmail;  
6  > use ...  
13  
14  /**  
15   * @property-read \Illuminate\Database\Eloquent\Collection|\App\Models\Team[] $ownedTeams  
16   */  
17  
18  /**  
19   * @property int $id  
20   * @property int|null $current_team_id  
21   * @property-read \App\Models\Team|null $team  
22   * @property-read \Illuminate\Database\Eloquent\Collection|\App\Models\Team[] $ownedTeams  
23   */  
24
```

```
© Team.php x  
1  <?php  
2  
3  namespace App\Models;  
4  
5  > use ...  
10  
11  /**  
12   * @property int $id  
13   * @property int $user_id  
14   * @property bool $personal_team  
15   * @property-read \App\Models\User $owner  
16   * @property-read \Illuminate\Database\Eloquent\Collection|\App\Models\User[] $members  
17   */  
18
```

Error 4

En el model de Team.php podem veure com ens troba un error en el tipus de propietat definida en la línia 30.

```
Line Models/Team.php
30 PHPDoc type array<int, string> of property App\Models\Team::$fillable is not covariant with PHPDoc type list<string> of overridden property Illuminate\Database\Eloquent\Model::$fillable.
    property.phpDocType
    You can fix 3rd party PHPDoc types with stub files:
    https://phpstan.org/user-guide/stub-files
```

El que haurem de fer per a solucionar aquest error serà, canviar el comentari **array<int, string>** per **list<string>** per a que larastan o pugui llegir bé i sense detectar problemes.

```
class Team extends JetstreamTeam
{
    /** @use HasFactory<\Database\Factories\TeamFactory> */
    use HasFactory;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
```

Error 5

En aquest error també podem trobar el mateix cas anterior, de manera que seguirem el mateix procediment de ficar **list<string>** per a solucionar el error.

```
class User extends Authenticatable
{
    use HasApiTokens;

    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory;
    use HasProfilePhoto;
    use HasTeams;
    use Notifiable;
    use TwoFactorAuthenticatable;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
```

Error 6

En aquest error, literalment podrem trobar el mateix error en la mateixa solució que el anterior punt.

```
Line   Models/User.php
-----
54      PHPDoc type array<int, string> of property App\Models\User::$hidden is not covariant with PHPDoc type list<string> of overridden property
      Illuminate\Database\Eloquent\Model::$hidden.
      property.phpDocType
      You can fix 3rd party PHPDoc types with stub files:
      https://phpstan.org/user-guide/stub-files
```

```
/**
 * The attributes that should be hidden for serialization.
 *
 * @var list<string>
 */
protected $hidden = [
    'password',
    'remember_token',
    'two_factor_recovery_codes',
    'two_factor_secret',
];
```

Error 6

Un altre cop, el mateix error, en la mateixa solucio utilitzant list en comptes d'array.

```
Line   Models/User.php
-----
66      PHPDoc type array<int, string> of property App\Models\User::$appends is not covariant with PHPDoc type list<string> of overridden property
      Illuminate\Database\Eloquent\Model::$appends.
      property.phpDocType
      You can fix 3rd party PHPDoc types with stub files:
      https://phpstan.org/user-guide/stub-files
```

```
/**
 * The accessors to append to the model's array form.
 *
 * @var list<string>
 */
no usages
protected $appends = [
    'profile_photo_url',
];
```

Error 8

En aquest error podrem veure com tenim una propietat que en principi larastan la detecta com a que no existeix.

```
-----  
Line    Models/User.php  
-----  
67      Property 'profile_photo_url' does not exist in model.  
       rules.modelAppends  
-----
```

Per a solucionar aquest petit problema bastarà en actualitzar el composer.

```
alumnat@steven:~/Documents/VideosAppSteven$ composer update  
Loading composer repositories with package information  
Updating dependencies  
Lock file operations: 4 installs, 44 updates, 0 removals  
- Upgrading egulias/email-validator (4.0.2 => 4.0.3)  
- Upgrading laravel/fortify (v1.24.5 => v1.25.3)  
- Upgrading laravel/framework (v11.33.2 => v11.39.0)  
- Upgrading laravel/jetstream (v5.3.3 => v5.3.4)  
- Upgrading laravel/pint (v1.18.2 => v1.20.0)  
- Upgrading laravel/prompts (v0.3.2 => v0.3.3)  
- Upgrading laravel/sail (v1.38.0 => v1.40.0)  
- Upgrading laravel/sanctum (v4.0.4 => v4.0.7)  
- Upgrading laravel/serializable-closure (v2.0.0 => v2.0.1)  
- Upgrading league/commonmark (2.5.3 => 2.6.1)  
- Locking league/uri (7.5.1)  
- Locking league/uri-interfaces (7.5.0)  
- Upgrading livewire/livewire (v3.5.12 => v3.5.18)  
- Upgrading miledetest/miledetestlib (4.8.06 => 4.8.09)
```

Error 9

Ens trobarem en un error en el controlador de Videos al metode testedBy()

```
Line   app/Http/Controllers/VideosController.php
42     Method App\Http\Controllers\VideosController::testedBy() should return Illuminate\Http\JsonResponse but return statement is missing.
       return.missing
```

El problema es deu a que el metode **testedBy** està definit per a retornar un objecte de tipus **\Illuminate\Http\JsonResponse**, pero no té una instrucció **return** en el cas en que es trobi un video.

```
/**
 * Retorna una lista de testers asociados al video.
 *
 * @param int $id
 * @return \Illuminate\Http\JsonResponse
 */
no usages  👤 Steven *
public function testedBy($id)
{
    // Busca el video por ID
    $video = Video::find($id);

    // Si el video no existe, devuelve un error 404
    if (!$video) {
        return response()->json(['error' => 'Video not found'], status: 404);
    }
}
```

Per solucionar aquest problema, ens assegurarem que el mètode sempre retorni una resposta JSON, fins i tot en el cas que el vídeo existeixi.

```
public function testedBy($id)
{
    // Busca el video por ID
    $video = Video::find($id);

    // Si el video no existe, devuelve un error 404
    if (!$video) {
        return response()->json(['error' => 'Video not found'], status: 404);
    }

    // Devuelve una lista ficticia de testers asociados al video
    // (sustituir este arreglo por la lógica real que desees implementar)
    $testers = [
        ['id' => 1, 'name' => 'Tester 1', 'email' => 'tester1@example.com'],
        ['id' => 2, 'name' => 'Tester 2', 'email' => 'tester2@example.com'],
    ];

    // Devuelve los testers en formato JSON
    return response()->json(['testers' => $testers], status: 200);
}
```

Error 10

L'error que esteu veient és al **fitxer routes/console.php** està intentant utilitzar la variable **this** fora d'una classe. A Laravel, els fitxers a la carpeta routes no estan dins de classes, per la qual cosa \$this no pot ser usat directament.

```
-----  
Line    routes/console.php  
-----  
8        Undefined variable: $this  
        variable.undefined  
-----
```

Per solucionar aquest error, haurem d'usar **Artisan::command** d'una manera que no depengui de **\$this** per trucar a mètodes com a comment.

```
<?php  
  
use Illuminate\Foundation\Inspiring;  
use Illuminate\Support\Facades\Artisan;  
  
Artisan::command( signature: 'inspire', function () {  
    // Usar `info()` para mostrar el mensaje en la consola  
    info(Inspiring::quote());  
})->purpose( description: 'Display an inspiring quote')->hourly();
```

Error 11

En el test de videos podrem observar com no troba la propietat **id** definida correctament.

```
Line   tests/Feature/VideosTest.php
25     Access to an undefined property Illuminate\Database\Eloquent\Model::$id.
       property.notFound
       💡 Learn more: https://phpstan.org/blog/solving-phpstan-access-to-undefined-property
```

Per a solucionar aquest error anirem al model de videos i definirem totes les propietats que haguem creat, per evitar qualsevol tipus d'error.

```
/**
 * App\Models\Video
 *
 * @property int $id
 * @property string $title
 * @property string $description
 * @property string $url
 * @property string|null $previous
 * @property string|null $next
 * @property int $series_id
 * @property \Carbon\Carbon|null $published_at
 * @property \Carbon\Carbon|null $created_at
 * @property \Carbon\Carbon|null $updated_at
 */
```


Errors 12 i 13

Els errors que trobem amb el test de **ProfileInformationTest** es deuen al fet que **Livewire\Testable** ja no utilitza **\$state** com una propietat accessible directament.

```
Line   tests/Feature/ProfileInformationTest.php
-----
21     Access to an undefined property Livewire\Features\SupportTesting\Testable::$state.
       property.notFound
       Learn more: https://phpstan.org/blog/solving-phpstan-access-to-undefined-property
22     Access to an undefined property Livewire\Features\SupportTesting\Testable::$state.
       property.notFound
       Learn more: https://phpstan.org/blog/solving-phpstan-access-to-undefined-property
```

Per a solucionar-ho, en comptes d'accedir directament a **\$component->state**, utilitzare el mètode **get()** per obtenir l'estat actual del component.

```
public function test_current_profile_information_is_available(): void
{
    $this->actingAs($user = User::factory()->create());

    $component = Livewire::test( name: UpdateProfileInformationForm::class);

    $state = $component->get('state'); // Obté l'estat del component

    $this->assertEquals($user->name, $state['name']);
    $this->assertEquals($user->email, $state['email']);
}
```

Errors finals

Tot i que no es una solució ideal, 12 o 13 errors aproximadament del nostre projecte estaven relacionats amb Eloquent, i amb aquesta configuració, millorem la qualitat del codi sense sacrificar la velocitat de desenvolupament. Aquesta estratègia permet gestionar millor els errors dinàmics sense generar conflictes amb l'ús estàndard d'Eloquent a Laravel.

```
includes:
  - vendor/larastan/larastan/extension.neon
  - vendor/nesbot/carbon/extension.neon

parameters:
  paths:
    - app/
    - routes/
    - tests/
  # Level 10 is the highest level
  level: 5

  ignoreErrors:
    # Suppress errors related to accessing undefined properties on Eloquent models
    - '#Access to an undefined property Illuminate\\Database\\Eloquent\\Model::.*#'

    # Suppress errors related to undefined methods on Eloquent models
    - '#Call to an undefined method Illuminate\\Database\\Eloquent\\Model::.*#'

# ignoreErrors:
#   - '#PHPDoc tag @var#'
```

Un cop solucionats tots els errors en el nostre projecte podrem veure com s'executa l'anàlisi sense trobar cap error.

```
alumnat@steven:~/Documents/VideosAppSteven$ ./vendor/bin/phpstan analyse
Note: Using configuration file /home/alumnat/Documents/VideosAppSteven/phpstan.neon.
57/57 [████████████████████] 100%

[OK] No errors
```