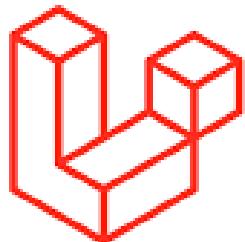


Projecte GLOBAL: VideosApp

4t SPRINT



Professor: Jordi Vega
Assignatures: M7, M8 i M9
Nom: Harvey
Cognoms: John Glover



INDEX

Corregir els errors del 3r sprint.	3
VideosManageController.php	4
VideosController.php	8
VideoHelper.php	9
Vistes per al CRUD	10
index.blade.php	10
create.blade.php	11
edit.blade.php	12
delete.blade.php	12
Vistes per als videos	13
resources/views/videos/index.blade.php	13
resources/views/videos/show.blade.php	14
Test user_with_permissions_can_manage_videos()	16
Permisos de vídeos per al CRUD	17
PermissionHelper.php	17
DatabaseSeeder.php	18
VideoTest	19
VideosManageControllerTest	21
web.php	23
VideosAppLayout.blade.php	24
Resultats finals	26
resources/markdown/terms.md	33
Errors Larstan	33



Corregir els errors del 3r sprint.

En cas que als testos no s'hagi comprovat que els usuaris en permisos puguin accedir a la ruta /videos/manage, modificar-ho.

Retroacció

Qualificació	10,00 / 10,00
Qualificat el	dimarts, 18 de febrer 2025, 12:30
Qualificat per	JV Jordi Vega Tomàs [PROF]
Comentaris de retroalimentació	Molt bon treball, tot correcte!

VideosManageController.php

Aquest fitxer és un controlador de Laravel que gestiona la lògica CRUD (Crear, Llegir, Actualitzar, Esborrar) per a vídeos. Les funcions del controlador inclouen la visualització d'una llista de vídeos, la creació de nous vídeos, l'edició i actualització de vídeos existents, així com l'eliminació de vídeos. També comprova els permisos dels usuaris per assegurar-se que només els usuaris autoritzats poden realitzar aquestes accions.

```
© VideosManageController.php ×  
1 <?php  
2  
3 namespace App\Http\Controllers;  
4  
5 > use ...  
12  
6 * Wharvey123 *  
13 class VideosManageController extends Controller  
14 {  
15     // Retorna el nom de la classe de test associada  
16     no usages new *  
17     public function testedBy(): string  
18     {  
19         return VideosControllerTest::class;  
20     }  
21 }
```

```
// Mostra la llista de vídeos (CRUD)  
* Wharvey123 *  
public function index(): View|Factory|Application  
{  
    // Comprovar permisos  
    if (!auth()->user()->can('manage videos')) {  
        abort(403, 'Unauthorized');  
    }  
    // Obtenir tots els vídeos  
    $videos = Video::all();  
    // Retornar la vista amb els vídeos  
    return view('videos.manage.index', compact('var_name: 'videos'));  
}
```

```
// Mostra el formulari de creació de vídeo
new *

public function create(): View|Factory|Application
{
    // Comprovar permisos
    if (!auth()->user()->can([ abilities: 'videos.create'])) {
        abort( code: 403, message: 'Unauthorized');
    }
    // Retornar la vista del formulari de creació
    return view( view: 'videos.manage.create');
}
```

```
// Desa un vídeo nou
new *

public function store(Request $request): RedirectResponse
{
    // Comprovar permisos
    if (!auth()->user()->can([ abilities: 'videos.create'])) {
        abort( code: 403, message: 'Unauthorized');
    }

    // Validar les dades del formulari
    $data = $request->validate([
        'title'      => 'required|string|max:255',
        'description' => 'required|string',
        'url'        => 'required|url',
        'published_at'=> 'nullable|date',
        // 'series_id'   => 'required|integer|exists:series,id',
        'series_id'   => 'nullable|integer|exists:series,id',
    ]);

    // Crear el nou vídeo amb les dades validades
    Video::create($data);

    // Redirigir a la vista de llista de vídeos amb un missatge d'èxit
    return redirect()->route( route: 'videos.manage.index')
        ->with('success', 'Video creat correctament.');
}
```

```
// Mostra el formulari d'edició de vídeo
new *
public function edit($id): View|Factory|Application
{
    // Comprovar permisos
    if (!auth()->user()->can( abilities: 'videos.edit')) {
        abort( code: 403, message: 'Unauthorized');
    }
    // Obtenir el video per id
    $video = Video::findOrFail($id);
    // Retornar la vista del formulari d'edició amb el video
    return view( view: 'videos.manage.edit', compact( var_name: 'video'));
}
```

```
// Actualitza el video
new *
public function update(Request $request, $id): RedirectResponse
{
    // Comprovar permisos
    if (!auth()->user()->can( abilities: 'videos.edit')) {
        abort( code: 403, message: 'Unauthorized');
    }

    // Validar les dades del formulari
    $data = $request->validate([
        'title'      => 'required|string|max:255',
        'description' => 'required|string',
        'url'        => 'required|url',
        'published_at'=> 'nullable|date',
        'series_id'   => 'nullable|integer|exists:series,id',
    ]);

    // Obtenir el video per id
    $video = Video::findOrFail($id);

    // Actualitzar el video amb les dades validades
    $video->update($data);

    // Redirigir a la vista de llista de vídeos amb un missatge d'èxit
    return redirect()->route( route: 'videos.manage.index')
        ->with('success', 'Video actualitzat correctament.');
}
```

```
// Mostra la confirmació d'eliminació
new *

public function delete($id): View\Factory\Application
{
    // Comprovar permisos
    if (!auth()->user()->can(abilities: 'videos.delete')) {
        abort(code: 403, message: 'Unauthorized');
    }
    // Obtenir el vídeo per id
    $video = Video::findOrFail($id);
    // Retornar la vista de confirmació d'eliminació amb el vídeo
    return view(view: 'videos.manage.delete', compact(var_name: 'video'));
}
```

```
// Elimina el vídeo
▲ Wharvey123 *
public function destroy($id): RedirectResponse
{
    // Comprovar permisos
    if (!auth()->user()->can(abilities: 'videos.delete')) {
        abort(code: 403, message: 'Unauthorized');
    }
    // Obtenir el vídeo per id
    $video = Video::findOrFail($id);

    // Eliminar el vídeo
    $video->delete();

    // Redirigir a la vista de llista de vídeos amb un missatge d'èxit
    return redirect()->route(route: 'videos.manage.index')
        ->with('success', 'Video eliminat correctament.');
}
```

VideosController.php

Aquest fitxer és un controlador de Laravel que gestiona la visualització de vídeos publicats a la pàgina principal i la visualització detallada de vídeos específics. També retorna el nom complet de la classe de test associada. Utilitza Eloquent per recuperar els vídeos de la base de dades i comprova si els vídeos estan publicats abans de mostrar-los.

```
② VideosController.php ×

 5 > use ...
11
12     * Wharvey123 *
13 class VideosController extends Controller
14 {
15     /**
16      * @return View|Factory|Application
17     */
18     public function index(): View|Factory|Application
19     {
20         // Recuperem tots els vídeos publicats (on 'published_at' no és null)
21         $videos = Video::whereNotNull(string: 'published_at')->get();
22
23         // Retornem la vista 'videos.index' amb la variable $videos
24         return view(view: 'videos.index', compact(var_name: 'videos'));
25     }
26
27     /**
28      * @param int $id
29      * @return View|Factory|Application
30     */
31     * Wharvey123 *
32     public function show(int $id): View|Factory|Application
33     {
34         // Obtenim el video per id
35         $video = Video::findOrFail($id);
36
37         // Comprovar si el video és publicat
38         if (!$video->published_at) {
39             abort(code: 404, message: 'Aquest vídeo no està disponible.');
40         }
41
42         // Retornem la vista amb el video
43         return view(view: 'videos.show', compact(var_name: 'video'));
44     }
45
46     /**
47      * @return string
48      */
49     public function testedBy(): string
50     {
51         return VideosControllerTest::class;
52     }
53 }
```

VideoHelper.php

Aquest fitxer és un ajudant de Laravel que conté una funció estàtica per crear o actualitzar vídeos per defecte a la base de dades. Aquesta funció defineix un conjunt de vídeos amb títols, descripcions, URLs, dates de publicació, i enllaços a la sèrie corresponent, i després utilitza `updateOrCreate` per inserir o actualitzar aquests vídeos a la base de dades.

```
© VideoHelper.php ×

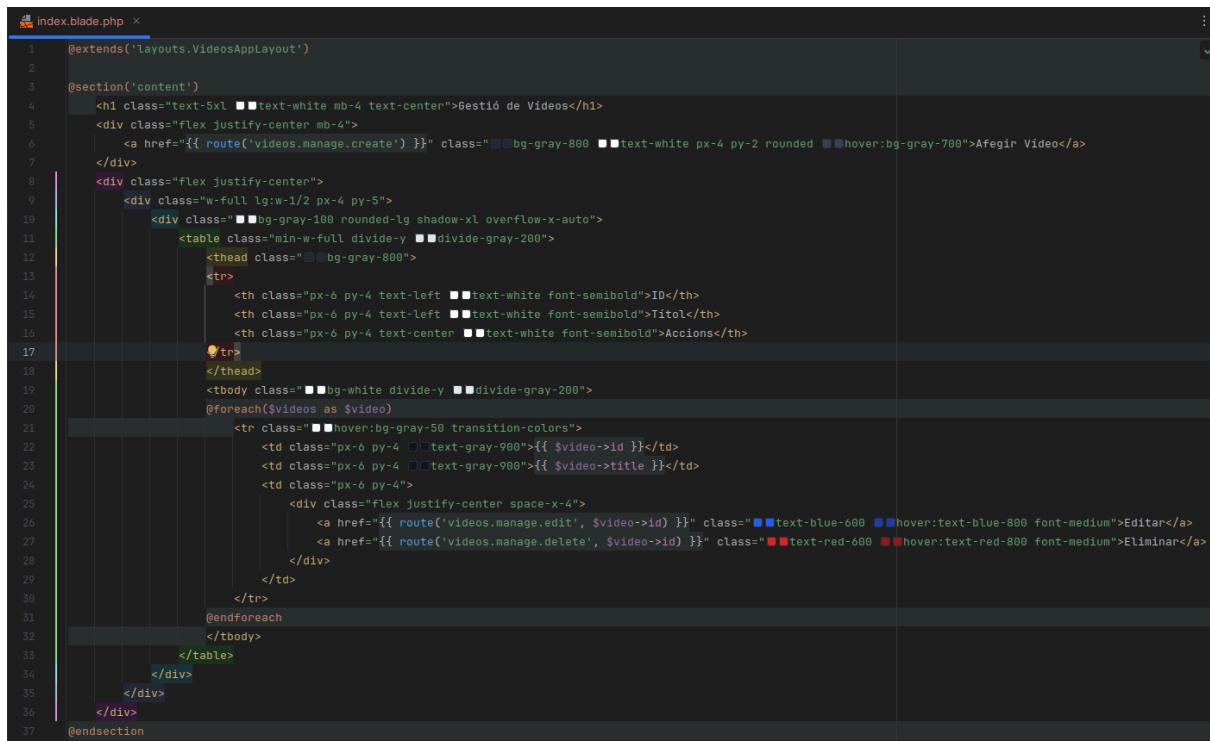
1 Wharvey123 *
2 class VideoHelper
3 {
4     /**
5      * Crea o actualitza vídeos per defecte a la base de dades.
6      * @return void
7      */
8     Wharvey123 *
9     public static function createDefaultVideos(): void
10    {
11        // Definim els vídeos per defecte
12        $defaultVideos = [
13            [
14                [
15                    'title' => 'Introducció a Laravel',
16                    'description' => 'Un vídeo introductori per aprendre els conceptes bàsics de Laravel.',
17                    'url' => 'https://www.youtube.com/embed/PQ0xIILBb7M',
18                    'published_at' => now(),
19                    'previous' => null,
20                    'next' => 2,
21                    'series_id' => 1,
22                ],
23            ],
24            [
25                [
26                    'title' => 'Controladors a Laravel',
27                    'description' => 'Apren com funcionen els controladors a Laravel i com gestionar les rutes.',
28                    'url' => 'https://www.youtube.com/embed/0YxgCH2R2bE',
29                    'published_at' => now(),
30                    'previous' => 1,
31                    'next' => 3,
32                    'series_id' => 1,
33                ],
34            ],
35            [
36                [
37                    'title' => 'Models a Laravel',
38                    'description' => 'Exploració dels models a Laravel i com interactuar amb la base de dades.',
39                    'url' => 'https://www.youtube.com/embed/f-pWNf0Ht1Y',
40                    'published_at' => now(),
41                    'previous' => 2,
42                    'next' => null,
43                    'series_id' => 1,
44                ],
45            ],
46        ],
47    ],
48
49    /**
50     * Per a cada vídeo per defecte, crear o actualitzar el registre a la base de dades
51     */
52    foreach ($defaultVideos as $video) {
53        Video::updateOrCreate(
54            ['title' => $video['title']], // Condició per determinar si el vídeo ja existeix
55            $video // Dades del vídeo
56        );
57    }
58}
```

Vistes per al CRUD

```
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ mkdir -p resources/views/videos/manage
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch resources/views/videos/manage/index.blade.php
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch resources/views/videos/manage/create.blade.php
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch resources/views/videos/manage/edit.blade.php
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch resources/views/videos/manage/delete.blade.php
```

[index.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel per gestionar els vídeos. Aquesta plantilla mostra una pàgina amb una taula que conté una llista de vídeos, amb opcions per afegir, editar i eliminar vídeos. També inclou enllaços per accedir a les pàgines de creació i eliminació de vídeos.



```
index.blade.php
1  @extends('layouts.VideosAppLayout')
2
3  @section('content')
4      <h1 class="text-5xl mb-4 text-white text-center">Gestió de Vídeos</h1>
5      <div class="flex justify-center mb-4">
6          <a href="{{ route('videos.manage.create') }}" class="bg-gray-800 text-white px-4 py-2 rounded hover:bg-gray-700">Afegir Video</a>
7      </div>
8      <div class="flex justify-center">
9          <div class="w-full lg:w-1/2 px-4 py-5">
10             <div class="bg-gray-100 rounded-lg shadow-xl overflow-x-auto">
11                 <table class="min-w-full divide-y divide-gray-200">
12                     <thead class="bg-gray-800">
13                         <tr>
14                             <th class="px-6 py-4 text-left text-white font-semibold">ID</th>
15                             <th class="px-6 py-4 text-left text-white font-semibold">Títol</th>
16                             <th class="px-6 py-4 text-center text-white font-semibold">Accions</th>
17                         </tr>
18                     </thead>
19                     <tbody class="bg-white divide-y divide-gray-200">
20                         @foreach($videos as $video)
21                             <tr class="hover:bg-gray-50 transition-colors">
22                                 <td class="px-6 py-4 text-gray-900">{{ $video->id }}</td>
23                                 <td class="px-6 py-4 text-gray-900">{{ $video->title }}</td>
24                                 <td class="px-6 py-4">
25                                     <div class="flex justify-center space-x-4">
26                                         <a href="{{ route('videos.manage.edit', $video->id) }}" class="text-blue-600 hover:text-blue-800 font-medium">Editar</a>
27                                         <a href="{{ route('videos.manage.delete', $video->id) }}" class="text-red-600 hover:text-red-800 font-medium">Eliminar</a>
28                                     </div>
29                                 </td>
30                             </tr>
31                         @endforeach
32                     </tbody>
33                 </table>
34             </div>
35         </div>
36     @endsection
```

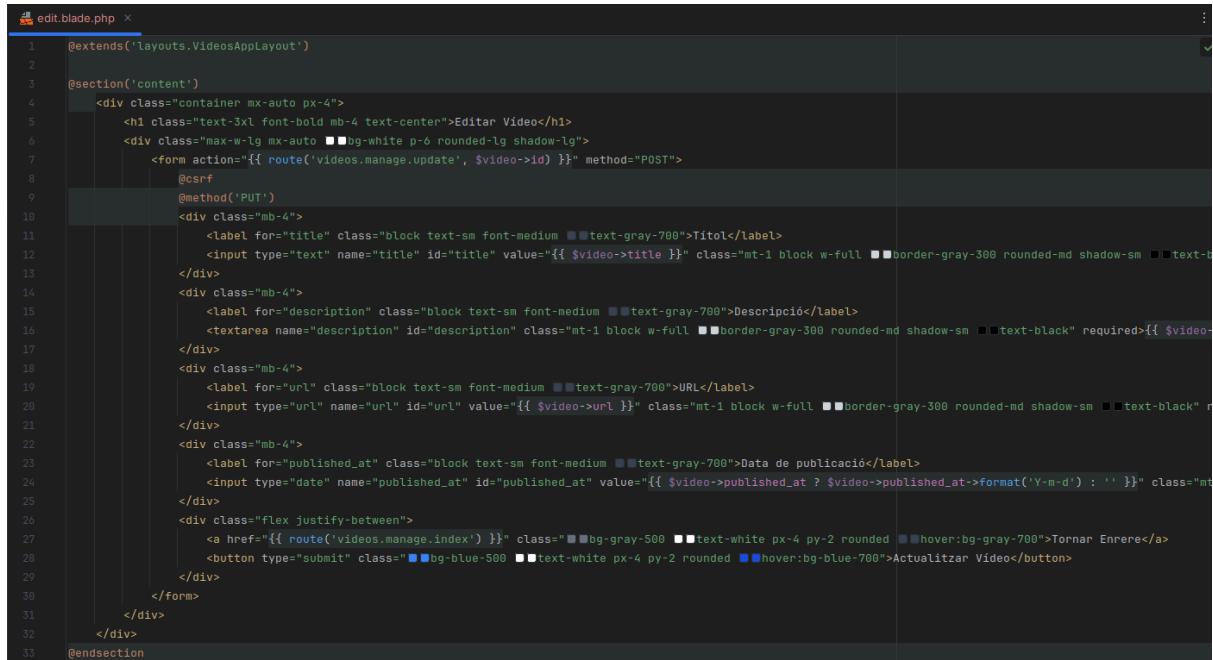
[create.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel per afegir un nou vídeo. Aquesta plantilla mostra un formulari on els usuaris poden introduir el títol, la descripció, l'URL i la data de publicació del vídeo. També inclou botons per tornar enrere i desar el vídeo.

```
create.blade.php ×
1  @extends('layouts.VideosAppLayout')
2
3  @section('content')
4      <div class="max-w-xl mx-auto bg-gray-800 p-6 rounded-lg shadow-lg">
5          <h1 class="text-2xl font-bold mb-4 text-white">Afegir Nou Video</h1>
6          <form action="{{ route('videos.manage.store') }}" method="POST">
7              @csrf
8              <div class="mb-4">
9                  <label for="title" class="block text-sm font-medium text-white">Títol</label>
10                 <input type="text" name="title" id="title" class="mt-1 block w-full p-2 bg-gray-700 text-white border border-gray-600 rounded shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2" required>
11             </div>
12             <div class="mb-4">
13                 <label for="description" class="block text-sm font-medium text-white">Descripció</label>
14                 <textarea name="description" id="description" class="mt-1 block w-full p-2 bg-gray-700 text-white border border-gray-600 rounded shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2" required></textarea>
15             </div>
16             <div class="mb-4">
17                 <label for="url" class="block text-sm font-medium text-white">URL</label>
18                 <input type="url" name="url" id="url" class="mt-1 block w-full p-2 bg-gray-700 text-white border border-gray-600 rounded shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2" required>
19             </div>
20             <div class="mb-4">
21                 <label for="published_at" class="block text-sm font-medium text-white">Data de publicació</label>
22                 <input type="date" name="published_at" id="published_at" class="mt-1 block w-full p-2 bg-gray-700 text-white border border-gray-600 rounded shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2" required>
23             </div>
24             {{-->
25             <div class="mb-4">
26                 <label for="series_id" class="block text-sm font-medium text-white">Sèrie</label>
27                 <select name="series_id" id="series_id" class="mt-1 block w-full p-2 bg-gray-700 text-white border border-gray-600 rounded shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2" required>
28                     @foreach($series as $serie)
29                         <option value="{{ $serie->id }}>{{ $serie->name }}</option>
30                     @endforeach
31                 </select>
32             </div>
33             ->}}
34             <a href="{{ route('videos.manage.index') }}" class="bg-gray-500 text-white px-4 py-2 rounded hover:bg-gray-700" style="width: 150px; display: inline-block">Anul·lar</a>
35             <button type="submit" class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-700" data-qa="submit-video" style="width: 150px; display: inline-block">Desar</button>
36         </form>
37     </div>
```

[edit.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel per editar un vídeo existent. Aquesta plantilla mostra un formulari on els usuaris poden actualitzar el títol, la descripció, l'URL i la data de publicació del vídeo. També inclou botons per tornar enrere i actualitzar el vídeo.



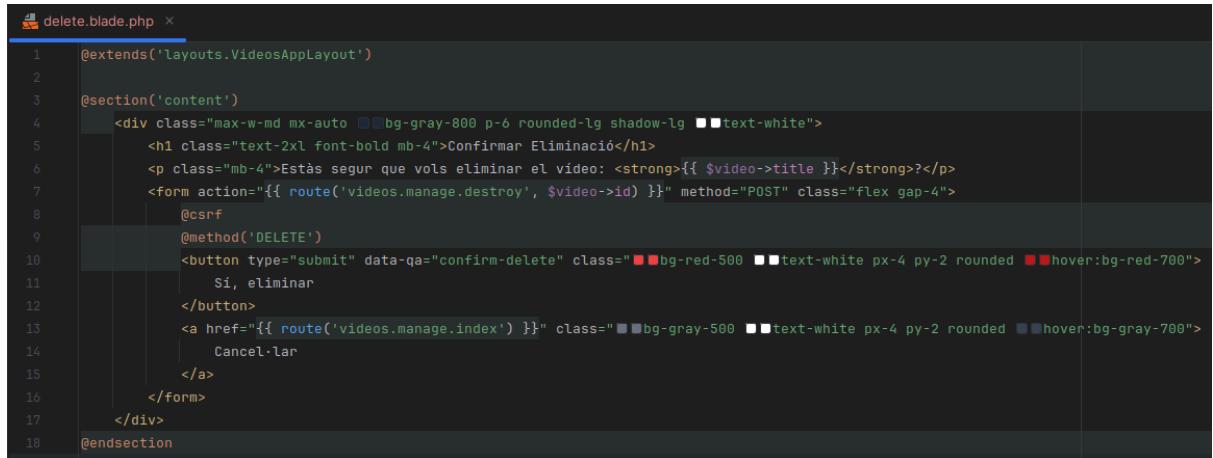
```

1  @extends('layouts.VideosAppLayout')
2
3  @section('content')
4      <div class="container mx-auto px-4">
5          <h1 class="text-3xl font-bold mb-4 text-center">Editar Video</h1>
6          <div class="max-w-lg mx-auto bg-white p-6 rounded-lg shadow-lg">
7              <form action="{{ route('videos.manage.update', $video->id) }}" method="POST">
8                  @csrf
9                  @method('PUT')
10                 <div class="mb-4">
11                     <label for="title" class="block text-sm font-medium text-gray-700">Títol</label>
12                     <input type="text" name="title" id="title" value="{{ $video->title }}" class="mt-1 block w-full border-gray-300 rounded-md shadow-sm text-black" required>
13                 </div>
14                 <div class="mb-4">
15                     <label for="description" class="block text-sm font-medium text-gray-700">Descripció</label>
16                     <textarea name="description" id="description" class="mt-1 block w-full border-gray-300 rounded-md shadow-sm text-black" required>{{ $video->description }}</textarea>
17                 </div>
18                 <div class="mb-4">
19                     <label for="url" class="block text-sm font-medium text-gray-700">URL</label>
20                     <input type="url" name="url" id="url" value="{{ $video->url }}" class="mt-1 block w-full border-gray-300 rounded-md shadow-sm text-black" required>
21                 </div>
22                 <div class="mb-4">
23                     <label for="published_at" class="block text-sm font-medium text-gray-700">Data de publicació</label>
24                     <input type="date" name="published_at" id="published_at" value="{{ $video->published_at ? $video->published_at->format('Y-m-d') : '' }}" class="mt-1 block w-full border-gray-300 rounded-md shadow-sm text-black" required>
25                 </div>
26                 <div class="flex justify-between">
27                     <a href="{{ route('videos.manage.index') }}" class="bg-gray-500 text-white px-4 py-2 rounded hover:bg-gray-700">Tornar Enrere</a>
28                     <button type="submit" class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-700">Actualitzar Video</button>
29                 </div>
30             </form>
31         </div>
32     </div>
33 @endsection

```

[delete.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel per editar un vídeo existent. Aquesta plantilla mostra un formulari on els usuaris poden actualitzar el títol, la descripció, l'URL i la data de publicació del vídeo. També inclou botons per tornar enrere i actualitzar el vídeo.



```

1  @extends('layouts.VideosAppLayout')
2
3  @section('content')
4      <div class="max-w-md mx-auto bg-gray-800 p-6 rounded-lg shadow-lg text-white">
5          <h1 class="text-2xl font-bold mb-4">Confirmar Eliminació</h1>
6          <p class="mb-4">Estàs segur que vols eliminar el video: <strong>{{ $video->title }}</strong>?</p>
7          <form action="{{ route('videos.manage.destroy', $video->id) }}" method="POST" class="flex gap-4">
8              @csrf
9              @method('DELETE')
10             <button type="submit" data-qa="confirm-delete" class="bg-red-500 text-white px-4 py-2 rounded hover:bg-red-700">
11                 Si, eliminar
12             </button>
13             <a href="{{ route('videos.manage.index') }}" class="bg-gray-500 text-white px-4 py-2 rounded hover:bg-gray-700">
14                 Cancel·lar
15             </a>
16         </form>
17     </div>
18 @endsection

```

Vistes per als vídeos

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch resources/views/videos/index.blade.php
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ ls resources/views/videos/
index.blade.php  manage  show.blade.php
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$
```

[resources/views/videos/index.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel per a una aplicació de vídeos anomenada "VideosApp". Defineix el títol de la pàgina com "VideosApp - Inici" i crea una vista que mostra una graella de vídeos. Per a cada vídeo, es mostra una imatge (obtinguda de YouTube o una imatge de substitució), el títol del vídeo i la data de publicació. Els vídeos es mostren dins de cartes estilitzades amb un enllaç que porta a la pàgina de detall del vídeo.

```
index.blade.php x
1  @extends('layouts.VideosAppLayout')
2
3  @section('title', 'VideosApp - Inici')
4
5  @php
6      // Si la funció getYoutubeThumbnail no existeix, la definim
7      if (!function_exists('getYoutubeThumbnail')) {
8          function getYoutubeThumbnail($url): string
9          {
10              // Usem una expressió regular per trobar l'ID del vídeo de YouTube a l'URL
11              preg_match('/(?:youtube\.com\/(?:[^/]+\/){2}|(?:v|e(?:mbd)?))\//.*[&v=](\w{11})/i', $url, $matches);
12              // Si trobem un ID de vídeo, retornem l'URL de la miniatura. Si no, retornem una imatge de substitució
13              return isset($matches[1]) ? 'https://img.youtube.com/vi/' . $matches[1] . '/hqdefault.jpg' : asset('images/placeholder.png');
14          }
15      }
16  @endphp
```



```
@section('content')
<div class="container mx-auto px-4">
    <!-- Títol principal -->
    <h1 class="text-3xl font-bold text-white my-6">VideosApp</h1>

    <!-- Grid de vídeos, similar a YouTube -->
    <div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
        @foreach($videos as $video)
            <div class="bg-gray-800 rounded-lg shadow-lg overflow-hidden">
                <!-- Enllaç al vídeo -->
                <a href="{{ route('video.show', $video->id) }}">
                    <!-- Imatge del vídeo, ja sigui la miniatura de YouTube o una imatge de substitució -->
                    title }}" class="w-full h-48 object-cover">
                    <div class="p-4">
                        <!-- Títol del vídeo -->
                        <h2 class="text-xl font-semibold text-white">{{ $video->title }}
                        <!-- Data de publicació del vídeo -->
                        <p class="text-gray-400 text-sm mt-2">
                            Publicat: {{ $video->formatted_published_at }}
                        </p>
                    </div>
                </a>
            </div>
        @endforeach
    </div>
    </div>
@endsection
```

[resources/views/videos/show.blade.php](#)

Aquest fitxer és una plantilla Blade de Laravel que mostra la pàgina de detall d'un vídeo per a l'aplicació "VideosApp". Defineix el títol de la pàgina com el títol del vídeo i mostra l'encapçalament amb el títol i la data de publicació en dos formats. També inclou un iframe per incrustar el vídeo de YouTube, la descripció del vídeo i els enllaços de navegació per accedir als vídeos anterior i següent, si estan disponibles.

```
show.blade.php ×
1  @extends('layouts.VideosAppLayout')
2
3  @section('title', $video->title)
4
5  @section('content')
6      <!-- Encapçalament -->
7      <div class="bg-gray-800 rounded-lg shadow-lg p-6">
8          <!-- Títol del vídeo -->
9          <h1 class="text-3xl font-bold text-white">{{ $video->title }}</h1>
10         <p class="text-sm text-gray-400 mt-2">
11             <!-- Data de publicació en dos formats -->
12             Publicat: {{ $video->formatted_published_at }}
13             <span class="text-gray-500">|</span>
14             {{ $video->formatted_for_humans_published_at }}
15         </p>
16     </div>
17
18     <!-- Contingut del vídeo -->
19     <div class="mt-8">
20         <div class="bg-gray-800 rounded-lg shadow-lg overflow-hidden">
21             <!-- Iframe per al vídeo de YouTube -->
22             <iframe
23                 class="w-full"
24                 style="height: calc(100vw * 9 / 16); max-height: 500px;"
25                 src="{{ $video->url }}"
26                 frameborder="0"
27                 allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
28                 allowfullscreen>
29             </iframe>
30         </div>
31     </div>
```

```
<!-- Descripció del video -->
<div class="mt-8 bg-gray-800 rounded-lg shadow-lg p-6">
    <h2 class="text-xl font-semibold text-white">Descripció</h2>
    <p class="text-gray-300 mt-4">{{ $video->description }}</p>
</div>

<!-- Navegació entre vídeos -->
<div class="mt-8 flex justify-between items-center text-sm">
    @if($video->previous)
        <!-- Enllaç al vídeo anterior -->
        <a href="{{ url('/video/' . $video->previous) }}">
            class="text-blue-400 hover:text-blue-600 font-medium">
                &larr; Video anterior
        </a>
    @else
        <!-- Missatge si no hi ha vídeo anterior -->
        <span class="text-gray-500">No hi ha vídeo anterior</span>
    @endif

    @if($video->next)
        <!-- Enllaç al vídeo següent -->
        <a href="{{ url('/video/' . $video->next) }}">
            class="text-blue-400 hover:text-blue-600 font-medium">
                Video següent &rarr;
        </a>
    @else
        <!-- Missatge si no hi ha vídeo següent -->
        <span class="text-gray-500">No hi ha vídeo següent</span>
    @endif
</div>
@endsection
```

Test user_with_permissions_can_manage_videos()

Aquesta funció és una prova unitària que comprova si un usuari amb permisos de gestor de vídeos pot gestionar vídeos en l'aplicació. Primer, inicia sessió com a gestor de vídeos i crea tres vídeos utilitzant una fàbrica de models. Després, accedeix a la pàgina de gestió de vídeos i verifica que l'usuari pugui accedir-hi i que els vídeos creats es mostrin correctament a la pàgina.

```
new *  
#[Test]  
public function user_with_permissions_can_manage_videos()  
{  
    // Arrange  
    $this->loginAsVideoManager();  
  
    // Crear tres vídeos  
    Video::factory()->count( count: 3)->create();  
  
    // Act: Accedir a la pàgina de gestió de vídeos  
    $response = $this->get(route( name: 'videos.manage.index'));  
  
    // Assert: Verificar que l'usuari pot accedir a la pàgina  
    $response->assertStatus( status: 200);  
  
    // Verificar que els vídeos es mostren a la pàgina  
    $response->assertSee(Video::first()->title);  
    $response->assertSee(Video::skip( int: 1)->first()->title);  
    $response->assertSee(Video::skip( int: 2)->first()->title);  
}
```

Permisos de vídeos per al CRUD

[PermissionHelper.php](#)

Aquest fitxer defineix una classe PermissionHelper que conté funcions per gestionar els permisos necessaris per administrar vídeos en una aplicació Laravel. La funció `define_gates` defineix una porta d'accés per permetre la gestió de vídeos a usuaris amb el permís necessari o als superadministradors. La funció `create_permissions` crea els permisos necessaris per al CRUD (Crear, Llegir, Actualitzar, Eliminar) de vídeos si no existeixen.

```
② PermissionHelper.php ×
1  <?php
2
3  namespace App\Helpers;
4
5  > use ...
6
7
8  class PermissionHelper
9  {
10
11     /** Defineix les portes d'accés per a la gestió de vídeos. */
12     public static function define_gates(): void
13     {
14         Gate::define('ability: manage-videos', function ($user) {
15             // Permet gestionar vídeos si l'usuari té el permís o és superadmin
16             return $user->hasPermissionTo('manage videos') || $user->isSuperAdmin();
17         });
18     }
19
20     /** Crea els permisos necessaris per al CRUD de vídeos. */
21     public static function create_permissions(): void
22     {
23         // Llista de permisos necessaris per gestionar els vídeos
24         $permissions = [
25             'manage videos',
26             'videos.create',
27             'videos.edit',
28             'videos.delete',
29         ];
30
31         // Per a cada permís, crea'l si no existeix
32         foreach ($permissions as $permission) {
33             Permission::firstOrCreate(['name' => $permission]);
34         }
35     }
}
```

DatabaseSeeder.php

Aquest fitxer defineix una classe DatabaseSeeder que s'utilitza per inicialitzar la base de dades amb dades per defecte. La funció run crea els permisos necessaris per gestionar els vídeos, crea diferents tipus d'usuaris (incloent superadministrador, professor, i gestor de vídeos), i assigna els permisos corresponents a cada rol. També assigna equips personals a cada usuari i crea vídeos per defecte.

```
② DatabaseSeeder.php ×
10  class DatabaseSeeder extends Seeder
11
12      // Wharvey123 *
13      public function run(): void
14      {
15          // Crear permisos (inclou 'videos.create', 'videos.edit' i 'videos.delete')
16          PermissionHelper::create_permissions();
17
18          // Crear usuaris sense assignar current_team_id (es farà a add_personal_team)
19          $defaultUser      = UserHelper::createDefaultUser();
20          $defaultProfessor = UserHelper::createDefaultProfessor();
21          $superadmin       = UserHelper::create_superuser();
22          $regularUser     = UserHelper::create_regular_user();
23          $videoManager     = UserHelper::create_video_manager_user();
24
25          // Assignar equips personals (current_team_id = user id)
26          UserHelper::add_personal_team($defaultUser);
27          UserHelper::add_personal_team($defaultProfessor);
28          UserHelper::add_personal_team($superadmin);
29          UserHelper::add_personal_team($regularUser);
30          UserHelper::add_personal_team($videoManager);
31
32          // Assignar permisos segons rol:
33          // Per exemple, el professor i el superadmin tenen 'manage videos'
34          $defaultProfessor->givePermissionTo('manage videos');
35          $superadmin->givePermissionTo('manage videos');
36
37          // Al Video Manager li assignem els permisos específics per al CRUD:
38          $videoManager->givePermissionTo('manage videos');
39          $videoManager->givePermissionTo('videos.create');
40          $videoManager->givePermissionTo('videos.edit');
41          $videoManager->givePermissionTo('videos.delete');
42
43          // Al Super Admin li assignem els permisos específics per al CRUD:
44          $superadmin->givePermissionTo('manage videos');
45          $superadmin->givePermissionTo('videos.create');
46          $superadmin->givePermissionTo('videos.edit');
47          $superadmin->givePermissionTo('videos.delete');
48
49          // Crear vídeos per defecte
50          VideoHelper::createDefaultVideos();
    }
```

VideoTest

La metodologia TDD (Test-Driven Development) i el patró AAA (Arrange, Act, Assert) són fonamentals per escriure tests efectius i mantenibles en el desenvolupament de programari. A continuació, es presenta una guia pas a pas per implementar aquestes metodologies en els teus tests, utilitzant com a exemple la classe VideosTest que has proporcionat.

1. Comprendre TDD i AAA

- **TDD (Test-Driven Development):** És una pràctica de desenvolupament on primer s'escriu un test que falla (perquè la funcionalitat encara no està implementada), després es desenvolupa el codi mínim necessari per passar el test, i finalment es refactoriza el codi per millorar-ne la qualitat mantenint els tests verds.
- **AAA (Arrange, Act, Assert):** És un patró per estructurar tests unitàries que consta de tres passos:
 - **Arrange (Preparació):** Configura l'escenari i les dades necessàries per al test.
 - **Act (Acció):** Executa l'acció o comportament que vols provar.
 - **Assert (Afirmació):** Verifica que el resultat obtingut és l'esperat.

2. Aplicar TDD i AAA en els teus tests

A continuació, es mostra com aplicar aquestes metodologies en els tests de la classe VideosTest.

Pas 1: Escriure un test que falli (TDD - Red)

Suposem que volem assegurar-nos que els usuaris sense permisos específics poden veure la pàgina de vídeos per defecte. Primer, escrivim un test per a aquesta funcionalitat:

En aquest punt, si la funcionalitat no està implementada, el test hauria de fallar.

Pas 2: Escriure el codi mínim per passar el test (TDD - Green)

Implementem la funcionalitat perquè el test passi. Això podria implicar definir la ruta /videos i el controlador corresponent:

Després d'aquesta implementació, el test hauria de passar.

Pas 3: Refactoritzar el codi (TDD - Refactor)

Amb el test passant, podem refactoritzar el codi per millorar-ne la qualitat, assegurant-nos que els tests continuen passant després de cada canvi.

3. Estructurar els tests amb el patró AAA

En els tests proporcionats, ja s'està seguint el patró AAA. Per exemple, en el test `user_with_permissions_can_see_default_videos_page`:

- **Arrange (Preparació):** Es crea un permís i un usuari amb aquest permís.
- **Act (Acció):** S'autentica l'usuari i es fa una sol·licitud GET a la pàgina de vídeos.
- **Assert (Afirmació):** Es verifica que la resposta és correcta (estat 200).

4. Consells addicionals

- **Nom dels tests:** Utilitza noms descriptius que indiquin clarament què estan provant.
- **Independència dels tests:** Assegura't que cada test pugui executar-se de manera independent i no depengui de l'estat deixat per altres tests.
- **Ús de factories:** Les factories són útils per crear dades de prova de manera senzilla i coherent.
- **Refactorització contínua:** Revisa i millora regularment els teus tests per mantenir-los nets i fàcils de mantenir.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan test --filter=VideosTest

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

  PASS  Tests\Feature\VideosTest
    ✓ user without permissions can see default videos page
    ✓ user with permissions can see default videos page
    ✓ not logged users can see default videos page

Tests:      5 passed (5 assertions)
Duration: 0.20s

alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$
```

VideosManageControllerTest

La metodologia **TDD (Test-Driven Development)** implica escriure primer les proves abans de desenvolupar el codi funcional. Aquesta pràctica assegura que el codi compleix amb els requisits específicats i facilita el manteniment i l'escalabilitat. Dins de les proves, el patró **AAA (Arrange, Act, Assert)** ajuda a estructurar-les de manera clara:

1. **Arrange (Preparació):** Configura les dades i l'entorn necessaris per a la prova.
2. **Act (Acció):** Executa l'acció o comportament que vols provar.
3. **Assert (Afirmació):** Verifica que el resultat obtingut coincideix amb el resultat esperat.

A continuació, es mostra una guia pas a pas per implementar aquestes metodologies en les proves del controlador de gestió de vídeos:

1. Preparació de l'entorn de proves:

- Utilitza la trait RefreshDatabase per assegurar que la base de dades es reinicia abans de cada prova, garantint un entorn net.
- Crea usuaris amb diferents rols i permisos utilitzant helpers com PermissionHelper i UserHelper. Això permet simular diferents nivells d'accés en les proves.

2. Estructura de les proves amb el patró AAA:

Arrange: En cada prova, autentica l'usuari adequat i prepara les dades necessàries. Per exemple, per a un usuari amb permisos de gestió de vídeos:

Act: Executa l'acció que vols provar, com ara accedir a una ruta específica o enviar una sol·licitud POST:

Assert: Verifica que el resultat és l'esperat, com ara un codi d'estat HTTP específic o la presència d'un registre a la base de dades:

3. Exemples de proves:

- **Usuari amb permisos pot veure la pàgina de creació de vídeos**
- **Usuari sense permisos no pot emmagatzemar vídeos**

4. Consideracions addicionals:

- **Reutilització de codi:** Utilitza mètodes auxiliars com loginAsVideoManager per evitar duplicació de codi i mantenir les proves netes i llegibles.
- **Cobertura de casos:** Assegura't de provar tant els casos en què l'usuari té els permisos necessaris com aquells en què no els té, per garantir una gestió adequada dels permisos en l'aplicació.



```
ałumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan test --filter=VideosManageControllerTest

  PASS Tests\Feature\Videos\VideosManageControllerTest
    ✓ login as video manager
    ✓ login as super admin
    ✓ login as regular user
    ✓ user with permissions can see add videos
    ✓ user without videos manage create cannot see add videos
    ✓ user with permissions can store videos
    ✓ user without permissions cannot store videos
    ✓ user with permissions can destroy videos
    ✓ user without permissions cannot destroy videos
    ✓ user with permissions can see edit videos
    ✓ user without permissions cannot see edit videos
    ✓ user with permissions can update videos
    ✓ user without permissions cannot update videos
    ✓ user with permissions can manage videos
    ✓ regular users cannot manage videos
    ✓ guest users cannot manage videos
    ✓ superadmins can manage videos

  Tests:  17 passed (71 assertions)
  Duration: 0.34s

ałumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$
```

web.php

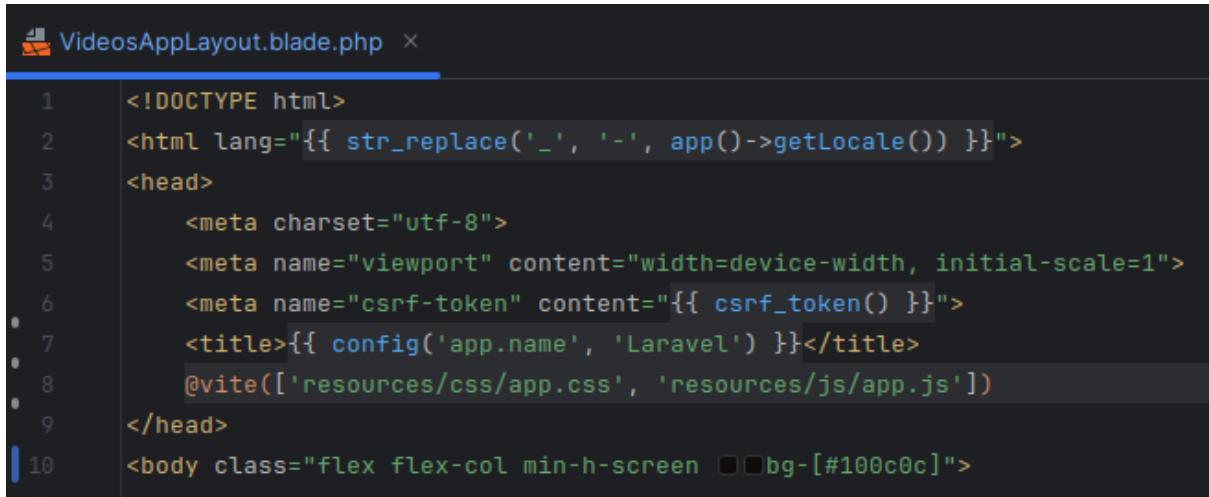
- **Ruta d'inici:** Redirecciona a la pàgina principal de vídeos.
- **Rutes amb autenticació:** Grup de rutes que requereixen que l'usuari estigui autenticat.
 - **Tauler de control:** Accés a la pàgina del tauler de control per a usuaris autenticats.
 - **Gestió de vídeos:** CRUD (crear, llegir, actualitzar i eliminar) de vídeos, accessible només per usuaris autenticats.
- **Ruta per mostrar un vídeo específic:** Accessible per a tots els usuaris.
- **Ruta per a la pàgina principal de vídeos:** Accessible per a tots els usuaris.
- **Ruta per a testos de vídeos:** Accessible per a tots els usuaris.

```
php web.php ×
1 <?php
2
3 > use ...
4
5
6
7 // Ruta per a la pàgina d'inici
8 Route::get('uri: "/', function () {
9     return redirect()->route('route: 'videos.index');
10 });
11
12 // Grup de rutes que requereixen autenticació
13 Route::middleware(['auth:sanctum', config(key: 'jetstream.auth_session'), 'verified'])->group(function () {
14     // Ruta per al tauler de control
15     Route::get('uri: '/dashboard', function () {
16         return view('view: 'dashboard');
17     })->name('name: 'dashboard');
18
19     // Rutes per a la gestió de vídeos (CRUD) només accessibles per usuaris autenticats
20     Route::prefix('prefix: 'videos/manage')->name('value: 'videos.manage.')->group(function () {
21         Route::get('uri: '/', [VideosManageController::class, 'index'])->name('name: 'index');
22         Route::get('uri: '/create', [VideosManageController::class, 'create'])->name('name: 'create');
23         Route::post('uri: '/', [VideosManageController::class, 'store'])->name('name: 'store');
24         Route::get('uri: '/{id}', [VideosManageController::class, 'show'])->name('name: 'show');
25         Route::get('uri: '/{id}/edit', [VideosManageController::class, 'edit'])->name('name: 'edit');
26         Route::put('uri: '/{id}', [VideosManageController::class, 'update'])->name('name: 'update');
27         Route::delete('uri: '/{id}', [VideosManageController::class, 'destroy'])->name('name: 'destroy');
28         Route::get('uri: '/{id}/delete', [VideosManageController::class, 'delete'])->name('name: 'delete');
29     });
30 });
31
32 // Ruta per mostrar un video específic, accessible per a tots els usuaris
33 Route::get('uri: '/video/{id}', [VideosController::class, 'show'])->name('name: 'video.show');
34
35 // Ruta per a la pàgina principal de vídeos, accessible per a tots els usuaris
36 Route::get('uri: '/videos', [VideosController::class, 'index'])->name('name: 'videos.index');
37
38 // Ruta per a testos de videos, accessible per a tots els usuaris
39 Route::get('uri: '/test-videos', [VideosController::class, 'testedBy']);
```

VideosAppLayout.blade.php

Aquest codi HTML defineix l'estructura bàsica d'una pàgina web per a una aplicació Laravel. Aquí tens una descripció breu:

- **Encapçalament del document HTML:** Inclou la configuració bàsica com la codificació de caràcters, la visualització adaptable, el token CSRF i el títol de l'aplicació. També inclou els arxius CSS i JavaScript necessaris.
- **Cos del document HTML:**
 - **Navbar:** Una barra de navegació superior que inclou un enllaç als vídeos i opcions d'autenticació. Mostra diferents opcions dependent si l'usuari està autenticat o no.
 - **Contingut principal:** Una secció que conté el contingut principal de la pàgina, que serà rendit per una vista específica.
 - **Footer:** Un peu de pàgina que mostra el copyright i el nom de l'aplicació.



The screenshot shows a code editor window with the file 'VideosAppLayout.blade.php' open. The code is a standard HTML template for a Laravel application, featuring DOCTYPE, head, and body sections with various meta tags, a title, and a vite script tag. The code is color-coded for syntax highlighting.

```
<!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>
    @vite(['resources/css/app.css', 'resources/js/app.js'])
</head>
<body class="flex flex-col min-h-screen bg-[#100c0c]">
```

```
<!-- Navbar -->
<nav class="bg-gray-800 p-6 shadow-lg">
  <div class="container mx-auto flex justify-between items-center">
    <!-- Enllaç a videos -->
    <a href="{{ route('videos.index') }}" class="text-white text-lg font-semibold">Videos</a>

    <!-- Autenticació -->
    <div class="ml-auto relative" x-data="{ open: false }">
      @auth
        <!-- Si l'usuari està autenticat -->
        <button @click="open = !open" class="flex items-center text-white focus:outline-none">
          
        </button>

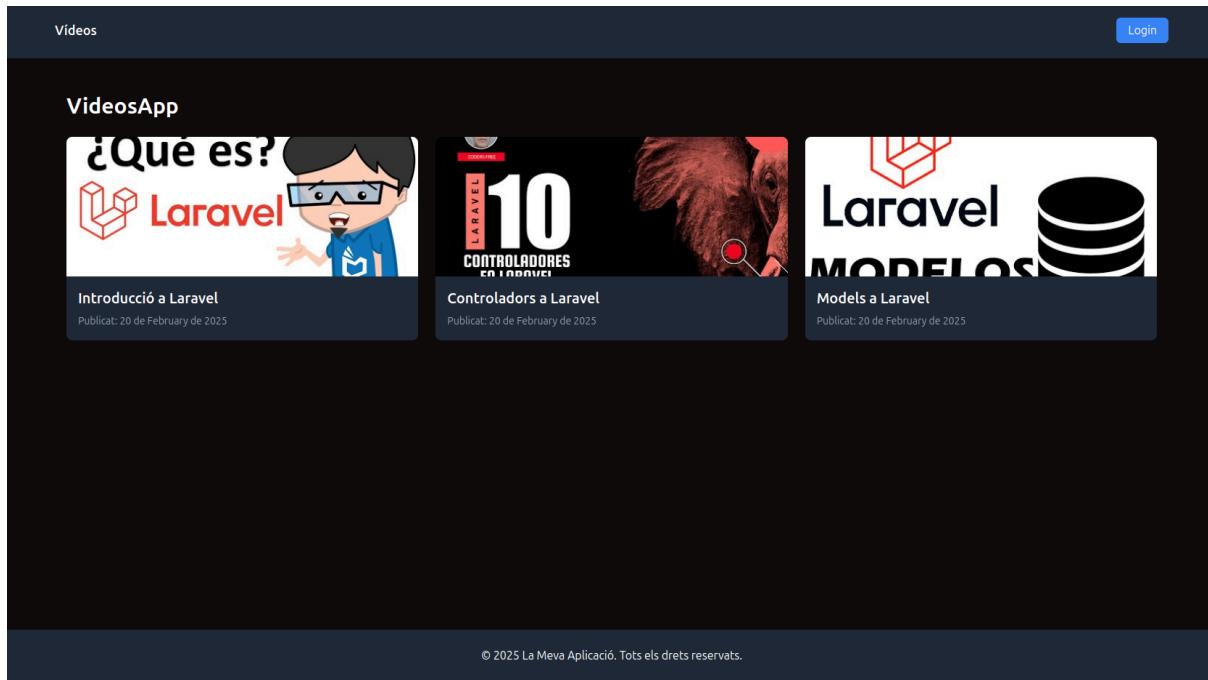
        <!-- Dropdown -->
        <div x-show="open" @click.away="open = false" class="absolute right-0 mt-2 w-48 bg-white rounded-lg shadow-lg py-2">
          <a href="{{ route('profile.show') }}" class="block px-4 py-2 text-gray-800 hover:bg-gray-200">Perfil</a>
          <form method="POST" action="{{ route('logout') }}>
            @csrf
            <button type="submit" class="block w-full text-left px-4 py-2 text-gray-800 hover:bg-gray-200">
              Logout
            </button>
          </form>
        </div>
      @else
        <!-- Si l'usuari NO està autenticat -->
        <a href="{{ route('login') }}" class="text-white bg-blue-500 px-4 py-2 rounded-md hover:bg-blue-600">Login</a>
      @endauth
    </div>
  </div>
</nav>
```

```
<!-- Contingut principal -->
<div class="flex-grow">
  <div class="container mx-auto py-6">
    @yield('content')
  </div>
</div>
```

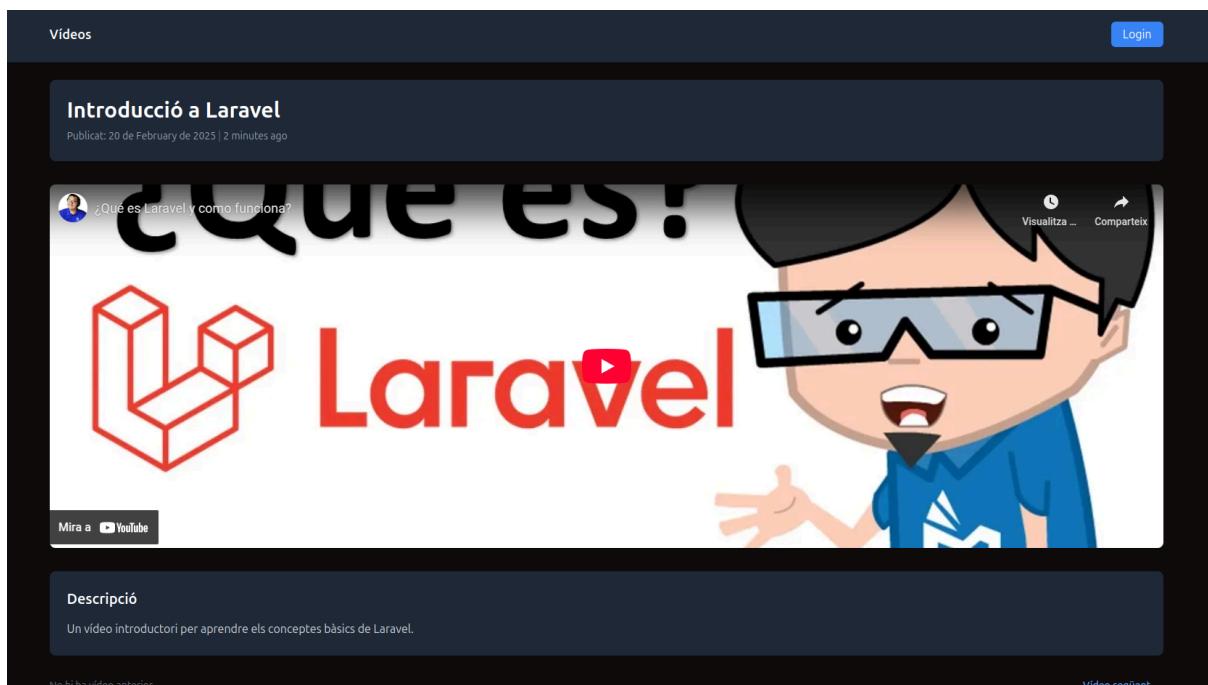
```
<!-- Footer -->
<footer class="bg-gray-800 p-6 mt-auto">
  <div class="container mx-auto text-center text-white">
    © {{ date('Y') }} La Meva Aplicació. Tots els drets reservats.
  </div>
</footer>
```

Resultats finals

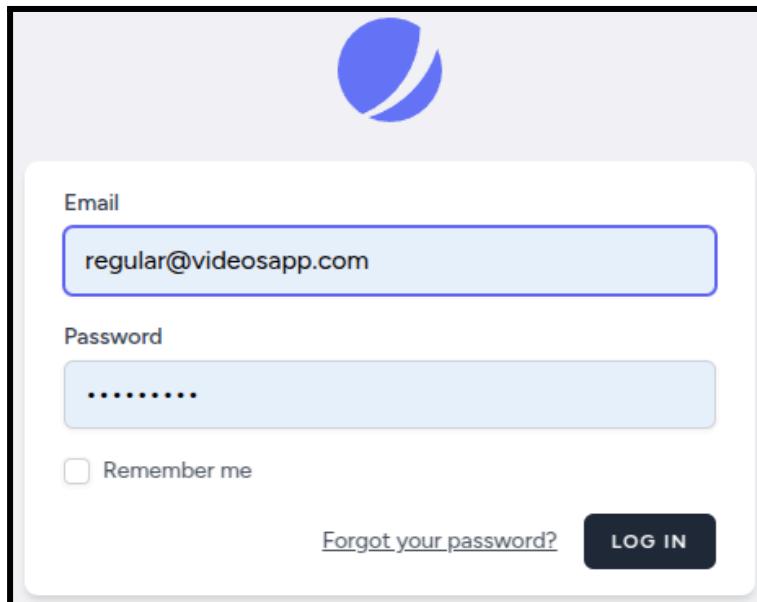
Entro a la pàgina principal i veure els vídeos.



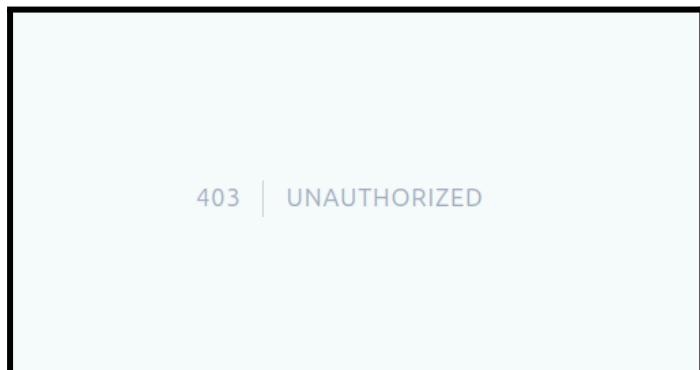
Si li dono clic a un vídeo, veure el vídeo en gran amb detalls.



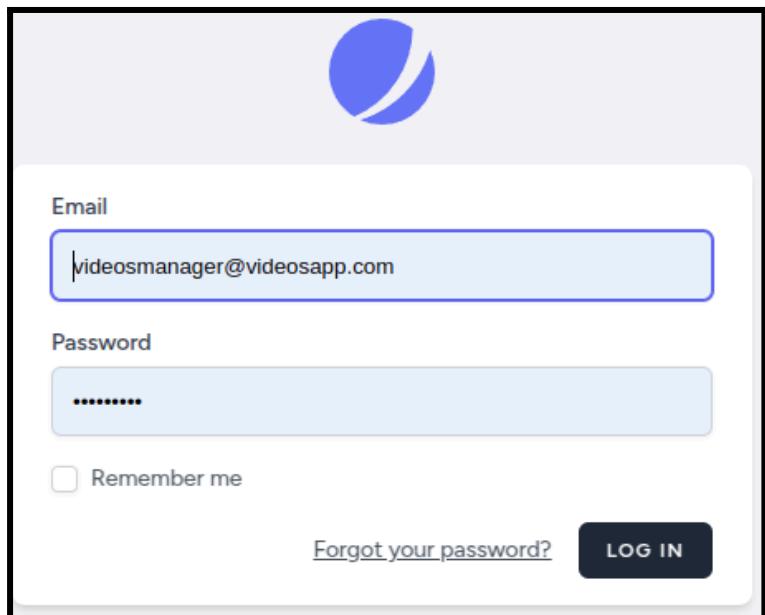
Entro a la ruta /video/manage amb un usuari normal com ara regular.



Veure que no deixa veure el CRUD.



Ara entro amb Vídeos Manager.



Em deixa veure el CRUD.

Gestió de Vídeos

[Afegir Vídeo](#)

ID	Títol	Accions
1	Introducció a Laravel	Editar Eliminar
2	Controladors a Laravel	Editar Eliminar
3	Models a Laravel	Editar Eliminar

Creo un nou vídeo.

Afegir Nou Vídeo

Títol

Descripció

URL

Data de publicació

Aquí el veurem creat.

Gestió de Vídeos

ID	Títol	Accions
1	Introducció a Laravel	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
2	Controladors a Laravel	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
3	Models a Laravel	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
4	Chill Aim	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

Si edito el vídeo posant un “2” i clico actualizar, veurem que és guarda bé.

Títol
Chill Aim 2

Descripció
test 2

URL
<https://www.youtube.com/embed/ddxj4ljlpQg>

Data de publicació
20/02/2025

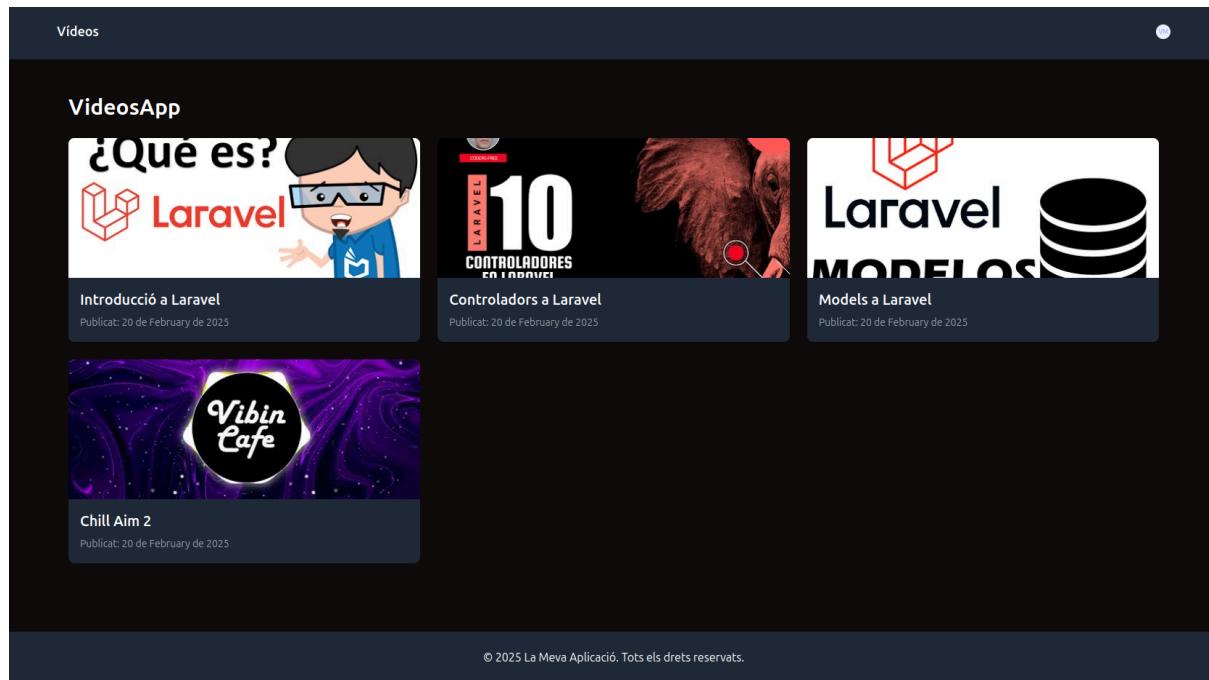
[Tornar Enrere](#) [Actualizar Vídeo](#)

Gestió de Vídeos

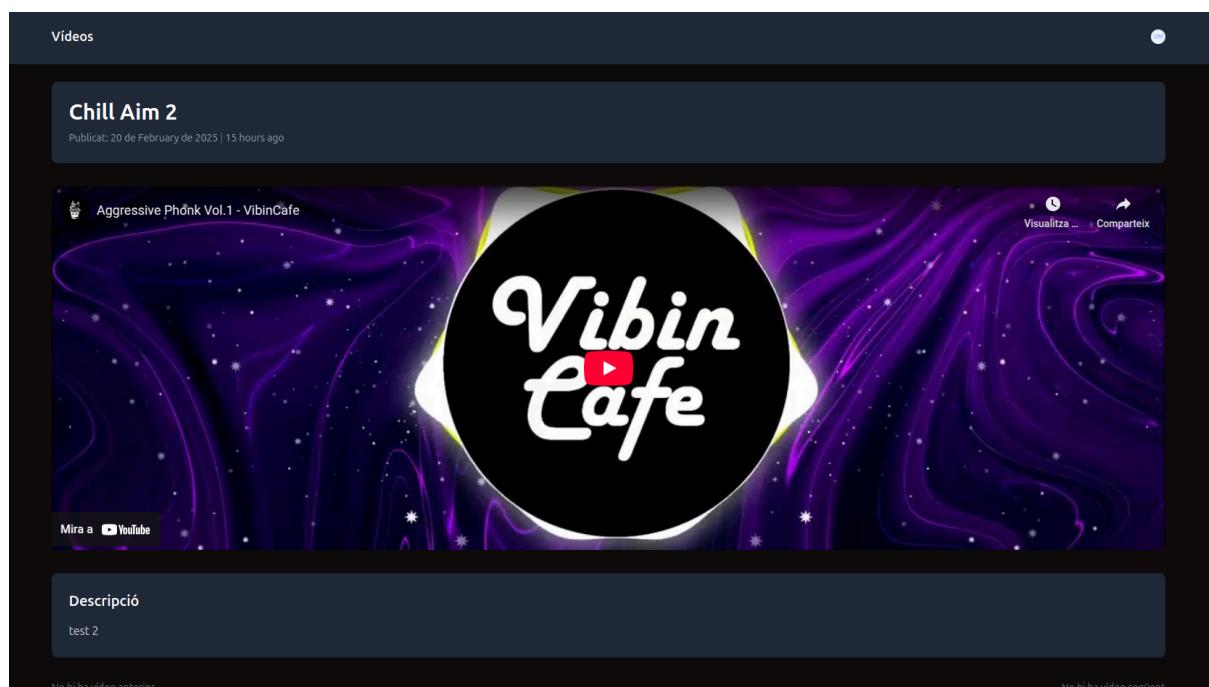
[Afegir Vídeo](#)

ID	Títol	Accions
1	Introducció a Laravel	Editar Eliminar
2	Controladors a Laravel	Editar Eliminar
3	Models a Laravel	Editar Eliminar
4	Chill Aim 2	Editar Eliminar

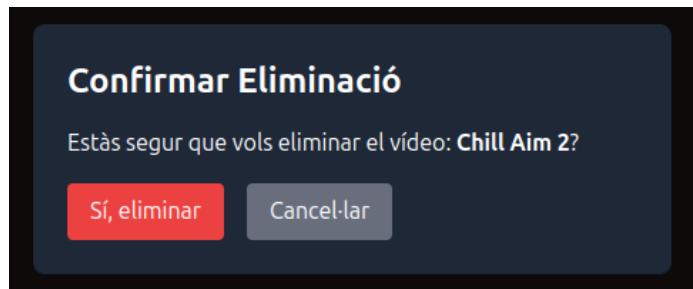
Tornem a la vista principal i veurem el vídeo afegit.



Farem clic al vídeo i el veurem en detall.



Finalment provarem d'eliminar el vídeo, i una vegada eliminat ja no apareix a la llista.



Gestió de Vídeos

Afegir Vídeo

ID	Títol	Accions
1	Introducció a Laravel	Editar Eliminar
2	Controladors a Laravel	Editar Eliminar
3	Models a Laravel	Editar Eliminar

resources/markdown/terms.md

Afegire el que he fet al sprint.

```
M+ terms.md x
53   - Creació de tests per assegurar que només els usuaris amb els permisos adequats poden gestionar vídeos.
54 - **Millors en la intereficie d'usuari:**
55   - Afegir indicadors visuals per mostrar els permisos de l'usuari actual.
56   - Millorar la navegació per a una millor experiència d'usuari.
57 - **Documentació:**
58   - Actualització de la guia del projecte per incloure les noves funcionalitats i instruccions per a la gestió de permisos i programació de tasques.
59
60 ### Sprint 4: Gestió de Vídeos
61
62 - **Correcció d'accés:**
63   - Ajustar accés a `/videos/manage` segons permisos.
64 - **Controlador `VideosManageController`:**
65   - Desenvolupament de mètodes `testedBy`, `index`, `create`, `store`, `show`, `edit`, `update`, `delete`, `destroy`.
66 - **Mètode `index`:**
67   - Afegit a `VideosController` per la pàgina pública.
68 - **Vídeos per defecte:**
69   - Verificació de la creació de 3 vídeos per defecte amb `VideoHelper` i `DatabaseSeeder`.
70 - **Vistes CRUD:**
71   - Creació de vistes (index, create, edit, delete) per a la gestió de vídeos.
72 - **Rutes:**
73   - Definició de rutes CRUD amb middleware d'autenticació.
74 - **Assignació de permisos:**
75   - Assignació de permisos específics als usuaris.
76 - **Tests:**
77   - Desenvolupament de tests seguint TDD i el patró AAA.
78 - **Interfiecie d'usuari:**
79   - Afegida la navegació (Navbar i Footer) a la plantilla principal.
80
```

Errors Larstan

No hi han errors.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ vendor/bin/phpstan analyse --memory-limit=1G
Note: Using configuration file /home/alumnat/Code/M7-8-9/VideoAppHarvey/phpstan.neon.
63/63 [██████████] 100%  
  

[OK] No errors  
  

alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$
```