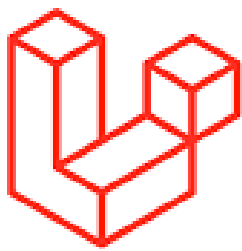


# Projecte GLOBAL: VideosApp

3r SPRINT



# Laravel

**Professor:** Jordi Vega

**Assignatures:** M7, M8 i M9


**Nom:** Harvey

**Cognoms:** John Glover



# INDEX

<b>Corregir els errors del 2n sprint.</b>	<b>3</b>
<b>Instal·lació paquets</b>	<b>3</b>
<b>Migració de taules</b>	<b>4</b>
<b>User.php</b>	<b>5</b>
<b>Userhelper.php</b>	<b>6</b>
<b>PermissionHelper.php</b>	<b>10</b>
<b>AppServiceProvider.php</b>	<b>11</b>
<b>DatabaseSeeder.php</b>	<b>12</b>
<b>VideosManageController.php</b>	<b>15</b>
<b>web.php</b>	<b>15</b>
<b>Publicar els stubs</b>	<b>18</b>
VideosManageControllerTest.php	19
UserTest.php	24
<b>Markdown terms Sprint 3</b>	<b>26</b>
<b>Larastan</b>	<b>27</b>
Error:	27
Solució:	27

 <b>INSTITUT DE L'EBRE</b> TORTOSA	GLOBAL M7, M8 i M9	DAM 2n
--	--------------------	-----------

## Corregir els errors del 2n sprint.

No hi han cap errors.

Retroacció	
Qualificació	10,00 / 10,00
Qualificat el	divendres, 31 de gener 2025, 15:58
Qualificat per	<div>JV</div> Jordi Vega Tomàs [PROF]

## Instal·lació paquets

Executare un composer require que carrgara el repositori i dependències.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ composer require spatie/laravel-permission
./composer.json has been updated
Running composer update spatie/laravel-permission
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking spatie/laravel-permission (6.12.0)
```

Publicare i migrare les taules actualitzades.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

INFO Publishing assets.

Copying file [vendor/spatie/laravel-permission/config/permission.php] to [config/permission.php] ..... DONE
Copying file [vendor/spatie/laravel-permission/database/migrations/create_permission_tables.php.stub] to [database/migrations/2025_02_04_181207_create_permission_tables.php] DONE

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

alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan migrate

INFO Running migrations.

2025_02_04_181207_create_permission_tables ..... 141.24ms DONE

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

## Migració de taules

Creare la migració de la taula per crear el super admin.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan make:migration add_super_admin_to_users_table --table=users

INFO Migration [database/migrations/2025_02_04_181702_add_super_admin_to_users_table.php] created successfully.

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

Dins del document creo 2 funcions, el primer afegeix el super\_admin a true quan detecta que está correcte, sinó, no el posa i fa un drop.

```
php 2025_02_04_181702_add_super_admin_to_users_table.php x
1  <?php
2
3  > use ...
6  no usages
7  class AddSuperAdminToUsersTable extends Migration
8  {
9      public function up()
10     {
11         Schema::table( table: 'users', function (Blueprint $table) {
12             $table->boolean( column: 'super_admin')->default( value: false)->after( column: 'remember_token');
13         });
14     }
15
16     public function down()
17     {
18         Schema::table( table: 'users', function (Blueprint $table) {
19             $table->dropColumn( columns: 'super_admin');
20         });
21     }
22 }
```

Executare el artisan migrate per migrar els canvis.

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

INFO Running migrations.

2025_02_04_181702_add_super_admin_to_users_table ..... 9.92ms DONE

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

## User.php

Al model d'usuaris procuro tenir el camp de `super_admin` i usar el `HasRoles` que gestiona rols i permisos.

```
© User.php x
14  class User extends Authenticatable
20      use Notifiable;
21      use TwoFactorAuthenticatable;
22      use HasRoles; // Trait per gestionar rols i permisos
23
24      /** @var list<string> */
25      no usages
26      protected $fillable = [
27          'name',
28          'email',
29          'password',
30          'current_team_id',
31          'super_admin',
32      ];
```

També afegire la funció `testedBy()` i `isSuperAdmin()`.

```
/** Funció de prova (per exemple, pot retornar informació addicional). */
no usages  ⚠ Wharvey123
public function testedBy(): string
{
    // Exemple: Retorna una cadena de prova o una relació si fos necessari
    return "Tested by " . $this->name;
}

/** Comprova si l'usuari és superadmin. */
no usages  ⚠ Wharvey123
public function isSuperAdmin(): bool
{
    return $this->super_admin === true;
}
```

## Userhelper.php

**createDefaultUser():** Aquesta funció crea o obté un usuari per defecte basant-se en la configuració.

- Obten les dades de l'usuari per defecte.
- Crea o obté un equip (Team) amb l'ID especificat.
- Actualitza o crea l'usuari amb l'email especificat.
- Actualitza l'equip amb el nou ID de l'usuari.
- Retorna l'usuari creat o actualitzat.

```
public static function createDefaultUser()
{
    // Obtenció de dades de l'usuari per defecte des de la configuració
    $userData = config(['key' => 'helpers.default_user']);

    // Creació o obtenció del primer equip (Team) amb l'ID especificat
    $team = Team::firstOrCreate(
        ['id' => $userData['team_id']],
        ['name' => 'Team ' . $userData['team_id'], 'user_id' => $userData['user_id'], 'personal_team' => true]
    );

    // Actualització o creació d'un usuari amb l'email especificat
    $user = User::updateOrCreate(
        ['email' => $userData['email']],
        [
            'name' => $userData['name'],
            'password' => Hash::make($userData['password']),
            'current_team_id' => $userData['team_id'],
        ]
    );

    // Actualització de l'equip amb el nou ID d'usuari
    $team->update(['user_id' => $user->id]);

    return $user;
}
```

**createDefaultProfessor():** Aquesta funció crea o obté un professor per defecte basant-se en la configuració.

- Obten les dades del professor per defecte.
- Crea o obté un equip (Team) amb l'ID especificat.
- Actualitza o crea el professor amb l'email especificat.
- Actualitza l'equip amb el nou ID del professor.
- Comprova si el rol 'superadmin' existeix, si no, el crea.
- Assigna el rol de 'superadmin' al professor.
- Retorna el professor creat o actualitzat.

```
public static function createDefaultProfessor()
{
    // Obtenció de dades del professor per defecte des de la configuració
    $professorData = config( key: 'helpers.default_professor');

    // Creació o obtenció del primer equip (Team) amb l'ID especificat
    $team = Team::firstOrCreate(
        ['id' => $professorData['team_id']],
        ['name' => 'Team ' . $professorData['team_id'], 'user_id' => $professorData['user_id'], 'personal_team' => true]
    );


    // Actualització o creació d'un professor amb l'email especificat
    $professor = User::updateOrCreate(
        ['email' => $professorData['email']],
        [
            'name' => $professorData['name'],
            'password' => Hash::make($professorData['password']),
            'current_team_id' => $professorData['team_id'],
            'super_admin' => true,
        ]
    );

    // Actualització de l'equip amb el nou ID de professor
    $team->update(['user_id' => $professor->id]);

    // Comprovar si el rol 'superadmin' existeix, si no, crear-lo
    $superadminRole = Role::firstOrCreate(['name' => 'superadmin']);

    // Assignar el rol de 'superadmin' al professor
    $professor->assignRole($superadminRole);

    return $professor;
}
```

 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

**add\_personal\_team(User \$user):** Aquesta funció crea o obté un equip personal per a un usuari.

- Crea o obté un equip personal (personal\_team) per a l'usuari especificat.
- L'equip creat porta el nom de l'usuari seguit de "Team".
- Retorna l'equip creat o obtingut.


```
public static function add_personal_team(User $user)
{
    // Creació o obtenció de l'equip personal (personal_team) per a un usuari
    return Team::firstOrCreate(
        [
            'user_id' => $user->id, 'personal_team' => true,
            'name' => $user->name . "'s Team"
        ]
    );
}
```

**create\_superuser\_user():** Aquesta funció crea o obté un usuari superadministrador amb tots els permisos.

- Actualitza o crea un usuari superadministrador amb l'email 'superadmin@videosapp.com'.
- Estableix el nom a 'Super Admin' i una contrasenya específica.
- Estableix l'ID de l'equip actual a 3.
- Marca l'usuari com a super administrador.
- Retorna l'usuari creat o actualitzat.

```
public static function create_superuser_user()
{
    // Actualització o creació d'un usuari superadmin amb tots els permisos
    return User::updateOrCreate(
        [
            'email' => 'superadmin@videosapp.com',
            [
                'name' => 'Super Admin',
                'password' => Hash::make( value: '123456789'),
                'current_team_id' => 3,
                'super_admin' => true,
            ]
        ]
    );
}
```



 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

**create\_regular\_user():** Aquesta funció crea o obté un usuari regular.

- Actualitza o crea un usuari regular amb l'email 'regular@videosapp.com'.
- Estableix el nom a 'Regular' i una contrasenya específica.
- Estableix l'ID de l'equip actual a 4.
- Retorna l'usuari creat o actualitzat.

```
public static function create_regular_user()
{
    // Actualització o creació d'un usuari regular
    return User::updateOrCreate(
        ['email' => 'regular@videosapp.com'],
        [
            'name' => 'Regular',
            'password' => Hash::make( value: '123456789'),
            'current_team_id' => 4,
        ]
    );
}
```

**create\_video\_manager\_user():** Aquesta funció crea o obté un usuari gestor de vídeos.

- Actualitza o crea un usuari gestor de vídeos amb l'email 'videosmanager@videosapp.com'.
- Estableix el nom a 'Video Manager' i una contrasenya específica.
- Estableix l'ID de l'equip actual a 5.
- Retorna l'usuari creat o actualitzat.

```
public static function create_video_manager_user()
{
    // Actualització o creació d'un usuari gestor de vídeos
    return User::updateOrCreate(
        ['email' => 'videosmanager@videosapp.com'],
        [
            'name' => 'Video Manager',
            'password' => Hash::make( value: '123456789'),
            'current_team_id' => 5,
        ]
    );
}
```

## PermissionHelper.php

Creare el fitxer de PermissionHelper.php.

```
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ touch app/Helpers/PermissionHelper.php
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ ls app/Helpers/
PermissionHelper.php  UserHelper.php  VideoHelper.php
aLumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$
```

**define\_gates():**

- Defineix una porta d'accés 'manage-videos'.
- Permet gestionar vídeos si l'usuari té el permís o és superadministrador.

**create\_permissions():**

- Crea el permís 'manage videos' si no existeix ja.

```
PermissionHelper.php x
1  <?php
2
3  namespace App\Helpers;
4
5  > use ...
6
7  6 usages
8  class PermissionHelper
9  {
10     /** Defineix les portes d'accés per a la gestió de vídeos. */
11     1 usage
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. */
21     2 usages
22     public static function create_permissions(): void
23     {
24         Permission::firstOrCreate(['name' => 'manage videos']);
25     }
26 }
```

## AppServiceProvider.php

**Classe AppServiceProvider:** Aquesta classe és un proveïdor de serveis d'autenticació per a la teva aplicació.

**boot():**

- Registra les polítiques d'autorització si n'hi ha, mitjançant `$this->registerPolicies()`.
- Defineix les portes d'accés cridant la funció `define_gates()` de `PermissionHelper`.

Aquest codi s'assegura que les polítiques d'autorització estiguin registrades i que les portes d'accés per a la gestió de permisos estiguin definides. Això permet controlar l'accés i els permisos dins de la teva aplicació.

```
© AppServiceProvider.php x
1  <?php
2
3  namespace App\Providers;
4
5  use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;
6  use App\Helpers\PermissionHelper;
7
8  3 usages  ⚡ Wharvey123 *
9  class AppServiceProvider extends ServiceProvider
10 {
11     /** Registra les polítiques d'autorització. */
12     no usages  ⚡ Wharvey123 *
13     public function boot(): void
14     {
15         // Registra les polítiques, si n'hi ha
16         $this->registerPolicies();
17
18         // Defineix les portes d'accés
19         PermissionHelper::define_gates();
20     }
21 }
```

## DatabaseSeeder.php

**Crear permisos:** Crida a `create_permissions()` per crear els permisos necessaris.

**Crear usuaris:** Crea diferents tipus d'usuaris sense assignar un `current_team_id` inicial.

**Assignar equips personals:** Assigna equips personals a cada usuari, establint així el `current_team_id`.


**Assignar permisos segons rol:** Assigna el permís per gestionar vídeos als usuaris segons el seu rol en aquest cas tindran permís `defaultProfessor`, `superadmin` i `videoManager`.

**Crear vídeos per defecte:** Crea vídeos per defecte cridant a `createDefaultVideos()`.

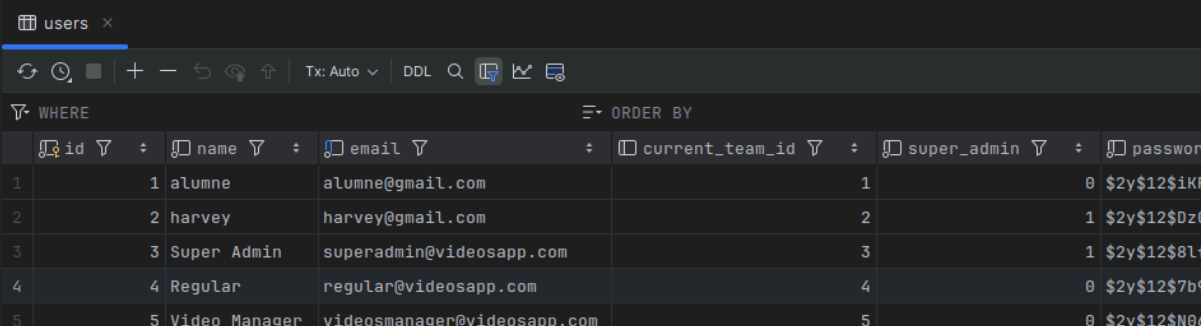
```

© DatabaseSeeder.php x
1  <?php
2
3  namespace Database\Seeders;
4
5  > use ...
9
no usages  Wharvey123 *
10 class DatabaseSeeder extends Seeder
11 {
12     Wharvey123 *
13     public function run(): void
14     {
15         // Crear permisos
16         PermissionHelper::create_permissions();
17
18         // Crear usuaris sense assignar current_team_id (aquest es farà a add_personal_team)
19         $defaultUser = UserHelper::createDefaultUser();
20         $defaultProfessor = UserHelper::createDefaultProfessor();
21         $superadmin = UserHelper::create_superadmin_user();
22         $regularUser = UserHelper::create_regular_user();
23         $videoManager = UserHelper::create_video_manager_user();
24
25         // Assignar equips personals i, per tant, current_team_id = user id
26         UserHelper::add_personal_team($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 (exemple)
33         $defaultProfessor->givePermissionTo('manage videos');
34         $superadmin->givePermissionTo('manage videos');
35         $videoManager->givePermissionTo('manage videos');
36
37         // Crear vídeos per defecte
38         VideoHelper::createDefaultVideos();
39     }
40 }

```

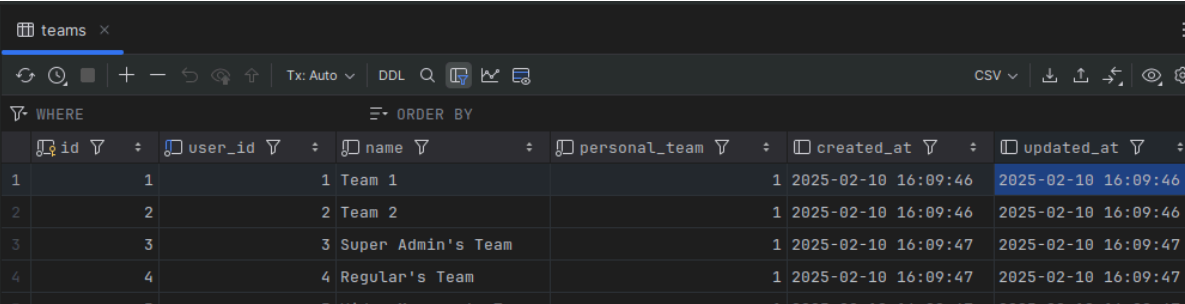
 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

La taula d'usuaris té 5 usuaris que son 2 regulars, 2 amb permisos de super admin i 1 video manager amb permisos per veure els vídeos.



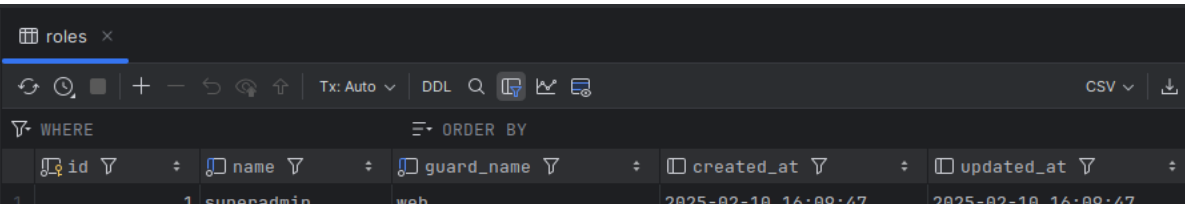
	id	name	email	current_team_id	super_admin	password
1	1	alumne	alumne@gmail.com	1	0	\$2y\$12\$iKR
2	2	harvey	harvey@gmail.com	2	1	\$2y\$12\$DzG
3	3	Super Admin	superadmin@videosapp.com	3	1	\$2y\$12\$8lf
4	4	Regular	regular@videosapp.com	4	0	\$2y\$12\$7b9
5	5	Video Manager	videosmanager@videosapp.com	5	0	\$2y\$12\$N06

Els equips és creen amb els noms adequats.

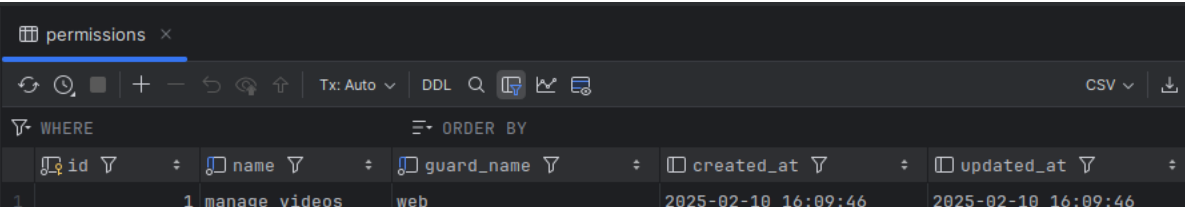


	id	user_id	name	personal_team	created_at	updated_at
1	1	1	Team 1	1	2025-02-10 16:09:46	2025-02-10 16:09:46
2	2	2	Team 2	1	2025-02-10 16:09:46	2025-02-10 16:09:46
3	3	3	Super Admin's Team	1	2025-02-10 16:09:47	2025-02-10 16:09:47
4	4	4	Regular's Team	1	2025-02-10 16:09:47	2025-02-10 16:09:47
5	5	5	Video Manager's Team	1	2025-02-10 16:09:47	2025-02-10 16:09:47


Les taules de rols i permissions veurem el rol de superadmin i el permís d'observació de vídeos que s'anomena manage videos.



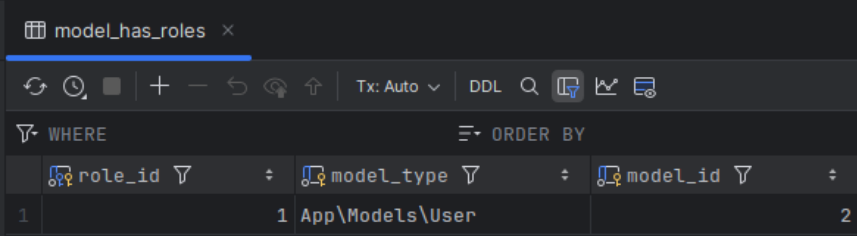
	id	name	guard_name	created_at	updated_at
1	1	superadmin	web	2025-02-10 16:09:47	2025-02-10 16:09:47



	id	name	guard_name	created_at	updated_at
1	1	manage videos	web	2025-02-10 16:09:46	2025-02-10 16:09:46

 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

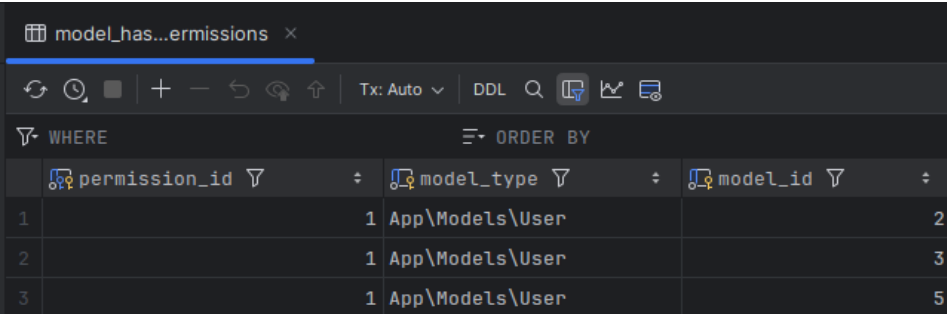
L'usuari que té assignat el rol de super admin en aquest cas és el default professor que és l'id 2.



model\_has\_roles

	role_id	model_type	model_id
1	1	App\Models\User	2

Els usuaris que tenen permisos per visualitzar els vídeos son 2,3 i 5 o en altres paraules default professor, superadmin i video manager.



model\_has\_permissions

	permission_id	model_type	model_id
1	1	App\Models\User	2
2	1	App\Models\User	3
3	1	App\Models\User	5

## VideosManageController.php

Creo el controlador amb la comanda php artisan make.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan make:controller VideosManageController

INFO Controller [app/Http/Controllers/VideosManageController.php] created successfully.

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

Dins del controlador tindre una funció de manage que comprovarà si l'usuari té permís per gestionar vídeos, i en cas de que pugui, mostra un missatge i en cas de que no, un error 403.


```

VideosManageController.php x
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class VideosManageController extends Controller
8  {
9      public function manage()
10     {
11         // Comprova si l'usuari té permís per gestionar vídeos
12         if (auth()->user()->can('abilities: manage-videos')) {
13             return response()->json(['message' => 'User can manage videos.'], status: 200);
14         }
15         abort( code: 403, message: 'Unauthorized');
16     }
17 }
```

## web.php

Estableixo una ruta que serà /videos/manage per comprovar si té permís l'usuari






```
Route::middleware(['auth'])->group(function () {
    Route::get( uri: '/videos/manage', [VideosManageController::class, 'manage'])->name( name: 'videos.manage');
});
```

 <b>INSTITUT DE L'EBRE</b> TORTOSA	GLOBAL M7, M8 i M9	DAM 2n
--	--------------------	-----------

Els 3 usuaris que tenen permís per visualitzar la pàgina amb el missatge son superadmin (Super Admin), videomanager (Video Manager) i defaultprofessor (harvey)

Super Admin's Team ↕
 Super Admin ▼

← ↻ ⓘ 127.0.0.1:8000/videos/manage






 chess
  iEduca
  Moodle
  Drive
  Gmail

Impressió fàcil ☐

```
{"message": "User can manage videos."}
```

Video Manager's Team ↕
 Video Manager ▼

← ↻ ⓘ 127.0.0.1:8000/videos/manage






 chess
  iEduca
  Moodle
  Drive
  Gmail

Impressió fàcil ☐

```
{"message": "User can manage videos."}
```

Team 2 ↕
 harvey ▼


← ↻ ⓘ 127.0.0.1:8000/videos/manage

 chess
  iEduca
  Moodle
  Drive
  Gmail

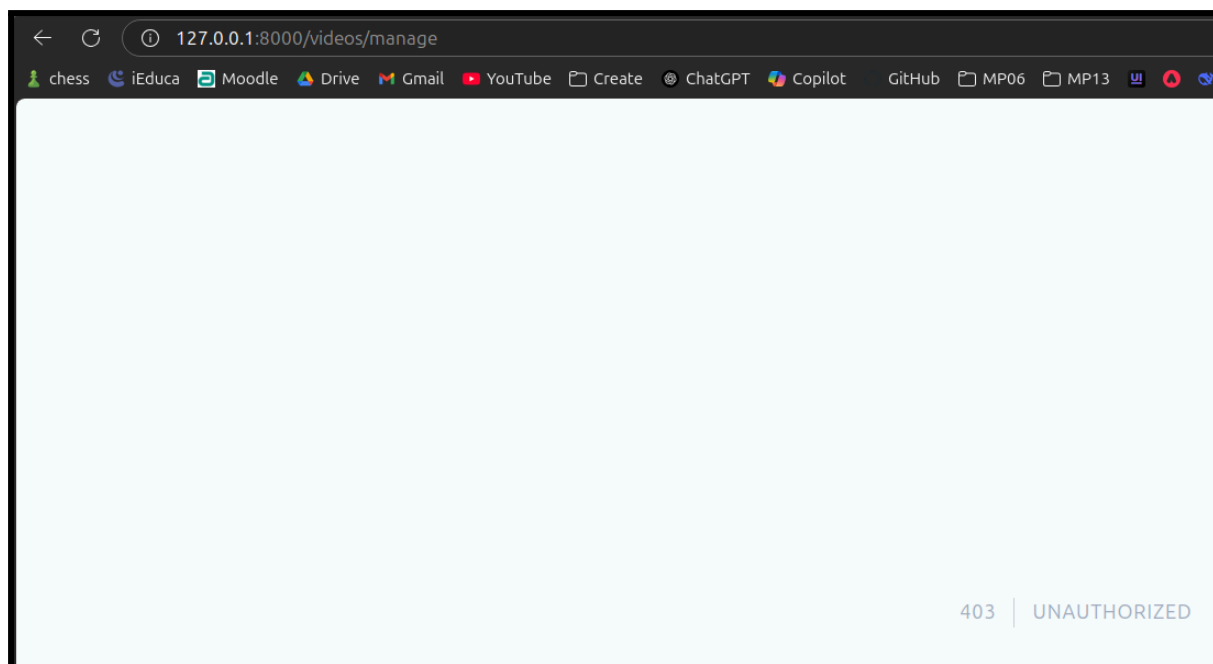
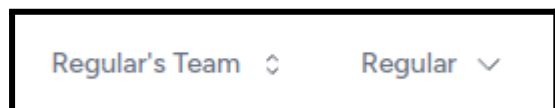
Impressió fàcil ☐

```
{"message": "User can manage videos."}
```



 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

Després si provem amb un usuari que no té accés com ara el regular, ens mostra el error 403 unauthorized.



## Publicar els stubs

Publicare els stubs amb la comanda **php artisan stub:publish** i després els afegire amb git add stubs.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan stub:publish  
  
INFO Stubs published successfully.  
  
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ git add stubs
```

Amb git status veure els canvis.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ git status  
En la branca main  
La vostra branca està al dia amb «origin/main».  
  
Canvis a cometre:  
(useu «git restore --staged <fitxer>...» per a fer «unstage»)  
  fitxer nou:      stubs/cast.inbound.stub  
  fitxer nou:      stubs/cast.stub  
  fitxer nou:      stubs/class.invokable.stub  
  fitxer nou:      stubs/class.stub  
  fitxer nou:      stubs/console.stub
```

# Tests

## VideosManageControllerTest.php

Creare el test VideosManageControllerTest a la carpeta tests/ Feature/Videos.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan make:test VideosManageControllerTest

INFO Test [tests/Feature/VideosManageControllerTest.php] created successfully.


alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ mv tests/Feature/VideosManageControllerTest.php tests/Feature/Videos/
```

**Classe VideosManageControllerTest:** Aquesta classe conté proves de característiques per assegurar-se que els usuaris tenen els permisos correctes per gestionar vídeos.

1. **setUp():**
  - Configura l'entorn de prova abans de cada test.
  - Crea el permís 'manage videos'.
2. **loginAsVideoManager():**
  - Crea un usuari gestor de vídeos.
  - Assigna el permís 'manage videos' a l'usuari.
  - Autentica l'usuari per a les proves.
3. **loginAsSuperAdmin():**
  - Crea un usuari superadministrador.
  - Assigna el permís 'manage videos' a l'usuari.
  - Autentica l'usuari per a les proves.
4. **loginAsRegularUser():**
  - Crea un usuari regular.
  - Autentica l'usuari per a les proves.
5. **test\_user\_with\_permissions\_can\_manage\_videos():**
  - Verifica que un usuari amb permisos pot gestionar vídeos.
  - Comprova que la resposta és 200 OK.
6. **test\_regular\_users\_cannot\_manage\_videos():**
  - Verifica que un usuari regular no pot gestionar vídeos.
  - Comprova que la resposta és 403 Forbidden.
7. **test\_guest\_users\_cannot\_manage\_videos():**
  - Verifica que un usuari convidat no pot gestionar vídeos.
  - Comprova que es redirigeix a la pàgina de login.
8. **test\_superadmins\_can\_manage\_videos():**
  - Verifica que un superadministrador pot gestionar vídeos.
  - Comprova que la resposta és 200 OK.

```
VideosManageControllerTest.php x
new *
10 class VideosManageControllerTest extends TestCase
11 {
12     use RefreshDatabase;
13
14     nou usages new *
15     protected function setUp(): void
16     {
17         parent::setUp();
18         // Crea el permís abans de cada test
19         PermissionHelper::create_permissions();
20     }
21
22     1 usage new *
23     protected function loginAsVideoManager()
24     {
25         $user = UserHelper::create_video_manager_user();
26         $user->givePermissionTo('manage videos');
27         $this->actingAs($user);
28         return $user;
29     }
30
31     1 usage new *
32     protected function loginAsSuperAdmin()
33     {
34         $user = UserHelper::create_superadmin_user();
35         $user->givePermissionTo('manage videos');
36         $this->actingAs($user);
37         return $user;
38     }
39
40     1 usage new *
41     protected function loginAsRegularUser()
42     {
43         $user = UserHelper::create_regular_user();
44         $this->actingAs($user);
45         return $user;
46     }
47 }
```

```
new *
44 public function test_user_with_permissions_can_manage_videos()
45 {
46     $this->loginAsVideoManager();
47     $response = $this->get(route( name: 'videos.manage'));
48     $response->assertStatus( status: 200);
49 }
50
new *
51 public function test_regular_users_cannot_manage_videos()
52 {
53     $this->loginAsRegularUser();
54     $response = $this->get(route( name: 'videos.manage'));
55     $response->assertStatus( status: 403);
56 }
57
new *
58 public function test_guest_users_cannot_manage_videos()
59 {
60     $response = $this->get(route( name: 'videos.manage'));
61     $response->assertRedirect(route( name: 'login'));
62 }
63
new *
64 public function test_superadmins_can_manage_videos()
65 {
66     $this->loginAsSuperAdmin();
67     $response = $this->get(route( name: 'videos.manage'));
68     $response->assertStatus( status: 200);
69 }
70 }
71
```

 <b>INSTITUT DE L'EBRE</b> TORTOSA	GLOBAL M7, M8 i M9	DAM 2n
--	--------------------	-----------


## Resultat

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

PASS Tests\Feature\Videos\VideosManageControllerTest
✓ user with permissions can manage videos
✓ regular users cannot manage videos
✓ guest users cannot manage videos
✓ superadmins can manage videos

Tests: 4 passed (5 assertions)
Duration: 0.19s

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

 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

## Metodología TDD i AAA:

Aquest fitxer conté proves de funcionalitat per al controlador de gestió de vídeos. Cada mètode de prova segueix el patró AAA de la següent manera:

1. **test\_user\_with\_permissions\_can\_manage\_videos**
  - **Arrange (Preparació):** Es crea un usuari amb el rol de gestor de vídeos mitjançant el mètode `loginAsVideoManager()`. Aquest mètode crea un usuari amb els permisos necessaris i inicia sessió com aquest usuari.
  - **Act (Acció):** Es realitza una petició GET a la ruta `videos.manage` utilitzant el mètode `get(route('videos.manage'))`. Això simula que l'usuari intenta accedir a la pàgina de gestió de vídeos.
  - **Assert (Afirmació):** Es verifica que la resposta té un estatus HTTP 200 (`assertStatus(200)`), la qual cosa indica que l'usuari amb permisos pot accedir correctament a la pàgina de gestió de vídeos.
2. **test\_regular\_users\_cannot\_manage\_videos**
  - **Arrange (Preparació):** Es crea un usuari regular sense permisos especials mitjançant el mètode `loginAsRegularUser()`. Aquest mètode crea un usuari estàndard i inicia sessió com aquest usuari.
  - **Act (Acció):** Es realitza una petició GET a la mateixa ruta `videos.manage`.
  - **Assert (Afirmació):** Es comprova que la resposta té un estatus HTTP 403 (`assertStatus(403)`), indicant que l'usuari sense els permisos necessaris no pot accedir a la pàgina de gestió de vídeos.
3. **test\_guest\_users\_cannot\_manage\_videos**
  - **Arrange (Preparació):** En aquest cas, no es crea cap usuari ni es realitza cap inici de sessió, simulant un usuari convidat (no autenticat).
  - **Act (Acció):** Es fa una petició GET a la ruta `videos.manage`.
  - **Assert (Afirmació):** Es verifica que la resposta redirigeix a la pàgina d'inici de sessió (`assertRedirect(route('login'))`), la qual cosa indica que els usuaris no autenticats són redirigits quan intenten accedir a la pàgina de gestió de vídeos.
4. **test\_superadmins\_can\_manage\_videos**
  - **Arrange (Preparació):** Es crea un usuari amb rol de superadministrador mitjançant el mètode `loginAsSuperAdmin()`. Aquest mètode crea un superadministrador i inicia sessió com aquest usuari.
  - **Act (Acció):** Es realitza una petició GET a la ruta `videos.manage`.
  - **Assert (Afirmació):** Es comprova que la resposta té un estatus HTTP 200 (`assertStatus(200)`), indicant que el superadministrador pot accedir a la pàgina de gestió de vídeos sense restriccions.

## UserTest.php

Creare el test UserTest a la carpeta tests/Unit.

```
alumnat@harvey:~/Code/M7-8-9/VideoAppHarvey$ php artisan make:test UserTest --unit

INFO Test [tests/Unit/UserTest.php] created successfully.


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

**Classe UserTest:** Aquesta classe conté proves unitàries per verificar la funcionalitat dels usuaris.

1. **test\_is\_super\_admin():**
  - **Arrange:** Crea un usuari superadministrador i un usuari regular utilitzant UserHelper.
  - **Act & Assert:**
    - Comprova que el mètode isSuperAdmin() del superadministrador retorna true.
    - Comprova que el mètode isSuperAdmin() de l'usuari regular retorna false.

```
UserTest.php x
1  <?php
2
3  namespace Tests\Unit;
4
5  > use ...
6
7  new *
8
9  class UserTest extends TestCase
10 {
11     use RefreshDatabase;
12
13     new *
14     public function test_is_super_admin()
15     {
16         // Arrange
17         $superadmin = UserHelper::create_superadmin_user();
18         $regular = UserHelper::create_regular_user();
19
20         // Act & Assert
21         $this->assertTrue($superadmin->isSuperAdmin(), 'message: 'El superadmin ha de retornar true en isSuperAdmin()');
22         $this->assertFalse($regular->isSuperAdmin(), 'message: 'Un usuari regular ha de retornar false en isSuperAdmin()');
23     }
24 }
```



 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

## Resultat

```

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

PASS Tests\Unit\UserTest
✓ is super admin

Tests: 1 passed (2 assertions)
Duration: 0.15s

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

```

## Metodología TDD i AAA:

Aquest fitxer conté proves unitàries per a la classe User. El mètode test\_is\_super\_admin segueix el patró AAA de la següent manera:

- **Arrange (Preparació):** Es creen dues instàncies d'usuari: una amb el rol de superadministrador (\$superadmin = UserHelper::create\_superadmin\_user()) i una altra com a usuari regular (\$regular = UserHelper::create\_regular\_user()).
- **Act (Acció):** Es crida al mètode isSuperAdmin() per a cadascun dels usuaris creats.
- **Assert (Afirmació):** Es verifica que el mètode retorna true per al superadministrador (\$this->assertTrue(\$superadmin->isSuperAdmin())) i false per a l'usuari regular (\$this->assertFalse(\$regular->isSuperAdmin())). Això assegura que el mètode isSuperAdmin() funciona correctament segons el tipus d'usuari.

## Markdown terms Sprint 3


Afegir a `resources/markdown/terms` el que heu fet al sprint.

```
### Sprint 3: Gestió de permisos i programació de tasques
```

```
- **Gestió de permisos:**  
  - Implementació del paquet 'spatie/laravel-permission' per a la gestió de rols i permisos.  
  - Creació dels rols Super Admin, Video Manager i Regular User.  
  - Assignació de permisos específics a cada rol, com ara 'manage videos' per al Video Manager.  
- **Programació de tasques:**  
  - Configuració de tasques programades utilitzant el planificador de Laravel per a operacions periòdiques, com ara la neteja de vídeos antics.  
- **Tests de permisos:**  
  - Creació de tests per assegurar que només els usuaris amb els permisos adequats poden gestionar vídeos.  
- **Millores en la interfície d'usuari:**  
  - Afegir indicadors visuals per mostrar els permisos de l'usuari actual.  
  - Millorar la navegació per a una millor experiència d'usuari.  
- **Documentació:**  
  - Actualització de la guia del projecte per incloure les noves funcionalitats i instruccions per a la gestió de permisos i programació de tasques.
```

### Sprint 3: Gestió de permisos i programació de tasques

- **Gestió de permisos:**
  - Implementació del paquet `spatie/laravel-permission` per a la gestió de rols i permisos.
  - Creació dels rols Super Admin, Video Manager i Regular User.
  - Assignació de permisos específics a cada rol, com ara `manage videos` per al Video Manager.
- **Programació de tasques:**
  - Configuració de tasques programades utilitzant el planificador de Laravel per a operacions periòdiques, com ara la neteja de vídeos antics.
- **Tests de permisos:**
  - Creació de tests per assegurar que només els usuaris amb els permisos adequats poden gestionar vídeos.
- **Millores en la interfície d'usuari:**
  - Afegir indicadors visuals per mostrar els permisos de l'usuari actual.
  - Millorar la navegació per a una millor experiència d'usuari.
- **Documentació:**
  - Actualització de la guia del projecte per incloure les noves funcionalitats i instruccions per a la gestió de permisos i programació de tasques.

 <b>INSTITUT DE L'EBRE</b> TORTOSA	<b>GLOBAL M7, M8 i M9</b>	<b>DAM</b> <b>2n</b>
--	---------------------------	-------------------------

## Larastan

Comprovar en Larastan tots els fitxers que heu creat.

### Error:

El error ens diu que el procés de anàlisis ha crashejat ja que ha arribat al límit de memòria.

```

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

-----
Error
-----

Child process error: PHPStan process crashed because it reached
configured PHP memory limit: 128M
Increase your memory limit in php.ini or run PHPStan with --memory-limit
CLI option.
while running parallel worker
-----

[ERROR] Found 1 error

```

### Solució:

La solució és especificar una memòria més gran, i per tant ja podeu veure el anàlisis correcte i veurem que 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

```