

# VIDEOSAPP

2024/25

## 3r SPRINT

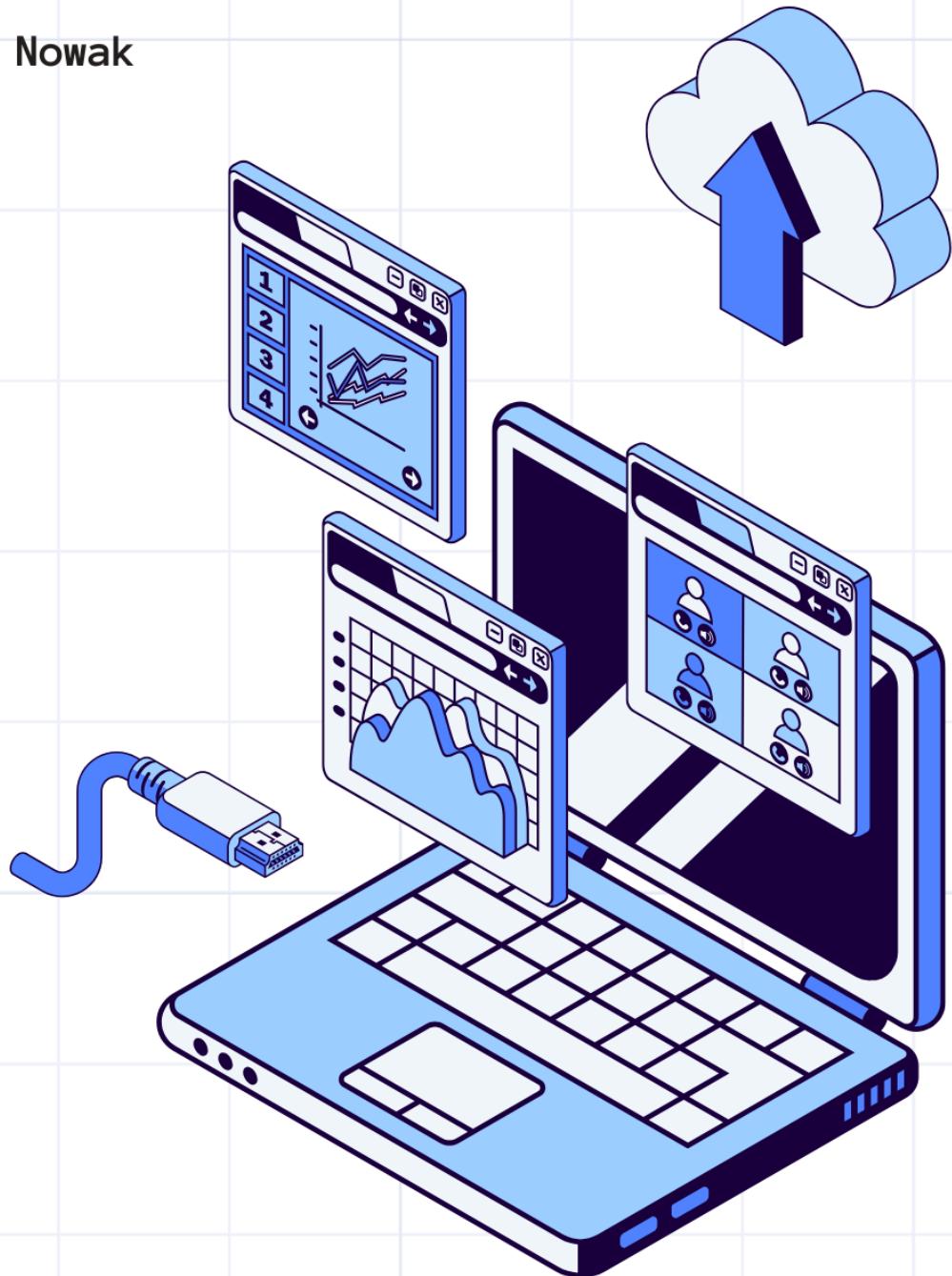
Projecte Global.

MP07 Desenvolupament d'interfícies

MP08 Programació multimèdia i dispositius mòbils

MP09 Programació de serveis i processos

Natalia Nowak



# Índex

Correcció d'errors	2
Instal·lació paquet	3
Creació de migracions	5
Modificació model d'usuaris	6
Crear i afegir funcions	7
Registrar polítiques d'autorització	9
Permissos DatabaseSeeder	10
Publicar stubs	11
Creació de VideoManageController	12
Creació UserTest	14
Markdown	15
Testos php artisan	16
Test Larastan	18

# Correcció d'errors

- Canviar el link dels vídeos per un iframe

```
public static function createDefaultVideos(): void
{
    Video::query()->create([
        'title' => 'Video 1 - Fire - BTS',
        'description' => 'Video de la canción Fire de BTS.<br><iframe width="560" height="315" src="https://www.youtube.com/embed/4ujQ0R2DMFM" frameborder="0" allowfullscreen></iframe>',
        'url' => 'https://www.youtube.com/watch?v=4ujQ0R2DMFM',
        'published_at' => now(),
    ]);

    Video::query()->create([
        'title' => 'Video 2 - Mic Drop - BTS',
        'description' => 'Video de la canción Mic Drop de BTS.<br><iframe width="560" height="315" src="https://www.youtube.com/embed/kTlv5_Bs8aw" frameborder="0" allowfullscreen></iframe>',
        'url' => 'https://www.youtube.com/watch?v=kTlv5_Bs8aw',
        'published_at' => now(),
    ]);
}
```

- Comprovar en HelpersTest la creació del vídeo per defecte:

```
public function test_default_video_creation(): void
{
    DefaultVideoHelper::createDefaultVideos();

    $video1 = Video::query()->where('title', 'Video 1 - Fire - BTS')->first();
    $video2 = Video::query()->where('title', 'Video 2 - Mic Drop - BTS')->first();

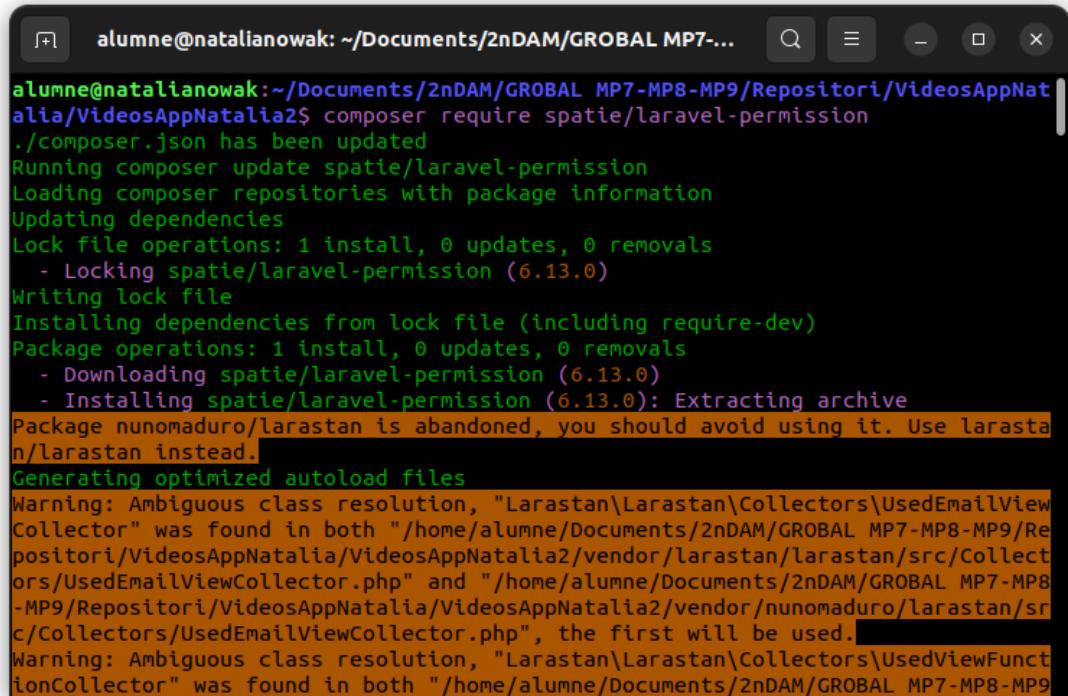
    $this->assertNotNull($video1);
    $this->assertEquals('Video de la canción Fire de BTS.<br><iframe width="560" height="315" src="https://www.youtube.com/embed/4ujQ0R2DMFM" frameborder="0" allowfullscreen></iframe>', $video1->description);
}
```

- testedBy() retorna la classe del test

```
public function testedBy(): string
{
    return self::class;
}
```

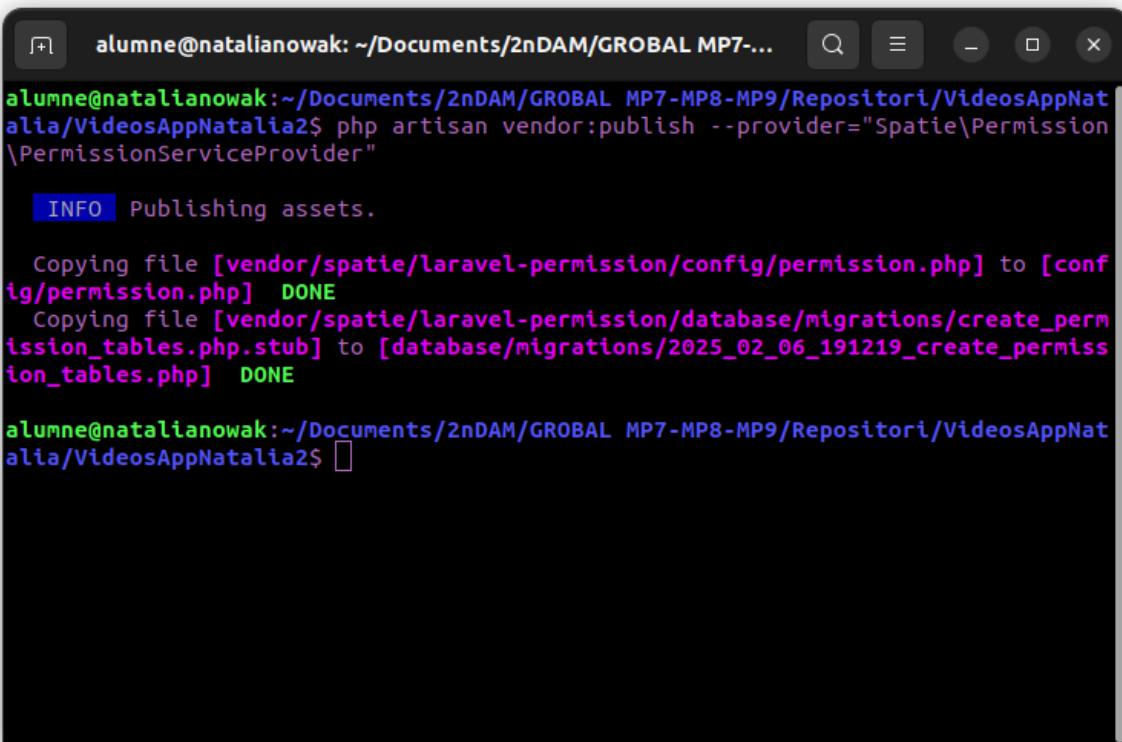
# Instal·lació paquet

- Instal·lem el paquet spatie/larastan



```
alumne@natalianowak: ~/Documents/2nDAM/GROBAL MP7-... alumne@natalianowak: ~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ 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 (6.13.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading spatie/laravel-permission (6.13.0)
- Installing spatie/laravel-permission (6.13.0): Extracting archive
Package nunomaduro/larastan is abandoned, you should avoid using it. Use larastan/larastan instead.
Generating optimized autoload files
Warning: Ambiguous class resolution, "Larastan\Larastan\Collectors\UsedEmailViewCollector" was found in both "/home/alumne/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2/vendor/larastan/larastan/src/Collectors/UsedEmailViewCollector.php" and "/home/alumne/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2/vendor/nunomaduro/larastan/src/Collectors/UsedEmailViewCollector.php", the first will be used.
Warning: Ambiguous class resolution, "Larastan\Larastan\Collectors\UsedViewFunctionCollector" was found in both "/home/alumne/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2/vendor/nunomaduro/larastan/src/Collectors/UsedViewFunctionCollector.php", the first will be used.
```

- Publiquem els assets corresponents del paquet



```
alumne@natalianowak: ~/Documents/2nDAM/GROBAL MP7-... alumne@natalianowak: ~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ php artisan vendor:publish --provider="Spatie\Permission\ServiceProvider"
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.stub] to [database/migrations/2025_02_06_191219_create_permission_tables.php] DONE
alumne@natalianowak: ~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ 
```

- Realitzem totes les migracions de nou

```
alumne@natalianowak:~/Documents/2nDAM/GROBAL_MP7-MP8-MP9/VideosApp2PerqueLaltreNoVa/VideosAppNatalia-main/VideosAppNatalia2$ php artisan migrate
    APPLICATION IN PRODUCTION.

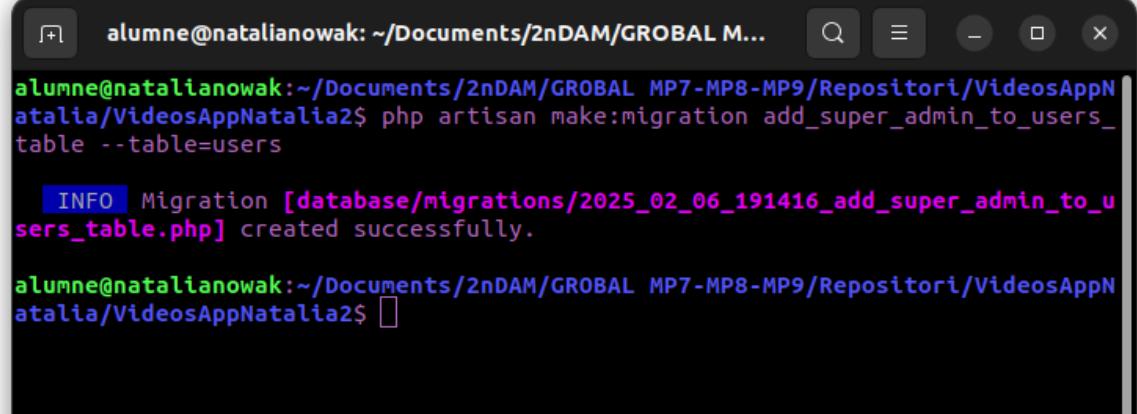
Are you sure you want to run this command? [Yes]
[INFO] Preparing database.
Creating migration table ..... 4.54ms DONE
[INFO] Running migrations.

0001_01_01_000000_create_users_table ..... 17.03ms DONE
0001_01_01_000001_create_cache_table ..... 5.87ms DONE
0001_01_01_000002_create_jobs_table ..... 13.94ms DONE
2025_01_14_145708_add_two_factor_columns_to_users_table ..... 6.56ms DONE
2025_01_14_145728_create_personal_access_tokens_table ..... 8.11ms DONE
2025_01_14_145728_create_teams_table ..... 5.71ms DONE
2025_01_14_145729_create_team_user_table ..... 5.84ms DONE
2025_01_14_145730_create_team_invitations_table ..... 7.97ms DONE
2025_01_22_165648_create_videos_table ..... 3.89ms DONE
2025_02_13_142453_create_permission_tables ..... 26.98ms DONE

alumne@natalianowak:~/Documents/2nDAM/GROBAL_MP7-MP8-MP9/VideosApp2PerqueLaltreNoVa/VideosAppNatalia-main/VideosAppNatalia2$
```

# Creació de migracions

- Creem la migració corresponent per a afegir super\_admin a la taula d'usuaris



```
alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ php artisan make:migration add_super_admin_to_users_table --table=users
INFO Migration [database/migrations/2025_02_06_191416_add_super_admin_to_users_table.php] created successfully.

alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$
```

- Configurem la migració per a que cree el camp i que estigui en *false* de forma predeterminada

```
public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->boolean('is_superuser')->default(false)->after('password');
    });
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::table('users', function (Blueprint $table) {
        $table->dropColumn('is_superuser');
    });
}
```

## Modificació model d'usuaris

- En el model User.php creem les dos següents funcions:

```
public function testedBy(): string
{
    return self::class;
}

public function isSuperAdmin(): bool
{
    return $this->super_admin;
}
```

# Crear i afegir funcions

- Creem les següents funcions al DefaultUserHelpers

```
public static function create_default_professor(): User
{
    $user = self::create_user('Professor', 'professor@videosapp.com');
    $user->assignRole('Professor');

    // Opcionalmente, si quieres que el profesor también sea superadmin
    if (config('app.professors_are_superadmins', false)) {
        $user->assignRole('Super Admin');
    }

    return $user;
}

public static function add_personal_team(User $user)
{
    $team = $user->teams()->create([
        'name' => $user->name . "'s Team",
        'personal_team' => true,
        'user_id' => $user->id, // Ensure user_id is set
    ]);

    $user->current_team_id = $team->id;
    $user->save();
}
```

```
public static function create_regular_user()
{
    return User::create([
        'name' => 'Regular',
        'email' => 'regular@videosapp.com',
        'password' => Hash::make('password'),
    ]);
}

public static function create_video_manager_user()
{
    return User::create([
        'name' => 'Video Manager',
        'email' => 'videosmanager@videosapp.com',
        'password' => Hash::make('password'),
    ]);
}

public static function create_superuser_user()
{
    return User::create([
        'name' => 'Super Admin',
        'email' => 'superadmin@videosapp.com',
        'password' => Hash::make('password'),
    ]);
}
```

# Registrar polítiques d'autorització

- Registrar les polítiques d'autorització i definir les portes d'accés

```
public function boot(): void
{
    $this->registerPolicies();

    Gate::define('manage-videos', function ($user) {
        return $user->isSuperAdmin();
    });

    Gate::define('view-videos', function ($user) {
        return $user->hasPermissionTo('view videos');
    });

    Blade::component('layouts.app', 'layouts.app');
}

protected $policies = [
    Video::class => VideoPolicy::class,
];
```

# Permissos DatabaseSeeder

- Afegim els permisos i els usuaris per defecte

```
public function run()
{
    Permission::create(['name' => 'view videos']);
    Permission::create(['name' => 'create videos']);
    Permission::create(['name' => 'manage videos']);

    $superAdminRole = Role::create(['name' => 'Super Admin']);
    $superAdminRole->givePermissionTo(Permission::all());

    $videoManagerRole = Role::create(['name' => 'Video Manager']);
    $videoManagerRole->givePermissionTo(['view videos', 'create videos']);

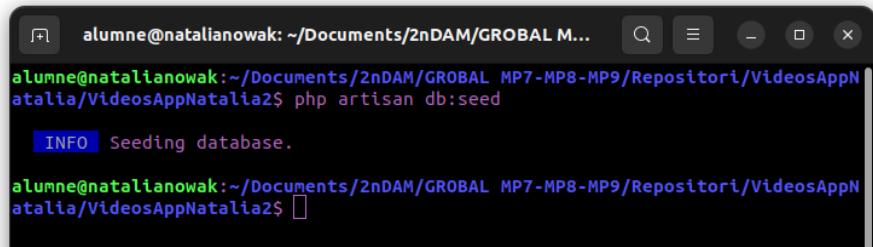
    $regularUserRole = Role::create(['name' => 'Regular User']);
    $regularUserRole->givePermissionTo(['view videos']);

    $superAdmin = User::create([
        'name' => 'Super Admin',
        'email' => 'superadmin@videosapp.com',
        'password' => Hash::make('123456789'),
    ]);
    $superAdmin->assignRole($superAdminRole);

    $videoManager = User::create([
        'name' => 'Video Manager',
        'email' => 'videomanager@videosapp.com',
        'password' => Hash::make('123456789'),
    ]);
    $videoManager->assignRole($videoManagerRole);

    $regularUser = User::create([
        'name' => 'Regular User',
        'email' => 'regular@videosapp.com',
        'password' => Hash::make('123456789'),
    ]);
    $regularUser->assignRole($regularUserRole);
}
```

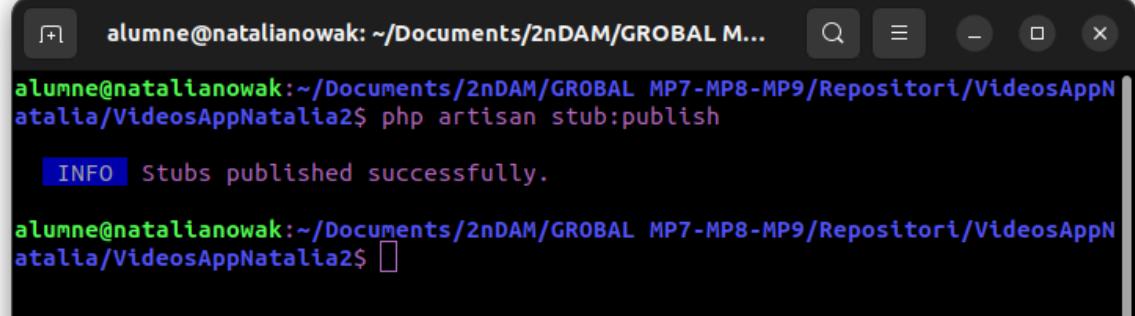
- Actualitzem la base de dades



A terminal window showing the command `php artisan db:seed` being run. The output indicates that the database is being seeded.

```
alumne@natalianowak:~/Documents/2nDAM/GROBAL M...
alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ php artisan db:seed
[INFO] Seeding database.
alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$
```

## Publicar stubs



```
alumne@natalianowak: ~/Documents/2nDAM/GLOBAL M...
alumne@natalianowak:~/Documents/2nDAM/GLOBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ php artisan stub:publish
INFO Stubs published successfully.

alumne@natalianowak:~/Documents/2nDAM/GLOBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia2$ 
```

# Creació de VideoManageController

- Creem el test amb les funcions corresponent

```
● ● ●

public function test_user_with_permissions_can_manage_videos()
{
    $user = User::factory()->create(['is_superuser' => false]);
    $user->givePermissionTo('update videos');
    $this->actingAs($user);

    $video = Video::factory()->create();

    $response = $this->get('/videos/' . $video->id . '/edit');

    $response->assertStatus(200);
    if (!empty(auth()->user()->name)) {
        $response->assertSee(auth()->user()->name);
    }
}

public function test_regular_users_cannot_manage_videos()
{
    $this->loginAsRegularUser();

    $video = Video::factory()->create();

    $response = $this->get('/videos/' . $video->id . '/edit');

    $response->assertStatus(403);
}

public function test_guest_users_cannot_manage_videos()
{
    $this->logout();

    $video = Video::factory()->create();

    $response = $this->get('/videos/' . $video->id . '/edit');
    $response->assertRedirect('/login');
}
```

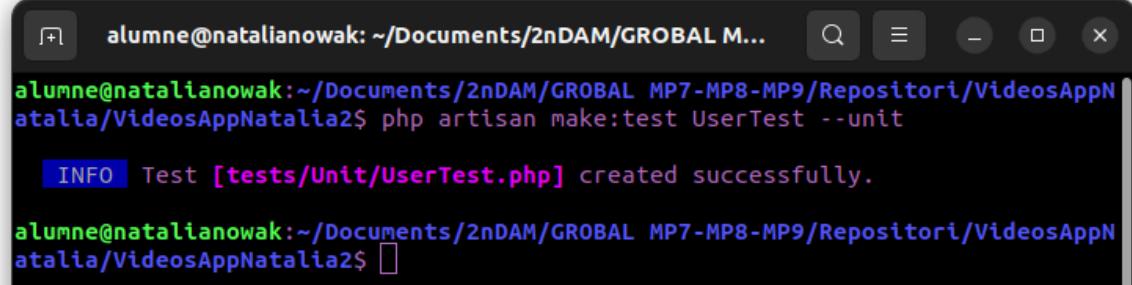
```
protected function loginAsVideoManager(): void
{
    $user = User::factory()->create();
    $user->assignRole('Video Manager');
    $this->actingAs($user);
}

protected function loginAsSuperAdmin(): void
{
    $user = User::factory()->create();
    $user->assignRole('Super Admin');
    $this->actingAs($user);
}

protected function loginAsRegularUser(): void
{
    $user = User::factory()->create();
    $user->assignRole('Regular User');
    $this->actingAs($user);
}
```

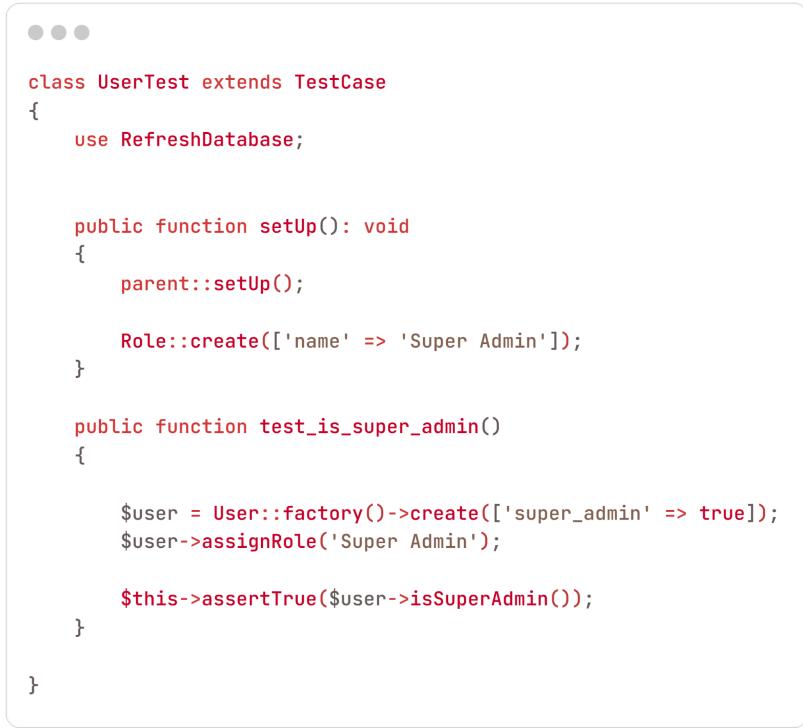
# Creació UserTest

- Creem el test i la funció



```
alumne@natalianowak:~/Documents/2nDAM/GROBAL M... 
alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia$ php artisan make:test UserTest --unit
INFO Test [tests/Unit/UserTest.php] created successfully.

alumne@natalianowak:~/Documents/2nDAM/GROBAL MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia$ 
```



```
class UserTest extends TestCase
{
    use RefreshDatabase;

    public function setUp(): void
    {
        parent::setUp();

        Role::create(['name' => 'Super Admin']);
    }

    public function test_is_super_admin()
    {

        $user = User::factory()->create(['super_admin' => true]);
        $user->assignRole('Super Admin');

        $this->assertTrue($user->isSuperAdmin());
    }
}
```

# Markdown

- Completem el markdown del Sprint 2 i comentem les tasques realitzades en aquest

## Sprint 3

1. Corregir errors del 2n sprint:
  - Vaig començar corregint els errors del segon sprint que havien aparegut durant les proves inicials.
2. Instal·ació del paquet `spatie/laravel-permission`:
  - Vaig instal·lar aquest paquet per gestionar els permisos i rols dels usuaris a l'aplicació. Seguint les instruccions de la [documentació d'instal·lació](#), vaig afegir-lo correctament al projecte.
3. Migració per afegir el camp `super_admin` a la taula `users`:
  - Vaig crear una migració per afegir el camp `super_admin` a la taula d'usuaris, per poder identificar quins usuaris tenen el rol de superadministrador.
4. Actualització del Model `User`:
  - Al model d'usuaris, vaig afegir les funcions `testedBy()` i `issuperAdmin()` per facilitar la verificació de permisos i rols.
5. Afegir el superadmin al professor a la funció `create_default_professor`:
  - Vaig actualitzar aquesta funció a `helpers` per afegir el superadmin al professor per defecte i separar la creació de l'equip del codi de creació dels usuaris mitjançant la nova funció `add_persona_team()`.
6. Creació de funcions per a usuaris regulars i de gestió de vídeos:
  - Vaig crear diverses funcions per crear usuaris regulars, gestors de vídeos i superadministradors. Això inclou funcions com `create_regular_user()`, `create_video_manager_user()`, i `create_superuser_user()` amb valors predeterminats per cada tipus d'usuari.
7. Definir polítiques i portes d'accés a `AppServiceProvider`:
  - A la funció `boot` de `AppServiceProvider`, vaig registrar les polítiques d'autorització i definir les portes d'accés per controlar qui pot veure o modificar què a l'aplicació.
8. Posar permisos i usuaris per defecte al `DatabaseSeeder`:
  - Vaig afegir els permisos i usuaris (superadmin, regular user i video manager) al `DatabaseSeeder`, per garantir que aquests usuaris estiguin disponibles a la base de dades inicialment.
9. Publicar els stubs per personalitzar-los:
  - Vaig seguir la guia per [personalitzar els stubs](#) per adaptar el codi generat per Laravel a les necessitats del projecte.
10. Crear el test 'VideoManagerTest' per provar les funcions del gestor de vídeos:
  - Vaig crear un nou test per provar les funcions del gestor de vídeos, com ara la creació, edició i eliminació de vídeos. Aquest test va ser útil per verificar que les funcions del gestor de vídeos funcionaven correctament.
11. Crear el test `UserTest`:
  - A la carpeta `tests/unit`, vaig crear el test `UserTest` i vaig afegir la funció `issuperAdmin()` per verificar que es detectés correctament si un usuari és superadministrador.
12. Afegir un registre a `resources/markdown/terms`:
  - Vaig actualitzar el fitxer de termes per incloure tot el que s'ha fet fins al moment en aquest sprint, per mantenir la documentació al dia.
13. Comprovar els fitxers amb Larastan:
  - Vaig utilitzar [Larastan](#) per analitzar tot el codi creat i detectar possibles errors de tipus i altres problemes en el codi.

# Testos php artisan

- Comprovem que els testos passen sense cap error

```
! team member roles can be updated → This test did not perform any assertions
✓ only team owner can update team member roles

PASS Tests\Feature\UpdateTeamNameTest
✓ team names can be updated

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

PASS Tests\Feature\VideosTest
✓ users can view videos
✓ users cannot view non-existing videos

Tests: 2 risky, 7 skipped, 52 passed (95 assertions)
Duration: 1.37s

alumne@natalianowak:~/Documents/2nDAM/GLOBAL_MP7-MP8-MP9/Repositori/VideosAppNatalia/VideosAppNatalia3$
```

- Principals correccions realitzades:

- Actualitzar les rutes:

```
Route::get('/', function () {
    return view('welcome');
});

Route::get('videos/{video}', [VideosController::class, 'show'])->name('videos.show');
Route::put('teams/{team}/members/{user}', [TeamController::class, 'updateRole']);
Route::put('/videos/{video}', [VideosController::class, 'update'])->name('videos.update');

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

- Adapta el model i el controller:

```
protected $casts = [
    'email_verified_at' => 'datetime',
    'super_admin' => 'boolean',
];

public function getAuthPassword()
{
    return $this->password;
}

public function teams(): \Illuminate\Database\Eloquent\Relations\HasMany
{
    return $this->hasMany(Team::class);
}
```

- Crea el TeamController

```
● ● ●

public function updateRole(Request $request, Team $team, User $user):
\Illuminate\Http\Response
{
    $this->authorize('update', $team);

    $team->users()->updateExistingPivot($user->id, ['role' => $request->role]);

    return response()->noContent();
}
}
```

# Test Larastan

Ni han molts errors que són predeterminats de Jetstream, els que he pogut arreglar són els següents:

- Adaptar el Model d'Users:

```
...
public function isSuperAdmin(): bool
{
    return $this->is_superuser;
}

public function teams(): \Illuminate\Database\Eloquent\Relations\BelongsToMany
{
    return $this->belongsToMany(Team::class);
}

public function hasTeamRole($team, $role): bool
{
    return $this->teams()
        ->where('team_id', $team->id)
        ->wherePivot('role', $role)
        ->exists();
}
```

- Afegir PHPDoc a certs documents, com per exemple:

```
...
public function updateRole(Request $request, Team $team, User $user):
\Illuminate\Http\Response
{
    $this->authorize('update', $team);

    $team->users()->updateExistingPivot($user->id, ['role' => $request->role]);

    return response()->noContent();
}
```

```
class VideosController extends Controller
{
    use AuthorizesRequests;

    /**
     * Muestra un video específico.
     *
     * @param int $id
     * @return \Illuminate\Contracts\View\View
     */
    public function show(int $id): View
    {
        $video = Video::query()->findOrFail($id);

        return view('videos.show', compact('video'));
    }
}
```

```
use RefreshDatabase;

/**
 * Test para verificar que los usuarios pueden ver videos existentes.
 *
 * @return void
 */
public function test_users_can_view_videos()
{
    $video = Video::factory()->create();

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

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

- Afegir el TeamController:

```
● ● ●

class TeamController extends Controller
{
    use AuthorizesRequests;

    /**
     * @throws AuthorizationException
     */
    public function updateRole(Request $request, Team $team, User $user):
\Illuminate\Http\Response
    {
        $this->authorize('update', $team);

        $team->users()->updateExistingPivot($user->id, ['role' => $request->role]);

        return response()->noContent();
    }
}
```