DOCUMENTACIÓN - DWSStreaming

Autor: [Luis José Soriano Moreno]

Resumen del Proyecto

El objetivo del proyecto es desarrollar una aplicación web mediante el uso el framework en su versión más reciente, Laravel 11. La plataforma permitirá a los usuarios visualizar un completo cartálogo de películas con sus correspondientes trailer. Además, en la parte back habrá un sistema de gestión de la aplicación, que requiere de autenticación, para que los administradores puedan gestionar el catálogo y manejar las bases de datos.

Objetivos

- Crear un catálogo de películas, series y documentales.
- Permitir la autenticación segura con Laravel Sanctum y la gestión de vistas por autor.
- Permitir que los administradores gestionen el catálogo.
- Aplicar el patrón MVC para una organización eficiente del proyecto.
- Utilizar bases de datos relacionales (MySQL) para almacenamiento persistente.

Funcionalidades Clave

- Registro e inicio de sesión de usuarios.
- Visualización del catálogo de películas y series.
- Sistema de roles (usuario y administrador).
- Gestión de contenido para administradores.
- Implementación de Laravel Sanctum para autenticación segura.
- Estilizado con CSS para mejorar la experiencia del usuario.

Requisitos

Dependencias principales:

- **PHP**: Versión 8.3
- **Laravel**: Versión 10.X o superior
- **Base de datos**: MySQL
- **Otras dependencias**: XAMPP, COMPOSER, Laravel Sanctum

Instalación

1. Clonar el repositorio y moverse a la carpeta

   git clone https://github.com/LuisoJSM/DWWS_OK.git
   cd /proyecto

2. Instalar dependencias:

   composer install

3. Configurar el archivo .env:
   - Duplicar .env.example y renombrarlo como .env.
   - Configurar la conexión con la base de datos.

4. Generar clave

   php artisan key:generate

5. Ejecutar migraciones y seeders:

   php artisan migrate –seed

6. Iniciar XAMPP. Verificar que los puertos de conexión corresponden con el archivo .env

7. Iniciar el servidor de desarrollo:

   php artisan serve

8. Abrir PhpMyAdmin para verificar la base de datos.

Rutas Principales

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

//RUTAS AUTENTICAR

// Mostrar el formulario de login
Route::get('/login', [LoginController::class, 'showLoginForm'])->name('login');

// Procesar el login
Route::post('/login', [LoginController::class, 'login']);

//RUTAS LOGOUT
// Route::post('/logout', [LoginController::class, 'logout'])->middleware('auth')->name('logout');
Route::get('/logout', [LoginController::class, 'logout'])->middleware('auth')->name('logout');
```

```
Route::post('/logout', [LoginController::class, 'logout'])->middleware('auth:sanctum')-
>name('logout');


//RUTA REGISTRO

// Mostrar formulario de registro
Route::get('/register', [RegisterController::class, 'showRegistrationForm'])->name('register');

// Procesar el formulario de registro
Route::post('/register', [RegisterController::class, 'register']);


//Ruta Front End

Route::get('/catalogo', action: [FrontEndController::class, "catalogo"])->name('catalogo');
Route::get('/home', action: [FrontEndController::class, "home"])->name('home');

//Ruta de administrador
Route::get('/admin', [AdminController::class, 'admin'])
->middleware('auth:sanctum')
->name('admin');


//RUTAS PROTEGIDAS POR AUTH
// Rutas administración de PELÍCULAS
Route::middleware('auth:sanctum')->group(function () {
 Route::get("formulario-peliculas", [PeliculasController::class, "formulario"])->name("formulario-
pelicula.formulario");
Route::get('lista-pelicula', [PeliculasController::class, 'listaPelicula'])->name('lista-pelicula.lista');
Route::post("formulario-peliculas", [PeliculasController::class, "agregarPelicula"])-
>name("formulario-pelicula.agregar");

// Rutas de DIRECTORES
Route::get("lista-director", [DirectorController::class, 'listaDirector'])->name('lista-director.lista');
Route::get('lista-director-pelicula/{id}', [DirectorController::class, 'listaDirectorPelicula'])-
>name('lista-director-pelicula.lista');

 // Rutas de ELENCO
Route::get('lista-elenco', [ElencoController::class, "listaElenco"])→name("lista-elenco.lista");

Route::get("formulario-elenco", [ElencoController::class, "formulario"])→name("formulario-
elenco.formulario");

Route::post("formulario-elenco", [ElencoController::class, "agregarElenco"])→name("formulario-
elenco.agregar");

 Route::get('elenco/{id}/peliculas', [ElencoController::class, 'listaElencoPeliculas'])-
>name('elenco.lista-elenco-peliculas');
});
```

Estructura del Proyecto

- app/: Código principal (Modelos, Controladores, Middleware).
- routes/: Definición de rutas.
- /lang: carpeta de traducciones
- database/: Migraciones y seeders.
- resources/: Vistas y archivos estáticos.
- .env: Configuraciones de entorno.

Endpoints API

Documentación de rutas

Método	Endpoint	Descripción
GET	/api/catalogo	Obtiene el listado de películas.
GET	/api/pelicula/{id}	Obtiene detalles de una película.
POST	/api/login	Inicia sesión con Sanctum.
POST	/api/register	Registra un nuevo usuario.

Ejemplo de petición POST a /api/login

```
{
  "email": "usuario@example.com",
  "password": "contraseña"
}
```

Ejemplo de respuesta

```
{
  "success": true,
  "token": "Bearer xyz123"
}
```

Pruebas

Para ejecutar las pruebas:

php artisan test

Cobertura esperada:

- Autenticación y autorización.

- Gestión del catálogo.
- Seguridad en el acceso a datos.

Mantenimiento

Tareas Comunes

- Limpiar caché:

  php artisan cache:clear

- Actualizar dependencias:

  composer update

Contribución

Flujo de trabajo recomendado

1. Crear una rama:

   git checkout -b feature/nueva-funcionalidad

2. Realizar cambios y confirmarlos:

   git commit -m "Descripción del cambio"

3. Subir la rama:

   git push origin feature/nueva-funcionalidad

4. Crear un pull request.

Reglas de contribución

- Escribir código limpio y documentado.
- Seguir las guías de estilo de Laravel.
