SUBIDA DE FICHEROS (LARAVEL + PHP)

> **Descripción**
>
> En este documento veremos como se usan ficheros en Laravel para poder guardar información e interactuar con ella.

📌 Índice

Conceptos clave

- Laravel usa **Flysystem** y define "**discos**" en `config/filesystems.php` (p. ej. `local`, `public`, `s3`). Para hacer público `storage/app/public`, ejecuta `php artisan storage:link`.

- Dónde guardas cambia cómo accedes:
  - **Privado (storage/app)**: accesible **solo** vía código (controlador que lee y devuelve contenido).
  - **Público (public/** o **storage/app/public**) : accesible por URL.

Flujo recomendado (checklist)

1. **Formulario** con `method="POST"` y `enctype="multipart/form-data"` (+ `@csrf`). Para múltiples: `name="ficheros[]"`.
2. **Validar** tipo/tamaño (`mimes` , `image` , `max`).
3. **Nombrar** de forma **única** (hash/UUID + extensión).
4. Guardar en disco **privado** si es sensible; **público** si es asset.
5. Exponer **contenido** (privado) con ruta protegida (controlador) o **URL** (público).

Snippets esenciales

Formulario (Blade)

```
<form method="POST" action="{{ route('avatar.store') }}"
enctype="multipart/form-data">
  @csrf
  <input type="file" name="avatar" accept="image/*" required>
  <button>Subir</button>
</form>
```

> *Laravel exige `enctype="multipart/form-data"` y CSRF.*

👁️ Controlador — Validación + guardado (público)

```
$request->validate([
  'avatar' => ['required','image','max:2048'], // ~2MB
]);

$path = $request->file('avatar')->store('avatars','public'); //
storage/app/public/avatars/...
// URL pública:
```

```php
$url = asset('storage/'.basename($path)); // o Storage::disk('public')->url($path)
```

> Operaciones y helpers según disco configurado.

🔒 Controlador — Guardado (privado) y servir contenido

```php
// Guardar privado (storage/app/avatars)
$path = $request->file('avatar')->storeAs('avatars', uniqid().'.jpg', 'local');

// Ruta protegida para servirlo:
return response()->file(storage_path('app/'.$path)); // o stream con headers mimeType
```

> *Privado ⇒ no accesible por URL directa; se sirve vía controlador.*

Storage API (útil en servicios, jobs)

```php
use Illuminate\Support\Facades\Storage;

Storage::disk('local')->put('archivo.txt','Contenido');
$contenido = Storage::disk('local')->get('archivo.txt');
Storage::disk('local')->delete('archivo.txt');
```

> Más utilidades: `exists`, `files`, `mimeType`, etc.

Mover a `public/` (no recomendado para sensibles)

```php
$request->file('file')->move(public_path('uploads'), $fileName);
```

> Rápido pero sin capa de control.

PHP puro (mínimo)

```php
// html: <form method="post" enctype="multipart/form-data">
move_uploaded_file($_FILES['fichero']['tmp_name'],
__DIR__.'/uploads/'.$_FILES['fichero']['name']);
```

> Recuerda comprobar tamaño/tipo y sanitizar el nombre.

⚠ php.ini (cuando "no sube")

Ajusta: `file_uploads=On`, `upload_max_filesize`, `post_max_size` (> upload_max_filesize), `max_file_uploads`, `max_input_time`, `memory_limit`, `max_execution_time`.