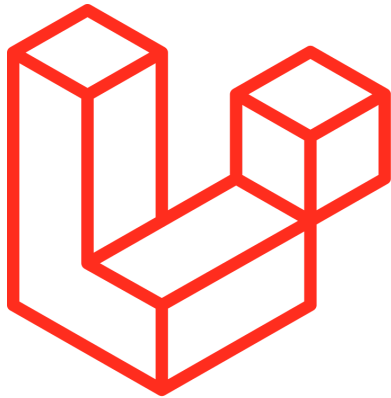


# PHP Y FRAMEWORK LARAVEL



# Laravel

## ☰ Descripción

Vamos a documentar y apuntar cosas útiles de Laravel para crear proyectos y manejarlo correctamente.

## 📌 ÍNDICE

- [PHP en dos minutos](#)
  - [Mini-snippet \(PHP incrustado\)](#)
- [Qué es un framework y por qué Laravel](#)
  - [Arquitectura y piezas que usarás el 99% del tiempo](#)
  - [Rutas → Controlador → Vista \(ejemplo mínimo\)](#)
  - [Formularios y CSRF en Laravel](#)

## PHP en dos minutos

- PHP es un lenguaje de servidor pensado para generar HTML dinámico; el servidor ejecuta el bloque `<?php ... ?>` y **sustituye su salida** en la respuesta web. Ejemplo “Hola mundo” incrustado en HTML.
- Variables: prefijo `$`, tipado dinámico (no hace falta declarar tipo), asignación por valor o referencia ( `&` ). Ojo con variables no inicializadas (avisos `E_NOTICE` ).
- Superglobales clave: `$_GET` , `$_POST` , `$_FILES` , `$_COOKIE` , `$_SESSION` , `$_SERVER` , `$_REQUEST` .

- Estructuras de control y funciones: sintaxis similar a C/Java; funciones con parámetros, valores por defecto, `return`, y (desde PHP 7) tipos en parámetros y retorno opcionales.
- Formularios: usar `method="get|post"`, recoger datos con `$_GET / $_POST`, y **validar siempre** (ideal en cliente + servidor).

## Mini-snippet (PHP incrustado)

```
<!doctype html>
<html>
<body>

    <?php echo "Hola mundo"; ?>

</body>
</html>
```

*Idea clave:* el servidor "inyecta" la salida en el HTML final.

---

## Qué es un framework y por qué Laravel

- Un **framework** aporta arquitectura, librerías y metodología (patrones, buenas prácticas) para evitar código repetitivo y acelerar desarrollo. Normalmente se usa **MVC**: Modelo (datos/ORM), Vista (HTML/Blade), Controlador (lógica/validación).
- **Laravel** (2011→): framework PHP de código abierto, sintaxis elegante, apoya-se mucho en componentes de Symfony. Requiere extensiones habituales de PHP (según versión; consulta doc de tu versión).

## Arquitectura y piezas que usarás el 99% del tiempo

- **Rutas** ( `routes/web.php` ): definen cómo entra el tráfico: `Route::get('/', '...')`, `Route::post(...)`, etc. También `Route::resource(...)` para CRUD REST.
- **Controladores** ( `app/Http/Controllers` ): agrupan lógica. Genera con Artisan: `php artisan make:controller UsuarioController -r`. Métodos típicos: `index`, `create`, `store`, `show`, `edit`, `update`, `destroy`.
- **Vistas** ( `resources/views` ): Blade `*.blade.php` con `@foreach`, `@if`, `{{ }}` ...
- **Modelos/Eloquent** ( `app/Models` ): ORM para consultar/insertar/actualizar: `Usuario::all()`, `Usuario::find(1)`, `Usuario::where(...)->get()`, `$usuario->save()`.

- **Storage** ( `storage/*` + `config/filesystems.php` ) y **public** ( `public/` ): gestión de archivos y punto de entrada `public/index.php` .
- **Artisan**: CLI de Laravel ( `php artisan list` , `php artisan help migrate` ) para scaffolding y tareas.
- **.env**: credenciales/config locales (p. ej., DB).

## Rutas → Controlador → Vista (ejemplo mínimo)

### routes/web.php

```
use App\Http\Controllers\UsuarioController;

Route::get('/', fn() => 'Bienvenido a la página principal'); // demo
Route::resource('usuarios', UsuarioController::class);        // CRUD
```

### app/Http/Controllers/UsuarioController.php

```
public function index() {
    $usuarios = \App\Models\Usuario::all();
    return view('usuarios.index', compact('usuarios'));
}
```

### resources/views/usuarios/index.blade.php

```
<h1>Usuarios</h1>
@forelse($usuarios as $u)
    <p>{{ $u->nombre }}</p>
@empty
    <p>No hay usuarios</p>
@endforelse
```

## Formularios y CSRF en Laravel

En vistas, cuando envíes formularios `POST/PUT/PATCH/DELETE` , incluye el token **CSRF** y el método cuando no sea `POST` :

```
<form method="POST" action="{{ route('usuarios.update', $u) }}">
    @csrf
    @method('PUT')
    <!-- inputs -->
</form>
```

Laravel exige token CSRF y método adecuado para validar la petición.