Laravel

Laravel es un framework PHP que nos ayuda a construir aplicaciones web siguiendo el patrón MVC (Modelo-Vista-Controlador).

Usuario→***Solicitud HTTP***→Rutas
Rutas→***Dirige a***→Controlador
Controlador→***Pide datos***→Modelo
Modelo→***Consulta SQL***→BBDD
BBDD→***Retorna datos***→Modelo
Modelo→***Da objetos***→Controlador
Controlador→***Envía datos***→Vista
Vista→***HTML generado***→**Usuario**

```
proyecto/
├ app/
│ ├ Http/
│ │ └ Controllers/
│ └ Models/
├ database/
│ └ migrations/
├ resources/
│ └ views/
│   └ layouts/
├ routes/
│ └ web.php
└ .env
```

Componentes principales:
- Rutas (**routes**): URL ↔ Acción.
- Controladores (**Controllers**): Lógica app.
- Modelos (**Models**): Tablas de la bbdd.
- Vistas (**views**): Respresentación (HTML).
- Eloquent ORM: BBDD sin escribir SQL.
- Blade: Motor de plantillas para vistas.

Archivo ".env" **¡¡¡OCULTAR!!!**
- DB_CONNECTION=mysql/sqlite/pgsql
- DB_DATABASE=bd
- DB_USERNAME=usuario_bd
- DB_PASSWORD=clave_bd
- APP_URL=https://dominio/proyecto/

Una **MIGRACIÓN** es un archivo PHP que describe cómo crear o modificar una tabla en la base de datos. Es como un plano arquitectónico para construir tu base de datos (DDL):
`php artisan make:migration create_tasks_table` → database/migrations/create_tasks_table.php
Edita y configura la tarea de crear la tabla con los atributos que necesitas y luego:
`php artisan migrate` → ejecuta la migración creando las tablas en la base de datos.

ELOQUENT es el ORM (Object-Relational Mapping) de Laravel. Nos permite trabajar con la base de datos usando objetos PHP en lugar de escribir SQL directamente.
`Task::all()`,`Task::find($id)`,`Task::create($data)`,`$task->update($data)`,`$task->delete()`
*SELECT * FROM tasks WHERE completada = 0;* → `Task::where(campo, valor)->get();`
`php artisan make:model Task` → app/Models/Task.php

Las **RUTAS** (web.php), para un **método HTTP** conectan las **URLs** de la consulta en la clase del **controlador** a un **método**. Pudiendo ponerle un nombre identificador de ruta (Ej: tasks.index).
`Route::get('/', function() {return redirect()->route('tasks.index');});` → Redirige a tasks.index
`Route::get('/tasks', [TaskController::class, 'index'])->name('tasks.index');` → Id = tasks.index
`Route::get('/tasks/create', [TaskController::class, 'create'])->name('tasks.create');`
`Route::post('/tasks', [TaskController::class, 'store'])->name('tasks.store');`
`Route::get('/tasks/{id}/edit', [TaskController::class, 'edit'])->name('tasks.edit');`
`Route::put('/tasks/{id}', [TaskController::class, 'update'])->name('tasks.update');`
`...php artisan route:list` → Comando para listar las rutas definidas

Un **CONTROLADOR** es una clase que gestiona la lógica de nuestra aplicación. Recibe peticiones, procesa datos y devuelve respuestas. Definiendo métodos como `index()` → muestra por defeto, `create()`→muestra para crear, `store()`→guarda, `destroy()`, `update()`...
`php artisan make:controller TaskController` → app/Http/Controllers/TaskController.php
Por ejemplo inserta en la clase del controlador el siguiente método:
`public function index() {`
`$tasks = Task::orderBy('fecha', 'desc')->get();` → Obtiene todas las tareas ordenadas
`return view('tareas.index', compact('tasks')); }` → Retorna la vista con los datos (array/s)

blogsaverroes.juntadeandalucia.es/ismo
Autor: Ismael Ponce Gordillo
Fecha: 3-Feb-2026 Ver.:1.0

BLADE es el motor de plantillas de Laravel. Nos permite escribir HTML con código PHP de forma más limpia y elegante.
- En las **plantillas** (resources/views/layouts/plantilla.blade.php) hay HTML con
  `@yield('titulo', 'Proyecto 1')` → inserta título con valor por defecto
  `@yield('contenido')` → inserta HTML desde la sesión('contenido') de la vista.
- En las **vistas** (/resources/views/tareas/index.php) hay:
  `@extends('layouts.plantilla')` → Hereda de un layout, insertando section() → yield()
  `@section('titulo','Listar')` → Cambia título
  `@section('contenido')` HTML `@endsection` → insertar HTML en yield('contenido') de la plantilla.
  `{{ $variable }}`→Imprime y escapa HTML, `@csrf` → token de seguridad en "form".
  `@if($condicion)..@else..@endif`, `@foreach($elementos as $elemento)..@endforeach`