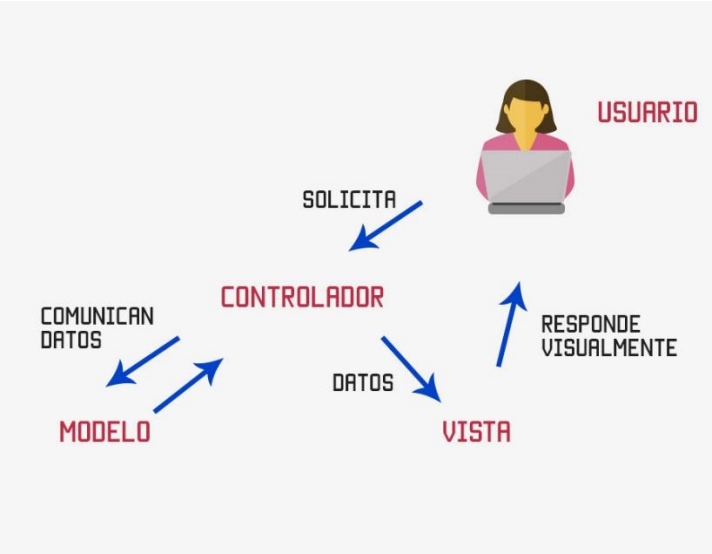
MVC es una arquitectura del software utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta utilizando 3 componentes (Vistas, Modelos y Controladores).

Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.



- **Modelo (Model)**
  Reglas de negocio + acceso a datos. Es la capa donde se trabaja con los datos mapean tablas y relaciones.
  en laravel se encuentran en App\Models.
- **Vista (View)**
  Presentación (HTML), visualización de las interfaces de usuario.
  En Laravel son plantillas **Blade** (p.ej. resources/views/categorias/index.blade.php).
  No hacen queries; muestran datos que les pasa el controlador.
- **Controlador (Controller)**

  Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc, es decir recibe la petición, llama al modelo/servicios, y retorna una **respuesta** (vista, JSON, redirect…). O de una forma más técnica: recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.
  En Laravel se encuentran en app/Http/Controllers/*.

Flujo típico
Navegador → routes/web.php → Controlador → (Modelo/Eloquent/DB) → Vista (Blade) → Respuesta

Ejemplo

```
// routes/web.php

use App\Http\Controllers\CategoriaController;
Route::get('/categorias', [CategoriaController::class, 'index'])->name('categorias.index');
```

```
// app/Http/Controllers/CategoriaController.php

use App\Models\Categoria;

class CategoriaController extends Controller {
  public function index() {
    $categorias = Categoria::all();
    return view('categorias.index',
compact('categorias'));
  }
}
```

```
// app/Models/Categoria.php

class Categoria extends
\Illuminate\Database\Eloquent\Model {
  protected $table = 'categorias';
  protected $primaryKey = 'id_categoria';
  public $timestamps = false;
  protected $fillable = ['nombre','descripcion'];
}
```

```
{{-- resources/views/categorias/index.blade.php --}}
<h1>Categorías</h1>
@foreach ($categorias as $c)
  <div>{{ $c->nombre }} — {{ $c->descripcion }}</div>
@endforeach
```

El usuario accede a la ruta Sistema.gob.ar/categorías , laravel busca esa ruta en su archivo routes/web , ese archivo le indica en qué función del controlador va a encontrar, el controlador interactúa con el modelo para obtener la respuesta y esa respuesta la pasa a la vista.

Estructura principal de Laravel 10

Raíz del proyecto (lo más importante):

- **`app/`**
  Tu código de dominio.
  - `Models/` (Eloquent),
  - `Http/Controllers/` (controladores),
  - `Http/Middleware/` (filtros),
  - `Policies/` (autorización).
- **`routes/`**
  - `web.php` (rutas web que devuelven vistas),
  - `api.php` (rutas API stateless),
  - `console.php` (comandos artisan),
  - `channels.php` (broadcast).
- **`resources/`**
  - `views/` (Blade),
  - `lang/` (traducciones),
  - `css/js` si usás Vite (opcional; podés usar CDNs o `public/`).
- **`public/`**
  Punto de entrada HTTP (`index.php`), assets públicos (imágenes, CSS/JS sueltos).
- **`config/`**
  Configuraciones (app, cache, database, mail…). 100% controlables via `.env`.
- **`database/`**
  Migraciones, seeders, factories (si las usás). *Tip*: podés seguir con `.sql` si preferís.
- **`bootstrap/`**
  Arranque del framework, cache de configuración/rotas.
- **`storage/`**
  Logs, cachés, archivos generados, `storage/app/public` (symlink a `public/storage`).
- **`vendor/`**
  Dependencias Composer.
- **Archivos clave**
  - `.env` (variables de entorno),
  - `composer.json` (dependencias),
  - `artisan` (CLI de Laravel).

Framework

Un framework es un conjunto de herramientas y patrones que proporcionan una estructura y una base de código para el desarrollo de aplicaciones, acelerando el proceso y facilitando tareas complejas, es decir te da piezas listas. Laravel es un framework PHP de código abierto que se basa en el patrón Modelo-Vista-Controlador (MVC) y ofrece herramientas como el motor de plantillas Blade y la interfaz Artisan para crear aplicaciones web de manera eficiente y organizada.

- **Estructura y base de código**:

  Un framework provee un marco de trabajo ya existente con funciones, bibliotecas y componentes integrados que los desarrolladores pueden usar.

- **Acelera el desarrollo**:

  Al no tener que escribir todo el código desde cero, un framework permite agilizar el proceso de creación de aplicaciones web o software.

- **Organización y orden**:

  Los frameworks promueven la organización del código, facilitando su mantenimiento y la colaboración entre desarrolladores.

- **Comodidad y eficiencia**:
  Proporciona soluciones predefinidas para tareas comunes como la autenticación, la gestión de bases de datos, el enrutamiento y la gestión de sesiones.

Qué es Laravel

Laravel es un popular framework PHP. En estos momentos es el framework más utilizado y de mayor progresión en la comunidad PHP y en general también es el más popular de los frameworks backend para desarrollo de proyectos en la web.

- **Patrón Arquitectónico MVC**:

  Sigue la arquitectura Modelo-Vista-Controlador (MVC), que separa la lógica de la aplicación en tres componentes distintos para mejorar la organización y la facilidad de desarrollo.

- **Herramientas destacadas**:
- **Artisan**: Una interfaz de línea de comandos (CLI) que permite ejecutar tareas comunes y automatizar procesos en la aplicación.
- **Eloquent ORM**: Un ORM (Object-Relational Mapper) que facilita la interacción con las bases de datos al permitir escribir consultas sobre objetos en lugar de SQL directamente.
- **Blade**: Un motor de plantillas que permite crear vistas de manera eficiente y reutilizar código.

Mas acerca de Eloquent (TRABAJAREMOS CON ESTO)

Eloquent es una herramienta muy útil para interactuar con bases de datos de una manera más sencilla y elegante en Laravel. Como ORM (mapeo objeto-relacional), Eloquent te permite gestionar tus datos de manera eficiente y fácilmente.

Con Eloquent, puedes utilizar modelos para representar tus tablas de base de datos y manipularlos en PHP. Esto significa que no tienes que escribir consultas SQL para interactuar con la base de datos, sino que puedes utilizar métodos en los modelos para realizar tareas comunes como guardar, actualizar y eliminar registros.

Además, Eloquent también te permite definir relaciones entre tus modelos, lo que hace que sea fácil y eficiente recuperar datos relacionados de múltiples tablas.

En resumen, Eloquent es una herramienta muy poderosa para gestionar tus datos en Laravel, lo que hace que el desarrollo de aplicaciones sea mucho más eficiente y fácil de mantener. Si estás buscando una manera elegante de interactuar con bases de datos en Laravel, Eloquent es definitivamente una excelente opción.