LARAVEL I - Primeros pasos

Titular	MARIA JOSE SALINAS PARRA
Etiquetas	Laravel
Autor	Henrry Gutiérrez, Maho Salinas

> **Hola!** 🦕 Esperamos que esta guía te sirva para empezar a usar Laravel, te prometemos que lo aprenderás fácilmente. No somos profesionales, así que si cometimos algún error, por favor, háznoslo saber. 🐥 Por cierto, si quieres ayudarnos a mejorar la guía lo agradeceríamos muchísimo ✨ - *Maho y Henrry*

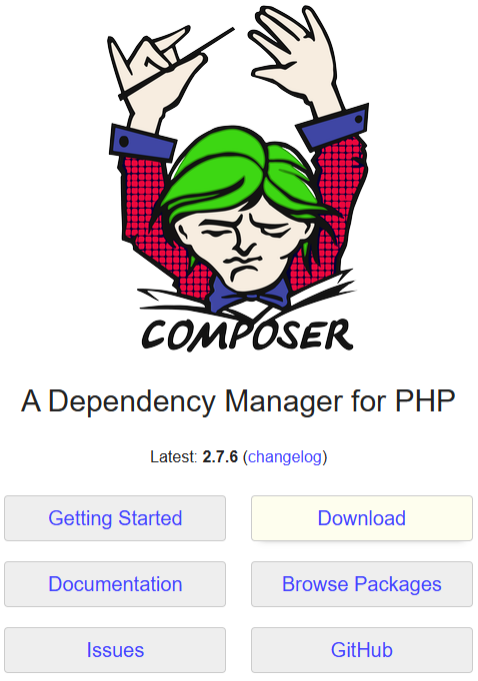
ÍNDICE

Instalación

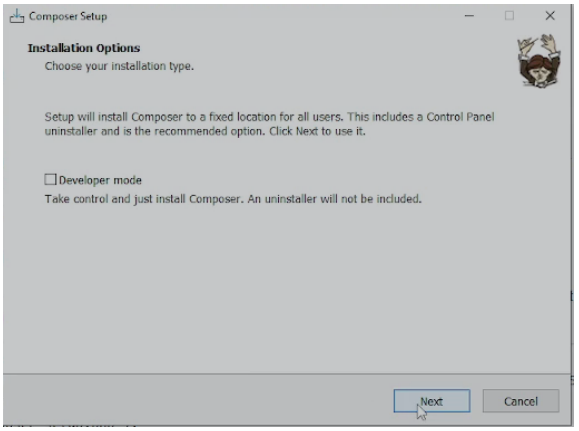
Usamos la documentación oficial de Laravel 🔗 Installation - Laravel 11.x - The PHP Framework For Web Artisans.

Pasos previos

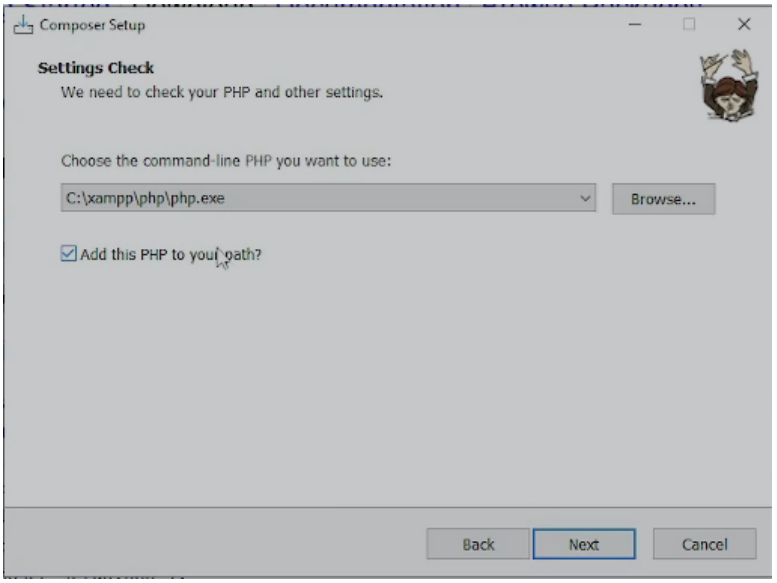
1. Antes de seguir con los pasos, asegúrate de tener instalado Composer. Si no lo tienes, lo puedes hacer desde este aquí🔗 Composer (getcomposer.org. Después hacer click en instalar.



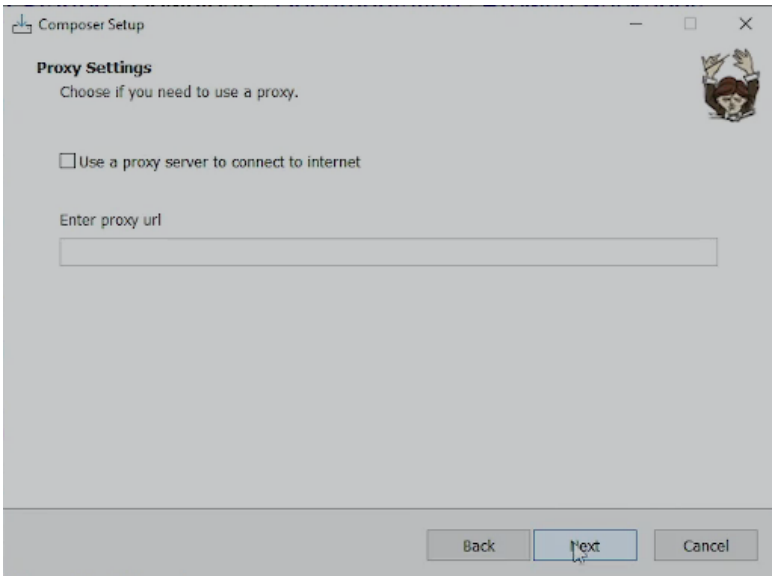
Una vez ahí puedes escoger cómo instalrlo, nosotros usamos el ejecutable Composer-Setup.exe



Marca la opción de añadir a la ruta

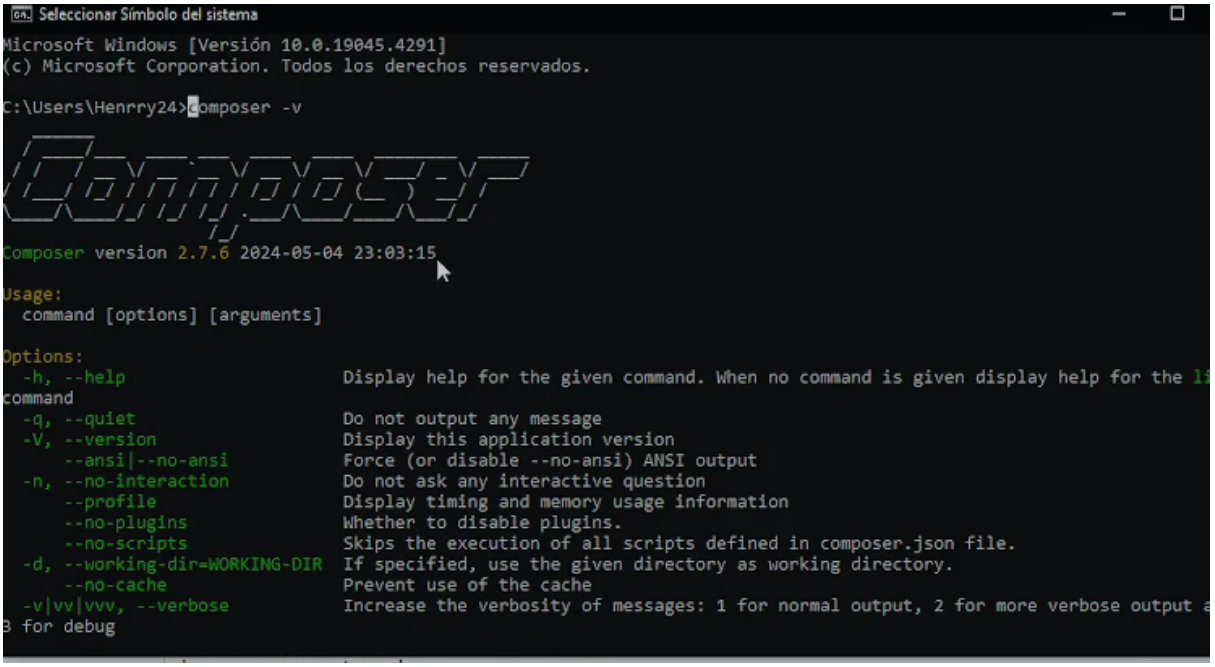


Aquí solo next, install y finish

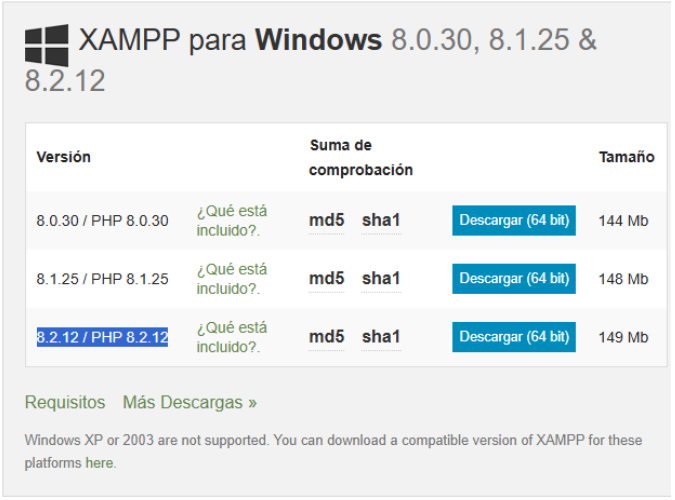


Si quieres verificar que lo tienes instalado, abre tu cmd y escribe composer -v, te saldrá algo así:

```
composer -v
```



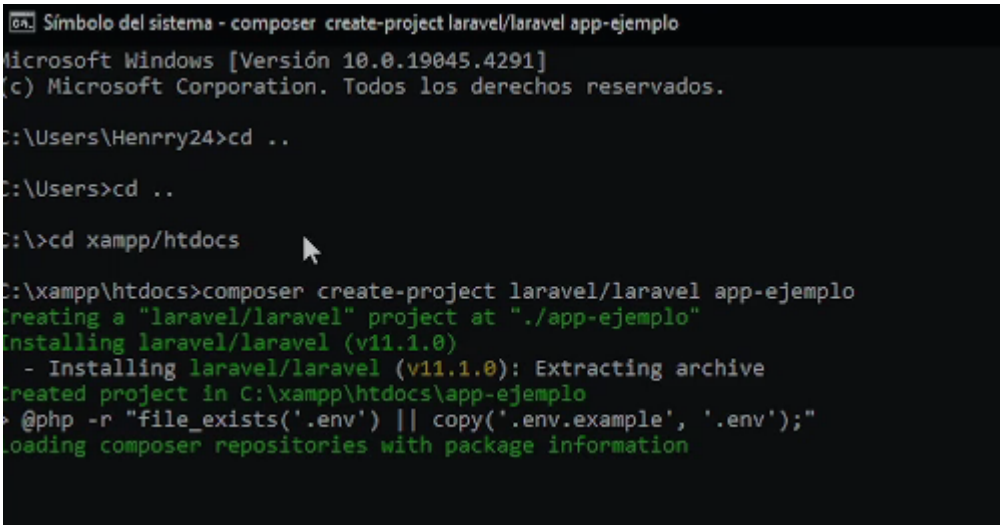
2. También debes tener instalado un entorno de desarrollo. Para practicar y con fines prácticos usamos XAMPP, pero puedes usar otros como Laragon, Docker, etc. Puedes descargar XAMPP aquí 🔗 Download XAMPP (apachefriends.org). Nosotros usamos la **versión 8.2.12 / PHP 8.2.12.** Esto es importante, porque instalaremos la última versión de laravel que usa esa versión de PHP. Así que asegúrate que es esa :)



Creación de proyecto

1. Después de haber instalado PHP y Composer, puedes crear un nuevo proyecto Laravel mediante el comando create-project de Composer. IMPORTANTE debes hacer esto dentro de tu carpeta web, en este caso htdocs.

```
composer create-project laravel/laravel example-app
```



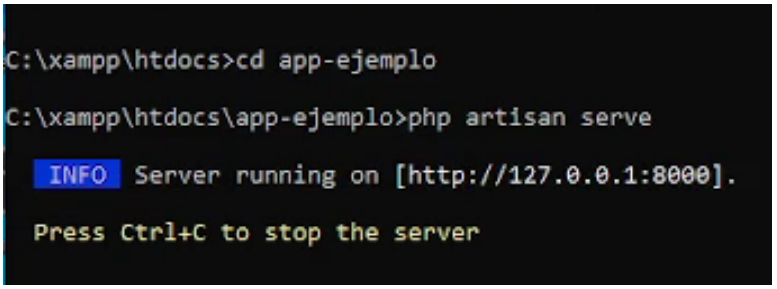
O puedes crear nuevos proyectos de Laravel instalando globalmente el instalador de Laravel a través de Composer.

```
composer global require laravel/installer

laravel new example-app
```

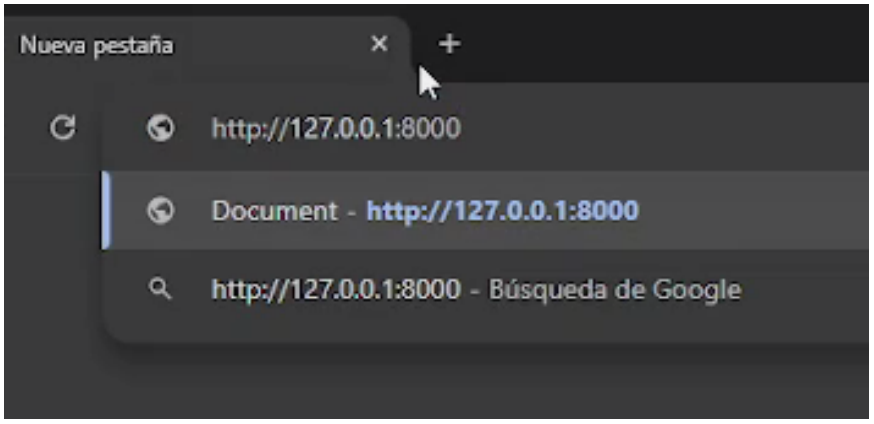
2. Ambas te crearán una carpeta llamada “example-app”. Una vez que se haya creado el proyecto, entra a la carpeta creada e inicia el servidor de desarrollo local de Laravel usando el comando de servicio de Laravel Artisan

Cuando quieras ver tu app web, ya no necesitas colocar la ruta, porque estará alojada en http://localhost:8000 o el link que te dé Laravel

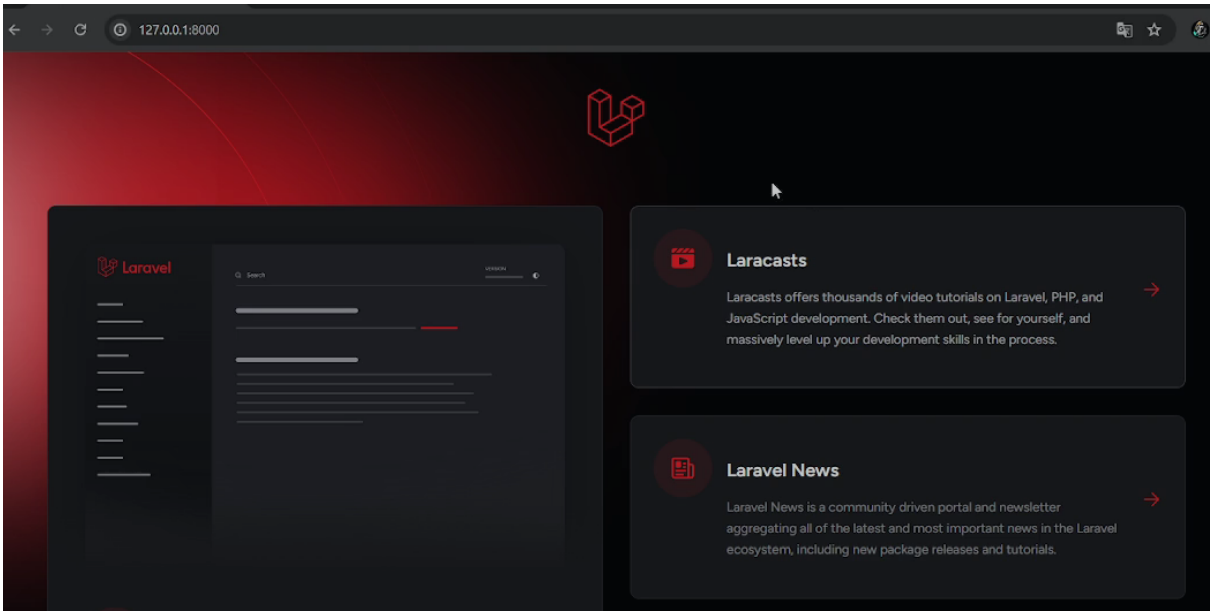
```
cd example-app

php artisan serve
```



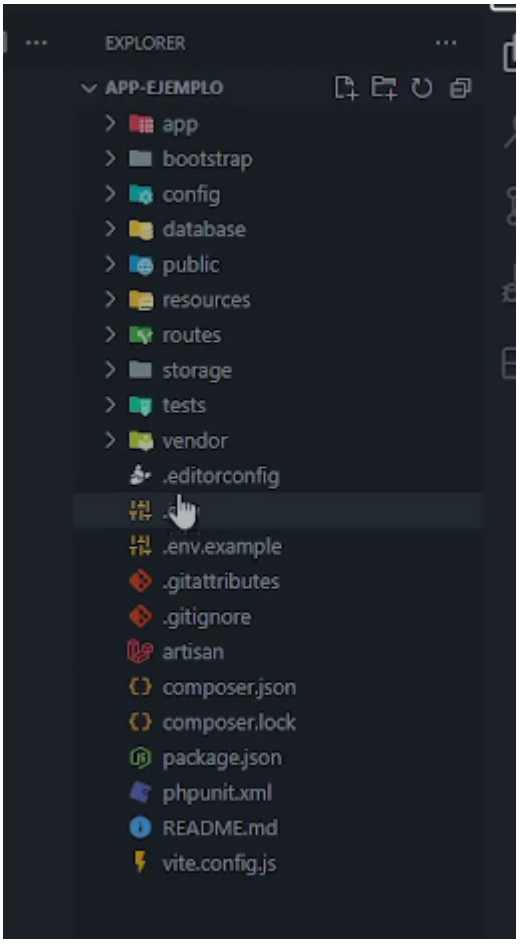
Si entras a esa dirección



Aparecerá esto, lo que significa que lo hiciste bien 👏



Puedes abrir esa carpeta de ejemplo con tu IDE favorito, en este caso Visual Studio Code. Verás las carpetas:



Info que quieres saber 📚

Laravel separa automáticamente en carpetas los diferentes archivos. Así que memoriza dónde se encuentran. Las vistas, son lo que verá el cliente, para eso Laravel tiene Blade. Los los controladores tienen las funciones y las rutas manejan la forma de comunicación entre archivos.

Mapa de rutas 🗺️

Dónde está qué cosa?

1. VISTAS → resources/views/
2. RUTAS → routes/web.php
3. CONTROLADORES → app/Http/Controllers/
4. MODELOS → app/models/
5. ARCHIVO DE CONFIGURACIÓN DE LA BASE DE DATOS → .env

Guía de *artisan* ⚔️

Estos comandos te harán la vida más sencilla

```
//Inicar el servidor de desarrollo d Laravel
php artisan serve

//Crear vista
php artisan make:view nombreDeVista

//Crear controlador
php arstisan make:controller

//Crear una tabla -> migraciones
php artisan make:migration nombre_migración

//Crear modelo
php artisan make:model nombre_modelo

//Ejecutar todas las migraciones
php artisan migrate

//Crear una nueva seed para probar la bd
php artisan make:seed

//Ejectuar todas las seeds pendientes
php artisan db:seed

//Crear un nuevo middleware para procesar una solicitud
php artisan make:middleware

//Mostrar las rutas definidas en Laravel
php artisan route:list
```

Qué es Blade?

Blade es el motor de plantillas de Laravel, diseñado para ser simple pero potente. Blade permite a los desarrolladores usar una sintaxis de plantilla

elegante y concisa para crear vistas en aplicaciones Laravel. - *ChatGPT*

1. Herencia de Plantillas

Blade facilita la creación de vistas que heredan de una plantilla base. Esto se hace a través de las directivas @extends, @sections @yield. **Ejemplo de una plantilla base (`layout.blade.php`):**

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, user
-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum
-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>@yield('title', 'Default Title')</title>
    @vite('resources/css/app.css')
</head>
<body>
    <header>
        <!-- Header content -->
    </header>
    <main>
        @yield('content')
    </main>
    <footer>
        <!-- Footer content -->
    </footer>
</body>
</html>
```

Ejemplo de una vista que extiende la plantilla base (`home.blade.php`):

```
@extends('components.layout')

@section('title', 'Home Page')

@section('content')
```

```
    <h1 class="text-3xl font-bold underline">
        Hello, world!
    </h1>
@endsection
```

2. Inclusión de Subvistas

Puedes incluir otras vistas dentro de una vista utilizando la directiva `@include`.

```
@include('partials.header')
```

Bucles

```
@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach
```

3. Variables y Escapado

```
<p>{{ $name }}</p> <!-- Escapado -->
<p>{!! $description !!}</p> <!-- No escapado -->
```

4. Componentes y Slots

Blade soporta la creación de componentes reutilizables, lo que ayuda a organizar y gestionar vistas complejas.

Componente (`alert.blade.php`)

```
<div class="alert alert-{{ $type }}">
    {{ $slot }}
</div>
```

Uso del componente:

```
@component('components.alert', ['type' => 'danger'])
    <strong>Whoops!</strong> Something went wrong!
@endcomponent
```

> 💡Nota: Para empezar, podemos usar Blade. Sin embargo, existe una mejor opción que suele escoger la mayoría y es usar **Laravel para Backend** y otros frameworks como **React o Angular para el Frontend**. Para eso se crea una API que consumirá la página y se maneja la parte visual con componentes, parecido a POO. Por lo que es necesario aprender bien JavaScript.

Blade permite usar variables PHP directamente en las vistas. Además, Blade escapa automáticamente los datos para evitar ataques XSS (Cross-Site Scripting).