DOXA EDITION

Autor:
GABRIEL HOYOS

2025

PHP Y LARAVEL:

La combinación perfecta para el desarrollo web ágil con Laragon

Primera Edición

ITQ
WWW.ITQ.EDU.EC
INVESTIGACIÓN

PHP y Laravel: La Combinación Perfecta para el Desarrollo Web Ágil con Laragon.

AUTOR: ING. GABRIEL HOYOS

PRIMERA EDICIÓN

AÑO: 2025

TRABAJO EN EDICIÓN:

EQUIPO EDITORIAL

MSc. ÁNGEL ANDRÉS TAYUPANTA GALLO

MSc. KEYERMAN TOAPANTA CISNEROS

ISBN:

DEDICATORIA

Esta obra se encuentra dedicada a mi amada familia, mi amada esposa Katherine mi fortaleza , mi refugio y mi amiga más leal y en quien siempre puedo encontrar un cariño sin igual y fuente de toda inspiración, mis padres Cesar y Lorena quienes siempre ha velado por mí porvenir al guiarme y brindarme el consejo oportuno para no claudicar en los momentos difíciles y a su vez inspirarme para salir adelante, mi Tía Verónica una segunda madre para mí que me ha inculcado siempre el hablar, mediar y llevar al consenso general, mi primo Ricardo el hermano de mi segunda madre un ser excepcional que aprecia el conocimiento y la cultura tanto o más que mi persona, sin duda un profesional y un ser humano que enriquece la vida de todo aquel que lo rodea, mis abuelos Cesar y Yolanda quienes junto a mis padres me han guiado por el camino de la educación y perseverancia, María Elena y Augusto que siempre tienen una palabra de aliento , bendición y cariño para que supere mis propias expectativas todos ellos son quienes siempre están y estarán en mi corazón. Todos ellos han formado parte fundamental en mi desarrollo tanto personal, así como profesional por ello agradezco a Dios por tenerlos en mi vida.

AGRADECIMIENTO

Esta obra agradezco al Instituto Superior Tecnológico Quito por ser parte del proceso dentro del desarrollo de esta obra su apoyo incondicional y confianza en este proyecto fueron fundamentales para lograr el objetivo. Gracias por brindarme el espacio y los recursos para dar vida a esta obra, también un agradecimiento especial y de una leve forma de reconocer y agradecer a cada empresa que ha depositado su confianza en mi persona ya que me permitieron adquirir la experiencia y conocimientos que se ven reflejados en esta obra ya que han contribuido a mi formación profesional a lo largo de los años, de igual forma agradecer a mis compañeros por su camaradería, apoyo y retroalimentación fueron invaluables gracias por compartir este viaje creativo conmigo y por enriquecer esta obra con sus perspectivas únicas, también un reconocimiento especial a Dios por hacer posible que esta obra se construya de forma idónea para seguir formando las mentes de futuro.

SOBRE EL AUTOR

Gabriel Hoyos Ingeniero de sistemas graduado en la mención desarrollo de aplicaciones para la gestión en la Universidad Politécnica Salesiana en el año 2018. Su carrera ha estado marcada por una basta trayectoria en proyectos relevantes en el manejo, supervisión, soporte e implementación de ERP's así como desarrollo y diseño de estos. Su experiencia y habilidades técnicas le han permitido destacarse en el ámbito de la informática, desarrollo y la tecnología gracias a una sólida comprensión de los sistemas de planificación de recursos empresariales (ERP), con experiencia en la implementación, configuración y personalización de varias plataformas ERP para satisfacer las necesidades específicas de las organizaciones. He liderado y gestionado proyectos de implementación de ERP desde la fase de planificación hasta la ejecución y el soporte post-implementación, asegurando el cumplimiento de plazos y presupuesto soy experto en trabajar con stakeholders para identificar y documentar requisitos de negocio, lo que me permite adaptar el ERP para mejorar la eficiencia operativa y apoyar los objetivos estratégicos de la empresa utilizado mi conocimiento en ERP´s para analizar y optimizar procesos de negocio, reduciendo costos y aumentando la productividad mediante la automatización y la mejora continua. Además de proporcionar soporte técnico experto y de capacitación a usuarios finales, ayudándolos a maximizar el uso del sistema ERP y resolver cualquier problema técnico que pueda surgir mediante un profundo conocimiento de las tecnologías y herramientas utilizadas en el desarrollo de interfaces de software, incluyendo lenguajes de programación, bibliotecas, y frameworks de frontend como HTML, CSS, JavaScript, React, Angular, entre otros. Al ser capaz de descomponer conceptos técnicos complejos en términos más simples que permitan facilitar la comprensión de los estudiantes, utilizando ejemplos claros y relevantes desarrollando planes de estudio que aborden tanto los fundamentos como las tendencias actuales en el desarrollo de interfaces, asegurando que los estudiantes adquieran conocimientos actualizados y aplicables. Esto a su vez le ha permitido mantenerse al día con las últimas tendencias y avances en el desarrollo de interfaces de software, incorporando nuevos conocimientos y tecnologías en la enseñanza siendo esta la pauta para comprometerse con el campo del desarrollo y trabajar con otros departamentos y disciplinas que me han permitido integrar el diseño, desarrollo e implementación de interfaces en un contexto más amplio, mostrando cómo se relaciona con otros aspectos dentro del desarrollo de software y la tecnología.

CONTENIDO

CONTENIDO 5

ÍNDICE DE FIGURAS 9

ÍNDICE DE TABLAS 11

Introducción 12

1 Capítulo 1 Introducción a PHP una mirada a su historia. 13

1.1 Creación del Lenguaje de Programación. 13

1.1.1 ¿Lenguaje versátil a través de la historia? 13

1.1.2 Historia de PHP 13

1.1.3 PHP 3 y la expansión (1997-2000) 14

1.1.4 PHP 4 y la consolidación (2000-2004) 14

1.1.5 PHP 5 y la orientación a objetos (2004-2014) 14

1.1.6 PHP 7 y las mejoras de rendimiento (2015-presente) 14

1.1.7 PHP 8 y el futuro (2020-presente) 15

1.1.8 Versión actual de PHP y mantenimiento del código 15

1.1.9 ¿Por qué creó PHP? 15

1.2 ¿PHP y cómo funciona su proceso de generación de código? 16

1.2.1 El significado de PHP a lo largo del tiempo. 16

1.3 Funcionamiento de PHP 16

Solicitud del cliente 16

Procesamiento en el servidor 18

Ejecución del código 18

1. Sintaxis de PHP 19

Generación de contenido dinámico 19

Envío de la respuesta 20

El servidor web envía la respuesta 21

El navegador recibe la respuesta 21

1.4 Características principales de PHP 21
Lenguaje de scripting: 21
Incrustación en HTML: 21
Multiplataforma: 21
Código abierto: 21
1.5 Características principales de un servidor web con PHP 22
Interpretación de código PHP 22
Soporte para bases de datos 22
Gestión de archivos 22
Seguridad: 22
1.6 Ruta de carga de archivos y carpetas principales 22
1.7 Carpetas principales: 23
1.8 Servidores PHP Preconfigurados 24
1.9 Servidores de Hosting PHP MySQL 24
2 Ventajas de Utilizar Servidores Preconfigurados 25
2.1 Facilidad de uso 25
2.2 Tiempo de instalación reducido 25
2.3 Compatibilidad 25
2.4 Soporte técnico 25
3 Frameworks PHP 27
3.1 CodeIgniter 27
3.1.1 Características Clave 28
3.2 Symfony 28
3.2.1 Características Clave 28
4 Elección del Framework Adecuado 29
4.1 Consejos para desarrolladores 29
4.2 Resumen del Capítulo 1 29

5 Capítulo 2 Instalación y Administración de Laravel un framework modernizado de PHP. 31

5.1 Introducción a los Frameworks PHP: La piedra angular de la eficiencia en el desarrollo web. 31

5.1.1 ¿Qué es un Framework PHP? 31

5.1.2 Ventajas de Utilizar Frameworks PHP 31

5.2 Laravel: Un Framework Moderno y Elegante 31

5.3 Características Clave de Laravel 32

5.3.1 Sistema de Rutas de Laravel: ¡El mapa de tu aplicación! 32

5.3.2 ¿Cómo funciona el enrutamiento? 32

5.3.3 Características clave del sistema de rutas de Laravel 33

5.3.4 Beneficios del sistema de rutas de Laravel 33

5.4 Eloquent ORM: ¡Tu puente a la base de datos! 34

5.4.1 ¿Cómo funciona Eloquent? 35

5.4.2 Beneficios de usar Eloquent: 36

5.5 Introducción a Blade: El Motor de Plantillas de Laravel 37

5.5.1 Características Clave de Blade 38

5.5.2 Directivas Más Utilizadas en Blade 38

5.6 Autenticación y Autorización en Laravel: Seguridad simplificada 40

5.7 Última versión de Laravel 40

6 Prerrequisitos para el Entorno de Desarrollo de Laravel. 40

6.1 Guía para Empezar con Laravel 40

6.2 Instalación de Server Web: Laragon 41

6.2.1 ¿Quién lo creó y por qué? 41

6.2.2 Instalación en Windows: 41

6.2.3 ¿Qué trae Laragon? 49

6.2.4 Laragon: Ventajas para el desarrollo de PHP y Laravel 50

6.3 Gestor de dependencias Composer 51

6.3.1 ¿Para qué sirve Composer en Laravel? 51
6.3.2 Instalación de Composer 51
6.4 Uso básico de Composer en Laravel 59
6.5 Beneficios de utilizar Composer 59
7 Artisan: la línea de comandos de Laravel (CLI) 61
7.1 Componentes importantes de Laravel: 62
7.2 Artisan: Aplicado a la Creación de componentes 62
7.3 Beneficios de Artisan 63
7.4 Resumen del Capítulo 2 65
8 Capítulo 3 Programando en Laravel una revisión de su proceso general. 67
8.1 Normas y Estructura Fundamentales en el Desarrollo con Laravel 67
8.1.1 Arquitectura Modelo-Vista-Controlador (MVC): 67
8.1.2 Estructura de Directorios 67
8.1.3 Rutas: 69
8.1.4 Controladores 72
8.1.5 Modelos 75
8.1.6 Vistas: 75
8.1.7 Pasos para crear archivos Blade: 76
8.1.8 Manejo de la sintaxis Blade: 76
8.1.9 Migraciones: 79
8.1.10 Seeders 81
8.1.11 Middleware: 83
8.1.12 Proveedores de servicios: 85
8.1.13 Eventos y listeners: 88
9 Resumen del Capítulo 3 90
BIBLIOGRAFÍA 92

ÍNDICE DE FIGURAS

Figura 1 Rasmus Lerdorf 13
Figura 2 Petición al servidor 17
Figura 3 Interpretación grafica de la resolución de peticiones 17
Figura 4 Representación de comunicación con servidor PHP 18
Figura 5 Ejecución de código PHP 18
Figura 6 Primeros paso con PHP 19
Figura 7 Código PHP Hola mundo 20
Figura 8 Taylor Otwell Creador de Laravel 32
Figura 9 Consulta en lenguaje SQL 35
Figura 10 Consulta en Eloquent ORM 35
Figura 11 Obtención de un registro con Eloquent 36
Figura 12 Proceso de creación de un nuevo registro con Eloquent 37
Figura 13 Proceso de actualización de un registro con Eloquent 37
Figura 14 Proceso de eliminación de un registro con Eloquent 37
Figura 15 Proceso de presentación de una variable en Blade 39
Figura 16 Proceso de presentación de una variable en Blade con condicional 39
Figura 17 Proceso de repetición de una variable tipo vector en Blade 39
Figura 18 Proceso de inclusión de una pantalla 39
Figura 19 Laragon opción de descargas 41
Figura 20 Instalación de ejecutable de Laragon. 42
Figura 21 Instalación Laragon parte 1: 42
Figura 22 Instalación de Laragon parte 2 43
Figura 23 Instalación de Laragon parte 3 43
Figura 24 Proceso de Instalación de paquetes de Laragon parte 4. 44
Figura 25 Proceso de instalación Laragon finalizado parte 5 45
Figura 26 Ubicación física de Laragon y Php. 45
Figura 27 Consola de Laragon 46
Figura 28 Levantamiento de Servicios de Laragon. 47
Figura 29 Inicio de la Página Principal de Laragon y del Proyecto 47
Figura 30 Laragon en ejecución. 48
Figura 31 Licencia de Laragon 49
Figura 32 Composer descarga de versión 52
Figura 33 Proceso de selección de la versión LTS de Composer. 53
Figura 34 Instalación de Composer desde descargas. 53

Figura 35 Instalación de Composer parte 1 54
Figura 36 Instalación de Composer parte 2 55
Figura 37 Instalación de Composer parte 3.1 55
Figura 38 Instalación de Composer parte 3.2 56
Figura 39 Instalación de Composer parte 4 57
Figura 40 Instalación de Composer parte 5 58
Figura 41 Finalización de instalación de Composer 58
Figura 42 Ruta Básica GET, ... 71
Figura 43: Ruta POST guiada a un controlador. 71
Figura 44 Rutas con Parámetros 72
Figura 45 Controlador en Laravel 73
Figura 46 Controlador en Laravel 2 74
Figura 47 Paso de parámetros con el Método View. 76
Figura 48 Elementos de vista Blade.php 77
Figura 49 Vista con estructura de control (lista_usuarios.blade.php) 77
Figura 50 Uso de plantillas (layout.blade.php) 78
Figura 51 Representación de uso de Extends y revisión de Sección en Laravel. 78
Figura 52 Uso de comando para la creación de la migración de Laravel. 80
Figura 53 Creación de migración con código PHP 80
Figura 54 Ejecución de la migración de Laravel 81
Figura 55 Ejecución de comando para la creación de seeders 82
Figura 56 Seeder para poblar datos de la tabla Usuarios 82
Figura 57 Registro Seeder en el archivo Seeders de PHP Laravel 83
Figura 58 Creación de Middleware en Laravel 84
Figura 59 Creación de Middleware para Login. 85
Figura 60 Registro del middleware en app/Http/Kernel.php 85
Figura 61 Creación de Service Provider para Cargar API Externa. 86
Figura 62 Service Provider que ejecuta API Externa. 87
Figura 63 Registro del Service Provider. 87
Figura 64 Comando de Artisan para crear Evento. 89
Figura 66 Generación de Evento (Constructor) parte 1 89
Figura 67 Creación de Función Handler para lanzar Evento 90
Figura 68 Proceso de Registro y Ejecución del Evento 90

ÍNDICE DE TABLAS

Tabla 1 Rutas de carpetas importantes en la instalación 22

Tabla 2 Tecnologías de Laragon 49

Tabla 3 Beneficios y características de Composer en Laravel 59

Tabla 4 Beneficios de Artisan en Laravel 63

Tabla 5 Detalle de carpetas relevantes en Laravel. 68

Introducción

En este documento vamos a revisar los conceptos básicos del lenguaje de programación de PHP y a su vez del framework de mismo lenguaje llamado Laravel permitiéndonos así tener una noción clara de trabajo, la forma y el modo en que se emplean en la actualidad para generar desarrollo web de forma más ágil e intuitiva para los desarrolladores. Esta forma de abordar el framework permitirá comprender la documentación inicial y su enorme capacidad para desarrollar productos web robustos y escalables que permitan tener una estructura imprescindible para el manejo y creación de un sistema, pero sobre todo que sus elementos sean capaces de reutilizarse dentro del entorno desarrollo especificado para el backend. Es decir, vamos a revisar proceso de desarrollo y como este ha evolucionado para mantener una estructura que permita generar aplicaciones robustas a lo largo del tiempo, cabe mencionar que este es un libro ha sido creado con el propósito de comprender el funcionamiento de cada uno de los elementos que se puede generar en Laravel.

En el primer capítulo abordaremos la historia y la creación del lenguaje de programación y como es y sigue siendo uno de los más utilizados en la web además de esto analizaremos la creación del framework de desarrollo y determinaremos como paso de ser un lenguaje espagueti a uno de los frameworks más demandados para la generación de sistemas hoy en la actualidad, en este proceso de aprendizaje revisaremos también parte de sus avances y como fue cambiando a lo largo de los años hasta ser la principal herramienta para la gestión y desarrollo de sistemas robustos en base a sus dependencias de composer que permite una mejor gestión de las dependencias del lenguaje de programación PHP.

En el segundo capítulo revisaremos la infraestructura de la herramienta y sus principales modos de trabajo por lo tanto gestionaremos la creación de un proyecto pequeño en Laravel, primero revisando los requerimientos previos para su instalación y que se necesitan para poder trabajar y así levantar el ambiente de desarrollo en un equipo ya sea una computadora de escritorio o una laptop.

En el tercer capítulo revisaremos las principales comandos y estructuras que se presentan en el framework de desarrollo una vez instalado y listo para su despliegue que permita generar una aplicación backend, también su modo de despliegue ya que esta herramienta depende de un servidor web externo en este caso utilizaremos Laragon, pero sí hay otras herramientas que permiten el manejo de servidor web además de Laragon debemos tener el equipo de trabajo instalado las dependencias mediante un orquestador donde el proyecto se vaya a generar.

1 Capítulo 1

Introducción a PHP una mirada a su historia.

1.1 Creación del Lenguaje de Programación.

PHP nace por la necesidad de los usuarios de la web de tener un código abierto que les permita trabajar de forma procesal, especialmente diseñado para el backend y así generar aplicaciones web con código que no se encuentre embebido dentro de HTML. En el mundo del desarrollo web, PHP se ha consolidado como un pilar fundamental en la creación de sitios y aplicaciones dinámicas. Desde sus inicios como un conjunto de herramientas para páginas personales, ha evolucionado hasta convertirse en un lenguaje robusto y versátil, adoptado por millones de desarrolladores en todo el mundo.

1.1.1 ¿Lenguaje versátil a través de la historia?

PHP, acrónimo recursivo de "PHP: Hypertext Preprocessor", es un lenguaje de scripting del lado del servidor que se incrusta en el código HTML. Esto significa que el código PHP se ejecuta en el servidor antes de que la página web se envíe al navegador del usuario. PHP se utiliza para generar contenido dinámico, interactuar con bases de datos, procesar formularios y realizar una amplia gama de tareas en el desarrollo web.

1.1.2 Historia de PHP

Creación y primeros años (1994-1997)

Figura 1

Rasmus Lerdorf

Nota: *Creador del Lenguaje de Programacion PHP .* ***Fuente:*** *(Pinterest, s.f.)*

Rasmus Lerdorf es el creador original de PHP. Su visión y contribuciones sentaron las bases para el lenguaje que conocemos hoy.

Rasmus Lerdorf, un programador danés-canadiense, creó PHP en 1994 como un conjunto de scripts en lenguaje C para mostrar su currículum vitae en línea y realizar un seguimiento del tráfico de su sitio web (Lerdorf & Tatroe, 2002). Inicialmente, PHP se conocía como "Personal Home Page Tools" (Herramientas para Páginas Personales). En 1995, Lerdorf lanzó PHP/FI (Form Interpreter), que permitía a los desarrolladores crear páginas web dinámicas con mayor facilidad.

1.1.3 PHP 3 y la expansión (1997-2000)

En 1997, dos programadores israelíes, Zeev Suraski y Andi Gutmans, reescribieron el analizador sintáctico de PHP y crearon PHP 3 lo cual introdujo nuevas características, como soporte para más protocolos y bases de datos, y se convirtió en un lenguaje más potente y versátil.

En 1998, se lanzó PHP 3.0, que incluía características como el soporte para sesiones y cookies, lo que permitió a los desarrolladores crear aplicaciones web con un nivel de complejidad superior.

1.1.4 PHP 4 y la consolidación (2000-2004)

PHP 4 también introdujo nuevas características, como el soporte para objetos y la capacidad de manejar grandes cantidades de datos por lo que, en el año 2000, se lanzó PHP 4, lo que permitió mejorar el rendimiento y la estabilidad del lenguaje. Durante esta época, PHP se consolidó como uno de los lenguajes de scripting del lado del servidor más populares.

1.1.5 PHP 5 y la orientación a objetos (2004-2014)

En 2004, se lanzó PHP 5, que introdujo mejoras significativas en el soporte para la programación orientada a objetos (POO), por ende, PHP 5 también mejoró el rendimiento y la seguridad del lenguaje, y agregó nuevas características como el soporte para excepciones y el manejo de bibliotecas estándar de PHP (SPL).

1.1.6 PHP 7 y las mejoras de rendimiento (2015-presente)

En 2015, se lanzó PHP 7, que trajo consigo mejoras significativas en el rendimiento y redujo el consumo de memoria. PHP 7 también agregó nuevas características, como el operador de nave espacial y la declaración de tipos escalares. Las versiones posteriores, como PHP 7.1, 7.2, 7.3 y 7.4, continuaron mejorando el rendimiento y agregando nuevas características y un mejor manejo de protocolos.

1.1.7 PHP 8 y el futuro (2020-presente)

En 2020, se lanzó PHP 8, que introdujo nuevas características como el compilador Just-In-Time (JIT), tipos de unión y atributos. PHP 8 continúa mejorando el rendimiento y agregando nuevas características para satisfacer las necesidades de los desarrolladores web modernos.

1.1.8 Versión actual de PHP y mantenimiento del código

La versión más reciente de PHP es la 8.3, lanzada el 23 de noviembre de 2023. Esta versión incluye nuevas características, mejoras de rendimiento y correcciones de errores.

El código de PHP es mantenido por un equipo de desarrolladores voluntarios de todo el mundo, conocido como el "PHP Development Team". Este equipo trabaja de forma colaborativa para mejorar el lenguaje, corregir errores y lanzar nuevas versiones.

1.1.9 ¿Por qué creó PHP?

Lerdorf creó PHP para simplificar el desarrollo de páginas web dinámicas. Su objetivo era proporcionar una herramienta que permitiera a los desarrolladores crear sitios web interactivos de manera más fácil y rápida.

El world wide web ha jugado un papel esencial en la expansión de la denominada corriente open source. Los defensores de esta corriente defienden el desarrollo de aplicaciones informáticas y su distribución libre, de forma gratuita; pero no solo eso sino que también proveen el código fuente de los programas desarrollados. Se trata, en definitiva, de que los usuarios puedan utilizar los programas sin ninguna restricción y puedan conocer si lo desean su funcionamiento interno (Cobo, Gómez, Pérez, & Rocha, 2005).

El ejemplo más emblemático de esta corriente open source lo constituye el sistema operativo Linux. En los últimos años está corriente se ha desarrollado enormemente y ya se pueden obtener en la propia red Internet todo tipo de programas basados en esta filosofía: servidores y navegadores web, entornos de programación, editores, herramientas ofimáticas así como grandes empresas como IBM o Yahoo, (Cobo, Gómez, Pérez, & Rocha, 2005) por ejemplo, han apostado muy fuerte por las soluciones open source y, sin lugar a dudas, el impacto de este tipo de soluciones parece destinado a incrementarse en el futuro debido a sus altas prestaciones y calidad, su menor coste y su alta difusión.

1.2 ¿PHP y cómo funciona su proceso de generación de código?

Está claro que la breve definición y explicación del funcionamiento de las características más destacadas del lenguaje de programación PHP se basan desde el principio de fundación de la world wide web ya que empezaron casi a la par para generar las primeras páginas que navegamos en el internet, la navegación ágil y fluida no era parte de este desarrollo ya que esto fue avanzando a medida que fueron cambiando los frameworks de trabajo y diseño de la web para que PHP pueda también intervenir en el mundo moderno del desarrollo tuvo que cambiar y adaptarse a los nuevas formas de desarrollar. A continuación, se detalla el contenido inicial de PHP como un lenguaje de scripting del lado del servidor que ha ganado popularidad en el desarrollo web desde su creación en 1994 por Rasmus Lerdorf. (Cobo, Gómez, Pérez, & Rocha, 2005) Su capacidad para integrarse fácilmente con HTML y su naturaleza de código abierto lo han convertido en una herramienta esencial para la creación de aplicaciones web dinámicas.

1.2.1 El significado de PHP a lo largo del tiempo.

PHP, que originalmente significaba "Personal Home Page Tools", ahora es un acrónimo recursivo que se traduce como "PHP: Hypertext Preprocessor". Este lenguaje permite la creación de contenido dinámico y la interacción con bases de datos, lo que lo hace ideal para el desarrollo de sitios web interactivos y aplicaciones web complejas (Lerdorf & Tatroe, 2002).

1.3 Funcionamiento de PHP

El proceso de ejecución de un script PHP implica varios pasos:

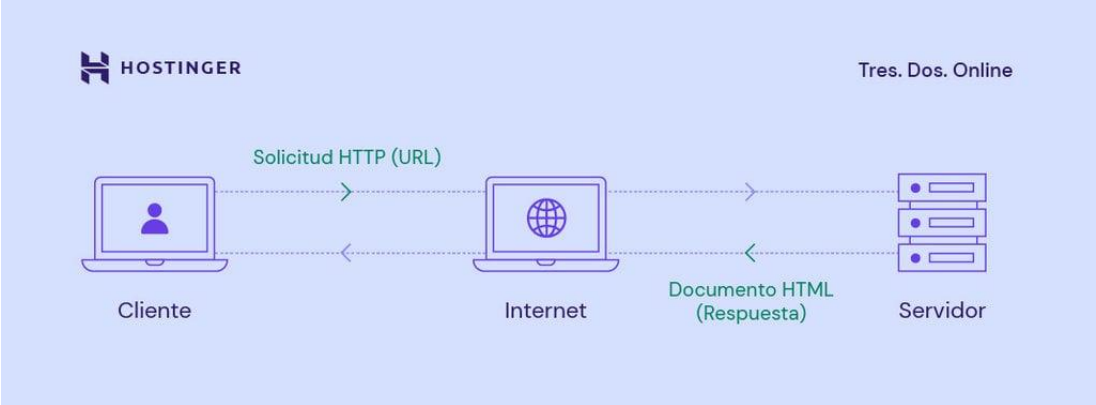
Solicitud del cliente: Cuando un usuario accede a una página web que contiene código PHP, su navegador envía una solicitud al servidor web es decir En el caso de PHP, el código se procesa en el servidor web antes de que el resultado se envíe al navegador. El intérprete lee el archivo PHP, lo analiza y lo ejecuta. Luego, el intérprete PHP devuelve un resultado que el servidor web envía como respuesta al navegador. El modelo cliente-servidor es fundamental en la arquitectura de Internet, donde un programa, conocido como cliente, solicita recursos o servicios a otro programa, el servidor. Cuando un usuario desea acceder a una página web, su navegador envía una petición HTTP a través de Internet.

Este proceso comienza con la búsqueda de la dirección IP del sitio mediante el Sistema de Nombres de Dominio (DNS). Una vez que el servidor web recibe la solicitud, procesa la petición a través de su servidor HTTP y busca los archivos necesarios. Si encuentra los datos solicitados, los envía de vuelta al navegador, permitiendo al usuario visualizar el contenido. En caso contrario, el servidor puede devolver un mensaje de error, como el común error 404 o 403 si hay problemas de permisos se podrían incluir ejemplos concretos de cómo se aplica este

modelo en diferentes servicios web y destacar la importancia de la seguridad en las comunicaciones entre cliente y servidor.

Figura 2

Petición al servidor



***Nota**: peticiones cliente servidor explicada por Hostinger. **Fuente**: (hostinger, s.f.)*

En la figura anterior observamos como PHP trabajara a nivel del cliente servidor mediante respuestas y como debemos interpretarlas de esta forma es el inicio de la comunicación que se establece para procesar las peticiones en el servidor.

Figura 3

Interpretación grafica de la resolución de peticiones

***Nota**: La forma de procesar solicitudes en PHP que por lo general siempre se la realiza en el intérprete y que arroja un resultado. **Fuente**: Autor*

En la siguiente figura podemos comprender que PHP trabajara a nivel del cliente servidor mediante respuestas y como se resuelve de forma interna por el intérprete del lenguaje de programación.

Procesamiento en el servidor: El servidor recibe la solicitud y busca el archivo PHP correspondiente es decir que el servidor PHP es un software que se instala en un servidor web para interpretar el código PHP de una aplicación como lo son XAMPP, Laragon entre otros.

Figura 4

Representación de comunicación con servidor PHP

***Nota**: Revisión de la comunicación de peticiones para PHP. **Fuente**: Autor.*

Ejecución del código: El servidor ejecuta el código PHP incrustado en el archivo a partir de este proceso es donde entran en juego el servidor PHP (Sánchez Cano, 2018) .

Figura 5

Ejecución de código PHP

***Nota**: proceso de peticiones get , post y respuestas desde una base de datos. **Fuente**: (Murcia, s.f.)*

Este software especializado es como un "traductor" que entiende el lenguaje PHP donde el servidor web le entrega el archivo PHP para que lo interprete es decir el servidor ejecuta el código PHP que está dentro del archivo. Este código puede generar contenido dinámico, como texto, imágenes o información de una base de datos por ende es como si el traductor leyera las instrucciones del expediente y las pusiera en práctica para generar el proceso de las instrucciones indicadas.

1. Sintaxis de PHP

Para empezar a programar en este lenguaje, se deben escribir las etiquetas <?php y ?>, y que remarcan el inicio de las sentencias de PHP los símbolos menor que y signo de pregunta indican dónde empezar y los signos pregunta mayor que nos indicaran donde finalizar la interpretación de los comandos de este lenguaje. Es un paso intermedio en el que se traduce y se ejecuta código de programación. del código. Este mecanismo permite que PHP pueda ser incrustado en diferentes tipos de documentos.

Figura 6

Primeros paso con PHP

```
1  <?php
2      echo "¡Hola, esto es código PHP!";
3  ?>
```

***Nota**: Forma procedimental de iniciar un programa de PHP. **Fuente**: Autor*

La instrucción echo permite escribir en pantalla un texto o el contenido de una variable. Es un espacio de memoria en el computador destinado para almacenar un valor. Si se trata de un texto, se debe encerrar entre comillas lo que queremos que se muestre, pero si es el contenido de una variable no es necesario hacerlo, simplemente escribimos echo seguido del nombre de la variable.

Generación de contenido dinámico: El código PHP puede generar contenido que incluye texto, imágenes y datos provenientes de bases de datos. Imaginemos que vamos a crear una página web que muestre la fecha y hora actual, o que salude al usuario por su nombre. Esto no lo puedes hacer con HTML estático, necesitas dinamismo PHP quien será el aliado perfecto a la hora de generar estos procesos que requieren un desempeño mayor.

Figura 7

Código PHP Hola mundo

```
PHP
```

```php
<!DOCTYPE html>
<html>
<head>
  <title>¡Hola!</title>
</head>
<body>
  <h1>¡Hola, <?php echo "mundo dinámico"; ?>!</h1>
</body>
</html>
```

***Nota**: Código para la generación de impresión en pantalla del mensaje Hola Mundo. **Fuente**: Autor*

¿Dónde las líneas de PHP inician con un <?php ...?>, es decir que estas etiquetas delimitan el código PHP que se ejecutará en el servidor. Además, la palabra reservada echo "mundo dinámico”; Esta línea imprime el texto "mundo dinámico" en la página web. Al acceder a hola.php desde tu navegador, se presentará en pantalla el mensaje "¡Hola, mundo dinámico!".

Envío de la respuesta: El servidor envía el contenido generado (en formato HTML, CSS, JavaScript) al navegador del usuario. Después de que el servidor PHP ha ejecutado el código y ha generado el contenido dinámico (HTML, texto, imágenes, etc.), este contenido debe ser enviado de vuelta al navegador del usuario para que pueda ser mostrado.

Pasos clave en el envío de la respuesta: El servidor PHP, después de procesar el código, entrega el contenido generado generalmente código HTML al servidor web. Este contenido es el resultado de la ejecución del código PHP y contiene la información que se mostrará al usuario.

El servidor web prepara la respuesta recibiendo el contenido generado por PHP y lo empaqueta en una respuesta HTTP esta respuesta incluye:

Código de estado: Un número que indica si la solicitud fue exitosa (200 OK) o si hubo algún error (404 No encontrado, 500 Error interno del servidor, etc.).

Cabeceras: Información adicional sobre la respuesta, como el tipo de contenido (text/html, image/jpeg, etc.), la fecha y hora, y otras directivas.

Cuerpo: El contenido principal de la respuesta, que en este caso es el código HTML generado por PHP.

El servidor web envía la respuesta: El servidor web toma la respuesta HTTP completa y la envía de vuelta al navegador del usuario a través de la conexión de red.

El navegador recibe la respuesta: El navegador del usuario recibe la respuesta HTTP del servidor web donde interpreta la respuesta es decir el navegador analizara la respuesta HTTP, incluyendo el código de estado, las cabeceras y el cuerpo. Por último, pero no menos importante el navegador muestra el contenido enviado y lo interpreta con código HTML contenido en el cuerpo de la respuesta y muestra la página web al usuario.

1.4 Características principales de PHP

PHP tiene varias características que lo hacen destacar en el ámbito del desarrollo web entre las cuales resaltan las siguientes:

Lenguaje de scripting: Se ejecuta línea por línea en el servidor sin necesidad de compilación previa.

Incrustación en HTML: Permite mezclar código PHP con HTML, facilitando la creación de páginas dinámicas.

Multiplataforma: Funciona en diversos sistemas operativos como Windows, Linux y macOS.

Código abierto: Su código fuente está disponible para ser modificado y distribuido libremente.

Ventajas de usar PHP

Posee una relativa facilidad de aprendizaje debido a su sintaxis sencilla y accesible para principiantes cuentan también con una amplia gama de funcionalidades por ejemplo ofrece numerosas funciones y bibliotecas que facilitan diversas tareas en el desarrollo web. Además, su comunidad y frameworks de trabajo existentes cuenta con una gran cantidad de documentación, tutoriales y cursos disponibles en línea por lo que lo convierte en uno de los lenguajes de backend con gran popularidad.

Por ende, es uno de los lenguajes más utilizados para el desarrollo web, lo que genera una alta demanda laboral para desarrolladores.

Desventajas de usar PHP

A pesar de sus muchas ventajas, PHP también presenta algunos inconvenientes a nivel de seguridad que históricamente han sido un dolor de cabeza y se han tenido vulnerabilidades, aunque se han realizado mejoras significativas en versiones recientes para suplir estos impases.

Su rendimiento en aplicaciones web a gran escala, como concurrencias elevadas y alto tráfico web genera un problema cuando no son bien cuidados los roles e inicios de sesión, aunque se ha implementado técnicas de optimización que permitan solventar este inconveniente. Otro pequeño inconveniente dentro de este lenguaje de programación seria parte de su sintaxis es un tanto inconsistente ya que pueden ser confusas para los nuevos programadores debido a su falta de uniformidad y el bien mencionado código spaghetti.

A pesar de ello PHP es un lenguaje poderoso y versátil que ha demostrado ser fundamental en el desarrollo web moderno. Su capacidad para generar contenido dinámico y su facilidad de uso lo convierten en una opción atractiva tanto para principiantes como para desarrolladores experimentados. A medida que la tecnología avanza, PHP continúa evolucionando, adaptándose a las nuevas demandas del desarrollo web y manteniendo su relevancia en un entorno competitivo.

1.5 Características principales de un servidor web con PHP

Interpretación de código PHP: Capacidad para ejecutar código PHP y genera contenido dinámico.

Soporte para bases de datos: Conexión y manipulación de bases de datos como MySQL o MariaDB.

Gestión de archivos: Capacidad para leer, escribir y manipular archivos en el servidor.

Seguridad: Mecanismos para proteger el servidor y las aplicaciones web de ataques.

1.6 Ruta de carga de archivos y carpetas principales

La ruta de carga de archivos va a depender en gran medida de las configuraciones del servidor web que se ha instalado en el equipo.

Algunas ubicaciones comunes se establecen en la siguiente tabla:

Tabla 1
Rutas de carpetas importantes en la instalación

Server Web	Ruta	Observaciones

XAMPP	C:\xampp\htdocs\	Ofrece almacenamiento de componentes reutilizables en Windows.
WAMP	C:\wamp64\www\	Mayor control sobre la implementación de proyectos en Windows.
LAMP	/var/www/html/	Ofrece almacenamiento específico de proyectos en Linux
Laragon	C:\laragon\www\	Ofrece almacenamiento de componentes reutilizables en Windows.

***Nota**: Servidores web. **Fuente**: Autor*

Estas rutas de configuración del servidor web permiten el almacenar los archivos iniciales del proyecto que va a ser desarrollado en PHP y dependerá mucho del server web que escojamos para poder manipular estas rutas donde nuestros proyectos serán creados y editados en la medida que se necesite.

1.7 Carpetas principales:

htdocs o www: Contiene los archivos de las páginas web.

logs: Guarda los registros de actividad del servidor.

conf: Almacena los archivos de configuración del servidor este archivo contiene la preparación del ambiente funcional de tal forma que represente todas las configuraciones esenciales para la ejecución y correcto funcionamiento de los proyectos.

En resumen, las principales diferencias entre servidores web de PHP es las tecnologías y librerías que se pueden ejecutar una vez que el servidor se encuentre instalado. En el mundo del desarrollo web, la elección del servidor adecuado para ejecutar PHP y gestionar bases de datos MySQL es crucial. Aunque se puede configurar manualmente un entorno de desarrollo instalando PHP, MySQL y Apache, existen soluciones más prácticas y eficientes que permiten diseñar los paquetes de servidores PHP preconfigurados.

Estos paquetes simplifican enormemente el proceso de configuración, permitiéndote enfocarte en el desarrollo de tu proyecto sin preocuparte por los detalles técnicos.

1.8 Servidores PHP Preconfigurados

Existen varios servidores PHP que combinan las siguientes tecnologías micro instaladas y permiten generar proyectos con la mismas : PHP, MySQL y Apache en un solo paquete de instalación, facilitando la configuración y despliegue del servidor web en el equipo que se va a trabajar algunos de los más populares son:

XAMPP: Uno de los más conocidos y utilizados, disponible para Windows, macOS y Linux. Ofrece una configuración sencilla y es ideal para proyectos de desarrollo local.

WAMP: Similar a XAMPP, pero diseñado específicamente para Windows. Es muy fácil de usar y permite una configuración rápida de PHP, MySQL y Apache.

MAMP: La versión para macOS, que también incluye Perl además de PHP y MySQL.

Laragon: Es un paquete de desarrollo local que combina Apache, MySQL y PHP, diseñado específicamente para usuarios de Windows. Aunque XAMPP ha sido una opción popular durante mucho tiempo, Laragon ofrece una serie de ventajas que lo hacen destacar en el ámbito del desarrollo web moderno.

1.9 Servidores de Hosting PHP MySQL

Una vez el proyecto sea desarrollado debemos llevarlo a n ambiente de producción mismo que se no se ejecuta directamente en un servidor local y debemos por lógica alojarlo en algún sitio web en Internet, hay excelentes opciones de hosting que incluyen PHP y MySQL. Algunos de los mejores proveedores de hosting PHP MySQL incluyen:

A2 Hosting: Conocido por su alto rendimiento y fiabilidad, ofrece velocidades de carga rápidas y soporte para múltiples versiones de PHP y MySQL.

Hostinger: Ofrece un excelente equilibrio entre precio y características, con soporte para hasta 300 bases de datos MySQL y versiones recientes de PHP.

Bluehost: Es una opción popular por su facilidad de uso y precio asequible, ideal para proyectos pequeños y medianos.

2 Ventajas de Utilizar Servidores Preconfigurados

2.1 Facilidad de uso

No requieren conocimientos avanzados de configuración es decir que al momento de instalarlo el usuario tendrá ya preconfiguradas ciertas características que le permitan funcionar al servidor web de forma más sencilla (Cobo, Gómez, Pérez, & Rocha, 2005).

2.2 Tiempo de instalación reducido

Puedes tener tu entorno de desarrollo listo en minutos o horas dependiendo siempre de las configuraciones de la aplicación, pero netamente es más eficiente en tiempos de configuración básicas.

2.3 Compatibilidad

Garantizan la compatibilidad entre PHP, MySQL y Apache por lo general estas tecnologías son las que nos permiten tener paginas dinámicas y que se puede gestionar y administrar desde el servidor web en las rutas indicadas en el apartado de la tabla 1. (Cobo, Gómez, Pérez, & Rocha, 2005)

2.4 Soporte técnico

Muchos proveedores de hosting ofrecen soporte técnico para ayudarte con cualquier problema lo que conlleva a una solución sencilla y eficiente para desarrollar proyectos web con PHP y MySQL y mantenerlos en servicio 24/7 que es lo que se espera de una aplicación web.

Es decir, como se generan los paquetes de servidores preconfigurados o los servicios de hosting especializados son opciones excelentes. Ambas alternativas te permiten enfocarte en el desarrollo de tu proyecto sin preocuparte por la complejidad de la configuración técnica que se encuentra en cada framework de desarrollo.

Tabla 2

PHP frameworks de trabajo.

FRAMEWORKS PHP	Laravel	Symfony	CodeIgniter
Fecha de lanzamiento	2011	2005	2006
Comunidad	Grande	Grande	Media
Documentación	Excelente	Excelente	Media
Velocidad	Rápido	Rápido	Rápido
Complejidad de aprendizaje	Alta	Alta	Baja/Media
Personalización	Baja	Alta	Media
Seguridad	Alta	Alta	Media
Soporte para móviles	Sí	Sí	No
Soporte para APIs	Sí	Sí	Sí
Compatibilidad con bases de datos	MySQL, PostgreSQL, SQLite, SQL Server	MySQL, PostgreSQL, Oracle, SQL Server, SQLite	MySQL, PostgreSQL, Oracle, SQLite, Microsoft BI

***Nota**: Frameworks de PHP. **Fuente**: Autor*

Esta diferenciación es fundamental para elegir la herramienta adecuada para cada proyecto, teniendo en cuenta las necesidades específicas y las preferencias del equipo de desarrollo.

Oficialmente, PHP se ha caracterizado en utilizar desde el año 2005 un framework como Symfony que no fue tan adaptada a algunas soluciones por su novel de complejidad en su instalación y uso , en si complicando el desorden que conlleva trabajar con PHP que a su vez se fue volviendo obsoleto pero que en su momento fue el primer framework de trabajo que no utilizaba PHP puro para su funcionamiento y que tenía un proceso complejo de aptar por ello es que fue cayendo en ese proceso de obsolescencia, luego se decantaron por la utilización de

CodeIgniter que en fue un gran avance y su forma de trabajar se volvió ya un proceso estructural que manejaba sus capas además de la instalación y manejo era mucho mas eficiente en servidores web que manejen un entorno LAMP , sus facilidades de instalación por comandos lo hacían apto para el manejo , actualización y generación de proyectos y a medida que el tiempo pasaba iban avanzando con su mejora en formas de manejar proyectos a nivel empresarial.

Otra diferencia es que PHP para crear una aplicación puede realizarlo desde cero con solo realizar una mezcla entre HTML y el backend configurado, en cambio si utilizamos un framework de trabajo siempre te deja la libertad de elegir qué empaquetador usar y ofrece diferentes opciones y dependiendo del servidor web en este caso composer será el elegido.

Aun así, existe personas que consideran la elección de un framework de trabajo es un tanto limitante por que impone un orden de trabajo. Aunque no hay un cambio constante en su forma de trabajar hoy por hoy el framework más utilizado de PHP se debate entre Laravel y CodeIgniter , la mayoría de la comunidad de desarrolladores considera que un framework es una biblioteca que incluye otras soluciones para crear una aplicación completa con reglas claras y casi sin configuración. (Carrión, Noriega, & Castillo, 2019) Por ejemplo, Symfony no se podría considerar como bibliotecas actualizadas por el mismo hecho de su descontinuación pero Laravel y un sistema de enrutado, un sistema de reorganizado del lado del servidor que permitirá mantener un estructura de diseño con mejores resultados.

3 Frameworks PHP

Estos frameworks brindan todas las características que necesitas para desarrollar e implementar y escalar en tu aplicación en producción y están trabajando para respaldar la visión de una arquitectura fullstack para el backend. Los frameworks PHP son estructuras predefinidas que proporcionan un conjunto de herramientas y componentes reutilizables para agilizar el desarrollo de aplicaciones web. Permiten a los desarrolladores centrarse en la lógica de negocio en lugar de reinventar la rueda con tareas comunes como el enrutamiento, la gestión de bases de datos y la seguridad.

3.1 CodeIgniter

CodeIgniter es un framework ligero y rápido, conocido por su simplicidad y facilidad de uso. Es una excelente opción para principiantes y proyectos que requieren un rendimiento óptimo (Sánchez Cano, 2018).

3.1.1 Características Clave

Ligero y Rápido: Ideal para proyectos pequeños y medianos que necesitan un buen rendimiento.

Fácil de Aprender: Ofrece una curva de aprendizaje suave y documentación

Flexibilidad: Permite personalizar fácilmente el framework según las necesidades del proyecto.

- Ventajas:

Ligero y rápido.

Fácil de aprender y utilizar.

Excelente rendimiento.

- Desventajas:

No incluye tantas características integradas como Laravel.

3.2 Symfony

Symfony es un framework robusto y flexible, conocido por su arquitectura modular y alto rendimiento. Es ideal para el desarrollo de aplicaciones web empresariales y proyectos que requieren un alto nivel de personalización (Soluciones digitales a medida, 2024).

3.2.1 Características Clave

Arquitectura Modular: Permite crear aplicaciones escalables y mantenibles.

Componentes Reutilizables: Facilita la integración de componentes en otros proyectos.

Alto Rendimiento: Ideal para aplicaciones complejas y de gran escala.

- Ventajas:

Arquitectura modular y flexible.

Alto rendimiento.

Excelente para aplicaciones web empresariales.

- Desventajas:

Puede ser complejo de aprender y configurar.

4 Elección del Framework Adecuado

La elección del framework adecuado depende de las necesidades específicas de tu proyecto, ya que este puede variar dependiendo de las funcionalidades técnicas del proyecto como lo son almacenamiento, concurrencia y escalabilidad a continuación se explican los criterios de selección para el framework adecuado teniendo en cuenta que esta es una mera observación subjetiva y como lo indico debe ser sujeto a pruebas y error para delimitar el correcto alcance del proyecto, es de mi agrado sugerir el uso de Laravel ya que este se mantiene en constante evolución lo cual a permitido que gane terreno en el desarrollo web (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023).

CodeIgniter: Ideal para proyectos ligeros y rápidos que requieren simplicidad y rendimiento.

Symfony: Ideal para proyectos empresariales y complejos que requieren flexibilidad y personalización (Sánchez Cano, 2018).

Laravel: Ideal para proyectos modernos y complejos que requieren una sintaxis elegante y una amplia gama de características.

4.1 Consejos para desarrolladores

Si eres nuevo en el desarrollo web, CodeIgniter es una excelente opción para aprender los conceptos básicos de los frameworks PHP. Una vez que tengas experiencia con PHP y MVC, Laravel es ideal para proyectos más complejos. Si te interesa el desarrollo web de gran escala en rendimiento, Symfony es una buena opción para investigar ya que no cuenta con tanta demanda por que los 2 primeros han venido surgiendo en el ámbito empresarial (Aguirre, 2021).

En resumen, cada framework tiene sus ventajas y desventajas. Al entender estas características, podrás elegir el que mejor se adapte a tus necesidades y habilidades, asegurando un desarrollo web eficiente y exitoso.

4.2 Resumen del Capítulo 1

En el primer capítulo de este libro hemos visto una breve reseña de cada hito importante en la historia de PHP de cómo fue un lenguaje de programación backend que logro despuntar con cada una de sus mejoras hasta llegar a un punto de estancamiento hasta el aparecimiento de los frameworks de trabajo que permitieron tener un mejor interprete y ahorrar tiempo en temas de configuración, lo cual nos permitió revisar la descripción del contenido sobre la creación de servidores de páginas web y sus estructuras son parte de los temas principales que trata PHP de abordar y mejorarlas en el capítulo también hicimos una

breve reseña histórica que permitió comprender como y el por qué se creó esta lenguaje ya que su creador decidió optimizar el proceso de integración de las páginas estáticas de HTML con una alternativa que permita diseñar de forma eficiente , practica y sencilla la creación de páginas web de forma tal que esto se adaptó a los nuevos sitios esto gracias al apoyo visionario del desarrollador Ramus Leadorf el cual busco proporcionar una visión general de la integración de funcionalidades de diseño de HTML y dinamismo de PHP y sus distintos enfoques y metodologías para desarrollo del lado del backend, y no solo esta sino muchas otras como lo hemos visto permitiendo así tener una clara idea para la mejora continua de resultados en las páginas web que manejamos en la actualidad.

En si podemos darnos cuenta de el correcto uso de los del servidor web y sus funcionalidades permiten a PHP, ser un lenguaje de programación más adaptativos lo que requiere compromiso y un trabajo de implementación adaptativo dependiendo del servidor web que se vaya a utilizar. Por el momento, es necesario saber el alcance de cada proyecto y como adminístralos para guiar a buen puerto la creación del sitio. A su vez hay que mencionar que el equipo de PHP está trabajando con los desarrolladores de paquetes para que cada una de las características sean más fáciles de implementar en la próxima generación de frameworks y así este lenguaje de programación de un salto de calidad tan esperado, aun que como bien indique en apartados anteriores del capítulo varias personas ya manejan este lenguaje de forma natural o bien con un framework de desarrollo backend.

Por último, pero no menos importante hemos revisado la clasificación actual de frameworks que trabajan con PHP y permiten su creación esto es el pie de inicio en nuestro camino a la creación de proyectos con PHP y la forma de saber implementar la herramienta que se va a usar para la creación del sitio web, esto determinara en gran medida que framework es el que nos beneficia para su mantenimiento a lo largo del tiempo, es así que manteniendo esta tendencia podremos crear páginas web a la vanguardia y con una documentación que está en auge en el ámbito del desarrollo.

5 Capítulo 2

Instalación y Administración de Laravel un framework modernizado de PHP.

5.1 Introducción a los Frameworks PHP: La piedra angular de la eficiencia en el desarrollo web.

En el mundo del desarrollo web, los frameworks PHP son herramientas esenciales que permiten a los desarrolladores crear aplicaciones web robustas y escalables de manera eficiente. Estos frameworks proporcionan una estructura predefinida, componentes reutilizables y convenciones que agilizan el proceso de desarrollo, permitiendo a los programadores centrarse en la lógica de negocio en lugar de reinventar la rueda con tareas comunes como el enrutamiento, la gestión de bases de datos y la seguridad.

5.1.1 ¿Qué es un Framework PHP?

Un framework PHP es un conjunto de herramientas y bibliotecas que facilitan el desarrollo de aplicaciones web al proporcionar una base sólida sobre la cual construir. En lugar de escribir código desde cero, los desarrolladores pueden aprovechar las funcionalidades del framework para crear aplicaciones más rápido y con mayor calidad. Esto no solo reduce el tiempo de desarrollo, sino que también mejora la consistencia y la mantenibilidad del código.

5.1.2 Ventajas de Utilizar Frameworks PHP

Reducción del Tiempo de Desarrollo: Al utilizar componentes predefinidos, los desarrolladores pueden ahorrar tiempo y enfocarse en la lógica de negocio del proyecto.

Mejora de la Calidad del Código: Los frameworks promueven prácticas de codificación estándar y estructuras organizadas, lo que resulta en un código más legible y mantenible.

Seguridad: Muchos frameworks incluyen características de seguridad integradas que ayudan a proteger las aplicaciones contra vulnerabilidades comunes.

Comunidad Activa: La mayoría de los frameworks tienen comunidades activas que ofrecen soporte, documentación y recursos adicionales.

5.2 Laravel: Un Framework Moderno y Elegante

Entre los frameworks PHP más populares, Laravel destaca por su sintaxis expresiva y su amplia gama de características integradas. Fue creado por Taylor Otwell en 2011 y ha ganado popularidad rápidamente debido a su facilidad de uso y su potente ecosistema de paquetes.

Figura 8

Taylor Otwell Creador de Laravel

Nota: *Entrevista del creador de Laravel para la revista AR de dinero y políticas.* ***Fuente****: (armoneyandpolitics, s.f.)*

5.3 Características Clave de Laravel

5.3.1 Sistema de Rutas de Laravel: ¡El mapa de tu aplicación!

En el desarrollo web de PHP debemos imaginar que tu aplicación web es una ciudad y que cada página o función es un edificio que pertenece a la misma. El sistema de rutas de Laravel es como el mapa de esa ciudad, que indica cómo llegar a cada edificio. Pero en si ¿Qué hace el sistema de rutas? Tomamos a la referencia de un mapa o guía que en si son las palabras claves en este apartado es decir que debemos redireccionar el manejo dinámico de nuestra aplicación web a través de los siguientes pasos:

Define las URLs: Asocia una URL (como /usuarios o /productos/123) con una función o un controlador que se ejecutará cuando se acceda a esa URL.

Dirige el tráfico: Decide qué código se ejecutará en función de la URL solicitada por el usuario.

Organiza la aplicación: Facilita la creación de aplicaciones web estructuradas y fáciles de mantener.

5.3.2 ¿Cómo funciona el enrutamiento?

Solicitud del usuario: El usuario ingresa una URL en su navegador y envía una solicitud al servidor web.

Enrutamiento: Laravel recibe la solicitud y analiza la URL.

Coincidencia de ruta: Laravel compara la URL con las rutas definidas en los archivos de rutas (generalmente routes/web.php para rutas web y routes/api.php para rutas de API).

Ejecución del controlador o función: Si se encuentra una ruta coincidente, Laravel ejecuta el controlador o la función asociada a esa ruta.

Respuesta: El controlador o la función genera una respuesta (HTML, JSON, etc.) que se envía al navegador del usuario.

5.3.3 Características clave del sistema de rutas de Laravel

Rutas simples: Permite definir rutas básicas con una URL y una función o controlador. Ejemplo: Route::get('/saludo', function () { return '¡Hola!'; });

Rutas con parámetros: Permite capturar valores dinámicos de la URL. Ejemplo: Route::get('/usuarios/{id}', [UserController::class, 'show']);

Rutas con nombres: Permite asignar nombres a las rutas para generar URLs de forma dinámica.Ejemplo: Route::get('/perfil', [UserController::class, 'perfil'])->name('perfil');

Agrupamiento de rutas: Permite agrupar rutas con características comunes (middleware, prefijos, etc.) Ejemplo: Route::middleware(['auth'])->group(function () { Route::get('/dashboard', [DashboardController::class, 'index']); });

Rutas de recursos: Permite generar automáticamente rutas para operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en un modelo. Ejemplo: Route::resource('fotos', FotoController::class);

5.3.4 Beneficios del sistema de rutas de Laravel

Flexibilidad: Permite definir rutas simples y complejas según las necesidades de la aplicación.

Organización: Facilita la creación de aplicaciones web estructuradas y fáciles de mantener.

Legibilidad: La sintaxis clara y concisa facilita la lectura y comprensión de las rutas.

Mantenibilidad: Los cambios en las rutas no afectan a otras partes de la aplicación.

El sistema de rutas de Laravel es un proceso de gran valía por el cual permite ordenar la navegación de tu sitio y de igual forma permite definir cómo se accede a las diferentes partes de tu aplicación web ampliando su flexibilidad y facilidad de uso lo convierten en una

característica idónea de Laravel. Ideal para proyectos de desarrollo modernos y complejos estas opciones de laravel son perfectos para aplicaciones web que requieren una sintaxis elegante y una amplia gama de características integradas.

La gran cantidad de paquetes con que Laravel cuenta permite mantener una comunidad activa y un ecosistema de paquetes robusto que facilitan la integración de nuevas tecnologías y servicios dado al gran número de sus versiones y actualizaciones que cuentan con mejora continua. Aunque puede ser complejo para principiantes, Laravel es relativamente fácil de aprender para desarrolladores con experiencia en PHP y MVC (Yoris, 2022).

Requiere un buen conocimiento de PHP y MVC para aprovechar al máximo sus características. Esto puede consumir más recursos del sistema en comparación con frameworks más ligeros como CodeIgniter, aunque definir rutas y controladores para manejar las solicitudes HTTP no sea lo ideal en este framework.

Laravel plantea el uso de rutas para mejorar el manejo de recursos además se da por configurado las opciones del ambiente en el archivo reservado ".env" de su acrónimo en ingles enviroment que permite definir rutas de trabajo y una mejor manera en aprovechamiento del lenguaje de programación y el modelo de trabajo MVC.

5.4 Eloquent ORM: ¡Tu puente a la base de datos!

ORM (Object-Relational Mapping) es una técnica que permite interactuar con bases de datos utilizando objetos en lugar de consultas SQL directas. Eloquent ORM es el ORM incluido en Laravel, que proporciona una capa de abstracción para simplificar y agilizar las operaciones de base de datos. Al utilizar Eloquent se puede interactuar con la base de datos de manera eficiente debido a que Eloquent ORM proporciona una capa de abstracción, permitiendo realizar operaciones de base de datos en forma de objetos PHP (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023).

Es decir que en lugar de escribir consultas en SQL como ejemplo:

Figura 9

Consulta en lenguaje SQL

```sql
SELECT * FROM usuarios WHERE id = 1;
```

Nota: *Consulta en SQL a la tabla usuarios.* ***Fuente****: Autor*

Esta consulta SQL se la puede realizar dentro del objeto modelo como se indica en el siguiente ejemplo, por ello puedes revisar el usuario con ID 1 de la tabla Usuario con la siguiente sentencia y manejándolo dentro de tu aplicación obviamente hay que realizar una serie de configuraciones en ".env" para apuntar a la base de datos a la que vamos a afectar aquí se indica un ejemplo de la instrucción de la consulta en objeto de php:

Figura 10

Consulta en Eloquent ORM

```php
$usuario = Usuario::find(1);
```

Nota: Consulta realizada en formato de objeto de PHP. **Fuente**: Autor

Donde ORM significa mapeo de objeto relacional y se encarga de realizar una subconsulta por debajo en el modelo y arrojar el valor en una variable directamente a tu aplicación web.

5.4.1 ¿Cómo funciona Eloquent?

Modelos: Eloquent utiliza modelos, que son clases PHP que representan tablas en la base de datos. Cada modelo tiene propiedades y métodos que corresponden a las columnas y operaciones de la tabla.

Mapeo: Eloquent mapea las tablas y columnas de la base de datos a los modelos y sus propiedades.

Operaciones: Eloquent proporciona métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos, así como para realizar consultas complejas.

Relaciones: Eloquent permite definir relaciones entre modelos, como uno a uno, uno a muchos y muchos a muchos, lo que facilita la gestión de datos relacionados.

5.4.2 Beneficios de usar Eloquent:

- Sintaxis intuitiva: Eloquent utiliza una sintaxis clara y concisa que facilita la lectura y escritura de código.
- Abstracción: Eloquent oculta los detalles de la base de datos, lo que permite a los desarrolladores centrarse en la lógica de la aplicación.
- Seguridad: Eloquent ayuda a prevenir inyecciones SQL y otras vulnerabilidades de seguridad.
- Productividad: Eloquent agiliza el desarrollo al proporcionar métodos predefinidos para tareas comunes.
- Mantenibilidad: Eloquent facilita la organización y el mantenimiento del código relacionado con la base de datos.

Ejemplos de uso de Eloquent:

En base a lo indicado anteriormente sobre el manejo de modelos dentro de Laravel que podemos indicar el proceso CRUD de las instrucciones en los ejemplos de Eloquent ORM y permitirnos ejecutar consultas de uso cotidiano dentro de nuestra aplicación a continuación se ingresara con los ejemplos de Eloquent para el manejo de un CRUD en la tabla usuarios. (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023).

Figura 11

Obtención de un registro con Eloquent

```php
$usuario = Usuario::find(1);
```

Nota: *Consulta realizada en formato de objeto de PHP.* ***Fuente:*** *Autor*

Figura 12

Proceso de creación de un nuevo registro con Eloquent

```php
$usuario = new Usuario;
$usuario->nombre = 'Juan';
$usuario->email = 'juan@example.com';
$usuario->save();
```

Nota: *Consulta realizada en formato de objeto de PHP.* ***Fuente****: Autor*

Figura 13

Proceso de actualización de un registro con Eloquent

```php
$usuario = Usuario::find(1);
$usuario->email = 'nuevo_email@example.com';
$usuario->save();
```

Nota: *Consulta realizada en formato de objeto de PHP.* ***Fuente****: Autor*

Figura 14

Proceso de eliminación de un registro con Eloquent

```php
$usuario = Usuario::find(1);
$usuario->delete();
```

Nota: *Consulta realizada en formato de objeto de PHP.* ***Fuente****: Autor*

Dentro de Laravel también existen otro tipo de bondades como es el manejo de plantillas que permitan estandarizar la creación del diseño web del proyecto permitiendo así manejarlo de manera siguiente manera:

5.5 Introducción a Blade: El Motor de Plantillas de Laravel

Blade es el motor de plantillas integrado en Laravel, diseñado para facilitar la creación de vistas dinámicas y reutilizables en aplicaciones web. A diferencia de otros motores de plantillas, Blade no restringe el uso de código PHP en tus plantillas, lo que significa que puedes

combinar HTML y PHP de manera fluida. Esto permite a los desarrolladores escribir código limpio y legible, manteniendo toda la potencia del lenguaje PHP (Soluciones digitales a medida, 2024).

5.5.1 Características Clave de Blade

Blade ofrece una serie de características que lo hacen destacar en el desarrollo web. Una de sus principales ventajas es su capacidad para heredar plantillas, lo que evita la duplicación de código y mejora la eficiencia del desarrollo en cuanto al diseño web se refiere. Además, Blade permite definir secciones en plantillas maestras que pueden ser reemplazadas o extendidas en plantillas hijas, lo que facilita la creación de diseños más complejos de manera sencilla. Otra característica importante es su sintaxis concisa, que utiliza directivas como @if, @foreach y @include para simplificar la escritura de código PHP (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023).

5.5.2 Directivas Más Utilizadas en Blade

Blade utiliza una serie de directivas que facilitan la creación de vistas dinámicas. estas directivas permiten manejar de forma idónea el diseño ya que va a facilitar la reutilización del código creado las directivas más comunes son @extends, @section y @yield, las cuales permiten construir una estructura flexible para las vistas. La directiva @extends es fundamental cuando quieres que una vista herede de una plantilla principal. Por otro lado, @section se utiliza para definir el contenido que llenará las áreas específicas de esa plantilla. Finalmente, @yield actúa como un marcador de posición en la plantilla maestra, esperando que otras vistas “inserten” contenido en esas áreas (Yoris, 2022).

Blade es ideal para desarrolladores que buscan crear aplicaciones web modernas y escalables. Su capacidad para compilar plantillas en código PHP puro y almacenarlas en caché mejora significativamente el rendimiento de las aplicaciones, ya que no añade sobrecarga adicional. Además, Blade permite crear componentes reutilizables que encapsulan lógica y HTML, lo que facilita la organización y mantenimiento del código. Por último, su escapado automático de contenido HTML ayuda a prevenir ataques XSS, mejorando la seguridad de las aplicaciones. En resumen, Blade es una herramienta que simplifica la creación de interfaces de usuario complejas y eficientes en Laravel.

Ejemplos de uso de Blade

Figura 15

Proceso de presentación de una variable en Blade

```
Blade
<h1>{{ $nombre }}</h1>
```

Nota: *Consulta realizada en formato de objeto de PHP en las vistas.* ***Fuente****: Autor*

Figura 16

Proceso de presentación de una variable en Blade con condicional

```
Blade
@if ($usuario->esAdmin())
    <p>El usuario es administrador.</p>
@else
    <p>El usuario no es administrador.</p>
@endif
```

Nota: *Consulta realizada en formato de objeto de PHP en las vistas.* ***Fuente:*** *Autor*

Figura 17

Proceso de repetición de una variable tipo vector en Blade

```
Blade
@foreach ($usuarios as $usuario)
    <p>{{ $usuario->nombre }}</p>
@endforeach
```

Nota: Consulta realizada en formato de objeto de PHP en las vistas. **Fuente:** Autor

Figura 18

Proceso de inclusión de una pantalla

```
Blade
@include('componentes.boton', ['texto' => 'Clic aquí'])
```

Nota: *Consulta realizada en formato de objeto de PHP en las vistas.* ***Fuente:*** *Autor*

5.6 Autenticación y Autorización en Laravel: Seguridad simplificada

Laravel proporciona un sistema de autenticación robusto y flexible que te permite implementar características de seguridad como:

- Registro e inicio de sesión de usuarios.
- Restablecimiento de contraseñas.
- Verificación de correo electrónico.
- Autorización de acceso a recursos.

5.7 Última versión de Laravel

Para mantener la información lo más actualizada posible, es recomendable que se realice la consulta de la documentación oficial de Laravel, ya que las versiones se actualizan con frecuencia. A fecha de hoy 2025, nos encontramos en la versión 12.x. Se puede encontrar la información más reciente en el sitio web oficial de Laravel: laravel.com

6 Prerrequisitos para el Entorno de Desarrollo de Laravel.

Dentro del proceso de instalación del lenguaje de programación se consideran que el equipo de trabajo donde vamos a iniciar con nuestros proyectos debe estar configurado y debe tener instalado los siguientes prerrequisitos para establecer un entorno de desarrollo óptimo para PHP y Laravel que es el framework que utilizaremos en nuestros proyectos (Yoris, 2022), es fundamental contar con los siguientes elementos que me van a permitir contar con un ambiente de desarrollo que sea apto para poder generar y crear proyectos en laravel:

6.1 Guía para Empezar con Laravel

Instalación: Instalar Laragon que es un servidor web que permitirá el trabajo fluido y la creación y administración de los proyectos de laravel.

Instalación: Utiliza Composer para instalar Laravel. Puedes crear un nuevo proyecto con el comando composer create-project laravel/laravel mi-proyecto.

Configuración Básica: Configura el archivo. env para establecer las variables de entorno, como la conexión a la base de datos.

Estructura del Proyecto: Familiarízate con la estructura de carpetas de Laravel y cómo organizar tu código.

En cuanto a lo que tiene que ver al paquete de implementación y en su defecto de configuración de laravel ya que permite administrar los paquetes necesarios para el funcionamiento de la herramienta la cual abordaremos en breve.

6.2 Instalación de Server Web: Laragon

Se puede definir a Laragon como un entorno de desarrollo local portátil, aislado y rápido para Windows. Esta herramienta hará la diferencia que simplifica la configuración de un servidor web local, y en si lo tomaremos como un servidor local web que permite a los desarrolladores ejecutar aplicaciones web sin la complejidad de configurar manualmente Apache, MySQL, PHP y otros componentes.

Se centra en el rendimiento, la simplicidad y la flexibilidad, ofreciendo un entorno de desarrollo ligero y fácil de usar.

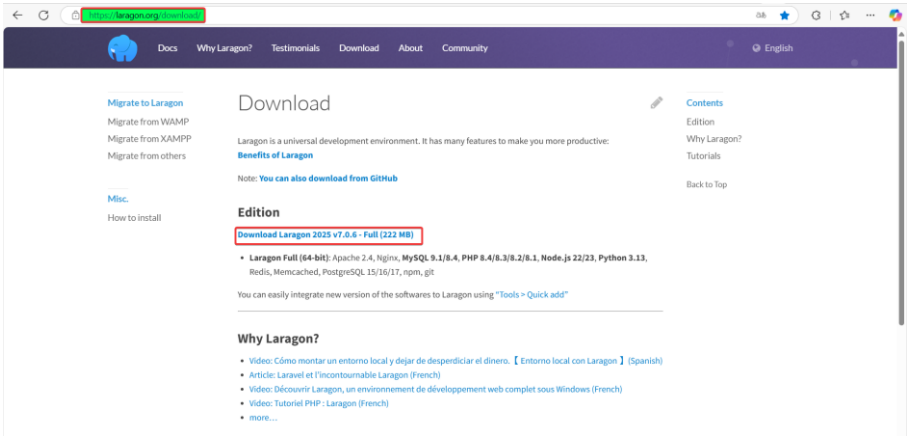
6.2.1 ¿Quién lo creó y por qué?

Laragon fue creado para simplificar el proceso de configuración de entornos de desarrollo en Windows, que tradicionalmente podía ser complicado y consumir mucho tiempo. Su principal objetivo es proporcionar a los desarrolladores una herramienta que sea versátil y que les permita concentrarse en la codificación en lugar de lidiar con problemas de configuración.

6.2.2 Instalación en Windows:

Figura 19

Laragon opción de descargas

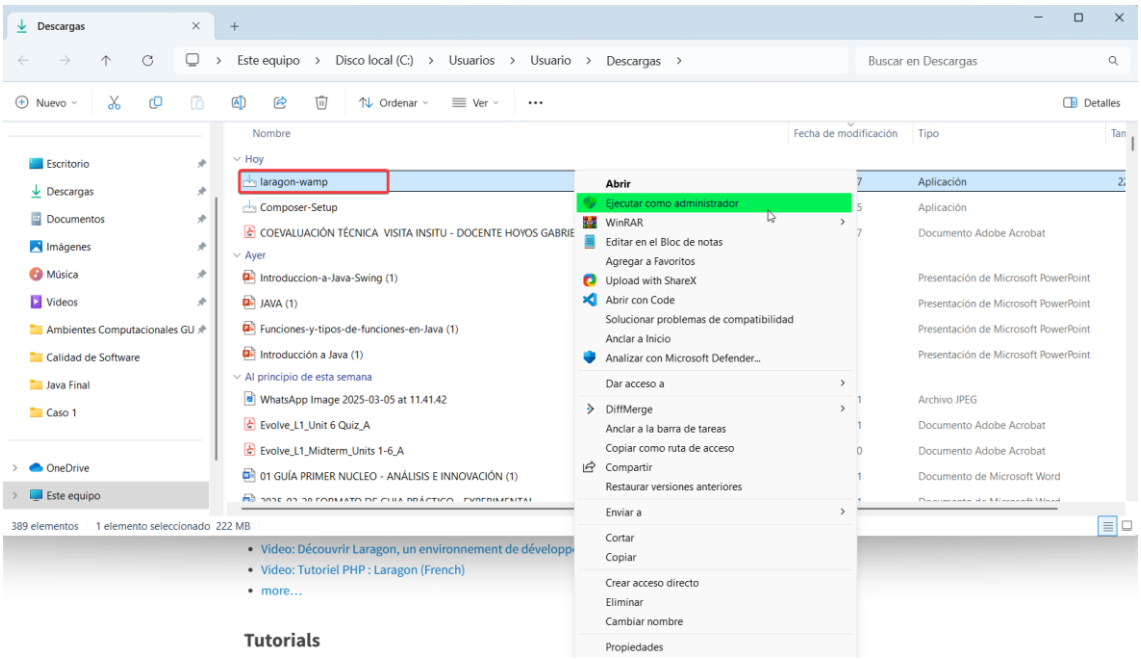


Nota: *Pagina web oficial de Laragon.* ***Fuente:*** *Autor*

Para el proceso de descarga e instalación de Laragon debemos visitar el sitio web oficial (laragon.org) y ubicarnos en la parte de descargas donde buscaremos la versión más reciente y estable de esta herramienta.

Figura 20

Instalación de ejecutable de Laragon.

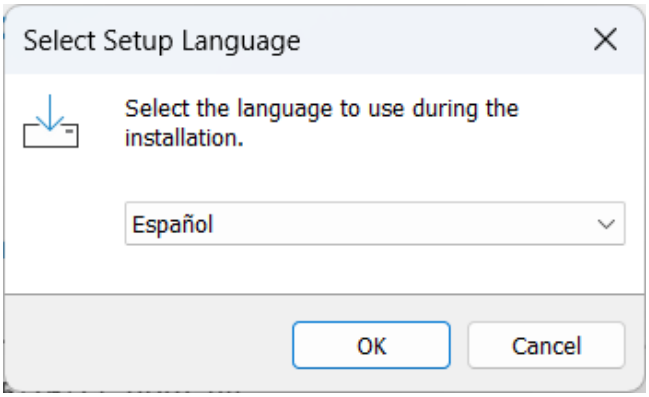


Nota: *Ubicación de descargas para instalación de ejecutable.* ***Fuente:*** *Autor*

Ejecutaremos el archivo de instalación descargado, para ello nos ubicaremos en la siguiente ruta: “C:\Users\Usuario\Downloads” y ubicaremos el archivo llamado “Laragon-warp.exe” daremos clic derecho y ejecutar como administrador para continuar con el proceso del asistente de instalación, seleccionando la ubicación de instalación y las opciones deseadas.

Figura 21

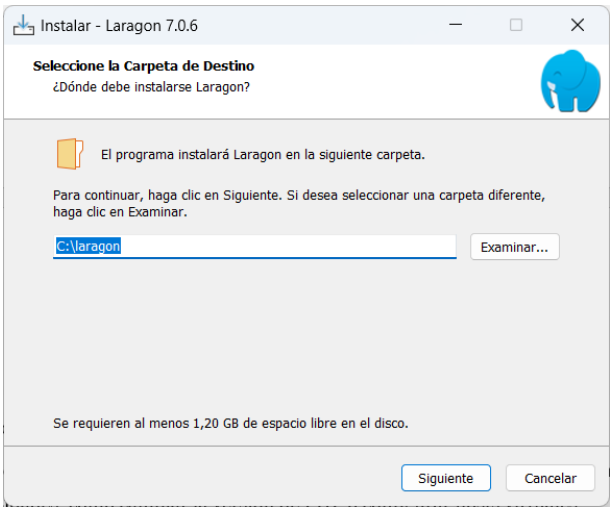
Instalación Laragon parte 1:



Nota: *Seleccionamos el lenguaje de instalación.* ***Fuente:*** *Autor*

Figura 22

Instalación de Laragon parte 2

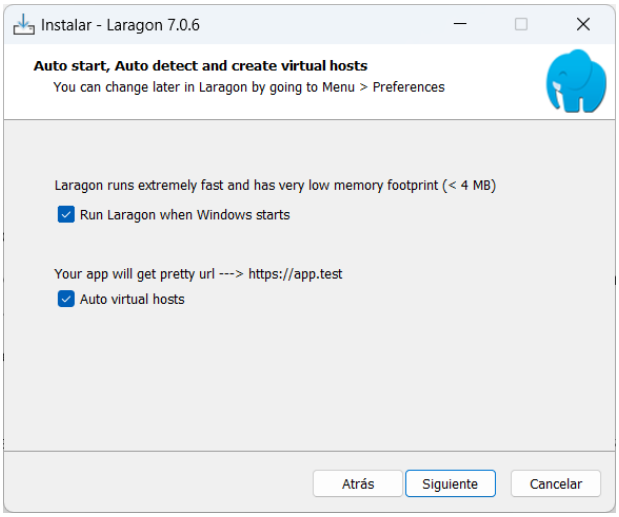


Nota: *Seleccionamos la ruta de instalación de Laragon.* ***Fuente****: Autor*

En esta configuración del proceso de instalación es importante definir esta ruta por que es necesaria para posteriores configuraciones y manejo de los proyectos de laravel.

Figura 23

Instalación de Laragon parte 3



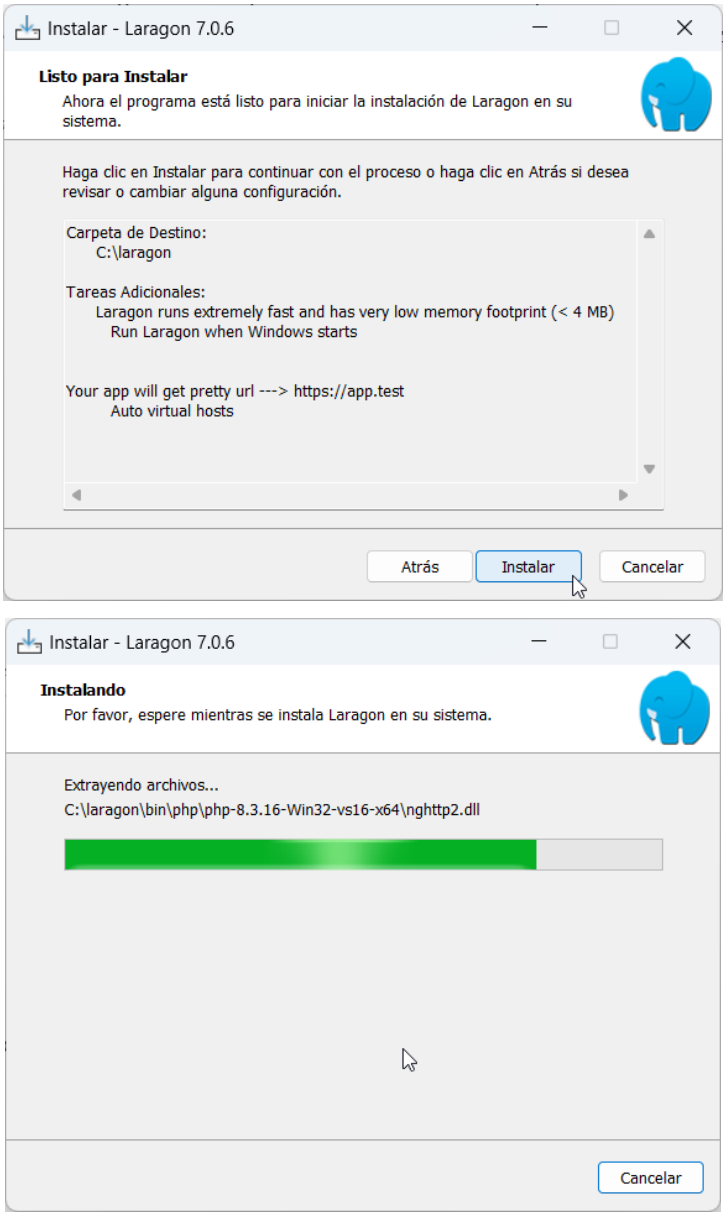
Nota: *Seleccionamos la configuración por defecto de Laragon.* ***Fuente****: Autor*

A continuación, damos clic en siguiente y procedemos con las instalación de todas las configuraciones realizadas con anterioridad, para poder continuar con el proceso de de

instalación vas a dar clic en instalar y esperaremos a que esta termine como se adjunta en las Figura 24.

Figura 24

Proceso de Instalación de paquetes de Laragon parte 4.



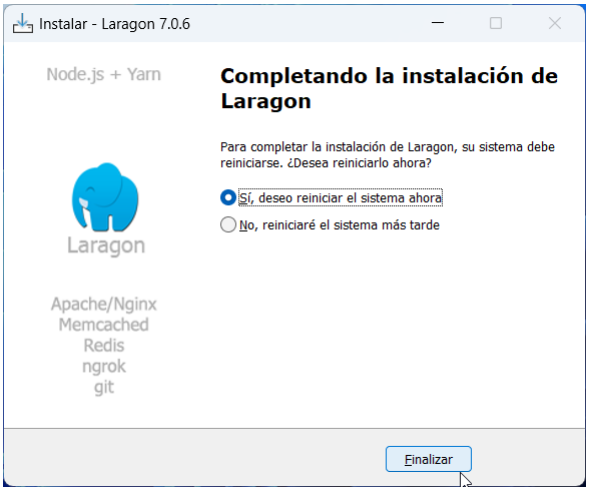
Nota: *instalación de PHP y sus versiones con Laragon.* ***Fuente:*** *Autor*

Nótese que el proceso es relativamente rápido y Laragon al ser un servidor con gran variedad de herramientas para utilizar en el desarrollo permite la gestión y visualización de estas una vez que sea ha dado clic en la opción de instalar a su vez podremos verificar sus versiones una vez iniciemos con el server y sus configuraciones. Al Finalizar el proceso de instalación se

genera un launcher en Windows que nos indica reiniciar el equipo por lo que es recomendable tomar esta opción para que pueda reflejarse todos los cambios realizados en el proceso de instalación y configuraciones de Laragon.

Figura 25

Proceso de instalación Laragon finalizado parte 5

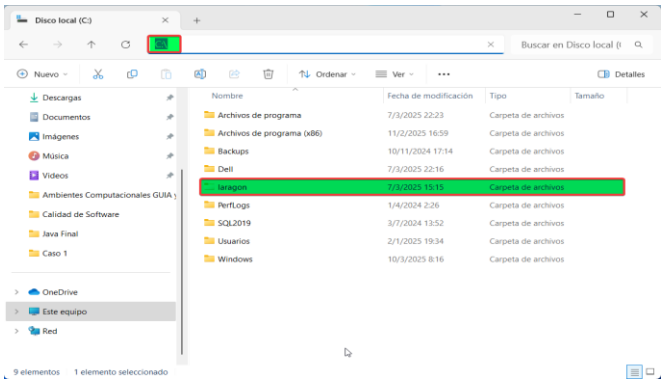


Nota: *Finalización del proceso de instalación de Laragon.* ***Fuente:*** *Autor*

Después de la instalación, puedes personalizar la configuración de Laragon según tus necesidades, como cambiar la versión de PHP o configurar hosts virtuales. Como podemos divisar en la Figura 26, Laragon se instala en una la ruta que seleccionamos en lo pasos posteriores y es aquí donde PHP se va a ubicar.

Figura 26

Ubicación física de Laragon y Php.



Nota: *Carpeta de Ubicación de Laragon.* ***Fuente:*** *Autor*

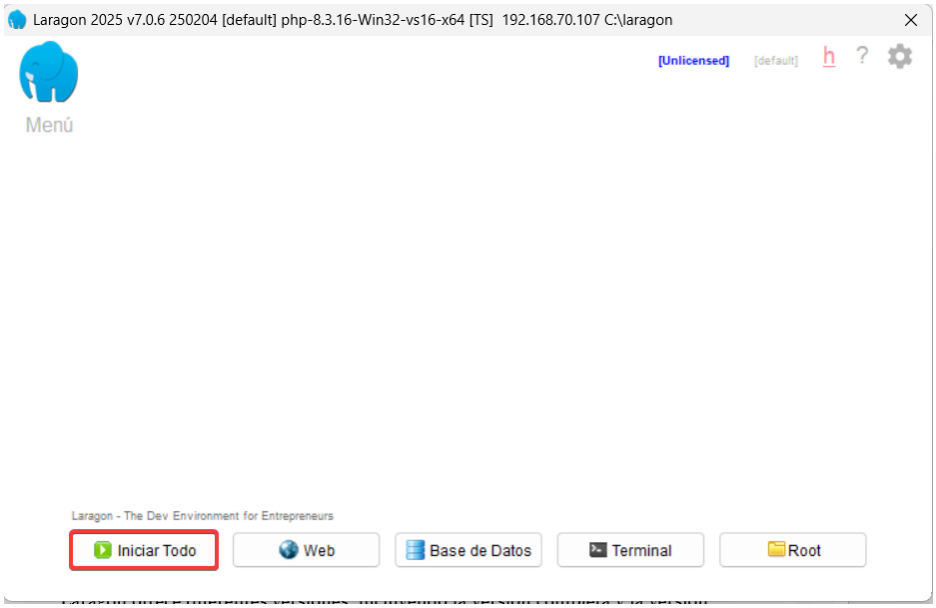
A continuación, indicaremos las rutas importantes de esta carpeta donde nosotros vamos a tener un ambiente de trabajo, donde se van a ubicar los proyectos y como se van a

realizar las configuraciones de esta herramienta, misma que cuenta con una consola grafica que se encuentra referenciada en la Figura 27 misma que permite revisar, actualizar y mejorar la administración e inicialización del servidor web según nuestras necesidades.

La administración de los proyectos y herramientas de Laragon permiten tener una mejor forma de gestionar los recursos necesarios con una interfaz amigable e intuitiva, si bien tiene una similitud con XAMPP este tiene sus configuraciones por archivos en ruta, mientras que Laragon permite realizar la configuración desde la misma consola y presentar e inicializar los servidores con un solo clic.

Figura 27

Consola de Laragon



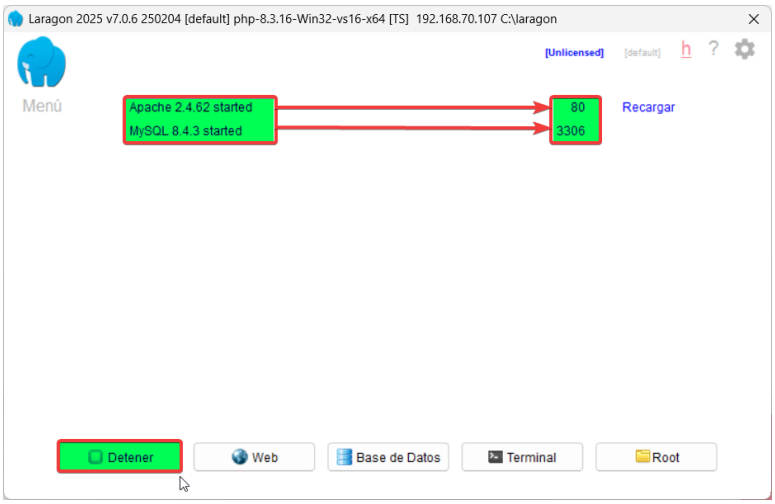
***Nota**: Proceso de inicialización de la consola de Laragon. **Fuente**: Autor*

A continuación, vamos a dar clic en iniciar y automáticamente los indicadores de las rutas importantes acceden al ambiente para empezar a trabajar con el server como lo es Apache , y con la Base de datos MySQL que levantara cada servicio con su respectivo puerto de ejecución en este caso Apache en el puerto 80 y la base de datos en el puerto 3306 que son los que comúnmente se utilizan a lo largo del desarrollo mismo que se indica en la Figura 28 de este documento.

La administración de las herramientas de Laragon mediante la consola permite tener una mejor perspectiva para gestionar los recursos necesarios y realizar las configuraciones pertinentes mediante la consola con una interfaz amigable e intuitiva.

Figura 28

Levantamiento de Servicios de Laragon.

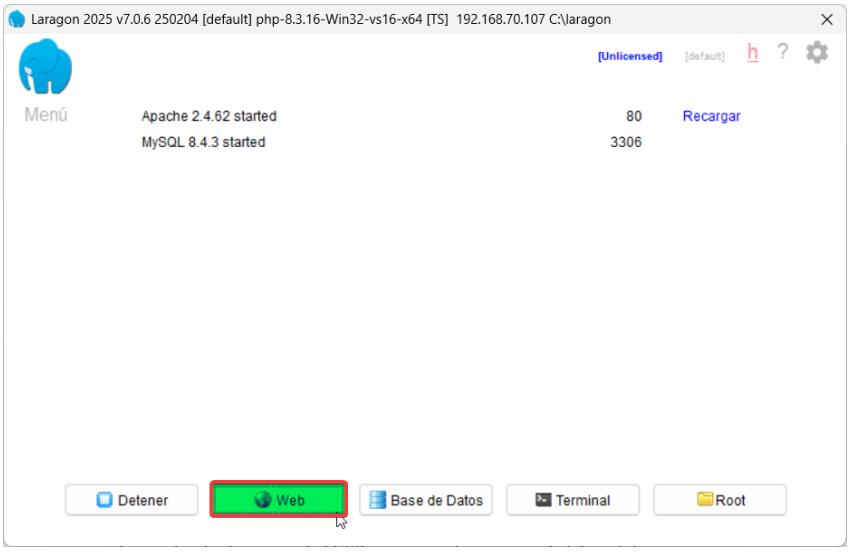


***Nota**: Proceso de inicialización de la consola de Laragon. **Fuente:** Autor*

La administración de las herramientas de Laragon mediante la consola permite tener una mejor perspectiva para gestionar los recursos necesarios y realizar las configuraciones pertinentes mediante la consola con una interfaz amigable e intuitiva.

Figura 29

Inicio de la Página Principal de Laragon y del Proyecto



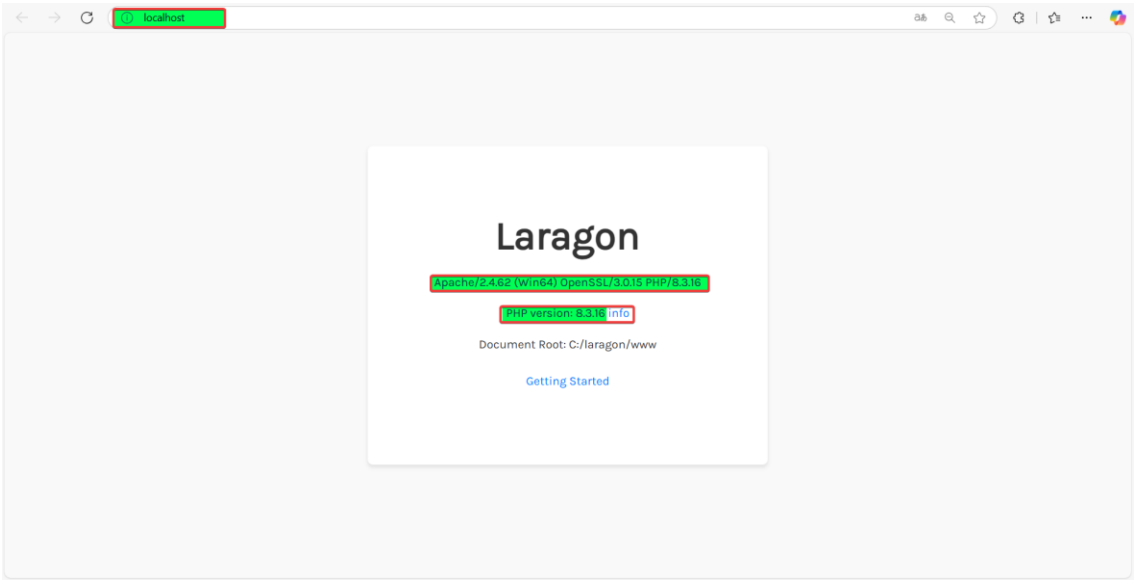
***Nota:** Proceso de inicialización de la consola de Laragon. **Fuente**: Autor*

Aquí una vez levantado el servicio de Apache podremos iniciar la pagina web o el sitio del proyecto desde el botón “Web” que se encuentra resaltado en la Figura 29.

En el mismo una vez que se dé clic a este botón podremos divisar en el navegador que se encuentre seleccionado como principal o navegador por defecto la presentación de la página principal index.php que viene por defecto una vez realizado esto podremos continuar con el tema de la configuración en si así lo deseamos podemos revisarla versión de PHP que Laragon ofrece, incluyendo la versión completa y la versión portable.

Figura 30

Laragon en ejecución.



Nota: *Proceso de inicialización de la consola de Laragon.* ***Fuente:*** *Autor*

La versión completa se instala en el sistema, mientras que la versión portable se puede ejecutar desde una unidad USB o una carpeta sin necesidad de instalación. Laragon tiene constantes actualizaciones por lo cual es recomendado siempre tener la última versión estable, también hay que tomar en cuenta que en la versión actual que es la v7.0.6 los creadores han tomado la iniciativa de utilizar el registro de una licencia para fomentar la ayuda y colaboración sobre esta herramienta.

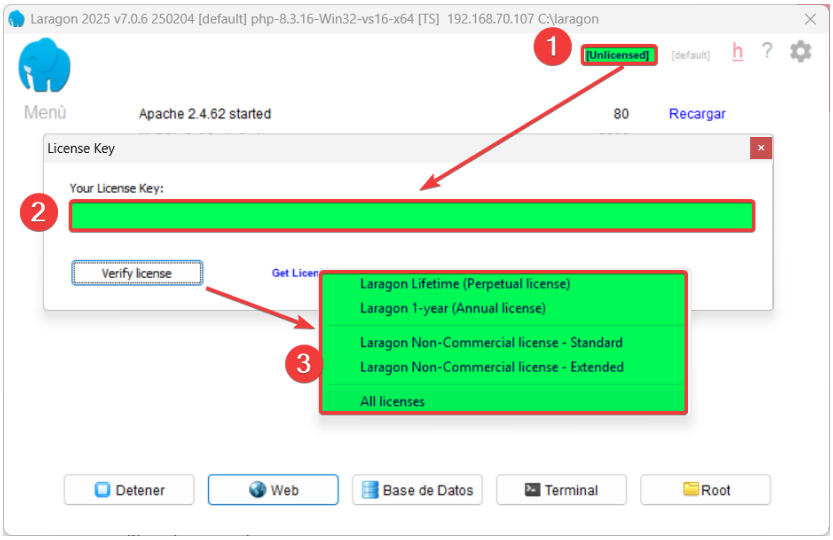
Hay que tener en cuenta que para poder trabajar con Laragon no es necesario comprar como tal la experiencia completa de la licencia para que no cause problemas podremos quedar registrados de forma gratuita, teniendo presente que siempre se va a presentar un pop-up que pregunte por la licencia al momento de editar o si topamos el hipervínculo que se encuentra en la parte superior de la herramienta en la opción 1 que se encuentra señalado en la Figura 31 del documento.

De igual forma solicitara el apoyo monetario para continuar con las actualizaciones, pero dado el caso y factor nuestra licencia se mantendrá en gratuita a menos que ustedes decidan así cambiarlo ya que cuenta con grandes opciones mismas que se visualizan en la opción 3 de la Figura 31.

Esta elección no afectara el desempeño y la forma de trabajo en Laragon y PHP ya que estas herramientas se presentan por defecto para el trabajo y desarrollo de aplicaciones cuando Laragon este activo.

Figura 31

Licencia de Laragon



Nota: *Proceso de inicialización de la consola de Laragon.* ***Fuente****: Autor*

6.2.3 ¿Qué trae Laragon?

Tabla 2

Tecnologías de Laragon

Categoría	Componente	Descripción
Servidor Web	Apache / Nginx	Seleccionable por el usuario. Permite ejecutar aplicaciones web localmente.
Base de Datos	MySQL / MariaDB	Seleccionable por el usuario. Almacena y gestiona datos para las aplicaciones.
Lenguajes de Programación	PHP	Lenguaje principal para desarrollo web.
	Node.js	Entorno de ejecución para JavaScript del lado del servidor.

	Python	Lenguaje de propósito general utilizado en desarrollo web y scripting.
	Ruby	Lenguaje dinámico utilizado en desarrollo web.
	Go	Lenguaje compilado utilizado en desarrollo de aplicaciones de alto rendimiento.
Herramientas	phpMyAdmin	Interfaz web para la administración de bases de datos MySQL/MariaDB.
	HeidiSQL	Cliente gráfico para la administración de bases de datos MySQL/MariaDB.
	Composer	Gestor de dependencias para PHP.
	Git	Sistema de control de versiones para el seguimiento de cambios en el código.
Gestión	Hosts Virtuales	Permite crear y administrar fácilmente hosts virtuales para diferentes proyectos.

***Nota**: Tabla que indica las tecnologías que trabajan en Laragon **Fuente**: Autor*

Aquí podemos revisar un resumen de las tecnologías que maneja la herramienta que hemos acabado de instalar, tanto la proporción de lenguajes de programación que maneja, así como bases de datos entre otros.

6.2.4 Laragon: Ventajas para el desarrollo de PHP y Laravel

Configuración rápida: Esta herramienta simplifica la configuración de un entorno de desarrollo PHP y Laravel, ahorrando tiempo y esfuerzo.

Aislamiento de proyectos: Permite crear entornos de desarrollo aislados para cada proyecto, evitando conflictos entre versiones de software.

Facilidad de uso: Su interfaz intuitiva y sus funcionalidades avanzadas facilitan el desarrollo y la depuración de aplicaciones PHP y Laravel.

Rendimiento: Laragon está optimizado para el rendimiento, lo que permite un desarrollo más rápido y eficiente.

Portabilidad: La versión portable de Laragon permite llevar el entorno de desarrollo a cualquier computadora.

Multiples versiones de PHP: Laragon permite tener diferentes versiones de PHP instaladas y cambiar entre ellas de forma sencilla.

Integración con Composer y Git: Laragon viene con Composer y Git preinstalados, lo que facilita la gestión de dependencias y el control de versiones.

En conclusión, Laragon es una excelente opción para los desarrolladores de PHP y Laravel que buscan un entorno de desarrollo local rápido, fácil de usar y configurar que sea potente y que se adapte a las necesidades cambiantes del Windows. (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023)

6.3 Gestor de dependencias Composer

Composer es un gestor de dependencias para PHP. En términos simples, es una herramienta que te permite declarar las bibliotecas (dependencias) que el proyecto necesita y las instala por ti. Pensemos que estas construyendo una casa: Composer sería el encargado de traer todos los materiales (ladrillos, cemento, etc.) que necesitas para construirla.

6.3.1 ¿Para qué sirve Composer en Laravel?

Dentro de Laravel, Composer juega un papel fundamental debido a que se encarga de administrar ciertas bibliotecas necesarias para la implementación del proyecto las cuales enumeraremos a continuación:

La gestión de dependencias dentro de Laravel permite administrar numerosas bibliotecas y paquetes que van a dar paso al correcto funcionamiento de proyecto. Composer se encargará de instalar y gestionar estas dependencias, asegurando que todas las versiones sean compatibles con la versión de Laravel que se está utilizando.

Composer es la forma recomendada de instalar Laravel. El comando composer create-project laravel/laravel nombre-del-proyecto descarga e instalara Laravel y todas sus dependencias. Luego composer te permite administrar paquetes adicionales para ampliar la funcionalidad de tu aplicación Laravel. Por ejemplo, puedes instalar paquetes para autenticación, APIs, pruebas, entre otros. (Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, 2023)

De igual forma la actualización de dependencias se puede gestionar mediante Composer ya que facilita la obtención de últimas versiones compatibles de las dependencias para tu proyecto.

6.3.2 Instalación de Composer

La instalación de Composer varía según el sistema operativo en el que nos encontremos trabajando, se procede a detallar los 3 sistemas operativas más utilizados y la instalación de la

herramienta, pero la instalación con figuras e imágenes se realizara solo en Windows 11 por tema tiempo:

Windows: Descarga el instalador de Composer desde el sitio web oficial: getcomposer.org. este puede ser el ".exe" o ". setup"

Linux: Descarga el instalador de Composer desde el sitio web oficial: getcomposer.org. este puede ser el ".elf" o ".tar.gz"

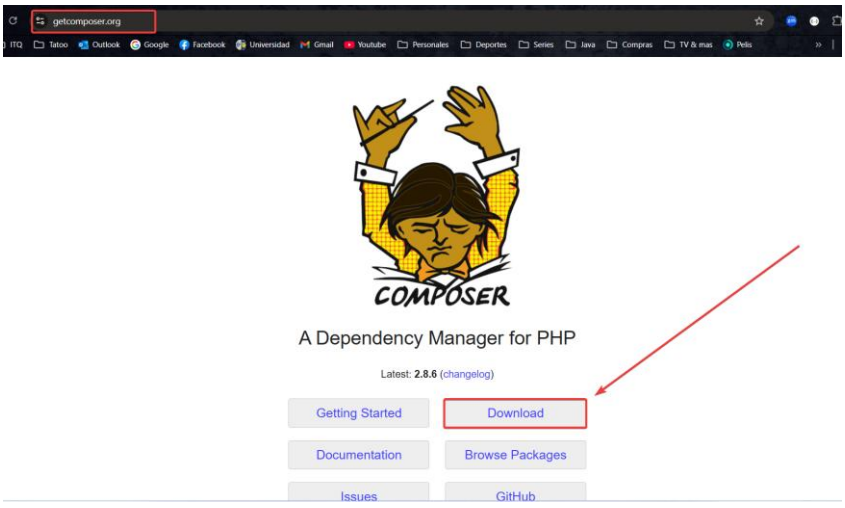
MacOs: Descarga el instalador de Composer desde el sitio web oficial: getcomposer.org. este puede ser el "AIX" o ". Exe"

Independiente del punto de ejecución el archivo está habilitado para instalarse por comandos que esto también beneficia en gran medida a usuarios del sistema operativo GNU / Linux para el manejo de paquetes mediante sudo pero de ser el caso si no manejas las actualizaciones por consola es preferible que se puedan generar las instalaciones por archivos preestablecidos por el fabricante del empaquetado de librerías. (Allen, Battle Ready Laravel, 2022)

A continuación, se procede con la instalación de este administrador de librerías en Windows para ello debemos visitar su sitio oficial para obtener el ejecutable acorde a nuestro sistema operativo, en este caso será Windows 11.

Figura 32

Composer descarga de versión

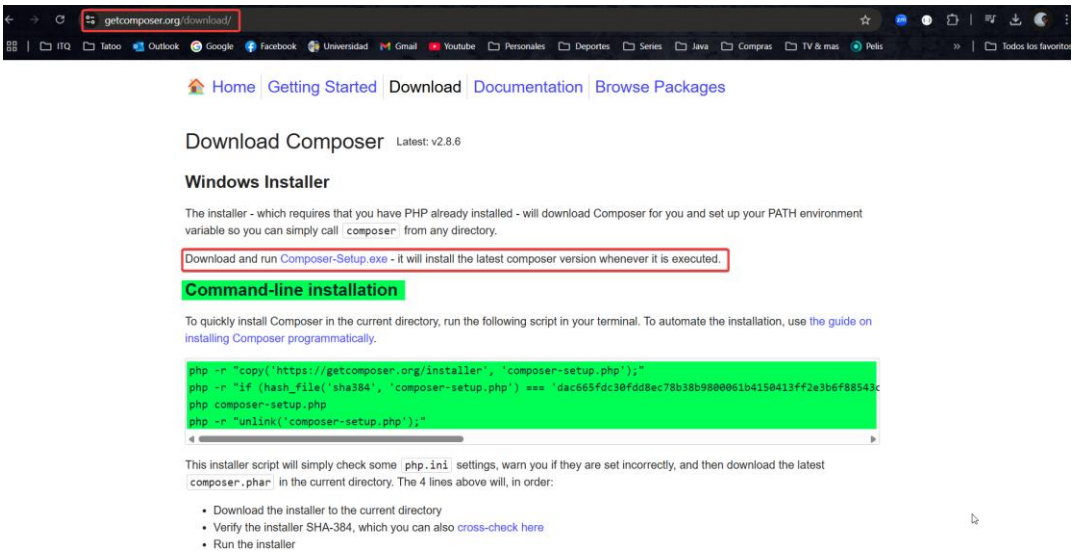


Nota: *Pagina web oficial de Composer.* ***Fuente:*** *Autor*

Descarga el instalador correcto para la versión de Composer estable o LTS y seguir las instrucciones del instalador como se muestra en las siguientes imágenes.

Figura 33

Proceso de selección de la versión LTS de Composer.

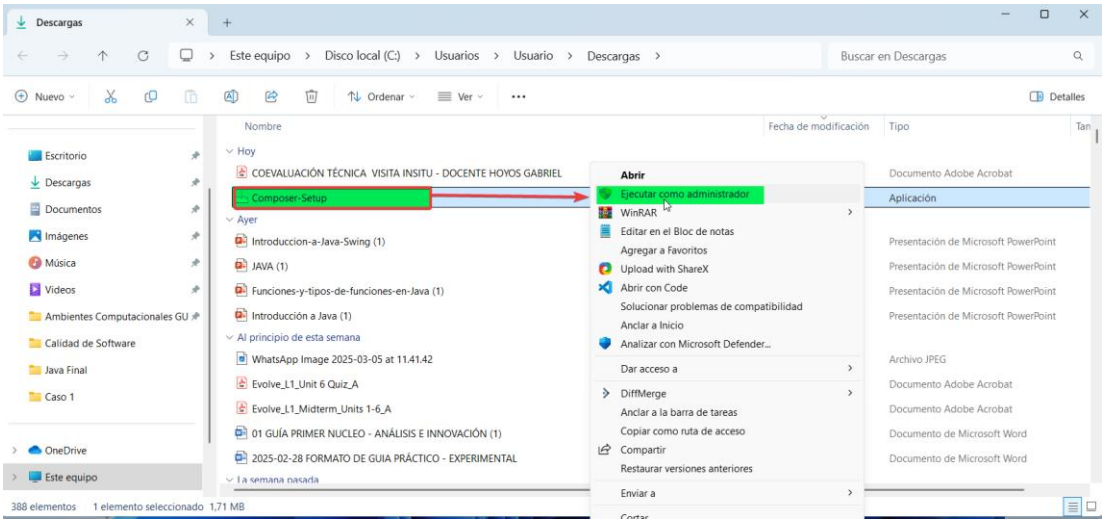


***Nota**: Descarga de Composer LTS. **Fuente**: Autor*

Tener en cuenta que la instalación se debe realizar antes que laravel por ser un prerrequisito para la creación de librerías o dependencias de PHP para empezar el proyecto

Figura 34

Instalación de Composer desde descargas.



***Nota:** Descarga de Composer LTS. **Fuente:** Autor*

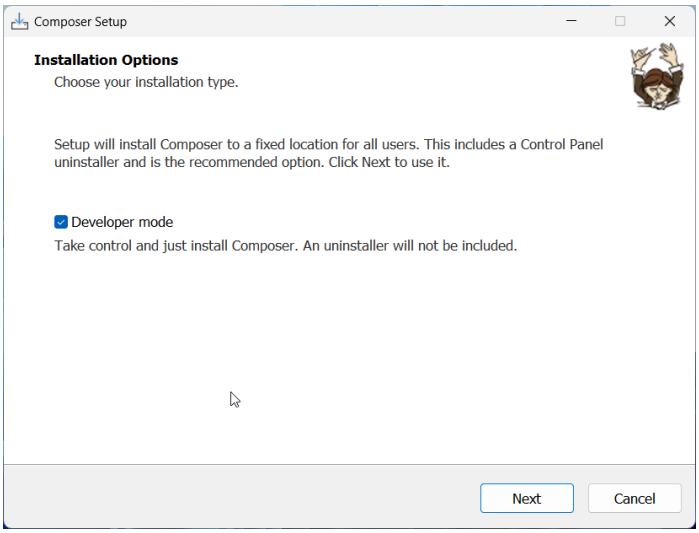
En este apartado una vez descargada la versión LTS (Latest Time Support) esto significa LTS son las siglas de Long Term Support (Platzi , 2020) o soporte a largo plazo en español. Es un término informático que se refiere a versiones de software que tienen soporte durante más

tiempo que las versiones estándar por ende es la versión que va a recibir más actualizaciones y es la más estable. Posterior a generar esta instalación como administrador podremos ver cómo se va creando las carpetas de Composer (Carrión, Noriega, & Castillo, 2019).

A continuación, comenzaremos con la instalación de composer seleccionando el modo de desarrollador para que puedas tener el control de toda la instalación y de sus librerías además de las configuraciones adicionales que permite generar un ambiente dentro del proceso general para la creación de un proyecto en Laravel, este es un complemento necesario para la generación de estos proyectos y el cumplimiento de la estructura principal del desarrollo de Laragon y Laravel como tal, permitiéndonos manipular cada aspecto necesario durante y después de la creación de los proyectos del framework de trabajo, así como sus elementos que se irán creando a medida que la necesidades lo requieran.

Figura 35

Instalación de Composer parte 1

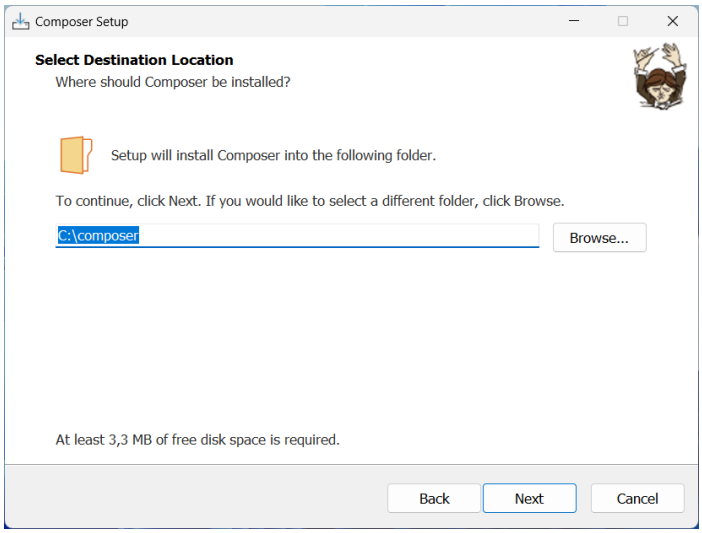


Nota: *Primera parte de la instalación de Composer.* ***Fuente:*** *Autor.*

En este apartado debemos seleccionar el modo de desarrollo para continuar con la configuración e instalación y desarrollo de proyectos con Laragon y Laravel.

Figura 36

Instalación de Composer parte 2

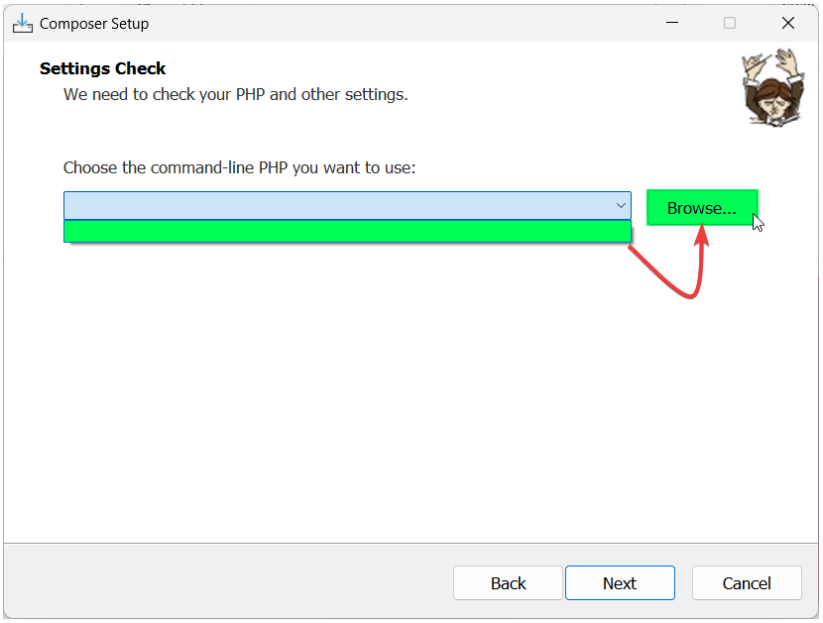


Nota: *Segunda parte de la instalación de Composer.* ***Fuente:*** *Autor.*

En esta parte de la instalación seleccionaremos la ubicación de las librerías que deben estar en lo posible en la misma ruta que la instalación de Laragon.

Figura 37

Instalación de Composer parte 3.1



Nota: *Tercera parte de la instalación de Composer.* ***Fuente:*** *Autor*

En el dinámico mundo del desarrollo PHP, la gestión eficiente de dependencias es la piedra angular para una correcta construcción de aplicaciones robustas y escalables. Composer

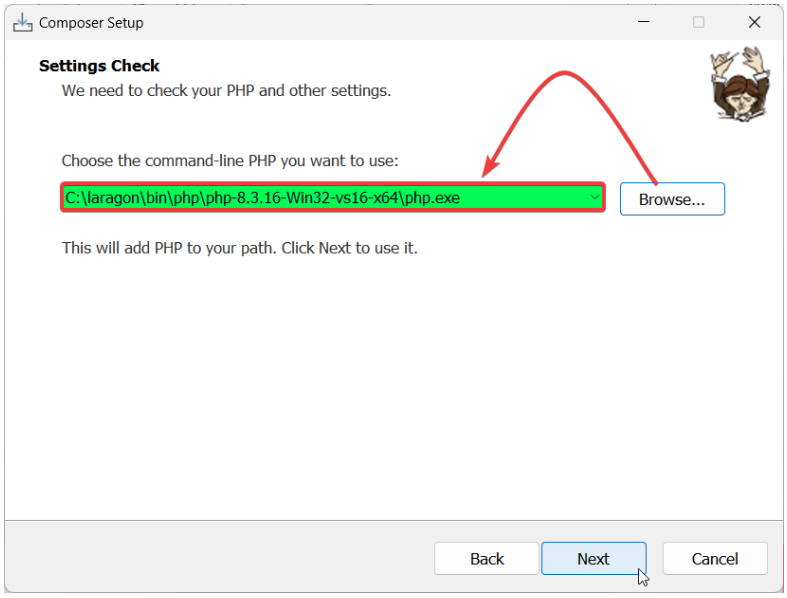
surge como aquella herramienta indispensable, revolucionando la forma en que los desarrolladores abordan la integración de bibliotecas y paquetes. Ya que la capacidad para automatizar la instalación, actualización y configuración de dependencias se simplifica significativamente durante el flujo de trabajo, permitiendo a los desarrolladores concentrarse en la lógica de sus aplicaciones. (Quijano, 2021)

En esta parte de la instalación Composer requiere que se ubique la ruta de instalación de PHP y que versión se esta utilizando es por eso que se debe definir la ruta de instalación en la misma ruta que Laragon, que es la herramienta que proveerá la extensión de línea de comandos de PHP como podemos revisar en la parte de selección no se encuentra configurada ninguna ruta por lo que debemos utilizar el botón de búsqueda y dirigirnos a través del explorador de Windows a la ruta correcta de selección del ejecutable de php mismo que proporcionará las funcionalidades necesarias para la instalación de las librerías de Composer (Aguirre, 2021).

Por lo que es importante el manejo correcto de esta tecnología, de tal forma que en la Figura 38 podremos visualizar la definición de ruta de la correcta para la instalación del gestor de dependencias de composer y php (Carrión, Noriega, & Castillo, 2019).

Figura 38

Instalación de Composer parte 3.2

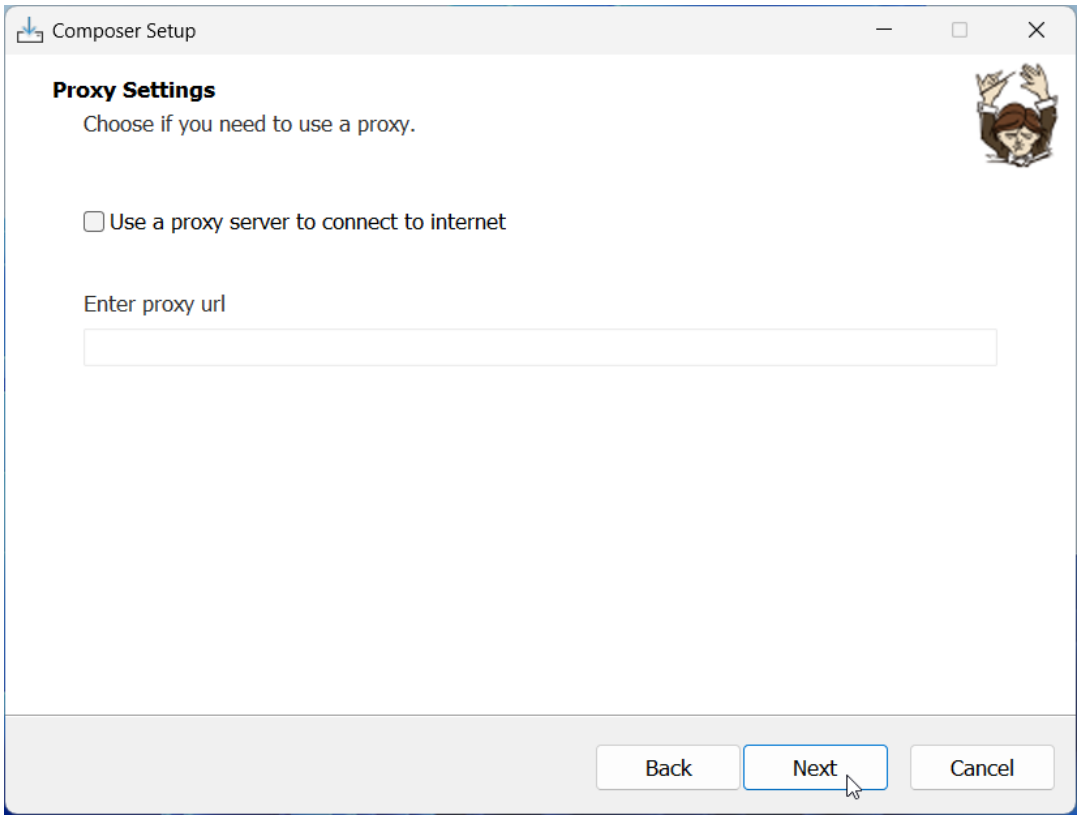


***Nota**: Tercera parte de la instalación de Composer. **Fuente**: Autor*

En este apartado ya tenemos definida la ruta del ejecutable de PHP a su vez podemos notar que estamos trabajando con la versión 8.3.16 que es una de las más actuales en cuanto a las funcionalidades agregadas al lenguaje de programación.

Figura 39

Instalación de Composer parte 4

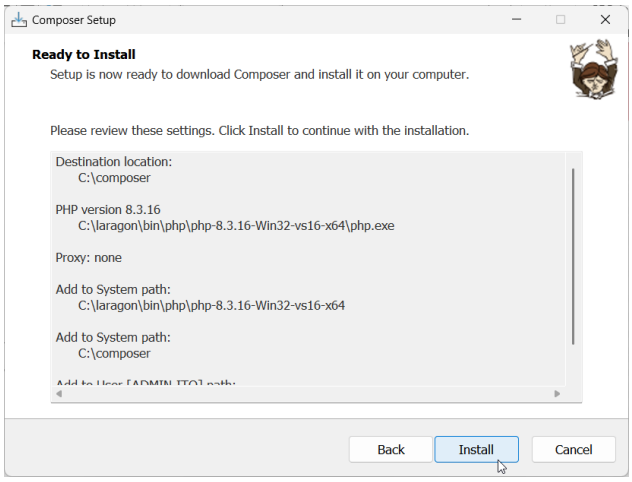


***Nota**: Cuarta parte de la instalación de Composer. **Fuente**: Autor*

En este caso no es necesario utilizar un proxy dado que manejaremos la herramienta de forma local y por ende el ambiente de desarrollo debe garantizar que la herramienta rinda acorde a lo esperado, por lo tanto, daremos clic en siguiente y continuaremos con la instalación de Composer. (Allen, Battle Ready Laravel, 2022).

Figura 40

Instalación de Composer parte 5



***Nota**: Quinta parte de la instalación de Composer resumen de componentes a instalar. **Fuente**: Autor*

Figura 41

Finalización de instalación de Composer

***Nota**: Final de la instalación de Composer. **Fuente**: Autor*

En la Figura 40 tenemos un resumen de la instalación de Composer con cada uno de sus componentes a instalar y la versión de PHP que va utilizar para la creación de los proyectos.

De igual forma la instalación de Composer toma unos segundos en realizarse una vez definida la ruta de Laragon y PHP por lo cual debemos esperar poco, a su vez se realiza un mensaje preventivo sobre el reinicio del equipo para que las nuevas instalaciones y

configuraciones se realicen correctamente, por último, daremos clic en el botón de finalizar y el launcher de instalación se cerrar completando así exitosamente la instalación del orquestador de librerías de PHP.

Mientras que macOS y Linux se puede realizar la instalación de Composer mediante la terminal utilizando comandos como curl o wget a su vez se recomienda consultar la documentación oficial de Composer para obtener instrucciones detalladas.

6.4 Uso básico de Composer en Laravel

composer install: Instala las dependencias definidas en el archivo composer.json.

composer update: Actualiza las dependencias a las últimas versiones compatibles.

composer require nombre-del-paquete: Instala un paquete específico.

composer remove nombre-del-paquete: Elimina un paquete.

6.5 Beneficios de utilizar Composer

Tabla 3

Beneficios y características de Composer en Laravel

Característica/Beneficio	Descripción	Uso en Laravel
Automatización	Automatiza la instalación y gestión de dependencias.	Laravel depende de numerosas bibliotecas (paquetes). Composer automatiza la instalación y actualización de estas dependencias, ahorrando tiempo y evitando errores manuales.
Consistencia	Asegura que todos los miembros del equipo utilicen las mismas versiones de las dependencias.	El archivo composer.lock generado por Composer garantiza que todos los miembros del equipo utilicen las mismas versiones exactas de las dependencias, evitando problemas de compatibilidad.
Facilidad de uso	Simplifica la instalación y actualización de paquetes.	Los comandos composer require, composer update y composer install simplifican la

		gestión de paquetes en proyectos Laravel.
Ecosistema	Permite acceder a una amplia gama de paquetes y bibliotecas.	Packagist, el repositorio principal de Composer, ofrece una vasta colección de paquetes que pueden ser utilizados en proyectos Laravel para agregar funcionalidades como autenticación, gestión de bases de datos, APIs, etc.
Autocarga de Clases	Genera un archivo de autocarga (autoload.php) que permite a PHP cargar automáticamente las clases de las dependencias instaladas.	Laravel utiliza el auto cargador de Composer para cargar automáticamente las clases de sus componentes y las dependencias instaladas, mejorando el rendimiento y la organización del código.
Gestión de Paquetes de Laravel	Permite instalar y gestionar paquetes específicos de Laravel, como Laravel Passport (autenticación API), Laravel Sanctum (autenticación SPA), Laravel Telescope (depuración), etc.	Laravel utiliza Composer para distribuir e instalar sus propios paquetes oficiales y contribuciones de la comunidad.
Scripting y Automatización	Permite definir scripts personalizados en el archivo composer.json para automatizar tareas comunes en el desarrollo de Laravel, como la ejecución de pruebas, la generación de documentación, etc.	Laravel utiliza scripts de Composer para ejecutar comandos de Artisan, como php artisan migrate, php artisan serve, etc.
Control de versiones	Permite especificar las versiones de las dependencias,	El archivo composer.json permite definir las versiones de

	asegurando la compatibilidad y estabilidad del proyecto.	las dependencias, asegurando la compatibilidad y estabilidad del proyecto.
Optimización del rendimiento	Permite generar un archivo de autocarga optimizado para producción, mejorando el rendimiento de la aplicación Laravel.	El comando composer dump-autoload --optimize genera un archivo de autocarga optimizado para producción, mejorando el rendimiento de la aplicación Laravel.

Información adicional relevante:

composer.json: Este archivo es el corazón de Composer. Define las dependencias del proyecto, las versiones requeridas y los scripts personalizados.

composer.lock: Este archivo registra las versiones exactas de las dependencias instaladas, asegurando la consistencia en todos los entornos.

Packagist: Es el repositorio principal de paquetes Composer, donde se encuentran la mayoría de las bibliotecas y paquetes disponibles para PHP.

Artisan: La interfaz de línea de comandos de Laravel se integra perfectamente con Composer, permitiendo ejecutar comandos relacionados con la gestión de paquetes y dependencias.

***Nota**: Tabla que presenta un resumen general de los beneficios de Composer. **Fuente**: Autor*

Brevemente hemos revisado los beneficios que el orquestador de Laravel proporciona, Composer es un instrumento notable que facilita el desarrollo de aplicaciones Laravel ya que proporciona la gestión de dependencias necesaria dentro de cada proyecto, lo que permite centrarte en la lógica de aplicación y generación de funcionalidad principal por ello es primordial para el avance correcto del proyecto ahorrándonos tiempo de configuración.

7 Artisan: la línea de comandos de Laravel (CLI)

Artisan se basa en comandos es decir que es la CLI de comandos de Laravel y cada que el usuario digita un comando en específico que realizara una tarea específica, como crear un controlador, generar una migración de base de datos o ejecutar una tarea programada. Los comandos se ejecutan desde la terminal utilizando el comando: “php artisan”.

7.1 Componentes importantes de Laravel:

Rutas: Definen cómo la aplicación responde a las solicitudes HTTP y HTTPS.

Controladores: Manejan la lógica de la aplicación y coordinan las interacciones entre el modelo y la vista proveyendo a este framework de una estructura MVC para poder trabajar con cada componente.

Modelos: Simbolizan las tablas de la base de datos y proporcionan métodos para interactuar con los datos.

Vistas: Definen las interfaces del usuario dentro de la aplicación, generalmente administradas a través de Blade de php que existe en laravel y que se hizo referencia en el capítulo 1 de este documento.

Migraciones: Permiten gestionar los cambios en la estructura de la base de datos.

Seeders: Permiten rellenar la base de datos con datos de prueba.

Middleware: Permiten filtrar las solicitudes HTTP que llegan a la aplicación.

Proveedores de servicios: Permiten registrar servicios y componentes en el contenedor de servicios de Laravel.

Eventos y listeners: Permiten implementar un sistema de eventos y listeners para desacoplar la lógica de la aplicación.

7.2 Artisan: Aplicado a la Creación de componentes

Es la CLI incluida en el framework PHP Laravel. Proporciona una serie de comandos que facilitan el desarrollo de aplicaciones Laravel. Esta CLI por sus siglas "Command-Line Interface" (Interfaz de Línea de Comandos). En programación, una CLI es una forma de interactuar con un programa o sistema operativo mediante comandos de texto en lugar de una interfaz gráfica de usuario (GUI). Artisan nos facilita la creación de componentes de Laravel utilizando comandos específicos. Por ejemplo:

php artisan make:controller NombreControlador -> Crea un nuevo controlador.

php artisan make:model NombreModelo -> Crea un nuevo modelo.

php artisan make:migration crear_tabla_nombres -> Crea una nueva migración.

php artisan make:seeder NombreSeeder -> Crea un nuevo seeder.

php artisan make:middleware NombreMiddleware -> Crea un nuevo middleware.

Cada uno de estos comandos nos permite la administración de proyecto mediante laravel, por lo tanto cada uno realizar una función específica de lo que le usuario solicito y cada uno se encargara de dar una funcionalidad distinta y única para acceder a los datos de la aplicación en la que se priorizara la creación de elementos que migran datos desde una base y que interactuaran con los otros elementos a través de un modelo y un controlador, si bien esto 2 son importantes sin una adecuada creación del modelo y la migración no se podría trabajar de forma eficiente con este framework de trabajo.

De igual menara Artisan no solo administrara la creación de los componentes o elementos que se utilizan a lo largo del tiempo en desarrollo si no también cuenta con comandos que brindan facilidades para la creación de proyectos en Laravel facilitando la vida de los desarrolladores y la creación de nuevos proyectos Laravel. El comando composer create-project laravel/laravel nombre-proyecto crea un nuevo proyecto Laravel con la estructura de directorios y archivos necesarios. (Allen, Battle Ready Laravel, 2022)

En si todo esto se indicara y explicara en el próximo capítulo de este documento por ello adjuntamos a continuación ejemplos que permitan generar esta administración como por ejemplo levantar la aplicación que se creara en el próximo capitulo, esto se puede tomar como una pequeña introducción a todos los procesos que debemos realizar para la generación de proyectos en Laravel (Aguirre, 2021).

Ejemplos de comandos Artisan:

php artisan serve: Inicia el servidor de desarrollo de Laravel.

php artisan migrate: Ejecuta las migraciones de base de datos.

php artisan db:seed: Ejecuta los seeders de base de datos.

php artisan route:list: Muestra la lista de rutas definidas en la aplicación.

php artisan cache:clear: Limpia la caché de la aplicación.

7.3 Beneficios de Artisan

Tabla 4

Beneficios de Artisan en Laravel

Beneficio	Descripción	Detalles Adicionales

Automatización de tareas	Artisan automatiza tareas comunes de desarrollo, ahorrando tiempo y esfuerzo.	▪ Generación de código MCM (controladores, modelos, migraciones). ▪ Ejecución de migraciones de bases de datos. ▪ Limpieza de caché. ▪ Creación de usuarios y roles ▪ Programación de tareas (comandos cron).
Consistencia	Asegura que los componentes de la aplicación se creen de forma consistente, siguiendo las mejores prácticas de Laravel.	▪ Estructura de directorios y archivos estandarizada. ▪ Uso de plantillas predefinidas para la generación de código. ▪ Reducción de errores humanos.
Productividad	Facilita el desarrollo de aplicaciones Laravel, permitiendo a los desarrolladores concentrarse en la lógica de la aplicación.	▪ Comandos concisos y fáciles de recordar. ▪ Interfaz de línea de comandos intuitiva. ▪ Rapidez en la creación de prototipos y nuevas funcionalidades.
Gestión de bases de datos	Simplifica la gestión de bases de datos a través de migraciones y seeds.	▪ Control de versiones de la estructura de la base de datos. ▪ Población de bases de datos con datos de prueba.

		▪ Facilita la colaboración en equipos de desarrollo.
Programación de tareas	Permite programar tareas repetitivas para que se ejecuten automáticamente.	▪ Envío de correos electrónicos. ▪ Limpieza de registros. ▪ Procesamiento de datos en segundo plano.
Pruebas	Facilita la ejecución de pruebas unitarias y de integración.	▪ Generación de stubs de pruebas. ▪ Ejecución de suites de pruebas. ▪ Análisis de cobertura de código.
Desarrollo de paquetes	Ayuda en la creación y gestión de paquetes Laravel.	▪ Generación de la estructura de paquetes. Publicación de activos. Ejecución de pruebas de paquetes.

***Nota**: Tabla que presenta un resumen general de los beneficios de Artisan. **Fuente**: Autor*

7.4 Resumen del Capítulo 2

En el segundo capítulo de este libro hemos visto la importancia de la instalación de los prerrequisitos para los proyectos Laravel, lo cual nos permitió tener un equipo restaurado para la creación de los mismos además hemos revisado las principales estructuras, primero hemos instalado Laragon uno de los temas principales para que Laravel pueda funcionar correctamente para optimizar su funcionamiento también hemos instalado los paquetes dependientes del proceso como lo es el orquestador Composer el cual permite interactuar con la terminal e instalar las dependencias para la creación de los proyectos de diseño , creación y administración de aplicaciones multiplataforma, hemos detallado paso a paso la instalación de Laragon y Composer, una vez realizada la instalación de ambas herramientas se recomienda la instalación de un editor de texto lo cual nos va a permitir generar la instalación de las dependencias en nuestro ambiente de trabajo, se presenta paso a paso de la instalación de los prerrequisitos de Laragon, ya que esto permitirá ver cómo trabaja Laravel y como su código fuente sirve para

administrar de mejor forma casi como un lenguaje nativo. Básicamente, esto permitirá que el lenguaje de programación se haga cargo de parte del trabajo del desarrollador gracias a los comandos básicos que permiten crear un mejor ambiente. Posterior a esto hemos realizado un código de ejemplo de cada estructura que maneja Laravel.

De tal forma se puede considerar lo siguiente en cuanto a las estructuras que maneja Laravel los comandos de Artisan y Blade son mecanismos de paso de datos que permiten administrar los elementos en una forma más sencilla y práctica lo que hace a laravel tan atractivo para la creación de páginas web. Mientras que el resto de frameworks de PHP no pueden administrar y gestionar la creación de estos elementos de forma práctica. Hay que recordar que existen otros comandos básicos para la creación de elementos secundarios, php artisan make permitirá gestionar la creación de elementos como Controladores, Modelos y Migraciones de base de datos (este último puede detallarse con un adecuado diseño de la base, pero dependerá de su complejidad y utilidad para generar las migraciones adecuadas) y @layouts es una herramienta para gestionar el estado en componentes funcionales. La elección entre los distintos frameworks de trabajo dependen de si la administración de los datos proporcionados desde fuera del marco de trabajo cumple con las expectativas de operación y si son gestionados de manera oportuna por lo cual Laravel y Composer serán los elegidos para construir aplicaciones web más eficientes y mantenibles.

8 Capítulo 3

Programando en Laravel una revisión de su proceso general.

En Laravel es necesario crear un proyecto mediante comandos básicos gracias a la instalación que realizamos en el capítulo anterior podremos definir un modelo de trabajo mediante comandos que brinda una experiencia de configuración sencilla los mismo que son parte del proceso de elaboración de la web que deseamos realizar, ahora bien, este proyecto se va a crear mediante comandos que interpretara composer y que trabajan en un lenguaje de configuración ya establecido y conocido como lo es PHP todos estos conceptos y manejo de nuestro framework lo veremos a continuación.

8.1 Normas y Estructura Fundamentales en el Desarrollo con Laravel

Laravel, un framework de código abierto que ha revolucionado la forma en que construimos proyectos web. Su enfoque basado en generación de configuraciones ha establecido estándares de eficiencia y mantenibilidad en el desarrollo backend. (Aguirre, 2021)

8.1.1 Arquitectura Modelo-Vista-Controlador (MVC):

Laravel plantea el uso del patrón de diseño MVC, que separa la lógica de la aplicación en tres componentes principales:

Modelo: Representa los datos y la lógica de negocio.

Vista: Define la interfaz de usuario.

Controlador: Maneja las interacciones entre el modelo y la vista.

Esta segmentación nos facilita la organización del código, la reutilización de componentes y el mantenimiento en general de la aplicación, ya que al mantener separados este marco de trabajo podremos definir responsabilidades sobre el equipo de desarrollo y mantener una distribución acorde a los estándares de laravel.

8.1.2 Estructura de Directorios

Laravel tiene una estructura de directorios predefinida que organiza los archivos de la aplicación de manera lógica. Los directorios más importantes se presentarán en el resumen de la tabla 5, la cual tiene como principal función indicar cuales son los procesos que se realizan dentro del lenguaje, así como una explicación a detalle de cada carpeta y su contenido.

Tabla 5

Detalle de carpetas relevantes en Laravel.

Directorio	Propósito Fundamental	Descripción Detallada (Perspectiva de un Experto)
app/	Lógica Central de la Aplicación	Este es el corazón de tu aplicación Laravel. Aquí residen los modelos (representación de tus datos), controladores (manejo de las solicitudes HTTP), middleware (filtrado de solicitudes), y otros componentes que definen el comportamiento de tu aplicación. Es donde la lógica de negocio toma forma.
routes/	Definición de Rutas	Este directorio es el mapa de tu aplicación. Aquí defines las rutas que conectan las URLs con las acciones de tus controladores. Las rutas son esenciales para el enrutamiento de solicitudes y la creación de APIs.
resources/views/	Almacenamiento de Vistas	Aquí se encuentran las plantillas Blade, el motor de plantillas de Laravel, que generan la interfaz de usuario de tu aplicación. Separar la lógica de presentación de la lógica de negocio es una práctica esencial, y este directorio lo facilita.
database/	Migraciones y Seeders de la Base de Datos	Este directorio es crucial para la gestión de la base de datos dado que las migraciones permiten controlar las versiones del esquema de la base de datos, y los seeders permiten poblar la base de datos con datos de prueba o datos iniciales.

public/	Archivos Públicos	Este directorio contiene los archivos que son accesibles públicamente, como CSS, JavaScript, imágenes y otros activos. Es el punto de entrada para todas las solicitudes web.
config/	Archivos de Configuración	Aquí se encuentran los archivos de configuración que controlan el comportamiento de tu aplicación Laravel. Desde la configuración de la base de datos hasta los servicios de terceros, este directorio es esencial para personalizar tu aplicación.

***Nota**: Tabla que presenta un resumen general de las carpetas importantes de Laravel. **Fuente**: Autor.*

Es decir que la separación de responsabilidades entre estos directorios es una de las características de Laravel, que promueve la organización y la mantenibilidad del código. Además, el uso de migraciones y seeders garantiza que la estructura de la base de datos sea coherente en todos los entornos de desarrollo.

La seguridad de la aplicación inicia en el correcto manejo de los archivos dentro del directorio public, ya que por lo general todos y cada uno de estos archivos se encuentra accesibles desde cualquier parte de la aplicación. A su vez el directorio config, permite una gran flexibilidad a la hora de personalizar cada parte del framework en nuestro beneficio.

8.1.3 Rutas:

Las rutas definen cómo nuestra aplicación responderá a las solicitudes de tipo HTTP. Laravel permite definir rutas utilizando diferentes métodos HTTP (GET, POST, PUT, DELETE, etc.). Es decir que las rutas son el núcleo de cada aplicación Laravel. En si podemos decir que el directorio routes/ actúa como el centro de control para todas las solicitudes HTTP entrantes. Aquí, definimos cómo nuestra aplicación responde a diversas URLs y métodos HTTP (GET, POST, PUT, DELETE, etc.). (Aguirre, 2021)

Archivos de Rutas Clave

Dentro de routes/, encontrarás varios archivos, cada uno con un propósito específico:

web.php:

Este es el archivo principal para definir rutas que generan interfaces web.

Las rutas aquí están protegidas por el middleware web, que proporciona características como sesiones y protección CSRF.

Aquí es donde la mayor parte de la lógica de enrutamiento de tu aplicación web residirá.

api.php:

Diseñado para definir rutas de API.

Las rutas aquí están protegidas por el middleware api, que normalmente maneja la autenticación basada en tokens.

Este archivo es esencial para construir APIs RESTful o GraphQL.

console.php:

Define rutas para comandos de la consola Artisan.

Si necesitas crear comandos personalizados para tareas administrativas, aquí es donde los registrarás.

channels.php:

Define canales para Laravel Broadcast.

Esto es usado para eventos en tiempo real.

Definiendo Rutas: La Sintaxis

Laravel proporciona una sintaxis elegante y expresiva para definir rutas:

Métodos HTTP:

Utilizamos Route::get(), Route::post(), Route::put(), Route::delete(), etc., para especificar el método HTTP que la ruta manejará.

URLs:

Definimos la URL que la ruta coincidirá.

Acciones:

Especificamos qué acción se ejecutará cuando la ruta coincida. Esto puede ser:

Una función de cierre (anónima).

Un método de controlador.

Ejemplos Prácticos

Aquí hay algunos ejemplos para ilustrar cómo se definen las rutas:

Ruta Básica GET:

Figura 42

Ruta Básica GET,

```php
Route::get('/', function () {
    return view('welcome');
});
```

***Nota**: Ejemplo GET de rutas en Laravel. **Fuente**: Autor.*

Aquí podemos divisar como se maneja la ruta básica con la URL y el método GET donde en el retorno seleccionamos el Blade al cual queremos que sea redirigido.

Ruta POST a un controlador:

Figura 43:

Ruta POST guiada a un controlador.

```php
Route::post('/usuarios', [UsuarioController::class, 'store']);
```

***Nota**: Ejemplo POST de rutas en Laravel. **Fuente**: Autor.*

Aquí podemos divisar como se maneja la ruta básica con la URL y el método POST donde en el retorno seleccionamos el controller al cual queremos que haga referencia y de igual forma se envían el nombre del controlador como un parámetro dentro de la URL.

Figura 44

Rutas con Parámetros

```php
Route::get('/usuarios/{id}', [UsuarioController::class, 'show']);
```

***Nota**: Ejemplo POST de rutas en Laravel con parametros. **Fuente**: Autor*

En esta ruta podemos observar cómo se envían parámetros a través de la URL para que se pueda gestionar o mostrar un usuario por su id en específico de la misma manera se debería realizar cuando hacemos una actualización.

8.1.4 Controladores

Los controladores manejan la lógica de la aplicación y coordinan las interacciones entre el modelo y la vista, es decir maneja y crea cada una de las funciones y métodos principales de cada proceso y por ende de su raciocinio.

Los controladores son clases PHP que contienen métodos o funciones que responden a cada una de las solicitudes HTTP. Los controladores se almacenan en el directorio app/Http/Controllers/ en esta ruta se almacenarán dependiendo la funcionalidad a la que hagan referencia y defiendan cada uno de sus aspectos para el servicio de nuestra aplicación web.

En la arquitectura Modelo-Vista-Controlador (MVC) de Laravel, los controladores son clases PHP que contienen métodos llamados "acciones". Es decir, estos métodos manejan las solicitudes HTTP (como cuando un usuario visita una página o envía un formulario) y determinan qué respuesta se debe enviar. Por lo tanto, su principal función es separar la lógica de tu aplicación de las rutas y las vistas, manteniendo tu código organizado y fácil de mantener.

Funciones principales de los controladores:

Recibir solicitudes: Los controladores reciben las solicitudes HTTP enviadas por los usuarios, es decir que cuando un usuario interactúa con tu aplicación (por ejemplo, visita una página, envía un formulario, o alguna adyacente a esta), se genera una solicitud HTTP y el controlador recibe esta solicitud para ser procesada.

Procesar la lógica: Contienen la lógica de tu aplicación, como interactuar con la base de datos entre otros por lo tanto este controlador contiene la lógica necesaria para manejar la solicitud. Esto puede incluir interactuar con la base de datos, validar datos, realizar cálculos, etc.

Coordinar interacciones: Actúan como intermediarios entre los modelos (que representan los datos) y las vistas (que muestran la interfaz de usuario). El controlador en este caso tan particular actúa como intermediario entre los modelos (que representan los datos) y las vistas (Blade de PHP).

Enviar respuestas: Los controladores envían respuestas HTTP, que pueden ser vistas HTML, datos JSON u otras formas de respuesta. Una vez que el controlador ha procesado la solicitud, envía una respuesta HTTP al usuario. Esta respuesta puede ser una vista HTML, datos JSON, una redirección, etc.

Ejemplo:

Figura 45

Controlador en Laravel.

PHP

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function showProfile($id)
    {
        $user = User::findOrFail($id);
        return view('user.profile', ['user' => $user]);
    }

    public function updateProfile(Request $request, $id)
    {
        $user = User::findOrFail($id);
        $user->name = $request->input('name');
        $user->email = $request->input('email');
        $user->save();
        return redirect()->route('user.profile', ['id' => $id]);
    }
}
```

***Nota**: Ejemplo de controlador con 2 funciones. **Fuente**: Autor.*

En el siguiente ejemplo, el UserController maneja las solicitudes relacionadas con los usuarios. Es decir que el método showProfile() muestra el perfil de un usuario específico. El método updateProfile() actualiza el perfil de un usuario.

En estos casos las rutas también permiten direccionar el proceso acorde a lo necesitado como se evidencia en el ejemplo de la Figura 45.

Figura 46

Controlador en Laravel 2

````
2.  **Controlador de productos:**

```php
    <?php

    namespace App\Http\Controllers;

    use App\Models\Product;
    use Illuminate\Http\Request;

    class ProductController extends Controller
    {
        public function index()
        {
            $products = Product::all();
            return view('product.index', ['products' => $products]);
        }


        public function show($id)
        {
            $product = Product::findOrFail($id);
            return view('product.show', ['product' => $product]);
        }

        // ... otros métodos
    }
    ```
````

***Nota**: Ejemplo de controlador con 2 funciones. **Fuente**: Autor*

El ProductController maneja las solicitudes relacionadas con los productos. El método index() muestra la lista de productos. El método show() por otro parte muestra los detalles de un producto específico.

El enrutamiento en Laravel es extremadamente flexible, lo que permite crear aplicaciones web y APIs complejas con facilidad además podemos deducir que el uso de controladores ayuda a mantener el código de enrutamiento limpio y organizado.

Los parámetros de ruta y las expresiones regulares proporcionan un control preciso sobre la coincidencia de rutas. Es muy importante entender el uso de los middlewares, para poder proteger las rutas de la manera correcta.

8.1.5 Modelos

En Laravel, los modelos son clases PHP que representan las tablas de tu base de datos. Cada modelo corresponde a una tabla, y sus atributos representan las columnas de esa tabla. Los modelos proporcionan una interfaz intuitiva para interactuar con los datos, permitiéndote realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera sencilla.

Laravel utiliza Eloquent ORM (Object-Relational Mapping) para simplificar la interacción con la base de datos. Un ORM es una técnica que mapea las tablas de la base de datos a objetos PHP, permitiéndote trabajar con los datos utilizando una sintaxis orientada a objetos en lugar de SQL directo. Los modelos se almacenan en el directorio app/Models/. (CodersFree, 2024)

Para la revisión de ejemplos dirigirse al capítulo 1 donde se indicó como trabaja Eloquent ORM el cual simplifica enormemente la interacción con la base de datos, permitiendo a los desarrolladores concentrarse en la lógica de la aplicación en lugar de escribir SQL complejo. Es decir que los modelos de Laravel proporcionan una capa de abstracción que hace que el código sea más legible, mantenible y escalable. Por ende es crucial comprender las relaciones entre modelos (uno a uno, uno a muchos, muchos a muchos) para construir aplicaciones robustas y eficientes.

8.1.6 Vistas:

Las vistas dentro del framework son el espacio ideal para crear los diseños HTML de la página propio de las estructuras del framework pero que es el directorio de vistas en Laravel, se puede considerar como la presentación segmentada y visible de tu Aplicación. (Allen, The Clean Coder's Guide to Laravel, 2021)

Las vistas en Laravel son archivos que contienen el código HTML que se muestra al usuario en el navegador. En otras palabras, definen la interfaz de usuario de tu aplicación. Laravel utiliza Blade, un motor de plantillas potente y flexible, para facilitar la creación de vistas dinámicas mismo que fue tratado en el capítulo 1 de este documento.

Blade el motor de platillas de Laravel te permite usar sintaxis especial dentro de tus archivos HTML para incluir lógica, variables y estructuras de control. Esto hace que sea más fácil crear vistas dinámicas y reutilizables.

8.1.7 Pasos para crear archivos Blade:

Las vistas se almacenan en el directorio resources/views/.

Los archivos de vista tienen la extensión.blade.php que es la extensión requerida para manejar vistas dentro de laravel.

Por ejemplo, puedes crear un archivo llamado bienvenida.blade.php para la página de inicio de tu aplicación.

8.1.8 Manejo de la sintaxis Blade:

Imprimir variables: Debemos usar la siguiente expresión {{ $variable }} para mostrar el valor de una variable.

Estructuras de control: Usa @if, @else, @elseif, @foreach, @while, etc., para agregar lógica a tus vistas.

Comentarios: Usa {{-- Comentario --}} para agregar comentarios que no se mostrarán en el navegador.

Plantillas: Usa @extends y @section para crear plantillas reutilizables.

Pasar datos a las vistas: Puedes pasar datos a tus vistas desde tus controladores usando el método view ().

Por ejemplo:

Figura 47

Paso de parámetros con el Método View.

```php
public function mostrarBienvenida() {
    $nombre = 'Usuario';
    return view('bienvenida', ['nombre' => $nombre]);
}
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

Para mostrar las vistas desde los controladores el método view () toma el nombre del archivo de vista (sin la extensión.blade.php) como primer argumento esto permite organizar tus vistas en subdirectorios para mantener tu código limpio y ordenado.

Utiliza plantillas para evitar la repetición de código y mantener una estructura coherente en tu aplicación. Aprovecha las directivas de Blade para simplificar la lógica en tus vistas además considerar el uso de componentes Blade para crear elementos de interfaz de usuario reutilizables permite tener una aplicación adaptable y mantenible (Stauffer, Laravel: Up & Running, 2019).

El uso de Blade y sus plantillas garantizara la reutilización de código y volviendo a la creación de plantillas algo parametrizable y reutilizable desde el punto de vista técnico ganamos tiempo en el desarrollo

Ejemplos Prácticos

Figura 48

Elementos de vista Blade.php

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Bienvenido</title>
</head>
<body>
    <h1>¡Hola, {{ $nombre }}!</h1>
</body>
</html>
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

Figura 49

Vista con estructura de control (lista_usuarios.blade.php)

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Usuarios</title>
</head>
<body>
    <h1>Usuarios:</h1>
    <ul>
        @foreach ($usuarios as $usuario)
            <li>{{ $usuario->nombre }}</li>
        @endforeach
    </ul>
</body>
</html>
```

***Nota**: Ejemplo de manejo de estructuras de una vista. **Fuente**: Autor*

En las figuras anteriores podemos revisar el proceso de creación de platillas en laravel a su vez se indica como se utilizan las estructuras de control y como se pueden indicar las variables dentro de nuestras platillas.

Figura 50

Uso de plantillas (layout.blade.php)

```HTML
<!DOCTYPE html>
<html>
<head>
    <title>@yield('titulo')</title>
</head>
<body>
    @yield('contenido')
</body>
</html>
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

En la Figura 50 podemos ver cómo se utilizan las plantillas de Blade en este caso la palabra reservada “yield” es una herramienta fundamental para crear plantillas reutilizables y gestionar el contenido dinámico de tus vistas. En esta sección explico como el uso de su función y cómo se utiliza:

@yield actúa como un marcador de posición dentro de una plantilla Blade. Su propósito es definir un área donde el contenido puede ser inyectado desde vistas hijas. En esencia, permite crear una plantilla base (layout) con secciones predefinidas que pueden ser llenadas con contenido específico en cada página.

Figura 51

Representación de uso de Extends y revisión de Sección en Laravel.

```HTML
@extends('layout')

@section('titulo', 'Página de Inicio')

@section('contenido')
    <h1>¡Bienvenido a la página de inicio!</h1>
    <p>Contenido de la página de inicio.</p>
@endsection
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

Tanto @extends establece la relación de herencia entre las vistas y @section define el contenido dinámico que se inyecta en las secciones de la plantilla base el uso de estas directivas

promueven la reutilización de código, la organización y la mantenibilidad en tus aplicaciones Laravel.

8.1.9 Migraciones:

Las migraciones permiten gestionar los cambios en la estructura de la base de datos. En lugar de modificar manualmente la base de datos, las migraciones te permiten definir los cambios en código, lo que facilita el control de versiones, la colaboración en equipo y el despliegue de aplicaciones cuando realizas cambios directos en la base de datos es preferible utilizar las migraciones de laravel las cuales se almacenan en el directorio database/migrations/ (Quijano, 2021).

Creación de Migraciones: Utilizas el comando de Artisan “php artisan make:migration nombre_de_la_migracion” para crear un nuevo archivo de migración. A su vez Laravel genera un archivo PHP en el directorio database/migrations/ con una estructura básica.

Definición de Cambios: En cada archivo de migración contiene dos métodos: up() y down() donde el método up() define los cambios que se deben aplicar a la base de datos (por ejemplo, crear tablas, agregar columnas, modificar índices). Mientras que el método down() define los cambios inversos que se deben aplicar para revertir la migración (por ejemplo, eliminar tablas, eliminar columnas).

Ejecución de Migraciones: Utilizas el comando de Artisan “php artisan migrate” para ejecutar todas las migraciones pendientes. Por ende, Laravel ejecuta los métodos up() de cada migración en orden cronológico (Rincón, 2024).

Reversión de Migraciones: Utilizas el comando de Artisan “php artisan migrate:rollback” para revertir la última migración ejecutada. Mientras Laravel ejecuta el método down() de la última migración para revertir los cambios, así mismo puedes reversar varias migraciones utilizando el parámetro --step.

Control de Versiones: Las migraciones se almacenan en el repositorio de código, lo que permite realizar un seguimiento de los cambios en la estructura de la base de datos y esto a su vez facilita la colaboración en equipo y el despliegue de aplicaciones en diferentes entornos.

Por lo tanto, es importante manejar esta herramienta tanto en la creación de tablas y su administración para la gestión de bases de datos en Laravel. Permiten definir los cambios en la estructura de la base de datos en forma de programación orientada a objetos esto facilita el control de versiones, la colaboración en equipo y el despliegue de aplicaciones evitando la

necesidad de modificar manualmente la base de datos, lo que reduce el riesgo de errores durante el proceso.

Figura 52

Uso de comando para la creación de la migración de Laravel.

Bash

```
php artisan make:migration crear_tabla_usuarios
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

Figura 53

Creación de migración con código PHP

PHP

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CrearTablaUsuarios extends Migration
{
    public function up()
    {
        Schema::create('usuarios', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->string('email')->unique();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('usuarios');
    }
}
```

***Nota**: Ejemplo de envió de parámetros a una vista. **Fuente**: Autor*

Figura 54

Ejecución de la migración de Laravel

```bash
php artisan migrate
```

***Nota**: Ejemplo de ejecución de Código para migraciones. **Fuente**: Autor*

8.1.10 Seeders

Los seeders permiten rellenar la base de datos con datos de prueba es decir son clases PHP que contienen métodos que insertan datos en la base de datos. Los seeders se almacenan en el directorio database/seeders/. (Quijano, 2021)

Estas clases de PHP te permiten insertar datos en tus tablas de base de datos. Se utilizan comúnmente para:

Datos de prueba: Rellenar la base de datos con datos ficticios para probar tu aplicación.

Datos iniciales: Insertar datos predeterminados que la aplicación necesita para funcionar correctamente (por ejemplo, roles de usuario, categorías de productos).

Creación de Seeders:

Utilizas el comando de Artisan “php artisan make:seeder NombreDelSeeder” para crear un nuevo archivo de seeder, de preferencia renombrarlo con algo referente al nombre de la tabla que va a poblar datos por lo que Laravel generara un archivo PHP en el directorio database/seeders/ con una estructura básica.

Al definir los datos en cada archivo de seeder contiene un método run() donde defines los datos que se insertarán en la base de datos. Puedes utilizar Eloquent ORM para interactuar con tus modelos y realizar inserciones de datos.

Al ejecutar los seeders se utilizara el comando de Artisan “php artisan db:seed” para ejecutar todos los seeders registrados en el archivo DatabaseSeeder.php, mismo que se crea con anterioridad. Caber recalcar que también puedes ejecutar un seeder específico utilizando el parámetro --class, por ejemplo: “php artisan db:seed --class=NombreDelSeeder”. (Allen, The Clean Coder's Guide to Laravel, 2021)

A continuación, revisaremos un ejemplo práctico que detallan cada uno de los pasos

Figura 55

Ejecución de comando para la creación de seeders

Bash

```
php artisan make:seeder UsuariosTableSeeder
```

***Nota**: Ejemplo de comando para crear seeder. **Fuente**: Autor*

Figura 56

Seeder para poblar datos de la tabla Usuarios

PHP

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

class UsuariosTableSeeder extends Seeder
{
    public function run()
    {
        DB::table('usuarios')->insert([
            [
                'nombre' => 'John Doe',
                'email' => 'john.doe@example.com',
                'password' => Hash::make('password'),
            ],
            [
                'nombre' => 'Jane Smith',
                'email' => 'jane.smith@example.com',
                'password' => Hash::make('password'),
            ],
        ]);
    }
}
```

***Nota**: Estructura de Seeder para la población de datos en la tabla Usuarios. **Fuente**: Autor*

Figura 57

Registro Seeder en el archivo Seeders de PHP Laravel

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        $this->call([
            UsuariosTableSeeder::class,
        ]);
    }
}
```

***Nota**: Estructura de Seeder para la población de datos en la tabla Usuarios. **Fuente**: Autor*

8.1.11 Middleware:

El middleware permite filtrar las solicitudes HTTP que llegan a la aplicación. El middleware se utiliza para realizar tareas como autenticación, autorización y registro se almacena en el directorio app/Http/Middleware/. (Aguirre, 2021)

En Laravel, el middleware actúa como una capa intermedia entre la solicitud HTTP entrante y la respuesta final. Es un mecanismo que te permite filtrar y modificar las solicitudes antes de que lleguen a tu controlador o ruta.

El middleware se utiliza para realizar una variedad de tareas, como:

Autenticación: Verificar si un usuario está autenticado antes de permitirle acceder a ciertas rutas.

Autorización: Verificar si un usuario tiene los permisos necesarios para realizar una acción específica.

Registro de actividad (logging): Registrar las solicitudes entrantes y salientes para fines de auditoría o depuración.

Manipulación de datos: Modificar los datos de la solicitud o la respuesta antes de que lleguen al controlador o se envíen al navegador.

Protección contra ataques: Prevenir ataques como la falsificación de solicitudes entre sitios (CSRF).

La creación de Middleware: se utiliza el comando de Artisan “php artisan make:middleware NombreDelMiddleware” para crear un nuevo archivo de middleware por lo tanto Laravel genera un archivo PHP en el directorio app/Http/Middleware/ con una estructura básica (Aguirre, 2021).

La definición de lógica de cada archivo de middleware contiene un método handle() donde defines la lógica que se ejecutará para cada solicitud donde el método handle() recibe la solicitud HTTP y una función $next como argumentos (Quijano, 2021).

La función $next se utiliza para pasar la solicitud al siguiente middleware en la cadena o al controlador/ruta final para ello debes registrar el middleware en el archivo app/Http/Kernel.php para que Laravel lo reconozca. (Allen, The Clean Coder's Guide to Laravel, 2021)

Puedes registrar middleware globalmente (se ejecuta para todas las solicitudes) o asignarlos a rutas específicas. El proceso de ejecución de middleware es cuando una solicitud HTTP llega a tu aplicación, Laravel ejecuta los middlewares registrados en el orden en que están definidos.

Si un middleware llama a $next($request), la solicitud pasa al siguiente middleware o al controlador/ruta. Si un middleware no llama a $next($request), la solicitud se detiene y se envía una respuesta al cliente.

Figura 58

Creación de Middleware en Laravel

```Bash
    php artisan make:middleware AutenticacionMiddleware
```

***Nota**: Comando para crear un middleware. **Fuente**: Autor*

Figura 59

Creación de Middleware para Login.

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AutenticacionMiddleware
{
    public function handle(Request $request, Closure $next)
    {
        if (!Auth::check()) {
            return redirect('/login');
        }

        return $next($request);
    }
}
```

***Nota**: Comando para crear un middleware. **Fuente**: Autor*

Figura 60

Registro del middleware en app/Http/Kernel.php

```php
// En el array $routeMiddleware:
'auth' => \App\Http\Middleware\AutenticacionMiddleware::class,
```

***Nota**: Comando para registrar un middleware. **Fuente**: Autor*

8.1.12 Proveedores de servicios:

Los proveedores de servicios permiten registrar servicios y componentes en el contenedor de servicios de Laravel estos a su vez se utilizan para configurar la aplicación y registrar bibliotecas de terceros estos se almacenan en el directorio app/Providers/.

Es decir, los servicios providers son clases PHP que se encargan de registrar y configurar los servicios que tu aplicación necesita. Un "servicio" en Laravel puede ser cualquier cosa, desde

componentes del framework (como el enrutador o el sistema de plantillas) hasta servicios personalizados que creas para tu aplicación. (Allen, The Clean Coder's Guide to Laravel, 2021)

Cada servicio provider tiene un método register() donde se registran los servicios en el contenedor de servicios de Laravel a su vez el contenedor de servicios es un componente que gestiona las dependencias de tu aplicación, permitiendo que los objetos se instancien y se inyecten automáticamente cuando se necesitan (Benabderrezak, 2025).

Los servicios providers también tienen un método boot() donde se realiza la configuración adicional y se arrancan los servicios registrados donde boot() se ejecutara después de que todos los servicios providers han sido registrados. Los servicios providers se registran en el archivo config/app.php en el array providers. Donde Laravel cargara los servicio providers registrados durante el arranque de la aplicación. (Allen, The Clean Coder's Guide to Laravel, 2021)

Por Ejemplo:

Figura 61

Creación de Service Provider para Cargar API Externa.

Bash

```
php artisan make:provider ApiServiceProvider
```

***Nota**: Comando para crear un servicio provider. **Fuente**: Autor*

En este ejemplo supongamos que necesitamos crear un servicio provider para registrar un servicio personalizado que maneja la conexión a una API externa, para obtener datos y conexiones de la API Externa deberá configurarse y registrarse un Service Provider que permite obtener esta conexión.

A continuación, se indica como realizar la configuración del provider que receptara los datos de la API externa, esto desde Laravel.

Figura 62

Service Provider que ejecuta API Externa.

PHP

```php
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use App\Services\ApiService;

class ApiServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->singleton(ApiService::class, function ($app) {
            return new ApiService(config('services.api.url'));
        });
    }

    public function boot()
    {
        // Configuración adicional si es necesario
    }
}
```

***Nota**: Comandos para obtener datos de una API Externa en un Service Provider. **Fuente**: Autor*

Figura 63

Registro del Service Provider.

PHP

```php
'providers' => [
    // ...
    App\Providers\ApiServiceProvider::class,
],
```

***Nota**: Registro de un Service Provider en config/app.php para su ejecución. **Fuente**: Autor*

Para finalizar tenemos la ejecución y registro del servicio creado donde es primordial realizar para que el mismo pueda ejecutarse cuando la aplicación sea lanzada o ejecutada. Los

Service providers son notables estructuras para la comunicación de la arquitectura de Laravel, permitiendo obtener modularidad y extensibilidad.

El contenedor de servicios facilita la gestión de dependencias y la inyección de objeto permitiéndonos configurar y arrancar servicios personalizados para tu aplicación.

8.1.13 Eventos y listeners:

Dentro de Laravel, los eventos y listeners son una implementación del patrón de diseño "Observador" muy similar a como SQL maneja sus Triggers o Disparadores que para su efecto se ejecutan siempre que se realiza una opción en específico así que de igual forma en Laravel nos permiten que diferentes partes la aplicación se comuniquen entre sí de forma desacoplada.

Eventos: Son objetos PHP que representan "algo que ha sucedido" en la aplicación. Por ejemplo, "Usuario registrado", "Pedido realizado", "Tarea completada".

Listeners: Son clases PHP que "escuchan" los eventos y ejecutan acciones específicas en respuesta a ellos. Por ejemplo, enviar un correo electrónico de bienvenida después de un "Usuario registrado" (Griffin, 2020).

Para poder inicializar un evento necesitamos el comando de Artisan “php artisan make:event NombreDelEvento” para que este pueda crear un nuevo archivo de evento donde Laravel genera un archivo PHP en el directorio app/Events/. El evento puede contener datos relevantes sobre lo que ha sucedido en una acción especifica dentro del sistema. De igual forma necesitamos inicializar con un comando de Artisan “php artisan make:listener NombreDelListener --event=NombreDelEvento” para crear un nuevo archivo de listener donde Laravel genera un archivo PHP en el directorio app/Listeners/. Para ello el listener contendrá un método handle() que se ejecuta cuando se dispara el evento, muy similar a los eventos de Java pero en PHP. (Quijano, 2021)

Debes registrar los eventos y listeners en el archivo app/Providers/EventServiceProvider.php dentro del array $listen, se procede a definir qué listeners deben ejecutarse para cada evento utilizando la función event() para que al disparar un evento desde cualquier parte de la aplicación el evento pueda ejecutarse por ende Laravel se encargara de notificar a los listeners registrados para la correcta ejecución, a continuación un ejemplo de lo indicado.

Figura 64

Comando de Artisan para crear Evento.

```bash
php artisan make:event UsuarioRegistrado
```

***Nota**: Creación de un evento en Laravel. **Fuente**: Autor*

Figura 65

Generación de Evento (Constructor) parte 1

```php
<?php
namespace App\Events;
use Illuminate\Queue\SerializesModels;
class UsuarioRegistrado
{
    use SerializesModels;
    public $usuario;
    public function __construct($usuario)
    {
        $this->usuario = $usuario;
    }
}
```

***Nota**: Creación de un evento en Laravel. **Fuente**: Autor*

Figura 66

Creación de Función Handler para lanzar Evento

````
2. **Listener: EnviarCorreoBienvenida**
   * `php artisan make:listener EnviarCorreoBienvenida --event=UsuarioRegistrado`
   * `app/Listeners/EnviarCorreoBienvenida.php`:

```php
   <?php
   namespace App\Listeners;
   use App\Events\UsuarioRegistrado;
   use Illuminate\Support\Facades\Mail;
   class EnviarCorreoBienvenida
   {
       public function handle(UsuarioRegistrado $event)
       {
           Mail::raw('Bienvenido, ' . $event->usuario->nombre, function ($message) use ($event) {
               $message->to($event->usuario->email);
               $message->subject('Bienvenido a nuestra aplicación');
           });
       }
   }
   ```
````

***Nota**: Creación de función Handler en Laravel. **Fuente**: Autor*

Figura 67

Proceso de Registro y Ejecución del Evento

````
3. **Registro en `EventServiceProvider.php`:**

```php
    protected $listen = [
        UsuarioRegistrado::class => [
            EnviarCorreoBienvenida::class,
        ],
    ];
    ```

4. **Disparo del evento:**

```php
    event(new UsuarioRegistrado($usuario));
    ```
````

***Nota**: Creación de función Handler en Laravel. **Fuente**: Autor*

9 Resumen del Capítulo 3

En el Tercer y último capítulo de este libro hemos visto la importancia del desarrollo de aplicaciones en el framework Laravel , lo cual nos permitió realizar un trabajo en equipo resolviendo una serie de problemas que se presentan a lo largo del desarrollo en general para ello se instalaron las dependencias en nuestro ambiente de trabajo, se presenta paso a paso la creación de un proyecto y posterior la configuración paso a paso de nuestro entorno de Laravel desde la construcción de procesos bash hasta la generación de las herramientas particulares de Laravel, esto en base al proceso de desarrollo general de cualquier sistema :

ORM: Componente principal maneja BDD en este caso Eloquent, nos permitía administrar la gestión de tablas y cambios de esta desde PHP.

Blade: Componente individual que representa cada render de visualización el mismo que trabaja con HTML y PHP.

Eventos: Componente que muestra una acción especifica cuando un usuario realiza un proceso.

Rutas: Son como las direcciones de tu casa. Cuando alguien visita una URL, Laravel utiliza las rutas para decidir qué mostrar. Puedes definir rutas para diferentes acciones, como mostrar un perfil o un blog.

Controladores: Son la parte lógica de tu aplicación. Reciben las solicitudes y deciden qué hacer con ellas, como mostrar una vista o interactuar con un modelo.

Es básicamente lo que Laravel ofrece en el desarrollo de aplicaciones web. Este framework de PHP utiliza el patrón MVC (Modelo-Vista-Controlador), lo que significa que puedes separar la lógica de negocio de la interfaz de usuario, haciendo que tu código sea más organizado y fácil de mantener. Con Laravel, puedes aprovechar funcionalidades preprogramadas como autenticación, enrutamiento y bases de datos, lo que simplifica mucho el proceso de desarrollo. Esto se puede implementar en un proyecto de Laravel de la siguiente manera: Se utiliza la herramienta Laragon para crear la estructura del proyecto. Luego se utilizan los comandos de Artisan que permitirán la creación y manipulación de elementos de Laravel de forma autónoma.

Middleware: Se utilizan para segmentar y modular el proceso de genera información entre componentes de laravel, como la creación automática de un Login.

Service Providers: Se utilizan eventos para manejar las interacciones de la API externa.

Seeders: Se utilizan condicionales para poblar la base de datos con data de prueba que sea congruente.

BIBLIOGRAFÍA

Aguirre, S. (20 de 08 de 2021). *FRAMEWORK TOTAL - Vol.1: Crea APPs desde Cero con Laravel + Booststrap + MySQ.* Buenos Aires: ebooks. Obtenido de React Dev: https://es.react.dev/learn/start-a-new-react-project

Allen, A. (2021). *The Clean Coder's Guide to Laravel.* UK: Digital Ocean.

Allen, A. (01 de 01 de 2022). Battle Ready Laravel. En A. Allen, *Battle Ready Laravel.* UK. Obtenido de Medium - Single Page Applications (SPA): https://medium.com/@teamtechsis/single-page-applications-spa-48b1b845b446

armoneyandpolitics. (s.f.). *armoneyandpolitics*. Obtenido de https://armoneyandpolitics.com/taylor-otwell-little-rock/

Benabderrezak, Y. (2025). *Laravel Advanced Topics.* Berlín, Alemania: ResearchGate. doi:10.13140/RG.2.2.14659.82729

Carrión, R., Noriega, A., & Castillo, D. d. (2019). *Usando Xampp Bootstrap y Wordpress.* Rama Solutions. Obtenido de https://books.google.com.ec/books?id=pP-uDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Cobo, Á., Gómez, P., Pérez, D., & Rocha, R. (2005). *PHP y MySQL Tecnologías para el desarrollo de aplicaciones web.* Santander: Ediciones Díaz de Santos.

CodersFree. (28 de 08 de 2024). *Coders Free*. Obtenido de Coders Free: https://codersfree.com/posts/eloquent-orm-en-laravel-como-trabajar-con-el-de-manera-eficiente

Griffin, J. (2020). *Domain-Driven Laravel.* NY: Apress. doi:10.1007/978-1-4842-6340-2

hostinger. (s.f.). *hostinger*. Obtenido de https://www.hostinger.es/tutoriales/que-es-un-servidor-web

Lerdorf, R., & Tatroe, K. (2002). *Programming PHP* (Primera ed.). California: O'Reilly Media. doi:9781565926103

Murcia, U. d. (s.f.). *Universidad de Murcia*. Obtenido de https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-14-php-1.html

Pinterest. (s.f.). *Pinterest*. Obtenido de https://es.pinterest.com/

Platzi . (07 de 03 de 2020). *Platzi* . Obtenido de Platzi que es LTS ? : https://platzi.com/discusiones/1667-linux/82582-que-es-lts/

Quijano, J. L. (28 de 07 de 2021). *Laravel: Curso práctico de formación .* RC Libros. Obtenido de CodeMotion - Una completa introducción a la librería React: https://www.codemotion.com/magazine/es/lenguajes-de-programacion/una-completa-introduccion-a-la-libreria-react/#:~:text=React%20fue%20creada%20por%20un,complejidad%20y%20mejorando%20el%20rendimiento.

Rincón, J. J. (2024). *Aprenda desarrollo web con Laravel desde cero.* Marcombo. doi:978-8426739603

Sánchez Cano, G. (2018). *Programacion Backend: Con Xampp Tech Stack.* Alfaomega. doi:9786075381701

Soluciones digitales a medida. (08 de 10 de 2024). *10code*. Obtenido de 10code: https://10code.es/blade-laravel/

Spinola Federico, M. (03 de 03 de 2023). *AluraCursos - Qué es Single Page Application*. Obtenido de AluraCursos - Qué es Single Page Application: https://www.aluracursos.com/blog/single-page-application

Stauffer, M. (2019). *Laravel: Up & Running.* Sebastopol: O'Reilly Media. doi:978-1492041214

Stauffer, M. (2023). *Laravel: Up & Running: A Framework for Building Modern PHP Apps.* O Reilly Media. Obtenido de https://reactjs.org/

Yoris, A. C. (2022). *Primeros pasos con Laravel 12, domina el framework PHP más popular: Aquí continúa tu camino en el desarrollo de aplicaciones web en PHP con Laravel + Rest Api y Vue 3 Full Sack.* Mexico.