

Sesión de Aprendizaje N° 12-B

Desarrollo Web BackEnd

Stack: PHP - laravel

PHP / laravel

1. Servidor Web: Apache HTTP Server

El **Apache HTTP Server**, comúnmente conocido simplemente como "**Apache**", es el servidor web de código abierto más popular y utilizado en el mundo. Su función principal es entregar contenido web a los usuarios.

Imagina que cuando tú escribes una dirección web (URL) en tu navegador (Chrome, Firefox, etc.) y presionas Enter, estás enviando una "**solicitud**" a un servidor. Apache es el software que reside en ese servidor y se encarga de:

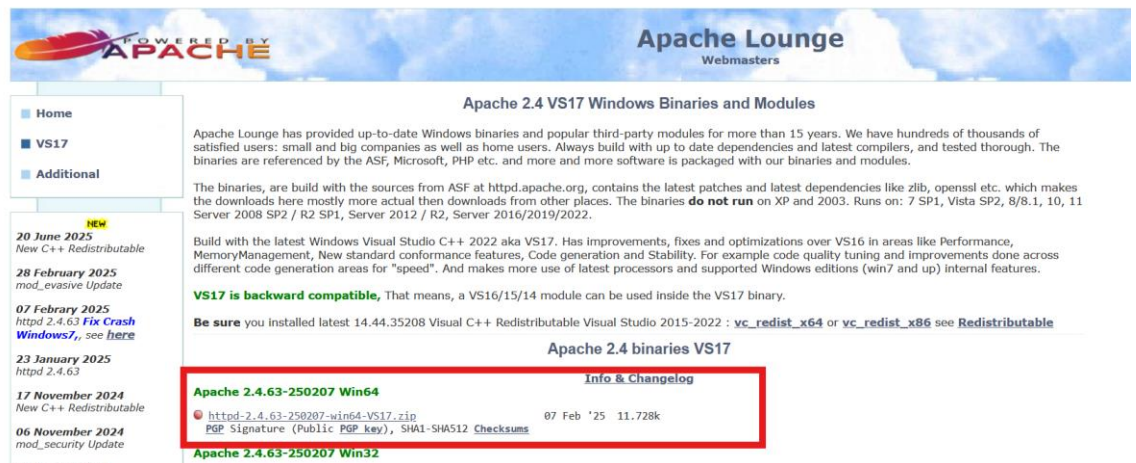
1. Recibir **request** HTTP/HTTPS del navegador.
2. Buscar el contenido solicitado HTML (y imágenes, videos, archivos PDF, etc.) en el servidor.
3. Enviar **response** (ese contenido) de vuelta al navegador del usuario.

Apache Lounge

Apache Lounge es un sitio web y una comunidad que se especializa en **proporcionar versiones compiladas** (binarios) del Apache HTTP Server, principalmente para sistemas operativos Windows.

Instalación del Servidor Portable

Descarga: <https://www.apachelounge.com/download/>



Puedes ejecutar Apache Lounge en Windows directamente desde la versión ZIP sin instalarlo como servicio.

1. Extrae el ZIP

Descomprime el archivo descargado (por ejemplo, httpd-2.4.xx-win64-VS17.zip) en una carpeta como D:\Server\Apache24.

2. Verifica dependencias

Asegúrate de tener instalado el paquete de Visual C++ Redistributable correspondiente (por ejemplo, VC15 o VC17). Si no, puedes descargarlo desde el sitio de Apache Lounge.

3. Configura el archivo httpd.conf

Abre D:\server\Apache24\conf\httpd.conf y revisa que tengas:

Listen 8080

ServerName localhost:8080

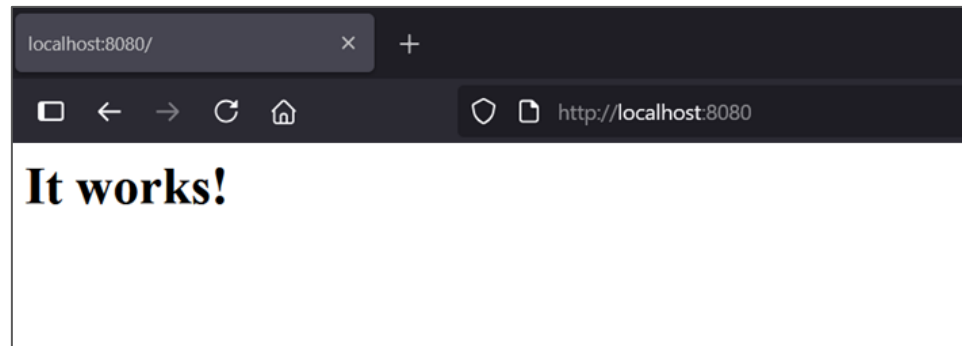
Define SRVROOT "D:\server\Apache24"

4. Ejecutar el Servidor httpd.exe

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.22631.5335]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\DevsServers\Apache24\bin>httpd.exe
```

5. Visualizar el Servidor



2. PHP - Hypertext Preprocessor

¿Qué es PHP?

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de programación de código abierto del lado del servidor, diseñado específicamente para el desarrollo web. Es uno de los lenguajes más utilizados en el mundo para construir sitios y aplicaciones web dinámicas.

1. Descarga el ZIP oficial

Descargas de PHP: <https://windows.php.net/download/> y elige la versión para tu arquitectura (x64 o x86). Asegúrate de descargar el archivo ZIP, no el instalador.

A screenshot of the official PHP website for Windows downloads. The page has a purple header with the PHP logo and 'PHP: Hypertext Preprocessor'. Below the header, there's a navigation bar with links: Home, Downloads, QA Releases, Snapshots, Team, and PHP.net site. The main content area is divided into several sections. On the left, there's a 'PHP For Windows' section with text about supporting PHP on Microsoft Windows and providing special builds. Below that is a 'PECL For Windows' section. Further down is a 'Which version do I choose?' section with sub-sections for 'IIS' and 'Apache'. On the right, there's a 'Binaries and sources Releases' section with a dropdown menu to select an option to direct access. Below this, there's a section for 'PHP 8.4 (8.4.8)' with links to download source code and tests package. The main part of the right section is for 'VS17 x64 Non Thread Safe (2025-Jun-03 17:39:44)' and lists three download options: 'Zip' (32.22MB), 'Debug Pack' (37.18MB), and 'Development package (SDK to develop PHP extensions)' (1.35MB), each with a SHA256 hash. At the bottom, there's a section for 'VS17 x64 Thread Safe (2025-Jun-03 17:40:02)'.

2. Extrae el contenido

Descomprime el archivo ZIP en una carpeta como D:\server\php.

3. Configura php.ini

Dentro de D:\server\php, copia php.ini-development y renómbralo como php.ini. Luego Abre el archivo php.ini con un editor de texto y configura así:

(habilita extensiones necesarias quitando el ; al inicio de líneas)

extension_dir = "ext"

extension=mysqli

extension=curl

4. Agrega PHP al PATH-

Ve a Configuración del sistema > Variables de entorno.

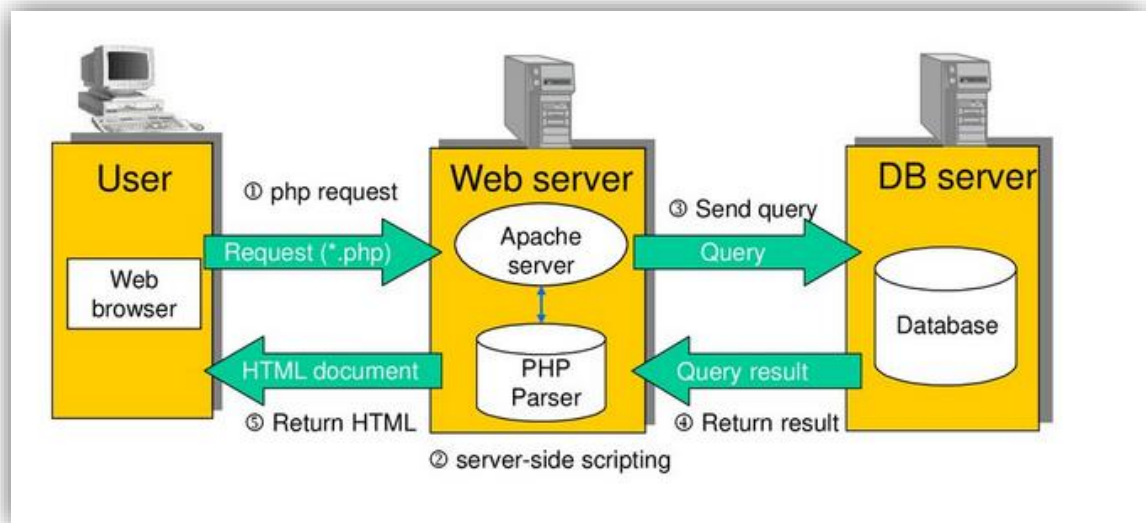
- En "Path", agrega la ruta D:\server\php.

- Reinicia la consola para que los cambios surtan efecto.

5. Verifica la instalación Abre una consola (CMD) y escribe:php -v

¿Cómo Funciona PHP?

Cuando solicitas **una página con código PHP**, el servidor pasa ese código a un **intérprete PHP**. Este intérprete procesa las instrucciones, que pueden incluir interactuar con bases de datos, manejar datos de formularios o generar contenido dinámico. **El resultado de este procesamiento es código HTML, CSS y JavaScript puro**, que el servidor luego envía a tu navegador. Tu navegador solo recibe y muestra este HTML final, sin ver nunca el código PHP original, haciendo que las páginas web sean dinámicas e interactivas.



Archivo de Configuración httpd.conf

El archivo httpd.conf es el corazón de la configuración de Apache HTTP Server. Contiene directivas que dictan cómo se comporta el servidor, desde qué puertos escucha hasta cómo maneja las solicitudes y qué contenido sirve. Algunas principales Directivas son:

- **ServerRoot "ruta":** Define el directorio base donde está instalada la configuración y los módulos de Apache.
- **Listen [IP:]puerto:** Indica a Apache en qué direcciones IP y puertos debe escuchar las solicitudes entrantes.
- **ServerName FQDN[:puerto]:** Establece el nombre de host y puerto que el servidor usa para identificarse a sí mismo.
- **DocumentRoot "ruta":** Especifica el directorio principal desde donde Apache sirve los archivos web por defecto.

- **DirectoryIndex nombre_archivo:** Define el archivo que Apache buscará y servirá por defecto cuando se solicita un directorio.
- **ErrorLog "ruta":** Define la ubicación del archivo donde Apache registra los errores del servidor.
- **LogLevel nivel:** Controla el nivel de detalle de los mensajes registrados en el ErrorLog.
- **LoadModule nombre_modulo:** Carga módulos adicionales para extender la funcionalidad de Apache.
- **User usuario / Group grupo:** (Unix/Linux) Define el usuario y grupo bajo los cuales se ejecuta el proceso de Apache.
- **<Context>:** Define configuraciones específicas para aplicaciones web.

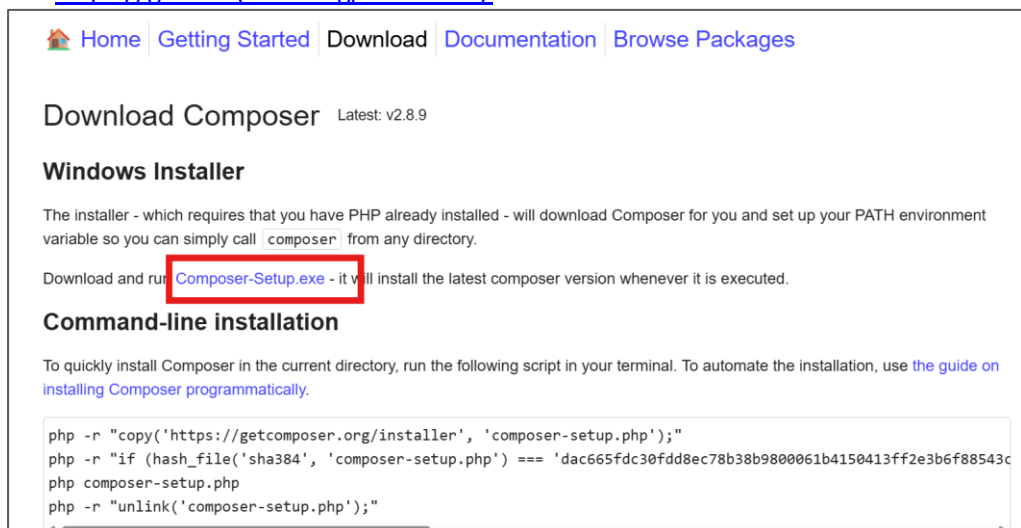
3. Gestión de Dependencias con Composer

Herramienta Composer

Composer es la herramienta fundamental y el gestor de dependencias de facto para PHP. Funciona de manera similar a otras herramientas como npm (Node.js) o pip (Python). Composer no instala paquetes globalmente en tu sistema; en su lugar, instala las dependencias por cada proyecto en un directorio local (generalmente vendor/).

Descargar Composer

Url: <https://getcomposer.org/download/>



The screenshot shows the 'Download Composer' page for version v2.8.9. It features a navigation bar with links: Home, Getting Started, Download, Documentation, and Browse Packages. The 'Download Composer' section is active, showing the 'Latest: v2.8.9' version. Below this, the 'Windows Installer' section is highlighted, explaining that the installer will download Composer and set up the PATH environment variable. A red box highlights the text 'Composer-Setup.exe' in the instructions. The 'Command-line installation' section provides a terminal script to install Composer programmatically.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === 'dac665fdc30fdd8ec78b38b980061b4150413ff2e3b6f88543c
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Instalación de Composer

1. Requisitos previos

Asegúrate de tener PHP instalado y agregado al PATH del sistema. Puedes verificarlo abriendo la consola (CMD) y escribiendo:

```
php -v
```

2. Ejecuta el instalador

- Abre el archivo descargado.
- Elige la ruta del ejecutable de PHP (por ejemplo, D:\php\php.exe).
- Acepta las opciones por defecto y finaliza la instalación.

3. Verifica la instalación

Abre una nueva consola y escribe:

```
>composer -V
```

4. Sintaxis PHP

Estructura básica. Todo código PHP debe ir entre etiquetas especiales:.

Sintaxis

```
<?php
// Tu código PHP aquí
?>
```

Ejemplo:

```
<?php
echo "Hola, mundo!";
?>
```

1. Comentarios

Puedes comentar tu código así:

```
// Comentario de una línea
# También válido para una línea
/* Comentario
de varias líneas */
```

2. Variables

Se declaran con el signo \$:

```
$nombre = "Ana";
$edad = 25;
```

Las variables sí distinguen entre mayúsculas y minúsculas (\$Color y \$color son diferentes), pero las funciones y palabras clave como echo no.

3. Salida de datos

``echo`` y ``print``

Se utilizan para mostrar información en pantalla. Aunque ambas son similares, ``echo`` es un poco más rápido y permite la salida de múltiples parámetros.

```
<?php
echo "¡Hola, mundo!";
print "PHP es genial";
?>
```

4. Control de flujo condicional

``if``, ``else`` y ``elseif``

Permiten ejecutar bloques de código basados en condiciones.

```
<?php
$edad = 20;
if ($edad >= 18) {
    echo "Eres mayor de edad.";
} elseif ($edad == 17) {
```

```
    echo "Casi llegas.";
} else {
    echo "Eres menor de edad.";
}
?>
```

`switch`

Es útil cuando se evalúa una misma variable contra diferentes valores.

```
<?php
$color = "azul";
switch ($color) {
    case "rojo":
        echo "El color es rojo.";
        break;
    case "azul":
        echo "El color es azul.";
        break;
    default:
        echo "Color no reconocido.";
}
?>
```

5. Estructuras de repetición (bucles)

`for`

Ejecuta un bloque de código un número definido de veces.

```
<?php
for ($i = 0; $i < 5; $i++) {
    echo "Número: " . $i . "<br>";
}
?>
```

`while` y `do...while`

Se utilizan para repetir un bloque de código mientras se cumpla una condición. El `do...while` se ejecuta al menos una vez.

```
<?php
$x = 0;
while ($x < 5) {
    echo "Valor: $x<br>";
    $x++;
}

$y = 0;
do {
    echo "Valor: $y<br>";
    $y++;
} while ($y < 5);
?>
```

``foreach``

Especialmente útil para recorrer arrays, ya que itera sobre cada elemento del array.

```
<?php
$frutas = ["Manzana", "Banana", "Cereza"];
foreach ($frutas as $fruta) {
    echo "Fruta: $fruta<br>";
}
?>
```

6. Manejo de archivos e inclusión

``include``, ``include_once``, ``require`` y ``require_once``

Permiten incorporar archivos externos dentro de tu script. La diferencia principal es la forma en que manejan errores:

- ``include``: Genera un `_warning_` si el archivo no se encuentra pero continúa la ejecución.
- ``require``: Muestra un `_fatal error_` si el archivo es inexistente y detiene la ejecución.

```
<?php
include 'funciones.php';
require 'config.php';
?>
```

7. Funciones

Definición y retorno de funciones

Puedes encapsular bloques de código en funciones para reutilizarlas y organizar tu programa.

```
<?php
function saludar($nombre) {
    return "¡Hola, $nombre!";
}
echo saludar("Carlos");
?>
```

8. Finalización del script

``exit`` o ``die``

Estas instrucciones detienen la ejecución del script. Se utilizan, por ejemplo, cuando ocurre un error crítico o se cumple una condición especial.

```
<?php
if (!file_exists("importante.txt")) {
    exit("El archivo importante no se encontró.");
}
?>
```

Estas instrucciones y estructuras son la base para desarrollar en PHP, permitiéndote construir desde scripts simples hasta aplicaciones web complejas.

5. Laravel Framework



Laravel es un framework de PHP de código abierto que se centra en la simplicidad, la elegancia y el uso de patrones modernos de desarrollo. Se apoya en el patrón Modelo-Vista-Controlador (MVC) para organizar la lógica de la aplicación, lo que facilita el mantenimiento del código y la separación de responsabilidades.

Laravel framework es muy extenso y su comunidad crece día a día para ayudar al desarrollo de aplicaciones web.

Link de Laravel Framework: <https://laravel.com/>

Documentación Laravel Framework: <https://laravel.com/docs/12.x>

Terminología Laravel

Rutas (Routes):

Son los puntos de entrada de la aplicación. Definidas en archivos como `routes/web.php` o `routes/api.php`, las rutas asignan una URL a una lógica específica (por ejemplo, a un controlador o a un `_closure_`) para responder a las solicitudes HTTP.

Controladores (Controllers):

Agrupar la lógica de negocio y gestionan las peticiones del usuario. Siguiendo el patrón MVC, los controladores reciben la solicitud, interactúan con los modelos y retornan una respuesta (típicamente a través de una vista).

Vistas y Blade:

Laravel utiliza Blade, un motor de plantillas sencillo y potente, para separar la presentación del contenido y la lógica. Blade facilita la incorporación de código PHP en HTML mediante directivas específicas, permitiendo crear interfaces limpias y reutilizables.

Eloquent ORM:

Es la herramienta que ofrece Laravel para interactuar con la base de datos. Eloquent mapea las tablas a clases `_modelo_`, lo que permite trabajar con datos mediante una sintaxis intuitiva para operaciones CRUD (crear, leer, actualizar y eliminar registros).

Middleware:

Son filtros o capas intermedias que se sitúan entre la solicitud y la respuesta. Se utilizan para tareas como la autenticación, autorización o la verificación de condiciones antes de acceder a ciertas rutas, aportando una capa de seguridad y personalización del flujo de la aplicación.

Artisan:

Es la interfaz de línea de comandos de Laravel. Con Artisan se automatizan tareas comunes del desarrollo, como la generación de código (controladores, migraciones, etc.), la ejecución de pruebas y la administración de la base de datos, entre otras funciones.

Migrations y Seeds:

Las migraciones permiten versionar y modificar la estructura de la base de datos de forma programática, facilitando el trabajo colaborativo. Por otro lado, los `_seeds_` son scripts destinados a poblar la base de datos con datos iniciales o de prueba.

Service Providers:

Constituyen el mecanismo central de Laravel para registrar y configurar los servicios y componentes que forman la aplicación. A través de los service providers se cargan aspectos fundamentales como la base de datos, el sistema de colas, el reloj de tareas, etc.

Facades:

Ofrecen una interfaz estática para clases y servicios que están registrados en el contenedor de servicios de Laravel. Esto facilita el uso de componentes como el sistema de cache, la base de datos o incluso el mismo Eloquent, simplificando la sintaxis sin sacrificar la flexibilidad.

Composer:

Aunque no es exclusivo de Laravel, Composer es el gestor de dependencias con el que se instala y actualiza el framework, junto con sus paquetes y bibliotecas, permitiendo un manejo eficiente de las versiones y dependencias en tus proyectos.

Instalación de Laravel con Composer

Antes de comenzar, asegúrate de tener instalado: PHP (versión 8.1 o superior), - Composer (el gestor de dependencias de PHP) y Node.js y NPM (opcional, pero útil para compilar assets)

1. Verificar configuración de php.ini

Debe habilitarse las siguientes extensiones

```
extension=zip
extension=fileinfo
extension=pdo_sqlite
extension=pgsql
```

2. Instalar con composer

Ejecutar el siguiente commando dentro de httdocs/laravel

```
composer create-project laravel/laravel nombre-de-tu-proyecto
```

3. Accede al directorio del proyecto

```
cd nombre-de-tu-proyecto
```

4. Inicia el servidor de desarrollo

```
php artisan serve
```

5. Luego abre tu navegador en <http://localhost:8000>.

Instalación de Laravel con Autenticación desde Laravel Installer

Antes de comenzar, asegúrate de tener instalado: PHP (versión 8.1 o superior), - Composer (el gestor de dependencias de PHP) y Node.js y NPM (opcional, pero útil para compilar assets)

1. Verificar configuración de php.ini

Debe habilitarse las siguientes extensiones

```
extension=zip
extension=fileinfo
extension=pdo_sqlite
extension=pgsql
```

2. Instala el instalador de Laravel globalmente

Abre tu terminal y ejecuta:

```
composer global require "laravel/installer"
```

3. Crea un nuevo Proyecto con Autenticación

Una vez instalado, puedes crear un nuevo proyecto con:

```
laravel new nombre-del-proyecto
```

```
D:\DevsServers\Apache24\htdocs\laravel>laravel new Lab13
```



```
Which starter kit would you like to install? [None]:
```

```
[none]   ] None
[react]  ] React
[vue]    ] Vue
[livewire] Livewire
> React
```

```
Which authentication provider do you prefer? [Laravel's built-in authentication]:
```

```
[laravel] Laravel's built-in authentication
[workos]  WorkOS (Requires WorkOS account)
> laravel
```

```
Which testing framework do you prefer? [Pest]:
```

```
[0] Pest
[1] PHPUnit
> 1
```

```
Creating a "laravel/react-starter-kit" project at "./Lab13"
```

Tambien

```
Would you like to run npm install and npm run build? (yes/no) [yes]:
> yes
```

4. Accede al directorio del proyecto

```
cd nombre-de-tu-proyecto
```

5. Inicia el servidor de desarrollo

```
php artisan serve
```

6. Luego abre tu navegador en <http://localhost:8000>.

Estructura de Directorios de Laravel

La estructura de directorios en un proyecto Laravel está pensada para organizar el código de manera lógica, facilitando tanto el desarrollo como el mantenimiento de la aplicación.

1. Raíz del proyecto

Aquí se encuentran archivos fundamentales como `composer.json`, `artisan` (la herramienta de línea de comandos de Laravel) y otros archivos de configuración. Este directorio es el punto de partida del proyecto.

2. app

Es el núcleo de la aplicación y contiene la lógica de negocio. Dentro de `app` se suelen encontrar subdirectorios como:

- Console: Contiene los comandos personalizados de Artisan.
- Http: Aquí se alojan los controladores, middleware, y formularios (requests).
- Providers: Registra los proveedores de servicios de la aplicación.
- Models (opcional): Aunque en versiones recientes puedes colocar modelos directamente en este directorio, algunos proyectos prefieren una carpeta específica para ellos.

3. bootstrap

Contiene el archivo `app.php`, encargado de iniciar el framework, y un directorio `cache` para almacenar archivos generados que ayudan a la optimización del rendimiento, como cachés de rutas o servicios.

4. config

Aquí se encuentran todos los archivos de configuración de la aplicación, abarcando desde la base de datos hasta la configuración de servicios externos y la gestión de la sesión, entre otros.

5. database

Aloja las migraciones (para versionar la base de datos), los factories (para generar datos de prueba) y los seeders (para poblar la base de datos con datos iniciales). También se puede encontrar, en algunos casos, una base de datos SQLite en este directorio.

6. public

Es el punto de entrada para todas las solicitudes HTTP, ya que contiene el archivo `index.php`. Además, en este directorio se alojan los activos públicos como imágenes, hojas de estilo (CSS) y archivos JavaScript.

7. resources

Aquí se gestionan los elementos que serán compilados o transformados, como las vistas (usualmente con el motor de plantillas Blade), archivos de lenguaje (para traducciones) y recursos sin compilar como CSS o JavaScript.

8. routes

En esta carpeta se definen todas las rutas de la aplicación. Por defecto, encontrarás archivos como:

- `web.php`: Incluye las rutas que conforman la interfaz de la aplicación web, a menudo agrupadas bajo un middleware que gestiona sesiones y protección CSRF.
- `api.php`: Contiene rutas orientadas a una API RESTful, que generalmente no mantienen estado.
- `console.php`: Para comandos basados en la línea de comandos, aunque existen múltiples enfoques sobre cómo organizar estas configuraciones.

9. storage

Este directorio se encarga de almacenar archivos generados por la aplicación, como logs, cachés, sesiones y archivos subidos. Se organiza en varias subcarpetas, por ejemplo, `app` para archivos específicos de tu aplicación, `framework` para cachés y compilaciones, y `logs` para los registros de actividad.

10. tests

Contiene los test unitarios y de funcionalidad que facilitan el aseguramiento de la calidad del código y la confiabilidad de la aplicación.

11. vendor

Es administrado por Composer y agrupa todas las dependencias del proyecto, incluyendo los paquetes de terceros necesarios para que el framework y otros componentes funcionen correctamente.

Segunda Unidad: Desarrollo PHP
GUÍA DE LABORATORIO N° 13:
DESARROLLO BACKEND PHP LARAVEL

Docente: Jaime Suasnabar Terrel

Fecha:

Duración: 90 minutos

Instrucciones:

OBJETIVOS

- Conocer el funcionamiento PHP y paso de parámetros

EJERCICIO 01. CREACIÓN DE PROYECTO WEB PHP LARAVEL

1. Crear una Aplicación Web PHP que muestre la fecha actual

1. Crear el proyecto Laravel

```
laravel new estudiantes-app  
cd estudiantes-app
```

2. Configurar conexión a MySQL

Edita tu archivo .env:

```
DB_DATABASE=academico  
DB_USERNAME=root  
DB_PASSWORD=tu_contraseña
```

3. Crear la migración del modelo Estudiante

```
php artisan make:model Estudiante -m
```

Edita database/migrations/xxxx_create_estudiantes_table.php:

```
public function up()  
{  
    Schema::create('estudiantes', function (Blueprint $table) {  
        $table->id('idEstudiante');  
        $table->string('nomEstudiante');  
        $table->string('dirEstudiante');  
        $table->string('ciuEstudiante');  
        $table->timestamps();  
    });  
}
```

Y luego ejecuta:

```
php artisan migrate
```

4. Agregar Tailwind CSS

Configura tailwind 3 con los siguientes comandos:

```
npm install -D tailwindcss@3 postcss autoprefixer
npx tailwindcss init -p
```

Configura tailwind.config.js y añade Tailwind en tu resources/css/app.css:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Compila assets:

```
npm install
npm run dev
```

5. Crear rutas, controlador y formulario

Rutas (resources/routes/web.php)

```
use App\Http\Controllers\EstudianteController;

Route::get('/', [EstudianteController::class, 'create']);
Route::post('/guardar', [EstudianteController::class, 'store']);
```

Controlador (app/Http/Controllers/EstudianteController.php)

Crear el controlador

```
php artisan make:controller EstudianteController
```

```
// app/Http/Controllers/EstudianteController.php
use App\Models\Estudiante;
use Illuminate\Http\Request;

public function create() {
    return view('formulario');
}

public function store(Request $request) {
    Estudiante::create([
        'nomEstudiante' => $request->nombre,
        'dirEstudiante' => $request->direccion,
        'ciuEstudiante' => $request->ciudad
    ]);
    return redirect('/')->with('mensaje', 'Estudiante registrado');
}
```

Modelo (app/Models/Estudiante.php)

Ya se creó en ítem 3

```
protected $table = 'estudiantes';
protected $primaryKey = 'idEstudiante';
protected $fillable = ['nomEstudiante', 'dirEstudiante', 'ciuEstudiante'];
```

6. Vista con Tailwind (resources/views/formulario.blade.php)

Crear la vista formulario

```
php artisan make:view formulario
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
```

```
<title>Registro</title>
@vite('resources/css/app.css')
</head>
<body class="bg-gray-100 flex justify-center items-center h-screen">
  <form action="/guardar" method="POST" class="bg-white p-6 rounded shadow-md w-96">
    @csrf
    <h2 class="text-xl font-bold mb-4">Nuevo Estudiante</h2>
    <input type="text" name="nombre" placeholder="Nombre" required class="w-full mb-3 p-2 border rounded">
    <input type="text" name="direccion" placeholder="Dirección" required class="w-full mb-3 p-2 border rounded">
    <input type="text" name="ciudad" placeholder="Ciudad" required class="w-full mb-3 p-2 border rounded">
    <button type="submit" class="bg-blue-500 text-white px-4 py-2 rounded w-full">Guardar</button>
  </form>
</body>
</html>
```