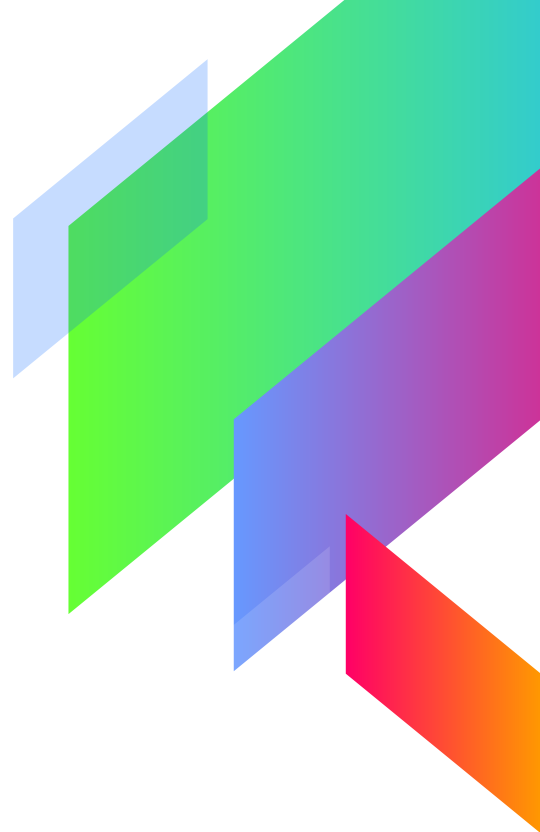




LARAVEL

CLASE 01



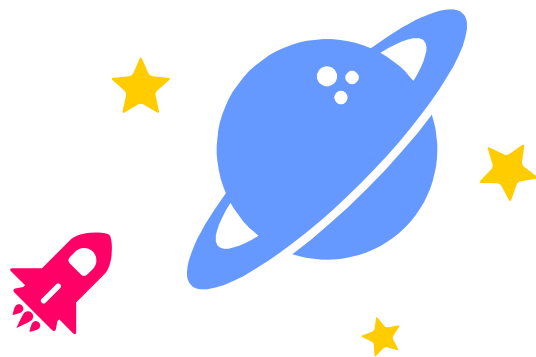


FRAMEWORK

Un framework nos provee un entorno de desarrollo con el cual podemos resolver problemas comunes más rápidamente.

¿Por qué usar un framework?

- › Nos permite organizar el código y nos brinda un sistema de archivos
- › Nos permite desarrollar más rápido y escribir menos código
- › Nos permite trabajar en equipo fácilmente utilizando los mismos criterios
- › Nos permite tener un lenguaje en común independiente del desarrollador



BIBLIOTECA

Una biblioteca (o librería) nos provee código para resolver un problema específico sin necesidad de mucho contexto.

COMPOSER

<https://getcomposer.org>

COMPOSER es un **gestor de dependencias** para proyectos **programados en PHP**. Esto quiere decir que nos permite instalar y mantener actualizados los paquetes de software necesarios para nuestro proyecto.



Accede a la **GUÍA DE INSTALACIÓN** de Composer:



- › **Instalación en LINUX**
- › **Instalación en MAC**
- › **Instalación en WINDOWS**

LARAVEL

<https://laravel.com>

LARAVEL es un `framework` de código abierto para desarrollar aplicaciones y servicios web de un modo mucho más rápido, fácil, y seguro.

Para poder comenzar a trabajar, es necesario crear un proyecto nuevo a través de composer.





INSTALACIÓN de Laravel:

Instalación en LINUX / MAC / WINDOWS

Via composer, usando la terminal:

```
$ composer create-project laravel/laravel nombre_proyecto
```

Esto va a tardar unos minutos, tengamos paciencia.



INSTALACIÓN de Laravel:

Archivo importantísimo

.env

Si este archivo no está, tendremos que copiar y pegar el archivo:

.env.example le colocamos de nombre **.env**

Luego ejecutar:

```
$ php artisan key:generate
```



Corriendo Laravel:

Via la terminal:

Ingresamos a la carpeta y corremos el servidor:

```
$ php artisan serve
```

```
Laravel development server started: <http://127.0.0.1:8000>
```



Visualizando Laravel:

Ingresamos al navegador con la ruta:

127.0.0.1:8000

Laravel

[DOCUMENTATION](#)

[LARACASTS](#)

[NEWS](#)

[NOVA](#)

**ES MOMENTO DE
PRACTICAR !**



FILE SYSTEM

PHP

Para cada sitio de nuestra web solemos crear un archivo nuevo, es decir, otro punto de entrada.

`/index.php`

`/registro.php`

LARAVEL

Con Laravel esto ya no es así, contamos con un punto de entrada único a la aplicación y distintas Clases que toman el pedido según la ruta.

`public/index.php`

FILE SYSTEM

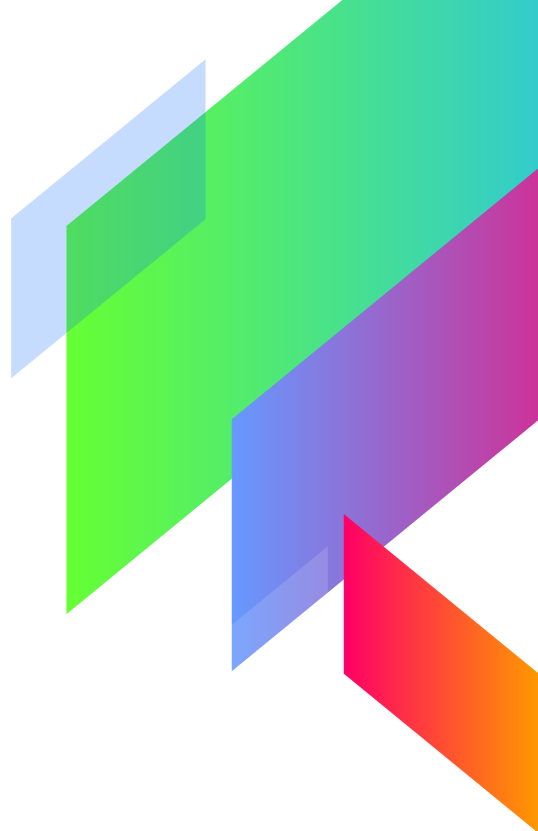
Es decir, debemos quitar la idea de que podemos identificar el archivo que controla el sitio a través de lo que vemos en la URL y en el file system.





ESTRUCTURA DE DIRECTORIOS

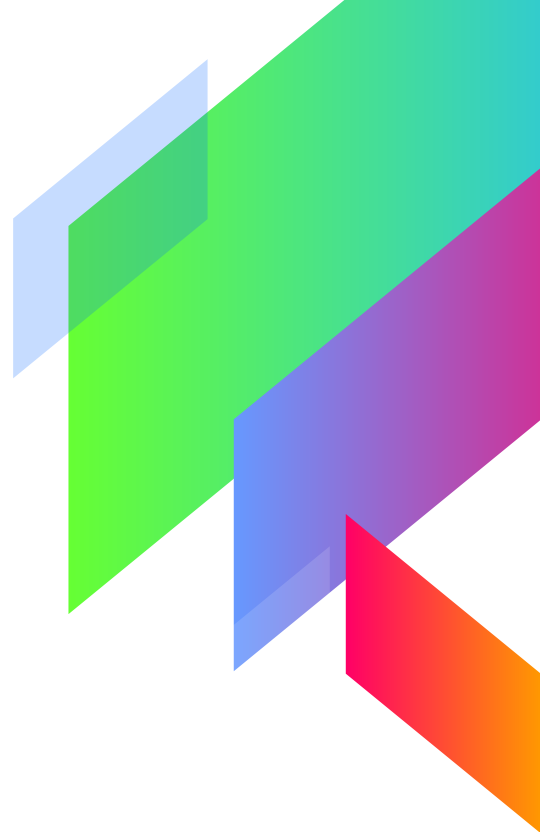
Conozcamos el la estructura de
directorios de Laravel





ROUTING

Conozcamos el sistema
de ruteo de Laravel



METODOS HTTP

MÉTODO

FUNCIÓN

GET

Se utiliza para **consultar** elementos en la base de datos.

POST

Se utiliza para **crear** elementos en la base de datos.

PUT / PATCH

Se utiliza para **modificar** elementos en la base de datos.

DELETE

Se utiliza para **eliminar** elementos en la base de datos.

> routes/web.php

```
<?php
```

```
//Definimos nuestra primera ruta en este archivo
```

```
Route::get('home', function () {  
    return 'Hello World';  
});
```

Para acceder a esta ruta debemos ingresar en el navegador a:

<http://localhost:8000/home>

Recuerda tener el servidor de laravel corriendo

```
<?php
```

```
//Rutas con parámetros
```

```
Route::get('home/{name}', function ($name) {  
    return 'Hello ' . $name;  
});
```

Laravel interpreta como una variable lo que está entre las llaves y lo inyecta a la función anónima automáticamente.

```
<?php
```

```
// Rutas con PARÁMETROS OPCIONALES
```

```
Route::get('post/{comment_id?}', function ($comment_id = null) {  
    if ($comment_id === null) {  
        return 'No hay comentarios';  
    } else {  
        return $comment_id;  
    }  
});
```

Para hacer la variable opcional al final le colocamos el '?', dado que Laravel inyecta esa variable en la función anónima podemos decirle que puede tener un valor por defecto.

**ES MOMENTO DE
PRACTICAR !**



```
> routes/web.php
```

```
<?php
```

```
// Rutas Hacia CONTROLADORES
```

```
Route::get('/movies', 'MoviesController@index);           // Consulta (lista)
```

```
Route::get('/movies/{id}', 'MoviesController@show');     // Consulta (item)
```

```
Route::post('/movies', 'MoviesController@store);         // Alta
```

```
Route::patch('/movies/{id}', 'MoviesController@update);  // Modificación
```

```
Route::delete('/movies/{id}', 'ActoresController@destroy); // Baja
```

The image features abstract geometric shapes in the corners. On the left, there are overlapping shapes in green, blue, orange, and purple. On the right, there are overlapping shapes in green, blue, purple, and orange. The central text is in a bold, black, sans-serif font.

**¡ HASTA LA
PROXIMA !**