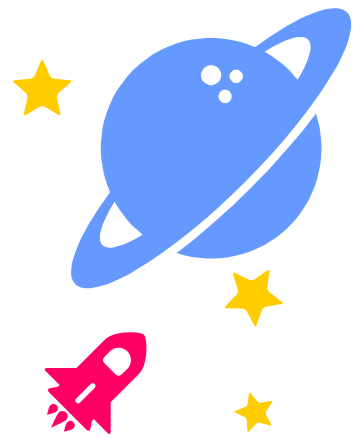


Abstract geometric shapes in the top-left corner, including a green parallelogram, a light blue parallelogram, a brown parallelogram, and a large purple parallelogram.

LARAVEL

CLASE 02

Abstract geometric shapes in the top-right corner, including a green parallelogram, a light blue parallelogram, a purple parallelogram, and a red parallelogram.



Laravel y Git

El archivo .gitignore

Iniciar un proyecto


1. `git init`
2. `git add .`
3. `git commit -m 'Iniciando con Laravel'`
4. `git remote add origin 'ruta del repo'`
5. `git push origin master`





Laravel y Git

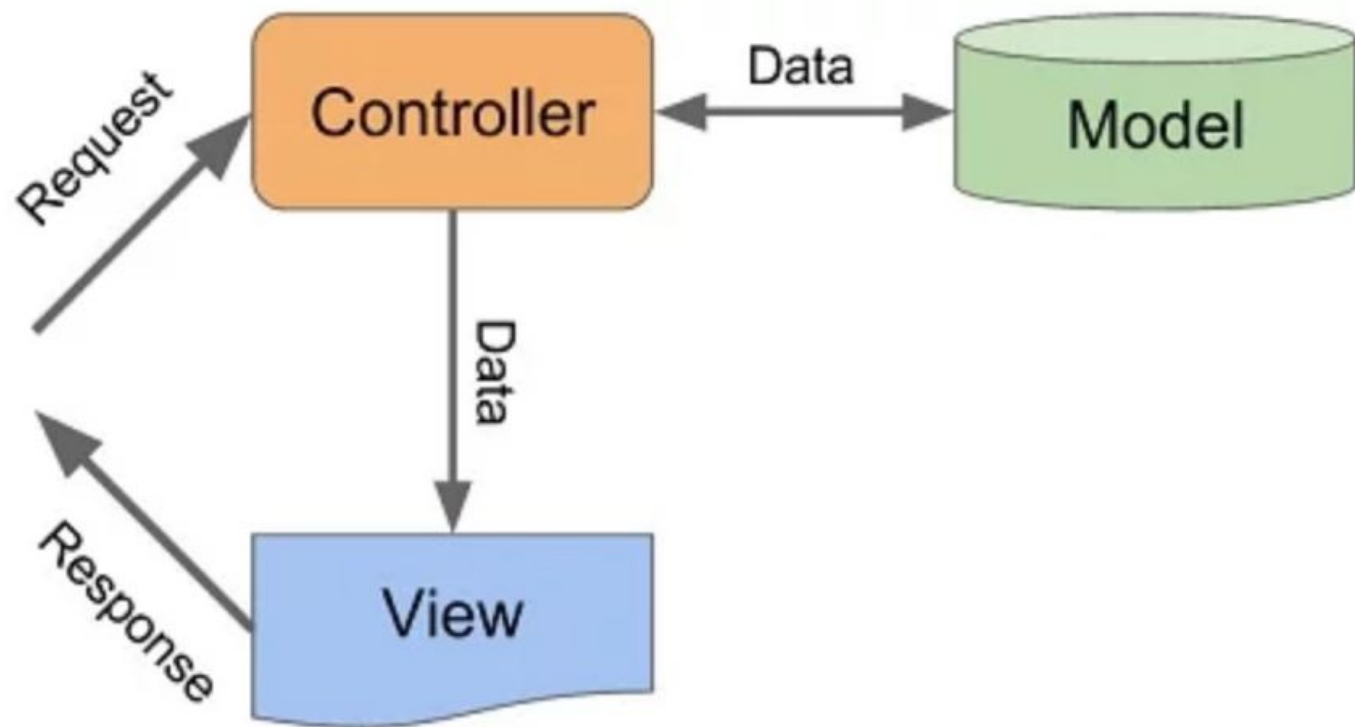
Clonar un proyecto

1. `git clone 'url.....'`
 2. `cd nombre_proyecto`
 3. `composer install`
 4. Crear el archivo `.env` usando el `.env.example`
 5. `php artisan key:generate`
- 



MVC

MVC es un patrón de arquitectura de software. Este sistema propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador. El objetivo es separar el manejo de datos y la lógica de negocio, de la interfaz de usuario.



MVC



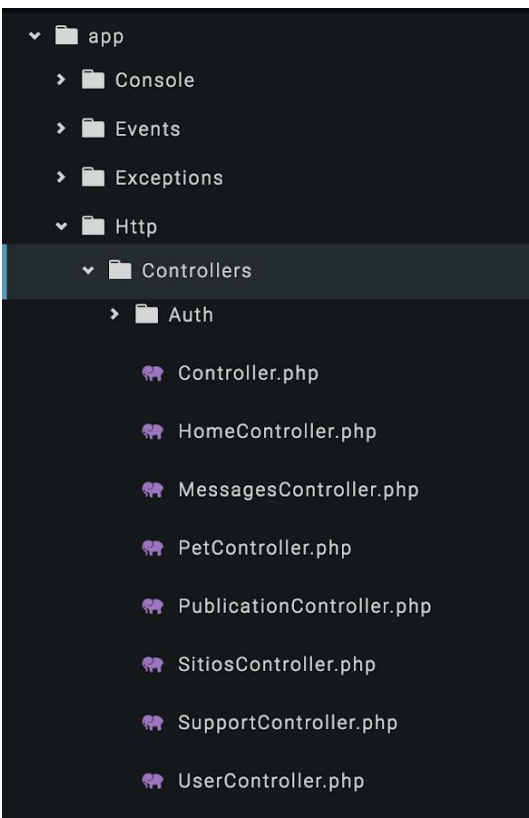
A series of overlapping, semi-transparent geometric shapes in shades of green, blue, orange, and purple, arranged in a stepped pattern in the top-left corner.

CONTROLLERS

¿Qué son y cómo funcionan los controladores?

A series of overlapping, semi-transparent geometric shapes in shades of green, blue, purple, and orange, arranged in a stepped pattern in the top-right corner.

<https://laravel.com/docs/master/controllers>



En lugar de definir toda la **lógica de tus peticiones** en un único archivo routes.php, puedes organizar el comportamiento de tu aplicación utilizando **clases controladoras**. Cada controlador puede agrupar en una clase la lógica de varias peticiones HTTP relacionadas.

The slide features decorative geometric shapes in the top-left and top-right corners. These shapes are composed of overlapping translucent polygons in various colors including green, blue, purple, orange, and red, creating a modern, abstract design.

CONTROLLERS

Los controllers cuentan con la lógica necesaria para procesar el pedido en cuestión, incluyendo la obtención de datos.



¿Cómo creamos un controlador?

Desde la consola de comandos, ejecutamos el siguiente código:

```
php artisan make:controller NombreControlador
```



> `App/Http/Controllers/HomeController.php` // La lógica de tu sitio se define en los controladores

```
<?php
    namespace App\Http\Controllers;

    use App\Http\Requests;
    use Illuminate\Http\Request;

    class HomeController extends Controller {
        /**
         * Show the application dashboard.
         *
         * @return \Illuminate\Http\Response
         */
        public function index() {
            return view('home');
        }
    }
```

> App/Http/Controllers/HomeController.php // Enviemos valores a nuestra vista

<?php

```
namespace App\Http\Controllers;
```

```
use App\Http\Requests;
```

```
use Illuminate\Http\Request;
```

```
class HomeController extends Controller {
```

```
    /**
```

```
     * Show the application dashboard.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index() {
```

```
        return view('home', ['titulo' => 'Hola mundo']);
```

o

```
        return view('home')->with('titulo', 'Hola mundo');
```

```
    }
```

```
}
```

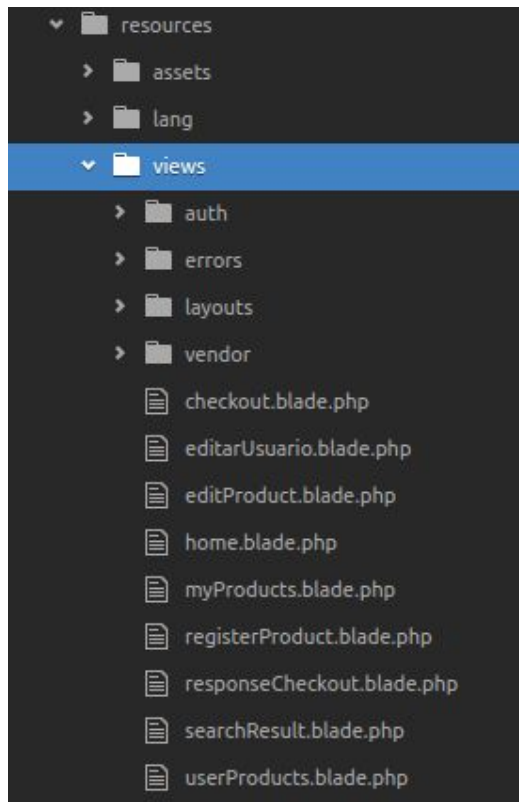
A cluster of overlapping, semi-transparent geometric shapes in shades of green, blue, orange, and purple, located in the top-left corner of the slide.

VISTAS

¿Qué son y cómo funcionan los vistas?

A cluster of overlapping, semi-transparent geometric shapes in shades of green, blue, purple, and orange, located in the top-right corner of the slide.

<https://laravel.com/docs/master/views>



En lugar de tener la lógica del pedido, el acceso a base de datos y el HTML en el mismo archivo, Laravel lo divide. Las vistas cuentan únicamente con lo que el usuario va a ver, es decir, el Frontend.

Las vistas se crean dentro de la carpeta **views**

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

```
<?php
```

```
Route::get('foo', function () {  
    return view('home');  
});
```

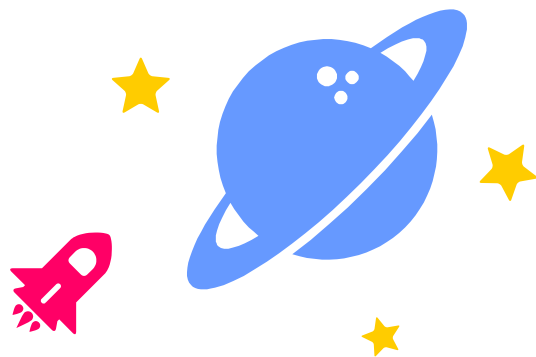
Así podemos decirle a un request que retorne determinada vista

**ES MOMENTO DE
PRACTICAR !**





B
L
@
D
E



BLADE

Blade template es el poderoso motor de plantillas que ya viene integrado en Laravel. Nos permite crear nuestras propias plantillas para tener vistas más organizadas, ahorrar código, y convertir PHP en HTML.

COMANDOS BLADE

Estructuras de control

@if @elseif @else
@unless
@for @while
@foreach
@forelse @empty

Directivas

@section @yield
@extends
@include @each
@push @stack

```
{{-- Comentario en Blade --}}
```

```
<h1>
```

```
@if(isset($alumno))
```

```
    Hola, {{ $alumno }}!
```

```
@elseif(date('w') == 4)
```

```
    Feliz jueves!
```

```
@else
```

```
    Aprendiendo Laravel!
```

```
@endif
```

```
</h1>
```

```
<h2>Profes:</h2>
```

```
<ul>
```

```
@foreach(['Guido', 'Gonzalo', 'Francisco'] as $profe)
```

```
<li>
```

```
{{ $profe }}
```

```
@unless($profe == 'Guido')
```

```
(Son profesores ayudantes)
```

```
@endunless
```

```
</li>
```

```
@endforeach
```

```
</ul>
```

// Cuando insertamos {{ \$var }} es lo mismo que hacer un echo(\$var);

```
{{-- Bucles --}}
```

```
<ol>
```

```
  @for($i = 1; $i <= 3; $i++)
```

```
    <li>{{ $i }}</li>
```

```
  @endfor
```

```
</ol>
```

```
<ol>
```

```
  @while($i++ < 10)
```

```
    <li>Me faltó {{ $i }}</li>
```

```
  @endwhile
```

```
</ol>
```

```
@forelse([ ] as $item)
```

```
  {{-- Si hubiera items, entraríamos acá --}}
```

```
  <strong>No creo que entre acá... ¿{{ $item }}?</strong>
```

```
@empty
```

```
  {{-- Cuando la lista está vacía, se muestra esto (una sola vez) --}}
```

```
  <em>La lista está vacía.</em>
```

```
@endforelse
```

views/layouts/default.blade.php

```
{{-- Nuestra plantilla default --}}
<html>
    <body>
        @yield('content')
    </body>
</html>
```

views/home.blade.php

```
{{-- Extendemos una plantilla genérica --}}
@extends('layouts.default')

@section('content')
    {{-- La plantilla "padre" define secciones con yield --}}
    <h1>{{ $titulo }}</h1>
    <p>{{ $resumen }}</p>

    {{-- Podemos definir secciones también, y tener valores por defecto --}}
    <input name="rating" type="number" value="{{ $rating }}">
@endsection
```

Nuestro proceso para crear cualquier sección de nuestro sitio debe ser siguiente:



**ES MOMENTO DE
PRACTICAR !**



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

**¡ HASTA LA
PROXIMA !**