

Abstract geometric shapes in the top-left corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a purple parallelogram, all overlapping and tilted at various angles.

# **LARAVEL**

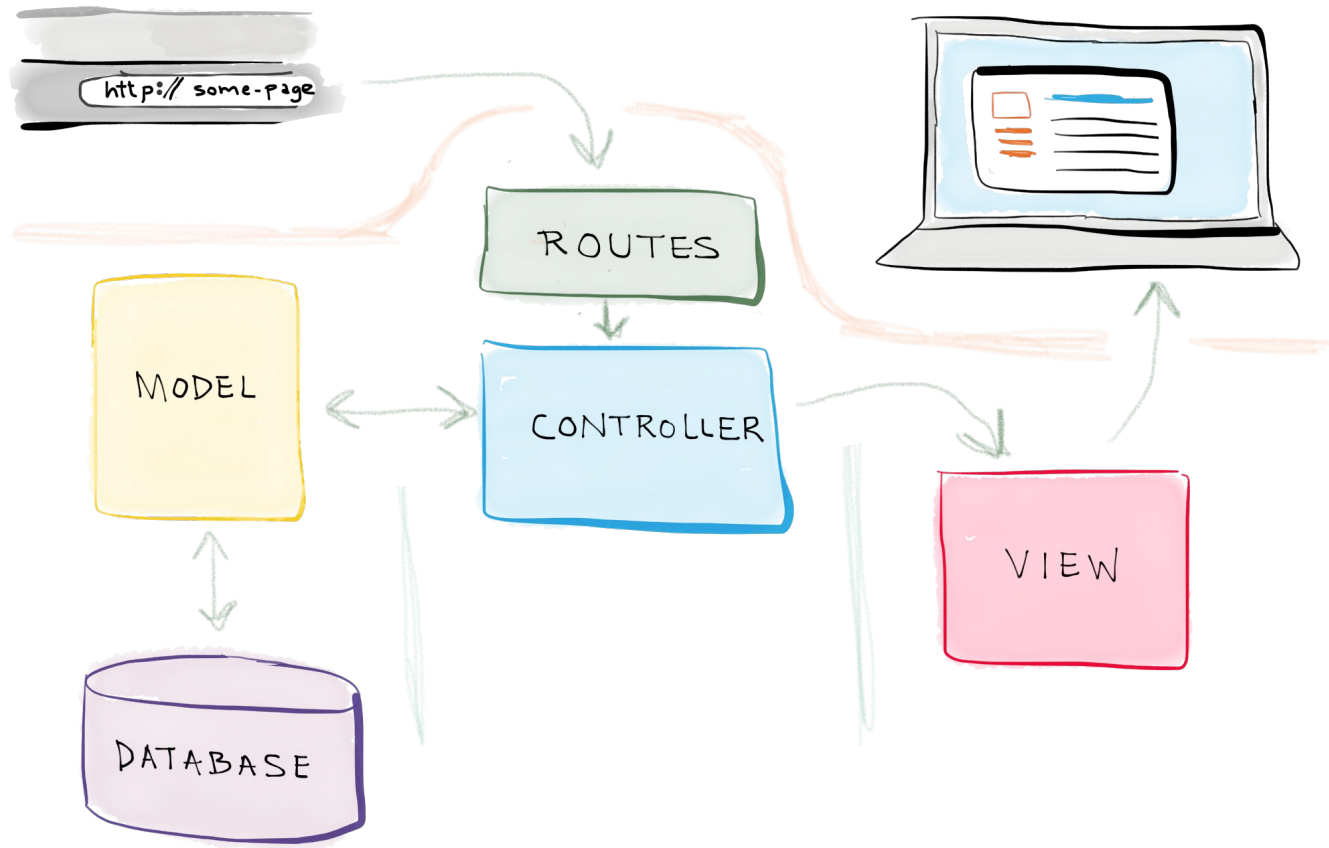
**AULA 02**

Abstract geometric shapes in the top-right corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a purple parallelogram, all overlapping and tilted at various angles.



# MVC

MVC é um padrão de arquitetura de software. Esse sistema propõe a construção de três componentes diferentes: o modelo, a visualização e o controlador. O objetivo é separar da interface de usuário o gerenciamento de dados e a lógica de negócios.

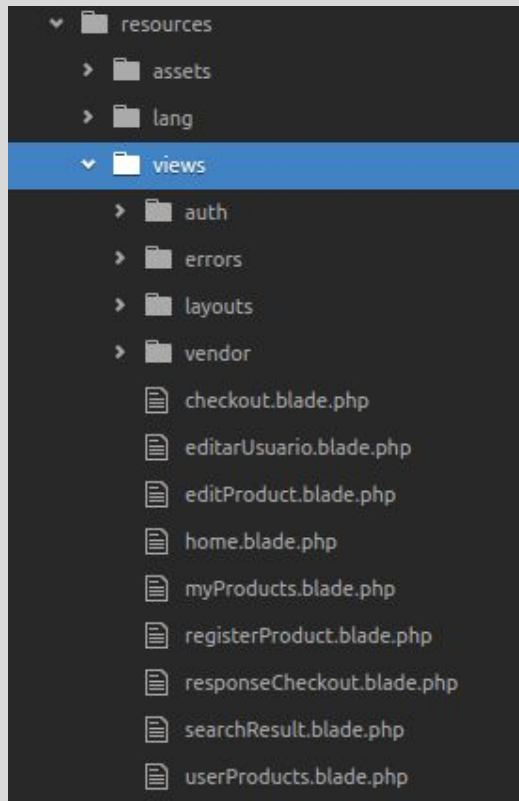


The slide features abstract geometric shapes in the corners. On the left, there are overlapping shapes in shades of green, blue, orange, and purple. On the right, there are overlapping shapes in shades of green, blue, purple, and orange. The background is a light gray.

# **VISUALIZAÇÕES**

O que são e como funcionam as visualizações?

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



Em vez de colocar a lógica do pedido, o acesso ao banco de dados e o HTML no mesmo arquivo, o Laravel divide tudo isso. As visualizações contêm apenas o que o usuário verá, ou seja, o front-end.

As visualizações são criadas dentro da pasta **views**

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

```
<?php
```

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

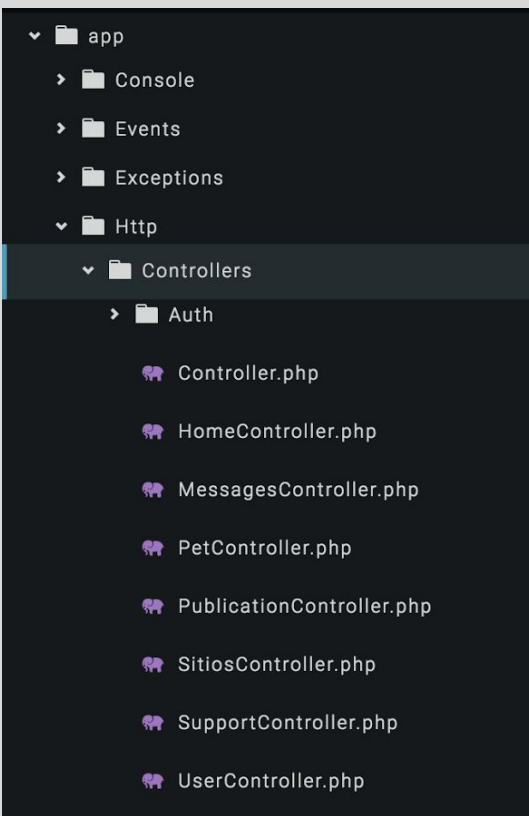
Dessa forma, é possível pedir que um request retorne uma determinada visualização

The slide 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 similar shapes in shades of green, blue, purple, and orange. The background is a light gray.

# CONTROLLERS

O que são e como funcionam os controladores?

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



Em vez de definir toda a **lógica dos pedidos** em um só arquivo routes.php, é possível organizar o comportamento do aplicativo utilizando **classes controladoras**. Os controladores podem agrupar em uma classe a lógica de vários pedidos HTTP relacionados.



The background features abstract, overlapping geometric shapes in various colors including green, blue, orange, purple, and red, creating a modern, layered effect.

# CONTROLLERS

Os controllers têm a lógica necessária para processar o pedido em questão, incluindo a obtenção de dados.

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

> App/Http/Controllers/HomeController.php // A lógica do site é definida nos 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 // Vamos enviar valores à visualização
<?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' => 'Olá, mundo']);
            ou
            return view('home')->with('titulo', 'Olá, mundo');
        }
    }
```

## Como é possível criar um controlador?

A partir da linha de comando, executar o seguinte código:

```
php artisan make:controller NomeControlador
```

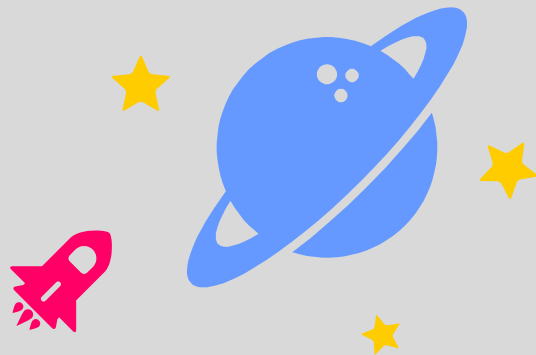
**HORA DE PRATICAR!**



Fazer o exercício 1



B  
L  
@  
D  
E



# BLADE

O Blade é um mecanismo potente de modelos que já vem integrado ao Laravel. Ele permite criar modelos para ter visualizações mais organizadas, economizar código e transformar PHP em HTML.

# COMANDOS BLADE

## Estruturas de controle

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

## De orientação

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



```
{{-- Comentário em Blade --}}
```

```
<h1>
```

```
@if(isset($aluno))
```

```
    Olá, {{ $aluno }}!
```

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

```
    Feliz quinta!
```

```
@else
```

```
    Aprendendo Laravel!
```

```
@endif
```

```
</h1>
```

```
<h2>Professores:</h2>
```

```
<ul>
```

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

```
<li>
```

```
{{ $professor }}
```

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

```
(São professores assistentes)
```

```
@endunless
```

```
</li>
```

```
@endforeach
```

```
</ul>
```

// Quando inserimos {{ \$var }} é  
igual a fazer um echo(\$var);

```
{{-- Loops --}}
```

```
<ol>
```

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

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

```
  @endfor
```

```
</ol>
```

```
<ol>
```

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

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

```
  @endwhile
```

```
</ol>
```

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

```
  {{-- Se houvesse itens, entrariam aqui --}}
```

```
  <strong>Acho que não entra aqui... ¿{{ $item }}?</strong>
```

```
@empty
```

```
  {{-- Quando a lista está vazia, se exibe isto (apenas uma vez) --}}
```

```
  <em>A lista está vazia.</em>
```

```
@endforelse
```

views/home.blade.php

```
{{-- Estendemos um modelo genérico --}}
@extends('layouts.default')

@section('content')
    {{-- O modelo "pai" define seções com yield --}}
    <h1>{{ $titulo }}</h1>
    <p>{{ $resumo }}</p>

    {{-- Também podemos definir seções e ter valores padrão --}}
    <input name="rating" type="number" value="{{ $rating }}">
@endsection
```

---

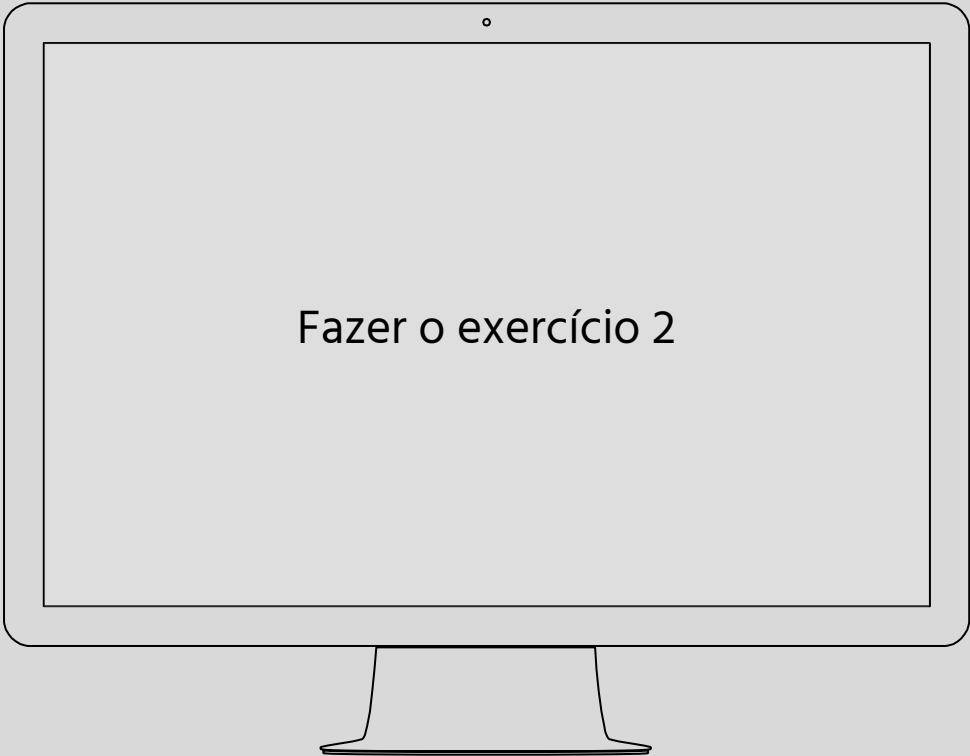
views/layouts/default.blade.php

```
{{-- Nosso modelo padrão --}}
<html>
    <body>
        @yield('content')
    </body>
</html>
```

**O processo para criar qualquer seção do nosso site deve ser o seguinte:**



**HORA DE PRATICAR!**



Fazer o exercício 2

The background features abstract, overlapping geometric shapes in various colors including light blue, green, cyan, magenta, orange, and red. These shapes are arranged in a way that creates a sense of depth and movement, framing the central text.

**ATÉ A  
PRÓXIMA!**