

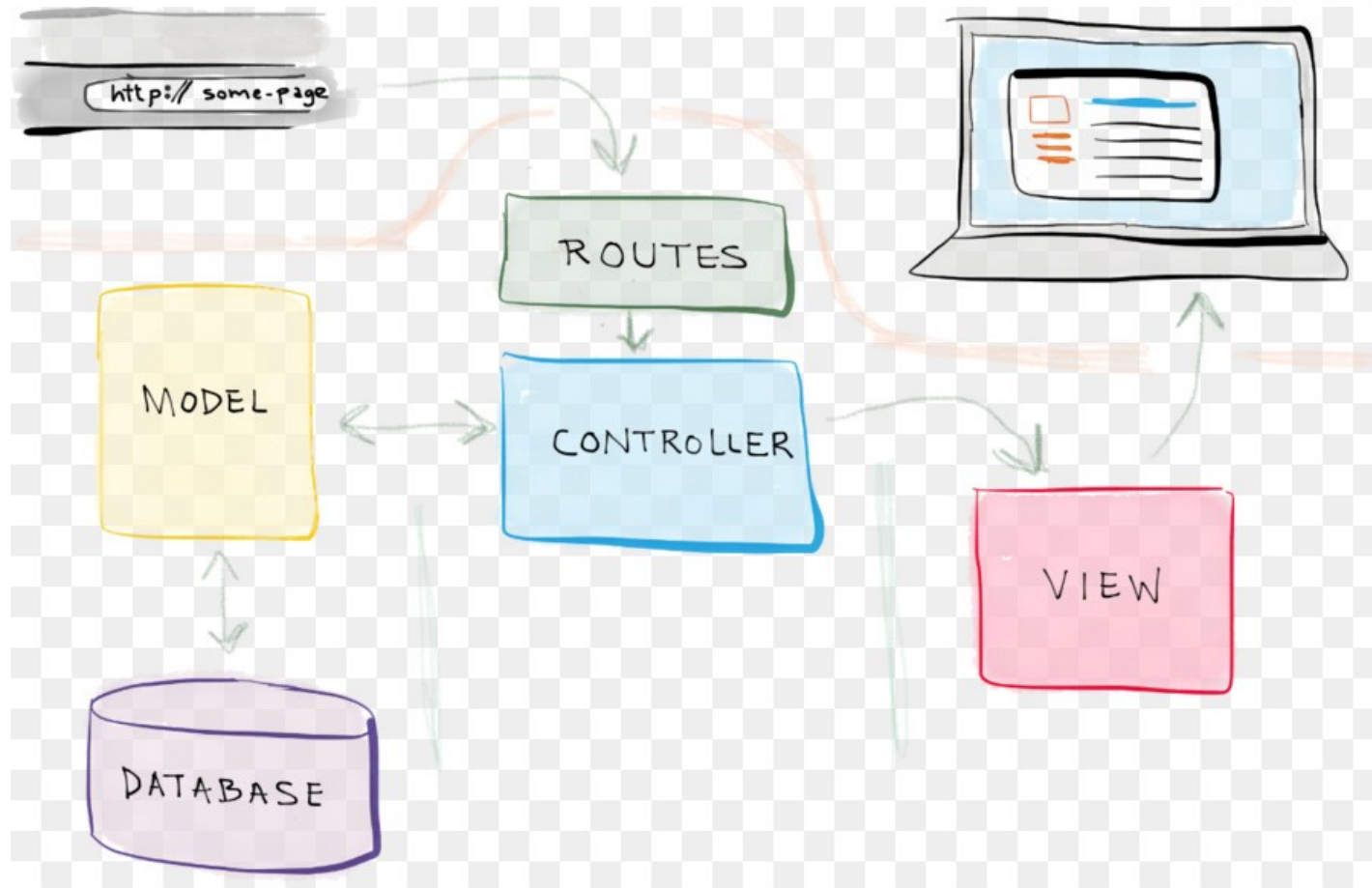
Desenvolvimento para Servidores 2

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Um framework é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma específica classe de software. [Deu89, JF88]

Você customiza um framework, para uma aplicação específica, através da criação de subclasses específicas para a aplicação, sendo essas subclasses específicas das classes abstratas do framework.

Os frameworks sempre tem um particular domínio de aplicação.



Fonte: <https://www.gratispng.com/png-tf44xi/>

Rotas

Laravel Rotas

A função essencial de qualquer framework de aplicativo web é receber solicitações de um usuário e distribuir respostas, geralmente por meio do HTTP(S).

Isso significa que definir as rotas de um aplicativo é o primeiro e mais importante projeto a ser considerado no aprendizado de um framework WEB.

Laravel

Rotas

Em um aplicativo Laravel, você definirá suas rotas “da web” em *routes/web.php* e suas rotas “de API” em *routes/api.php*.

Rotas da web são aquelas que serão visitadas pelo usuário final; rotas de API são referentes à sua API (se você tiver uma).

```
Route::get('/', function () {  
    return 'Oi Mundo';  
});
```

Laravel

Rotas - Exemplo

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

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

Laravel

Rotas - Exemplo

```
Route::get('/1', 'TasksController@home');  
Route::get('/2', 'TasksController@home2');  
  
Route::get('/teste', function(){  
    return 'olá ';  
});
```


Laravel

Rotas - Exemplo

```
Route::get('/ola/{nome?}', function($nome=null) {  
    if (isset($nome)){  
        return 'Olá '.$nome.' <br>';  
    } else {  
        return 'Olá anônimo';  
    }  
});
```

Laravel

Rotas - Exemplo

```
Route::get('/rotascomregras/{nome}/{n}',  
function($nome, $n){  
    for($x=0;$x<$n;$x++){  
        echo "Olá  $nome <br>";  
    }  
})->where('nome','[A-Za-z]+' )  
->where('n','[0-9]+' );
```

Laravel

Rotas - Exemplo

```
Route::prefix('/app')->group(function(){
    Route::get('/', function () {
        return view('app');
    })->name('app');
    Route::get('/user', function () {
        return view('user');
    })->name('app.user');;
    Route::get('/profile', function () {
        return view('profile');
    })->name('app.profile');;
});
```

Laravel

Rotas - Exemplo

```
Route::redirect('bla', '/app', 301);
```

```
Route::get('bla1', function() {  
    return redirect()->route('app');  
});
```

Laravel

Verbos de rotas

```
Route::get('/', function () {  
    return 'Oi Mundo';  
});  
  
Route::post('/', function () {});  
  
Route::put('/', function () {});  
  
Route::delete('/', function () {});  
  
Route::any('/', function () {});  
  
Route::match([ 'get', 'post'], '/', function () {});
```

Controladores

Laravel Controladores

Os controladores são classes que organizam a lógica de uma ou mais rotas em um único local.

Eles tendem a agrupar rotas semelhantes, principalmente se o aplicativo estiver estruturado de acordo com um formato tradicional de CRUD, nesse caso, um controlador pode manipular todas as ações que sejam executadas em um recurso específico.

Prática com um Controller

```
# php artisan make:controller --help
```

```
# php artisan make:controller ClienteControlador
```

```
# php artisan make:controller ClienteControlador --resource
```

```
Route::resource('cliente', 'ClienteControlador');
```


ClienteControlador Prática

```
class ClienteControlador extends Controller
{
    private $clientes = [
        ['id'=> 1, 'nome'=>'Orlando'],
        ['id'=> 2, 'nome'=>'Ana'],
        ['id'=> 3, 'nome'=>'Lucas'],
        ['id'=> 4, 'nome'=>'Lourdes']
    ];
}
```

Controladores

Método index

ClienteControlador

Método Index

```
public function index()
{
    echo '<ol>';
    foreach ($this->clientes as $cliente) {
        echo '<li>' . $cliente['nome'] . '</li>';
    }
    echo '</ol>';
}
```

ClienteControlador

Método Index

```
// resources/views/clientes/index.blade.php

<h3>Clientes </h3>

<a href="{ {route('cliente.create')}} "> Novo Cliente</a>

<ol>
    @foreach ($clientes as $cliente)
        <li>
            {{ $cliente['nome'] }}
            <a href="{ {route('cliente.edit',$cliente['id'])}} "> Editar</a>
        </li>
    @endforeach
</ol>
```

ClienteControlador

Método Index

```
public function index()  
{  
    $clientes = $this->clientes;  
    return view('clientes.index',compact(['clientes']));  
}
```

Controladores

Métodos create e store

ClienteControlador

Método create e store

```
// resources/views/clientes/create.blade.php
```

```
<h3>Novo Cliente </h3>
```

```
<form action="{{ route('cliente.store') }}" method="POST">
```

```
    @csrf
```

```
    <input type="text" name="nome">
```

```
    <input type="submit" value="Salvar">
```

```
</form>
```

ClienteControlador

Método create e store

```
public function create()  
{  
    return view('clientes.create');  
}
```

```
public function store(Request $request)  
{  
    $dados = $request->all();  
    dd($dados);  
}
```


ClienteControlador

Método create e store

```
public function store(Request $request)
{
    $id = count($this->clientes) + 1;
    $nome = $request->nome;
    $dados = ['id'=>$id, 'nome'=>$nome];
    $this->clientes[] = $dados;
    // dd($this->clientes);
    $clientes = $this->clientes;
    return view('clientes.index',compact(['clientes']));
}
```

Controladores Armazenando em Sessão

ClienteControlador Armazenando em Sessão



```
public function __construct(){
    $clientes = session('clientes');
    if(!isset($clientes)) {
        session(['clientes' => $this->clientes]);
    }
    public function index()
    {
        $clientes = session('clientes');
        return view('clientes.index',compact(['clientes']));
    }
```

ClienteControlador Armazenando em Sessão

```
public function store(Request $request)
{
    $clientes = session('clientes');
    $id = count($clientes) + 1;
    $nome = $request->nome;
    $dados = ['id'=>$id, 'nome'=>$nome];
    $clientes[] = $dados;
    session(['clientes' => $clientes]);
    return redirect()->route('cliente.index');
}
```

Controladores

Método show

ClienteControlador

Método show

```
/**
 * Display the specified resource.
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $clientes = session('clientes');
    $cliente = $clientes[$id - 1];
    return view('clientes.info', compact(['cliente']));
}
```

Controladores

Métodos Edit e Update

ClienteControlador

Método edit e update

```
public function edit($id)
{
    $clientes = session('clientes');
    $index = $this->getIndex($id, $clientes);
    $cliente = $clientes[$index];
    return view('clientes.edit', compact(['cliente']));
}
```


ClienteControlador

Método edit e update

```
public function update(Request $request, $id) {  
    $clientes = session('clientes');  
    $index = $this->getIndex($id, $clientes);  
    $clientes[$index]['nome'] = $request->nome;  
    session(['clientes' => $clientes]);  
    return redirect()->route('cliente.index');  
}
```

ClienteControlador

Método edit e update



```
// resources / views / clientes / edit.blade.php
```

```
<h3>Editar Cliente </h3>
```

```
<form action="{{ route('cliente.update', $cliente['id']) }}" method="POST">
```

```
    @csrf
```

```
    @method('PUT')
```

```
    <input type="text" name="nome" value="{{ $cliente['nome'] }}">
```

```
    <input type="submit" value="Salvar">
```

```
</form>
```

Controladores

Métodos Destroy

ClienteControlador

Método destroy

```
public function destroy($id) {  
    $clientes = session('clientes');  
    $index = $this->getIndex($id, $clientes);  
    array_splice($clientes, $index, 1);  
    session(['clientes' => $clientes]);  
    return redirect()->route('cliente.index');  
}
```

```
private function getIndex($id, $clientes) {  
    $ids = array_column($clientes, 'id');  
    $index = array_search($id, $ids);  
    return $index;  
}
```

ClienteControlador

Método destroy

```
// resource // views // clientes // index.blade.php
```

```
<ul>
```

```
    @foreach ($clientes as $cliente)
```

```
        <li>
```

```
            {{ $cliente['id'] }} | {{ $cliente['nome'] }}
```

```
            <a href="{{ route('cliente.edit', $cliente['id']) }}"> Editar</a>
```

```
            <a href="{{ route('cliente.show', $cliente['id']) }}"> Info</a>
```

```
            <form action="{{ route('cliente.destroy', $cliente['id']) }}" method="POST">
```

```
                @csrf
```

```
                @method('DELETE')
```

```
                <input type="submit" value="Apagar">
```

```
            </form>
```

```
        </li>
```

```
    @endforeach
```

```
</ul>
```

ClienteControlador

Método destroy

```
public function destroy($id) {  
    $clientes = session('clientes');  
    $index = $this->getIndex($id, $clientes);  
    array_splice($clientes, $index, 1);  
    session(['clientes' => $clientes]);  
    return redirect()->route('cliente.index');  
}
```

```
private function getIndex($id, $clientes) {  
    $ids = array_column($clientes, 'id');  
    $index = array_search($id, $ids);  
    return $index;  
}
```

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Documentação



<https://laravel.com/docs/7.x/routing>

<https://laravel.com/docs/7.x/controllers>