

02

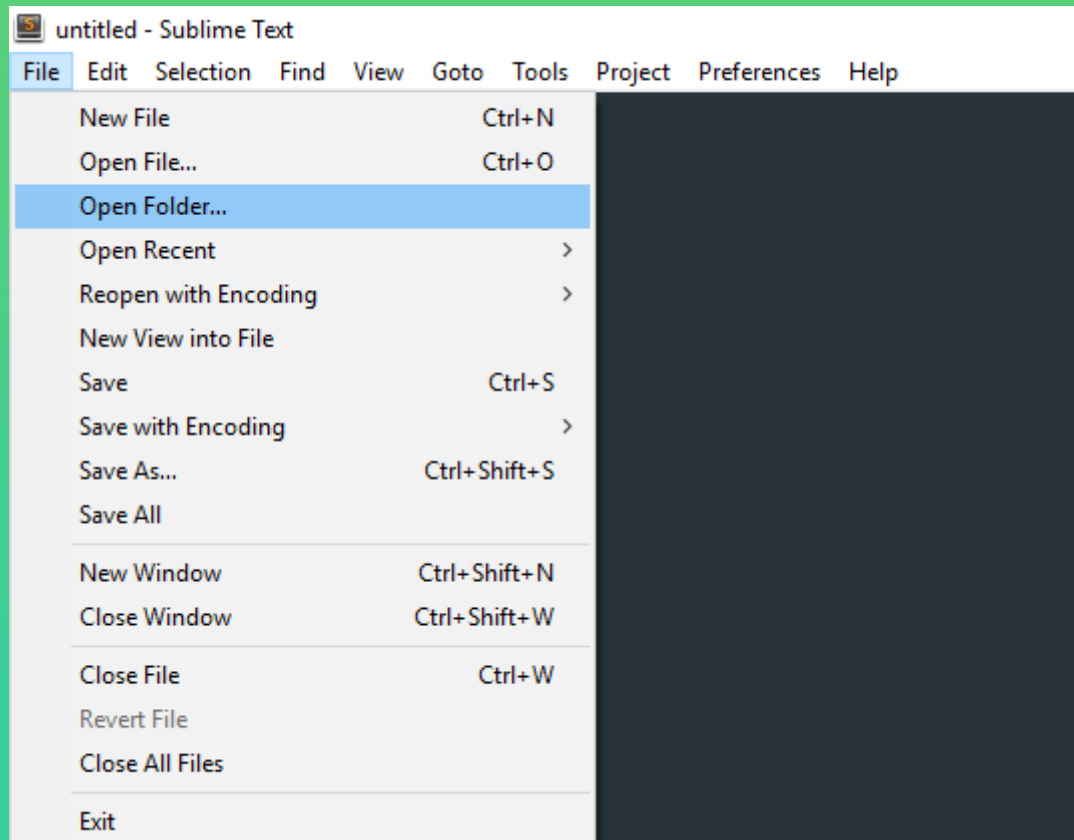
WEBAPP LIKE A BOSS

ESTE É O
LARAVEL




























Usando o Sublime



Selezione o folder

C:\WT-NMP\WWW\laravel

FOLDERS

-  `exemplo`
 -  `app`
 -  `bootstrap`
 -  `config`
 -  `database`
 -  `public`
 -  `resources`
 -  `storage`
 -  `tests`
 -  `vendor`
-  `.env`
-  `.env.example`
-  `.gitattributes`
-  `.gitignore`
-  `artisan`
-  `composer.json`
-  `composer.lock`
-  `gulpfile.js`
-  `package.json`
-  `phpspec.yml`
-  `phpunit.xml`
-  `readme.md`
-  `server.php`



O que são estes diretórios?



/app

Guarda toda a **lógica** do seu projeto quanto ao código PHP.
É nela que você salvará seus **models**!

- **Padrão:** com o nome sempre no singular, teremos um model para cada tabela do banco de dados!

FOLDERS

- exemplo
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Autor.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - artisan
 - composer.json
 - composer.lock
 - autoload.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Autor extends Model

{

//

}



O que são estes diretórios?



/app/Console

Salve aqui seus comandos para o **Artisan**.



Artisan

É uma interface de linha de comando que já vem com o Laravel!
Ele será o seu **melhor amigo** de hoje em diante! :D

C:\WT-NMP\WWW\web-app\



```
$ php artisan
```

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




O que são estes diretórios?



/app/Events

Salvaremos aqui eventos que serão disparados pela framework.



O que são estes diretórios?



/app/Exceptions

Salvaremos aqui exceptions customizadas para o seu sistema.



O que são estes diretórios?



/app/Http

Salvaremos aqui todas as regras dos controllers e como interagem com as rotas do seu web app!



O que são estes diretórios?



/app/Http/routes.php

Este é um arquivo muito especial! Nele você salvará todas as rotas e com qual controller elas conversarão!

FOLDERS

- exemplo
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Controllers
 - Middleware
 - Requests
 - Kernel.php
 - routes.php
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Auth.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - nitattributes

<?php

```
/*  
-----  
Application Routes  
-----  
  
Here is where you can register all of the routes for an application.  
It's a breeze. Simply tell Laravel the URIs it should respond to  
and give it the controller to call when that URI is requested.  
  
*/  
  
Route::get('/', function () {  
    return view('welcome');  
});
```

FOLDERS

- exemplo
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Controllers
 - Middleware
 - Requests
 - Kernel.php
 - routes.php
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Auth.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - nitattributec

```
<!DOCTYPE html>
<html>
  <head>
    <title>Laravel</title>

    <link href="https://fonts.googleapis.com/css?family=Lato:100" rel="stylesheet" type="text/css">

    <style>=
  </style>
</head>
<body>
  <div class="container">
    <div class="content">
      <div class="title">Laravel 5</div>
    </div>
  </div>
</body>
</html>
```

Laravel 5



Seja RESTful!

É um padrão de rotas que alia os verbos HTTP com recursos do site.

VERBO HTTP	ROTA	MÉTODO NO CONTROLLER
GET	/models	Index
GET	/models/create	Create
POST	/models	Store
GET	/models/{id}	Show
GET	/models/{id}/edit	Edit
PUT	/models/{id}	Update
DELETE	/models/{id}	Delete

FOLDERS

- web-app
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Controllers
 - Middleware
 - Requests
 - Kernel.php
 - routes.php
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Model.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - nitattributes

<?php

```
/*
|-----
| Application Routes
|-----
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::get('/models', 'ModelosController@index');
Route::get('/models/create', 'ModelosController@create');
Route::post('/models', 'ModelosController@store');
Route::get('/models/{id}', 'ModelosController@show');
Route::get('/models/{id}/edit', 'ModelosController@edit');
Route::put('/models/{id}', 'ModelosController@update');
Route::delete('/models/{id}', 'ModelosController@delete');
```



O que são estes diretórios?



/app/Http/Controllers

Aqui ficarão os controllers!

- **Padrão:** geralmente faremos um controller para cada model e ele terá o nome [NomeDosModelosController] sempre no plural!



O que são estes diretórios?



/app/Http/Middleware

Salvaremos aqui middlewares, que são nada a mais que filtros de rotas!



O que são estes diretórios?



/app/Http/Kernel.php

Este é um arquivo muito especial! Nele você salvará todas as rotas e com qual controller elas conversarão!

FOLDERS

- web-app
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Controllers
 - Middleware
 - Requests
 - Kernel.php
 - routes.php
 - Jobs
 - Listeners
 - Policies
 - Providers
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - .gitattributes
 - nitianore

<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{

/**

* The application's global HTTP middleware stack.

*

* @var array

*/

protected \$middleware = [
 \Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode::class,

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

];

/**

* The application's route middleware.

*

* @var array

*/

protected \$routeMiddleware = [
 'auth' => \App\Http\Middleware\Authenticate::class,

'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::

class,

'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,

];



O que são estes diretórios?



/app/Http/Request

Salvaremos aqui requisições especiais para criação de modelos!



O que são estes diretórios?



/app/Jobs

Salvaremos aqui procedimentos do nosso sistema que podem ser executados em uma lista de tarefas assíncrona!



Evite lentidão!

Muitas tarefas diárias consomem muito processamento! As **Jobs** são tarefas que executam em paralelo em uma outra thread ou horário.





O que são estes diretórios?



/app/Listeners

Aqui salvaremos listeners, caso seu evento seja tão complexo ao ponto de disparar várias sub-rotinas!



O que são estes diretórios?



/app/Policies

Aqui serão definidas as regras de acesso dos usuários



O que são estes diretórios?



/app/Providers

Aqui ficarão serviços de funcionalidades complexas.
Normalmente são executados antes de ocorrer a request, no boot do kernel do Laravel.



O que são estes diretórios?



/bootstrap

É uma pasta de inicialização do Laravel para manter o padrão de desenvolvimento em comunidade de alto nível!



O que são estes diretórios?



/config

Salvaremos aqui as configurações globais do nosso sistema!



Arquivos de ambiente

Há um arquivo muito especial na root do seu projeto chamado **.env**.

Ele substitui os valores de produção para dev!



Eu tenho **.env**

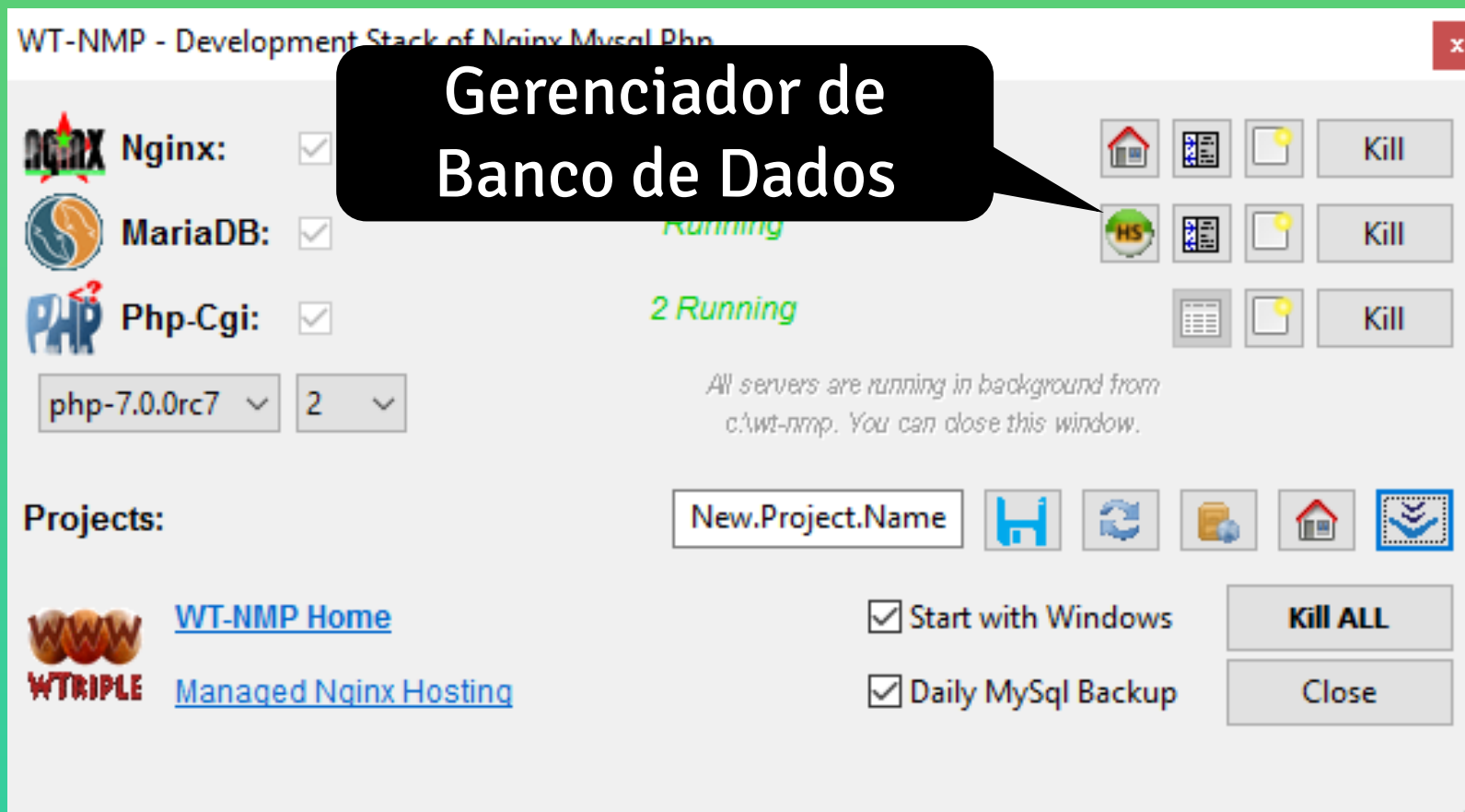
FOLDERS

- exemplo
 - app
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml
- phpunit.xml
- readme.md
- server.php

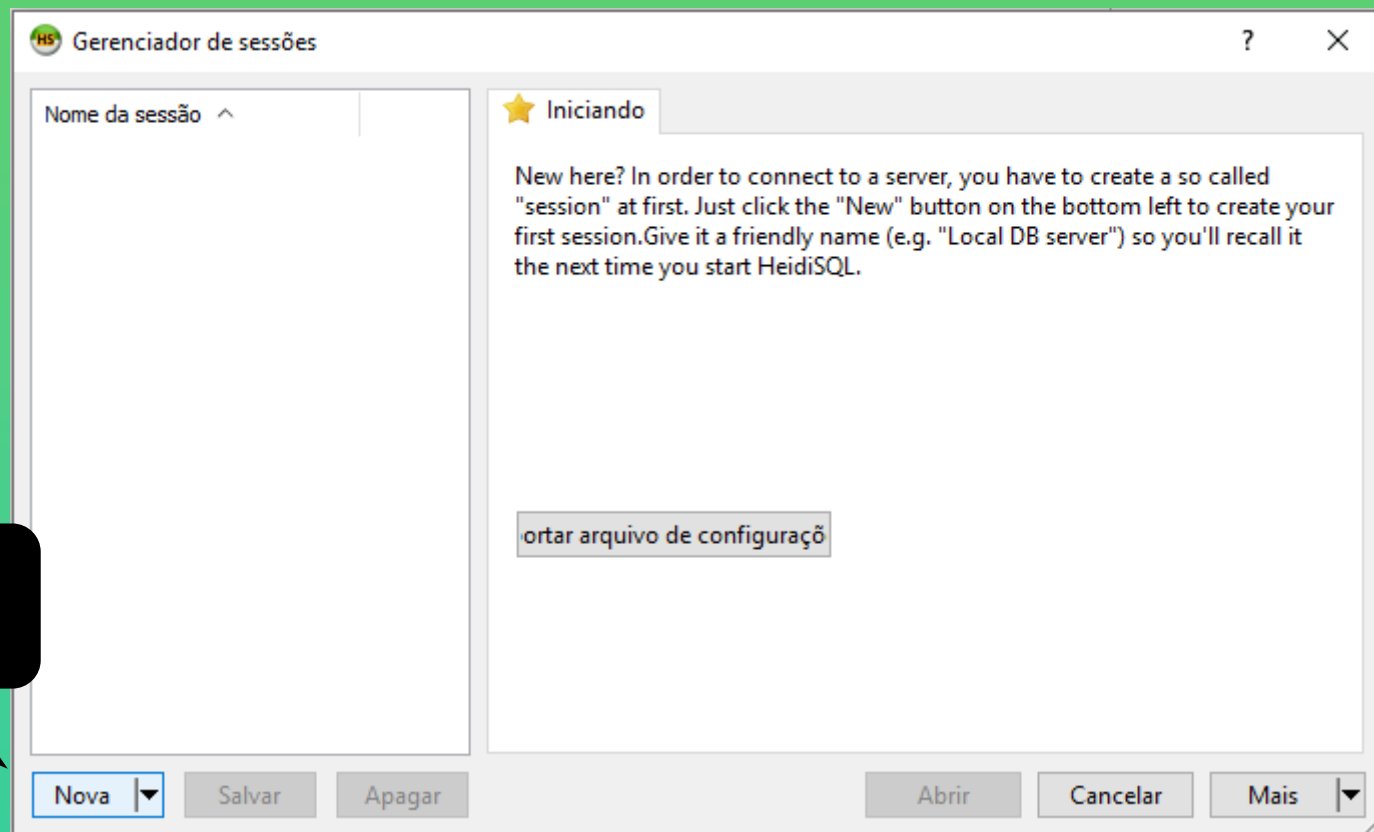
`APP_ENV=local``APP_DEBUG=true``APP_KEY=VnMInI2i7hpQDvATDJ1hGhbQrqvUjVy5``DB_HOST=localhost``DB_DATABASE=exemplo-laravel``DB_USERNAME=root``DB_PASSWORD=``CACHE_DRIVER=file``SESSION_DRIVER=file``QUEUE_DRIVER=sync``MAIL_DRIVER=smtp``MAIL_HOST=mailtrap.io``MAIL_PORT=2525``MAIL_USERNAME=null``MAIL_PASSWORD=null``MAIL_ENCRYPTION=null`



Crie o seu banco de dados



Nova conexão



Localhost

Gerenciador de sessões

Nome da sessão ^

Unnamed

Configurações Avançado Estatísticas

Tipo de rede: MySQL (TCP/IP)

Servidor / IP: 127.0.0.1

☐ Solicitar as credenciais

☐ Usar a autenticação do Windows

Usuário: root

Senha:

Porta: 3306

☐ Protocolo compactado entre cliente e servidor

Bancos de dados: Separados por ponto e vírgula

Comentário:

Nova Salvar Apagar Abrir Cancelar Mais

Crie



Filtro de banco de dados Filtro de tabela

Servidor: localhost Consulta

Localhost

- Editar Alt+Enter
- Excluir...
- Esvaziar tabela(s)... Shift+Del
- Executar rotina(s)...
- Criar novo**
- Limpar dados da aba de filtro
- Exportar banco de dados como SQL
- Manutenção
- Encontrar texto no servidor Shift+Ctrl+F
- Editor de tabela em massa
- Expandir todos
- Recolher todos
- Opções da árvore de estilo
- Imprimir... Ctrl+P
- Atualizar F5
- Desconectar

- Banco de dados**
- Tabela
- Copiar tabela
- View
- Stored Routine
- Trigger
- Evento

Tamanho Itens Últim... Tabel... Views Funç... Proc... Trigg... Event... Colação padrão

```
52 /* CharacterSet: utf8mb4 */
53 SHOW STATUS;
54 SHOW VARIABLES;
55 SHOW DATABASES;
56 /* Entrando na sessão "localhost" */
```



Filtro de banco de dados Filtro de tabela

Localhost

Servidor: localhost

Consulta

Bancos de dados (9)

Variáveis

Status

Processos

Estatísticas de Comandos

Banco de dados ^	Tamanho	Itens	Últim...	Tabel...	Views	Funç...	Proc...	Trigg...	Event...	Colaço padrão
<input type="checkbox"/> creepsheaven-dev										
<input type="checkbox"/> information_schema										
<input type="checkbox"/> migrator_hidoctor										
<input type="checkbox"/> mysql										
<input type="checkbox"/> performance_schema										
<input type="checkbox"/> test										
<input type="checkbox"/> vitta-cep										
<input type="checkbox"/> vitta-core										
<input type="checkbox"/> vitta-migrator										

Criar banco de dados...

Nome: web-app

Colaço: utf8_general_ci

Servidores padrão: utf8_general_ci

Ok

Cancelar

Código CREATE:

CREATE DATABASE `web-app` /*!40100 COLLATE

```
54 SHOW VARIABLES;
55 SHOW DATABASES;
56 /* Entrando na sessão "Localhost" */
57 SHOW COLLATION;
58 SHOW VARIABLES LIKE 'collation_server';
```

FOLDERS

- exemplo
 - app
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml
- phpunit.xml
- readme.md
- server.php

`APP_ENV=local``APP_DEBUG=true``APP_KEY=VnMInI2i7hpQDvATDJ1hGhbQrqvUjVy5``DB_HOST=localhost``DB_DATABASE=exemplo-laravel``DB_USERNAME=root``DB_PASSWORD=``CACHE_DRIVER=file``SESSION_DRIVER=file``QUEUE_DRIVER=sync``MAIL_DRIVER=smtp``MAIL_HOST=mailtrap.io``MAIL_PORT=2525``MAIL_USERNAME=null``MAIL_PASSWORD=null``MAIL_ENCRYPTION=null`

Laravel 5



O que são estes diretórios?



/database

Salvaremos aqui tudo o que for referente a banco de dados!



O que são estes diretórios?



/database/factories

Aqui ficarão receitas de geradores de models para popular nosso banco de dados com informações aleatórias.



O que são estes diretórios?



/database/migrations

Salvaremos aqui as **Migrations**, que são classes que farão manipulações nas nossas tabelas do banco!

FOLDERS

- web-app
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - .gitkeep
 - 2014_10_12_000000_creat
 - 2014_10_12_100000_creat
 - seeds
- .gitignore
- public
- resources
- storage
- tests
- vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password', 60);
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('users');
    }
}
```



Crie a primeira tabela do seu banco!

Voltaremos ao nosso amigo **Artisan!**

Pense no primeiro dado de informação que você terá no seu banco...

C:\WT-NMP\WWW\web-app\



```
$ php artisan make:migration create_models_table
```

FOLDERS

- web-app
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - .gitkeep
 - 2014_10_12_000000_create_users_table.php
 - 2014_10_12_100000_create_password_resets_table.php
 - 2015_11_26_162411_create_models_table.php
 - seeds
 - .gitignore
- public
- resources
- storage
- tests
- vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpcs.xml

<?php

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateModelsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        //
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}
```



Crie a primeira tabela do seu banco!

Para mais informações sobre as criação de colunas, condição de existência, etc:

<http://laravel.com/docs/5.1/migrations>



```
Schema::create('autores', function (Blueprint $table)
{
    $table->increments('id');
    $table->string('nome');
    $table->string('sobrenome');
    $table->string('email')->unique();
    $table->string('foto');
    $table->timestamps();
});
```

C:\WT-NMP\WWW\web-app\app\Http\Controllers\ModelsController.php



```
public function down()  
{  
    Schema::drop('autores ');  
}
```



Composer e PSR-2

No momento que você cria qualquer classe usando o artisan:



/app/Http/Controllers

C:\WT-NMP\WWW\web-app\app\Http\Controlers\ModelsController.php



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




Composer e PSR-2

Migrations e Seeders **não têm namespace**, portanto são adicionados no arquivo **autoload.php** automaticamente que o composer gera!

Caso queria deletar Migrações ou Seeders lembre-se de rodar o seguinte código depois para remove-los do **autoload.php**:

C:\WT-NMP\WWW\web-app\



```
$ composer dump-autoload -o
```



Crie a primeira tabela do seu banco!

C:\WT-NMP\WWW\web-app\



```
$ php artisan migrate
```

executa os todos os up()

```
$ php artisan migrate:rollback
```

executa os todos os down() em ordem inversa



Crie a primeira tabela do seu banco!

Agora vamos criar um Model para poder interagir com o banco de dados através do Laravel!

C:\WT-NMP\WWW\web-app\



```
$ php artisan make:model Model
```

FOLDERS

- web-app
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Model.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - artisan
 - composer.json
 - composer.lock
 - nullfile.ic

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Model extends Model

{

//

}



Crie a primeira tabela do seu banco!

Agora vamos criar um Seeder para popular nosso banco!

C:\WT-NMP\WWW\web-app\



```
$ php artisan make:seeder ModelSeeder
```

FOLDERS

- web-app
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - seeds
 - .gitkeep
 - DatabaseSeeder.php
 - ModelSeeder.php
 - .gitignore
 - public
 - resources
 - storage
 - tests
 - vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml
- phpunit.xml

<?php

```
use Illuminate\Database\Seeder;

class ModelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //
    }
}
```



```
use App\Model;
```

```
...
```

```
for ($i=0; $i < 100; $i++) {  
    Model::create([  
        'coluna1' => 'valor',  
        'coluna2' => 'valor',  
        'coluna3' => 100,  
        'coluna4' => 'valor',  
    ])  
}
```

FOLDERS

- exemplo
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - .gitkeep
 - 2014_10_12_000000_creal
 - 2014_10_12_100000_creal
 - 2015_12_03_170728_creal
 - seeds
 - .gitkeep
 - AutoresTableSeeder.php
 - DatabaseSeeder.php
 - .gitignore
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - artisan
 - composer.json

<?php

use Illuminate\Database\Seeder;

class AutoresTableSeeder extends Seeder

{

/**

public function run()

{

\$faker = Faker\Factory::create();

\$faker->addProvider(new Faker\Provider\Base(\$faker));

\$faker->addProvider(new Faker\Provider\Lorem(\$faker));

\$faker->addProvider(new Faker\Provider\pt_BR\Internet(\$faker));

\$faker->addProvider(new Faker\Provider\pt_BR\Person(\$faker));

\$faker->addProvider(new Faker\Provider\pt_BR\PhoneNumber(\$faker));

for (\$i=0; \$i < 100; \$i++) {

App\Autor::create([

'nome' => \$faker->firstName,

'sobrenome' => \$faker->lastName,

'email' => \$faker->email,

'telefone' => \$faker->cellphoneNumber,

'foto' => 'http://lorempixel.com/400/200/people/' . \$i . '/',

}

}

}

Documentação do Faker:

<https://github.com/fzaninotto/Faker>

FOLDERS

- exemplo
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - seeds
 - .gitkeep
 - AutoresTableSeeder.php
 - DatabaseSeeder.php
 - .gitignore
 - public
 - resources
 - storage
 - tests
 - vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml

<?php

```
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Model::unguard();

        $this->call(AutoresTableSeeder::class);

        Model::reguard();
    }
}
```



Crie a primeira tabela do seu banco!

Agora podemos rodar o Seeder pelo Artisan:

C:\WT-NMP\WWW\web-app\



```
$ php artisan db:seed
```



O que são estes diretórios?



/public

Aqui ficarão todos os arquivos publicos do seu web app!





O que são estes diretórios?



/resources

Aqui ficarão todos os arquivos geradores de compilados frontend além as **views** e **layouts**!



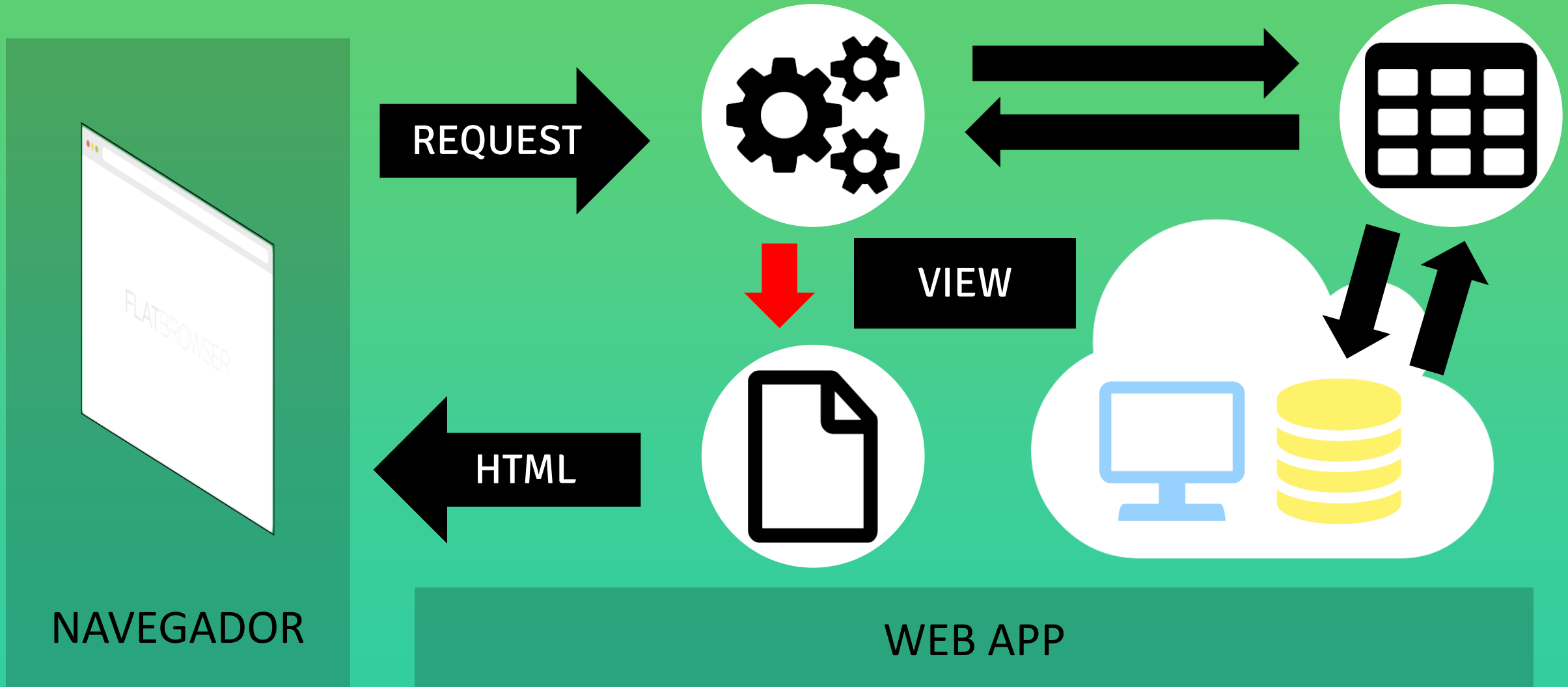
{less}

Sass

FINALMENTE VIEWS



MVC – Model View Controller





Controlers e Views!

Usaremos o Blade, o motor de templates para Laravel

Documentação do Blade

<http://laravel.com/docs/5.1/blade>



/resources/views/layouts



/resources/views/model



```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <title>@yield('title')</title>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
rel="stylesheet" />
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
</head>

<body>
  <div class="container">
    <div class="col-sm-12">
      @yield('content')
    </div>
  </div>
</body>
```





```
@extends('layouts.public')
```

```
@section('title', 'Autores')
```

```
@section('content')
```

```
    <h2>Autores</h2>
```

```
    <hr>
```

```
    @foreach($autores as $autor)
```

```
        <a href="/autores/{{ $autor->id }}">
```

```
            
```

```
        </a>
```

```
    @endforeach
```

```
@stop
```



```
@extends('layouts.public')
```

```
@section('title', 'Autor')
```

```
@section('content')
```

```
<h2>{{ $autor->nome }}</h2>
```

```
<hr>
```

```
<div class="col-md-4">
```

```

```

```
</div>
```

```
<div class="col-md-8">
```

```
<p>{{ $autor->nome }}</p>
```

```
<p>{{ $autor->sobrenome }}</p>
```

```
<p>{{ $autor->telefone }}</p>
```

```
<p>{{ $autor->email }}</p>
```

```
</div>
```

```
@stop
```



Controllers e Views!

Criaremos agora o Controller para interagir com as Rotas e o Model

C:\WT-NMP\WWW\web-app\



```
$ php artisan make:controller ModelsController
```



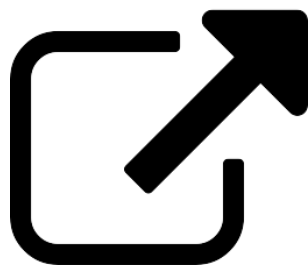
```
public function index()
{
    $autores = Autor::all();
    return view('autores.index')->with('autores',$autores);
}
```

```
public function show($id)
{
    $autor = Autor::findOrFail($id);
    return view('autores.show')->with('autor',$autor);
}
```



Controlers e Views!

Agora visite os links:



webapp.dev/models

webapp.dev/models/1

FOLDERS

- web-app
 - app
 - Console
 - Events
 - Exceptions
 - Http
 - Controllers
 - Middleware
 - Requests
 - Kernel.php
 - routes.php
 - Jobs
 - Listeners
 - Policies
 - Providers
 - Model.php
 - User.php
 - bootstrap
 - config
 - database
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - nitattributes

<?php

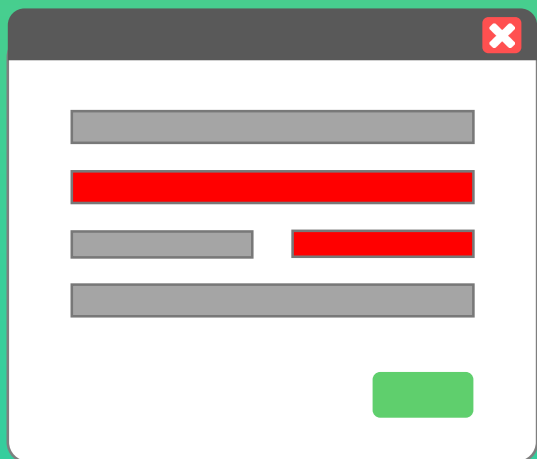
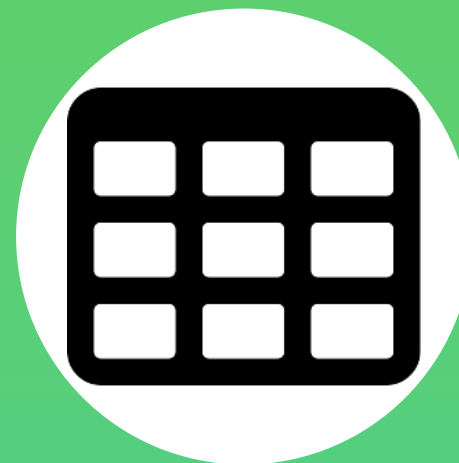
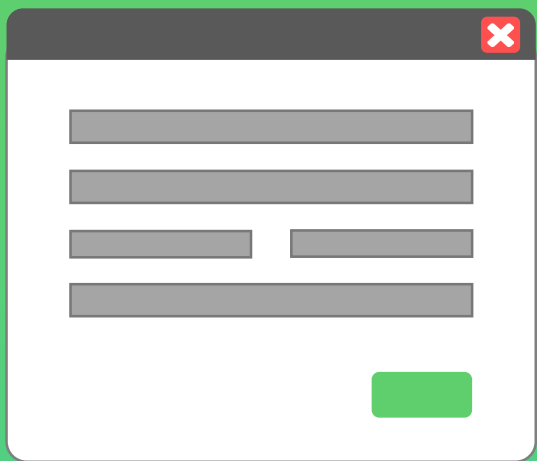
```
/*
|-----
| Application Routes
|-----
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::get('/models', 'ModelosController@index');
Route::get('/models/create', 'ModelosController@create');
Route::post('/models', 'ModelosController@store');
Route::get('/models/{id}', 'ModelosController@show');
Route::get('/models/{id}/edit', 'ModelosController@edit');
Route::put('/models/{id}', 'ModelosController@update');
Route::delete('/models/{id}', 'ModelosController@delete');
```



Agora você!

1. Crie uma **view** para o formulário de **/models/create** method **POST** na rota **/models** para criar um novo element via cadastro;
2. Edite o method do controller **create()** para **retornar** a view do formulário que você criou;
3. Edite o method do controller **store()** para **salvar** o objeto que você acabou de enviar pelo formulário, depois redirecione o usuário para o **GET** do **/models**;





Formulário