



WOKSHOP



TUTORIAL LARAVEL 5.4

WORKSHOP INFOTEC'18

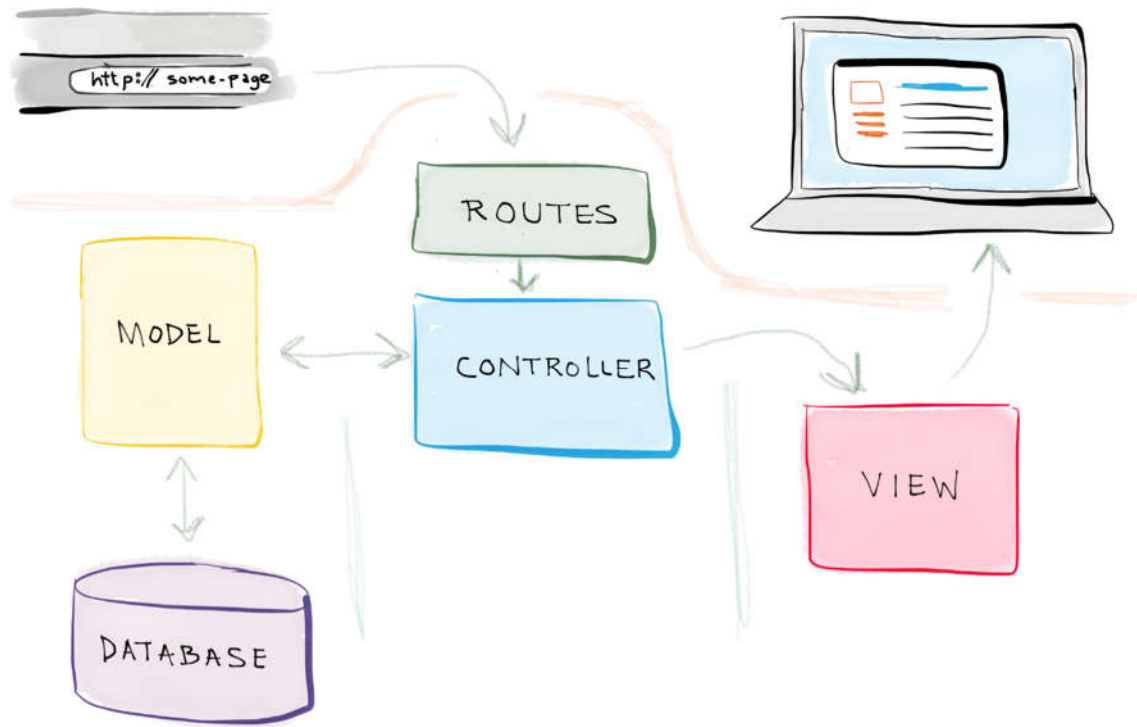
BRUNO MATIAS



## Conteúdo

|                                       |    |
|---------------------------------------|----|
| MVC.....                              | 2  |
| Configuração do ambiente.....         | 3  |
| Instalação <i>Composer</i> .....      | 3  |
| Instalação de Laravel.....            | 8  |
| Configuração .....                    | 10 |
| Configuração .ENV.....                | 11 |
| Corrigir um erro de versão .....      | 11 |
| Criar um sistema de autenticação..... | 12 |
| Instalação do template AdminLTE ..... | 13 |
| Listar itens da base de dados .....   | 15 |
| Instalar DataTables.....              | 20 |
| Seeders.....                          | 22 |
| Perfil do aluno .....                 | 23 |

# MVC



1

<sup>1</sup> <https://selftaughtcoders.com/from-idea-to-launch/lesson-17/laravel-5-mvc-application-in-10-minutes/>

## Configuração do ambiente

**Pré-requisitos:** XAMPP instalado

O *Composer* é uma ferramenta para gestão de dependências para o PHP que tem ganho adeptos a um ritmo alucinante e tem-se tornado cada vez mais indispensável. Com algumas poucas linhas de configurações podes definir todas as bibliotecas de terceiros ou mesmo que desejas/precisas utilizar no teu projeto, o *Composer* encarrega-se de fazer o *download* e criar um *autoloader* deixando-as prontas para utilizares.

Se estás a ponderar começar a usar *frameworks*, então vais habituar-te a usar o *Composer* muitas vezes.

Antes de mais tens de entender o que são dependências. Numa *framework* existe uma série de *lib's* (bibliotecas de código) que contém ferramentas e camadas de abstração que vais usar para aumentar a segurança, a velocidade de desenvolvimento, melhorar a manutenção e tornar o teu código mais "*clean*".

Para teres uma ideia de como vais usar o *Composer* no teu desenvolvimento *Laravel*, pensa que atualmente as *frameworks* web crescem muito rápido, e as suas bibliotecas aos poucos acabam por ficar pesadas e com atualizações frequentes. Gerir a atualização de todas essas diversas bibliotecas seria de loucos, e a pensar na tua preguiça o *Composer* encarrega-se de tudo isso.

Vamos então por mãos à obra e instalar o *Composer*. Aqui mostro-te como é feito para máquinas com sistema operativo *Windows 10*, mas se não for o teu caso, não te preocupes, no site <https://getcomposer.org/doc/00-intro.md> tens a forma de instalação para outros sistemas operativos.

### Instalação *Composer*

Começa por fazer o download do executável <https://getcomposer.org/Composer-Setup.exe>

Clica no executável.

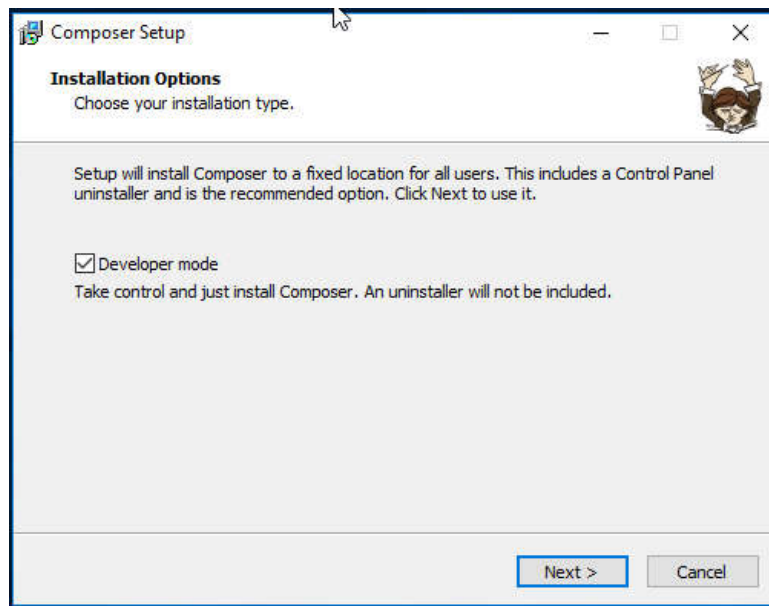


Figura 1 - Primeira janela do Composer

De seguida seleciona a diretoria onde pretendes instalar, sugiro que mantenas a que está por defeito.

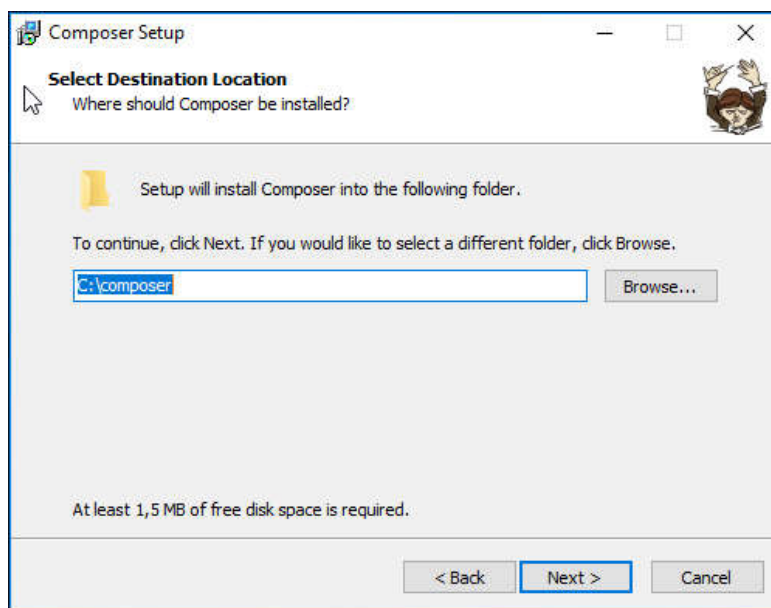


Figura 2 - Segunda Janela Composer

Chega a altura de escolher o ficheiro de *PHP*, em principio debes ter uma opção por defeito, isto se estiver tudo bem com o teu *XAMPP*.

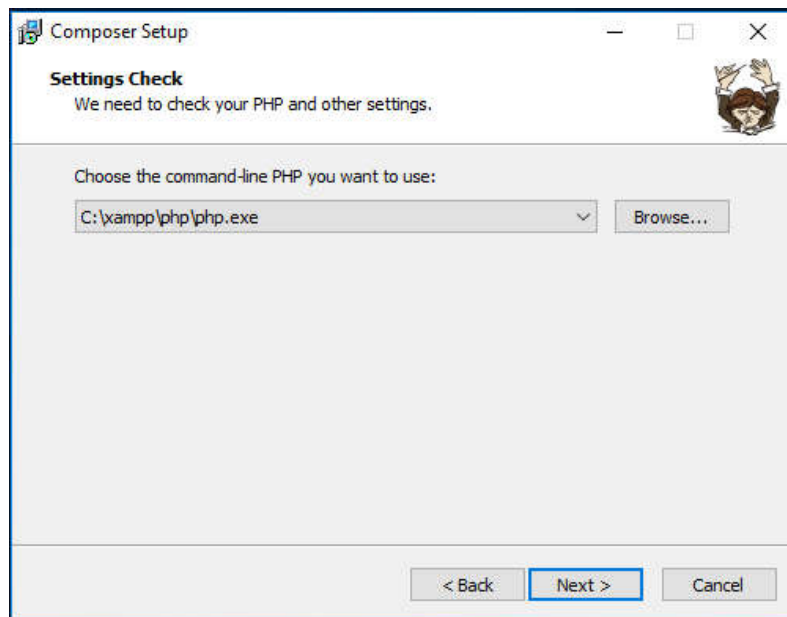


Figura 3 - Terceira Janela Composer

No nosso caso não vamos usar proxy para ligar à internet, no entanto se estiveres a instalar em servidores de um domínio que tenhas um proxy na rede, configura aqui, pois será importante para poderes fazer o download de todos os pacotes corretamente.

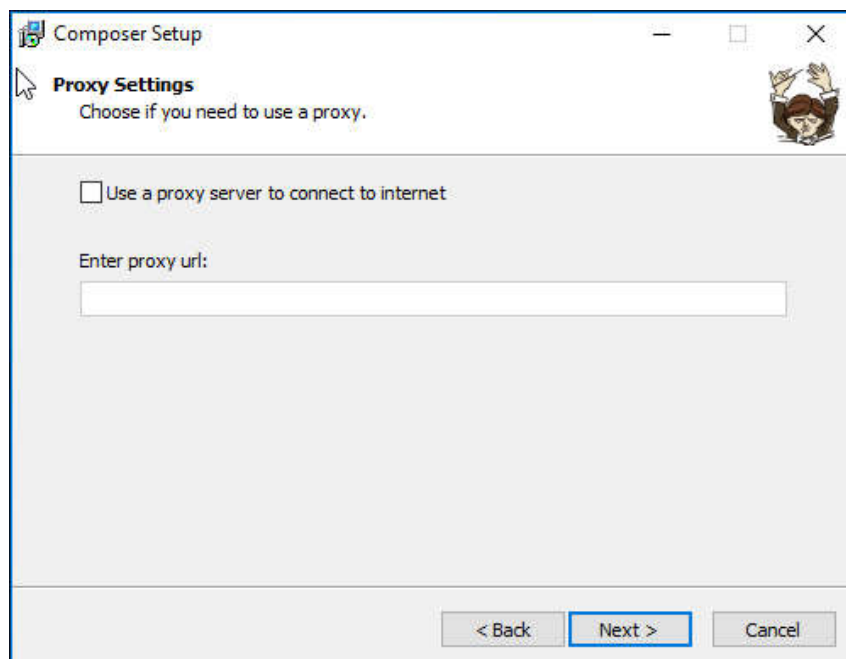


Figura 4 - Quarta Janela Composer

A última janela é um resumo de toda a configuração que acabaste de criar.

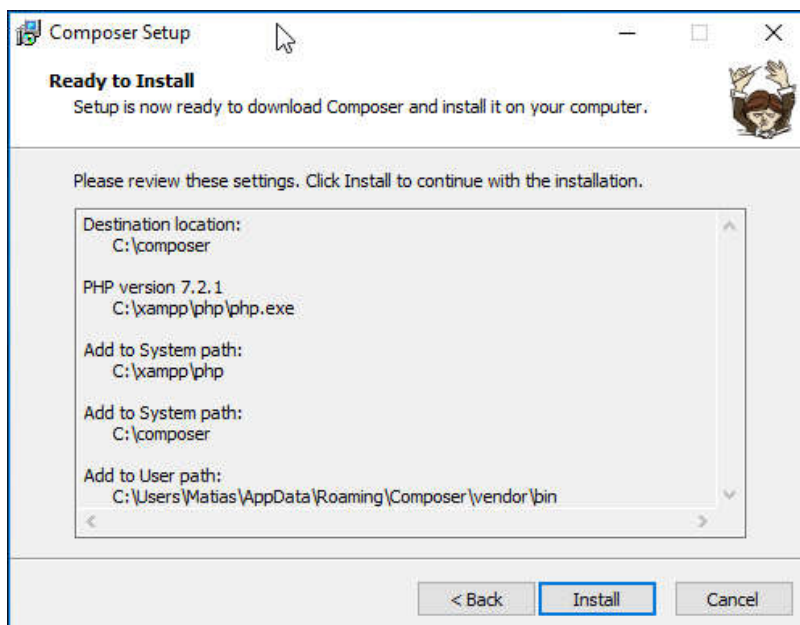


Figura 5 - Quinta Janela Composer

Por fim, vamos instalar. No final vais ter uma janela como a que mostro abaixo, a informar que a instalação foi concluída com sucesso e que tudo o que precisas para começar a trabalhar está no sitio correto bem como as variáveis de ambiente criadas.

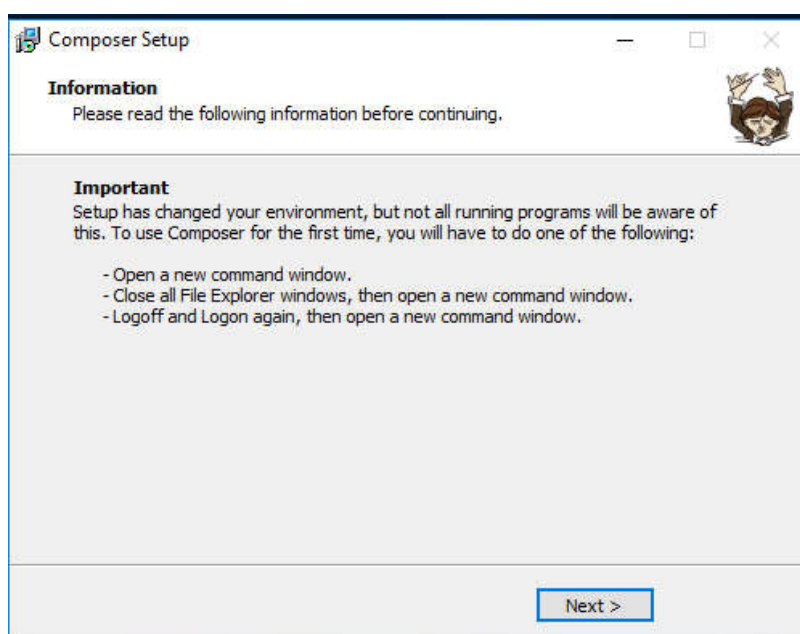


Figura 6 - Sexta Janela Composer

E claro por fim tens a janela final para encerrar o instalador.

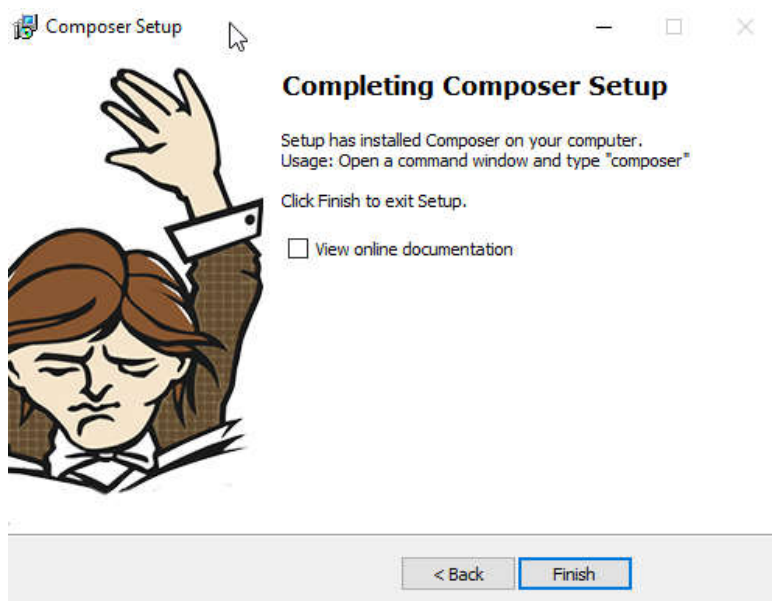


Figura 7 - Setima Janela Composer

Para testar se tudo está corretamente funcional, basta abrir uma linha de comandos e correr o comando **COMPOSER -V**, o resultado é a versão do *Composer* que está instalada.

```
Selecionar Linha de comandos
Microsoft Windows [Version 10.0.16299.15]
(c) 2017 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Matias>composer -V
Composer version 1.6.3 2018-01-31 16:28:17
```

Figura 8 - Teste de instalação do composer e ver a versão instalada

Com este teste, tens também a certeza que vais poder executar comandos *Composer*, isto porque o teu sistema operativo já reconheceu o próprio comando de **composer -V**.

Se o teu sistema operativo Windows for uma atualização de uma versão anterior, tens de reiniciar o computador.



## Instalação de Laravel

Existem diversas versões do *Laravel*, à data da elaboração deste documento a versão mais atual é a 5.6, lançada há poucos dias.

Aqui vou mostrar como instalar a 5.4, duas versões anteriores e a razão disso está relacionada com a dependência de pacotes, ou seja, alguns dos pacotes que vamos encontrar ainda não foram atualizados para as versões mais recentes, salvaguardando assim alguma incompatibilidade.

Começamos por abrir uma janela de linha de comandos, neste caso podes usar a do *Windows* ou usar o botão "Shell" do *XAMPP*.

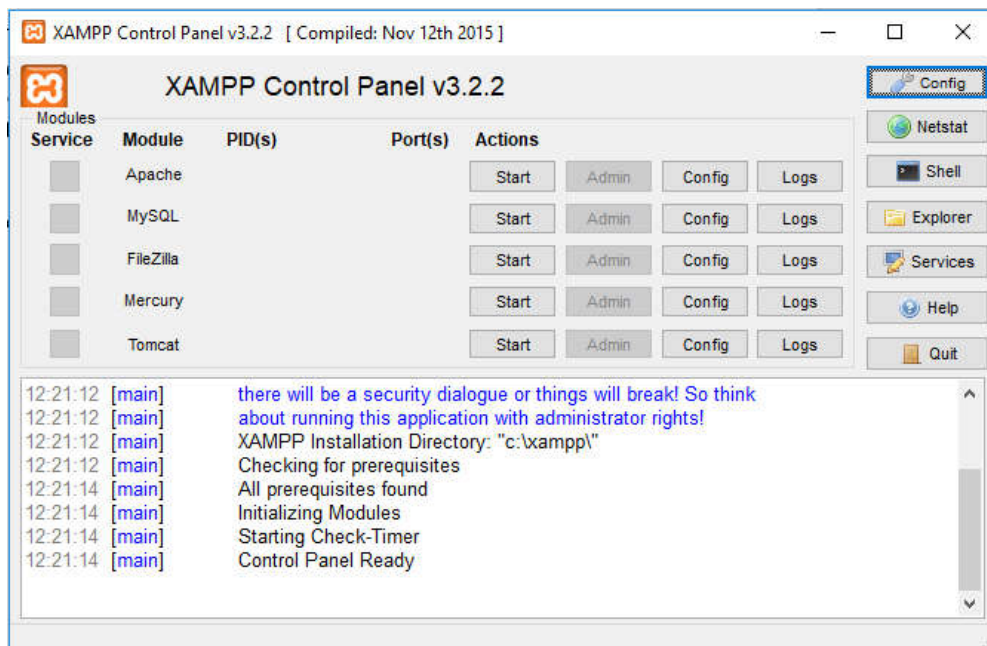


Figura 9 - Painel de controlo do XAMPP

A vantagem de usares a "Shell" do *XAMPP* é que já vai abrir na pasta correspondente onde vamos trabalhar.

A instalação pode ser feita de duas maneiras, ou através do *Laravel Installer* ou via *Composer*. Aqui será demonstrado a forma de instalação usando o *Composer*.

Entramos na diretoria *htdocs*.

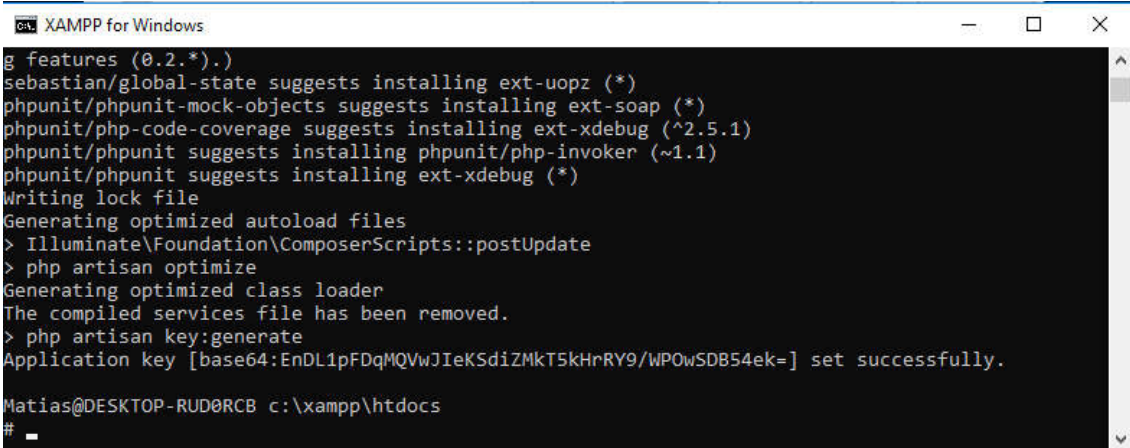
E corremos o comando:

```
composer create-project --prefer-dist laravel/laravel infotec "5.4.*"
```

Este comando vai criar um novo projeto do tipo *Laravel* com o nome *infotec* na versão 5.4.\*.

Não te esqueças que precisas de acesso à *Internet* para fazer isto, o *Composer* vai realizar o *download* de todos os ficheiros necessários, isto vai demorar uns minutos, tudo vai depender da tua máquina e da tua *Internet*.

No final vais ficar com uma janela semelhante à Figura 10 - Fim da instalação do Laravel



```
g features (0.2.*).
sebastian/global-state suggests installing ext-uopz (*)
phpunit/phpunit-mock-objects suggests installing ext-soap (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.5.1)
phpunit/phpunit suggests installing phpunit/php-invoker (~1.1)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postUpdate
> php artisan optimize
Generating optimized class loader
The compiled services file has been removed.
> php artisan key:generate
Application key [base64:EnDL1pFDqMQVwJIeKSdiZMkT5kHrRY9/WPOwSDB54ek=] set successfully.
Matias@DESKTOP-RUD0RCB c:\xampp\htdocs
#
```

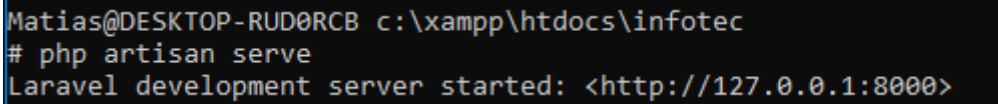
Figura 10 - Fim da instalação do Laravel

Nota que a própria instalação gera uma *Application Key* em base64 para garantir segurança.

Para testar a instalação, ainda na linha de comandos, entramos na diretoria do projeto criado, *infotec* e corremos o comando:

**php artisan serve**

Irás obter uma resposta igual à Figura 11 - Colocar o projeto a correr



```
Matias@DESKTOP-RUD0RCB c:\xampp\htdocs\infotec
# php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
```

Figura 11 - Colocar o projeto a correr

A partir deste momento, tens de manter sempre esta janela aberta (podes minimizar), pois este é o *webserver* embutido que vai correr o projeto. Sim, não precisas de correr o Apache no *XAMPP*.

Nesta janela, conforme o trabalho que fores realizando podes observar que vai existir ação, para já vamos ignorar, isso faz tudo parte do *Laravel*.

No teu browser, vais visitar o endereço devolvido pela linha de comandos, <http://127.0.0.1:8000> ou em alternativa <http://localhost:8000>.

Se tudo correu vai, vais obter o ecrã de boas vindas do *Laravel*.

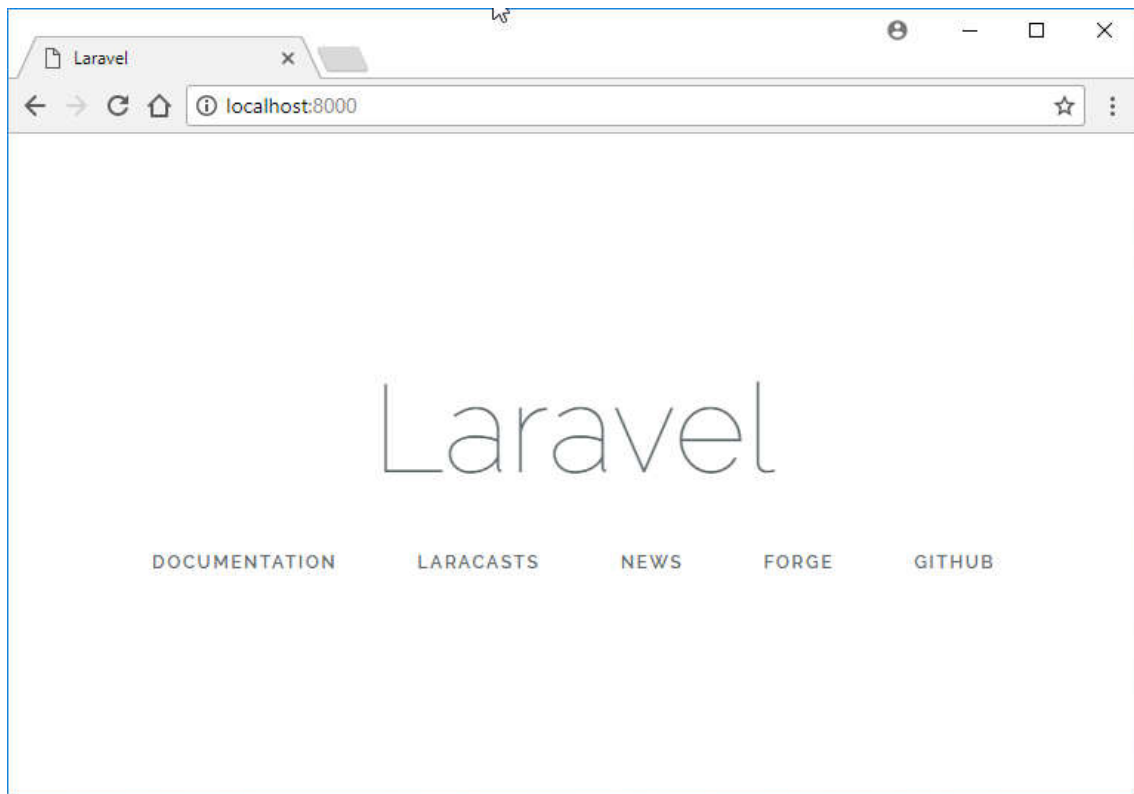


Figura 12 - Ecrã de boas vindas

## Configuração

A partir de agora podes usar um IDE à tua escolha, ou em alternativa um editor de texto.

Todas as configurações gerais são feitas na diretoria “*config*”, e mais à frente vamos falar da organização de ficheiros.

Muitas vezes, é útil ter valores de configuração diferentes com base no ambiente em que a aplicação está a ser executada. Exemplo, podes usar um driver de cache diferente para o ambiente local e outro para o servidor de produção.

Para fazer isso o *Laravel* utiliza a biblioteca *DotEnv PHP*. Sempre que crias uma nova instalação do *Laravel*, a diretoria de raiz contém um ficheiro *.env.example*.

Se a instalação do *Laravel* for realizada via *Composer*, o nome do ficheiro é alterado automaticamente para *.env*.

## Configuração .ENV

No ficheiro `.env` encontras a configuração para a base de dados:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Bem como outro tipo de configurações como por exemplo o envio de e-mail.

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Para já vamos configurar apenas a ligação à base de dados. Deves colocar a correr o servidor *MySQL* do *XAMPP*, e podes arrancar também com o *Apache* que nos vai permitir aceder temporariamente ao *phpmyadmin* para criar a base de dados.

Acede ao *phpmyadmin* (em `localhost/phpmyadmin`) e cria a base de dados *infotec*.

Volta ao teu ficheiro `.env` e preenche com os campos correctos. Deixo o exemplo do nosso caso.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=infotec
DB_USERNAME=root
DB_PASSWORD=
```

Corrigir um erro de versão.

Nem tudo são rosas, e nesta versão existe um erro que, entretanto, já foi reportado e parece que vai ser corrigido nas novas versões.

Acede ao ficheiro *AppServiceProvider.php* que encontras na diretoria do teu projeto em `infotec/app/Providers`

Adiciona ao *public function boot()* a seguinte linha

```
Schema::defaultStringLength(191);
```

Não esquecendo de colocar no topo o *import*

```
use Illuminate\Support\Facades\Schema;
```

## Criar um sistema de autenticação

Uma das coisas que vai perceber em *Laravel* é que em muitas coisas vais usar a linha de comandos. Não te assustes, pode parecer complicado, mas isso vai facilitar em muito a tua vidinha.

Vamos pensar que queres ter um site com *frontend* e uma área de *backend* para gerir alguma coisa.

Por exemplo que no teu *backend* vais gerir as pessoas inscritas neste workshop.

Para isso precisas de um sistema de autenticação com registo e login. Vamos então por mão à obra, ou melhor ao comando.

Atenção não feches a janela que tens aberta, pois essa está a manter o teu servidor *Laravel* a correr.

Numa nova linha de comandos, entra no htdocs, e depois na pasta de projeto e corre o comando:

```
php artisan make:auth
```

Depois disso, corre o comando:

```
php artisan migrate
```

O *migrate* vai migrar a base de dados para o sistema de autenticação. Vão ser criadas todas as tabelas necessárias, controladores, *models* e vistas.

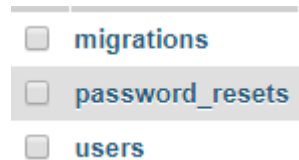


Figura 13 - Tabelas criadas com migrate

Posto isto, vamos ao browser e fazemos refresh à pagina e deves ver atualmente as opções para registo e login.

LOGIN

REGISTER

# Laravel

DOCUMENTATION

LARACASTS

NEWS

FORGE

GITHUB

Figura 14 - Sistema de autenticação

Podes agora fazer o teu registo de utilizador, se obtiveres algum erro, tenta reiniciar o servidor fechando a janela de comandos e abrindo novamente e correndo o comando `php artisan serve`.

Atualiza o browser e irás verificar que imediatamente o teu registo na base de dados é feito e és redirecionado para a página de dashboard.

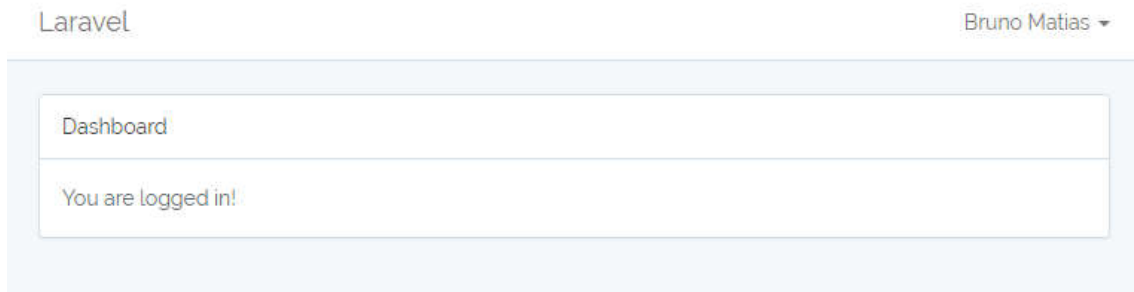


Figura 15 – Dashboard

Como viste é simples, se fores artista, podes começar a desenhar a tua aplicação junto com um layout apelativo e cheio de usabilidade. Caso a tua onda seja mais *backend*, podes optar por escolher um *template* já disponível e prontinho a usar. A seguir mostro-te como usar o famoso *AdminLTE*.

#### Instalação do template AdminLTE

O pacote mais famoso de *AdminLTE* já bem estruturado para *Laravel* é o *Acacha* (<https://github.com/acacha/adminlte-laravel>).

No link acima encontras todos os detalhes para a instalação. Mas não stresses que vou aqui deixar também o passo-a-passo para ti.

Voltamos à linha de comandos e vamos pedir ao *Composer* que trate da instalação de pacotes, usando o comando para a nossa versão de *Laravel* 5.4:

```
composer require "acacha/admin-lte-template-laravel:4.*"
```

Depois de concluído, precisamos registar o *Service Provider*, para isso vais ao teu projeto alteramos o ficheiro `config/app.php`

Adiciona a linha seguinte no final de todos os *providers*:

```
/*
 * Acacha AdminLTE template provider
 */
Acacha\AdminLTETemplateLaravel\Providers\AdminLTETemplateServiceProvider::class,
```

No mesmo ficheiro, mais abaixo fazemos o mesmo para o *ALIAS*

```
/*  
 * Acacha AdminLTE template alias  
 */  
'AdminLTE' => Acacha\AdminLTETemplateLaravel\Facades\AdminLTE::class,
```

Por fim, voltamos à linha de comandos e fazemos a publicação de todos os registos que efetuamos.

```
php artisan vendor:publish --tag=adminlte --force
```

Voltamos ao browser e fazemos *refresh*. Deves obter um resultado idêntico à imagem abaixo, já com o *template* de *AdminLTE*

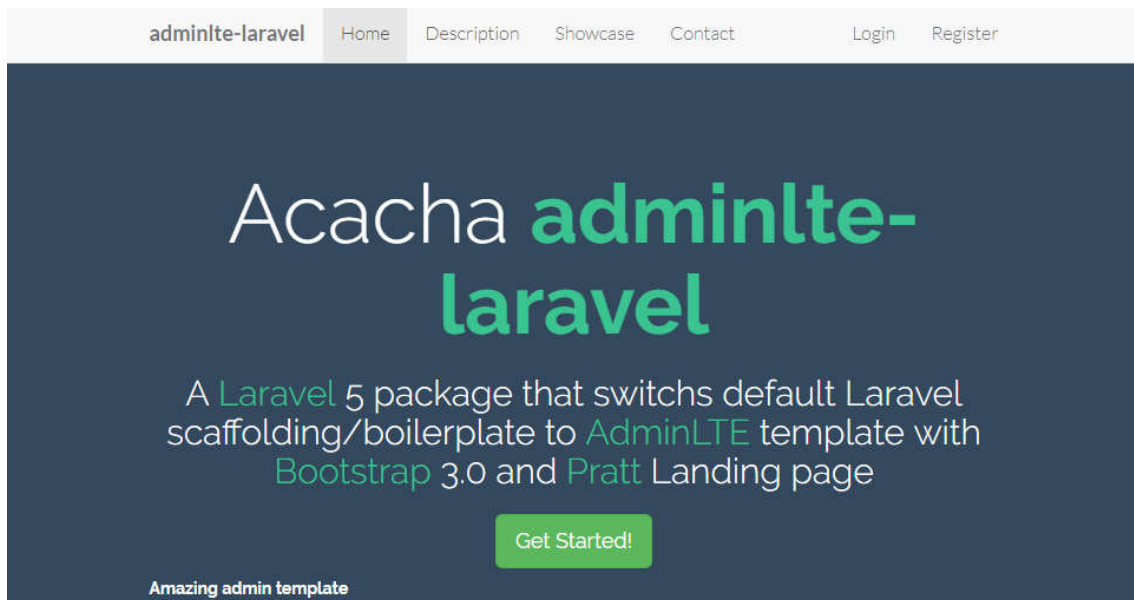


Figura 16 - Acacha – AdminLTE

Tenta fazer login novamente com as credenciais do utilizador que crias-te anteriormente e deves entrar no *dashboard* do *AdminLTE*.

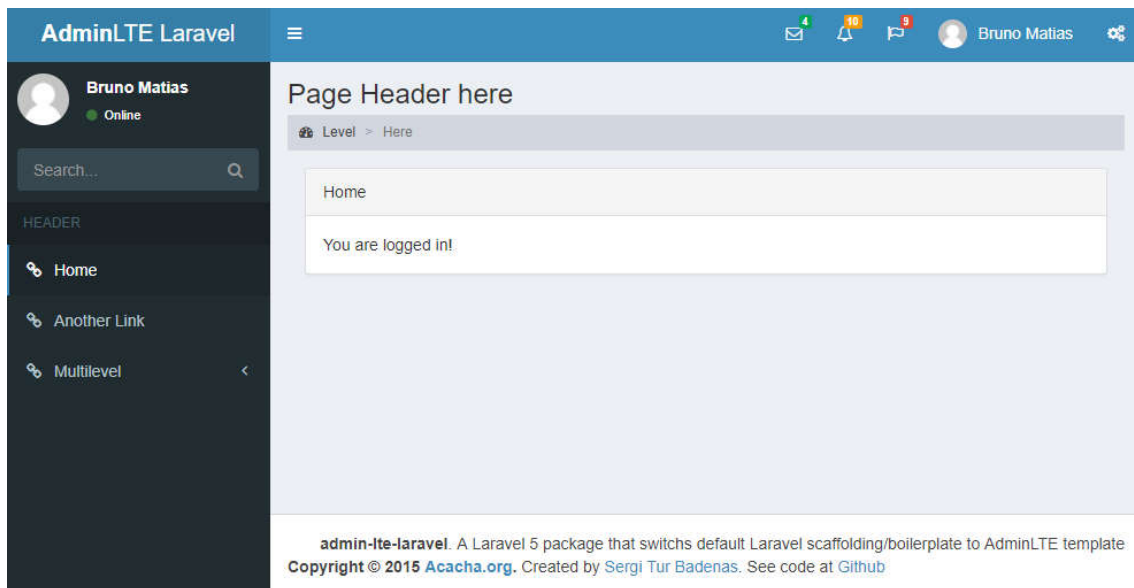


Figura 17 - Dashboard Acacha – AdminLTE

Listar items da base de dados

Vamos agora criar um item do menu lateral e preparar para ter um *CRUD* de alunos inscritos num workshop no *Infotec*.

Para isso vamos aprender como criar tabelas e relacionamentos com o *Laravel*.

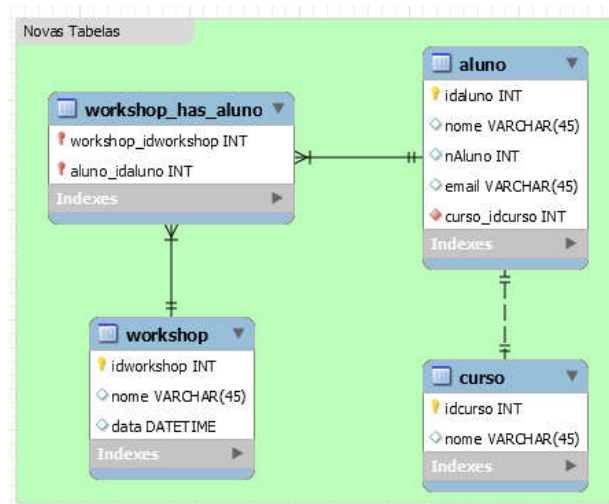


Figura 18 - Primeira versão das tabelas a construir

Voltamos à nossa linha de comandos, sempre dentro da pasta do projeto e vamos criar a primeira migração.

```
php artisan make:migration create_curso_table
```

Isto vai criar a tabela de curso no *Laravel* e prepará-la para ser migrada para a base de dados. NOTA: Este comando só por si não cria a tabela na base de dados, mas sim um ficheiro que vamos configurar para criar a tabela como desejamos.



Depois de correr o comando, vamos abrir o ficheiro que se encontra em *infotec/database/migrations/2018\_03\_13\_XXXXX\_create\_curso\_table.php*

O ficheiro criado tem esta denominação para nos ajudar no futuro, este *timestamp* permite mais tarde se necessário realizar um *rollback* e voltar às origens se alguma coisa der barracada.

Editamos o ficheiro e adicionamos:

```
public function up()
{
    Schema::create('curso', function (Blueprint $table) {
        $table->increments('id');
        $table->string('nome');
    });
}
```

Voltamos a fazer o mesmo para criar a tabela de aluno

```
php artisan make:migration create_aluno_table
```

e adicionamos ao ficheiro

```
public function up()
{
    Schema::create('aluno', function (Blueprint $table) {
        $table->increments('id');
        $table->string('nome');
        $table->string('nAluno');
        $table->string('email');

        $table->integer('curso_id')->unsigned();

        $table->foreign('curso_id')
            ->references('id')->on('curso')
            ->onDelete('cascade');
    });
}
```

Como podes ver, criamos aqui a relação existente entre as duas tabelas.

Agora que viste como podemos criar tabelas, vamos seguir em frente e mais tarde podemos criar tabelas e relações sempre que precisamos, ou atualizar campos.

Se fores verificar à tua base de dados ainda não tens lá nada, por isso vamos dizer ao *Laravel* para “oficializar as coisas” e migrar tudo para a base de dados.

Na linha de comandos:

```
php artisan migrate
```

e deves obter um resultado semelhante à imagem seguinte.

```
C:\xampp\htdocs\infotec>php artisan migrate
Migrating: 2018_03_03_213132_create_curso_table
Migrated: 2018_03_03_213132_create_curso_table
Migrating: 2018_03_03_213804_create_aluno_table
Migrated: 2018_03_03_213804_create_aluno_table
```

Figura 19 - migração de tabelas

Se fores ver o resultado ou em *MySQL Workbench* ou no *phpmyadmin* tens algo como

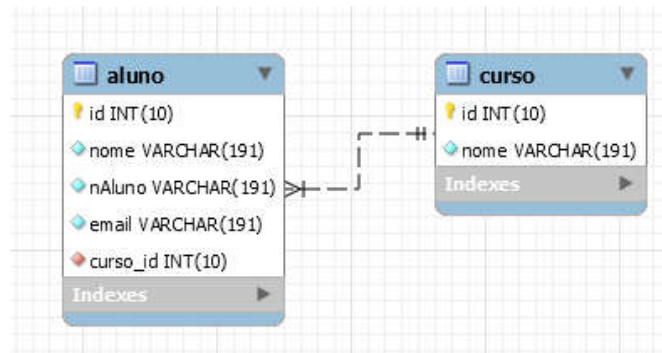


Figura 20 - tabelas criadas com o migration

Vamos então arregaçar as mangas e por mãos à obra ao código.

Novamente na linha de comandos vamos criar a *model* para identificar a tabela da base de dados que vamos usar.

```
php artisan make:model Aluno
```

As *models* são criadas na pasta *infotec/app*

Dentro da *model* coloca o seguinte código:

```
class Aluno extends Model
{
    protected $table = 'aluno';
}
```

Precisas agora de criar um controlador:

```
php artisan make:controller AlunoController --resource
```

O controlador foi criado na diretoria *infotec/app/Http/Controllers*

Se reparares bem dentro do *controller* já tens criadas as funções `index()`; `create()`; `store()`; `show()`; `edit()`; `update()`; `destroy()`;

Vamos começar por trabalhar o `index()` que vai devolver todos os registos da tabela de alunos para a nossa *view* que vai listar os registos.

O que basicamente é colocar:

```
public function index()
{
    return view('alunos.index');
}
```

Vamos agora tratar das *views*, e vamos começar por editar o ficheiro do menu lateral para adicionar um novo item de Alunos.

Não te assustes com o caminho do ficheiro, na realidade vais perceber que se torna prático dividir o *template* em partes, é mais fácil futuramente para modificações.

Acede a `/infotec/resources/views/vendor/adminlte/layouts/partials/sidebar.blade.php`

E aqui vais editar um html comum.

Exemplo do código que coloquei.

```
<li class="treeview">
  <a href="#">
    <i class='fa fa-user'></i>
    <span>{{ trans('adminlte_lang::message.gestao') }}</span>

    <i class="fa fa-angle-left pull-right"></i>
  </a>
  <ul class="treeview-menu">
    <li>
      <a href="{{ url('gestao/alunos') }}">{{
        trans('adminlte_lang::message.listar') }}</a>
    </li>
  </ul>
</li>
```

Assinalado a **amarelo** está a forma de realizar traduções. Para isso basta ir à diretoria e alterar o ficheiro: `infotec/vendor/acacha/admin-lte-template-laravel/resources/lang/en/message.php`

E adicionar à lista que já lá encontras as mensagens que queres traduzir.

```
'listar'          => 'Listar alunos',
'gestao'          => 'Gestão',
```

Escolhi a diretoria EN, porque não alterei a linguagem `infotec/config/app.php`

Assinalado a **verde**, tens a rota, ou seja, o caminho que indicamos para chegar à página que vai listar os dados.

Para isso vamos ao ficheiro `infotec/routes/web.php`

E adiciona: `Route::resource('gestao/alunos', 'AlunoController');`

Depois disto vamos necessitar de criar a página onde vamos mostrar a informação. Para isso na diretoria `infotec/resources/views/`

Cria a pasta `alunos` e lá dentro o ficheiro `index.blade.php`

Coloca o seguinte código no ficheiro que acabaste de criar.

```
@extends('adminlte::layouts.app')
@section('contentheader_title')
    Gestão de Alunos
@endsection
@section('main-content')
    <div style="float:right;">
        <a href="{{ URL::to('gestao/alunos/create') }}">
            <button type="button" class="btn btn-block btn-primary">
                <i class="fa fa-plus"></i> Adicionar Aluno
            </button>
        </a><br><br>
    </div>
    <br><br>
    <div class="box box-primary">
        <div class="row">
            <div class="col-sm-12">
                <div class="box-body">
                    <table class="table table-hover table-condensed"
                    style="width:100%" id="alunos-table">
                        <thead>
                            <tr>
                                <th>Número</th>
                                <th>Nome</th>
                                <th>Ações</th>
                            </tr>
                        </thead>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

E vais obter um resultado como o da imagem:



Figura 21 - listagem de alunos

Ainda não aparece nenhuma informação porque ainda não definimos nada para mostrar dados. Para isso vamos usar *dataTables* que vai permitir adicionar dinamismo à nossa aplicação.

#### Instalar DataTables

Para instalar o *jQuery* de *DataTables* vamos recorrer ao nosso gestor de pacotes, para nos facilitar tudo, e vamos usar o pacote *yajra* disponível no *github* em <https://github.com/yajra/laravel-datatables>. E para facilitar a nossa vida, o autor ainda mostra como devemos de instalar. Vou replicar aqui a explicação.

Correr o comando:

```
composer require yajra/laravel-datatables-oracle:"~8.0"
```

Quando o teu *composer* terminar de atualizar, seguimos com a configuração.

Atualizar os *providers* e *aliases* no ficheiro *infotec/config/app.php*

```
'providers' => [
    ...,
    Yajra\DataTables\DataTablesServiceProvider::class,
]

'aliases' => [
    ...,
    'DataTables' => Yajra\DataTables\Facades\DataTables::class,
]
```

E por fim fazemos a publicação na linha de comandos

```
php artisan vendor:publish --provider=Yajra\DataTables\DataTablesServiceProvider
```

No *controller* aluno adicionamos:

```
public function getAlunos()
{
    return Datatables::of(Aluno::query())
        ->addColumn('action', function ($aluno) {
            return '<a href="alunos/' . $aluno->id . '/edit" class="btn btn-
primary tooltips"><i class="fa fa-pencil fa fa-white"></i></a>';
        })
        ->rawColumns(['estado', 'action'])
        ->make(true);
}
```

Não esquecendo de incluir no topo

```
use App\Aluno;
use Yajra\DataTables\DataTables;
```

Temos de adicionar à rota, o código seguinte, para conseguir aceder à função.

```
Route::get('gestao/alunos/getAlunos', 'AlunoController@getAlunos')
->name('alunos.getAlunos');
```

Na view (index.blade.php) adicionar ao final o script

```
@push('scriptDatatables')
<script>
    $(function() {
        $('#alunos-table').DataTable({
            language: {
                url: '//cdn.datatables.net/plug-
ins/1.10.16/i18n/Portuguese.json'
            },
            processing: true,
            serverSide: true,
            ajax: '{!! route('alunos.getAlunos') !!}',
            columns: [
                { data: 'nome', name: 'nome' },
                { data: 'nAluno', name: 'nAluno' },
                { data: 'action', name: 'action', orderable: false,
searchable: false}
            ]
        });
    });
</script>
@endpush

</div>
```

Para ativar o script temos de adicionar à lista de scripts em  
infotec/resources/views/vendor/adminlte/layouts/partial/scripts.blade.php

```
<script
src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></script>
```

```
...
@stack('scriptDatatables')
```

Adicionar em  
infotec/resources/views/vendor/adminlte/layouts/partial/htmlheader.blade.php

```
<link rel="stylesheet"
href="//cdn.datatables.net/1.10.7/css/jquery.dataTables.min.css">
```

Vamos testar e o resultado que deves obter é:

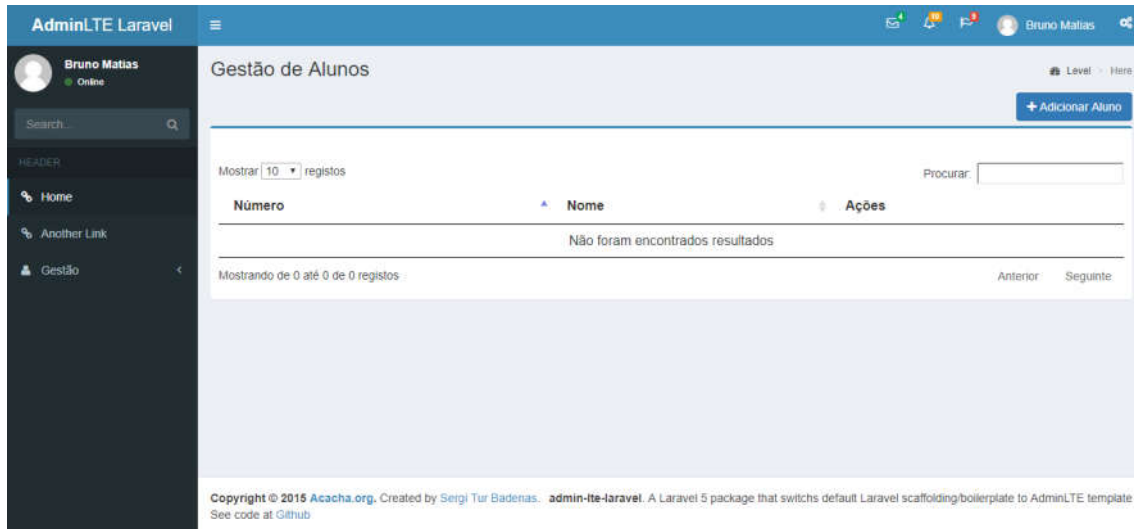


Figura 22 - Lista de alunos com DataTables

No entanto podes ver que não temos dados a serem listados, isto porque a tabela está vazia. Tens duas opções, ou vais popular a tabela manualmente no *phpmyadmin*, ou usamos outro recurso do *Laravel* para popular com dados aleatórios para testes.

## Seeders

Vamos aqui popular a tabela de alunos.  
Na linha de comandos corremos:

```
php artisan make:seeder AlunoTableDataSeeder
```

se fores ao ficheiro `infotec/database/seeds/AlunoTableDataSeeder.php` colocamos:

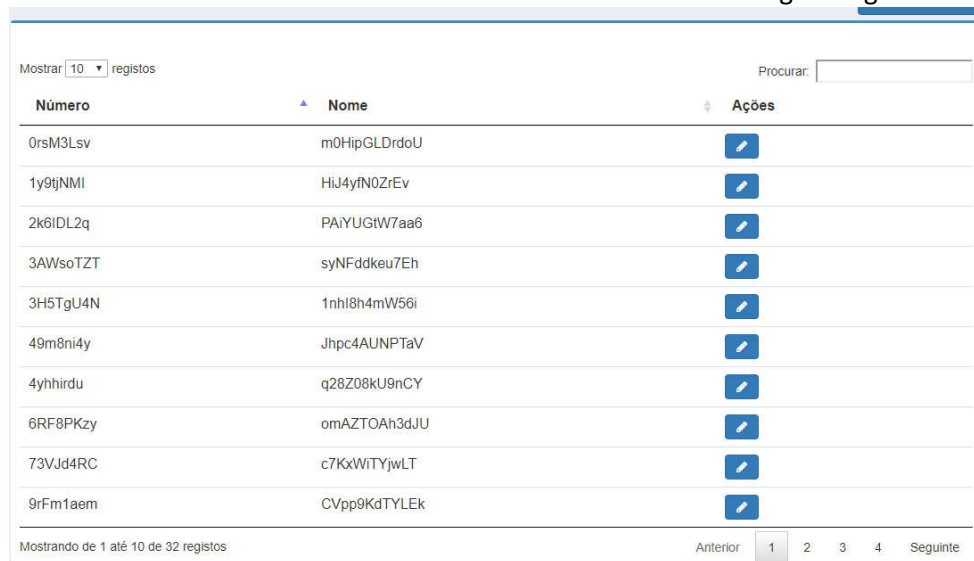
```
public function run()
{
    for ($i=0; $i < 32; $i++) {
        DB::table('aluno')->insert([
            'nome' => str_random(8),
            'nAluno' => str_random(12),
            'email' => str_random(12).'@mail.com',
            'curso_id' => '1'
        ]);
    }
}
```

Aqui parto do principio que já existe um curso na tabela de cursos, com o id=1

E por fim corremos o comando

```
php artisan db:seed --class=AlunoTableDataSeeder
```

E depois disso deves ter um resultado no browser do semelhante à imagem seguinte:













| Número   | Nome         | Ações                                                                               |
|----------|--------------|-------------------------------------------------------------------------------------|
| 0rsM3Lsv | m0HipGLDrdoU |  |
| 1y9tjNMI | HiJ4yiN0ZrEv |  |
| 2k6IDL2q | PAiYUGtW7aa6 |  |
| 3AWsoTZT | syNFddkeu7Eh |  |
| 3H5TgU4N | 1nhl8h4mW56i |  |
| 49m8ni4y | Jhpc4AUNPTaV |  |
| 4yhhirdu | q28Z08kU9nCY |  |
| 6RF8PKzy | omAZTOAh3dJU |  |
| 73VJd4RC | c7KxWtYjwLT  |  |
| 9rFm1aem | CVpp9KdTYLEk |  |

Figura 23 - Datatables de alunos a mostrar resultados

## Perfil do aluno

Vamos focar agora em realizar um READ à base de dados para mostrar os dados do aluno. Quando clicares no botão de show que mostrará os compôs, mas permitirá também fazer o edit futuramente.

Para isso temos de definir a página que irá mostrar a informação e a rota para essa página.

Mãos à obra e criamos a página do perfil. Como estamos a usar o AdminLTE, se visitares a página do template podes copiar o código HTML com o perfil já desenhado, em <https://adminlte.io/themes/AdminLTE/pages/examples/profile.html>, basto-nos adaptar ao que precisamos. Guarda este código, vais precisar dele. (Eu vou colocar neste documento mais à frente)

Vamos ao controlador do aluno em *infotec/app/Http/Controllers/AlunoController.php* e na função *edit* :

```
public function edit($id)
{
    return view('alunos.edit', ['aluno' => Aluno::findOrFail($id)]);
}
```

Em *infotec/resources/views/alunos* cria o ficheiro *edit.blade.php* e nesse ficheiro copia o código seguinte para lá.

```
@extends('adminlte::layouts.app')
@section('contentheader_title')
    Perfil do aluno
@endsection
```



```

@section('main-content')
<section class="content">
  <div class="row">
    <div class="col-md-3">
      <!-- Profile Image -->
      <div class="box box-primary">
        <div class="box-body box-profile">
          
          <h3 class="profile-username text-center">{{ $aluno->nome }}</h3>
          <p class="text-muted text-center">{{ $aluno->email }}</p>
          <ul class="list-group list-group-unbordered">
            <li class="list-group-item">
              <b>#</b> <a class="pull-right">{{ $aluno->nAluno }}</a>
            </li>
            <li class="list-group-item">
              <b>Following</b> <a class="pull-right">543</a>
            </li>
            <li class="list-group-item">
              <b>Friends</b> <a class="pull-right">13,287</a>
            </li>
          </ul>
          <a href="#" class="btn btn-primary btn-block"><b>Follow</b></a>
        </div>
      <!-- /.box-body -->
    </div>
    <!-- /.box -->
    <!-- About Me Box -->
    <div class="box box-primary">
      <div class="box-header with-border">
        <h3 class="box-title">About Me</h3>
      </div>
      <!-- /.box-header -->
      <div class="box-body">
        <strong><i class="fa fa-book margin-r-5"></i> Education</strong>
        <p class="text-muted">
          B.S. in Computer Science from the University of Tennessee at
          Knoxville
        </p>
        <hr>
        <strong><i class="fa fa-map-marker margin-r-5"></i>
          Location</strong>
        <p class="text-muted">Malibu, California</p>
        <hr>
        <strong><i class="fa fa-pencil margin-r-5"></i> Skills</strong>
        <p>
          <span class="label label-danger">UI Design</span>
          <span class="label label-success">Coding</span>
          <span class="label label-info">Javascript</span>
          <span class="label label-warning">PHP</span>
          <span class="label label-primary">Node.js</span>
        </p>
        <hr>
        <strong><i class="fa fa-file-text-o margin-r-5"></i>
          Notes</strong>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
          fermentum enim neque.</p>
      </div>
    <!-- /.box-body -->
  </div>
  <!-- /.box -->
</div>
<!-- /.col -->
<div class="col-md-9">
  <div class="nav-tabs-custom">

```

```

        <ul class="nav nav-tabs">
            <li class="active"><a href="#activity" data-
toggle="tab">Activity</a></li>
            <li><a href="#timeline" data-toggle="tab">Timeline</a></li>
            <li><a href="#settings" data-toggle="tab">Settings</a></li>
        </ul>
        <div class="tab-content">
            <div class="active tab-pane" id="activity">
                Activity
            </div>
            <!-- /.tab-pane -->
            <div class="tab-pane" id="timeline">
                Timeline
            </div>
            <!-- /.tab-pane -->
            <div class="tab-pane" id="settings">
                Settings
            </div>
            <!-- /.tab-pane -->
        </div>
        <!-- /.tab-content -->
    </div>
    <!-- /.nav-tabs-custom -->
</div>
<!-- /.col -->
</div>
<!-- /.row -->
</section>
@endsection

```

Assinalado a **amarelo** tens a forma de mostrar os campos que são devolvidos pelo controlador para a vista. Vais obter um resultado semelhante à imagem:

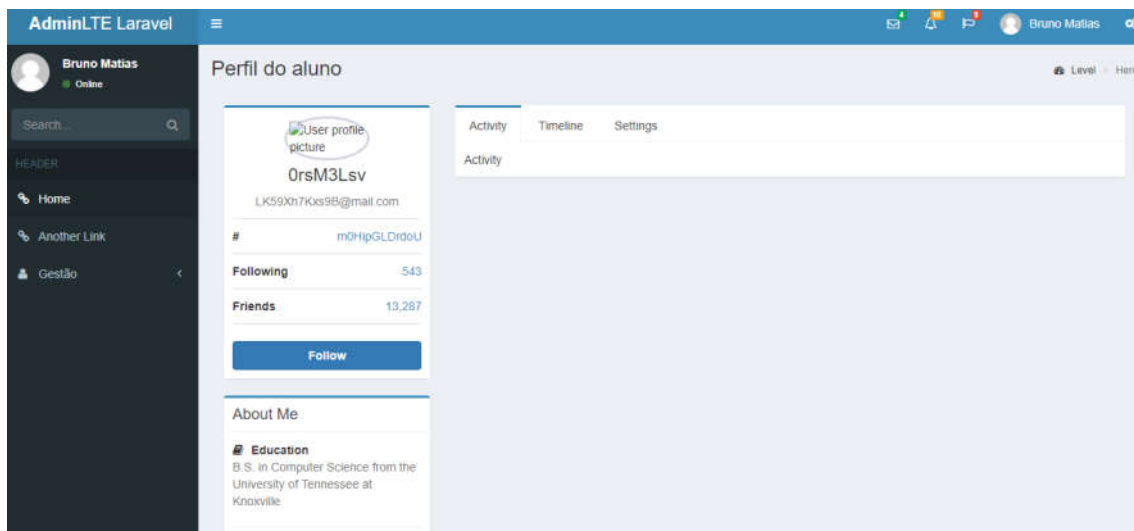


Figura 24 - perfil do aluno

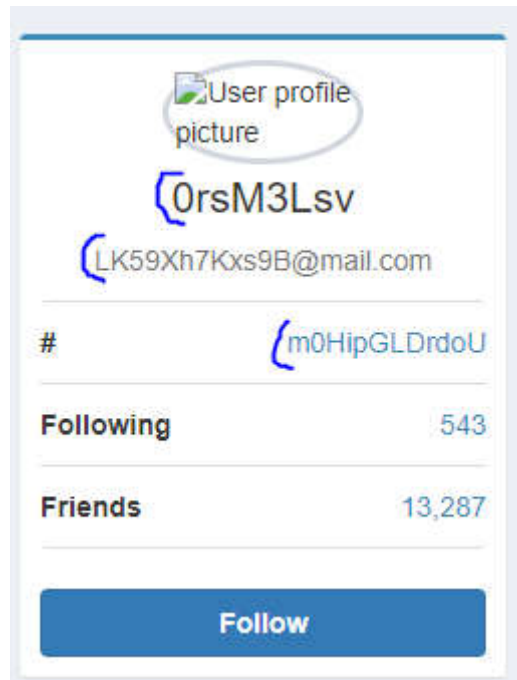


Figura 25 - assinalado os dados do aluno

Este foi um breve tutorial para começares a trabalhar com a *framework Laravel*. Para mais informações consulta a documentação em <https://laravel.com/docs/5.6>