**Laravel 5.3**

**CODECASTS (Fábio Vedovelli)**

<https://www.youtube.com/watch?v=25JnwUAH99U&list=PLy5T05I_eQYM4-B6W374JkAuh_wOrs3Pj>

Resumo do curso feito por Roberto Pinheiro

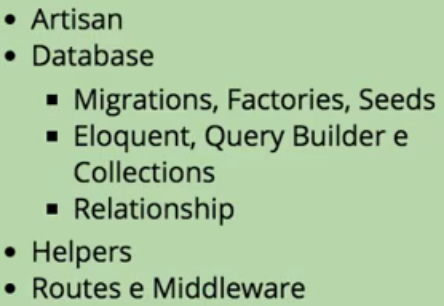
# Aula 01 - Introdução

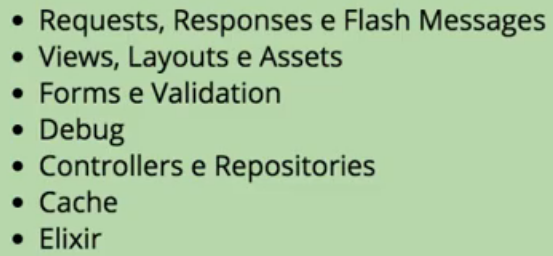
## Framework Laravel

É uma caixa de ferramentas que serve para trabalhar com PHP.

O Laravel permite criar de uma forma muito rápida um CRUD protegido por senha

## O que será visto no curso





# Aula 02 - Criando o projeto

A instalação é feita através de uma ferramenta chamada Composer, que é um gerenciador de dependências para o PHP.

Instale o Composer em seu computador.

Há duas formas de instalar o Laravel:

### Via Laravel Installer

O comando a seguir instala, de forma global, o Laravel no computador.

composer global require laravel/installer

O comando a seguir cria o projeto:

laravel new <nome\_projeto>

### Via Composer Create-Project

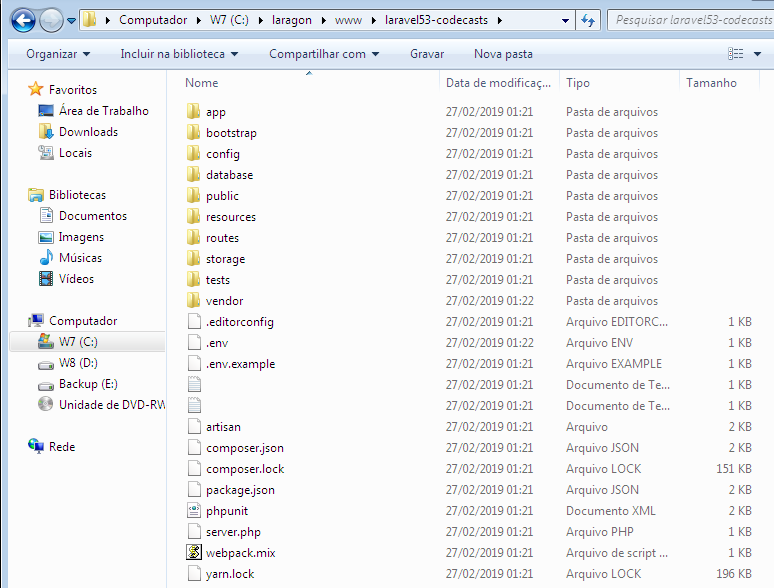
O comando abaixo instala o Laravel e cria o projeto na pasta ativa:

composer create-project --prefer-dist laravel/laravel <nome\_projeto>

## Instalação do Laravel e criação do projeto

composer global require laravel/installer

laravel new laravel53-codecasts

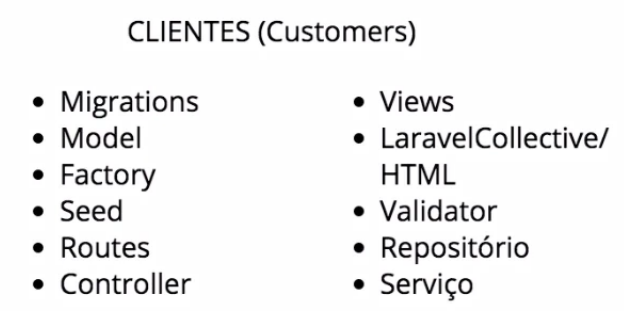


# Aula 05 - Planejamento

Será desenvolvido uma aplicação para controle de clientes e suas compras.

A aplicação precisa de um nome!

O que é necessário criar para que a funcionalidade dos clientes seja completa:



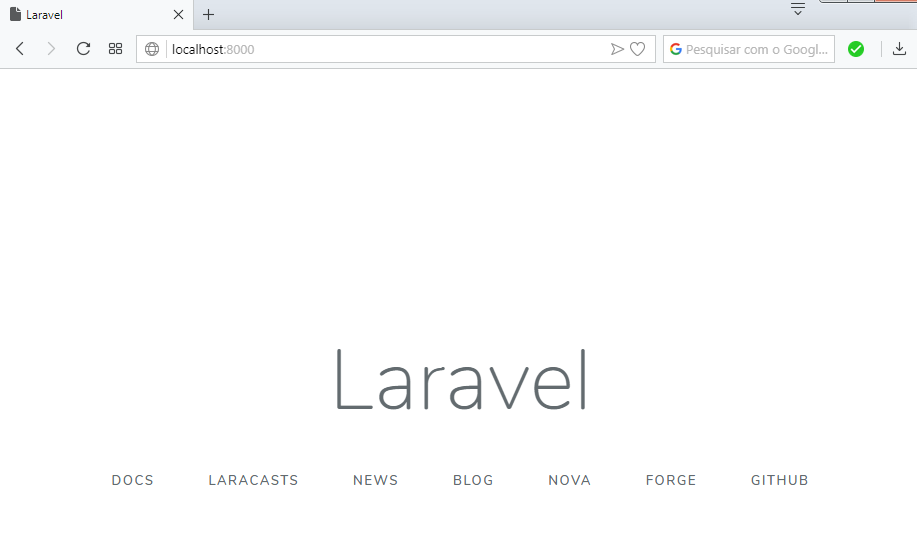
# Aula 06 - Nomear a aplicação

## Iniciando o servidor

php artisan serve

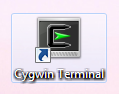
No browser:

localhost:8000



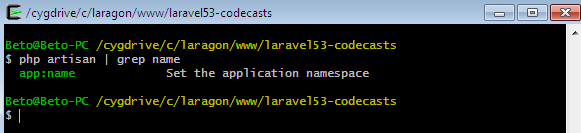
## Terminal Bash no Windows

Para a permitir que várias versões do Microsoft Windows possam, de certa forma, agir como um sistema Unix você pode experimentar o [CygWin](https://www.cygwin.com/), que é uma coleção de ferramentas de software livre originalmente desenvolvidas por **Cygnus Solutions**. Sua principal intenção é portar softwares que rodam em sistemas **POSIX** (como sistemas Linux, sistemas **BSD**, e sistemas Unix) para que rodem em Windows com pouco mais do que uma recompilação. O Cygwin é atualmente mantido por funcionários da [Red Hat](https://www.redhat.com/) e outras pessoas.

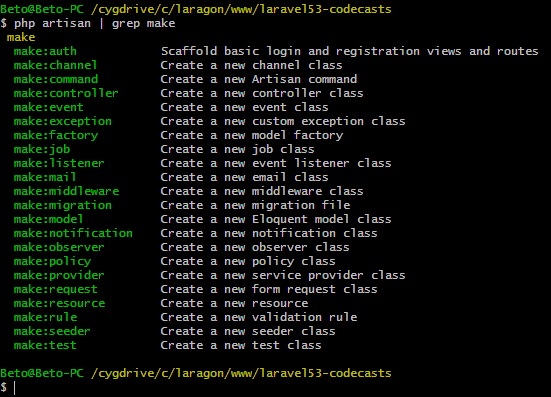


### Usando o Bash para filtrar comandos do artisan

php artisan | grep name



php artisan | grep make



# Aula 07 - Migrations - Overview

## Criando o banco de dados

Acesse o phpmyadmin e crie o banco de dados com o nome laravel53-codecasts

## Configurando o acesso ao banco de dados

**Arquivo .env**

DB\_CONNECTION=mysql

DB\_HOST=localhost

DB\_PORT=3306

DB\_DATABASE=laravel53-codecasts

DB\_USERNAME=root

DB\_PASSWORD=

## Migrations padrões

O Laravel já vem com duas migrations padrões:

2014\_10\_12\_000000\_create\_users\_table.php

2014\_10\_12\_100000\_create\_password\_resets\_table.php

**database/migrations/2014\_10\_12\_000000\_create\_users\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

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->bigIncrements('id');

$table->string('name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('users');

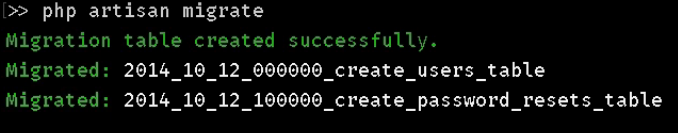
}

}

## Criando as tabelas

Para criar as tabelas:

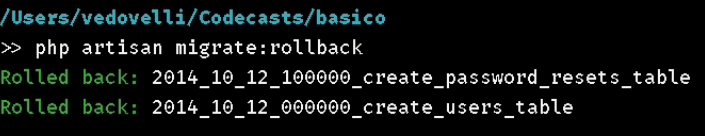
php artisan migrate



## Apagando as tabelas

Para voltar atrás e remover as tabelas:

php artisan migrate:rollback



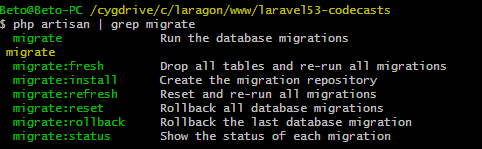
## Tipos de campos - Schema

A classe schema contém uma variedade de tipos de campos que podem ser especificados na construção das tabelas:

| **Command** | **Description** |
| --- | --- |
| $table->bigIncrements('id'); | Auto-incrementing UNSIGNED BIGINT (primary key) equivalent column. |
| $table->bigInteger('votes'); | BIGINT equivalent column. |
| $table->binary('data'); | BLOB equivalent column. |
| $table->boolean('confirmed'); | BOOLEAN equivalent column. |
| $table->char('name', 100); | CHAR equivalent column with an optional length. |
| $table->date('created\_at'); | DATE equivalent column. |
| $table->dateTime('created\_at'); | DATETIME equivalent column. |
| $table->dateTimeTz('created\_at'); | DATETIME (with timezone) equivalent column. |
| $table->decimal('amount', 8, 2); | DECIMAL equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->double('amount', 8, 2); | DOUBLE equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->enum('level', ['easy', 'hard']); | ENUM equivalent column. |
| $table->float('amount', 8, 2); | FLOAT equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->geometry('positions'); | GEOMETRY equivalent column. |
| $table->geometryCollection('positions'); | GEOMETRYCOLLECTION equivalent column. |
| $table->increments('id'); | Auto-incrementing UNSIGNED INTEGER (primary key) equivalent column. |
| $table->integer('votes'); | INTEGER equivalent column. |
| $table->ipAddress('visitor'); | IP address equivalent column. |
| $table->json('options'); | JSON equivalent column. |
| $table->jsonb('options'); | JSONB equivalent column. |
| $table->lineString('positions'); | LINESTRING equivalent column. |
| $table->longText('description'); | LONGTEXT equivalent column. |
| $table->macAddress('device'); | MAC address equivalent column. |
| $table->mediumIncrements('id'); | Auto-incrementing UNSIGNED MEDIUMINT (primary key) equivalent column. |
| $table->mediumInteger('votes'); | MEDIUMINT equivalent column. |
| $table->mediumText('description'); | MEDIUMTEXT equivalent column. |
| $table->morphs('taggable'); | Adds taggable\_id UNSIGNED BIGINT and taggable\_type VARCHAR equivalent columns. |
| $table->multiLineString('positions'); | MULTILINESTRING equivalent column. |
| $table->multiPoint('positions'); | MULTIPOINT equivalent column. |
| $table->multiPolygon('positions'); | MULTIPOLYGON equivalent column. |
| $table->nullableMorphs('taggable'); | Adds nullable versions of morphs() columns. |
| $table->nullableTimestamps(); | Alias of timestamps() method. |
| $table->point('position'); | POINT equivalent column. |
| $table->polygon('positions'); | POLYGON equivalent column. |
| $table->rememberToken(); | Adds a nullable remember\_tokenVARCHAR(100) equivalent column. |
| $table->smallIncrements('id'); | Auto-incrementing UNSIGNED SMALLINT (primary key) equivalent column. |
| $table->smallInteger('votes'); | SMALLINT equivalent column. |
| $table->softDeletes(); | Adds a nullable deleted\_at TIMESTAMP equivalent column for soft deletes. |
| $table->softDeletesTz(); | Adds a nullable deleted\_at TIMESTAMP (with timezone) equivalent column for soft deletes. |
| $table->string('name', 100); | VARCHAR equivalent column with a optional length. |
| $table->text('description'); | TEXT equivalent column. |
| $table->time('sunrise'); | TIME equivalent column. |
| $table->timeTz('sunrise'); | TIME (with timezone) equivalent column. |
| $table->timestamp('added\_on'); | TIMESTAMP equivalent column. |
| $table->timestampTz('added\_on'); | TIMESTAMP (with timezone) equivalent column. |
| $table->timestamps(); | Adds nullable created\_at and updated\_atTIMESTAMP equivalent columns. |
| $table->timestampsTz(); | Adds nullable created\_at and updated\_atTIMESTAMP (with timezone) equivalent columns. |
| $table->tinyIncrements('id'); | Auto-incrementing UNSIGNED TINYINT (primary key) equivalent column. |
| $table->tinyInteger('votes'); | TINYINT equivalent column. |
| $table->unsignedBigInteger('votes'); | UNSIGNED BIGINT equivalent column. |
| $table->unsignedDecimal('amount', 8, 2); | UNSIGNED DECIMAL equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->unsignedInteger('votes'); | UNSIGNED INTEGER equivalent column. |
| $table->unsignedMediumInteger('votes'); | UNSIGNED MEDIUMINT equivalent column. |
| $table->unsignedSmallInteger('votes'); | UNSIGNED SMALLINT equivalent column. |
| $table->unsignedTinyInteger('votes'); | UNSIGNED TINYINT equivalent column. |
| $table->uuid('id'); | UUID equivalent column. |
| $table->year('birth\_year'); | YEAR equivalent column. |

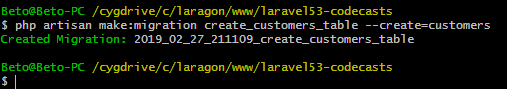
# Aula 08 - Migrations específicas

## Filtrando os comandos migrate

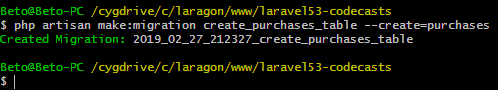


## Criando as migrations

php artisan make:migration create\_customers\_table --create=customers



php artisan make:migration create\_purchases\_table --create=purchases



**database/migrations/2019\_02\_27\_211109\_create\_customers\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCustomersTable extends Migration

{

public function up()

{

Schema::create('customers', function (Blueprint $table) {

$table->increments('id');

$table->string('name', 150);

$table->string('city', 50);

$table->string('state', 2);

$table->datetime('birthdate');

$table->boolean('special\_customer');

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('customers');

}

}

**database/migrations/2019\_02\_27\_212327\_create\_purchases\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePurchasesTable extends Migration

{

public function up()

{

Schema::create('purchases', function (Blueprint $table) {

$table->increments('id');

$table->integer('customer\_id')->unsigned();

$table->string('amount', 10);

$table->string('description');

$table->timestamps();

});

}

public function down()

{

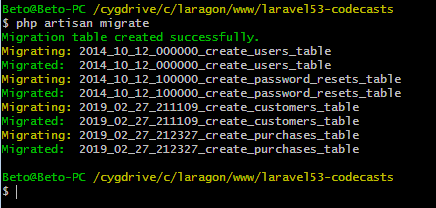
Schema::dropIfExists('purchases');

}

}

## Criando as tabelas

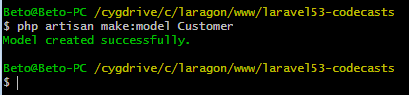
php artisan migrate



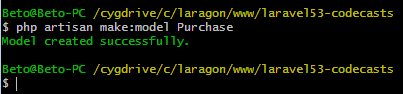
# Aula 09 - Model Factories

## Criando as models

php make:model Customer

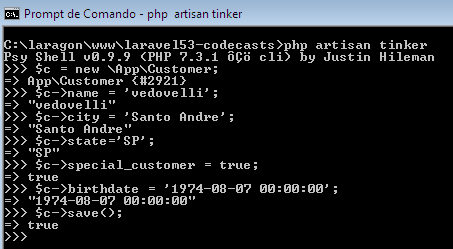


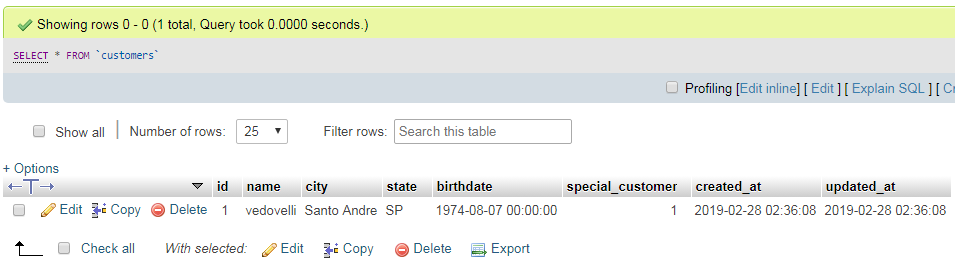
php make:model Purchase



## Inserindo um registro com o tinker

php artisan tinker





## Criando uma factory

php artisan make:factory CustomerFactory

## Usando a factory para criar registros aleatórios

**database/Factory/CustomerFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(App\Customer::class, function (Faker $faker) {

return [

'name' => $faker->name,

'city' => $faker->city,

'state' => $faker->stateAbbr,

'birthdate' => $faker->datetime(),

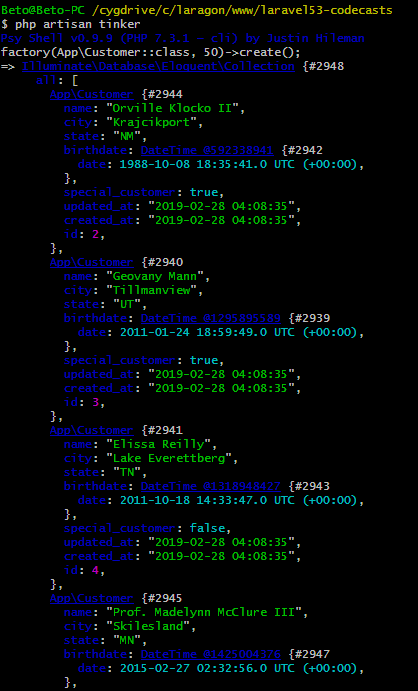
'special\_customer' => $faker->boolean

];

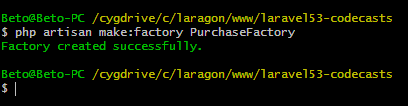
});

No terminal:

factory(App\Customer::class, 50)->create();







Acesse a URL:

https://github.com/fzaninotto/Faker

## Formatters

Each of the generator properties (like name, address, and lorem) are called "formatters". A faker generator has many of them, packaged in "providers". Here is a list of the bundled formatters in the default locale.

### Faker\Provider\Base

randomDigit // 7

randomDigitNotNull // 5

randomNumber($nbDigits = NULL, $strict = false) // 79907610

randomFloat($nbMaxDecimals = NULL, $min = 0, $max = NULL) // 48.8932

numberBetween($min = 1000, $max = 9000) // 8567

randomLetter // 'b'

// returns randomly ordered subsequence of a provided array

randomElements($array = array ('a','b','c'), $count = 1) // array('c')

randomElement($array = array ('a','b','c')) // 'b'

shuffle('hello, world') // 'rlo,h eoldlw'

shuffle(array(1, 2, 3)) // array(2, 1, 3)

numerify('Hello ###') // 'Hello 609'

lexify('Hello ???') // 'Hello wgt'

bothify('Hello ##??') // 'Hello 42jz'

asciify('Hello \*\*\*') // 'Hello R6+'

regexify('[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}'); // sm0@y8k96a.ej

### Faker\Provider\Lorem

word // 'aut'

words($nb = 3, $asText = false) // array('porro', 'sed', 'magni')

sentence($nbWords = 6, $variableNbWords = true) // 'Sit vitae voluptas sint non voluptates.'

sentences($nb = 3, $asText = false) // array('Optio quos qui illo error.', 'Laborum vero a officia id corporis.', 'Saepe provident esse hic eligendi.')

paragraph($nbSentences = 3, $variableNbSentences = true) // 'Ut ab voluptas sed a nam. Sint autem inventore aut officia aut aut blanditiis. Ducimus eos odit amet et est ut eum.'

paragraphs($nb = 3, $asText = false) // array('Quidem ut sunt et quidem est accusamus aut. Fuga est placeat rerum ut. Enim ex eveniet facere sunt.', 'Aut nam et eum architecto fugit repellendus illo. Qui ex esse veritatis.', 'Possimus omnis aut incidunt sunt. Asperiores incidunt iure sequi cum culpa rem. Rerum exercitationem est rem.')

text($maxNbChars = 200) // 'Fuga totam reiciendis qui architecto fugiat nemo. Consequatur recusandae qui cupiditate eos quod.'

### Faker\Provider\en\_US\Person

title($gender = null|'male'|'female') // 'Ms.'

titleMale // 'Mr.'

titleFemale // 'Ms.'

suffix // 'Jr.'

**name($gender = null|'male'|'female') // 'Dr. Zane Stroman'**

firstName($gender = null|'male'|'female') // 'Maynard'

firstNameMale // 'Maynard'

firstNameFemale // 'Rachel'

lastName // 'Zulauf'

### Faker\Provider\en\_US\Address

cityPrefix // 'Lake'

secondaryAddress // 'Suite 961'

state // 'NewMexico'

**stateAbbr // 'OH'**

citySuffix // 'borough'

streetSuffix // 'Keys'

buildingNumber // '484'

city // 'West Judge'

streetName // 'Keegan Trail'

streetAddress // '439 Karley Loaf Suite 897'

postcode // '17916'

address // '8888 Cummings Vista Apt. 101, Susanbury, NY 95473'

country // 'Falkland Islands (Malvinas)'

latitude($min = -90, $max = 90) // 77.147489

longitude($min = -180, $max = 180) // 86.211205

### Faker\Provider\en\_US\PhoneNumber

phoneNumber // '201-886-0269 x3767'

tollFreePhoneNumber // '(888) 937-7238'

e164PhoneNumber // '+27113456789'

### Faker\Provider\en\_US\Company

catchPhrase // 'Monitored regional contingency'

bs // 'e-enable robust architectures'

company // 'Bogan-Treutel'

companySuffix // 'and Sons'

jobTitle // 'Cashier'

### Faker\Provider\en\_US\Text

realText($maxNbChars = 200, $indexSize = 2) // "And yet I wish you could manage it?) 'And what are they made of?' Alice asked in a shrill, passionate voice. 'Would YOU like cats if you were never even spoke to Time!' 'Perhaps not,' Alice replied."

### Faker\Provider\DateTime

unixTime($max = 'now') // 58781813

dateTime($max = 'now', $timezone = null) // DateTime('2008-04-25 08:37:17', 'UTC')

dateTimeAD($max = 'now', $timezone = null) // DateTime('1800-04-29 20:38:49', 'Europe/Paris')

iso8601($max = 'now') // '1978-12-09T10:10:29+0000'

date($format = 'Y-m-d', $max = 'now') // '1979-06-09'

time($format = 'H:i:s', $max = 'now') // '20:49:42'

dateTimeBetween($startDate = '-30 years', $endDate = 'now', $timezone = null) // DateTime('2003-03-15 02:00:49', 'Africa/Lagos')

dateTimeInInterval($startDate = '-30 years', $interval = '+ 5 days', $timezone = null) // DateTime('2003-03-15 02:00:49', 'Antartica/Vostok')

dateTimeThisCentury($max = 'now', $timezone = null) // DateTime('1915-05-30 19:28:21', 'UTC')

dateTimeThisDecade($max = 'now', $timezone = null) // DateTime('2007-05-29 22:30:48', 'Europe/Paris')

dateTimeThisYear($max = 'now', $timezone = null) // DateTime('2011-02-27 20:52:14', 'Africa/Lagos')

dateTimeThisMonth($max = 'now', $timezone = null) // DateTime('2011-10-23 13:46:23', 'Antarctica/Vostok')

amPm($max = 'now') // 'pm'

dayOfMonth($max = 'now') // '04'

dayOfWeek($max = 'now') // 'Friday'

month($max = 'now') // '06'

monthName($max = 'now') // 'January'

year($max = 'now') // '1993'

century // 'VI'

timezone // 'Europe/Paris'

Methods accepting a $timezone argument default to date\_default\_timezone\_get(). You can pass a custom timezone string to each method, or define a custom timezone for all time methods at once using $faker::setDefaultTimezone($timezone).

### Faker\Provider\Internet

email // 'tkshlerin@collins.com'

safeEmail // 'king.alford@example.org'

freeEmail // 'bradley72@gmail.com'

companyEmail // 'russel.durward@mcdermott.org'

freeEmailDomain // 'yahoo.com'

safeEmailDomain // 'example.org'

userName // 'wade55'

password // 'k&|X+a45\*2['

domainName // 'wolffdeckow.net'

domainWord // 'feeney'

tld // 'biz'

url // 'http://www.skilesdonnelly.biz/aut-accusantium-ut-architecto-sit-et.html'

slug // 'aut-repellat-commodi-vel-itaque-nihil-id-saepe-nostrum'

ipv4 // '109.133.32.252'

localIpv4 // '10.242.58.8'

ipv6 // '8e65:933d:22ee:a232:f1c1:2741:1f10:117c'

macAddress // '43:85:B7:08:10:CA'

### Faker\Provider\UserAgent

userAgent // 'Mozilla/5.0 (Windows CE) AppleWebKit/5350 (KHTML, like Gecko) Chrome/13.0.888.0 Safari/5350'

chrome // 'Mozilla/5.0 (Macintosh; PPC Mac OS X 10\_6\_5) AppleWebKit/5312 (KHTML, like Gecko) Chrome/14.0.894.0 Safari/5312'

firefox // 'Mozilla/5.0 (X11; Linuxi686; rv:7.0) Gecko/20101231 Firefox/3.6'

safari // 'Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10\_7\_1 rv:3.0; en-US) AppleWebKit/534.11.3 (KHTML, like Gecko) Version/4.0 Safari/534.11.3'

opera // 'Opera/8.25 (Windows NT 5.1; en-US) Presto/2.9.188 Version/10.00'

internetExplorer // 'Mozilla/5.0 (compatible; MSIE 7.0; Windows 98; Win 9x 4.90; Trident/3.0)'

### Faker\Provider\Payment

creditCardType // 'MasterCard'

creditCardNumber // '4485480221084675'

creditCardExpirationDate // 04/13

creditCardExpirationDateString // '04/13'

creditCardDetails // array('MasterCard', '4485480221084675', 'Aleksander Nowak', '04/13')

// Generates a random IBAN. Set $countryCode to null for a random country

iban($countryCode) // 'IT31A8497112740YZ575DJ28BP4'

swiftBicNumber // 'RZTIAT22263'

### Faker\Provider\Color

hexcolor // '#fa3cc2'

rgbcolor // '0,255,122'

rgbColorAsArray // array(0,255,122)

rgbCssColor // 'rgb(0,255,122)'

safeColorName // 'fuchsia'

colorName // 'Gainsbor'

### Faker\Provider\File

fileExtension // 'avi'

mimeType // 'video/x-msvideo'

// Copy a random file from the source to the target directory and returns the fullpath or filename

file($sourceDir = '/tmp', $targetDir = '/tmp') // '/path/to/targetDir/13b73edae8443990be1aa8f1a483bc27.jpg'

file($sourceDir, $targetDir, false) // '13b73edae8443990be1aa8f1a483bc27.jpg'

### Faker\Provider\Image

// Image generation provided by LoremPixel (http://lorempixel.com/)

imageUrl($width = 640, $height = 480) // 'http://lorempixel.com/640/480/'

imageUrl($width, $height, 'cats') // 'http://lorempixel.com/800/600/cats/'

imageUrl($width, $height, 'cats', true, 'Faker') // 'http://lorempixel.com/800/400/cats/Faker'

imageUrl($width, $height, 'cats', true, 'Faker', true) // 'http://lorempixel.com/grey/800/400/cats/Faker/' Monochrome image

image($dir = '/tmp', $width = 640, $height = 480) // '/tmp/13b73edae8443990be1aa8f1a483bc27.jpg'

image($dir, $width, $height, 'cats') // 'tmp/13b73edae8443990be1aa8f1a483bc27.jpg' it's a cat!

image($dir, $width, $height, 'cats', false) // '13b73edae8443990be1aa8f1a483bc27.jpg' it's a filename without path

image($dir, $width, $height, 'cats', true, false) // it's a no randomize images (default: `true`)

image($dir, $width, $height, 'cats', true, true, 'Faker') // 'tmp/13b73edae8443990be1aa8f1a483bc27.jpg' it's a cat with 'Faker' text. Default, `null`.

### Faker\Provider\Uuid

uuid // '7e57d004-2b97-0e7a-b45f-5387367791cd'

### Faker\Provider\Barcode

ean13 // '4006381333931'

ean8 // '73513537'

isbn13 // '9790404436093'

isbn10 // '4881416324'

### Faker\Provider\Miscellaneous

boolean // false

boolean($chanceOfGettingTrue = 50) // true

md5 // 'de99a620c50f2990e87144735cd357e7'

sha1 // 'f08e7f04ca1a413807ebc47551a40a20a0b4de5c'

sha256 // '0061e4c60dac5c1d82db0135a42e00c89ae3a333e7c26485321f24348c7e98a5'

locale // en\_UK

countryCode // UK

languageCode // en

currencyCode // EUR

emoji // 😁

### Faker\Provider\Biased

// get a random number between 10 and 20,

// with more chances to be close to 20

biasedNumberBetween($min = 10, $max = 20, $function = 'sqrt')

### Faker\Provider\HtmlLorem

//Generate HTML document which is no more than 2 levels deep, and no more than 3 elements wide at any level.

randomHtml(2,3) // <html><head><title>Aut illo dolorem et accusantium eum.</title></head><body><form action="example.co

## Modifiers

Faker provides three special providers, unique(), optional(), and valid(), to be called before any provider.

// unique() forces providers to return unique values

$values = array();

for ($i=0; $i < 10; $i++) {

// get a random digit, but always a new one, to avoid duplicates

$values []= $faker->unique()->randomDigit;

}

print\_r($values); // [4, 1, 8, 5, 0, 2, 6, 9, 7, 3]

// providers with a limited range will throw an exception when no new unique value can be generated

$values = array();

try {

for ($i=0; $i < 10; $i++) {

$values []= $faker->unique()->randomDigitNotNull;

}

} catch (\OverflowException $e) {

echo "There are only 9 unique digits not null, Faker can't generate 10 of them!";

}

// you can reset the unique modifier for all providers by passing true as first argument

$faker->unique($reset = true)->randomDigitNotNull; // will not throw OverflowException since unique() was reset

// tip: unique() keeps one array of values per provider

// optional() sometimes bypasses the provider to return a default value instead (which defaults to NULL)

$values = array();

for ($i=0; $i < 10; $i++) {

// get a random digit, but also null sometimes

$values []= $faker->optional()->randomDigit;

}

print\_r($values); // [1, 4, null, 9, 5, null, null, 4, 6, null]

// optional() accepts a weight argument to specify the probability of receiving the default value.

// 0 will always return the default value; 1 will always return the provider. Default weight is 0.5 (50% chance).

$faker->optional($weight = 0.1)->randomDigit; // 90% chance of NULL

$faker->optional($weight = 0.9)->randomDigit; // 10% chance of NULL

// optional() accepts a default argument to specify the default value to return.

// Defaults to NULL.

$faker->optional($weight = 0.5, $default = false)->randomDigit; // 50% chance of FALSE

$faker->optional($weight = 0.9, $default = 'abc')->word; // 10% chance of 'abc'

// valid() only accepts valid values according to the passed validator functions

$values = array();

$evenValidator = function($digit) {

return $digit % 2 === 0;

};

for ($i=0; $i < 10; $i++) {

$values []= $faker->valid($evenValidator)->randomDigit;

}

print\_r($values); // [0, 4, 8, 4, 2, 6, 0, 8, 8, 6]

// just like unique(), valid() throws an overflow exception when it can't generate a valid value

$values = array();

try {

$faker->valid($evenValidator)->randomElement(1, 3, 5, 7, 9);

} catch (\OverflowException $e) {

echo "Can't pick an even number in that set!";

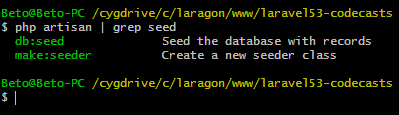
}

If you would like to use a modifier with a value not generated by Faker, use the passthrough() method. passthrough() simply returns whatever value it was given.

$faker->optional()->passthrough(mt\_rand(5, 15));

# Aula 10 - Seeders

## Filtrando os comandos seed com grep



## Criando factory e seeder para a tabela Customer

**database/Factory/CustomerFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(App\Customer::class, function (Faker $faker) {

return [

'name' => $faker->name,

'city' => $faker->city,

'state' => $faker->stateAbbr,

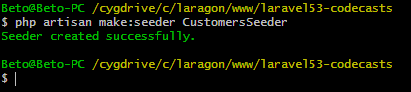
'birthdate' => $faker->datetime(),

'special\_customer' => $faker->boolean

];

});

php artisan make:seeder CustomersSeeder



**database/seeds/CustomersSeeder.php**

<?php

use Illuminate\Database\Seeder;

use App\Customer;

class CustomersSeeder extends Seeder

{

public function run()

{

Customer::truncate();

factory(App\Customer::class, 50)->create();

}

}

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

// $this->call(UsersTableSeeder::class);

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

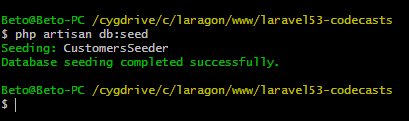
}

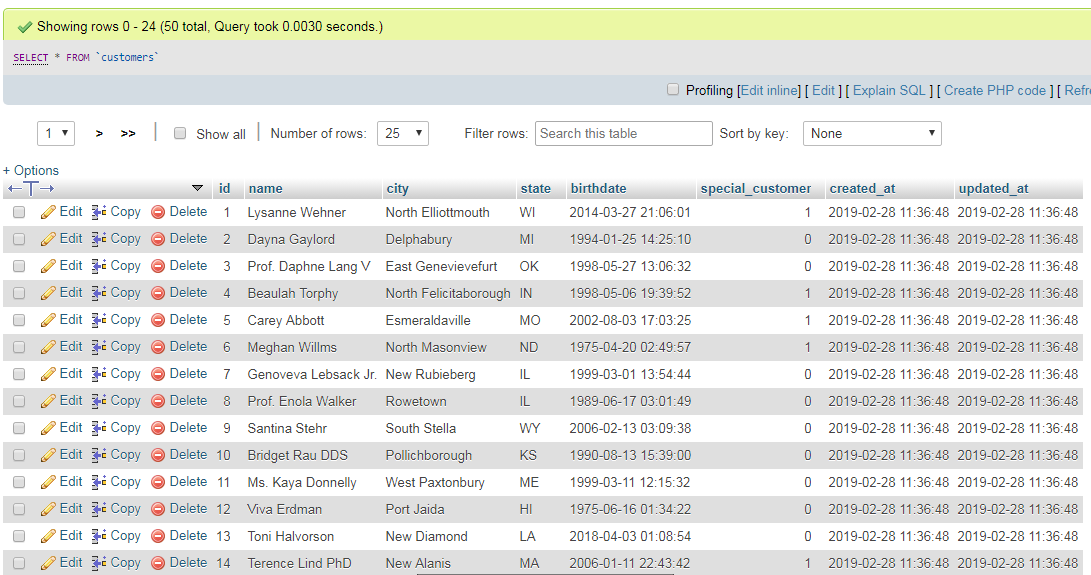
}

## Populando a tabela Customer

No terminal execute o seguinte comando:

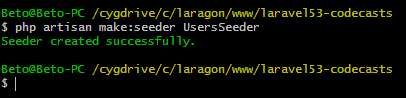
php artisan db:seed





## Criando factory e seeder para a tabela de usuários

php artisan make:seeder UsersSeeder



**database/seeds/UsersSeeder.php**

<?php

use Illuminate\Database\Seeder;

use App\User;

class UsersSeeder extends Seeder

{

public function run()

{

User::truncate();

$user = new User();

$user->name = 'Fabio Vedovelli';

$user->email = 'vedovelli@gmail.com';

$user->password = bcrypt(123456);

$user->email\_verified\_at = now();

$user->save();

factory(App\User::class, 50)->create();

}

}

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

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

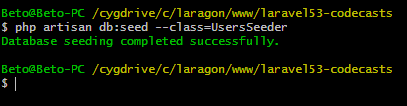
$this->call(CustomersSeeder::class);

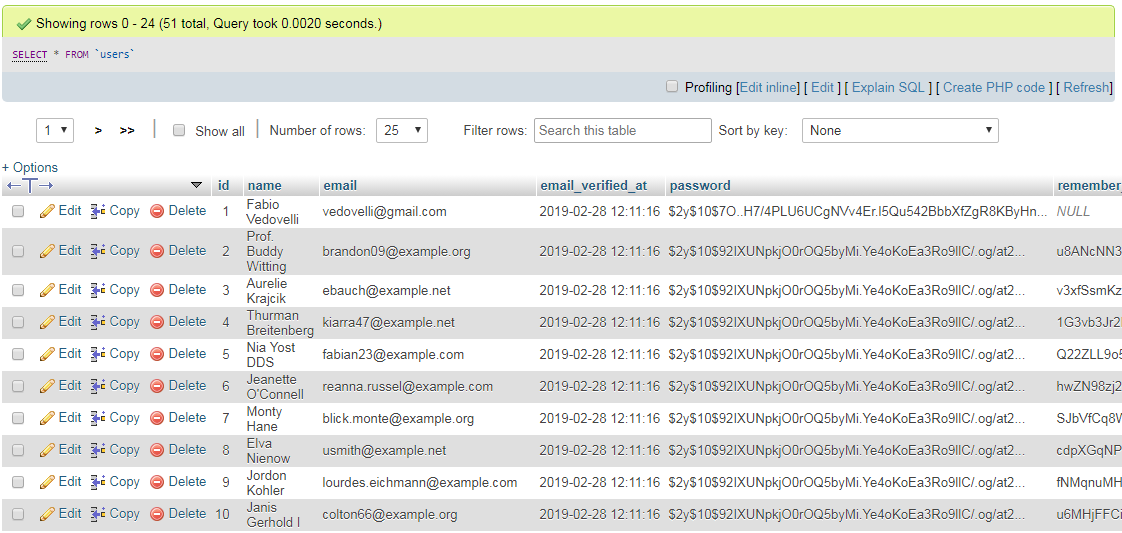
}

}

## Populando apenas a tabela de usuários

php artisan db:seed --class=UsersSeeder





## Criando o factory e o seeder para a tabela de compras

**database/Factory/PurchaseFactory.php**

<?php

use Faker\Generator as Faker;

use App\Customer;

$factory->define(App\Purchase::class, function (Faker $faker) {

return [

'customer\_id' => $faker->numberBetween($min = 1, $max = 50),

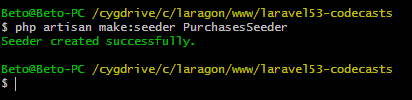
'amount' => $faker->randomFloat(2, 1, 400),

'description' => $faker->sentence(),

];

});

php artisan make:seeder PurchasesSeeder



**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

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

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

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

}

}

**database/seeds/PurchasesSeeder.php**

<?php

use Illuminate\Database\Seeder;

use App\Purchase;

class PurchasesSeeder extends Seeder

{

public function run()

{

Purchase::truncate();

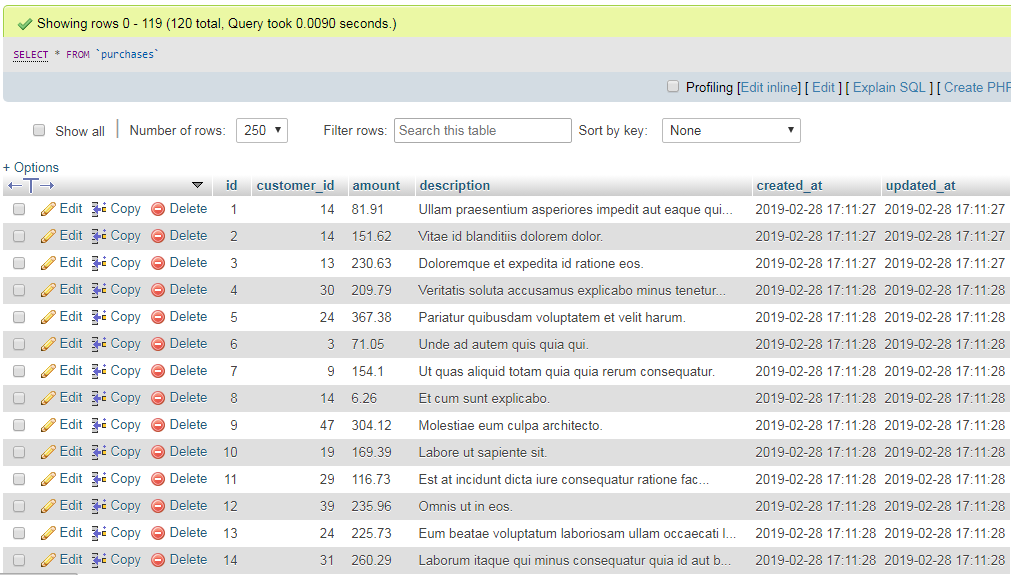
factory(App\Purchase::class, 120)->create();

}

}

## Populando apenas a tabela de compras

php artisan db:seed --class=PurchasesSeeder



# Aula 11 - Models

## Query Builder

**routes/web.php**

Route::get('/testes', function () {

$query1 = DB::table('customers')

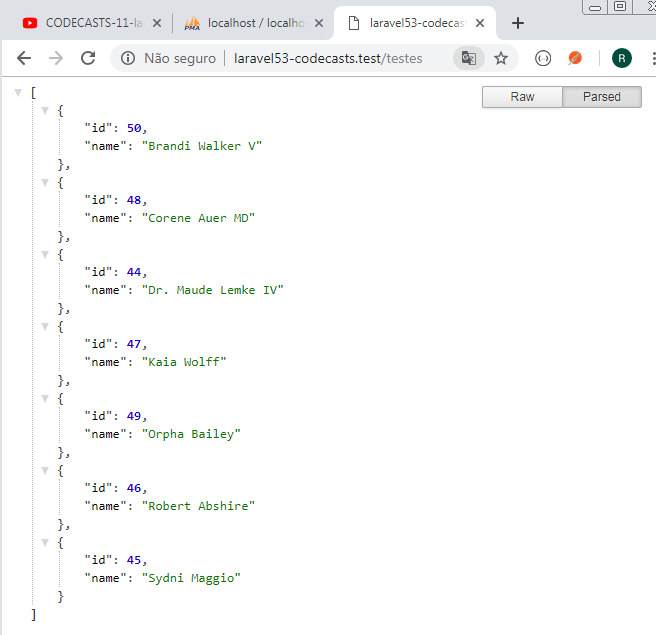
->where('id','>=', 44)

->orderBy('name')

->get(['id','name']);

return $query1;

});



## Eloquent

**routes/web.php**

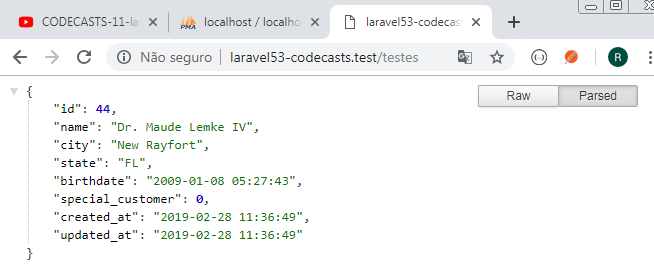
use App\Customer;

Route::get('/testes', function () {

$query2 = Customer::find(44);

return $query2;

});



**routes/web.php**

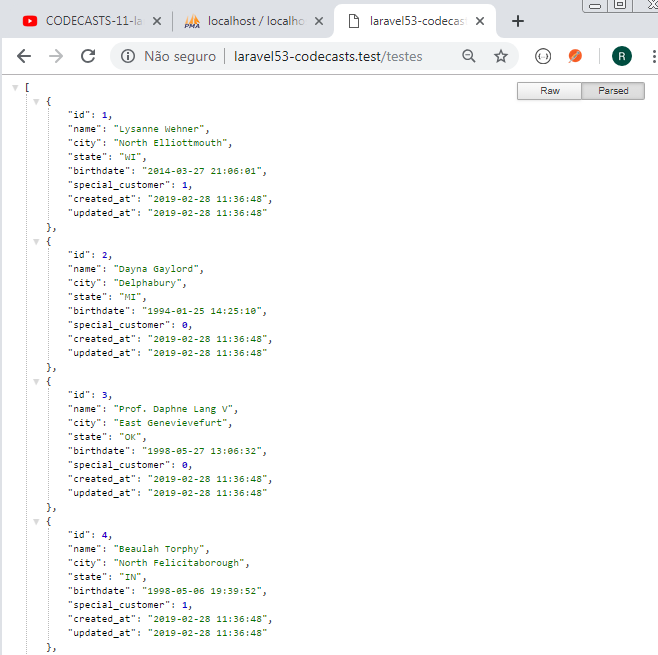
use App\Customer;

Route::get('/testes', function () {

$query3 = Customer::all();

return $query3;

});



**routes/web.php**

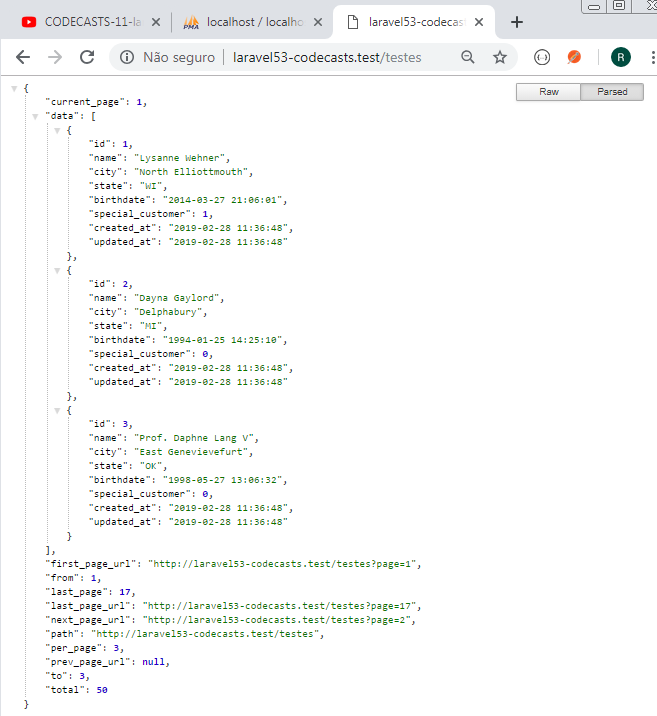
use App\Customer;

Route::get('/testes', function () {

$query4 = Customer::paginate(3);

return $query4;

});



## Relacionamento de tabelas

**app/Customer.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

public function purchases(){

// Um cliente possui muitas compras

return $this->hasMany(Purchase::class);

}

}

**routes/web.php**

Route::get('/testes', function () {

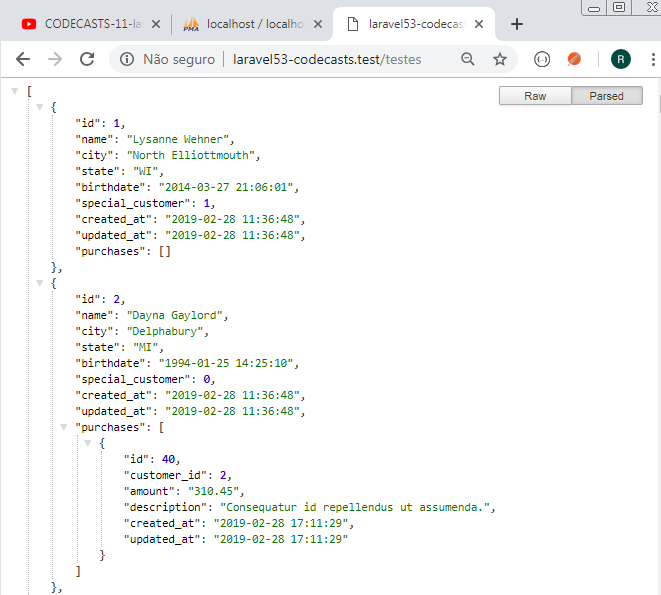
// Relacionamento

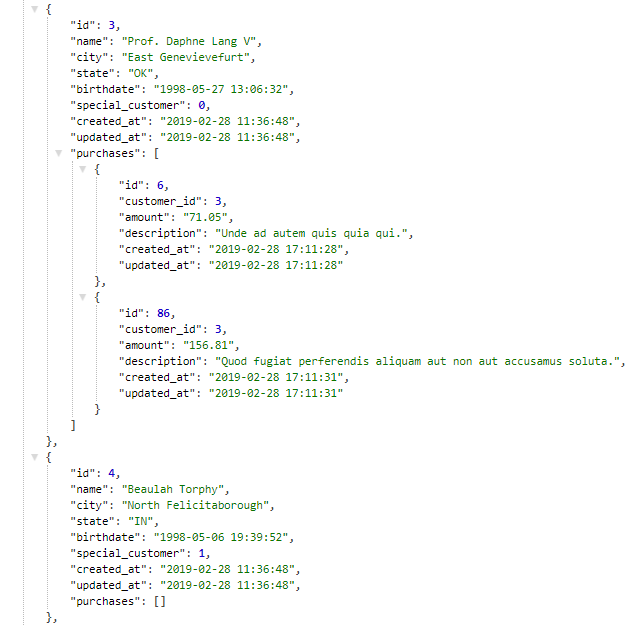
$query5 = Customer::with('purchases')->get();

return $query5;

});

### Compras feitas por cliente





# Aula 12 - Models - Listar registros

**routes/web.php**

use App\Customer;

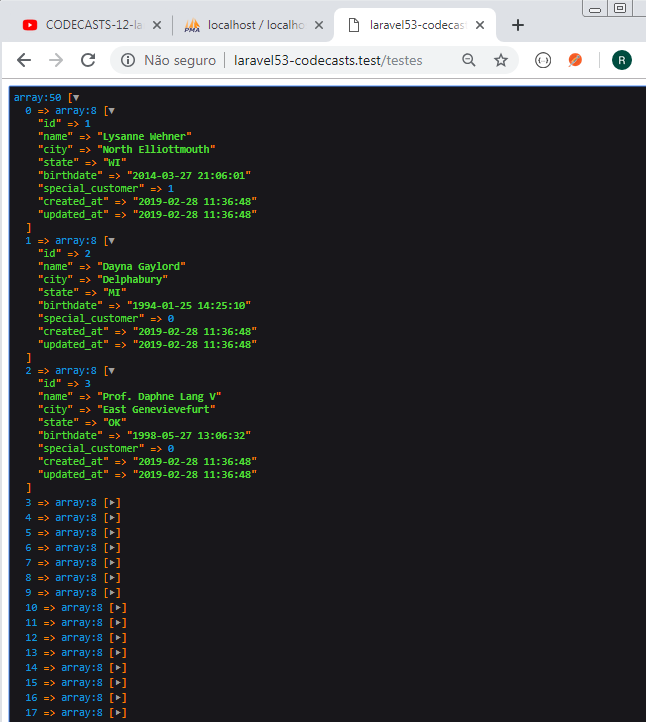
use App\Purchase;

Route::get('/testes', function () {

$customer1 = Customer::all();

dd($customer1->toArray());

});



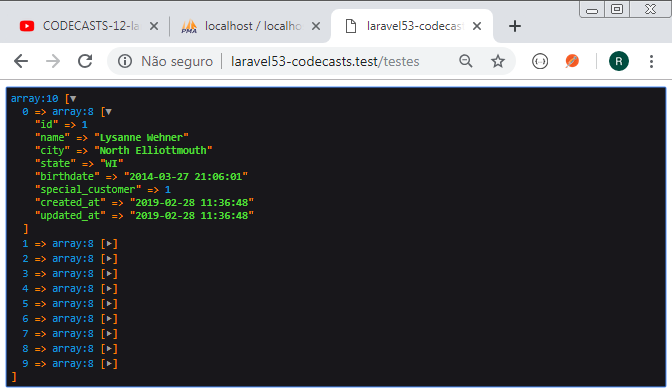
**routes/web.php**

Route::get('/testes', function () {

$customer2 = Customer::with('purchases')->get();

dd($customer2->toArray());

});



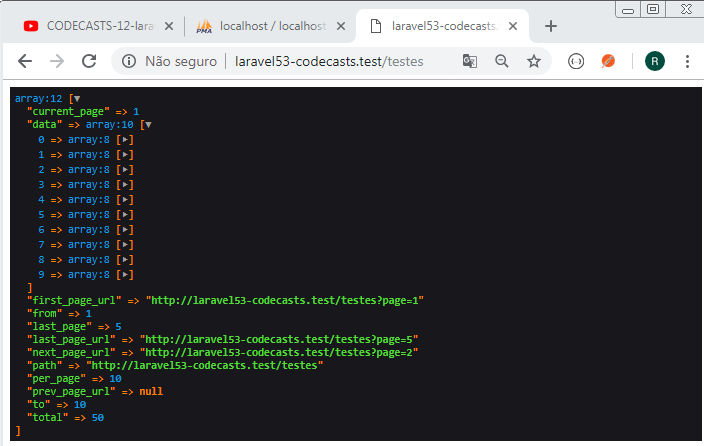
**routes/web.php**

Route::get('/testes', function () {

$customer3 = Customer::take(10)->get();

dd($customer3->toArray());

});



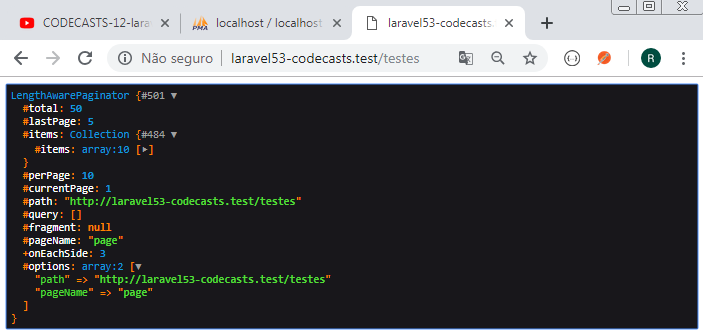
**routes/web.php**

Route::get('/testes', function () {

$customer4 = Customer::paginate(10);

dd($customer4);

});



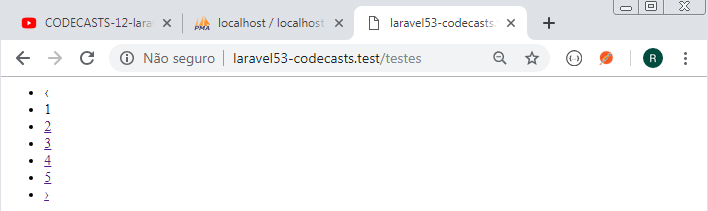
**routes/web.php**

Route::get('/testes', function () {

$customer5 = Customer::render();

echo $customer5;

});



# Aula 13 - Models - Criar registros

Efetuar um truncate na tabela customer e posteriormente entrar com o seguinte código:

**routes/web.php**

use App\Customer;

// CRIAR NOVO CUSTOMER

Route::get('/testes2', function () {

$customer = new Customer();

$customer->name = 'Fabio Vedovelli';

$customer->city = 'Santo Andre';

$customer->state = 'SP';

$customer->birthdate = '1974-08-07 00:00:00';

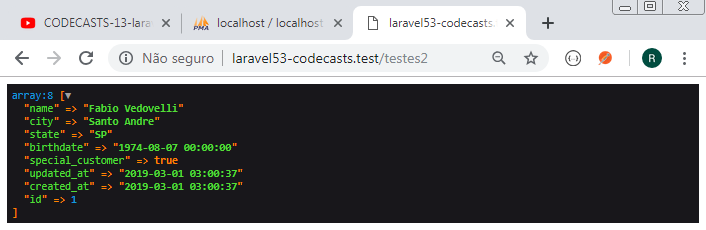
$customer->special\_customer = true;

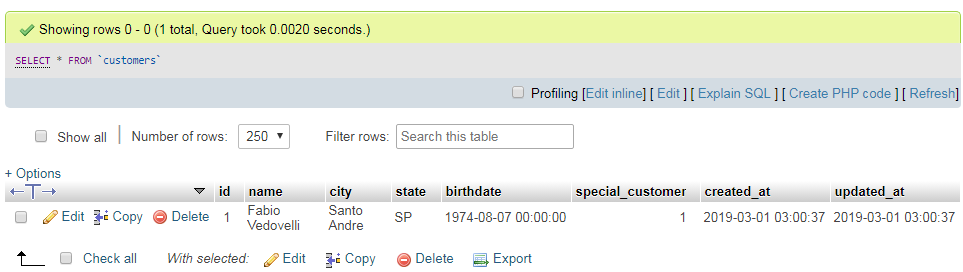
$customer->save();

dd($customer->toArray());

});

Enquanto o método save não for chamado, o objeto ficará criado apenas em memória.





**routes/web.php**

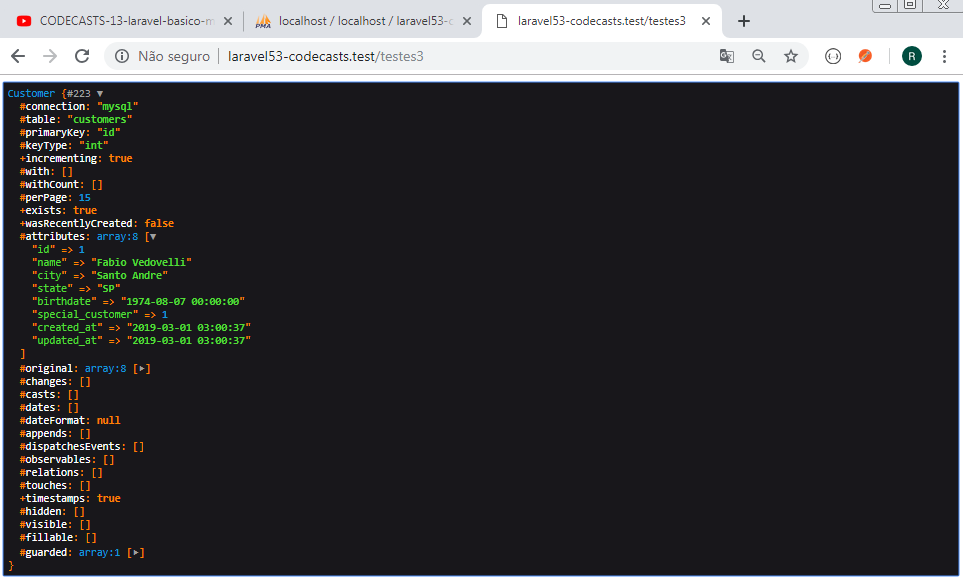
// TRAZER CUSTOMER ID 1

Route::get('/testes3', function () {

$customer = Customer::find(1);

dd($customer);

});



**app/Customer.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

protected $casts = [

'special\_customer' => 'boolean',

];

public function purchases(){

// Um cliente possui muitas compras

return $this->hasMany(Purchase::class);

}

}

**routes/web.php**

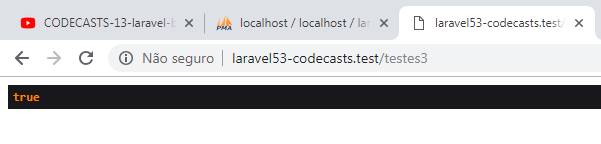
// TRAZER CUSTOMER ID 1

Route::get('/testes3', function () {

$customer = Customer::find(1);

dd($customer->special\_customer);

});



**routes/web.php**

// TRAZER CUSTOMER ID 1

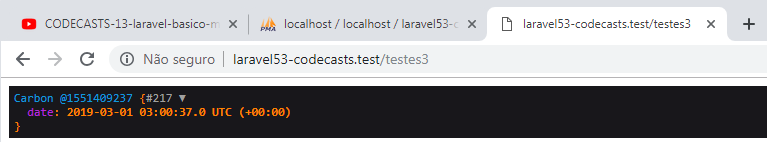
Route::get('/testes3', function () {

$customer = Customer::find(1);

// dd($customer->special\_customer);

dd($customer->created\_at);

});



Carbon é uma classe para trabalhar com dados do tipo data. Essa bibiloteca faz comparação entre datas.

**app/Customer.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

protected $casts = [

'special\_customer' => 'boolean',

'birthdate' => 'date'

];

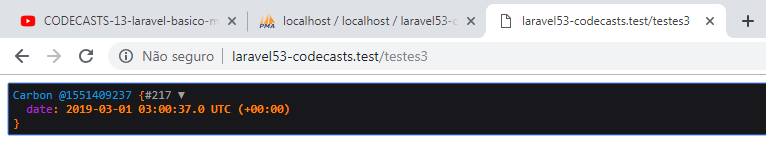
public function purchases(){

// Um cliente possui muitas compras

return $this->hasMany(Purchase::class);

}

}



# Aula 14 - Models - Atualizar registros

**routes/web.php**

use App\Customer;

Route::get('/testes4', function () {

$data = [

'name' => 'Pedro',

'birthdate' => '1989-01-01 12:00:00'

];

$customer = Customer::find(1);

$customer->update($data);

return 'Alterado com sucesso!';

});

**app/Customer.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

protected $fillable = [

'name', 'city', 'state', 'birthdate', 'special\_customer'

];

protected $casts = [

'special\_customer' => 'boolean',

'birthdate' => 'date'

];

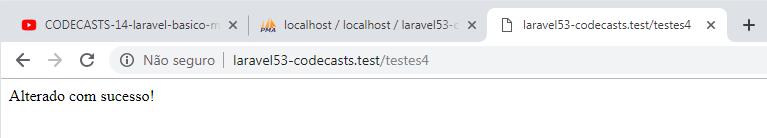
public function purchases(){

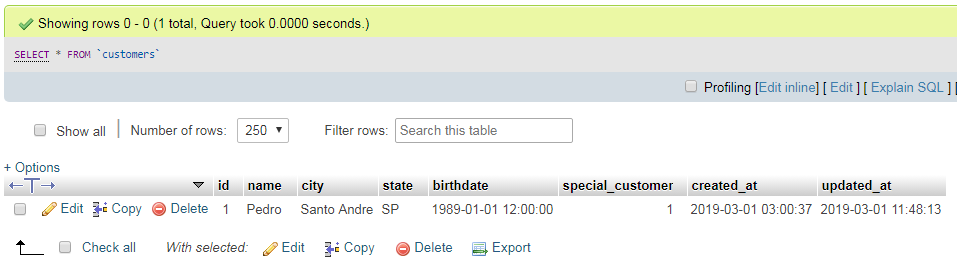
// Um cliente possui muitas compras

return $this->hasMany(Purchase::class);

}

}





**routes/web.php**

// INSERINDO REGISTRO

Route::get('/testes5', function () {

$data = [

'name' => 'Fabio Vedovelli',

'city' => 'Santo Andre',

'state' => 'SP',

'birthdate' => '1974-08-07 00:00:00',

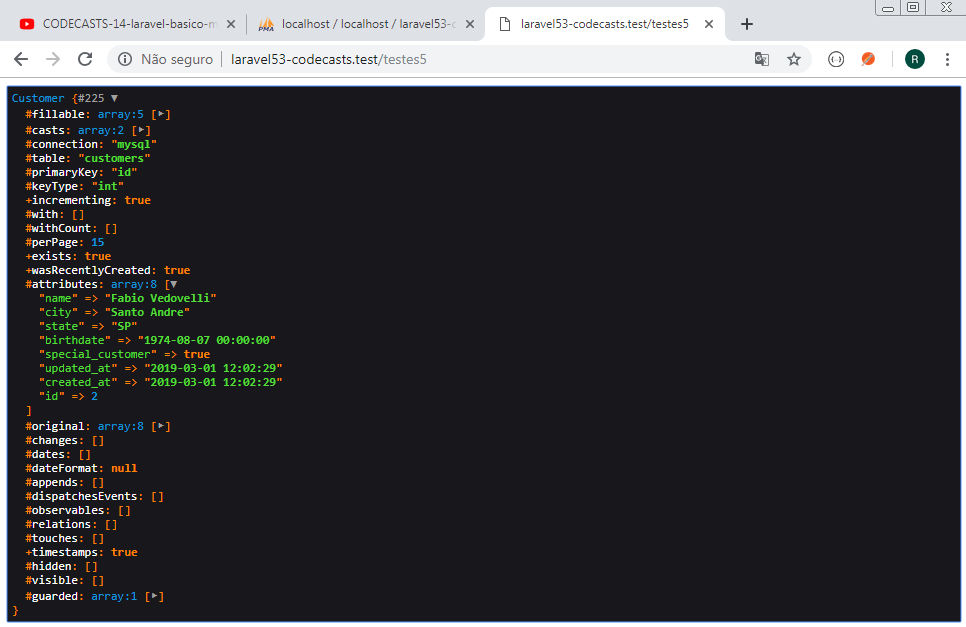
'special\_customer' => true

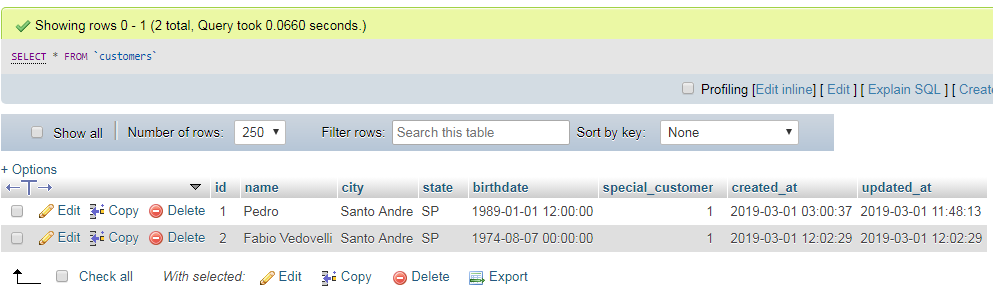
];

$customer = Customer::create($data);

dd($customer);

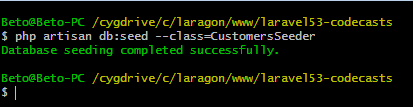
});





Popular novamente o banco de dados:

php artisan db:seed --class=CustomersSeeder



# Aula 16 - Rotas

// Listar dados de todos os clientes

// admin/clientes

// método GET

// Listar dados de um cliente específico

// admin/clientes/1

// método GET

// Criar (inserir) novo cliente

// admin/clientes

// método POST

// Alterar dados (atualizar) de um cliente

// admin/clientes/1

// método PUT

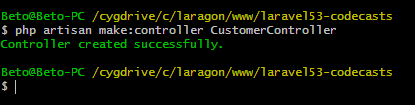
// Excluir um cliente específico

// admin/clientes/1

// método DELETE

## Criando o controller para Customer

php artisan make:controller CustomerController



## Configurando as rotas

**routes/web.php**

Route::group(['prefix' => 'admin'], function(){

Route::get('clientes', ['as' => 'customer.index', 'uses' => 'CustomerController@index']);

Route::get('cliente/{id}', ['as' => 'customer.get', 'uses' => 'CustomerController@get']);

Route::post('cliente', ['as' => 'customer.create', 'uses' => 'CustomerController@create']);

Route::put('cliente/{id}', ['as' => 'customer.update', 'uses' => 'CustomerController@update']);

Route::delete('cliente/{id}', ['as' => 'customer.delete', 'uses' => 'CustomerController@delete']);

});

**app/Http/Controllers/CustomerController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CustomerController extends Controller

{

public function index(){

return 'Hello!';

}

public function get($id){

return $id;

}

public function create(){

}

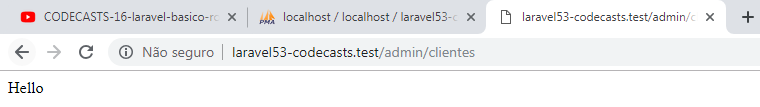
public function update($id){

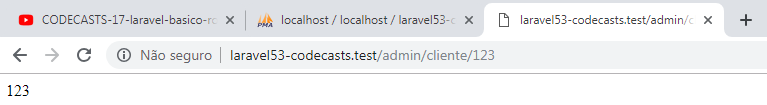
}

public function delete($id){

}

}

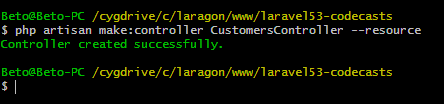




# Aula 17 - Rotas - Resource

## Criando o controller

php artisan make:controller CustomersController --resource



**app/Http/Controllers/CustomersController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CustomersController extends Controller

{

public function index()

{

//

}

public function create()

{

//

}

public function store(Request $request)

{

//

}

public function show($id)

{

//

}

public function edit($id)

{

//

}

public function update(Request $request, $id)

{

//

}

public function destroy($id)

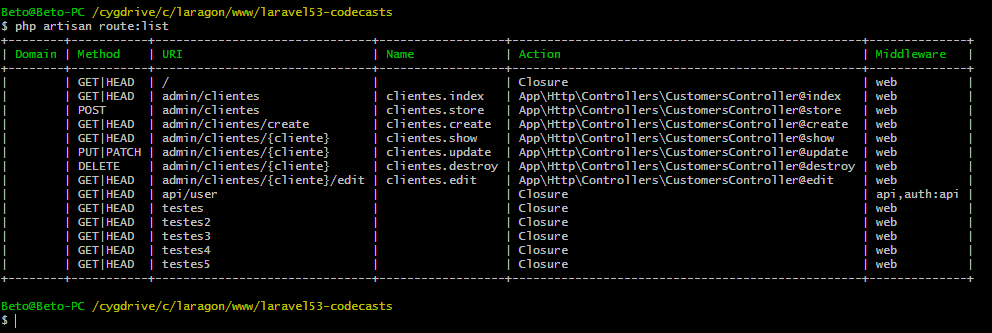
{

//

}

}

php artisan route:list



**routes/web.php**

Route::group(['prefix' => 'admin'], function(){

Route::resource('clientes', 'CustomersController');

});

# Aula 19 - Views

**app/Http/Controllers/CustomersController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CustomersController extends Controller

{

public function index()

{

return view('customers.index');

}

public function create()

{

//

}

public function store(Request $request)

{

//

}

public function show($id)

{

//

}

public function edit($id)

{

//

}

public function update(Request $request, $id)

{

//

}

public function destroy($id)

{

//

}

}

**resources/views/customers/index.php**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Codecasts - Laravel - Básico</title>

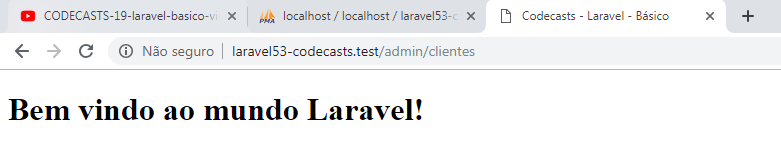
</head>

<body>

<h1>Bem vindo ao mundo Laravel!</h1>

</body>

</html>



# Aula 20 - Views - Blade

**app/Http/Controllers/CustomersController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Customer;

class CustomersController extends Controller

{

public function index()

{

$customers = Customer::all();

return view('customers.index')->with(compact('customers'));

}

}

**resources/views/customers/index.blade.php**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Codecasts - Laravel - Básico</title>

</head>

<body>

<h1>Bem vindo ao mundo Laravel!</h1>

@foreach($customers as $customer)

<p>{{$customer->name}}</p>

@endforeach

</body>

</html>



# Aula 21 - Bootstrap

**app/Http/Controllers/CustomersController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Customer;

class CustomersController extends Controller

{

public function index()

{

$customers = Customer::all();

return view('customers.index')->with(compact('customers'));

}

public function edit($id)

{

return "Edit customer $id";

}

public function update(Request $request, $id)

{

//

}

public function destroy($id)

{

return "Delete customer $id";

}

}

**resources/views/customers/index.blade.php**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Codecasts - Laravel - Básico</title>

<link rel="stylesheet" type="text/css" href="{{'/css/app.css'}}">

</head>

<body>

<div class="container">

<h1 style="margin: 15px 0px" class="text-center">Lista de clientes</h1>

<table class="table bordered table-striped">

<thead>

<th>Id</th>

<th>Name</th>

<th>City</th>

<th>State</th>

<th>Birthdate</th>

<th class="text-center">Special Customer</th>

<th>Ações</th>

</thead>

<tbody>

@foreach($customers as $customer)

<tr>

<td>{{$customer->id}}</td>

<td>{{$customer->name}}</td>

<td>{{$customer->city}}</td>

<td>{{$customer->state}}</td>

<td>{{$customer->birthdate}}</td>

@if($customer->special\_customer == 1)

<td class="text-center">Yes</td>

@else

<td class="text-center">No</td>

@endif

<td>

<a style="padding: 5px;" class="btn btn-xs btn-primary" href="/admin/clientes/{{$customer->id}}/edit"> Edit</a>

<a style="padding: 5px;" class="btn btn-xs btn-danger" href="/admin/clientes/{{$customer->id}}"> Delete</a>

</td>

</tr>

@endforeach

</tbody>

</table>

</div>

</body>

</html>

