[Populando sua Base de Dados com Factory | by Gabriel Albuquerque | Medium](https://g4br.medium.com/populando-sua-base-de-dados-com-factory-8e46294a891)

**requisitos: php 7 ou superior, composer update**

INSTALAÇÃO via composer:

- composer global require laravel/installer

- composer require laravel/installer

DESINSTALAÇÃO DO LARAVEL via composer:

- composer global remove laravel/installer

**SERVER DO LARAVEL**:

- php artisan serve

https:\\127.0.0.1:8000

**CRIAÇÃO DE PROJETO:**

-composer new laravel/laravel

**-composer create-project --prefer-dist laravel/laravel marketplace\_l6 "6.\*"**

(para instalar versão apartir da 6);

**COMANDOS:**

Migrations:

(possibilidade de criação de novas tabelas, adicão de indices cria o versionamento da criação da base de dados)

-php artisan migrate:install (verificar se está conectado ao DB)

create-table-migrate sucessfully.

(cria na nossa base: )

-php artisan make: [nome do arquivo de migração] (cria o arquivo)

NA PASTA: database/migrations

-php artisan migrate (ele roda as migrações e salva os registros na table migration)

-----------------------------**ROLLBACK**---------------------------------

php artisan migrate:rollback remove a nossa migração.

adicionava a tabela produto, ele remove a tabela.

Podemos rodar novamente: PHP Artisan migrate;

O parametro Rollback, aceita voltar steps como parametro:

php artisan migrate:rollback --step=3;

volta 03 passos;

-----------------------**APAGA TODAS AS TABELAS E REFAZ**---------

- php artisan migrate:fresh

NOTA: se tem tabelas na sua base, que não foram criadas com migration, ele apaga os seus dados.

----------------------------------**REFRESH**-----------------------------------------

- php artisan migrate:refresh

NOTA: trabalha em cima das migrações registradas, re-executando.

-----------------------------**O que são SEEDS**------------------------------------

As SEEDS permitem que crie dados para nossas tabelas, pra fazer testes das nossas aplicações.

ex: criar usuarios produtos.

criar uma classe seeders

php artisan make:seeder UserTableSeeder;

criaremos uma classe na seeder

NOTA: com esse comando podemos executar as seeders quantas vezes quisermos:

php artisan migrate:refresh --seed

Ele faz o reverso e todas as migrações e adicionar as seeds.

------------------------------**CONHECENDO AS FACTORYS**----------------------

-> as Factorys são auxiliares para nossas SEEDS.

-> As factorys Definem Modelos para que você possa fazer uma criação, de uma massa grande de dados

factory(\app\User::class, 40)->create() chamando o metodo create

->Dessa forma conseguimos criar 40 usuarios utilizando as factorys:

php artisan migrate:refresh --seed;

ou

php artisan migrate:seed

-------------------MAIS COMANDOS E CONCLUINDO MIGRATIONS-----

resetar todas as migrações:

- php artisan migrate:reset (faz o Rollback de tudo);

Posso fazer um refresh passando passos (steps)

- php artisan migrate:refresh --step=2

**Models:**

**Models User** – Dentro da pasta **App**

Nota, que posso iniciar a criação dos meus Models e chamar o nome do meu model:

php artisan make:model Product

[ele irá criar na pasta App um arquivo chamado: Product.php]

Quais são as convenções do Laravel para esses dois Models:

**Conhecendo o ORM- Object Ordertion Mapping**

ORM e um cara que traduz, uma Base Relacional (nosso banco de dados Mysql) para um ponto que orientado a objetos que é o nossos Models.

Poderemos buscar dados, salvar dados.

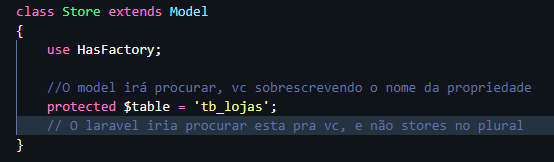
**Eloquent -> e o cara que faz isso pra gente**.

Basicamente como o Model se traduz referente a nossa base de dados, definimos nossa tabela stores.

O laravel tem uma convenção de interpretação desse model, ele pega o nome do model (store) -> ele irá procurar este model automaticamente no plural lá no banco de dados.

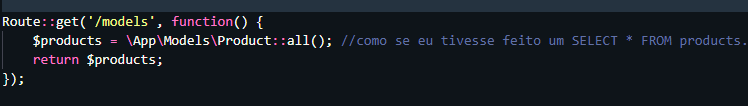
* Se eu quiser modificar o nome da tabela:

Poderemos chamar uma propriedade protegida, chamada $table e definir o nome da tabela:



**ROUTES:**

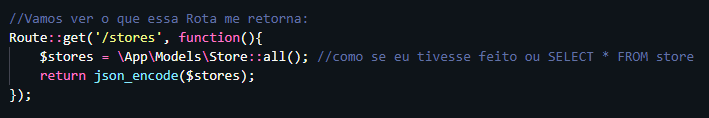
Adicionando uma rota, para visualizarmos o Model:



Fazendo a alteração da Route e acessando 127.0.0.1/models

Vai devolver um array = [ ] vazio, pois não temos nada dentro da tabela product.

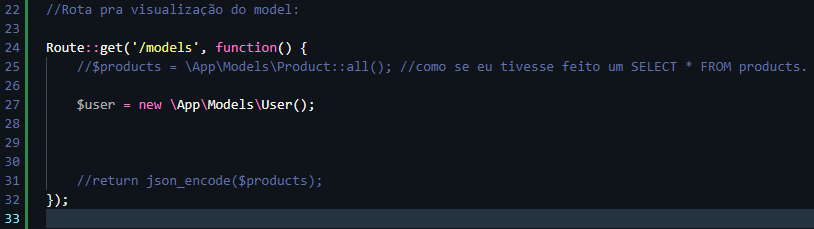
Se Adicionarmos outra Route, para lermos o Models Store, que já possui alguns dados que colocamos com seeds, inserimos dados nele.



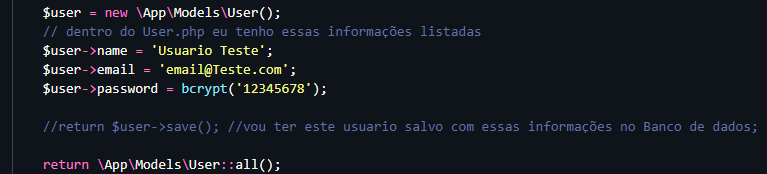
Active Record: (Padrão de Projeto)

Declarar uma variável $user e pegar a instancia do meu USER

Chamando diretamente a instancia do nosso user:



O active Record ele me permite trabalhar dentro do Laravel as colunas do **Banco de Dados** como se fossem **Atributos** da minha classe:



Note que irá me retornar um array como consta abaixo:



Atualização de dados:

O assim como o método ::all e equivalente a um SELECT \* FROM Products

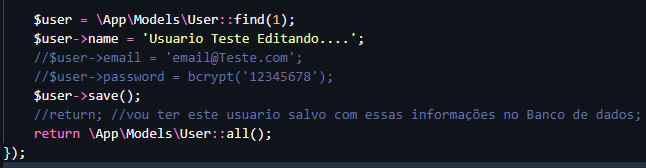
Podemos utilizar uma função chamada find();

$user = \App\User::Find(1); neste caso, quero que recupere o Id 1

$user->name = ‘Usuario Teste Editando’;

Oha o que o método SAVE irá fazer:

$user->save();



Quando criamos uma instancia nova $user = \App\Models\User com o método save()

**$user->save() *você irá criar o usuário no Banco de Dados;***

**$user = \App\Models\User::find(1); Por um método de Busca do Eloquent**

**Find( ) e ele espera o ID da linha que queremos, eu Atualizo as informações do Usuário.**

Essa e uma das formas do Active Record de adicionar dados no Banco, ou Atualizar as informações.

**QUERYS NO ELOQUENT**