Recapitulando aula passada ...

Vimos que em cada rota, como funciona a ordem que o framework vai entender. Ele vai acessar a nossa rota através do arquivo web.php na passa “routes”, onde estarão as rotas da web. Ele vai ver que eu peguei uma rota e vai acessar essa rota, pra acessar essa rota eu vou lá no controlador dessa rota, vou acionar a função e nessa função eu vou fazer o que eu quero que aconteça nessa rota.

Esse controlador vai ser o que de fato vai controlar, ele que vai intermediar o que o usuário está vendo, com o banco de dados, ele que irá cadastras os dados inseridos pelo usuário no banco de dados.

**Cadastrando clientes**

Criando tela de cadastro e Formulário.

Vamos criar uma nova rota para cadastramento de clientes e um novo controlador para o cadastro de clientes.

Para criar o controlador usaremos o comando: php artisan make:controller ClienteController

Que estará na pasta app\http\controller.

O caminho da rota será:

Route::get('/cliente/cadastrar',[ClienteController::class, 'create']);

Sempre lembrando de especificar em web.php o caminho que nesse caso é :

use App\Http\Controllers\ClienteController;

Como padrão do laravel o nome da minha rota será “create”, pois esta rota estará mandando para um cadastro.

Então esse Create está no meu ClienteController, e vamos criar nossa public function.

<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class ClienteController extends Controller

{

    public function create(){

        return view('cliente.cadastrar');

    }

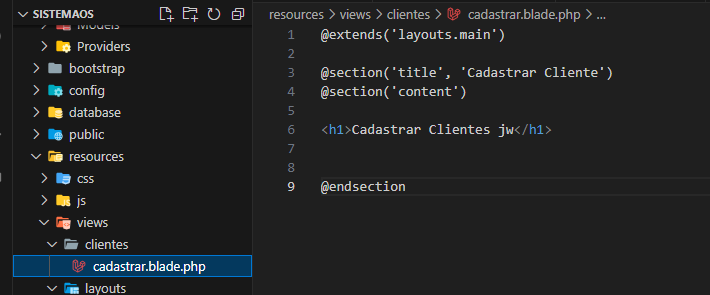
}

Min 12.45

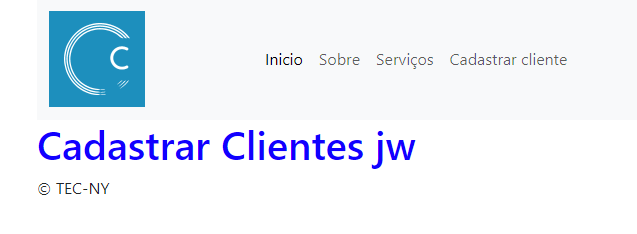
Então será redirecionado para a view cliente.cadastrar

Que estará na pasta que criaremos em view, na pasta clientes e o arquivo cadastrar.blade.php

Com o conteúdo h1 para visualizarmos na tela se esta funcionado.



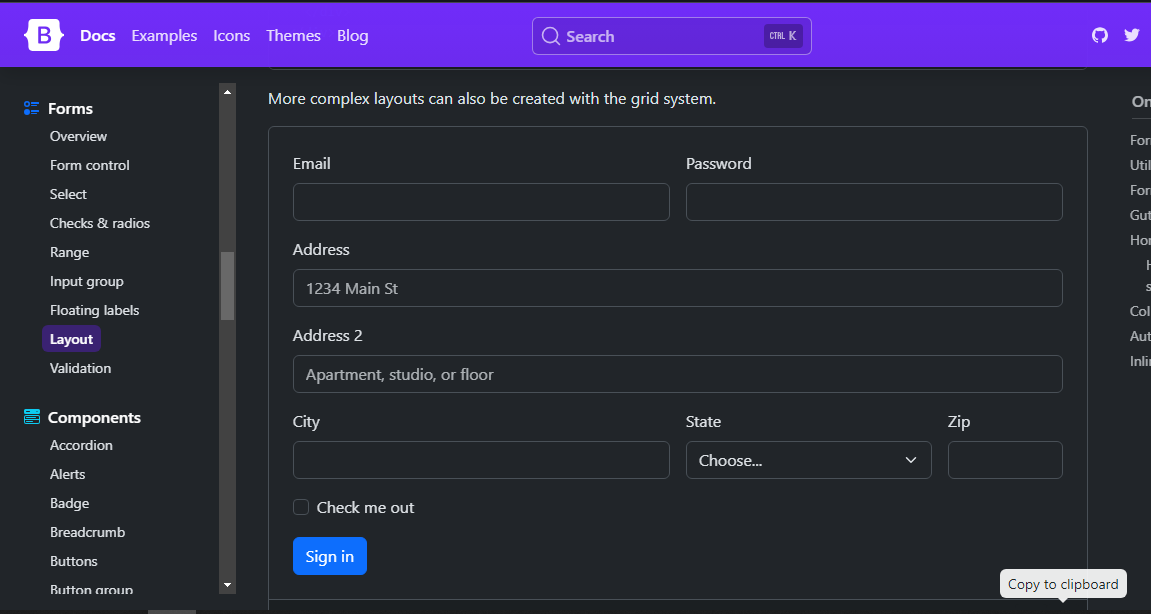
E como podemos ver abaixo funcionou o teste.



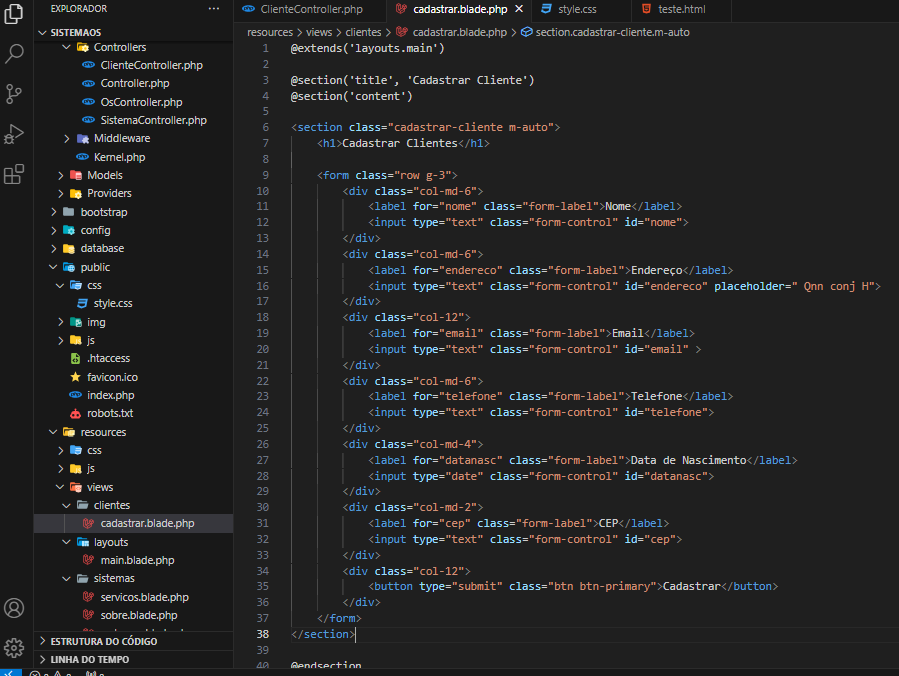
Parei no minuto 14:26

**CRIANDO FORMULARIO**

Formulário usando bootstrap e o que ele irá ter como: Nome, Endereço, Telefone, Email, Datanasc

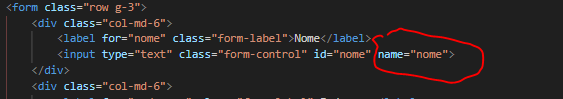


Código abaixo



**Enviando dados para o banco de dados.**

Para enviarmos os dados do nosso formulário para o banco de dados utilizaremos o atributo “name”, que é o que vai ser enviado para o nosso banco de dados, e esse name deve estar com o mesmo nome que iremos declaramos no banco de dados.



Agora iremos enviar esses dados através da action e method.



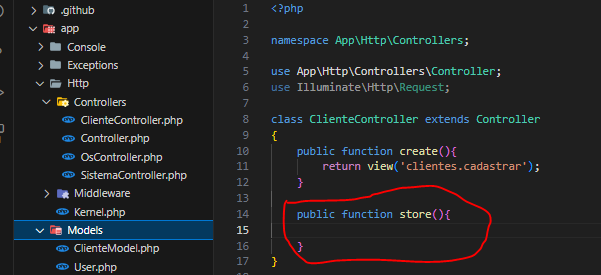
Quando o usuário clicar vai mandar para rota cliente.

**Enviando dados através do método post**.

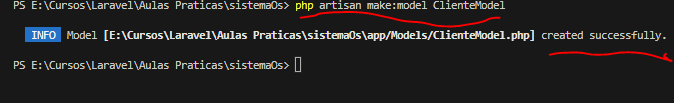
Criaremos uma nova rota com método post para enviar nossos dados, através da classe padrão do laravel chamada “store” que vai fazer o cadastro.



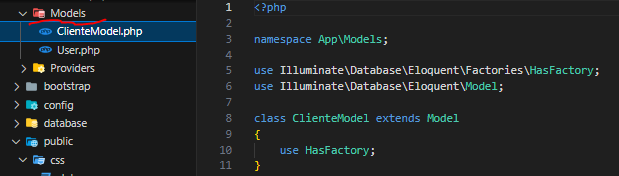
Agora iremos criar uma função em clientecontroller



e em seguida criar nosso cliente.model para estar fazendo a integração com o banco de dados.



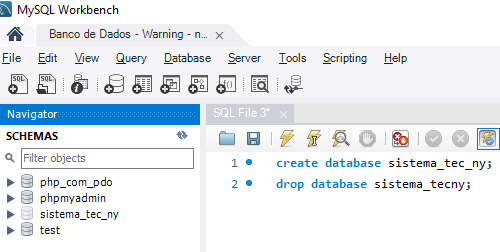
E model criado... Pasta model -> clientemodel.php



E criaremos também o arquivo OsModel que só usaremos, mas pra frente...

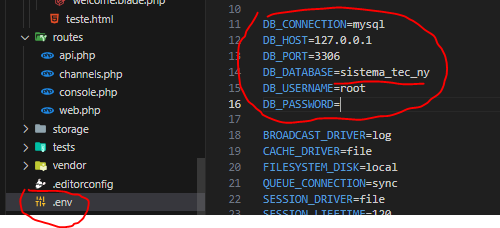
**Criando base de dados**

com o workbench vamos criar nossa base de dado logo abaixos...



E agora vamos linkar nossa base com nosso sistema, como iremos fazer isso?

No arquivo .env do laravel, e vamos renomear o padrão do laravel para o que criamos como “sistema\_tec\_ny”



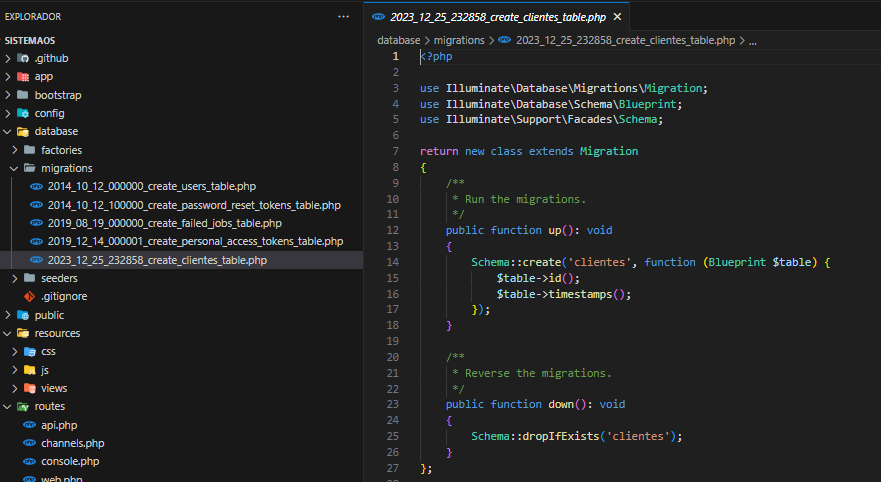
**Migrations**

Agora vamos crias nossas migrations, que é o que vai fazer nossa relação com o banco de dados.

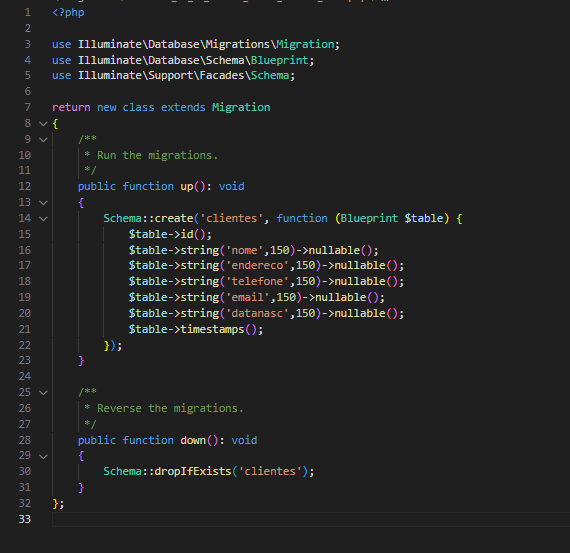
Min 46:55

Com o comando php artisan make:migration create\_clientes\_table

Vai ser criado na pasta database/migrations nossa tabela.

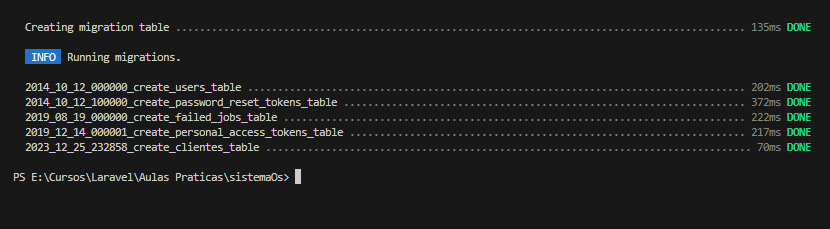


Essa migrations é como eu quero que meu banco de dados se comporte. Abaixo esta as alterações que precisamos para mandar para o banco d dados.

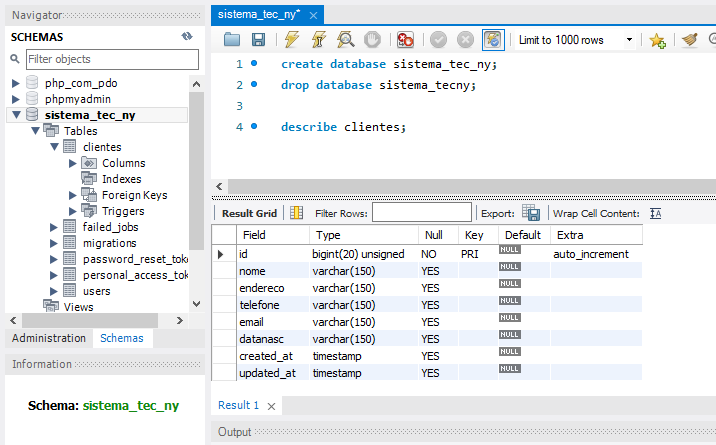


Para ver o status, usamos:

Php artisan migrate:status ou Php artisan migrate



Agora se acessarmos nosso banco de dados, poderemos ver nossa tabela clientes abaixo.



Podemos ver que em datanasc esta com varchar 150 e queremos um date. Então vamos dar um rollback para voltar essa migrate.

Comando -> php artisan migrate: rollback

Agora se voltarmos no banco de dados e usarmos o comando describe clientes; vai aparecer que essa tabela não existe mais.

Outro problema que podemos evitar quanto ao campo data é os erros futuros por conta de localização, hora e fuso horário. Então para evitar isso, vamos fazer o seguinte:

Em App/Models/ClientesModel.php Vamos escrever o seguinte comando:

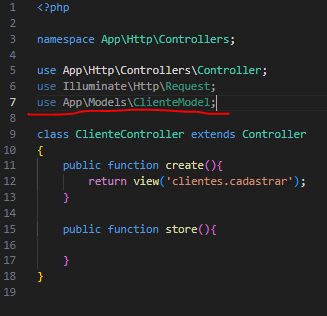
Protected $dates = [‘datanasc’];

E pronto.

Agora vamos podemos dar o comando php artisan migrate e estará tudo certo. Garantimos que não vai haver problemas, mas para frente.

Agora iremos para o nosso controlador dizer o que a função store irar fazer:

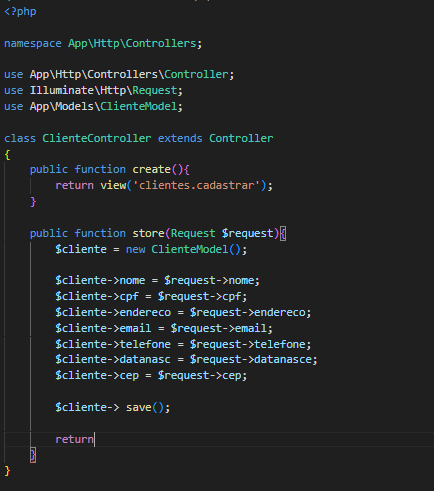
Primeiro iremos acessar o nosso cliente Model:



Esse meu cliente model, está pegando o meu modelo de cliente.

E agora, esse meu store vai fazer que, cada atributo da minha base de dados, vai receber o que o usuário enviar.

Vamos instanciar um novo objeto cliente, ele é um novo cliente model. E vai ficar igual o exemplo abaixo.



@csrf: serve como uma diretiva para mandar o formulário de maneira segura para o banco de dados.



Quando tentarmos fazer o teste de cadastro vai dar um erro porque a tabela foi criada com o nome “Clientes” e no nosso código está clienteModel.

Então iremos apagar e refazer a parte do código e colocar o nome certo da tabela feita no banco de dados.

Começando pelo clienteModel, iremos apagar essa pasta, e criar uma nova como o nome ‘Clientes’ com o comando php artisan make:model Clientes. E alterar o nome errado para o certo em clinteController e o caminha que está sendo herdado de use App\Models\ClienteModel para use App\Models\Clientes;