Apostila de Laravel

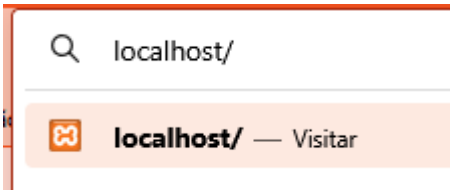
Para entender como o framework Laravel funciona, deve-se entender como o PHP funciona, uma vez que ele é um framework da linguagem de promação PHP.

Para entender melhor como o PHP funciona, considere a seguinte situação, a seguir.

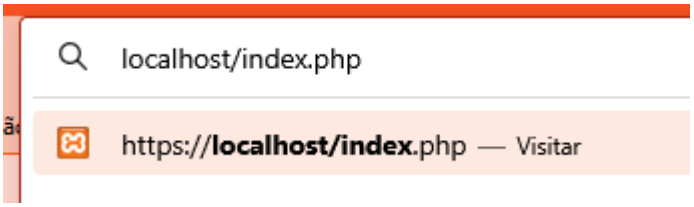
Servidores Web como o PHP procura por padrão um arquivo cujo o nome é index na pasta da aplicação ou em sub pastas. Suponha que a aplicação Xampp tenha acabado de ser instalada. Dessa forma toda vez que for acessada a porta 80 desse servidor e buscará na pasta htdocs um arquivo com o nome index.php, caso o nome do arquivo não tenha sido informado.

Vamos destrinchar isso em pedaços menores.

Ao acessar o endereço, a aplicação encaminhará a solicitação para a pasta htdocs do Xampp.

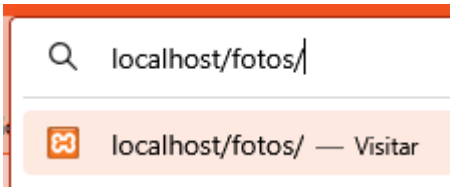


O mais óbvio seria o usuário digitar o endereço completo, incluindo o nome do arquivo:



Mas por que isso não é necessário?

Isso não é necessário, porque a própria a aplicação entenderá que que toda vez que um usuário acessar a aplicação sem informar o nome do arquivo, que deverá procurar um arquivo index.php na pasta em questão. Isso ocorre também em subpastas, como a seguir:

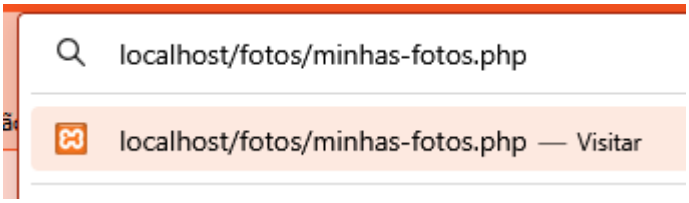


Na imagem acima veja que as barras servem para indicar o nome da pasta. Logo, na pasta principal da aplicação existe uma pasta fotos. E ao acessar a pasta fotos, a aplicação procurará por um arquivo index.php nessa pasta. Caso não tenha

esse arquivo, aparecerá a mensagem de erro 404, indicando que a url não foi encontrada.

Ou seja, embora a url completa seria https://localhost/index,php, a adição do nome do arquivo é facultativa.

Outra forma de acessar uma url, seria da imagem a seguir:



No caso do exemplo acima, deverá existir um arquivo cujo nome seja minhas-fotos.php.

Em suma, essa é a forma de acessar url em PHP.

A url é composta pelo domínio, no exemplo acima é localhost

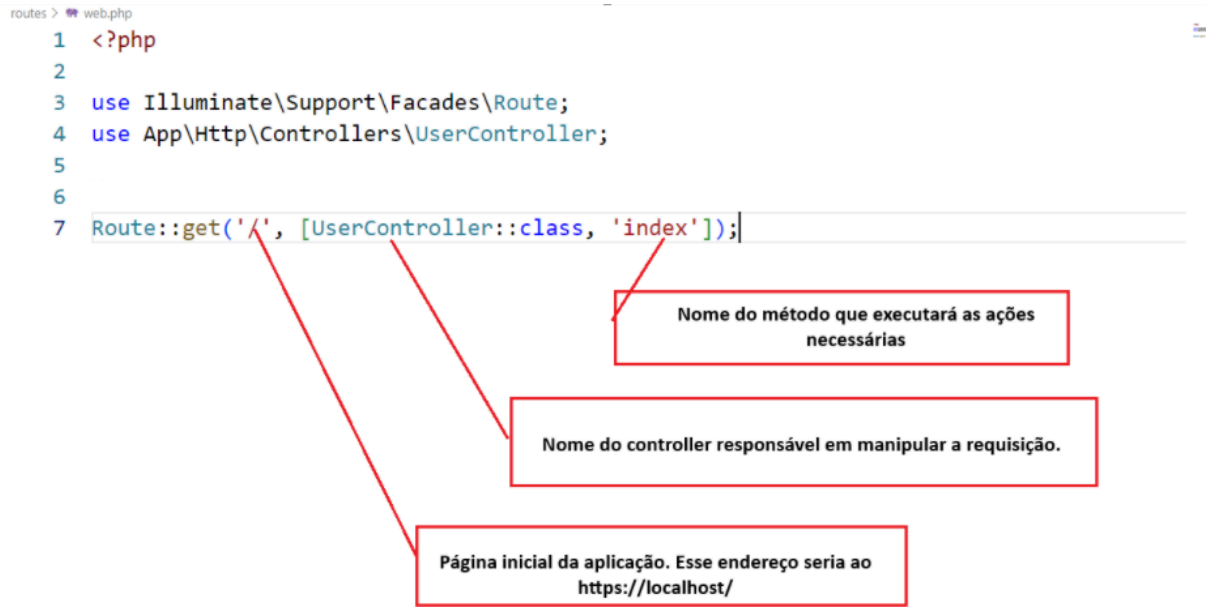
A url é composta por pastas que possa compor a estrutura local da aplicação. No exemplo a pasta tem o nome de fotos.

A url é composta por arquivos que devem ser informados na última parte da url. No caso do arquivo ser index,php, não há a necessidade de informar.

Laravel

Agora que sabemos como o PHP funciona, vamos entender a estrutura do Laravel.

Na pasta principal onde o Laravel é instalado existe um arquivo index.php. Em suma, a configuração desse index.php delega a responsabilidade de encontar os endereços que o usuário digitar ao arquivo web,php. Esse arquivo fica na pasta route/web,php. Ele gerencia o que chamamos de rotas da aplicação.



Dentro do Laravel a imagem acima indica como devem ser criadas as url que receberão as requsições externas da aplicação.

Por exemplo. Suponha que exista a necessidade de criar uma página que terá como responsabilidade listar os usuários da aplicação. Dessa forma será necessário criar mais uma rota externa, seguindo a mesma estrutura indicada acima.

```
Route::get('/usuarios', [UserController::class, listarUsuarios]);
```

Pronto. Você acabou de criar uma rota externa para a sua aplicação onde existirá um método responsável por listar os usuários.

Cada rota terá uma classe controller responsável por receber essas requisições. Afinal de conta, o que seria uma classe controller.

Essas classes são responsáveis por reunir todos os métodos que manipularão as requisições relacionadas aos usuários da aplicação.

E o que são métodos?

Métodos estão na ponta final dessa requisição.

Ao digitar https//localhost/usuarios na aplicação Laravel. o arquivo web.php será acionado e procurará pela rota `'/usuarios'`

Após isso, ele procurará pelo controller responsável em manipular essa requisição. Nesse caso é o `UserController::class.` E por último o controller procurará pelo método responsável em tratar essa requisição, nesse caso é o médoto `'getUsuarios'.`

Passo a passo para criar sua primeira rota:

No arquivo web.php, adicione a rota para listar usuários da aplicação.

```
Route::get('/listar-usuarios', [UserController::class, 'getUsuarios']);
```

Por padrão no topo desse arquivo já foi feito o import da classe Route que é responsável por gerenciar as requisições.
Dessa forma não esqueça de fazer o import da classe UserController.

```
use App\Http\Controllers\UserController;
```

Agora será necessário criar um arquivo na pasta app/Http/Controllers com o nome UserController.php
Por padrão a classe deverá ter o mesmo nome do arquivo.
Coloque o conteúdo abaixo no arquivo recém criado, UserController.php

```
<?php
/*
O namespace serve para indicar para o laravel
o caminho desse arquivo. Assim como você precisa
incluir funcionalidades de outro arquivos em um
arquivo atual, usando a função include, no Laravel
esse import é feito usando a função use.
*/
namespace App\Http\Controllers;
/*
O uso dessa função use é para poder utilizar ass
funcionalidades
do arquivo na página atual. Dessa forma, é necessário indicar
o namespace
e por último indicar o nome da classe
*/
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;
```

```
class UserController extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;


}
```

Agora que o arquivo foi criado com o nome da classe UserController, haverá necessidade de ter um método com o nome indicado no arquivo web.php, a saber, getUsuarios.

Em Laravel, assim como em php, para criar métodos dentro da classe é necessário indicar o controlador de acesso da função além da terminologia function. Nesse caso usaremos public. Logo a estrutura deverá ser:

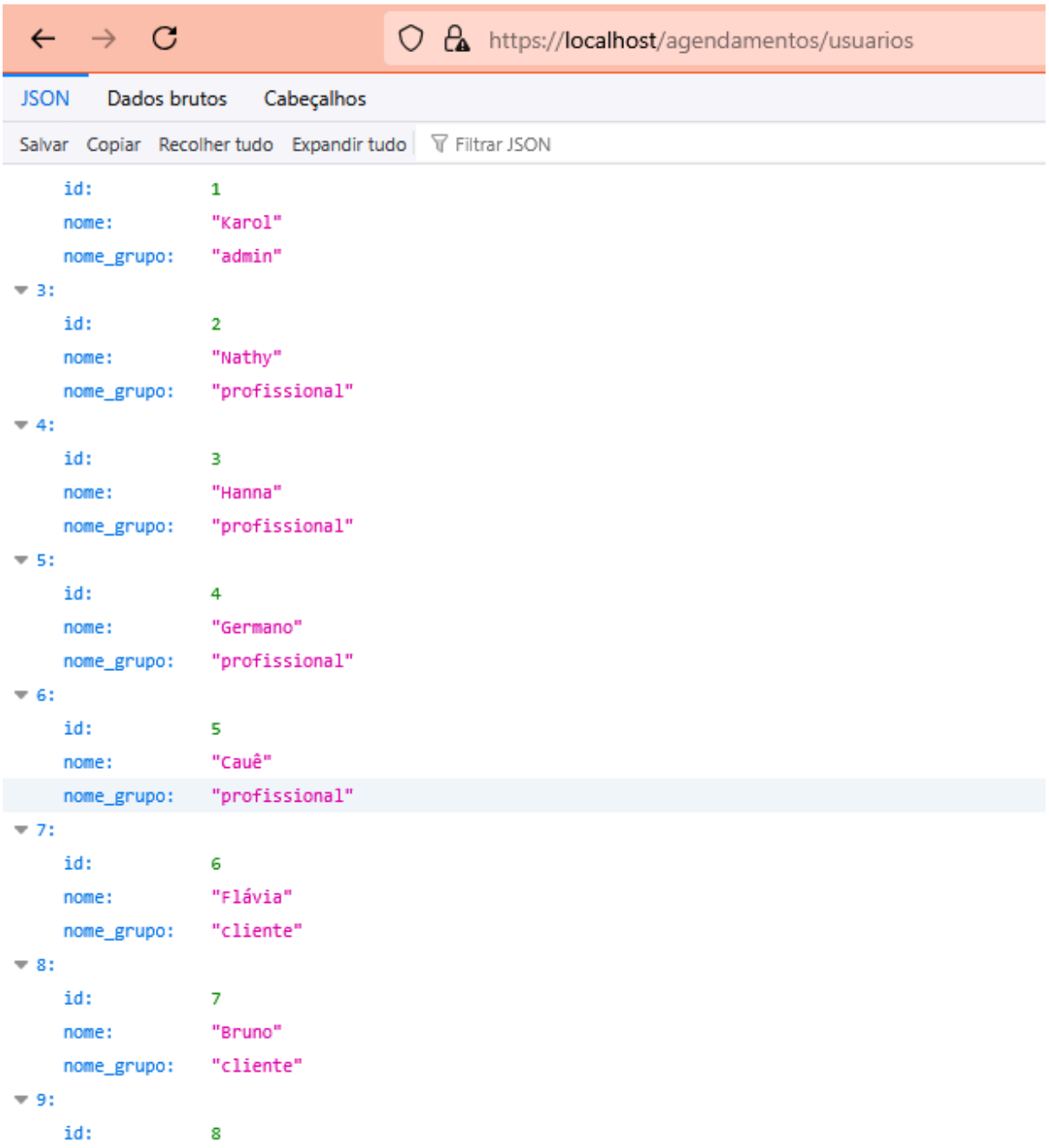
```
public function getUsuarios(){


}
```

Esse método ainda não retorna nada. Deverá ter uma estrutura de dados dados no retorno. Dessa forma adicionaremos um array no retorno:

```
public function getusuarios(){
  return response()->json([
    ["nome" => "Karol"],
    ["nome" => "Nathy chata"],
    ["id" => 1, "nome" => "Karol", "nome_grupo" => "admin"],
    ["id" => 2, "nome" => "Nathy", "nome_grupo" =>"profissional"],
    ["id" => 3, "nome" => "Hanna", "nome_grupo" => "profissional"],
    ["id" => 4, "nome" => "Germano", "nome_grupo" => "profissional"],
    ["id" => 5, "nome" => "Cauê", "nome_grupo" => "profissional"],
    ["id" => 6, "nome" => "Flávia", "nome_grupo" => "cliente"],
    ["id" => 7, "nome" => "Bruno", "nome_grupo" => "cliente"],
    ["id" => 8, "nome" => "Josué", "nome_grupo" => "cliente"],
    ["id" => 9, "nome" => "Matheus", "nome_grupo" => "cliente"],
    ["id" => 10, "nome" => "Ludi", "nome_grupo" => "cliente"]
        ]);
    }
```

Ao digitar localhost/usuarios deverá aparecer a lista de usuários.



Configurando o model

O model tem a resposabilidade de associar um dado model a uma tabela no banco de dados. Ou seja, para cada tabela deverá constar um model que terá a função de carregar as infomações do banco de dados, assim como a atualização, inserção ou a exclusão de dados.

Na pasta app/models crie um arquivo com o nome User.php

Coloque o conteúdo abaixo nesse aquivo recém criado.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{



}
```

Repare que o nome da classe é o mesmo do arquivo, User. Não esquecer de incluir o par de chaves, onde ficará o conteúdo dessa classe.

Após isso precisamos configurar esse model para se conectar com a tabela correspondente no banco de dados.

Precisamos indicar o nome da tabela para classe User. Dessa forma inclua a linha abaixo que identificará o nome da tabela.

```php
protected $table = 'usuarios';
```

Agora vamos informar as colunas dessa tabela, iniciando pela a chave primária. Sendo assim, inclua a linha abaixo logo depois da linha incluída anteriormente.

```php
protected $primaryKey = 'id_usuario';
```

Continuando, precisamos passar as demais colunas para o Laravel saber quais são as colunas de cada tabela. Acrescente a linha abaixo:

```php
protected $fillable = [


];
```

Essa variável acima é do tipo array. Dentro desse array informe todas as demais colunas, além da primary key, que já foi informada da linha anteriormente configurada.

```
protected $fillable = [
    'nome',
    'email',
    'id_grupo',
    'data_nascimento',
];
```

Pronto. Agora a sua aplicação Laravel já sabe o nome da tabela e o nome de cada coluna da tabela correspondente.

Uma das colunas dessa tabela é o índice oriundo de uma outra tabela, indicando a relação da tabela usuarios ea tabela grupos.

Para entender a relação entre User e Grupo, devemos entender que ao criar um usuário, um dos campos a ser preenchido será o id_grupo. Mas essa informação não será qualquer. O id_grupo virá da tabela grupos, ou seja, para ser informado esse dado, deverá exitir um registro na tabela de grupos previamente.

Vamos entender isso melhor.

Suponha que você já tenha criado 4 registro na tabela grupo, onde:

1 - admin

2 - profissional

3 - cliente

4 - gerente

Para criar um registro usuário, um dos campos a ser preenchido será o id_grupo. Ou seja, o sistema deverá listar os grupos previamente criados e será escolhido um dos valores acima. E não poderá ter um usuário cujo id_grupo seja 5 ou mais, uma vez que só tem 4 registros.

Fica mais fácil para entender que o usuário pertencerá a um grupo e o grupo poderá ter vários usuários, já que o id_grupo 3, por exemplo, poderá ser utiizado por diversos usuários.

Precisamos informar para o Laravel essa relação para que seja possível que ele consiga trazer o nome do grupo ao invés de trazer somente o índice.

Na classe User precisamos incluir a função que ficará responsável em associar o usuário ao seu grupo.

Inclua esse script no arquivo User.php

```
public function grupo()
{
  return $this->belongsTo(Grupo::class, 'id_grupo', 'id_grupo');
}
```

Acima é usada a terminologia belongsTo para indicar que o usuário pertence ao Grupo.

Dessa forma crie um arquivo com o nome Grupo.php para finalizar essa configiração. Coloque o conteúdo abaixo no arquivo.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;


class Grupo extends Model
{

    protected $table = 'grupos';

    protected $primaryKey = 'id_grupo';

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
```

```
    protected $fillable = [
        'nome_grupo',
    ];

    public function usuario()
    {
        return $this->hasMany(User::class, 'id_grupo', 'id_grupo');
    }


}
```

Repare que no arquivo Grupo.php foi colocado um método parecido, no entanto, ele utiliza a terminologia hasMany para indicar que o grupo poderá ser utilizado em vários usuários.

Inserindo itens no banco de dados

Agora vamos criar no UserController um método para salvar dados no banco de dados.

Crie uma rota que será utilizada para incluir um usuário na tabela de usuarios. Lembre que toda ação na aplicação começa na configuração de rotas no arquivo web.php.

```
Route::get('/create-usuario', [UserController::class, 'createUsuario']);
```

Agora no controller crie um método com o mesmo nome indicado acima, createUsuario

Dentro desse método coloque o script responsável por criar um usuário.

```
public function createUsuario(Request $request)
    {
        $usuarioCriado = User::create([
            'name' => $request->name,
```

```
            'email' => $request->email,
            'id_grupo' => $request->id_grupo,
            'data_nascimento' => $request->data_nascimento,
        ]);

        return response()->json($usuarioCriado);
    }
```

O método create logo após o nome da classe User é responsável por salvar novos dados na base de dados.

Como estamos utilizando o método get temos que enviar as chaves e valores na própria url.

Não esqueça de importar as classes User e Request que serão utilizadas nessa página.

```
use Illuminate\Http\Request;
use App\Models\User;
```

Dessa forma teremos que seguir a seguinte estrutura:

localhost/agendamentos/create-usuario?name=Camila&email:camila@mail.com&id_grupo=3&data_nascimento=2000-01-06

Observe os seguintes pontos na url.

Após a rota create-usuario foi colocado um ponto de interrogação. Isso indica que as informações após são parâmetros e não parte do endereço da página.

Deverá seguir o seguinte: chave=valor separados por & (e comercial). Dessa forma a chave deverá ser exatamente o texto no model de User e o valor após o sinal de igual o conteúdo a ser inserido.

Faça o mesmo para todos dos models criados até agora.