1. Authentication

O Laravel pode facilitar muito a vida do programador quando o projeto precisar de um controle de usuários e de autenticação.

Ele já vem de fábrica com diversos elementos (Models, Controlers e Views) específicos para implementar usuários no seu projeto.

Mas para informar ao framework que o seu site vai mesmo fazer uso desse recurso basta digitar o seguinte comando (dentro da pasta do seu projeto):

```
php artisan make:auth
```

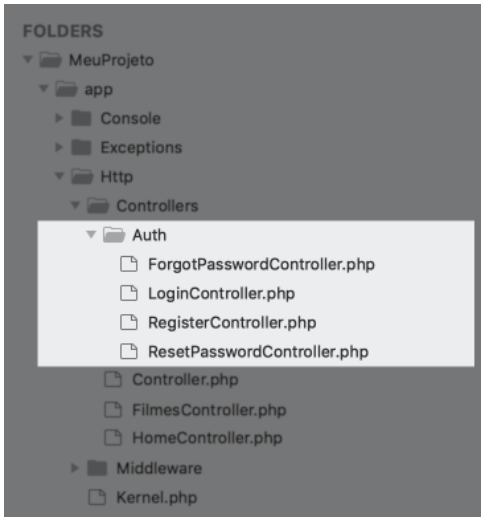
Ao executar essa linha de comando o `artisan` vai construir e preparar mais algumas coisas.

Primeiro no arquivo de rotas (`routes/web.php`):

```php
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
```

Veja que automaticamente duas novas linhas são incluídas no arquivo.
A primeira `Auth::routes()` cria as rotas necessárias para que o usuário possa a partir de agora por exemplo visualizar a página de login, de renovação de senha e de registro.
E a segunda é apenas uma nova página `home` para apresentar o usuário já logado no seu sistema.

Agora vamos conferir os controllers (`app/http/controllers`):



Todos esses controladores de dentro da pasta Auth foram criados automaticamente exatamente para implementar a renovação de senha, login e cadastro do novo usuário.

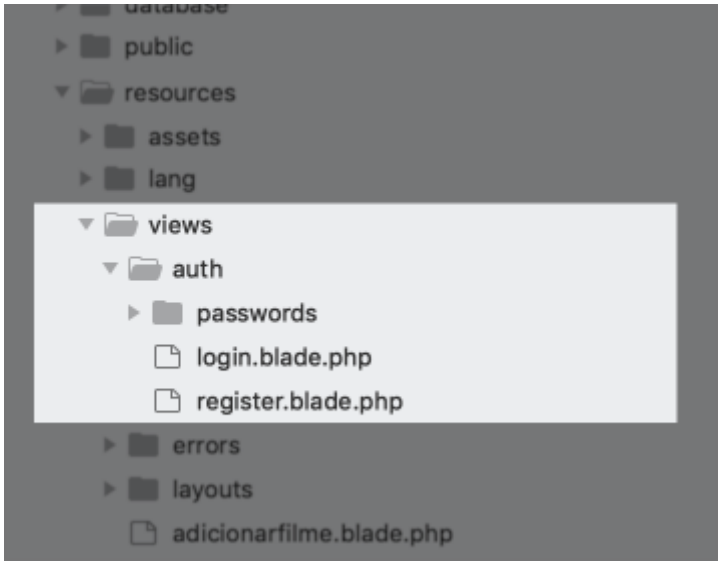
Veja agora os models (dentro da pasta `app`):

```php
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];
}
```

Na verdade esse arquivo `User.php` já existia quando instalamos o Framework.
E agora sabemos porque =)

Ele é um modelo como qualquer outro, que representa a tabela `users` lá do banco de dados.

E novas views também foram criadas (`resources/views`):



Todas as views criadas dentro da pasta auth servirão para exibir os formulários de login, de registro do novo usuário e também de renovação de senha.

Você pode, é claro, modificar e traduzir essas views como achar melhor.
São arquivos do Blade e representam exatamente a página HTML que o usuário visualiza.

2. **Middleware**
   app/Http/Middleware

Lembra quando falamos que o fluxo das chamadas no Laravel?

*A **Rota** chama o **Controlador** e o **Controlador** chama a **View***

E se for necessário obter alguma informação do banco de dados:

*O **Controlador** chama o **Modelo***

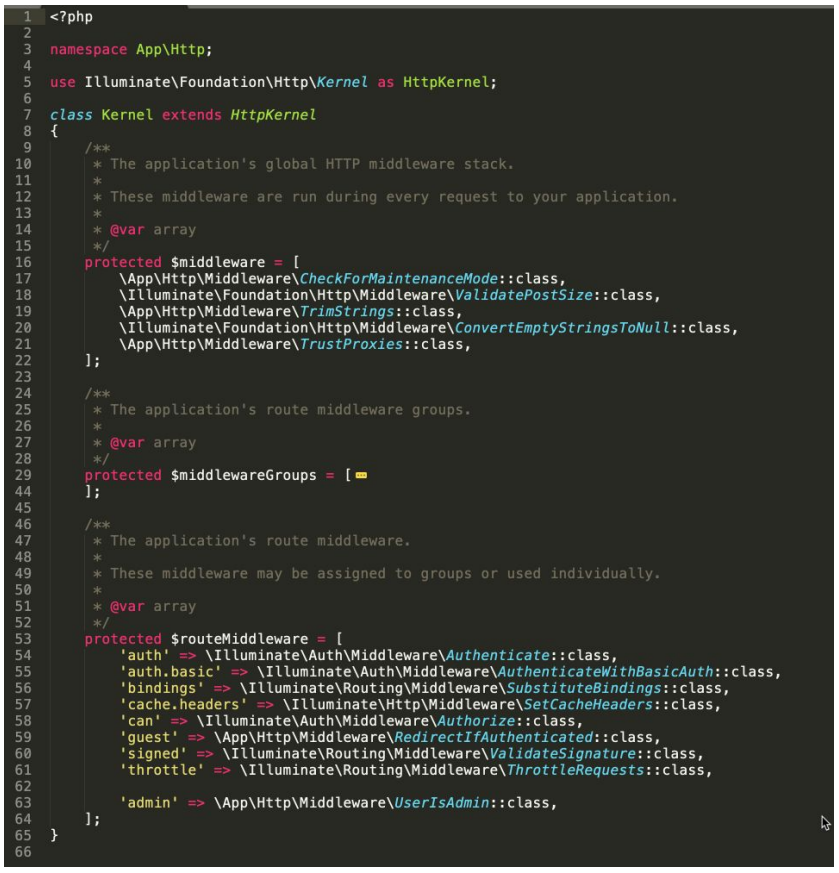
Na verdade, entre a Rota e o Controlador existe o Middleware =)

Ele é responsável por interceptar uma chamada feita para uma rota e fazer um processamento de alguma coisa antes de encaminhá-la para o controlador.

O Middleware é muito utilizado para verificar se o usuário está logado no sistema na hora que acessou uma rota, caso contrário força ele a ser direcionado para a tela de login.
Alguns sites usam o Middleware para forçar o usuário a aceitar os termos de uso, por exemplo.

Existem middleware globais e específicos para rotas.

O Laravel vem de fábrica com muitos já implementados.
E todos estão registrados no arquivo `app/Http/Kernel.php`.



```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array
     */
    protected $middleware = [
        \App\Http\Middleware\CheckForMaintenanceMode::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
        \App\Http\Middleware\TrustProxies::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array
     */
    protected $middlewareGroups = [
    ];

    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used individually.
     *
     * @var array
     */
    protected $routeMiddleware = [
        'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

        'admin' => \App\Http\Middleware\UserIsAdmin::class,
    ];
}

```

Os Middlewares globais são executados em cada pedido feito para o aplicativo, sem importar o caminho acessado.
Eles são declarados no array `$middleware` do arquivo `Kernel.php`.
E os Middlewares específicos por rota são aqueles que, como indica o nome, são executados apenas nas rotas especificadas.
Eles são declarados no array `$routeMiddleware` dentro do arquivo `Kernel.php`.

Se achar necessário, você pode criar um novo Middleware utilizando a linha de comando:

```
php artisan make:middleware TermosAceitos
```

Depois de executar este comando o novo arquivo será gerado na pasta app/Http/Middleware.
E depois é só registrá-lo no arquivo `Kernel.php`, ou no array com os middlewares globais ou no array com os middlewares de rota =)

Sempre que quiser "proteger" a sua rota com um Middleware basta incluir o método `middleware()` no final da sua rota:

```
// Permite acessar essa rota apenas se o usuário estiver logado
Route::get('/filmes/exibir', 'FilmesController@adicionarFilme')->middleware('auth');
```