Curso de PhpUnit com Laravel

Versão 1.0

Prof. Glauco Todesco

Aula00 – Ambiente

1. Laravel: 8.1.0
2. Php: 7.4.9
3. Composer: 1.10.10
4. MySQL:
5. Visual Studio Code
6. XAMMP:
7. PhpUnit: 9.4.1

Links:

PhpUnit -> <https://phpunit.de/>

<https://laravel.com/docs/8.x/testing>

Aula01 – Criando um projeto

1. Na linha de comando crie um projeto Laravel:

**create-project laravel/laravel cursophpunit**

1. Acesse a pasta:

**cd cursophpunit**

1. Abra o projeto no VSCode:

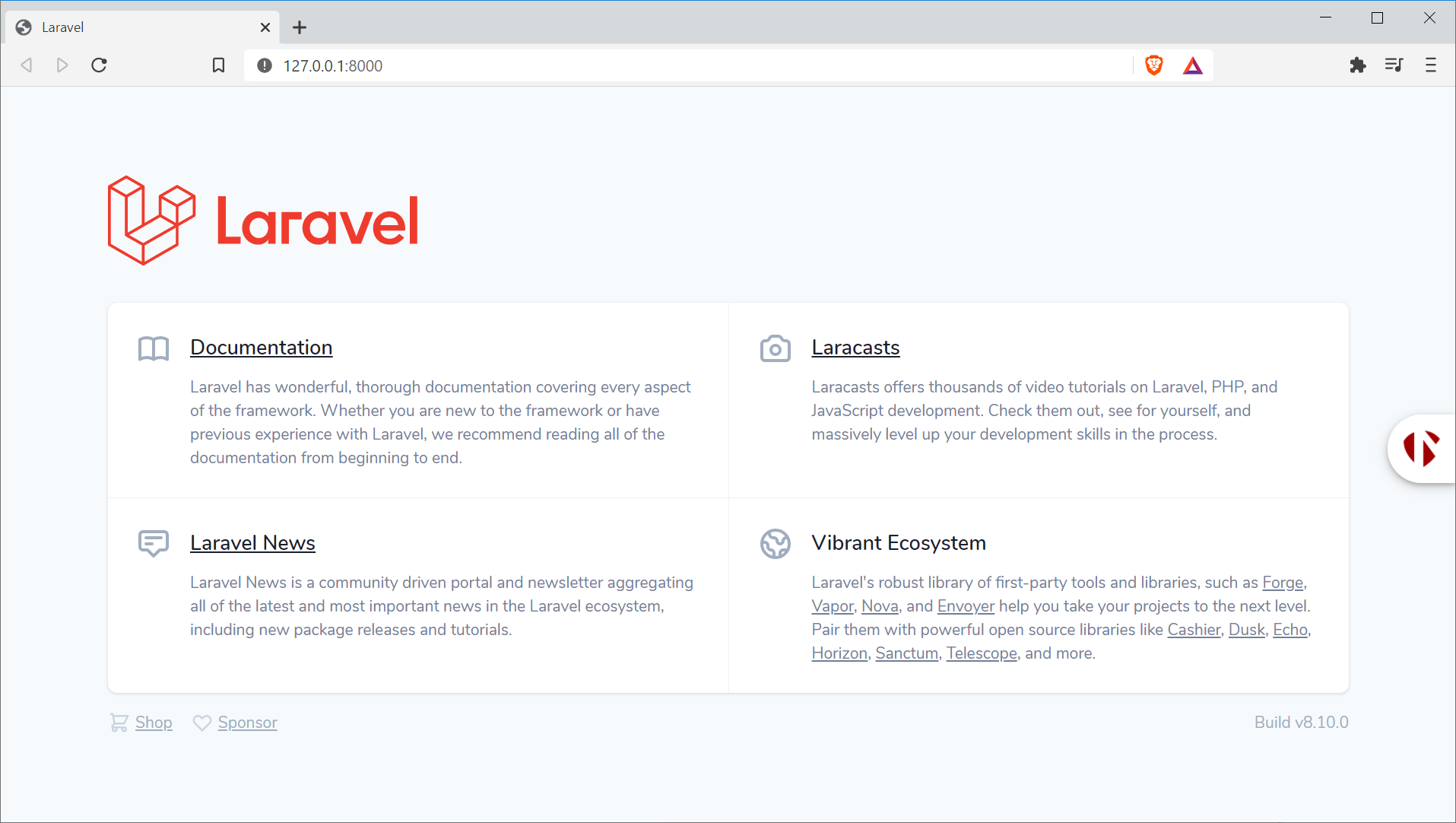
**code .**

1. Abra um terminal no VSCode e execute a aplicação com o servidor do artisan:

**php artisan serve**

1. Abra a aplicação pelo navegador:

**http://127.0.0.1:8000/**



Aula02 – Analisando a estrutura do projeto

1. O Laravel usa o PHPUnit para realizar os testes automatizados.
2. A configuração do PHPUnit é realizada através do arquivo **phpunit.xml** localizado na raiz do projeto.

<?xml version="1.0" encoding="UTF-8"?>

<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:noNamespaceSchemaLocation="./vendor/phpunit/phpunit/phpunit.xsd"

         bootstrap="vendor/autoload.php"

         colors="true"

>

    <testsuites>

        <testsuite name="Unit">

            <directory suffix="Test.php">./tests/Unit</directory>

        </testsuite>

        <testsuite name="Feature">

            <directory suffix="Test.php">./tests/Feature</directory>

        </testsuite>

    </testsuites>

    <coverage processUncoveredFiles="true">

        <include>

            <directory suffix=".php">./app</directory>

        </include>

    </coverage>

    <php>

        <server name="APP\_ENV" value="testing"/>

        <server name="BCRYPT\_ROUNDS" value="4"/>

        <server name="CACHE\_DRIVER" value="array"/>

        <!-- <server name="DB\_CONNECTION" value="sqlite"/> -->

        <!-- <server name="DB\_DATABASE" value=":memory:"/> -->

        <server name="MAIL\_MAILER" value="array"/>

        <server name="QUEUE\_CONNECTION" value="sync"/>

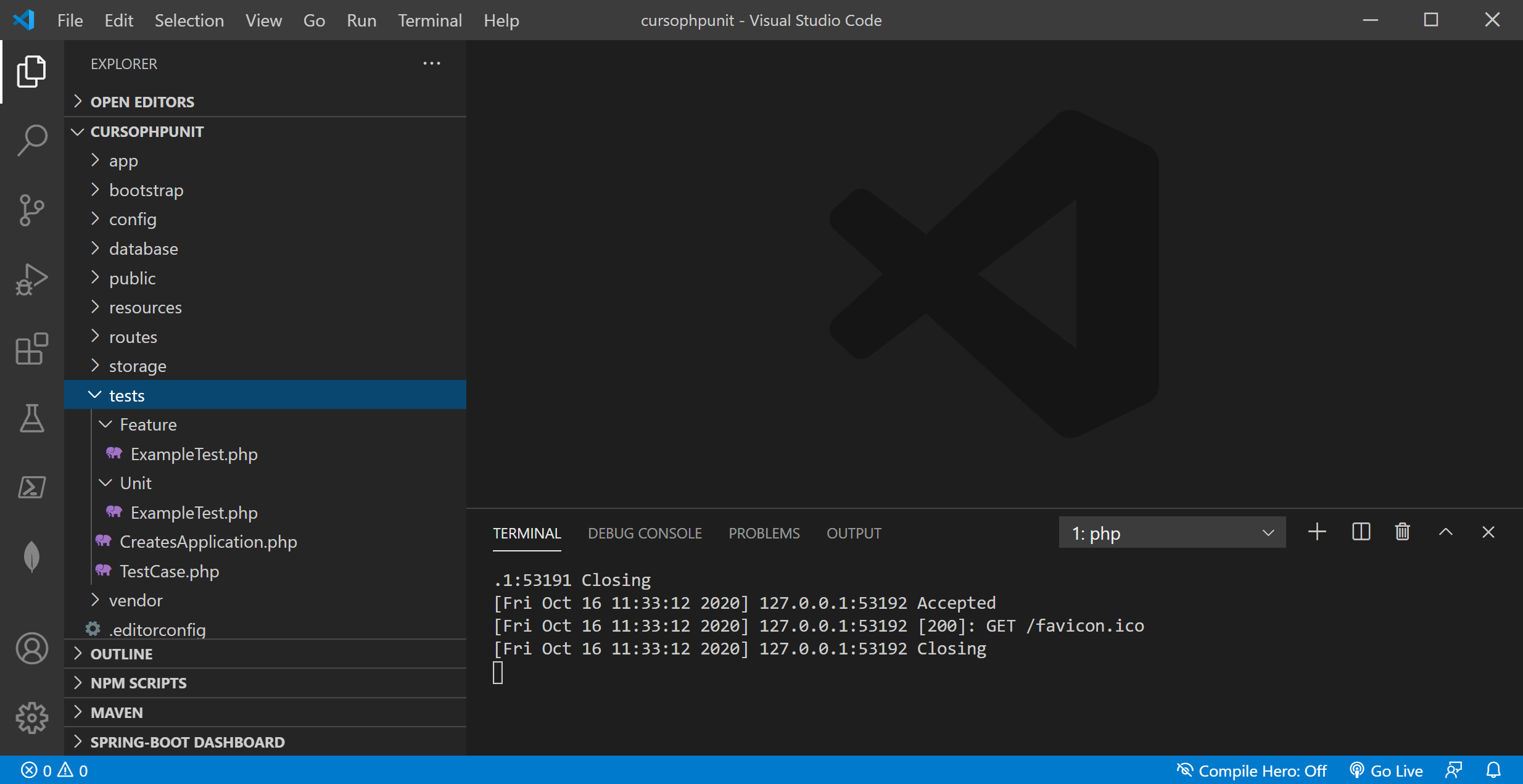
        <server name="SESSION\_DRIVER" value="array"/>

        <server name="TELESCOPE\_ENABLED" value="false"/>

    </php>

</phpunit>

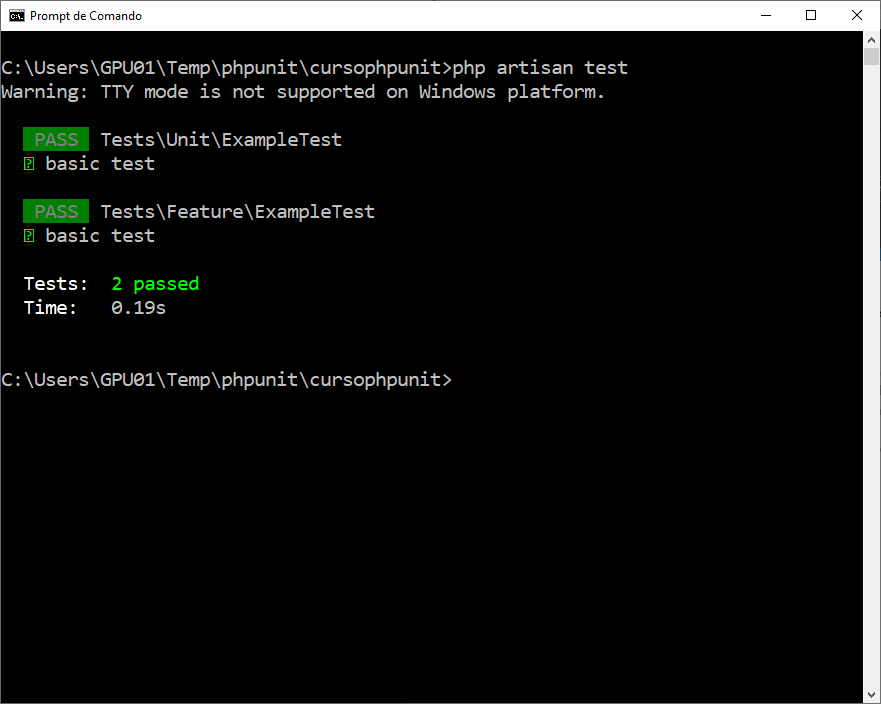
1. Todos os testes deverão ser colocados dentro da pasta **tests** do projeto:



1. O projeto já vem com alguns exemplos criados. Abra e veja os exemplos.
2. A pasta de testes contém dois diretórios: F**eature** e **Unit**. Testes localizados dentro da pasta **Unit** são testes de unidades. Uma unidade de teste é o menor teste que podemos fazer dentro de um projeto, geralmente testando um método de uma classe. Testes localizados dentro da pasta **Feature** são testes mais complexos envolvendo a interação de diversos objetos, requisições HTTP, arquivos JSON e acesso ao banco de dados.
3. Para executar os testes de exemplos, execute o seguinte comando:

**vendor\bin\phpunit** ou **php artisan test**

1. Uma saída parecida com essa será apresentada, no qual os dois testes existentes passaram



1. Vamos fazer uma alteração no teste Feature, no qual o status esperado seja o 404 (Not Found).

<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;

use Tests\TestCase;

class ExampleTest extends TestCase

{

    /\*\*

     \* A basic test example.

     \*

     \* @return void

     \*/

    public function testBasicTest()

    {

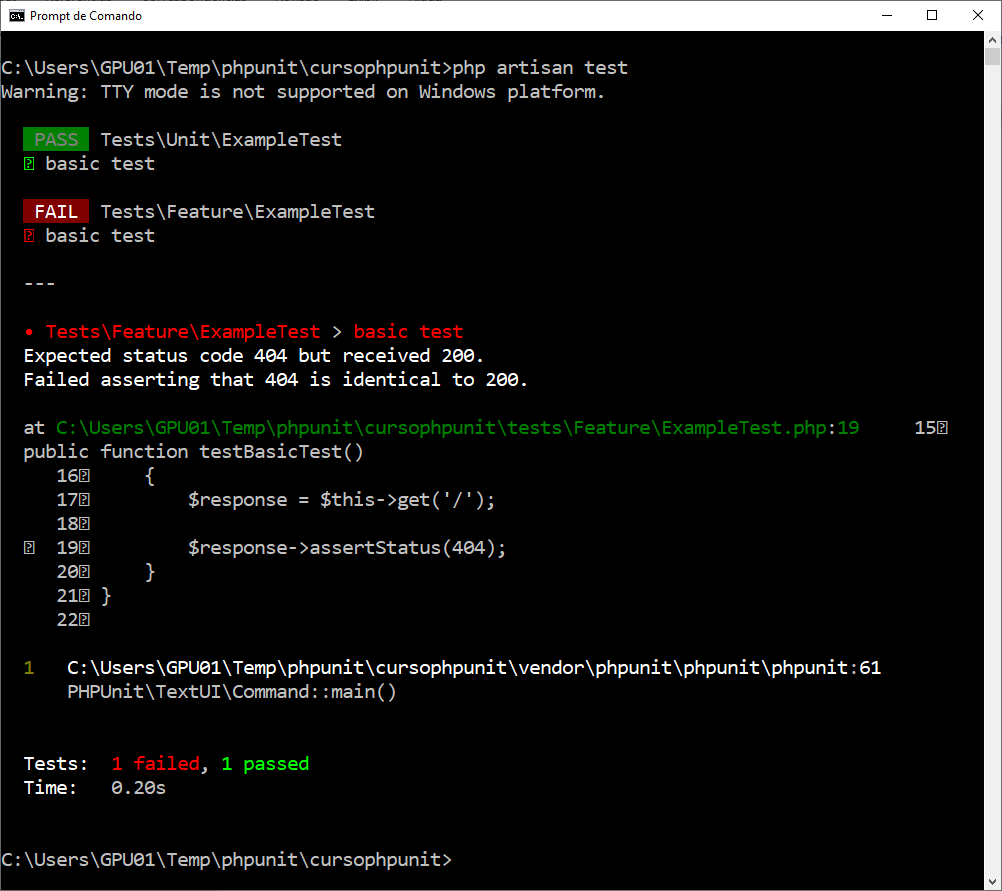
        $response = $this->get('/');

        $response->assertStatus(404);

    }

}

1. Execute os testes novamente e veja o resultado. Agora um teste falhou e o outro passou.



1. Criando um teste de unidade:

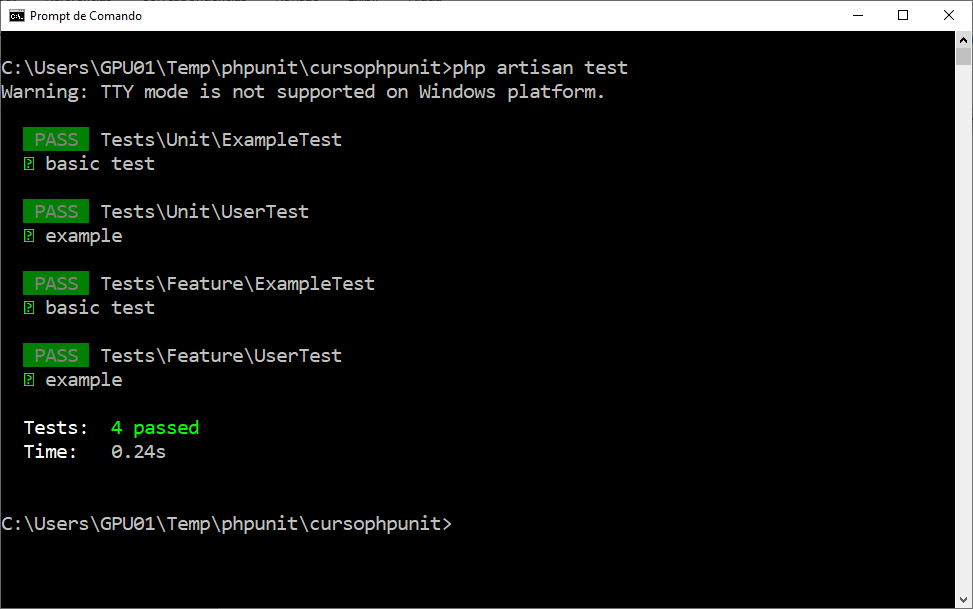
**php artisan make:test UserTest --unit**

1. Criando um teste de feature:

**php artisan make:test UserTest**

1. Temos agora 4 testes. Execute os testes e veja os resultados:

**php artisan test**



Aula 03 – Introdução a testes de unidade, testes de integração e testes de navegação.

1. Vamos configurar o projeto com as demais dependências:

* Crie a base de dados no MySQL blogdb. Configure a base de dados no .env
* Execute o migration para criar as tabelas.
* Adicione o front-end com bootstrap e a opção de autenticação:
  1. **composer require laravel/ui**
  2. **php artisan ui bootstrap --auth**
  3. **npm install && npm run dev**
* Adicione o forms:
  1. **composer require laravelcollective/html**
* Adicione a tradução em português do laravel
* Instale o dusk para realizar teste de browser
  1. **composer require --dev laravel/dusk**
  2. **php artisan dusk:install**
* Configure o .env para que o dusk funcione corretamente.

**APP\_URL=http://localhost:8000**

1. Habilitando o banco de dados de teste no phpunit.xml

<server name="DB\_CONNECTION" value="sqlite"/>

<server name="DB\_DATABASE" value=":memory:"/>

1. Para executar pela linha de comando:
2. Limite de 100 caracteres para cada método de teste

Nome dos testes:

ActionVerb+WhoOrWhatToDO+ExpectedBehavior

1. Testando as rotas globais
2. Testando o redirect para o home usuários não logados.
3. Testando navegação
   1. **php artisan dusk:make RegisterUserTest**

Aula 04 – Criando um comando no Artisan

1. Na linha de comando execute:

**php artisan make:command RunTestsCommand**

1. No projeto em “**app/Console/Commands**” foi criado o arquivo **RunTestsCommand**
2. Abra o arquivo **RunTestsCommand** e defina o nome do comando:

  protected $signature = 'tests';

1. Defina a descrição do comando:

 protected $description = 'Execute all tests';

1. Defina o que será executado:

public function handle()

    {

        Artisan::call('test');

        $this->info(Artisan::output());

        Artisan::call('dusk');

        $this->info(Artisan::output());

        return 0;

    }

1. Veja agora todos os comandos:

**php artisan**

1. Execute o comando criado:

**php artisan tests**