Docker

Quando está desenvolvendo um Sistema ou Site vai ter que colocar no ar, e dois processos irão ocorrer:

1) Configuração de um ambiente de desenvolvimento no seu computador para conseguir fazer o sistema rodar;
2) Criar um ambiente no servidor para rodar, muitas vezes os sistemas não são iguais, você desenvolve no Windows e manda para o servidor Linux, e isso pode trazer problemas devido os ambientes serem diferentes, ou seja, o sistema que funcionava em um local não funciona mais em outro.

Então veio os sistemas de padronizações como o Docker, que criar container (por isso o desenho da baleia com vários containers), cria um ambiente isolado que têm as ferramentas e sistemas necessários para o funcionamento do sistema. É algo totalmente isolado do sistema operacional que está rodando.

Instalando o Docker no seu sistema operacional e tudo que utilizar no sistema/site vai está no Docker, ou seja, dentro de um container que utiliza o Docker. Não importa se o sistema é Windows, Linux, ou qualquer outro que seja, com Docker tem a possibilidade de criar um ambiente exatamente igual no computador e no servidor. Ele cria uma imagem (replica) de tudo que você precisa.

Diferentemente da virtualização que traz todo o sistema operacional e os aplicativos para sua máquina ocupando memória do seu computador, a imagem traz apenas o que é necessário para o funcionamento do aplicativo. Por exemplo, se você gerar uma imagem do MySQL, o Docker irá pegar apenas os recursos necessários do Linux para rodar o MySQL, sem precisar ter todo o Sistema Operacional Linux na sua imagem.

Com o Docker é possível configurar, por exemplo, um servidor nginx/apache usando php 7.4, com o MySQL 7, não sendo necessário a instalação, basta configurar o container.

A primeira coisa que precisa no PC ou servidor é o Docker Machine, isto é, a máquina que vai estar rodando, e acima dela tem os containers criados. O container pode ser desligado e atualizado quando necessário.

A base do Docker Machine são os containers, em paralelo, as imagens e o Docker compose que é a ferramenta para controlar, ativar, desativar, fazer builds nos containers. Além disso, tem o Docker Hub (mercado de imagem docker), um site pago que tem várias imagens para utilizar. Também pode criar os containers e transformar em imagem, e mandar para o Docker Hub. O outro produto é Docker Desktop só funciona no Windows e também no MAC, não tem no Linux. Ele é um processo visual que facilita o processo.

LINUX

- Criando um container:

Primeiramente entre no site do git é baixe a pasta docker-flex:

https://github.com/flexpeak/docker-flex e copiar o link do botão “Code” => https://github.com/flexpeak/docker-flex.git.

2) Dentro da pasta Projetos que já criamos em outras aulas, abra o terminal, em seguida, clone os containers: git clone https://github.com/flexpeak/docker-flex.git

3) No vscode abrir a pasta docker-flex, copiar e colar o arquivo “.env.exemple”, depois renomear a cópia para “.env”.

O arquivo “.env” configura a variável de ambiente do Docker, algumas vezes é usado para fazer uma configuração dinâmica ou colocar senha ou configurar a permissão de um usuário que não pode comitar. O Docker usa em vários momentos a variável de ambiente. Exemplos: no built ou na hora de rodar o container. (https://imasters.com.br/devsecops/como-utilizar-variaveis-de-ambiente-no-docker-serie-devops)

Terminal na pasta docker-flex: ~Documentos\Projeto\docker-flex

Executar o comando: docker-compose -f docker-compose.yaml config (consegue visualizar as variáveis de ambiente que tem no arquivo (docker-compose.yaml config).

Os valores de **host, database, username, password** estão vindo de variáveis de ambiente, isto permite uma maior flexibilidade, pois você pode ter diferentes valores para produção e teste, estas variáveis são carregadas a partir do arquivo **".env"** que está localizado na raiz do projeto.

Essa etapa só é necessária caso você tenha mais de um usuário usando o Docker, senão deixar o padrão **1000** para não terem problemas com permissões.

```
USER_UID=1000
USER_GID=1000
```

No “.env” iremos configurar o usuário para isso vá ao terminal e localize-o.

Para saber a identificação do usuário: **whoami** => academy03

Para pegar o usuário filtrado: **cat /etc/passwd** => academy03:x:**1003** (aparece lá no final).

O comando cat concatenar arquivos. É comumente usado para exibir o conteúdo de um ou vários arquivos de texto, combinar arquivos anexando o conteúdo de um arquivo ao final de outro arquivo e criar novos.

O arquivo /etc/passwd é um arquivo de texto com uma entrada por linha, representando uma conta de usuário. Geralmente, a primeira linha descreve o usuário root, seguido pelo sistema e pelas contas de usuário normais. Novas entradas são anexadas no final do arquivo.

- Trocar no arquivo “.env”

```
## CONFIGURACAO PERMISSAO USUARIO FLEXDOCK

USER_UID=1002
USER_GID=1002
```

```
## CONFIGURAÇÃO MYSQL
MYSQL_ROOT_PASSWORD=root
MYSQL_USER=ivana
MYSQL_PASSWORD=1234
DATA_LIB=~/.docker
```

5) Levantar o container (serviço)

- Precisa estar na pasta docker-flex

- Terminal: sudo docker-compose up -d

- “docker-compose up”: to build the app with the updated Compose file (docker-compose.yaml), and run it.

- --detach , “-d”: executar contêineres em segundo plano (libera o terminal para rodar outros comandos, caso contrário, irá fica rodando e precisará abrir outro terminal).

- OBS.: caso dê erro de IP inválido no comando acima é porque o ubuntu já instala o apache e quer usar a porta 80, logo, precisa parar o apache: sudo service apache2 stop.

Em sala de aula, como temos mais de um usuário, **sempre** iniciaremos com os seguintes passos (não precisa realizar na sua máquina pessoal):

sudo service apache2 stop

sudo docker-compose build

sudo docker-compose down

sudo docker-compose up -d

- Abrir o browser com: localhost, se aparecer a mensagem 403 Forbidden (nginx/1.21.0), está tudo certo.

Isso corre devido parte1.conf está com a indexação de diretório desabilitada.

~Documentos\Projetos\docker-flex\nginx\sites\parte1.conf

```
location / {
    try_files $uri $uri/ /index.php$is_args$args;
}
```

try_files $uri $uri/significa que, a partir do diretório raiz, tente o arquivo apontado por uri, se ele não existir, tente um diretório (daí o /). Quando o nginx acessa um diretório, ele tenta indexá-lo e retornar a lista de arquivos dentro dele para o navegador/cliente, no entanto, por padrão, a indexação de diretório está desabilitada e, portanto, retorna o erro "Erro Nginx 403: índice de diretório de [pasta] é proibido ". Mais a frente resolveremos esse problema com o chmod.

Instalando o Laravel

O gerenciador de dependências utilizado pelo Laravel é o Composer, portanto, para instalar o framework é necessário que o Composer já esteja instalado.

No terminal: precisa estar dentro da pasta docker-flex

Entrar no container do php

sudo docker-compose exec --user=flexdock php7 bash

-u, --user USER, executa o comando como usuário flexdock.

Quando está usando o Git Bash: winpty docker-compose exec --user=flexdock php7 bash

- Irá aparecer: /var/www$, isso quer dizer que está em um container php (não existe uma máquina física, porém, no arquivo docker-compose.yaml mapeia os arquivos da máquina física com o container do Docker-flex =>../:/var/www/).

- O comando **ls -l** mostra o mapeamento da pasta projeto.

No /var/www, vamos criar o projeto parte1

composer create-project laravel/laravel parte1

Fechar o terminal: **exit**

Na pasta docker-flex, abrir o vscode: **code .**

Uma vez instalado você verá uma lista de pastas.

- **app/**
  - Diretório dos arquivos da sua aplicação.
- **bootstrap/**
  - Arquivos de inicialização, são chamados em cada requisição ao servidor.
- **config/**
  - Arquivos de configuração como banco de dados, serviços, etc.
- **database/**
  - Pasta **migrations** que guardam os arquivos que fazem as alterações na estrutura do banco, como nome das tabelas, colunas, etc;
  - Pasta **"seeds"** que seriam arquivos e classes que geram registros no banco de dados, seja para testes ou para possuir informações iniciais de utilização.
- **public/**
  - São os arquivos públicos do sistema, normalmente é aqui que são colocados os arquivos de imagens, CSS e JS.
- **resources/**
  - Arquivos de recursos como bibliotecas JS e CSS, arquivos com arrays de traduções de mensagens, normalmente usados numa aplicação multilinguagem e as próprias views (parte visual do seu site, como HTML, CSS e JavaScript) do sistema.

- **storage/**
  - Arquivos de cache, sessões (quando usado armazenamento em arquivo), views compiladas e logs.
- **tests/**
  - Arquivos de testes do sistema.
- **vendor/**
  - Pasta criada pelo composer para controle e versionamento de bibliotecas.

Após breve explicação, vamos terminar a configuração do projeto parte1.

Entrar na pasta docker-flex para configuração do .config: docker-flex/nginx/sites, copiar e colar o arquivo "laravel.conf", e renomear a cópia para parte1.conf

Com o parte1.conf trocar:

- server name laravel.teste => server name **parte1.ivana** (**url**)

- root/var/www/laravel/public => root/var/www/**parte1**/public (**nome da pasta**)

Não esqueça de salvar!

Retornando ao terminal, todas as vezes que alterar o .conf tem que restart.

- Entra no terminal novamente: ~/Documentos/Projetos/docker-flex
- Reinicia o docker: **sudo docker-compose restart**

- Editando o arquivo host: **sudo nano /etc/hosts**, com a seta até o final do arquivo colocar: **127.0.0.1 parte1.ivana**, em seguida, Ctrl +O (salvar) e Ctrl + X (sair).

Como funciona o arquivo hosts?

O arquivo hosts é a primeira etapa para a transposição do nome de uma máquina DNS em endereço IP. Quando realizamos uma consulta nos servidores DNS de endereço na web, este é o arquivo interrogado.

Qual é a utilidade do arquivo hosts?

Para cada domínio configurado neste arquivo, uma conexão direta com o endereço IP indicado é aberta. Ele funciona como uma espécie de agenda de telefones. Se o endereço for 127.0.0.1, por exemplo, a conexão será feita no localhost - ou seja, na própria máquina.

- Entrar no browser e acessar: http://parte1.ivana

The stream or file "/var/www/parte1/**storage**/logs/laravel.log" could not be opened in append mode: failed to open stream: Permission denied

- ~/Documentos/Projetos/parte1 tem a pasta "storage" que não tem permissão de leitura => drwxrwxrwx

```
total 304
drwxr-xr-x  7 academy03 academy03   4096 set 18 14:40 app
-rw-r--r--  1 academy03 academy03   1686 set 18 14:40 artisan
drwxr-xr-x  3 academy03 academy03   4096 set 18 14:40 bootstrap
-rw-r--r--  1 academy03 academy03   1735 set 18 14:40 composer.json
```

```
-rw-r--r--  1 academy03 academy03 238856 set 18 14:47 composer.lock
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 config
drwxr-xr-x  5 academy03 academy03   4096 set 18 14:40 database
-rw-r--r--  1 academy03 academy03    473 set 18 14:40 package.json
-rw-r--r--  1 academy03 academy03   1202 set 18 14:40 phpunit.xml
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 public
-rw-r--r--  1 academy03 academy03   3964 set 18 14:40 README.md
drwxr-xr-x  6 academy03 academy03   4096 set 18 14:40 resources
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 routes
-rw-r--r--  1 academy03 academy03    563 set 18 14:40 server.php
drwxr-xr-x  5 academy03 academy03   4096 set 18 14:40 storage
drwxr-xr-x  4 academy03 academy03   4096 set 18 14:40 tests
drwxr-xr-x 44 academy03 academy03   4096 set 18 14:47 vendor
-rw-r--r--  1 academy03 academy03    559 set 18 14:40 webpack.mix.js
```

Vamos dar permissão

sudo chmod 777 -R storage

-R: altera, recursivamente, as permissões de arquivos.

```
total 304
drwxr-xr-x  7 academy03 academy03   4096 set 18 14:40 app
-rw-r--r--  1 academy03 academy03   1686 set 18 14:40 artisan
drwxr-xr-x  3 academy03 academy03   4096 set 18 14:40 bootstrap
-rw-r--r--  1 academy03 academy03   1735 set 18 14:40 composer.json
-rw-r--r--  1 academy03 academy03 238856 set 18 14:47 composer.lock
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 config
drwxr-xr-x  5 academy03 academy03   4096 set 18 14:40 database
-rw-r--r--  1 academy03 academy03    473 set 18 14:40 package.json
-rw-r--r--  1 academy03 academy03   1202 set 18 14:40 phpunit.xml
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 public
-rw-r--r--  1 academy03 academy03   3964 set 18 14:40 README.md
drwxr-xr-x  6 academy03 academy03   4096 set 18 14:40 resources
drwxr-xr-x  2 academy03 academy03   4096 set 18 14:40 routes
-rw-r--r--  1 academy03 academy03    563 set 18 14:40 server.php
drwxrwxrwx  5 academy03 academy03   4096 set 18 14:40 storage
drwxr-xr-x  4 academy03 academy03   4096 set 18 14:40 tests
drwxr-xr-x 44 academy03 academy03   4096 set 18 14:47 vendor
-rw-r--r--  1 academy03 academy03    559 set 18 14:40 webpack.mix.js
```

Veja que foi destacado a pasta que foi concedida as permissões de leitura, escrita e execução.

Vamos entender o chmod, suponhamos que queiramos configurar as permissões da pasta /home/temp, com o seguinte comando:

sudo chmod 710 /home/temp

Explicando:

rwx	r--	---
u	g	o
usuário	grupo	outros

ugoa

u = o dono do arquivo (**user**);

g = os usuários que são membros do mesmo grupo do arquivo (**group**);

o = os usuários que não membros do grupo do arquivo (**others**);

a = todos os usuários do sistema (**all**).

Permissões de acesso:

r (**r**ead) = 1 (leitura)

w (**w**rite) = 2 (gravação)

x (e**x**ecute) = 4 (execução (para arquivos) ou autorização de acesso (para diretórios))

u= usuário // para o usuário vou dar permissão total r=1 w=2 x=4 = total 1+2+4 = 7

g= grupo // para grupo permissão leitura = 1

o= outros // outros nenhuma permissão = 0

- Entrar no browser e acessar: http://parte1.ivana (agora aparece a página do Laravel)

Voltando ao nosso projeto parte1

- Entrar na pasta parte1

- Abrir o vscode, em seguida acessar as pastas parte1/routes/web.php e executar os comandos (um por vez) abaixo no browser.

```
Route::get('/', function () {
    echo 'Olá Mundo';
});
//no browser http:/parte1.ivana

Route::get('/meunome', function () {
    echo 'Olá! Ivana';
});
//no browser http:/parte1.ivana/meunome

Route::get('/minha-idade', function () {
    $anoNascimento = 2015;
    $idade = date('Y') - $anoNascimento;
    echo $idade;
});
//no browser http://parte1.ivana/minha-idade

Route::get('/minha-idade_/{anoNascimento}', function ($anoNascimento) {
    $idade = date('Y') - $anoNascimento;
    echo $idade;
});
//no browser http://parte1.ivana/minha-idade_/1952

Route::get('/meu-nome-idade/{nome}/{anoNascimento}', function ($nome,
$anoNascimento) {
    $idade = date('Y') - $anoNascimento;
    echo $nome;
    echo "</br>";
```

```
    echo $idade;
});
//no browser http://parte1.ivana/meu-nome-idade/Maria/1952
```

- No vscode + diretório parte1/resources/views/ tem um único arquivo chamado “welcome.blade.php”
- Para essa etapa tem que criar o “index.blade.php” no diretório parte1/resources/views

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  Olá!
</body>
</html>

// parte1 + routes + web.php
Route::get('/html',function() {
  return view('index');
});
```

WINDOWS

- Para instalar o Docker: https://www.docker.com/

OBS1.: instalar o WSL para computadores com Windows 10 Home (o Windows não é o indicado para trabalhar com o Docker, nem tão pouco o Home, mas ele permite a instalação.)

OBS2.: caso tenha o Laragon instalado é preciso desabilitar para não carregar quando o Windows inicializar (botão direito no botão "janela" + App e Recursos + Inicialização + localizar o Laragon + Desativar). Laragon usa a porta 3306 do mysql.

OBS3.: caso tenha o SQL Server instalado também precisa desativar devido a porta 3306 do mysql (iniciar + Serviços + localizar o mysql + botão direito no mysql + propriedades + drop down "Tipo de inicialização" + Desativado + botão "OK".

Em resumo, qualquer aplicativo que possa usar alguma porta que precisará no Laravel, precisará ser desativado.

Caso fique alguma dúvida de alguns conceitos, leia-os na parte do Linux.

- Criando um container:

Primeiramente entre no site do git é baixe a pasta docker-flex:

1) https://github.com/flexpeak/docker-flex e copiar o link do botão "Code" => https://github.com/flexpeak/docker-flex.git.
2) Dentro da pasta Projetos que já criamos em outras aulas, abra o terminal, em seguida, clone os containers: git clone https://github.com/flexpeak/docker-flex.git
3) No vscode abrir a pasta docker-flex, copiar e colar o arquivo ".env.exemple", depois renomear a cópia para ".env".
   O arquivo ".env" configura a variável de ambiente do Docker, algumas vezes é usado para fazer uma configuração dinâmica ou colocar senha ou configurar a permissão de um usuário que não pode comitar. O Docker usa em vários momentos a variável de ambiente. Exemplos: no built ou na hora de rodar o container. (https://imasters.com.br/devsecops/como-utilizar-variaveis-de-ambiente-no-docker-serie-devops)

   Terminal entrar na pasta docker-flex: C:\Projetos\docker-flex
   Executar os seguintes comandos: docker-compose -f docker-compose.yaml config (consegue visualizar as variáveis de ambiente que tem no arquivo (docker-compose.yaml config).

   Os valores de **host, database, username, password** estão vindo de variáveis de ambiente, isto permite uma maior flexibilidade, pois você pode ter diferentes valores para produção e teste, estas variáveis são carregadas a partir do arquivo **".env"** que está localizado na raiz do projeto.

No ".env" iremos configurar o usuário para isso vá ao terminal e localize-o.

Diferentemente do Linux que usar o comando “sudo”, o Windows não há a preocupação de verificar qual usuário está usando o Docker. Logo, a configuração padrão é:

```
## CONFIGURACAO PERMISSAO USUARIO FLEXDOCK

USER_UID=1000
USER_GID=1000

## CONFIGURAÇÃO MYSQL
MYSQL_ROOT_PASSWORD=root
MYSQL_USER=ivana
MYSQL_PASSWORD=1234
DATA_LIB=~/.docker

#CONFIGURACAO IP_LOCAL
IP_LOCAL=192.168.0.105
```

5) Levantar o container (serviço)

- Precisa estar com o Docker aberto

- Terminal: docker-flex up -d

- “docker-compose up”: to build the app with the updated Compose file (docker-compose.yaml), and run it.

- --detach , “-d”: executar contêineres em segundo plano (libera o terminal para rodar outros comandos, caso contrário, irá fica rodando e precisará abrir outro terminal)

- Abrir o browser com: localhost, se aparecer a mensagem 403 Forbidden (ngnx/1.21.0), está tudo certo.

Isso corre devido parte1.conf está com a indexação de diretório desabilitada.

C:/Documentos/Projetos/docker-flex/nginx/sites/parte1.conf

```
location / {
    try_files $uri $uri/ /index.php$is_args$args;
}
```

try_files $uri $uri/significa que, a partir do diretório raiz, tente o arquivo apontado por uri, se ele não existir, tente um diretório (daí o /). Quando o nginx acessa um diretório, ele tenta indexá-lo e retornar a lista de arquivos dentro dele para o navegador/cliente, no entanto, por padrão, a indexação de diretório está desabilitada e, portanto, retorna o erro "Erro Nginx 403: índice de diretório de [pasta] é proibido ". Mais a frente resolveremos esse problema com o chmod.

- docker-compose exec --user=flexdock php7 bash

-u, --user USER executa o comando com o usuário flexdock.

Instalando o Laravel

O gerenciador de dependências utilizado pelo Laravel é o Composer, portanto, para instalar o framework é necessário que o Composer já esteja instalado.

No terminal: precisa estar dentro da pasta docker-flex

Entrar no container do php

sudo docker-compose exec --user=flexdock php7 bash

-u, --user USER, executa o comando como usuário flexdock.

Quando está usando o Git Bash: winpty docker-compose exec --user=flexdock php7 bash

- Irá aparecer: /var/www$, isso quer dizer que está em um container php (não existe uma máquina física, porém, no arquivo docker-compose.yaml mapeia os arquivos da máquina física com o container do Docker-flex =>../:/var/www/).

- O comando **ls -l** mostra o mapeamento da pasta projeto.

No /var/www, vamos criar o projeto parte1

composer create-project laravel/laravel parte1

Fechar o terminal: **exit**

Na pasta docker-flex, abrir o vscode: **code .**

Uma vez instalado você verá uma lista de pastas.

- **app/**
  - Diretório dos arquivos da sua aplicação.
- **bootstrap/**
  - Arquivos de inicialização, são chamados em cada requisição ao servidor.
- **config/**
  - Arquivos de configuração como banco de dados, serviços, etc.
- **database/**
  - Pasta **migrations** que guardam os arquivos que fazem as alterações na estrutura do banco, como nome das tabelas, colunas, etc;
  - Pasta **"seeds"** que seriam arquivos e classes que geram registros no banco de dados, seja para testes ou para possuir informações iniciais de utilização.
- **public/**
  - São os arquivos públicos do sistema, normalmente é aqui que são colocados os arquivos de imagens, CSS e JS.
- **resources/**
  - Arquivos de recursos como bibliotecas JS e CSS, arquivos com arrays de traduções de mensagens, normalmente usados numa aplicação multilinguagem e as próprias views (parte visual do seu site, como HTML, CSS e JavaScript) do sistema.

- **storage/**
    - Arquivos de cache, sessões (quando usado armazenamento em arquivo), views compiladas e logs.
- **tests/**
    - Arquivos de testes do sistema.
- **vendor/**
    - Pasta criada pelo composer para controle e versionamento de bibliotecas.

Após breve explicação, vamos terminar a configuração do projeto parte1.

Entrar na pasta docker-flex para configuração do .config: docker-flex/nginx/sites, copiar e colar o arquivo "laravel.conf", e renomear a cópia para parte1.conf

Com o parte1.conf trocar:

- server name laravel.teste => server name **parte1.ivana** (**url**)

- root/var/www/laravel/public => root/var/www/**parte1**/public (**nome da pasta**)

Não esqueça de salvar!

Retornando ao terminal, todas as vezes que alterar o .conf tem que restart

Retornando ao terminal

- Entra no terminal novamente: C:\Projetos\docker-flex (se estiver na pasta, permaneça nela)

- Reinicia o docker: docker-compose restart

- Editando o arquivo host: https://king.host/wiki/artigo/como-editar-o-arquivo-hosts-no-windows/

- Abra o explore no caminho: C:\Windows\System32\drivers\etc

- Localize o arquivo hosts e com o botão direito do mouse abra-o com o Notepad++. No final do arquivo acrescentar: 127.0.0.1 parte1.ivana + Salvar (dependendo da instalação no Windows o Notepad++ pedirá permissão para acessar como administrador, confirme e salve novamente).

Como funciona o arquivo hosts?

O arquivo hosts é a primeira etapa para a transposição do nome de uma máquina DNS em endereço IP. Quando realizamos uma consulta nos servidores DNS de endereço na web, este é o arquivo interrogado.

Qual é a utilidade do arquivo hosts?

Para cada domínio configurado neste arquivo, uma conexão direta com o endereço IP indicado é aberta. Ele funciona como uma espécie de agenda de telefones. Se o endereço for 127.0.0.1, por exemplo, a conexão será feita no localhost - ou seja, na própria máquina.

- Entrar no browser e acessar: http://parte1.ivana

The stream or file "/var/www/parte1/**storage**/logs/laravel.log" could not be opened in append mode: failed to open stream: Permission denied

- c:/Projetos/docker-flex/parte1 tem a pasta “storage” que não tem permissão de leitura => drwxrwxrwx

```
flexdock@e94bfd12d548:/var/www/parte1$ ls -l
total 257
-rw-r--r-- 1 flexdock flexdock   3964 Sep  8 17:33 README.md
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 app
-rw-r--r-- 1 flexdock flexdock   1686 Sep  8 17:33 artisan
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 bootstrap
-rw-r--r-- 1 flexdock flexdock   1735 Sep  8 17:33 composer.json
-rw-r--r-- 1 flexdock flexdock 238407 Sep  8 17:41 composer.lock
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 config
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 database
-rw-r--r-- 1 flexdock flexdock    473 Sep  8 17:33 package.json
-rw-r--r-- 1 flexdock flexdock   1202 Sep  8 17:33 phpunit.xml
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 public
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 resources
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 routes
-rw-r--r-- 1 flexdock flexdock    563 Sep  8 17:33 server.php
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 storage
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 tests
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:42 vendor
-rw-r--r-- 1 flexdock flexdock    559 Sep  8 17:33 webpack.mix.js
```

Vamos dar permissão é do Linux, logo precisamos estar no container do php para concedê-las.e

Entrar na pasta docker-flex

```
Entrar no container do php como root
sudo docker-compose exec php7 bash

Entrar na pasta parte1
root@f92eb7c9c32f:/var/www# cd parte1

Executar o chmod
root@f92eb7c9c32f:/var/www/parte1# chmod 777 -R storage
root@f92eb7c9c32f:/var/www/parte1# chmod 777 -R bootstrap/cache

-rw-r--r-- 1 flexdock flexdock   3964 Sep  8 17:33 README.md
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 app
-rw-r--r-- 1 flexdock flexdock   1686 Sep  8 17:33 artisan
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 bootstrap
-rw-r--r-- 1 flexdock flexdock   1735 Sep  8 17:33 composer.json
-rw-r--r-- 1 flexdock flexdock 238407 Sep  8 17:41 composer.lock
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 config
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 database
-rw-r--r-- 1 flexdock flexdock    473 Sep  8 17:33 package.json
-rw-r--r-- 1 flexdock flexdock   1202 Sep  8 17:33 phpunit.xml
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 public
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 resources
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 routes
-rw-r--r-- 1 flexdock flexdock    563 Sep  8 17:33 server.php
drwxrwxrwx 1 flexdock flexdock    512 Sep  8 17:33 storage
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:33 tests
drwxr-xr-x 1 flexdock flexdock    512 Sep  8 17:42 vendor
-rw-r--r-- 1 flexdock flexdock    559 Sep  8 17:33 webpack.mix.js
```

Veja que foi destacado a pasta que foi concedida as permissões de leitura, escrita e execução.

- Abrir o vscode + parte1 + routes + web.php e executar os comandos (um por vez) abaixo no browser.

Na linha 1 acessamos o método estático **get** do objeto Route. Veremos mais à frente o motivo para a utilização de um verbo na definição de uma rota. A Barra: "**/**" é o endereço que acessaremos, referente a página inicial da aplicação.
Continuando na Linha 1, **function ()** é a declaração de uma função anônima (sem nome). Na Linha 3 **return** retornará o texto "Olá Mundo!" quando a rota for acessada.

```
1. Route::get('/', function () {
2.   echo 'Olá Mundo!';
3. });

//no browser http:/parte1.ivana

Route::get('/meunome', function () {
   echo 'Olá! Ivana';
});
//no browser http:/parte1.ivana/meunome

Route::get('/minha-idade', function () {
   $anoNascimento = 2015;
   $idade = date('Y') - $anoNascimento;
   echo $idade;
});
//no browser http://parte1.ivana/minha-idade

Route::get('/minha-idade_/{anoNascimento}', function ($anoNascimento) {
   $idade = date('Y') - $anoNascimento;
   echo $idade;
});
//no browser http://parte1.ivana/minha-idade_/1952

Route::get('/meu-nome-idade/{nome}/{anoNascimento}', function ($nome,
$anoNascimento) {
   $idade = date('Y') - $anoNascimento;
   echo $nome;
   echo "</br>";
   echo $idade;
});
//no browser http://parte1.ivana/meu-nome-idade/Maria/1952
```

- No vscode + diretório parte1/resources/views/ tem um único arquivo chamado “welcome.blade.php”

- Para essa etapa tem que criar o “index.blade.php” no diretório parte1/resources/views

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   Olá!
</body>
</html>

// parte1 + routes + web.php
Route::get('/html',function() {
   return view('index');
});
```