

DADOS DA ESTÓRIA

Data	Sprint	Versão	Estória	Autor
08/11/2020	Não se aplica	v.1	Docker + Laravel + PhpStorm ou VSCode	Luan Cardoso

SUMÁRIO

Dados da Estória	0
Sumário	0
Estória	1
1. Requisitos.....	1
2. Tutorial em Vídeo	1
3. Repositório.....	1
4. Laradock.....	1
5. Instalando e Rodando o Laradock	1
5.1.1. Clone o repositório do laradock dentro da raiz do teu projeto:.....	1
5.1.2. Entre na pasta do repositório baixado e copie o env-example e renomeie esta cópia para .env:	1
5.1.3. Rodando o docker:.....	2
6. Acessando o bash dos containers para executar comandos	4
6.1.1. Acessando o bash para rodar comandos como Artisan, Composer, PHPUnit, Gulp, etc:	4
7. Acessando o banco de Dados	5
8. rodando e acessando a aplicação Laravel.....	7
8.1.1. Criando o projeto:.....	7
9. Migrations.....	10
10. Xdebug	12
10.1.1. Instalando a extensão no chrome:	12
10.1.2. Habilitando no Laradock:	12
10.1.3. PhpStorm:.....	13
10.1.4. VSCode:.....	17
11. Comandos úteis do Docker	20

ESTÓRIA

1. REQUISITOS

Para começar você já ter instalado em sua máquina:

- Docker
- Git
- MySQL Workbench ou outro cliente para MySQL de sua escolha
- PHPStorm com licença ou VSCode
- Não instale nada mais em sua máquina, tudo que for utilizar no projeto DEVE ser instalado no Docker, este é o conceito de **Infrastructure as Code**, assim quando qualquer outra pessoa baixar o repo do Docker, com apenas um comando, já terá o ambiente pronto para desenvolvimento.

2. TUTORIAL EM VÍDEO

Este mesmo tutorial também foi desenvolvido em vídeo, segue link abaixo:

<https://youtu.be/OWWjIS7mEHk>

3. REPOSITÓRIO

O repositório deste tutorial também pode ser encontrado no link abaixo:

<https://github.com/luancardosolc/laravel-with-laradock>

4. LARADOCK

Laradock é um ambiente de desenvolvimento PHP completo para Docker, se você precisar de Nginx, já tem, precisou de Apache, já tem, precisou de Redis, já tem, entre outros muitos recursos, perfeito para iniciantes em Docker, por isso faremos uso dele, segue link da doc oficial abaixo:

<https://laradock.io/>

5. INSTALANDO E RODANDO O LARADOCK

5.1.1. Clone o repositório do laradock dentro da raiz do teu projeto:

```
git clone https://github.com/laradock/laradock.git laradock-projetoum
```

Obs: Importante renomear esta pasta com um nome único para que cada laradock seja único para cada aplicação, ao usar o mesmo Docker container para várias aplicações o propósito do Docker é derrotado, que é isolar ambientes. Pode-se seguir a estrutura acima se preferir, laradock-projetoum, ou seja, laradock + nome do projeto.

5.1.2. Entre na pasta do repositório baixado e copie o env-example e renomeie esta cópia para .env:

```
cp env-example .env
```

No arquivo .env, mude a seguinte variável para o nome do teu projeto:

```
11 APP_CODE_PATH_CONTAINER=/var/www
12
13 # You may add flags to the path `:cached`, `:delegated`. When using Docker Sync add `:nocopy`
14 APP_CODE_CONTAINER_FLAG=:cached
15
16 # Choose storage path on your machine. For all storage systems
17 DATA_PATH_HOST=~/.laradock/data
18
19 ### Drivers #####
20
21 # All volumes driver
22 VOLUMES_DRIVER=local
23
24 # All Networks driver
25 NETWORKS_DRIVER=bridge
26
27 ### Docker compose files #####
28
29 # Select which docker-compose files to include. If using docker-sync append `:docker-compose.sync.yml` at the end
30 COMPOSE_FILE=docker-compose.yml
31
32 # Change the separator from : to ; on Windows
33 COMPOSE_PATH_SEPARATOR=:
34
35 # Define the prefix of container names. This is useful if you have multiple projects that use laradock to have separate containers per project.
36 COMPOSE_PROJECT_NAME=projeto001
37
38 ### PHP Version #####
39
40 # Select a PHP version of the Workspace and PHP-FPM containers (Does not apply to HHVM).
41 # Accepted values: 7.4 - 7.3 - 7.2 - 7.1 - 7.0 - 5.6
42 PHP_VERSION=7.3
43
44 ### Phalcon Version #####
45
46 # Select a Phalcon version of the Workspace and PHP-FPM containers (Does not apply to HHVM). Accepted values: 3.4.0+
47 PHALCON_VERSION=4.0.5
48
49 ### PHP Interpreter #####
50
51 # Select the PHP Interpreter. Accepted values: hhvm - php-fpm
52 PHP_INTERPRETER=php-fpm
53
54 ### Docker Host IP #####
55
56 # Enter your Docker Host IP (will be appended to /etc/hosts). Default is `10.0.75.1`
57 DOCKER_HOST_IP=10.0.75.1
58
59 ### Remote Interpreter #####
```

Mude também as configurações para já habilitar o Xdebug, conforme este doc no item [Xdebug](#) alterando as variáveis que habilitam o xdebug para true e então siga para o próximo passo.

5.1.3. Rodando o docker:

O comando abaixo irá subir no container do laradock os serviços Nginx e Mysql:

```
docker-compose up nginx mysql
```

É normal demorar bastante na primeira vez que executar este comando, pq ele irá fazer download de todas imagens do Docker que o Laradock poderá utilizar.

A mensagem abaixo indica que o build do Docker foi feito com sucesso:

```
MINGW64: c:/projetos/jll/laradock-projetoum
---> Running in 8ec44870df22
Removing intermediate container 8ec44870df22
---> cf8304f60f1c
Step 14/17 : ADD ./startup.sh /opt/startup.sh
---> 895d59702fe8
Step 15/17 : RUN sed -i 's/\r/\n/' /opt/startup.sh
---> Running in 24e659729052
Removing intermediate container 24e659729052
---> 1ecc05c42b26
Step 16/17 : CMD ["/bin/bash", "/opt/startup.sh"]
---> Running in 81fcc2a2cf65
Removing intermediate container 81fcc2a2cf65
---> 368d827742b7
Step 17/17 : EXPOSE 80 81 443
---> Running in 53ee5cb096d8
Removing intermediate container 53ee5cb096d8
---> 77f46fdc7cd6

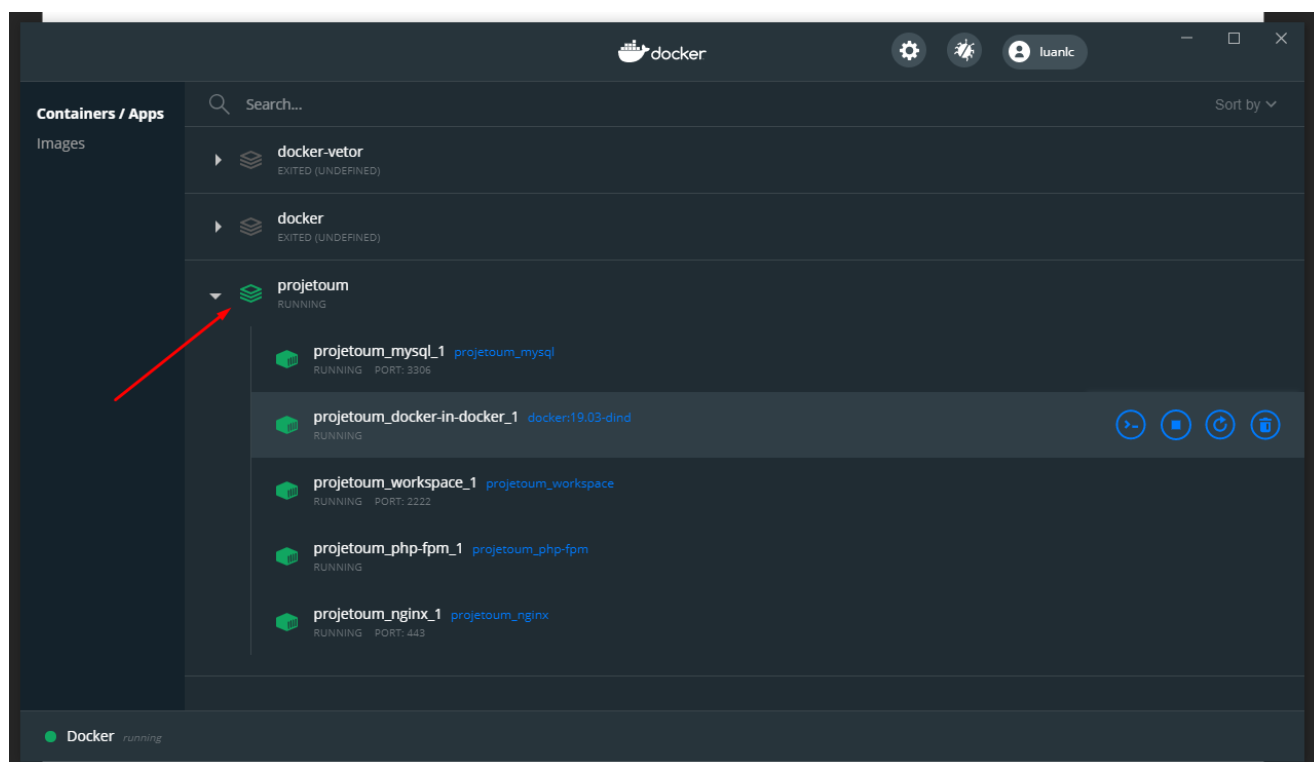
Successfully built 77f46fdc7cd6
Successfully tagged projetoum_nginx:latest
Image for service nginx was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Creating projetoum_mysql_1 ... done
Creating projetoum_docker-in-docker_1 ... done
Creating projetoum_workspace_1 ... done
Creating projetoum_php-fpm_1 ... done
Creating projetoum_nginx_1 ... done
Attaching to projetoum_mysql_1, projetoum_nginx_1
mysql_1 | 2020-11-08 17:29:44+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql_1 | 2020-11-08 17:29:44+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql_1 | 2020-11-08 17:29:44+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql_1 | 2020-11-08T17:29:45.245762Z 0 [Warning] [MY-010915] [Server] 'NO_ZERO_DATE', 'NO_ZERO_IN_DATE' and 'ERROR_FOR_DIVISION_BY_ZERO' sql modes should be used with strict mode. They will be merged with strict mode in a future release.
mysql_1 | 2020-11-08T17:29:45.246471Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) starting as process 1
mysql_1 | 2020-11-08T17:29:45.250839Z 0 [Warning] [MY-013242] [Server] --character-set-server: 'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a future release. Please consider using UTF8MB4 in order to be unambiguous.
mysql_1 | 2020-11-08T17:29:45.259545Z 0 [Warning] [MY-010159] [Server] Setting lower_case_table_names=2 because file system for /var/lib/mysql/ is case insensitive
mysql_1 | 2020-11-08T17:29:45.293273Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql_1 | 2020-11-08T17:29:47.948256Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql_1 | 2020-11-08T17:29:48.411076Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql_1 | 2020-11-08T17:29:48.712512Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql_1 | 2020-11-08T17:29:48.713567Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql_1 | 2020-11-08T17:29:48.759572Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql_1 | 2020-11-08T17:29:48.992809Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.22' socket: '/var/run/mysqld/mysql.sock' port: 3306 MySQL Community Server - GPL.
```

O Comando abaixo irá listar os serviços em execução, como vemos abaixo, Nginx, PHP e Mysql estão sendo executados:

docker ps

```
luanc@DESKTOP-ML48T1E MINGW64 /c:/projetos/jll/laradock-projetoum (master)
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
34b117d039e5   projetoum_nginx                    "/docker-entrypoint...." 2 minutes ago  Up 2 minutes  0.0.0.0:80-81->80-81/tcp, 0.0.0.0:443->443/tcp
78773b041d9c   projetoum_php-fpm                  "docker-php-entrypoi..." 2 minutes ago  Up 2 minutes  9000/tcp
96c658028a44   projetoum_workspace                "/sbin/my_init"          2 minutes ago  Up 2 minutes  0.0.0.0:8001->8000/tcp, 0.0.0.0:8080->8080/tcp, 0.0.0.0:2222->22/tcp, 2375-2376/tcp
497dc7b415c7   docker:19.03-dind                 "dockerd-entrypoint...." 2 minutes ago  Up 2 minutes  2375-2376/tcp
32fae115d115   projetoum_mysql                    "docker-entrypoint.s..." 2 minutes ago  Up 2 minutes  0.0.0.0:3306->3306/tcp, 33060/tcp
projetoum_mysql_1
projetoum_mysql_1
```

Ou você pode ver os containers em execução pelo próprio dashboard do “Docker Desktop”:



Para parar todos os containers em execução, basta clicar no botão stop do 'projeto', ou rodar o comando abaixo:

```
docker stop $(docker ps -q)
```

6. ACESSANDO O BASH DOS CONTAINERS PARA EXECUTAR COMANDOS

6.1.1. Acessando o bash para rodar comandos como Artisan, Composer, PHPUnit, Gulp, etc:

```
docker-compose exec workspace bash
```

```
luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/jll/laradock-projetoum (master)
$ docker-compose exec workspace bash
the input device is not a TTY. If you are using mintty, try prefixing the command with 'winpty'

luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/jll/laradock-projetoum (master)
$ winpty docker-compose exec workspace bash
bash: '$\r': command not found
bash: '$\r': command not found
bash: /root/aliases.sh: line 119: syntax error near unexpected token `'$\r''
'ash: /root/aliases.sh: line 119: `function mkd() {
root@96c658028a44:/var/www# ls -la
total 8
drwxr-xr-x 1 laradock laradock 512 Nov  8 16:58 .
drwxr-xr-x 1 root     root     4096 Nov  8 17:22 ..
drwxrwxrwx 1 root     root     512 Nov  8 15:06 laradock-projetoum
drwxr-xr-x 1 laradock laradock 512 Nov  7 19:40 learningpathBrito
root@96c658028a44:/var/www# |
```

Você também pode acessar o serviço específico diretamente:

```
docker exec -it {workspace-container-id ou container-name} bash
```

Ex:

```
docker exec -it {workspace-container-id} bash
```

```
luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/jll/laradock-projetoum (master)
$ docker exec -it projetoum_php-fpm_1 bash
the input device is not a TTY. If you are using mintty, try prefixing the command with 'winpty'

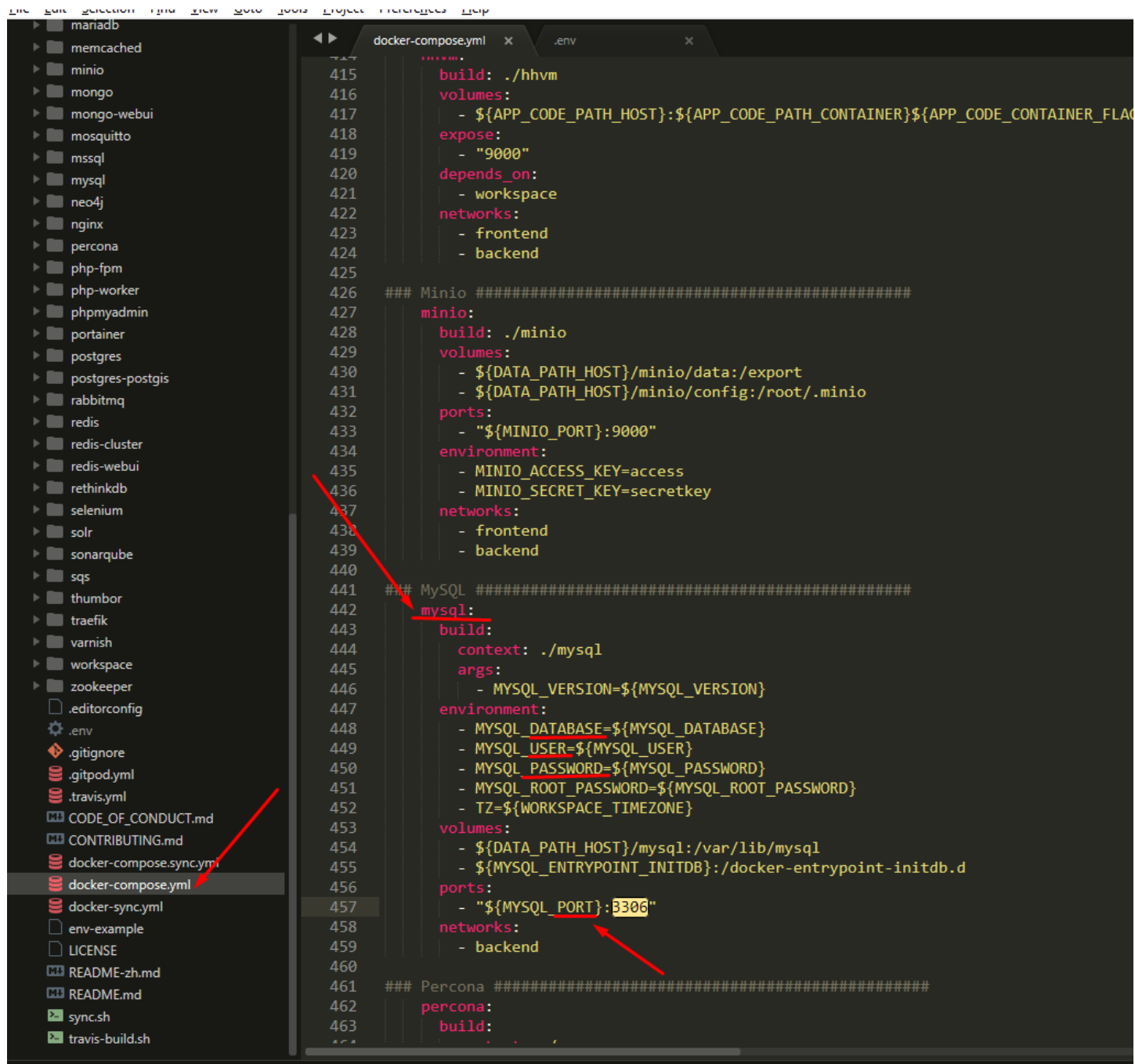
luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/jll/laradock-projetoum (master)
$ winpty docker exec -it projetoum_php-fpm_1 bash
root@78773b041d9c:/var/www# ls -la
total 4
drwxr-xr-x 1 www-data www-data 512 Nov  8 16:58 .
drwxr-xr-x 1 root     root     4096 Oct 13 09:15 ..
drwxrwxrwx 1 root     root     512 Nov  8 15:06 laradock-projetoum
drwxr-xr-x 1 www-data www-data 512 Nov  7 19:40 learningpathBrito
root@78773b041d9c:/var/www#
```

Obs:

- No Windows, as vezes é necessário acrescentar a palavra 'winpty' a frente do comando Docker.
- Comandos 'docker-compose' devem ser rodados na mesma pasta em que se encontra o arquivo **docker-compose.yml**.

7. ACESSANDO O BANCO DE DADOS

Para que possamos ver a *String Connection* necessária para acessar o serviço do MySQL, basta ir até o arquivo **docker-compose.yml** e observar o trecho abaixo:



```
415     build: ./hhvm
416     volumes:
417     - ${APP_CODE_PATH_HOST}:${APP_CODE_PATH_CONTAINER}${APP_CODE_CONTAINER_FLAG}
418     expose:
419     - "9000"
420     depends_on:
421     - workspace
422     networks:
423     - frontend
424     - backend
425
426     ### Minio #####
427     minio:
428     build: ./minio
429     volumes:
430     - ${DATA_PATH_HOST}/minio/data:/export
431     - ${DATA_PATH_HOST}/minio/config:/root/.minio
432     ports:
433     - "${MINIO_PORT}:9000"
434     environment:
435     - MINIO_ACCESS_KEY=access
436     - MINIO_SECRET_KEY=secretkey
437     networks:
438     - frontend
439     - backend
440
441     ### MySQL #####
442     mysql:
443     build:
444     context: ./mysql
445     args:
446     - MYSQL_VERSION=${MYSQL_VERSION}
447     environment:
448     - MYSQL_DATABASE=${MYSQL_DATABASE}
449     - MYSQL_USER=${MYSQL_USER}
450     - MYSQL_PASSWORD=${MYSQL_PASSWORD}
451     - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
452     - TZ=${WORKSPACE_TIMEZONE}
453     volumes:
454     - ${DATA_PATH_HOST}/mysql:/var/lib/mysql
455     - ${MYSQL_ENTRYPOINT_INITDB}:/docker-entrypoint-initdb.d
456     ports:
457     - "${MYSQL_PORT}:3306"
458     networks:
459     - backend
460
461     ### Percona #####
462     percona:
463     build:
```

Nele encontramos:

Host (nome do serviço): mysql

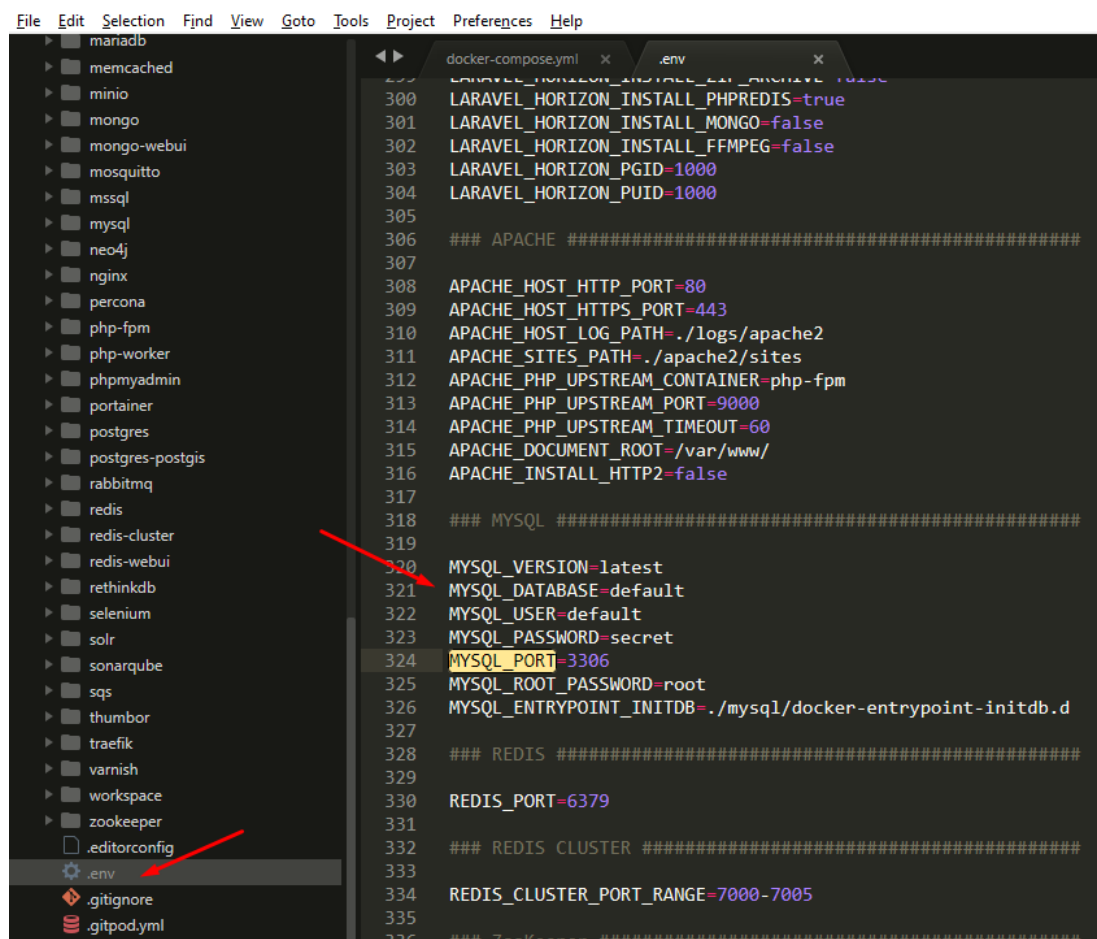
Database: MYSQL_DATABASE

User: MYSQL_USER

Pass: MYSQL_PASSWORD

Porta: MYSQL_PORT

Como você deve ter percebido, são variáveis, e encontramos os valores dela no `.env`:



```
File Edit Selection Find View Goto Tools Project Preferences Help
maradb
memcached
minio
mongo
mongo-webui
mosquitto
mssql
mysql
neo4j
nginx
percona
php-fpm
php-worker
phpmyadmin
portainer
postgres
postgres-postgis
rabbitmq
redis
redis-cluster
redis-webui
rethinkdb
selenium
solr
sonarqube
sq
thumbor
traefik
varnish
workspace
zookeeper
.editorconfig
.env
.gitignore
.gitpod.yml

docker-compose.yml x .env x
LARAVEL_HORIZON_INSTALL_PHPREDIS=true
LARAVEL_HORIZON_INSTALL_MONGO=false
LARAVEL_HORIZON_INSTALL_FFMPEG=false
LARAVEL_HORIZON_PGID=1000
LARAVEL_HORIZON_PUID=1000
### APACHE #####
APACHE_HOST_HTTP_PORT=80
APACHE_HOST_HTTPS_PORT=443
APACHE_HOST_LOG_PATH=./logs/apache2
APACHE_SITES_PATH=./apache2/sites
APACHE_PHP_UPSTREAM_CONTAINER=php-fpm
APACHE_PHP_UPSTREAM_PORT=9000
APACHE_PHP_UPSTREAM_TIMEOUT=60
APACHE_DOCUMENT_ROOT=/var/www/
APACHE_INSTALL_HTTP2=false
### MYSQL #####
MYSQL_VERSION=latest
MYSQL_DATABASE=default
MYSQL_USER=default
MYSQL_PASSWORD=secret
MYSQL_PORT=3306
MYSQL_ROOT_PASSWORD=root
MYSQL_ENTRYPOINT_INITDB=./mysql/docker-entrypoint-initdb.d
### REDIS #####
REDIS_PORT=6379
### REDIS CLUSTER #####
REDIS_CLUSTER_PORT_RANGE=7000-7005
### ZooKeeper #####
```

Agora sim temos nossa String Connection:

Host (nome do serviço ao invés do ip quando `.env` do laravel): mysql / 127.0.0.1

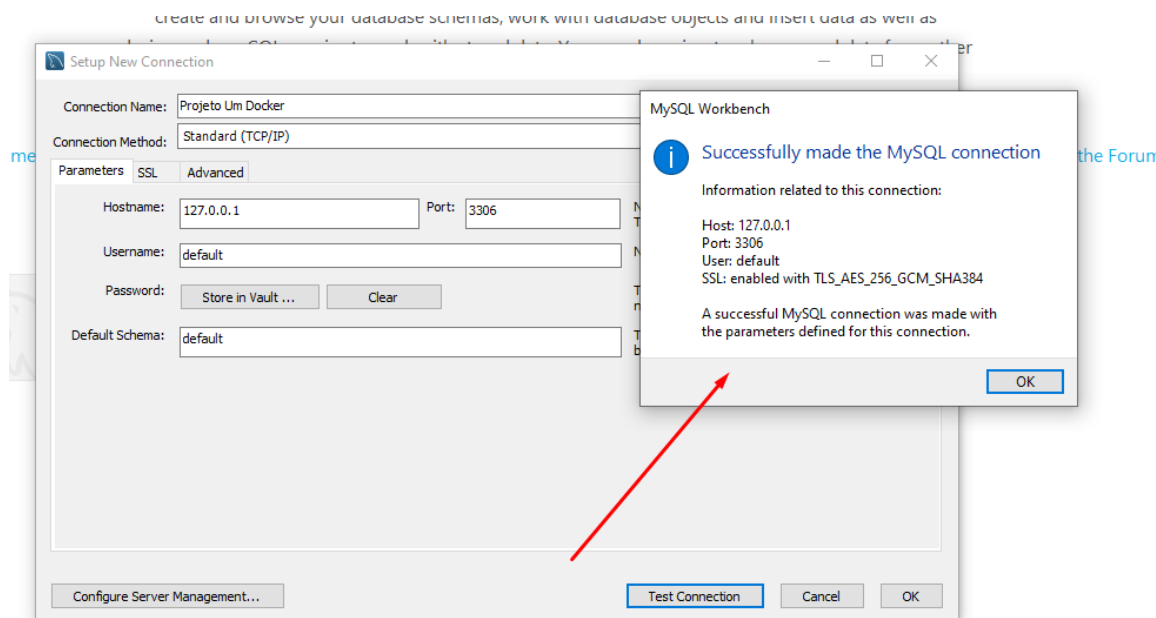
Database: default

User: default

Pass: secret

Porta: 3306

Conexão feita com sucesso no MySQL WorkBench com as credenciais acima:



8. RODANDO E ACESSANDO A APLICAÇÃO LARAVEL

A documentação do Laravel é muito bem feita e repleta de bons exemplos, então caso tenha maiores dúvidas basta acessá-la em:

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

8.1.1. Criando o projeto:

Acesse o workspace do docker que acabamos de criar, digitando o comando abaixo dentro da pasta raiz do laradock:

`docker-compose exec workspace bash`

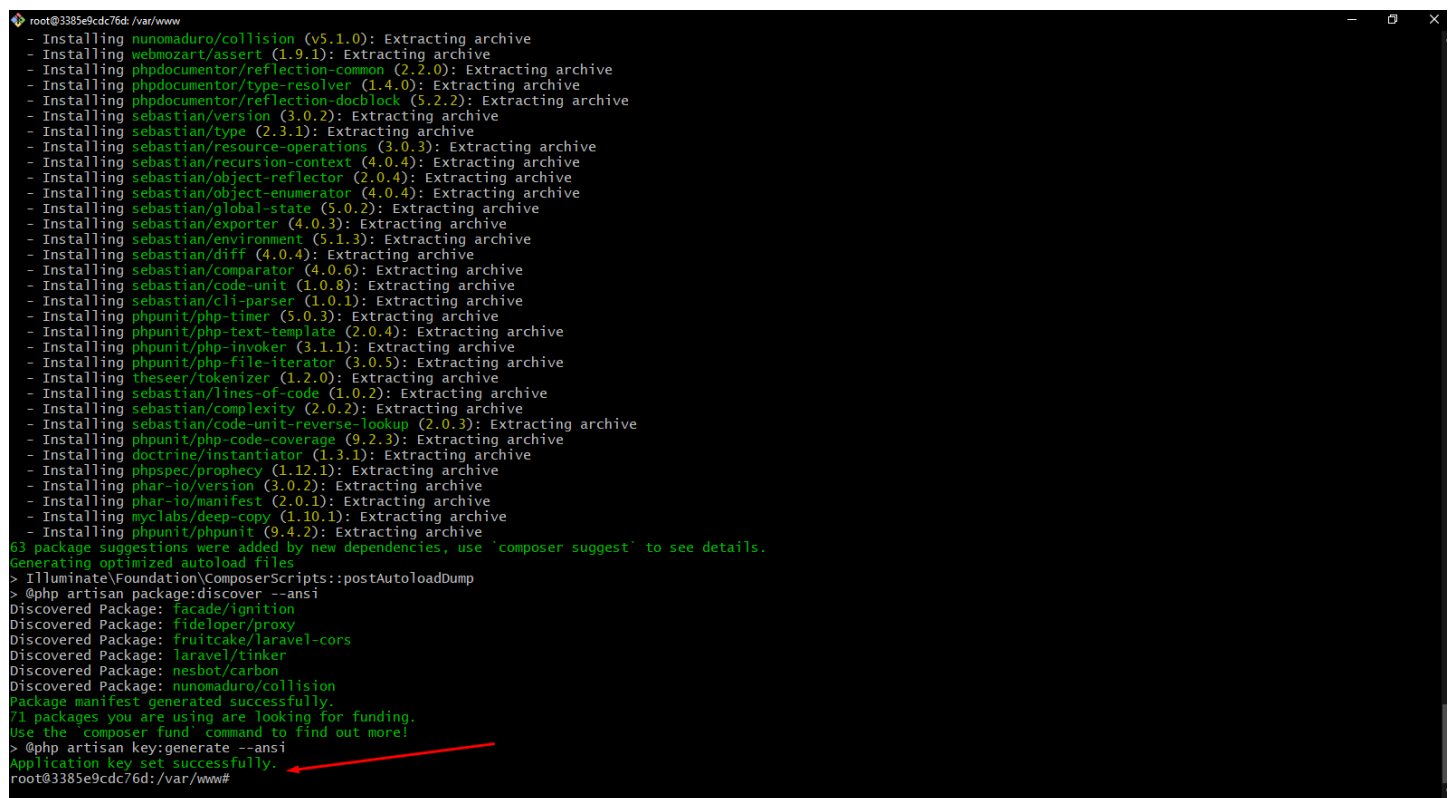
```
luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/j11/projetoum/laradock-projetoum (master)
$ docker-compose up -d nginx mysql
projeto Docker-in-docker_1 is up-to-date
Starting projeto_mysql_1 ...
projeto_workspace_1 is up-to-date
Starting projeto_mysql_1 ... done
Starting projeto_nginx_1 ... done

luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/j11/projetoum/laradock-projetoum (master)
$ docker-compose exec workspace bash
the input device is not a TTY. If you are using mintty, try prefixing the command with 'winpty'

luanc@DESKTOP-ML48IJF MINGW64 /c/projetos/j11/projetoum/laradock-projetoum (master)
$ winpty docker-compose exec workspace bash
bash: $'\r': command not found
bash: $'\r': command not found
bash: /root/.aliases.sh: line 119: syntax error near unexpected token `${\r}'
'ash: /root/.aliases.sh: line 119: `function mkd() {
npm notice
npm notice New patch version of npm available! 7.0.8 -> 7.0.9
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.0.9
npm notice Run npm install -g npm@7.0.9 to update!
root@3385e9cdc76d:/var/www# ls -la
total 8
drwxrwxrwx 1 root root 512 Nov  8 19:02 .
drwxr-xr-x 1 root root 4096 Nov  8 17:22 ..
drwxrwxrwx 1 root root 512 Nov  8 15:06 laradock-projetoum
root@3385e9cdc76d:/var/www#
```


Estando na pasta que deseja criar o projeto Laravel, iremos rodar o comando abaixo para criar o projeto na raiz desta pasta, observe que o laradock também deve estar dentro desta pasta conforme exibido no print acima:

```
composer create-project --prefer-dist laravel/laravel
```



```
root@3385e9cdc76d: /var/www#
- Installing nunomaduro/collision (v5.1.0): Extracting archive
- Installing webmozart/assert (1.9.1): Extracting archive
- Installing phpdocumentor/reflection-common (2.2.0): Extracting archive
- Installing phpdocumentor/type-resolver (1.4.0): Extracting archive
- Installing phpdocumentor/reflection-docblock (5.2.2): Extracting archive
- Installing sebastian/version (3.0.2): Extracting archive
- Installing sebastian/type (2.3.1): Extracting archive
- Installing sebastian/resource-operations (3.0.3): Extracting archive
- Installing sebastian/recursion-context (4.0.4): Extracting archive
- Installing sebastian/object-reflector (2.0.4): Extracting archive
- Installing sebastian/object-enumerator (4.0.4): Extracting archive
- Installing sebastian/global-state (5.0.2): Extracting archive
- Installing sebastian/exporter (4.0.3): Extracting archive
- Installing sebastian/environment (5.1.3): Extracting archive
- Installing sebastian/diff (4.0.4): Extracting archive
- Installing sebastian/comparator (4.0.6): Extracting archive
- Installing sebastian/code-unit (1.0.8): Extracting archive
- Installing sebastian/cli-parser (1.0.1): Extracting archive
- Installing phpunit/php-timer (5.0.3): Extracting archive
- Installing phpunit/php-text-template (2.0.4): Extracting archive
- Installing phpunit/php-invoker (3.1.1): Extracting archive
- Installing phpunit/php-file-iterator (3.0.5): Extracting archive
- Installing theseer/tokenizer (1.2.0): Extracting archive
- Installing sebastian/lines-of-code (1.0.2): Extracting archive
- Installing sebastian/complexity (2.0.2): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (2.0.3): Extracting archive
- Installing phpunit/php-code-coverage (9.2.3): Extracting archive
- Installing doctrine/instantiator (1.3.1): Extracting archive
- Installing phpspec/prophecy (1.12.1): Extracting archive
- Installing phar-io/version (3.0.2): Extracting archive
- Installing phar-io/manifest (2.0.1): Extracting archive
- Installing myclabs/deep-copy (1.10.1): Extracting archive
- Installing phpunit/phpunit (9.4.2): Extracting archive
63 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
71 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan key:generate --ansi
Application key set successfully.
root@3385e9cdc76d: /var/www#
```

Print acima evidencia que a aplicação foi criada com sucesso.

Rode os comandos abaixo para mover os arquivos para a raiz da pasta e deletar a pasta que sobrou vazia:

```
shopt -s dotglob
```

```
mv laravel/* .
```

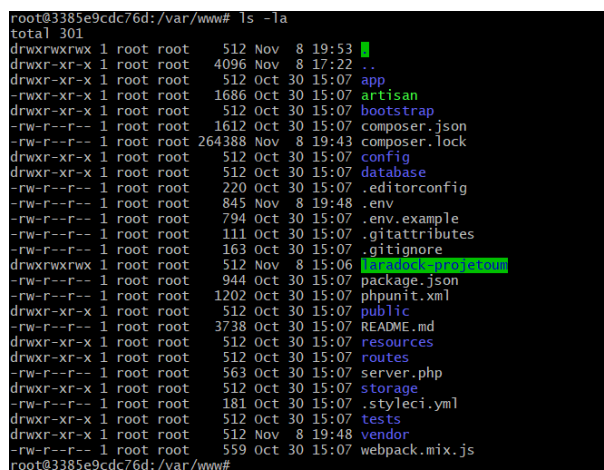
```
rm -R laravel/
```

```
chmod -R 777 storage/
```

```
chmod -R 777 bootstrap/cache/
```

```
chmod -R 777 public
```

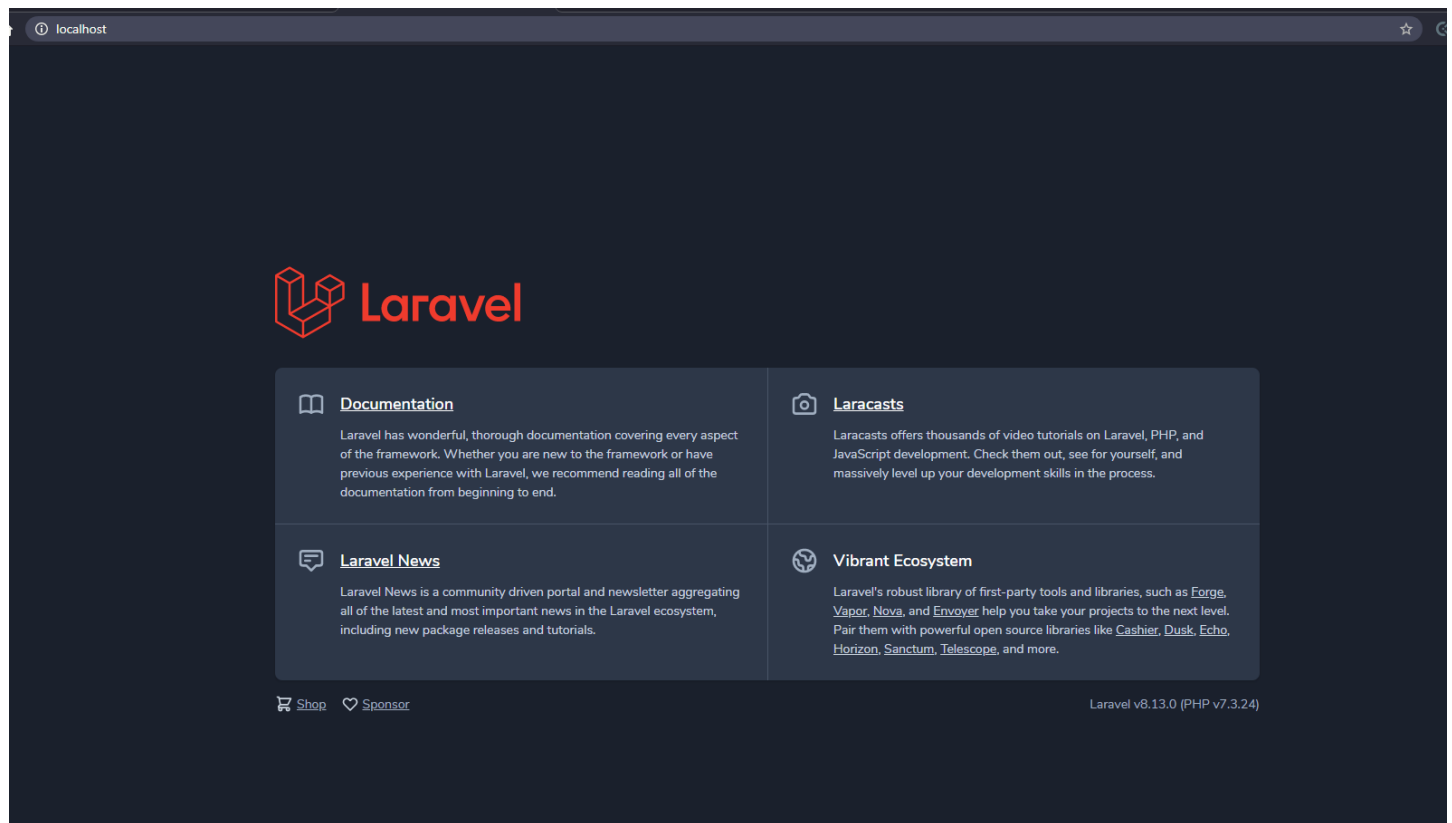
Após rodar estes comandos, a pasta raiz do projeto ficará da seguinte maneira:



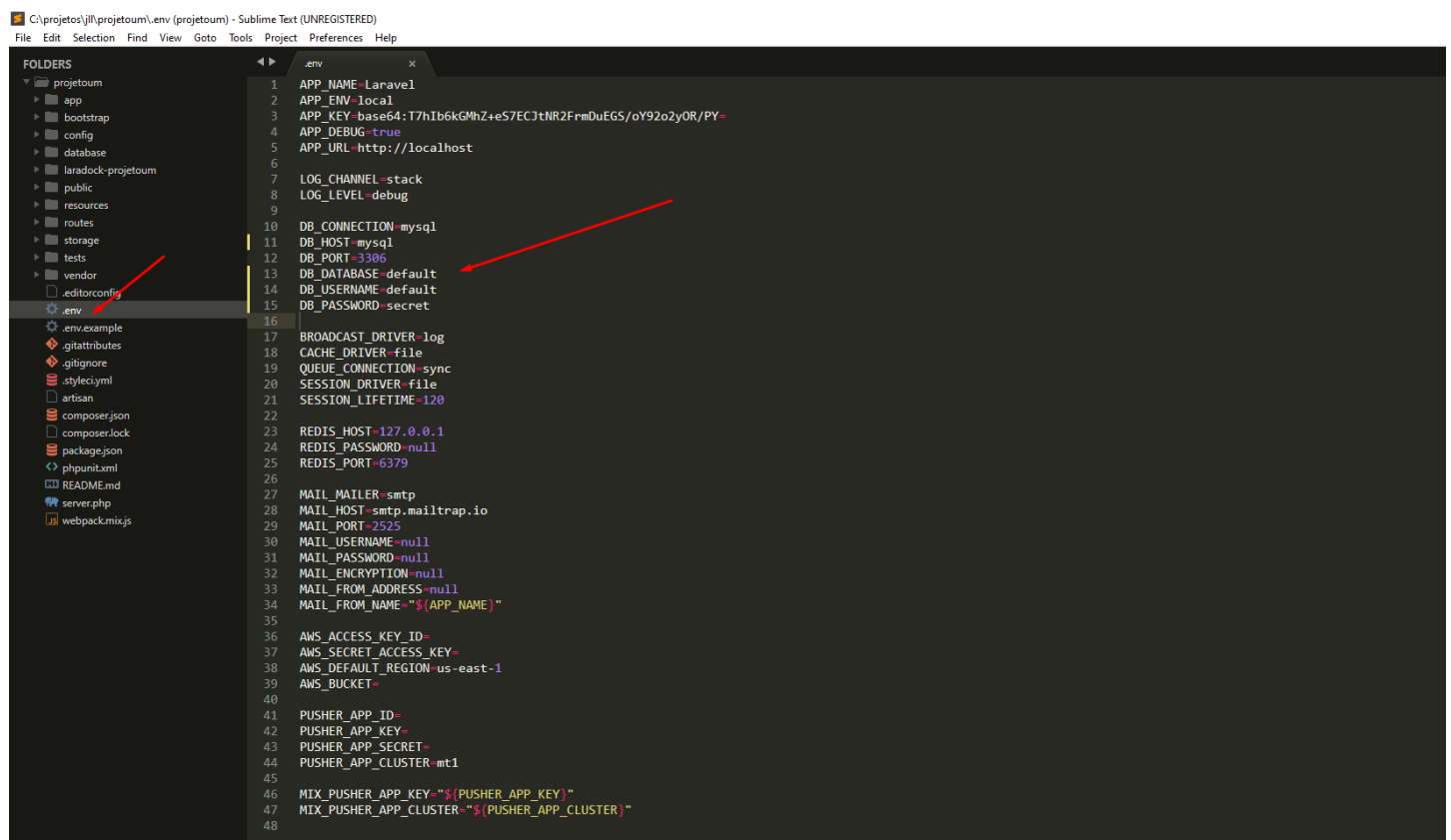
```
root@3385e9cdc76d: /var/www# ls -la
total 301
drwxrwxrwx 1 root root 512 Nov  8 19:53 .
drwxr-xr-x 1 root root 4096 Nov  8 17:22 ..
drwxr-xr-x 1 root root 512 Oct 30 15:07 app
-rwxr-xr-x 1 root root 1686 Oct 30 15:07 artisan
drwxr-xr-x 1 root root 512 Oct 30 15:07 bootstrap
-rw-r--r-- 1 root root 1612 Oct 30 15:07 composer.json
-rw-r--r-- 1 root root 264388 Nov  8 19:43 composer.lock
drwxr-xr-x 1 root root 512 Oct 30 15:07 config
drwxr-xr-x 1 root root 512 Oct 30 15:07 database
-rw-r--r-- 1 root root 220 Oct 30 15:07 editorconfig
-rw-r--r-- 1 root root 845 Nov  8 19:48 .env
-rw-r--r-- 1 root root 794 Oct 30 15:07 .env.example
-rw-r--r-- 1 root root 111 Oct 30 15:07 .gitattributes
-rw-r--r-- 1 root root 163 Oct 30 15:07 .gitignore
drwxrwxrwx 1 root root 512 Nov  8 15:06 laradock
-rw-r--r-- 1 root root 944 Oct 30 15:07 package.json
-rw-r--r-- 1 root root 1202 Oct 30 15:07 phpunit.xml
drwxr-xr-x 1 root root 512 Oct 30 15:07 public
-rw-r--r-- 1 root root 3738 Oct 30 15:07 README.md
drwxr-xr-x 1 root root 512 Oct 30 15:07 resources
drwxr-xr-x 1 root root 512 Oct 30 15:07 routes
-rw-r--r-- 1 root root 563 Oct 30 15:07 server.php
drwxr-xr-x 1 root root 512 Oct 30 15:07 storage
-rw-r--r-- 1 root root 181 Oct 30 15:07 styleci.yml
drwxr-xr-x 1 root root 512 Oct 30 15:07 tests
drwxr-xr-x 1 root root 512 Nov  8 19:48 vendor
-rw-r--r-- 1 root root 559 Oct 30 15:07 webpack.mix.js
root@3385e9cdc76d: /var/www#
```

O que vemos acima é uma instalação normal do Laravel com a pasta do laradock adicionada.

Agora basta acessar o seu browser e ver que aplicação já está rodando:



Agora, para que a nossa aplicação possa acessar o banco de dados do Docker, basta alterar o .env para acessar o BD que configuramos nos passos anteriores:



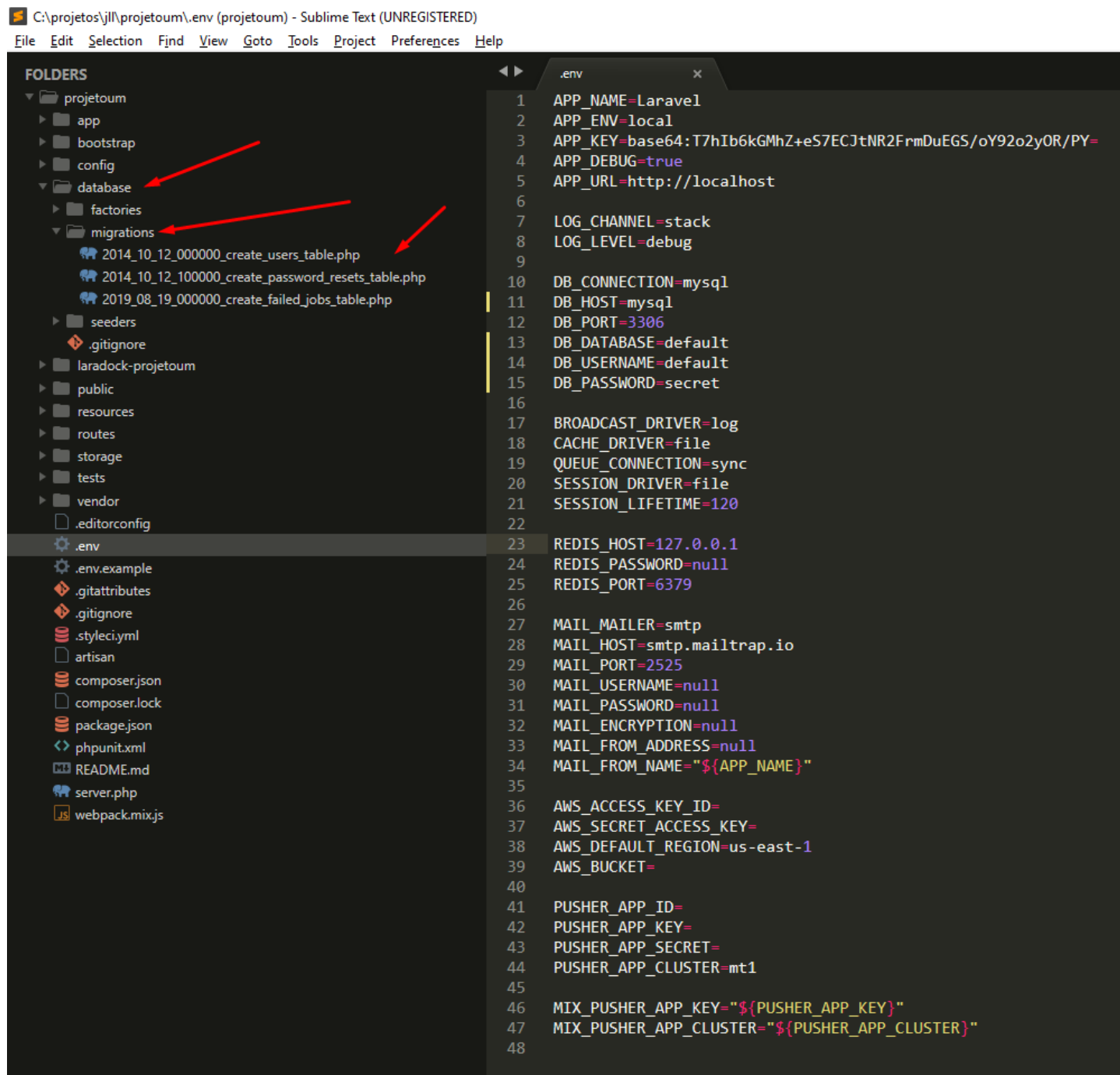
Agora para testar de fato a conexão, vamos para o próximo passo, Migrations 😎.

9. MIGRATIONS

Para mais informações, leia a doc oficial:

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

Migrations são como um controle de versão do seu banco de dados, permitindo que modifiquemos e compartihemos a estrutura do banco. Por padrão, a versão 8 do Laravel já vem com 8 migrations, como se vê no print abaixo:



The screenshot shows a Sublime Text editor window titled "C:\projetos\jill\projeto\env (projeto) - Sublime Text (UNREGISTERED)". The left sidebar displays the "FOLDERS" panel with a tree view of the project structure. The "database" folder is expanded, showing "migrations" and "seeders". Three red arrows point to the "migrations" folder, the "2014_10_12_000000_create_users_table.php" file, and the "2014_10_12_100000_create_password_resets_table.php" file. The main editor area displays the contents of the ".env" file, which includes configuration for the application name, environment, key, debug mode, URL, logging, database connection, cache, queue, session, mail, and AWS services.

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:T7hIb6kGMhZ+eS7ECJtNR2FrmDuEGS/oY92o2y0R/PY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_LEVEL=debug
9
10 DB_CONNECTION=mysql
11 DB_HOST=mysql
12 DB_PORT=3306
13 DB_DATABASE=default
14 DB_USERNAME=default
15 DB_PASSWORD=secret
16
17 BROADCAST_DRIVER=log
18 CACHE_DRIVER=file
19 QUEUE_CONNECTION=sync
20 SESSION_DRIVER=file
21 SESSION_LIFETIME=120
22
23 REDIS_HOST=127.0.0.1
24 REDIS_PASSWORD=null
25 REDIS_PORT=6379
26
27 MAIL_MAILER=smtp
28 MAIL_HOST=smtp.mailtrap.io
29 MAIL_PORT=2525
30 MAIL_USERNAME=null
31 MAIL_PASSWORD=null
32 MAIL_ENCRYPTION=null
33 MAIL_FROM_ADDRESS=null
34 MAIL_FROM_NAME="${APP_NAME}"
35
36 AWS_ACCESS_KEY_ID=
37 AWS_SECRET_ACCESS_KEY=
38 AWS_DEFAULT_REGION=us-east-1
39 AWS_BUCKET=
40
41 PUSHER_APP_ID=
42 PUSHER_APP_KEY=
43 PUSHER_APP_SECRET=
44 PUSHER_APP_CLUSTER=mt1
45
46 MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
47 MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
48
```

Cada uma delas possui o "script" para alterar e para desfazer as alterações feitas por ela própria. No exemplo abaixo, a migration possui instruções para criar a tabela de usuários e também para desfazer essas alterações, o que seria equivalente a deletar esta tabela:

C:\projetos\jlf\projeto\database\migrations\2014_10_12_000000_create_users_table.php (projeto) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

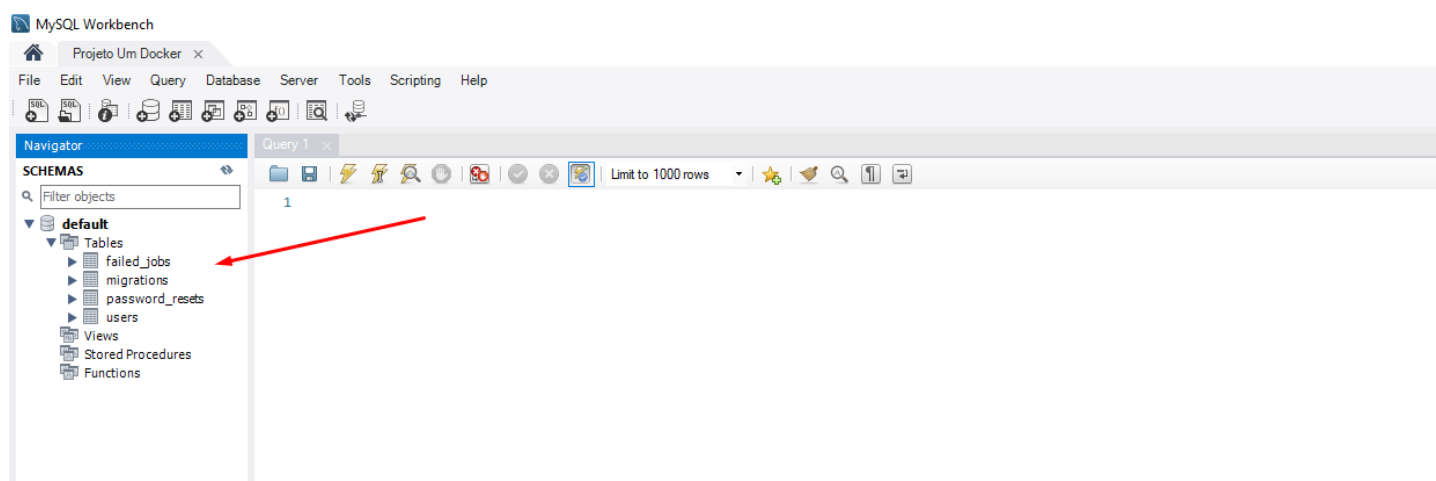
- projeto
- app
- bootstrap
- config
- database
 - factories
 - migrations
 - 2014_10_12_000000_create_users_table.php
 - 2014_10_12_100000_create_password_resets_table.php
 - 2019_08_19_000000_create_failed_jobs_table.php
 - seeders
- .gitignore
- laradock-projet
- public
- resources
- routes
- storage
- tests
- vendor
 - .editorconfig
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - .styleci.yml
 - artisan
 - composer.json
 - composer.lock
 - package.json
 - phpunit.xml
 - README.md
 - server.php
 - webpack.mix.js

```
1 k?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateUsersTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->id();
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->timestamp('email_verified_at')->nullable();
21             $table->string('password');
22             $table->rememberToken();
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32     public function down()
33     {
34         Schema::dropIfExists('users');
35     }
36 }
37
```

Para rodar as migrations pendentes, basta executar o comando abaixo de dentro do workspace do docker:

```
root@3385e9cdc76d:/var/www# php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (379.52ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (126.16ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (139.39ms)
root@3385e9cdc76d:/var/www#
```

Como podem ver as migrations foram rodadas com sucesso, podemos, inclusive, checar no banco de dados que as alterações foram executadas com sucesso:



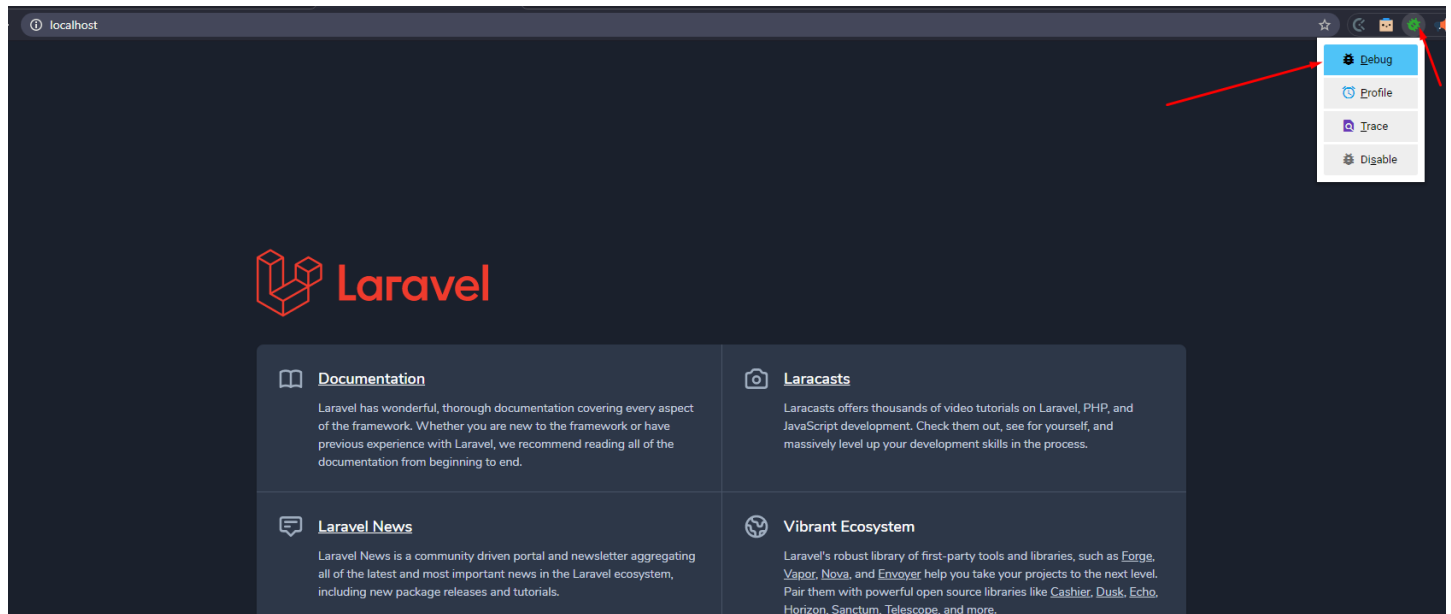
10. XDEBUG

10.1.1. Instalando a extensão no chrome:

Para o Rodar o Xdebug, o primeiro passo é instalar esta extensão no chrome:

<https://chrome.google.com/webstore/detail/xdebug-helper/eadndfjplgieldjbigjakmdgkmoaaaoc>

Após instalar, habilite-a clicando no ícone abaixo:

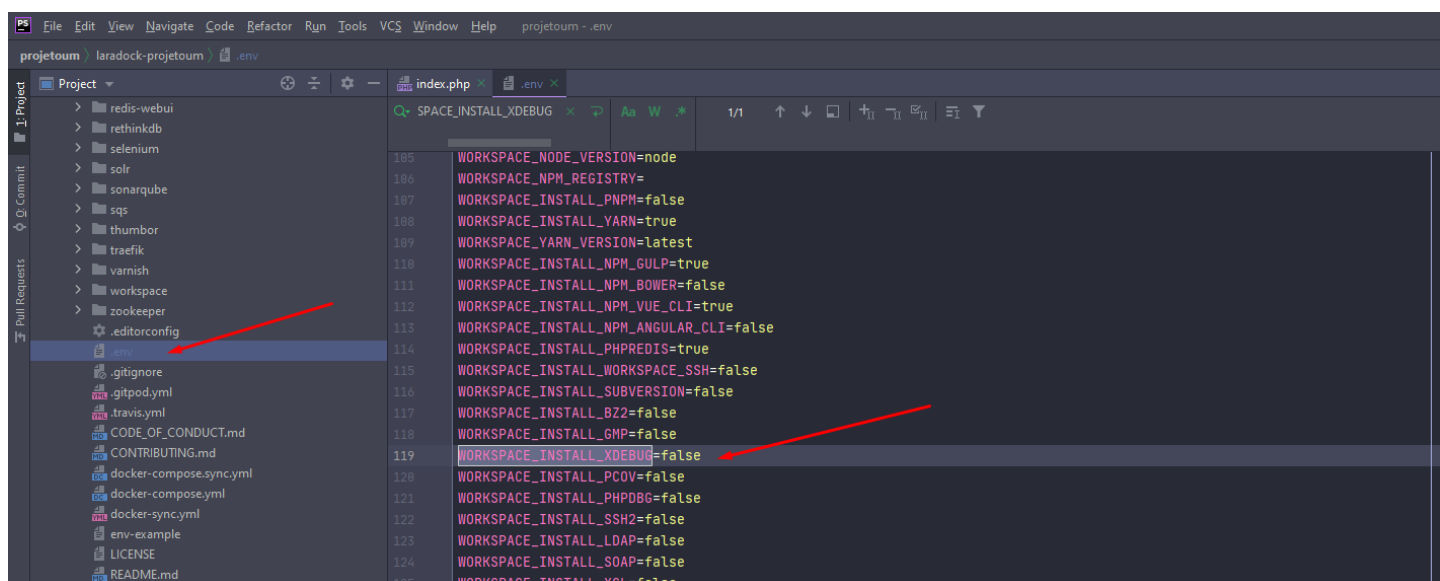


10.1.2. Habilitando no Laradock:

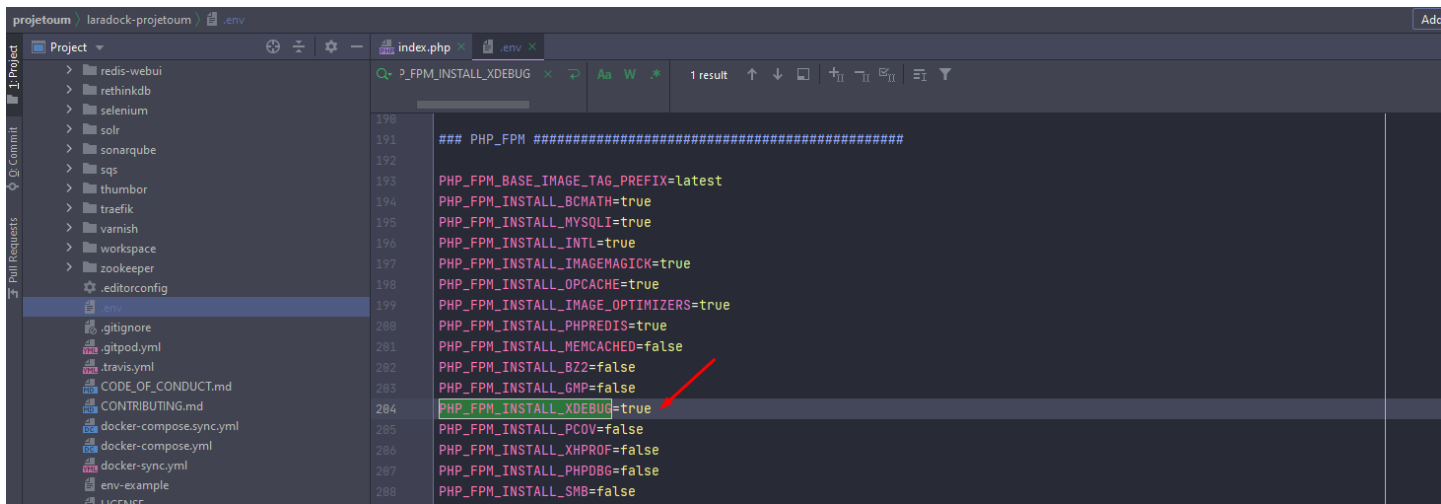
First install xDebug in the Workspace and the PHP-FPM Containers:

Primeiro, instale o XDebug no workspace no no container PHP-FPM:

- Abra o arquivo .env do Laradock
- Procure pela variável WORKSPACE_INSTALL_XDEBUG



- Mude o seu valor para true
- Procure pela variável PHP_FPM_INSTALL_XDEBUG e mude-a para true



De dentro da raiz da pasta do Laradock, para todos os containers e rode o comando abaixo para fazer o Re-build dos containers e assim instalar o Xdebug:

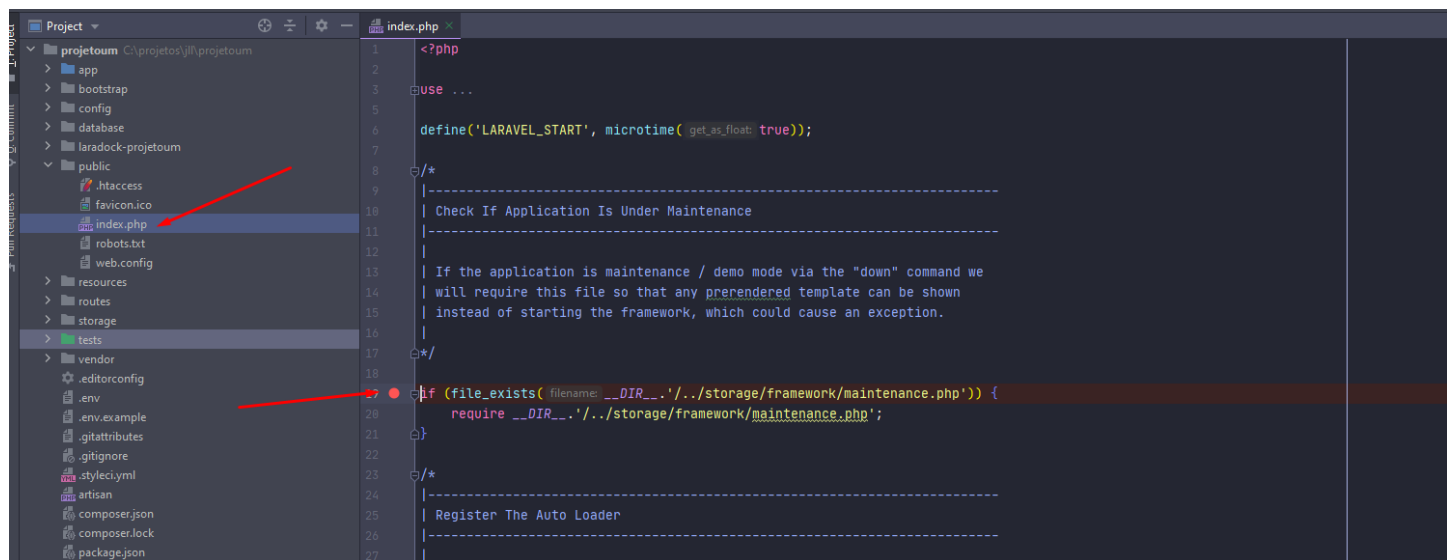
```
docker-compose build php-fpm
```

ou

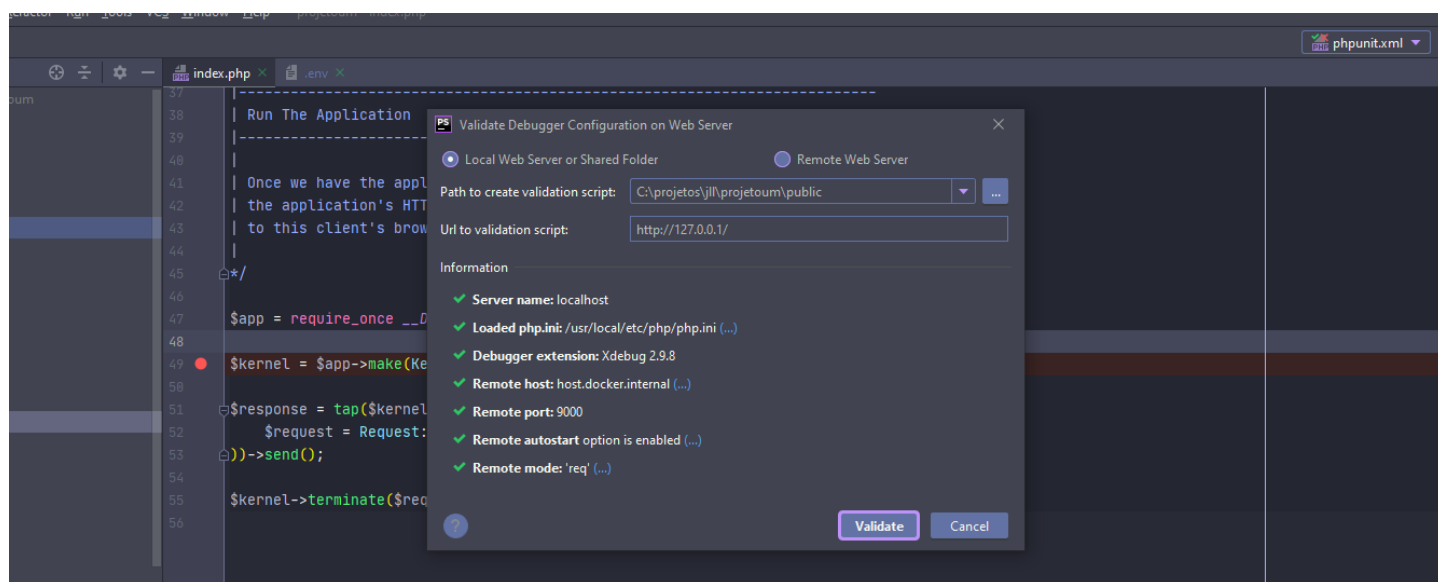
```
docker-compose up -d --no-deps -build nginx mysql
```

10.1.3. PHPStorm:

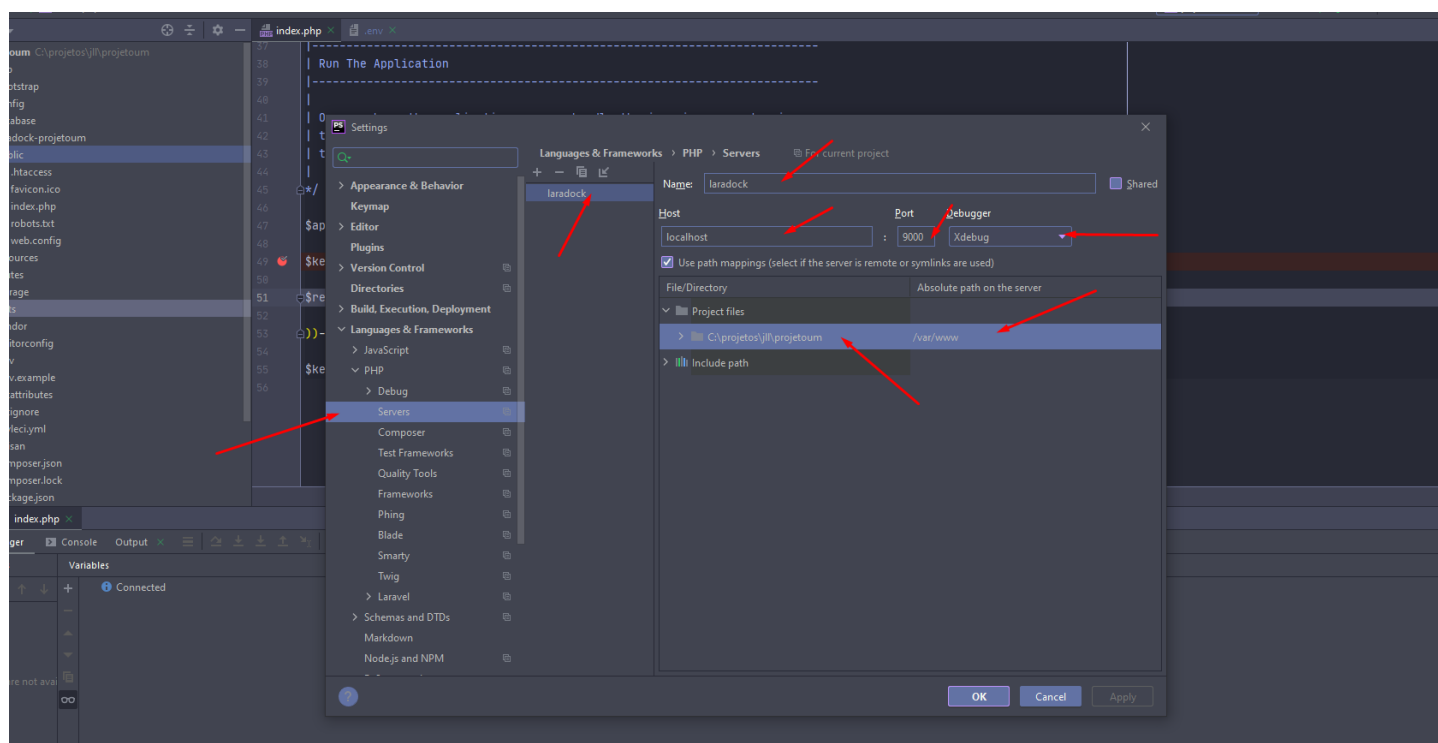
Abra o projeto no PHP Storm e coloque um breakpoint no arquivo index.php, clicando no espaço entre a linha e o código:



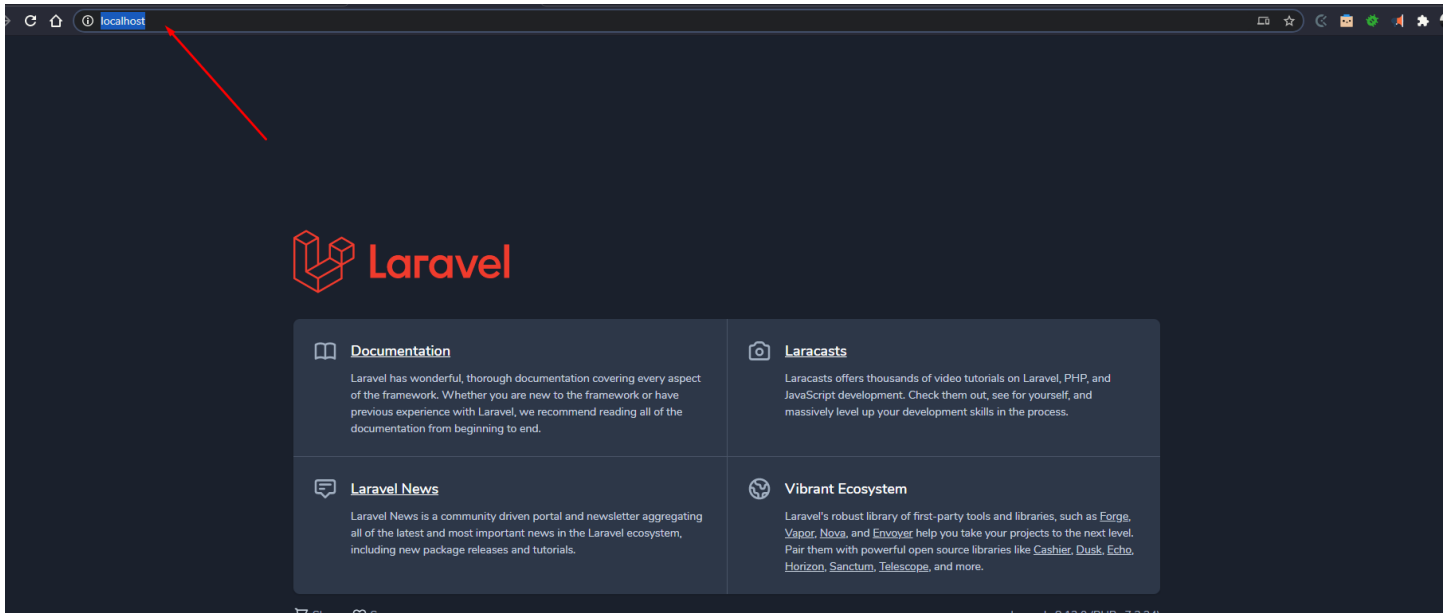
Diga ao PHPStorm, para “ouvir” conexões de depuração:



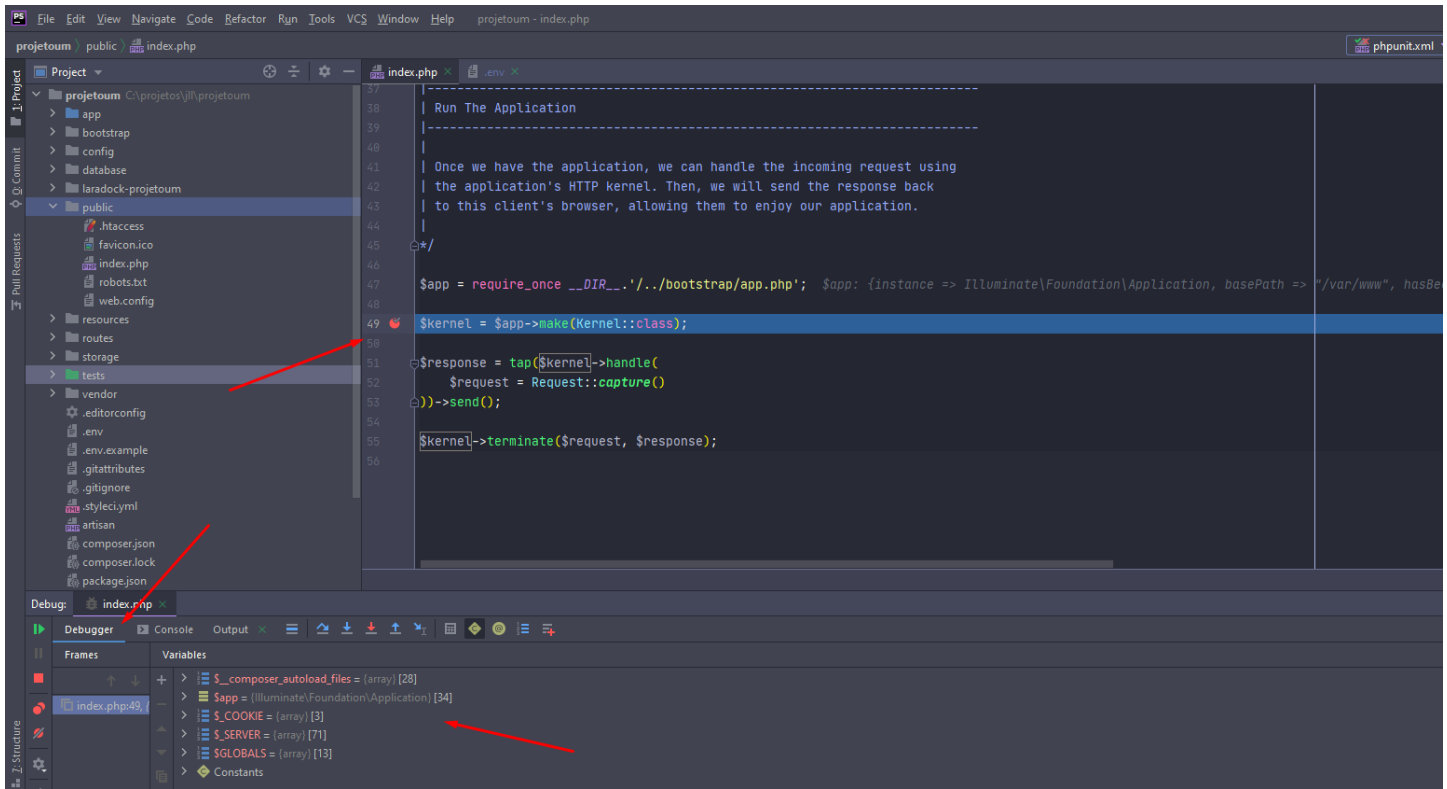
Em Settings, configure um novo servidor, pois ao utilizar Docker, é como se estivesse depurando um servidor remoto:



Recarregue a página do sistema:

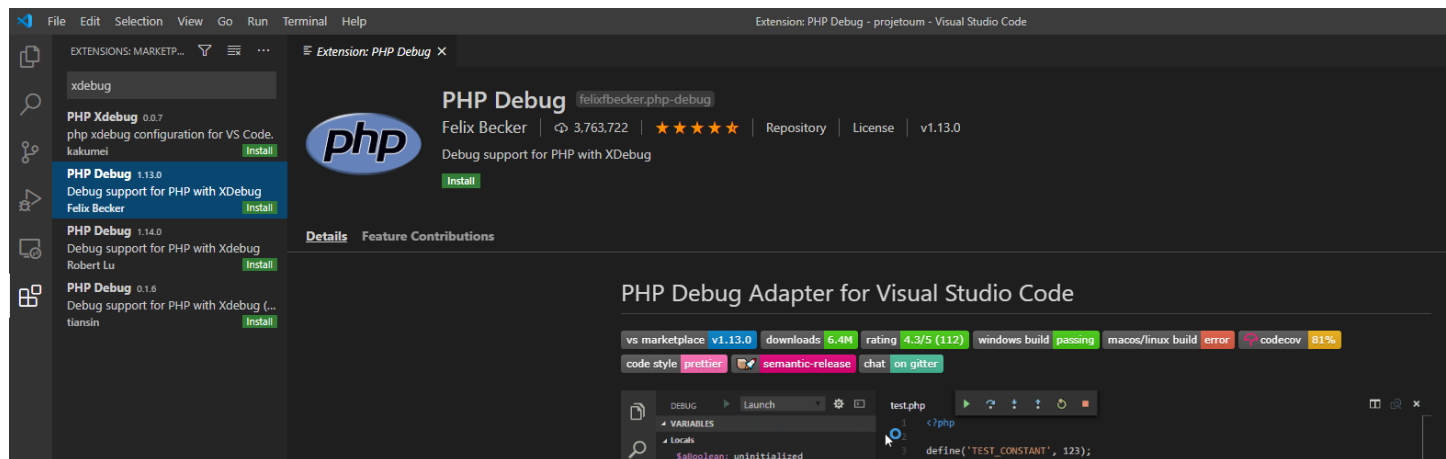


Pronto, o depurador estará funcionando normalmente:

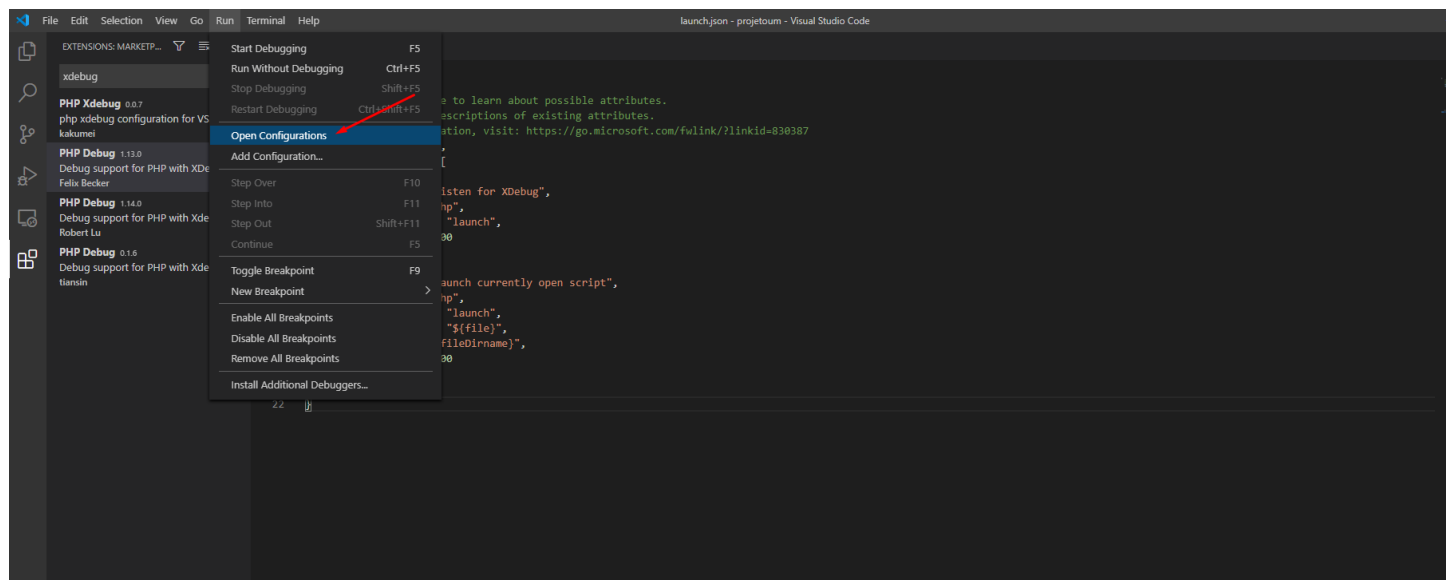


10.1.4. VSCode:

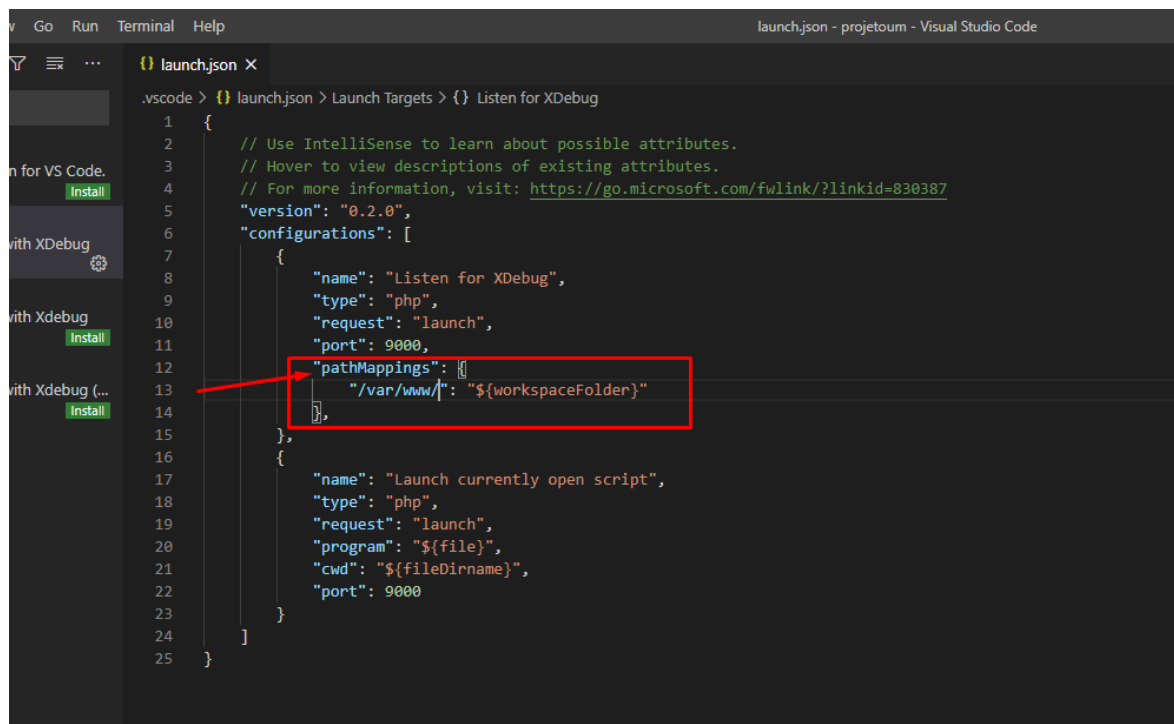
Primeiro, instale a seguinte extensão:



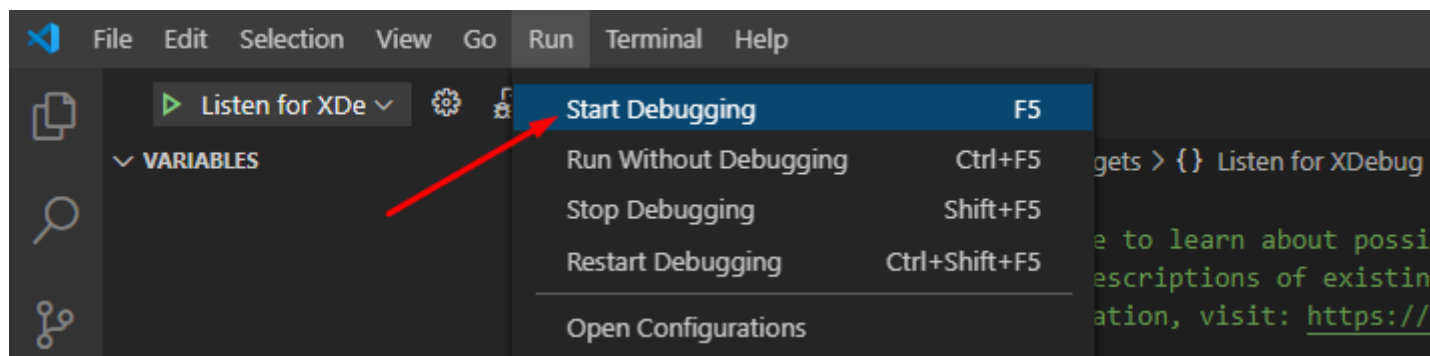
Configure o depurador:



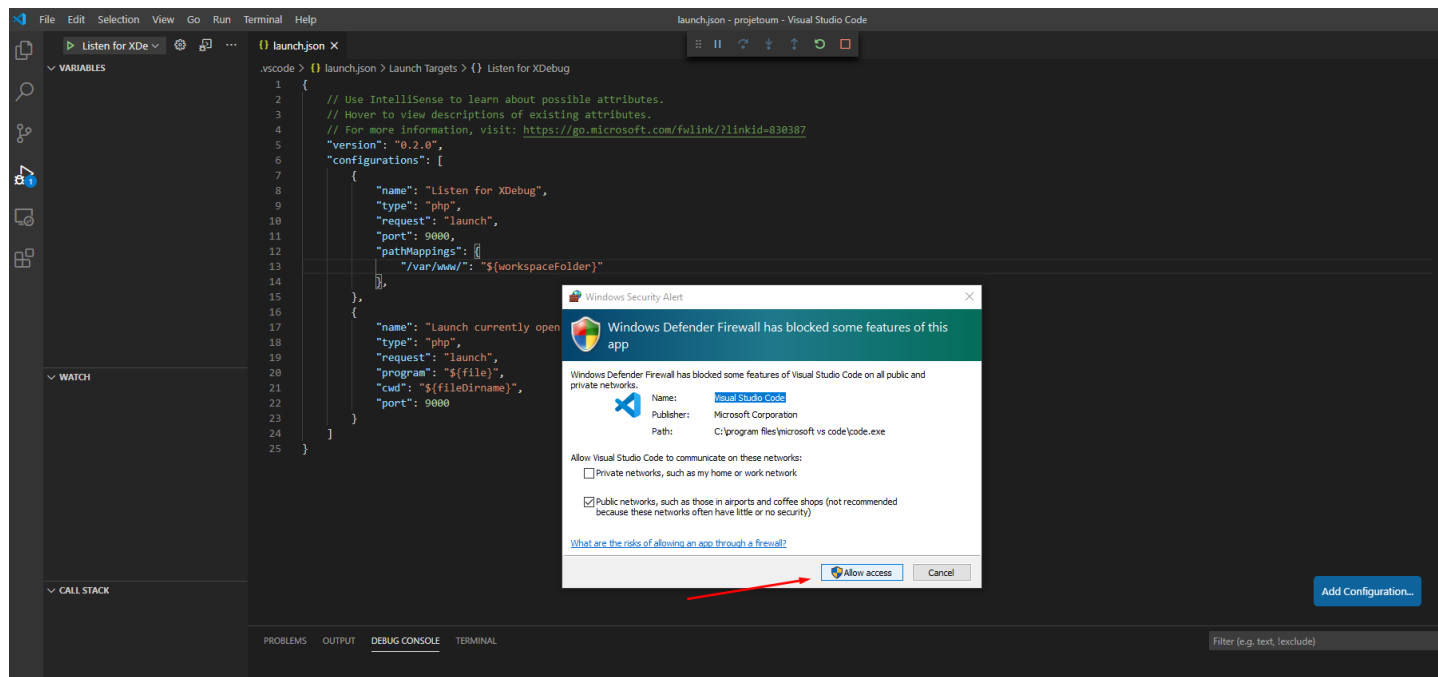
Abaixo de “port”, coloque a seguinte configuração:



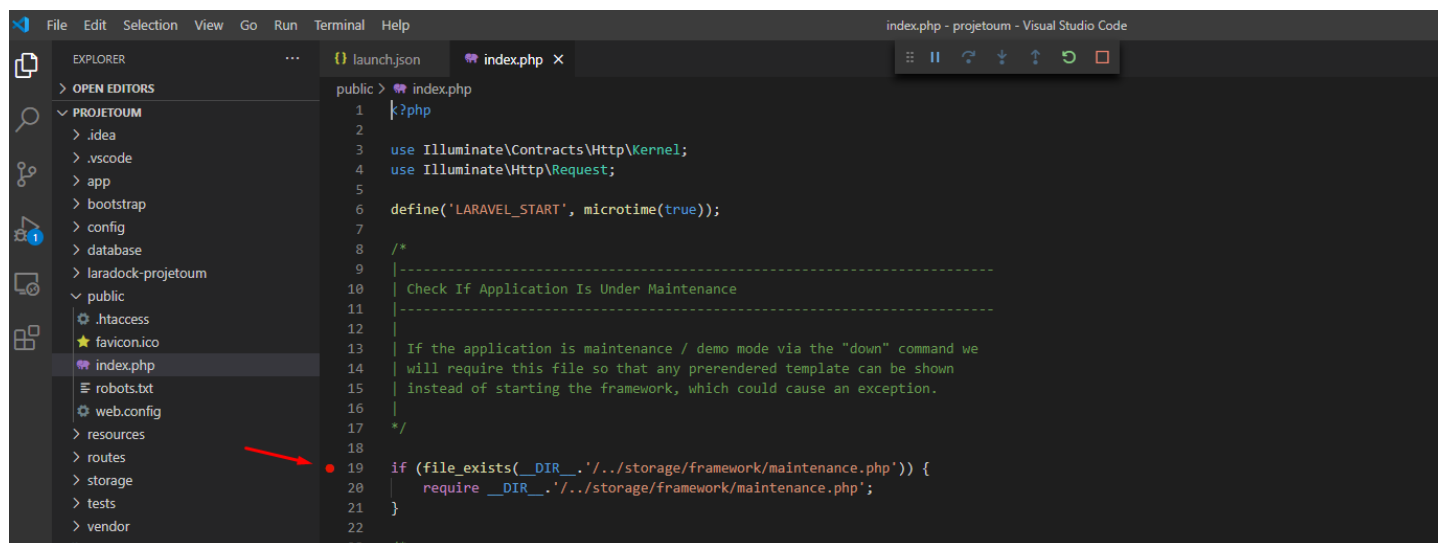
Clique em “Start Debugging”:



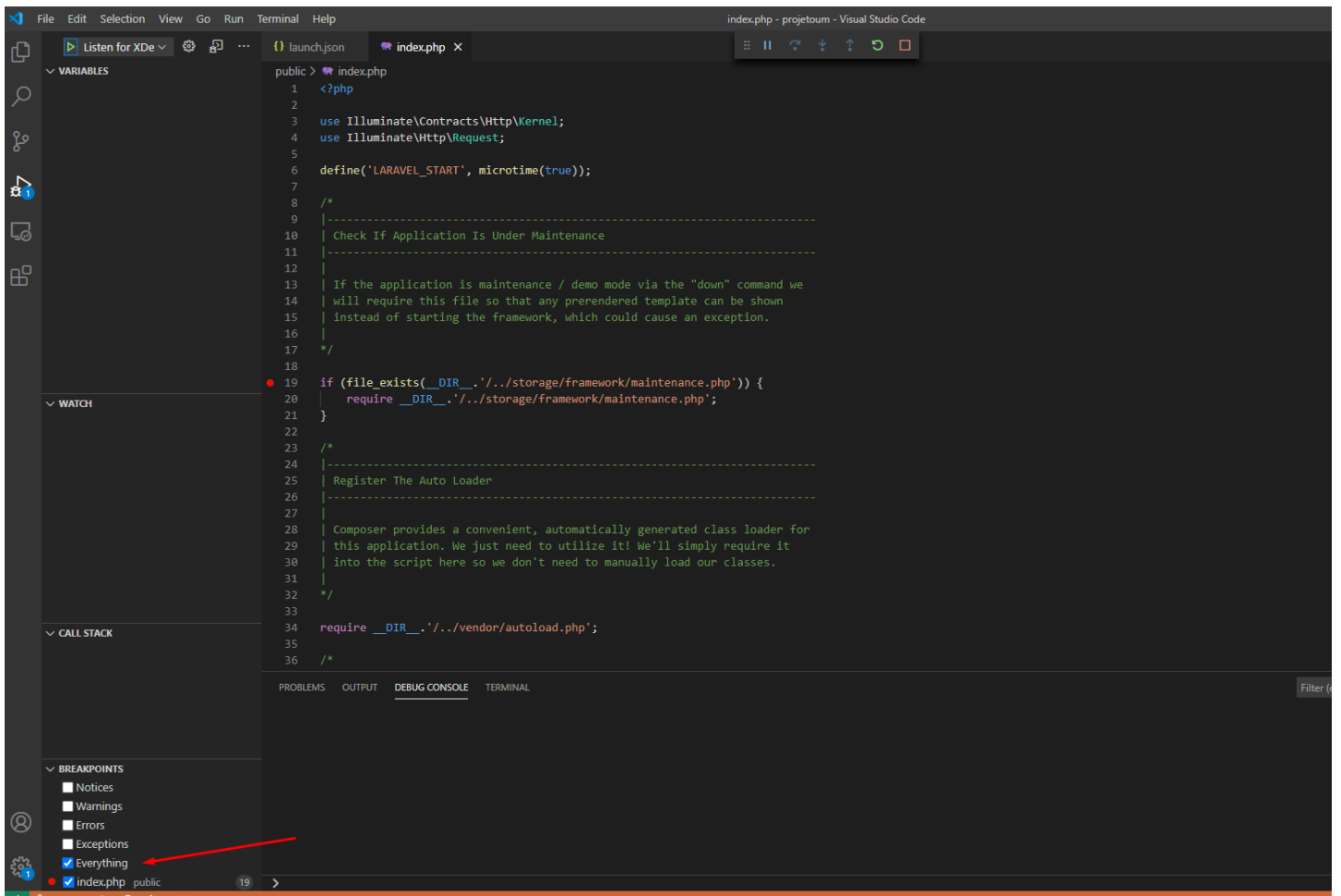
Permita a conexão:



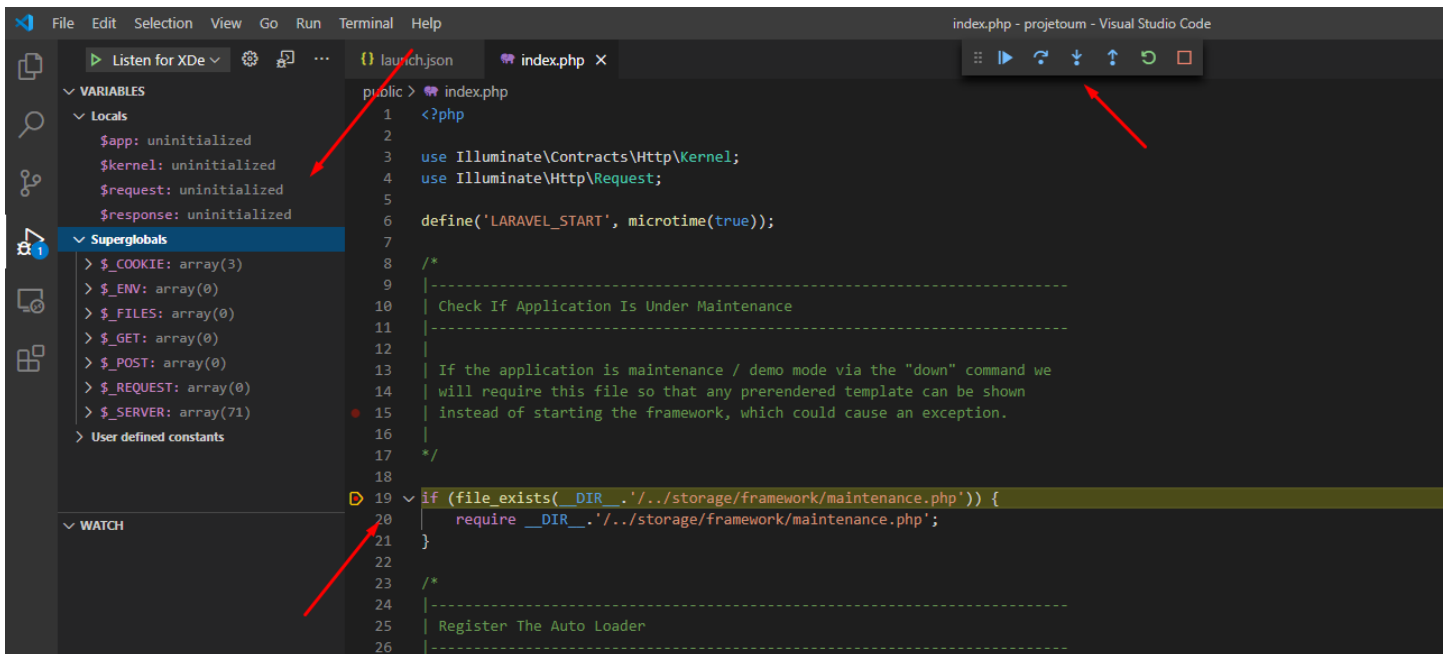
Marque um breakpoint:



Desmarque o depurador para não parar em todos os warnings, erros e etc, mas parar apenas no breakpoint:



Sucesso!! Seu depurador já está funcionando:



11. COMANDOS ÚTEIS DO DOCKER

Listar serviços em execução:

```
docker ps
```

Acessar o terminal do serviço de um container:

```
docker exec -it php56 bash
```

Recriar o container:

```
docker-compose up --build --force-recreate
```

Recriar um serviço específico do container:

```
docker-compose up -d --no-deps --build php56
```

Limpar todos containers

```
docker system prune
```

Limpar todos containers + Dados

```
docker system prune -a
```