

**PAULO HENRIQUE DA SILVA CORREIA**

**ATIVIDADE DOCKER**

**SÃO JOSÉ DOS CAMPOS**

**2020**

**PAULO HENRIQUE DA SILVA CORREIA**

## SUMÁRIO

<b>RESUMO.....</b>	<b>3</b>
<b>OBSERVAÇÕES GERAIS.....</b>	<b>3</b>
<b>INSTALANDO DOCKER .....</b>	<b>3</b>
1.1 INSTALAÇÃO.....	3
1.2 TESTANDO A INSTALAÇÃO .....	4
<b>PROJETO .....</b>	<b>4</b>
1.1 INTRODUÇÃO .....	4
1.2 PREPARAÇÃO.....	4
1.3 CRIANDO O PROJETO .....	5
1.4 BANCO DE DADOS .....	5
1.5 APLICAÇÃO NODE.....	11
1.6 APLICAÇÃO PHP.....	16
<b>2 CONCLUSÕES .....</b>	<b>19</b>
<b>REFERÊNCIAS.....</b>	<b>20</b>

## RESUMO

Atividade Docker

**Objetivo:** Criar um serviço Docker integrando dois containers no serviço.

## OBSERVAÇÕES GERAIS

Para a criação do Docker, estarei utilizando o sistema operacional Linux Ubuntu 20 LTS

## INSTALANDO DOCKER

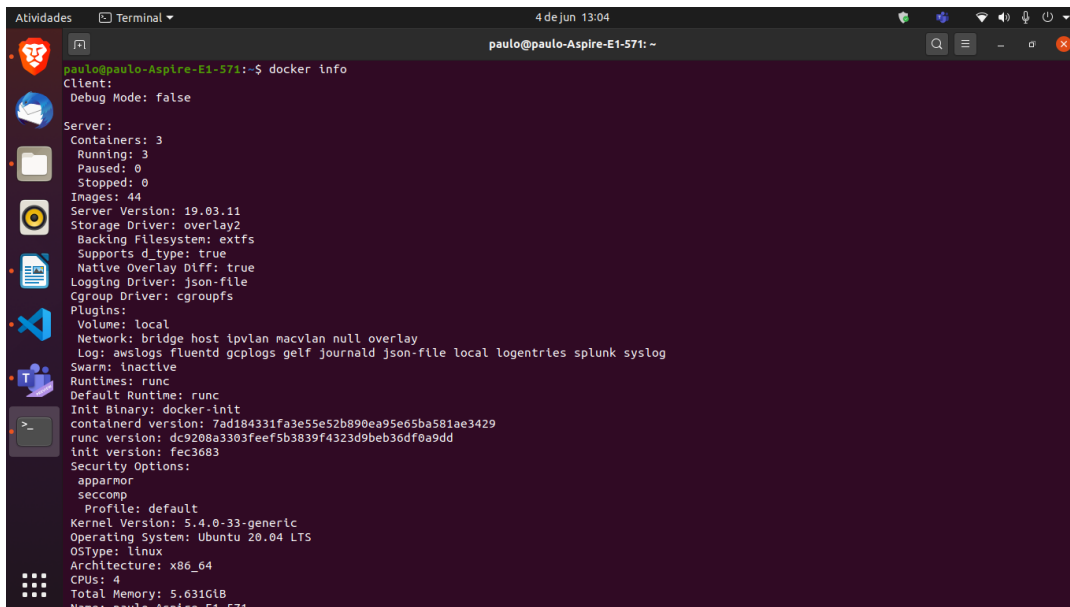
### 1.1 INSTALAÇÃO

Iremos instalar o Docker CE (Community Edition), para isso abra o terminal do Linux e digite os seguintes comandos.

1. `Curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
  2. `sudo add-apt-repository "deb`
  3. `sudo apt install software-properties-common -y`
  4. `sudo apt install docker-ce -y`
  5. `sudo usermod -aG docker $USER`
  6. Após estes passos, reinicie o computador para aplicar as alterações
- Aviso: o comando do item 5 adiciona o usuário atual ao grupo "docker".

## 1.2 TESTANDO A INSTALAÇÃO

Depois de ter reiniciado o computador abra o terminal do Ubuntu e digite “Docker info”, irá aparecer uma tela semelhante a imagem de baixo.

A screenshot of a terminal window on a Ubuntu system. The terminal title bar shows 'Atividades' and 'Terminal'. The user is logged in as 'paulo' on a machine named 'paulo-Aspire-E1-571'. The command 'docker info' has been executed, displaying detailed information about the Docker environment. The output includes details about the client, server, containers, images, storage driver, logging driver, plugins, swarm, runtimes, and system specifications like kernel version and architecture.

```
paulo@paulo-Aspire-E1-571:~$ docker info
Client:
 Debug Mode: false

Server:
 Containers: 3
  Running: 3
  Paused: 0
  Stopped: 0
 Images: 44
 Server Version: 19.03.11
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs

Plugins:
 Volumes: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog

Swarm: inactive

Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb30df0a9dd
 init version: fec3683

Security Options:
 apparmor
 seccomp
  Profile: default

Kernel Version: 5.4.0-33-generic
 Operating System: Ubuntu 20.04 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 5.63GiB
 Name: paulo-Aspire-E1-571
```

## PROJETO

### 1.1 INTRODUÇÃO

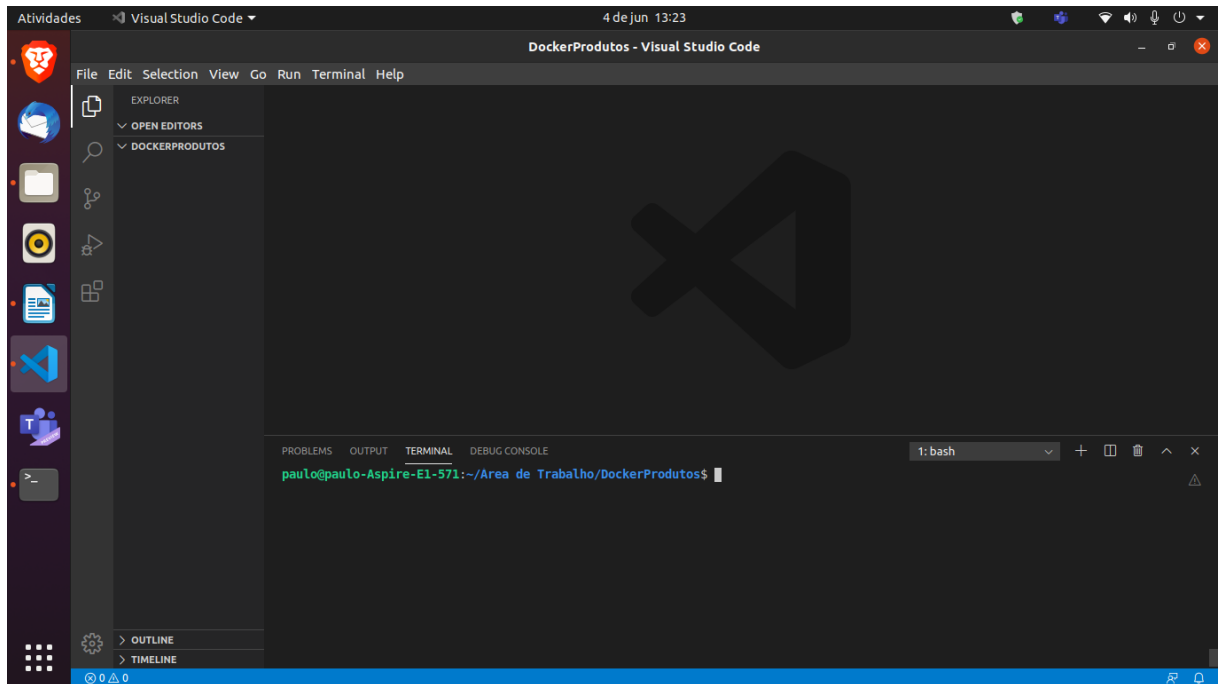
O projeto que iremos fazer consiste em uma aplicação PHP que lê um JSON vindo de uma aplicação NODE.js que puxa de um banco de dados MySQL.

### 1.2 PREPARAÇÃO

Para iniciarmos o nosso projeto, primeiro iremos precisar de um editor de texto, estarei utilizando o Visual Studio Code, caso você queira instalar na sua máquina, siga o tutorial deste link: <<https://www.edivaldobrito.com.br/visual-studio-code-no-ubuntu/>>

### 1.3 CRIANDO O PROJETO

Vamos criar a nossa pasta onde irá conter todos os arquivos fontes que precisaremos. Estarei criando a pasta “DockerProdutos” na área de trabalho e abrindo esta pasta com o Visual Studio Code.



### 1.4 BANCO DE DADOS

Para o nosso banco vamos criar a pasta “api” e dentro desta pasta criaremos outra chamada “db”. Dentro desta pasta “db” criaremos um arquivo chamado “Dockerfile” e dentro dele colocaremos o seguinte conteúdo:

```
FROM mysql
MYSQL_ROOT_PASSWORD 123
```

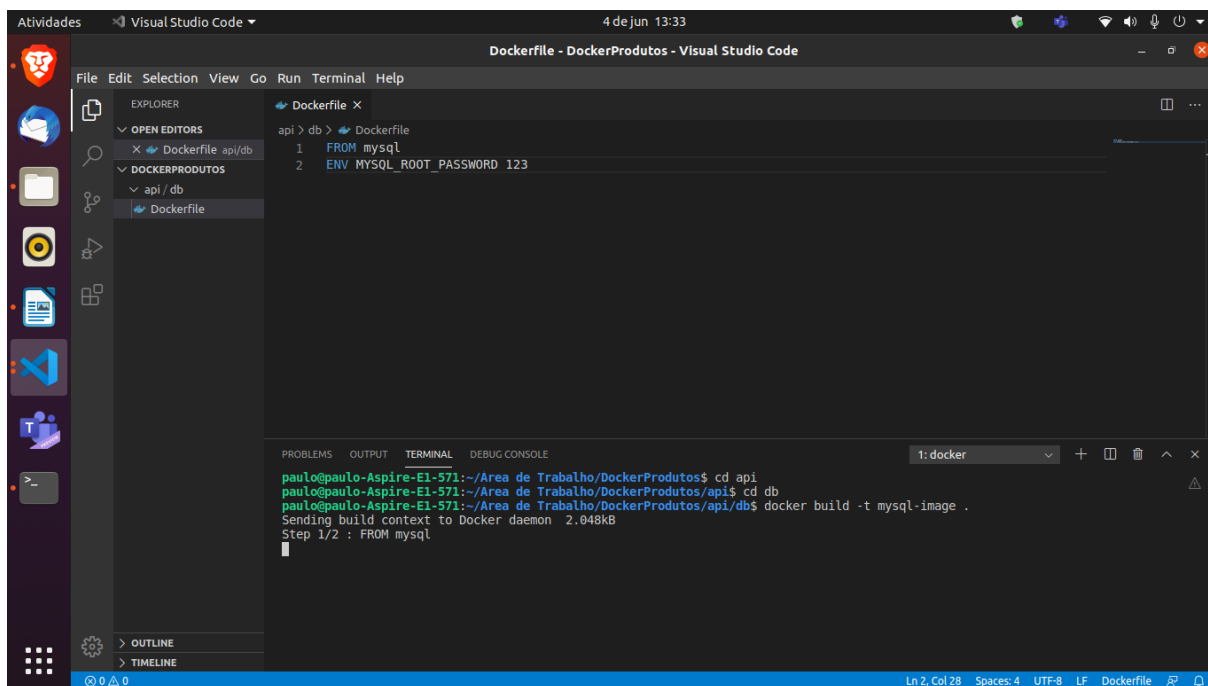
Explicando o que foi escrito, o FROM diz ao Docker que iremos puxar uma imagem e separado por espaço o nome da imagem que será baixada que no nosso caso é o mysql.

Na segunda linha estamos passando qual será a senha do root, no caso “123”.

Agora vamos criar a nossa imagem docker personalizada com base nesta imagem do mysql. Para isto, abra o terminal do Visual Studio Code e navegue até a pasta “db”. Em seguida digite o seguinte comando:

“docker build -t mysql-image .”

Este comando chama o docker dando um build no arquivo Dockerfile e daremos o nome de “mysql-image” (-t significa tag (apelido)). Durante a execução do comando irá baixar alguns arquivos como nas imagens a baixo.

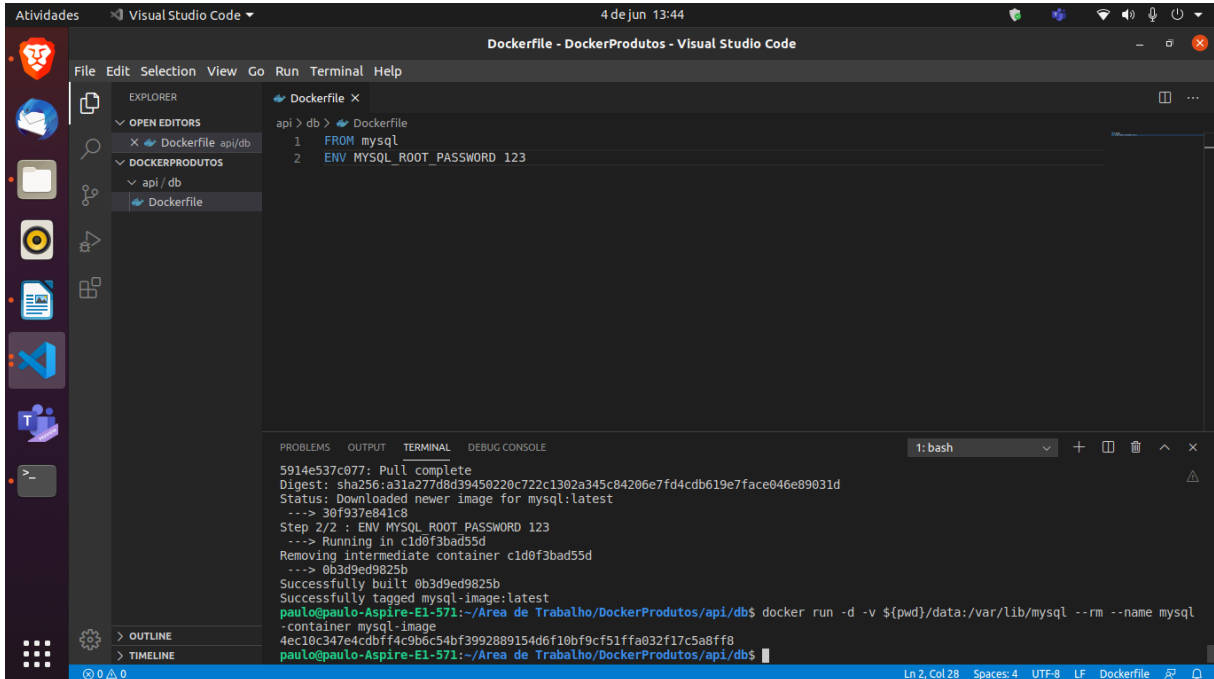


Agora que já realizamos o build da nossa imagem do MySQL, vamos levantar o container. Para isso, ainda na pasta “db” que é onde o arquivo “Dockerfile” se encontra, digite o seguinte comando:

```
docker run -d -v ${pwd}/data:/var/lib/mysql --rm --name mysql-container mysql-image
```

Explicando, iremos chamar a aplicação “docker” pedindo para executar em forma de background (-d) compartilhando nossa pasta local com o container para que quando desligarmos o computador e ligarmos novamente, teremos nosso banco com todos os registros(-v \${pwd}/data:/var/lib/mysql). Removeremos qualquer outro container com este nome (--rm) e daremos o nome do container que estamos subindo de mysql-container utilizando o mysql-imagem (apelido que demos mais cedo).

Após executarmos o comando, deverá retornar um resultado semelhante a este:



The screenshot shows the Visual Studio Code interface with the Dockerfile editor open. The Dockerfile contains the following content:

```
api > db > Dockerfile
1 FROM mysql
2 ENV MYSQL_ROOT_PASSWORD 123
```

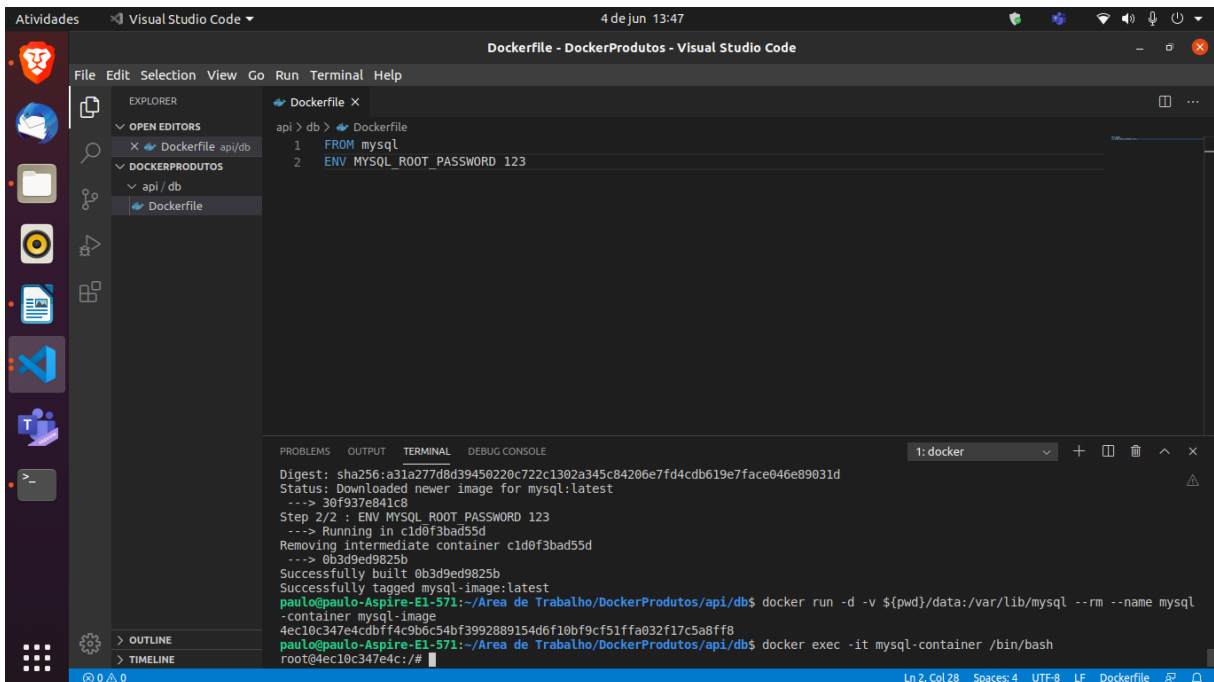
The terminal output shows the following steps:

```
5914e537c077: Pull complete
Digest: sha256:a31a277d8d39450220c722c1302a345c84206e7fd4cdb619e7face046e89031d
Status: Downloaded newer image for mysql:latest
----> 30f937e841c8
Step 2/2 : ENV MYSQL_ROOT_PASSWORD 123
----> Running in c1d0f3bad55d
Removing intermediate container c1d0f3bad55d
----> 0b3d9ed9825b
Successfully built 0b3d9ed9825b
Successfully tagged mysql-image:latest
paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos/api/db$ docker run -d -v ${pwd}/data:/var/lib/mysql --rm --name mysql
- container mysql-image
4ec10c347e4cdbff4c9b6c54bf3992889154d6f10bf9cf51ffa032f17c5a8ff8
paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos/api/db$
```

Agora que conseguimos levantar o mysql-container, vamos verificar se o mysql está conectando normalmente, para isso digite o seguinte comando:

```
docker exec -it mysql-container /bin/bash
```

Explicando, estamos chamando a aplicação docker pedindo para executar de forma interativa (-it) a pasta “bin/bash” do mysql-container. Se o container subiu com sucesso teremos uma resposta igual a esta.



The screenshot shows the Visual Studio Code interface with the Dockerfile editor open. The Dockerfile contains the following content:

```
api > db > Dockerfile
1 FROM mysql
2 ENV MYSQL_ROOT_PASSWORD 123
```

The terminal output shows the following steps:

```
Digest: sha256:a31a277d8d39450220c722c1302a345c84206e7fd4cdb619e7face046e89031d
Status: Downloaded newer image for mysql:latest
----> 30f937e841c8
Step 2/2 : ENV MYSQL_ROOT_PASSWORD 123
----> Running in c1d0f3bad55d
Removing intermediate container c1d0f3bad55d
----> 0b3d9ed9825b
Successfully built 0b3d9ed9825b
Successfully tagged mysql-image:latest
paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos/api/db$ docker run -d -v ${pwd}/data:/var/lib/mysql --rm --name mysql
- container mysql-image
4ec10c347e4cdbff4c9b6c54bf3992889154d6f10bf9cf51ffa032f17c5a8ff8
paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos/api/db$ docker exec -it mysql-container /bin/bash
root@4ec10c347e4c:/#
```

Após a execução, digite “exit” para voltar ao terminal do Visual Studio Code. Agora vamos criar um arquivo .sql contendo a criação do database e das tabelas que iremos utilizar no projeto. Dentro da pasta “db” criaremos um arquivo .sql com o nome de “database.sql” e dentro dele colocaremos as seguintes informações:

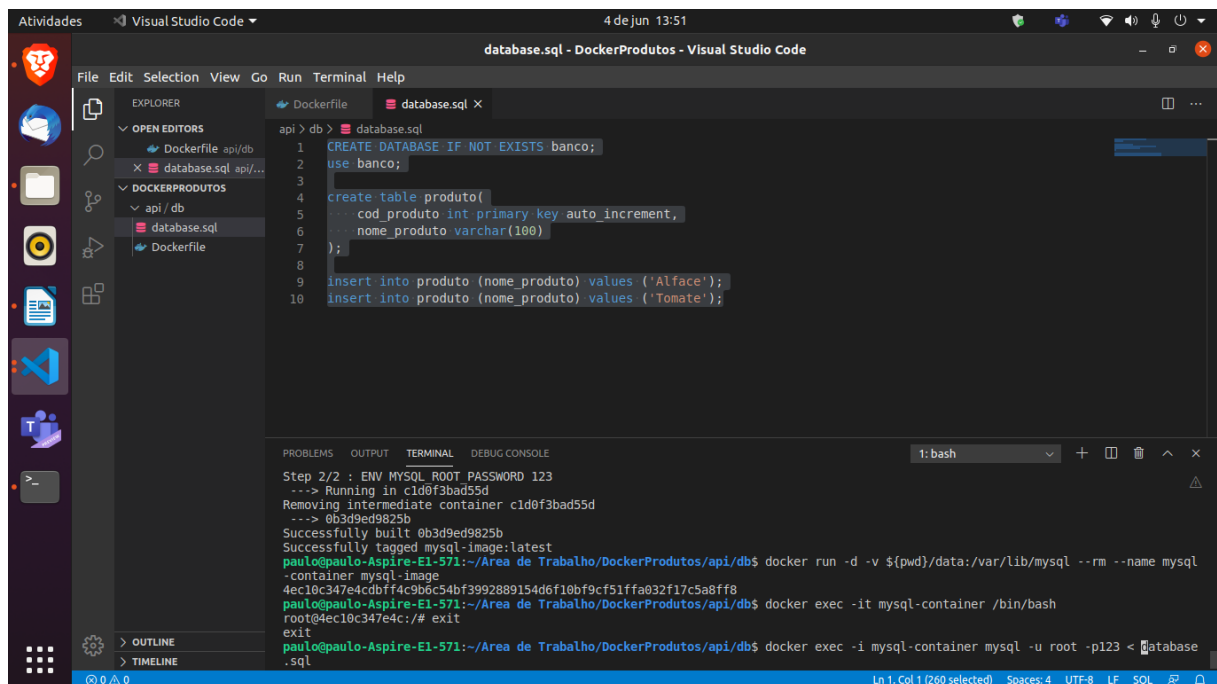
```
CREATE DATABASE IF NOT EXISTS banco;
use banco;
```

```
create table produto(
  cod_produto int primary key auto_increment,
  nome_produto varchar(100)
);
```

```
insert into produto (nome_produto) values ('Alface');
insert into produto (nome_produto) values ('Tomate');
```

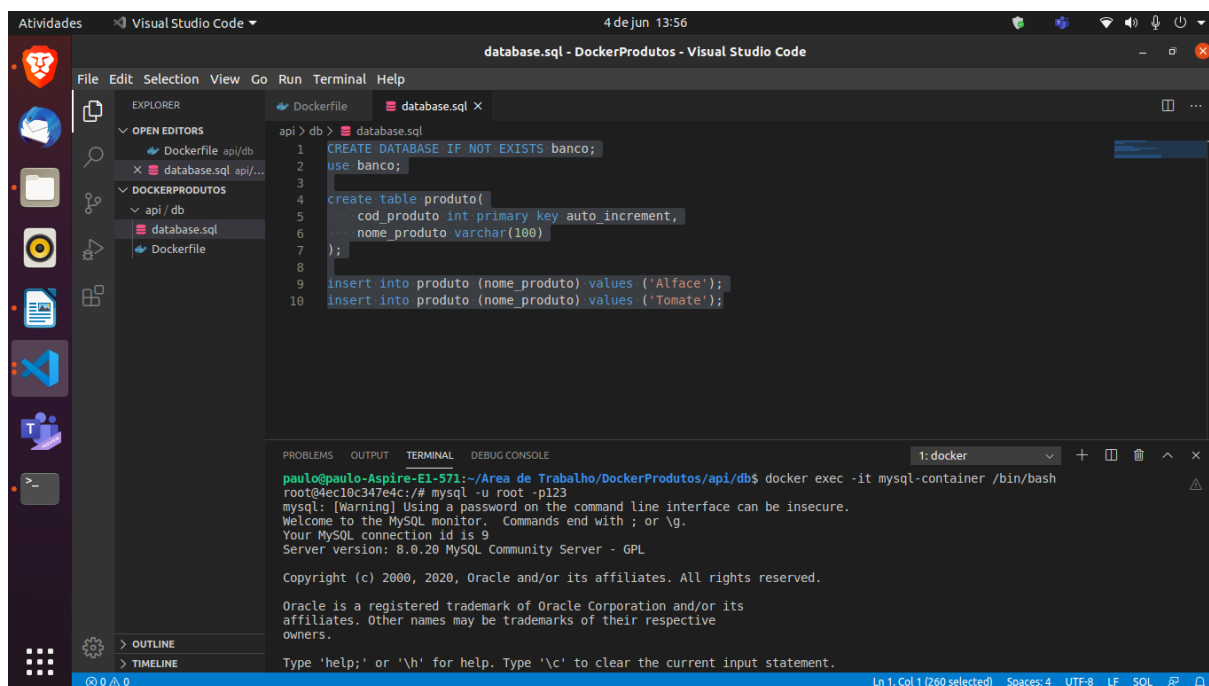
Agora para importarmos este arquivo para o nosso banco de dados, iremos digitar o seguinte comando com o mysql-container ainda rodando (Para verificar, digite “docker ps”, será exibido os containers ativos).

```
docker exec -i mysql-container mysql -u root -p123 < database.sql
```





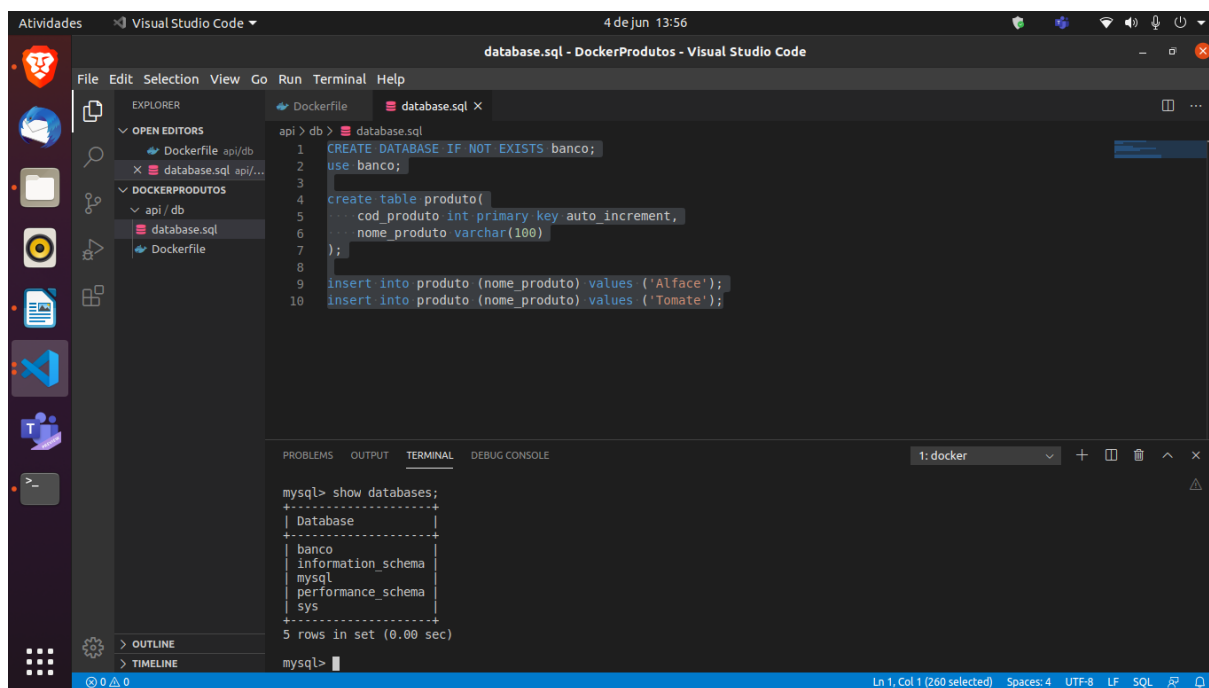
Vamos ver se realmente importou com sucesso nossas informações. Entrando no terminal do container mysql com o mesmo comando que executamos acima, escreva “mysql -u root -p123”, será aberto o banco de dados, com isso digitaremos “show databases;” se o banco de dados que nós criamos estiver listado, vamos digitar “use banco” e após isso vamos digitar “select \* from produto;”, deverá ser listado os produtos que inserimos no arquivo .sql.



The screenshot shows the Visual Studio Code interface with the Dockerfile open in the editor. The Dockerfile contains the following SQL commands:

```
1 CREATE DATABASE IF NOT EXISTS banco;
2 use banco;
3
4 create table produto(
5     cod_produto int primary key auto_increment,
6     nome_produto varchar(100)
7 );
8
9 insert into produto (nome_produto) values ('Alface');
10 insert into produto (nome_produto) values ('Tomate');
```

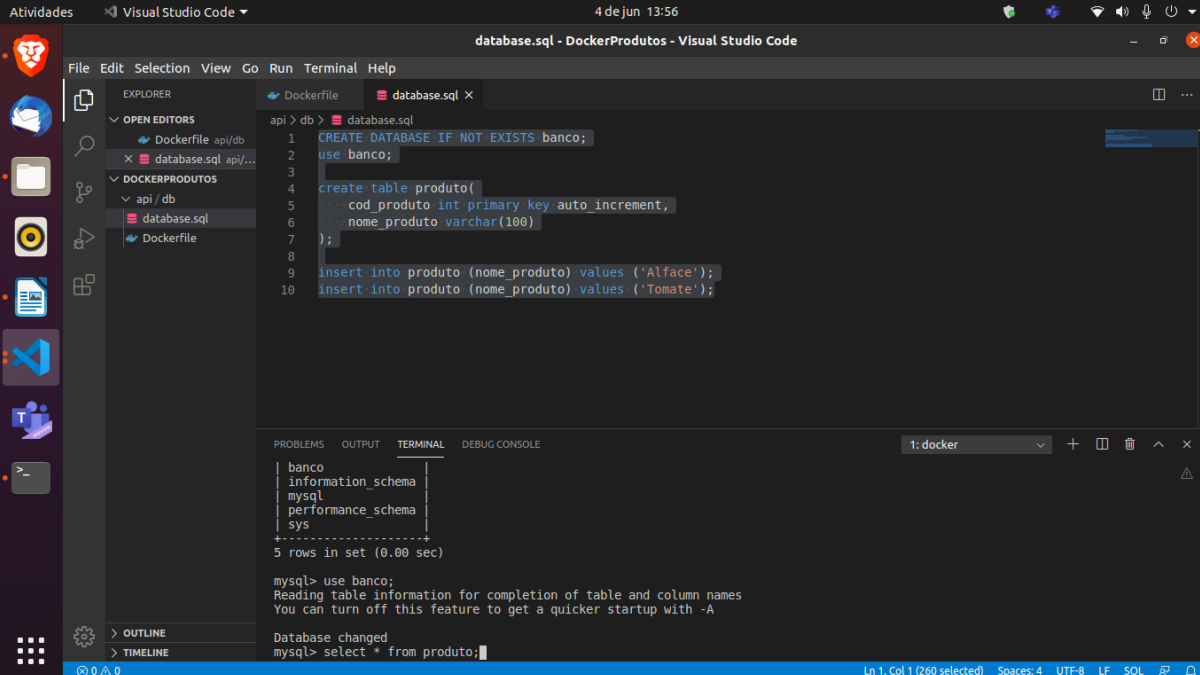
The terminal output shows the command `docker exec -it mysql-container /bin/bash` being executed, resulting in a MySQL prompt. The user enters `mysql -u root -p123`, and the terminal displays the MySQL version and server information.



The screenshot shows the Visual Studio Code interface with the Dockerfile open in the editor. The terminal output shows the command `mysql -u root -p123` being executed, resulting in a MySQL prompt. The user enters `show databases;`, and the terminal displays the list of databases:

```
mysql> show databases;
+-----+
| Database |
+-----+
| banco    |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.00 sec)
```

The user then enters `use banco;` and the terminal displays the MySQL prompt.



Visual Studio Code interface showing the Dockerfile and database.sql files. The terminal output shows the MySQL container startup:

```

1 CREATE DATABASE IF NOT EXISTS banco;
2 use banco;
3
4 create table produto(
5     cod_produto int primary key auto_increment,
6     nome_produto varchar(100)
7 );
8
9 insert into produto (nome_produto) values ('Alface');
10 insert into produto (nome_produto) values ('Tomate');

```

Terminal output:

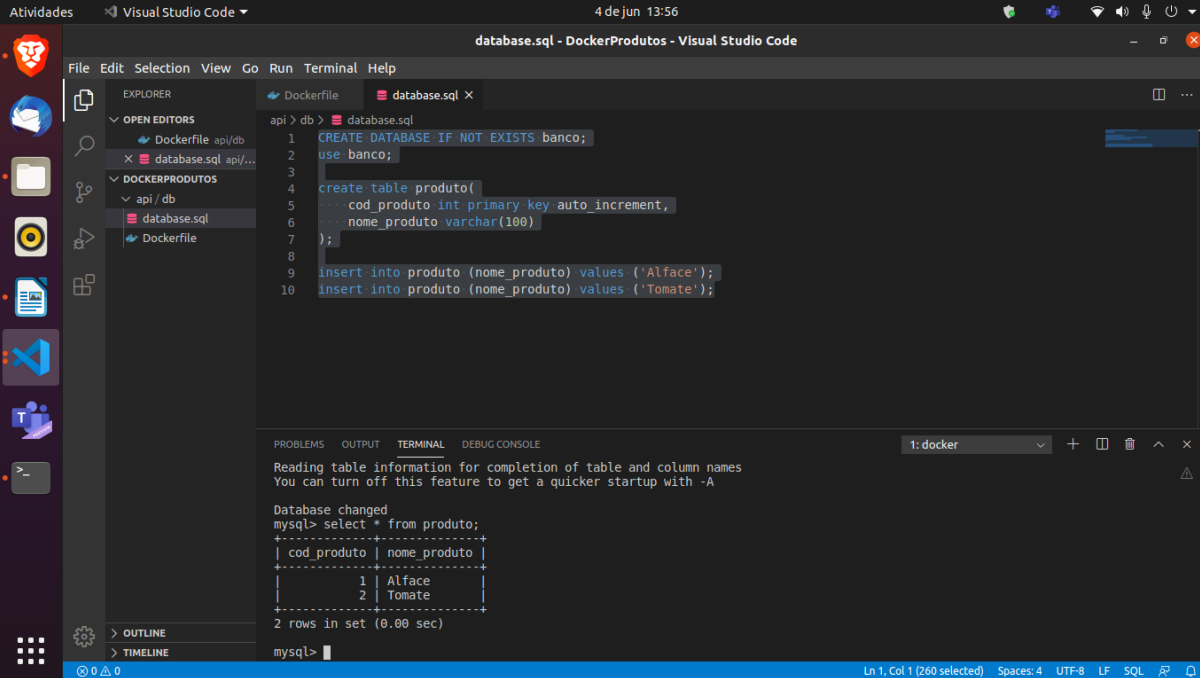
```

banco
information_schema
mysql
performance_schema
sys
+-----+
5 rows in set (0.00 sec)

mysql> use banco;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from produto;

```



Visual Studio Code interface showing the Dockerfile and database.sql files. The terminal output shows the MySQL container startup:

```

1 CREATE DATABASE IF NOT EXISTS banco;
2 use banco;
3
4 create table produto(
5     cod_produto int primary key auto_increment,
6     nome_produto varchar(100)
7 );
8
9 insert into produto (nome_produto) values ('Alface');
10 insert into produto (nome_produto) values ('Tomate');

```

Terminal output:

```

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from produto;
+-----+
| cod_produto | nome_produto |
+-----+
| 1 | Alface |
| 2 | Tomate |
+-----+
2 rows in set (0.00 sec)

mysql>

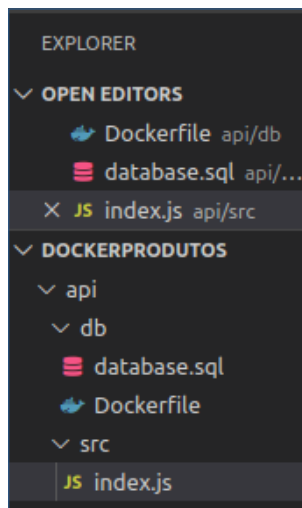
```

Pronto, nosso banco de dados está configurado e pronto para o uso. Para sair do MySQL, digite “exit” você voltará para o terminal do mysql-container, digite novamente “exit” para voltar para o terminal do Visual Studio Code.

## 1.5 APLICAÇÃO NODE

Nesta parte do relatório nós iremos criar a aplicação node que ficará responsável por ler o banco de dados e disponibilizar um JSON com os produtos, então vamos continuar...

Dentro da pasta “api”, crie a pasta “src” e dentro desta pasta, crie o arquivo “index.js”, a estrutura irá ficar igual ao da imagem abaixo.



Neste arquivo coloque o seguinte conteúdo:

```
const express = require('express');
const mysql = require('mysql');

const app = express();

const connection = mysql.createConnection({
  host: 'mysql-container',
  user: 'root',
  password: '123',
  database: 'banco'
});

connection.connect();

app.get('/produtos', function(req, res){
  connection.query('Select p.cod_produto, p.nome_produto from produto p', function (error, results){
    if (error){
      throw error
    }

    res.send(results.map(item => ({cod_produto : item.cod_produto, nome_produto:
    item.nome_produto})));
  });
});

app.get('/', function(req, res){
  res.send('oi');
});

app.listen(9001, '0.0.0.0', function(){
  console.log('Ouvindo na 9001-');
});
```

Atráves do terminal do Visual Studio Code navegue até a pasta “api” e digite o seguinte comando e configure as coisas que serão pedidas.

“npm init”.

Ainda nesta pasta (api), digite no terminal do Visual Studio Code os seguintes comandos:

“npm install --save-dev nodemon”

“npm install --save express mysql”

Após a execução destes comandos, note que no seu diretório foi criado a pasta “node\_modules” e os arquivos “package-lock.json” e “package.json”. Agora com o arquivo index.js criado e configurado, vamos criar o arquivo “Dockerfile” dentro da pasta “api” e vamos colocar dentro dele o seguinte conteúdo:

```
FROM node:alpine
WORKDIR /usr/app
COPY package.json /usr/app
COPY package-lock.json /usr/app
RUN npm install
# Bundle app source
COPY .
EXPOSE 9001
CMD npm start
```

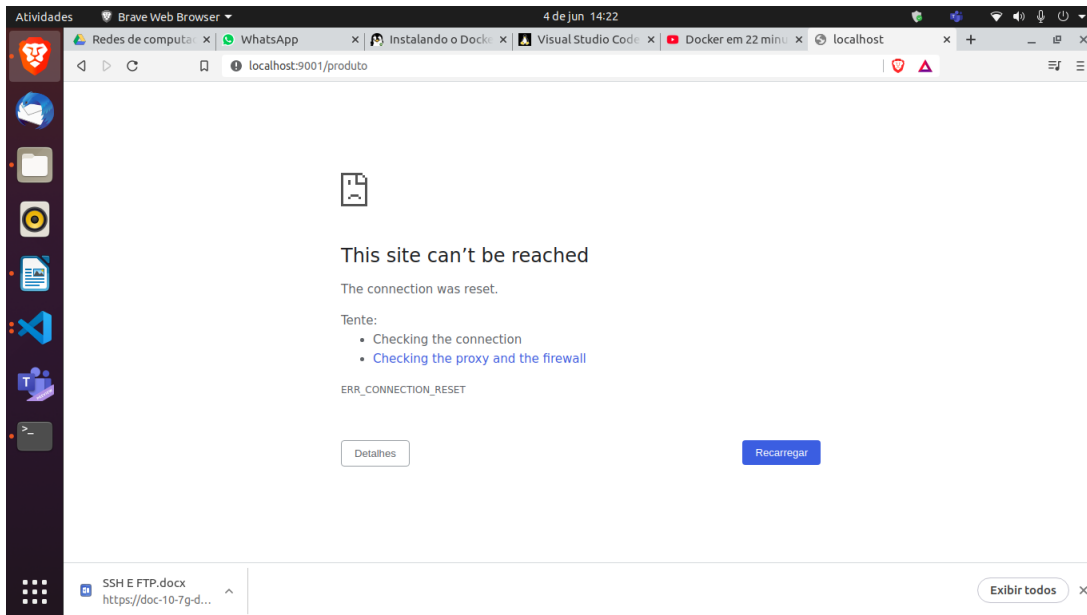
Estamos puxando a imagem node:alpine, falando que vamos usar o diretório usr/app copiando os arquivos packages\* para dentro desta pasta e mandando o container executar o npm install para instalar as dependências que salvamos mais a cima, estaremos copiando a pasta src e expondo a porta 9001 do container.

Vamos dar o comando “docker build -t node-image .” dentro da pasta “api” e em seguida vamos subir nosso container com o seguinte comando:

```
docker run -d -v ${pwd}:/home/node/app -p 9001:9001 --link mysql-container
--rm --name node-container node-image
```

O “-p” significa que vamos usar a nossa porta 9001 para comunicar com a 9001 do container (NOSSA\_PORTA : CONTAINER\_PORTA) e o “--link” serve para o próprio node-container se comunicar automaticamente com o mysql-container sem precisarmos passar o IP do container. Este “--link” tem utilidade dentro do nosso arquivo “index.js” na hora de passar o host (Passamos o nome do container ao invés do IP).

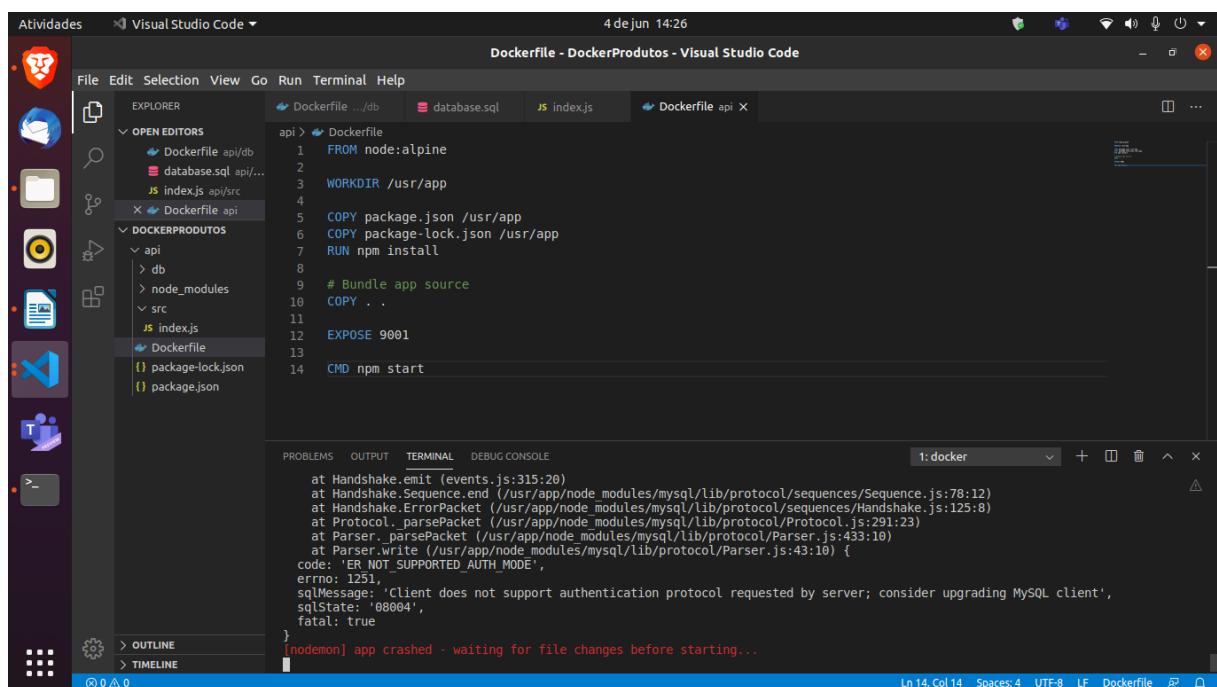
Ao abrirmos nosso navegador e na URL e digitarmos “localhost:9000/produtos”, será retornado os produtos do banco de dados.



Caso uma tela igual a esta seja mostrada ao invés das informações do banco pause o container executando o seguinte comando “`docker stop node-container`” e execute o comando de subir o node-container sem o -d, isso fará com que o terminal fique travado nessa operação exibindo os passos que o container está dando.

`docker run -v ${pwd}:/home/node/app -p 9001:9001 --link mysql-container --rm --name node-container node-image`

Se caso você executar este comando e o erro for igual ao da imagem, siga os passos a seguir.



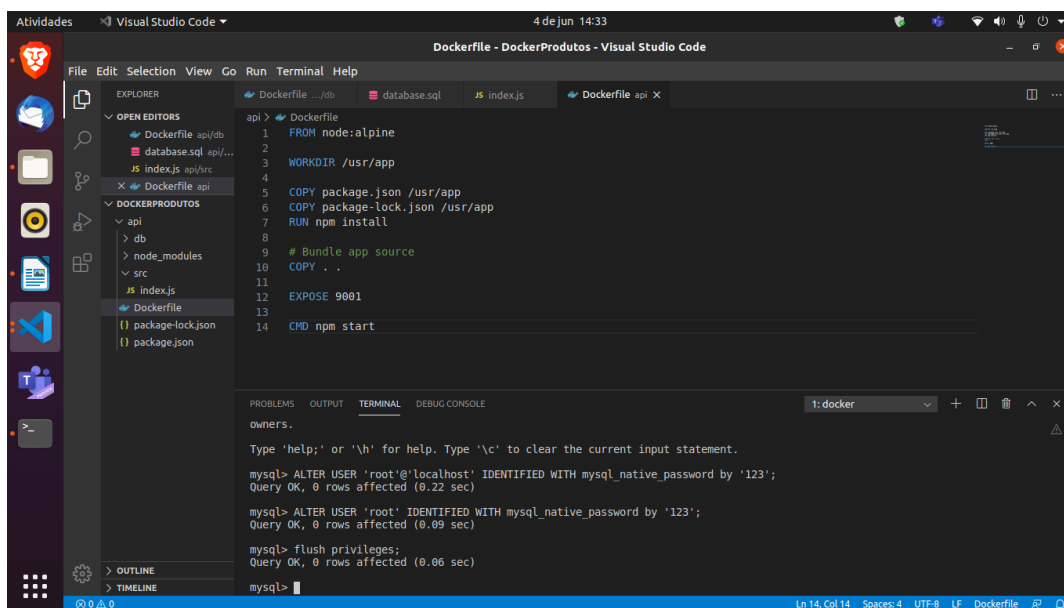
### 3. Corrigindo erro node-container.

Abra o terminal do mysql-container com o comando “docker exec -it mysql-container /bin/bash” e entre dentro do mysql digitando “mysql -u root -p1234”. Dentro deste terminal, digite os comandos abaixo:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '123';
```

```
ALTER USER 'root' IDENTIFIED WITH mysql_native_password BY '123';
```

```
Flush privileges;
```



```

Dockerfile
1 FROM node:alpine
2
3 WORKDIR /usr/app
4
5 COPY package.json /usr/app
6 COPY package-lock.json /usr/app
7 RUN npm install
8
9 # Bundle app source
10 COPY . .
11
12 EXPOSE 9001
13
14 CMD npm start
  
```

```

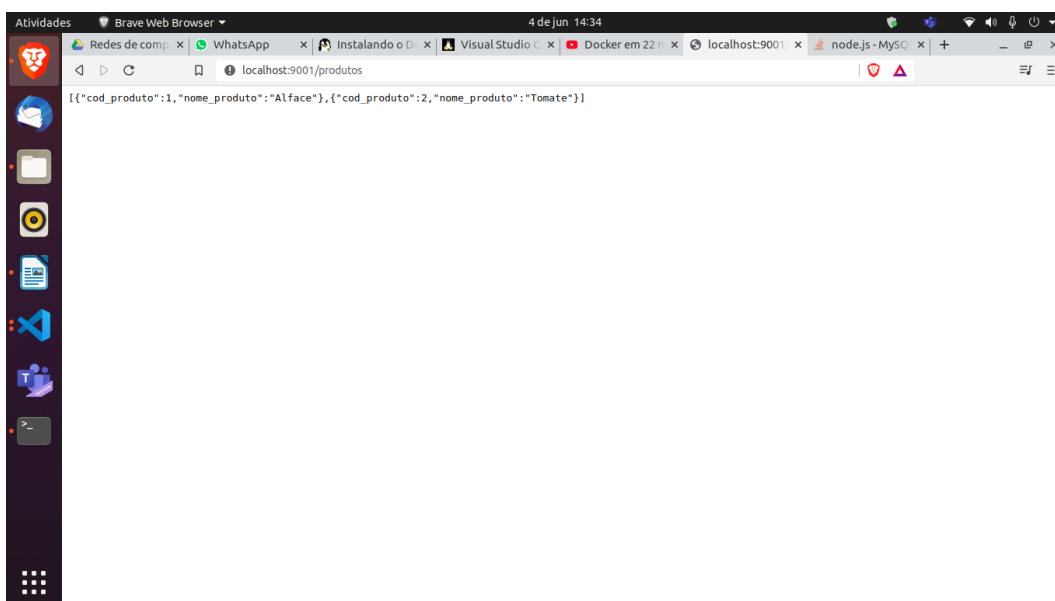
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by '123';
Query OK, 0 rows affected (0.22 sec)

mysql> ALTER USER 'root' IDENTIFIED WITH mysql_native_password by '123';
Query OK, 0 rows affected (0.09 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.06 sec)

mysql>
  
```

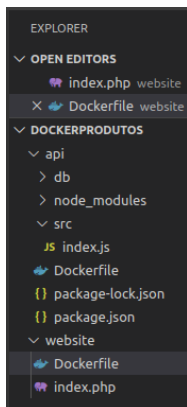
Depois de ter executado estes dois comandos, reinicie o seu container do node e tente novamente conectar no “localhost:9001”, será mostrado uma tela igual a esta.



## 1.6 APLICAÇÃO PHP

Por último, vamos criar a nossa aplicação PHP que irá ler o JSON fornecido pelo node-container e exibirá em uma tabela.

Crie a pasta “website” na raiz pro projeto. Dentro desta pasta crie o arquivo “index.php” e o arquivo “Dockerfile” como na imagem abaixo.



No arquivo “Dockerfile” coloque o seguinte conteúdo:

```
FROM php:7.2-apache  
COPY ./var/www/html/
```



No arquivo “index.php” coloque o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
<title>PHP</title>
</head>
<body>
<?php
$result = file_get_contents("http://node-container:9001/produtos");
$produtos = json_decode($result);
?>

<table class="table">
<thead>
<tr>
<th>Codigo</th>
<th>Nome</th>
</tr>
</thead>
<tbody>
<?php
foreach($produtos as $linha){?>
<tr>
<td><?php echo $linha->cod_produto; ?></td>
<td><?php echo $linha->nome_produto; ?></td>
</tr>
<?php }
?>
</tbody>
</table>
</body>
</html>
```

Navegue até a pasta “website” pelo terminal do Visual Studio Code e de um build no Dockerfile com o comando “docker build -t php-image .”

The screenshot shows the Visual Studio Code interface with the 'Dockerfile - DockerProdutos' file open. The Dockerfile contains the following content:

```

1 FROM php:7.2-apache
2 COPY . /var/www/html/

```

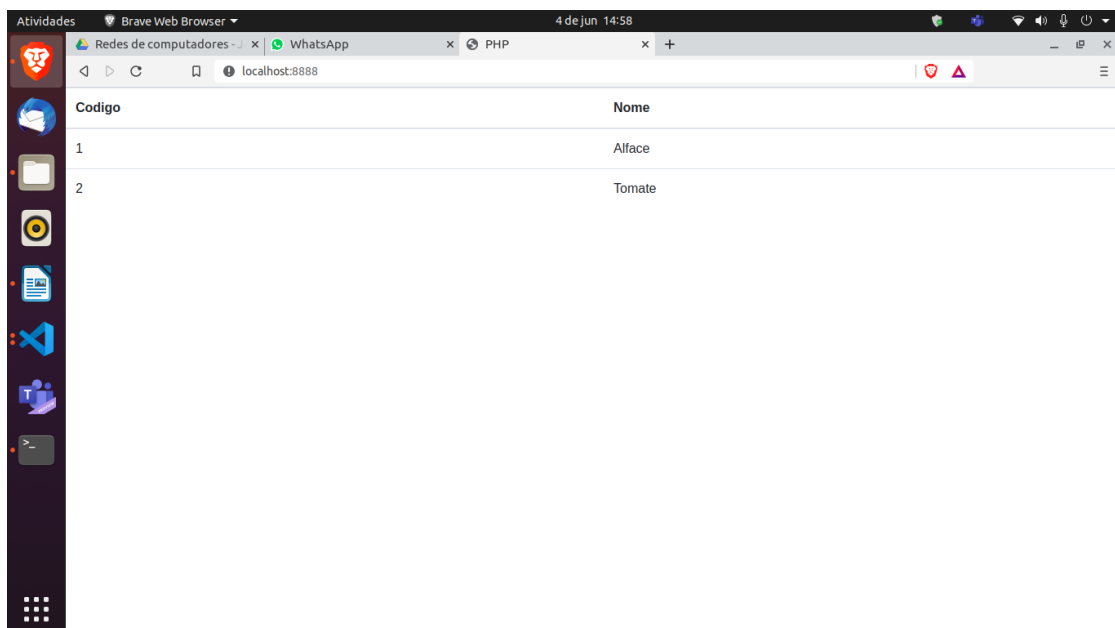
The terminal output shows the execution of the 'docker build' command:

```

paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos$ cd website/
paulo@paulo-Aspire-E1-571:~/Area de Trabalho/DockerProdutos/website$ docker build -t php-image .
Sending build context to Docker daemon  2.56kB
Step 1/2 : FROM php:7.2-apache
7.2-apache: Pulling from library/php
afb6ec6fdc1c: Already exists
3d895574014b: Pull complete
c389f0ad041b: Downloading [==>]  1.596MB/76.65MB
c201f6a5d6f9: Download complete
e87f853892aa: Downloading [==>]  1.35MB/18.68MB
998b2113b400: Download complete
b3c0b4710d3b: Waiting
4e342cb20e11: Waiting
c591e1fe994d: Waiting

```

Depois disso, suba o container PHP com o comando “`docker run -d -v ${pwd}:/var/www/html -p 8888:80 --link node-container --rm --name php-container php-image`”. Vá até o navegador e digite “localhost:8888” será exibido a tela abaixo mostrando os produtos trazidos do node.



## 2 CONCLUSÕES

Utilizando este tutorial, você conseguiu criar 3 containers que conversam entre si, um para banco de dados utilizando o MySQL, outro para disponibilizar estes dados de forma JSON e o último para se alimentar dos arquivos JSON mostrando em forma de tabela.

## REFERÊNCIAS

Programador a bordo - Docker em 22 Minutos – teoria e prática (Rápido!)– 25/03/2019 – < <https://www.youtube.com/watch?v=Kzcz-EVKBEQ>> – Acesso em: 27/05/2020.

Brito – Edivaldo – Como instalar o Visual Studio Code no Ubuntu – 12/12/2019 – <<https://www.edivaldobrito.com.br/visual-studio-code-no-ubuntu/>> – Acesso em: 04/06/2020.

MySQL 8.0 – Client does not support authentication protocol requested by server; consider upgrading MySQL client – 05/2018 – < <https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server>> – Acesso em: 29/05/2020.

Santana – Luriel – Instalando o Docker-CE no Ubuntu 18.04 LTS – 20/03/2020 – < <https://www.vivaolinux.com.br/dica/Instalando-o-Docker-CE-no-Ubuntu-1804-LTS>> – Acesso em: 04/06/2020.