

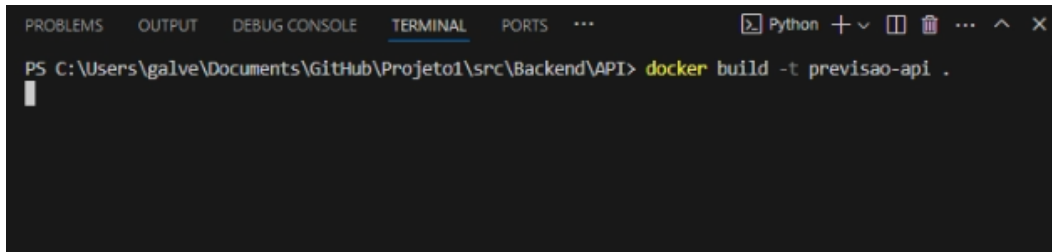
Relatório de Implantação do Modelo Preditivo com API Flask e Docker

Guilherme Alves de Oliveira e Oliveira
João Pedro Lima Paulo
Lara Marina de Oliveira
Giulia Nogueira Lopes De Sá
Beatriz de Castilho Ferreira

Criação da API REST

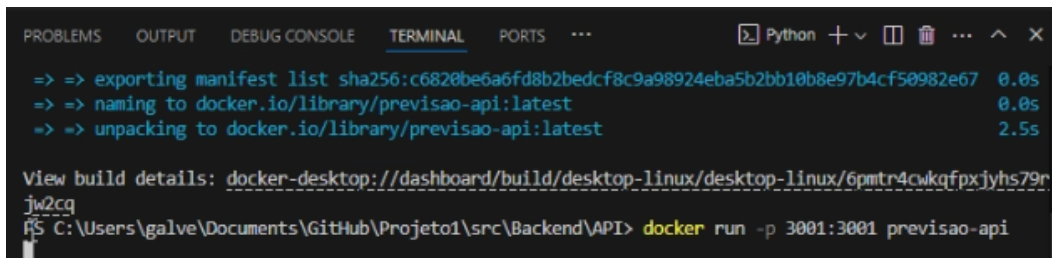
Foi implementada uma API utilizando o framework Flask, responsável por consumir um modelo preditivo de preços de corridas. A API possui o endpoint /predict, que recebe dados de origem e destino via JSON, consulta a Google Maps API para obter distância e duração, processa os dados conforme o modelo original treinado no Colab e retorna o valor previsto da corrida para diferentes categorias.

Aqui estamos gerando uma imagem no Docker localmente:



```
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API> docker build -t previsao-api .
```

Estamos aqui rodando localmente a aplicação via Docker:



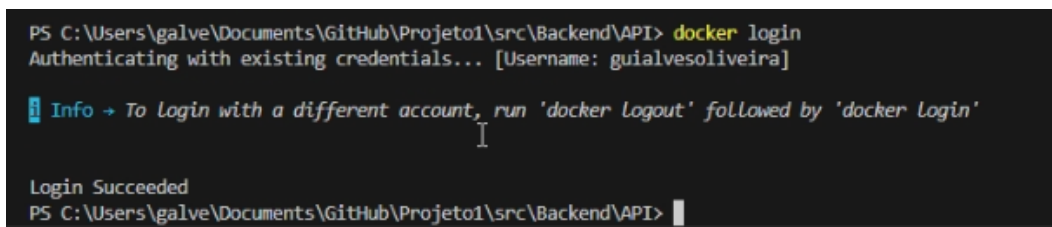
```
=> => exporting manifest list sha256:c6820be6a6fd8b2bedcf8c9a98924eba5b2bb10b8e97b4cf50982e67 0.0s
=> => naming to docker.io/library/previsao-api:latest 0.0s
=> => unpacking to docker.io/library/previsao-api:latest 2.5s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/6pmt4cwkqfpxjyhs79rjw2cq
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API> docker run -p 3001:3001 previsao-api
```

Deploy na Nuvem (AWS EC2)

Foi utilizada uma instância EC2 na AWS (Amazon Web Services) como provedor de nuvem. O Docker foi instalado na instância e configurado para executar a aplicação.

Localmente fazemos o login:

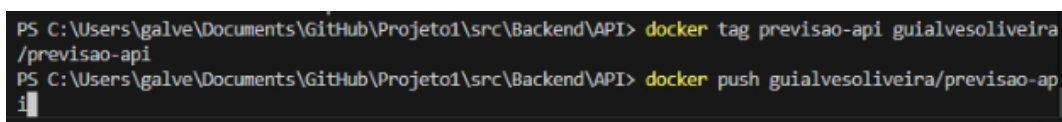


```
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API> docker login
Authenticating with existing credentials... [Username: guialvesoliveira]

Info -> To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API>
```

Alteramos o nome da aplicação para que tenha o caminho da nossa conta do docker hub, e fazemos o push:



```
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API> docker tag previsao-api guialvesoliveira/previsao-api
PS C:\Users\galve\Documents\GitHub\Projeto1\src\Backend\API> docker push guialvesoliveira/previsao-api
```

Dentro da EC2 da AWS, fazemos o pull da aplicação:

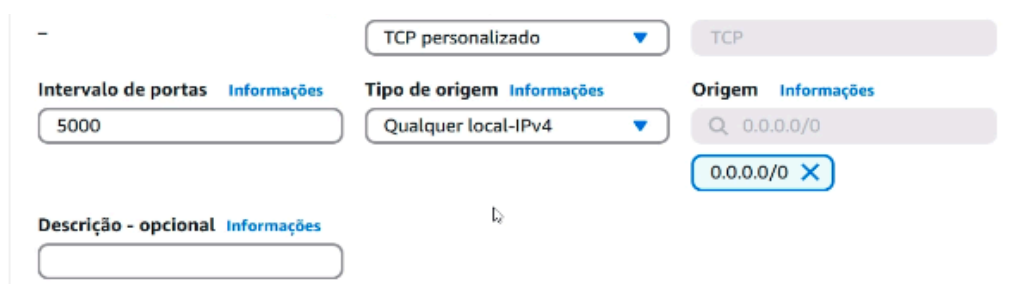
```
[ec2-user@ip-172-31-36-98 ~]$ sudo docker pull guialvesoliveira/previsao-api
Using default tag: latest
latest: Pulling from guialvesoliveira/previsao-api
254e724d7786: Already exists
5e3380732964: Already exists
5cc9686f2aa9: Already exists
5d66a27e733a: Already exists
160899883894: Already exists
c6269518ecb8: Pull complete
b8e194e15f28: Pull complete
e2b44dfbd932: Pull complete
Digest: sha256:c6820be6a6fd8b2bedcf8c9a98924eba5b2bb10b8e97b4cf50982e676751598d
Status: Downloaded newer image for guialvesoliveira/previsao-api:latest
docker.io/guialvesoliveira/previsao-api:latest
[ec2-user@ip-172-31-36-98 ~]$
```

E por fim rodamos a aplicação:

```
5000 failed: port is already allocated.
[ec2-user@ip-172-31-36-98 ~]$ sudo docker run -p 3001:3001 guialvesoliveira/previsao-api
* Serving Flask app 'ApiModel'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production W
SGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.3:5000
Press CTRL+C to quit
* Restarting with stat
```

Mas temos um porém.

Para podermos acessar essa aplicação rodando, dentro da AWS devemos criar um IP elástico, e liberar as portas que iremos utilizar na aplicação. Nesse caso iremos utilizar as portas 5000 tanto de entrada e saída nos *Security Groups*.



O IP elástico atribuído à nossa EC2 foi o 15.229.102.105, como nossa porta de abertura foi a 5000, a API deve ser acessada por 15.229.102.105:5000.

A utilização da API deve ser feita via Json, utilizando a rota [/predict](#) como um *post*:

```
{
  "origin": ""
  "destination": ""
}
```

Faça o teste com o [Postman](#)

Containerização com Docker

Foi criado um Dockerfile contendo todos os passos necessários para construção da imagem da API, incluindo a instalação de dependências via requirements.txt, cópia dos arquivos da aplicação, definição da porta 5000 e inicialização do Flask.

Dockerfile:

```
FROM python:3.10-slim
```

```
WORKDIR /app
```

```
COPY . /app
```

```
RUN pip install --upgrade pip
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 5000
```

```
CMD ["python", "ApiModel.py"]
```