VisionHive

DEVOPS TOOLS & CLOUD COMPUTING

João Victor Michaeli - 555678
Larissa Muniz - 557197
Henrique Garcia - 558062

Links

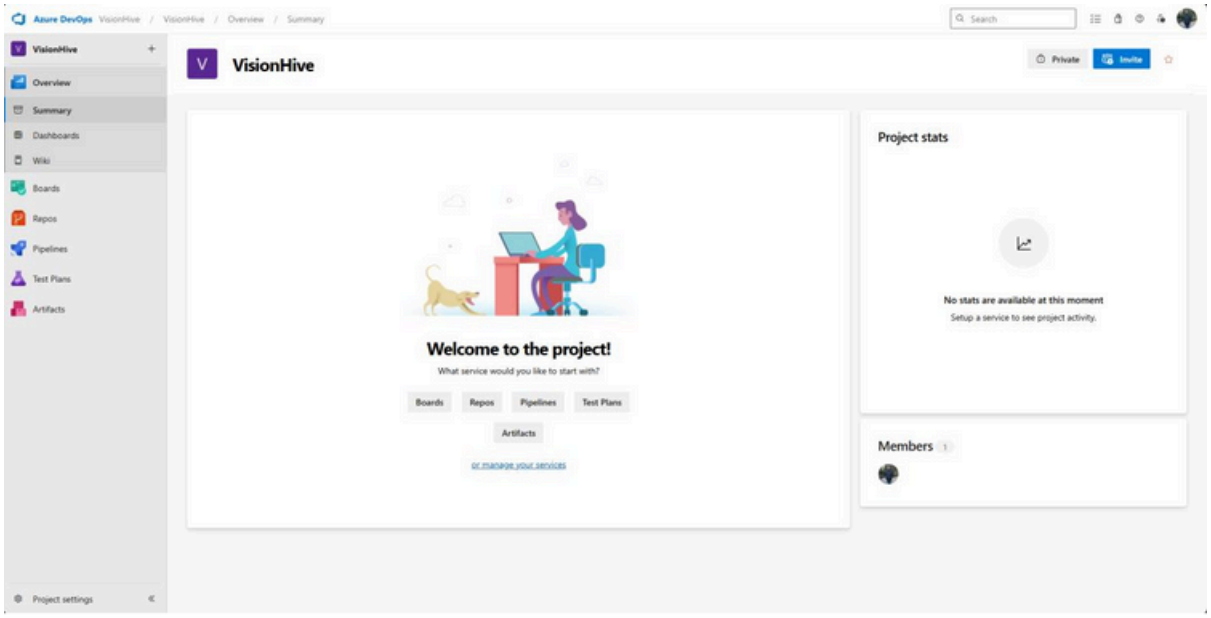
Link para o vídeo da criação da Vm

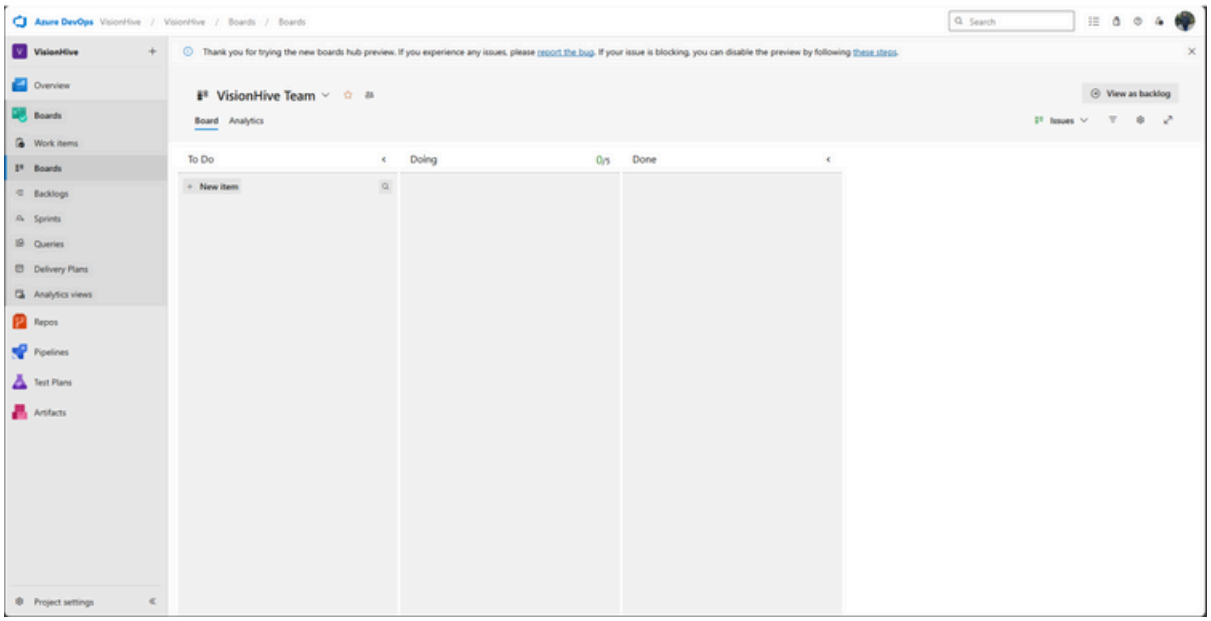
https://youtu.be/qW7178f7yGY

Link do repositório github

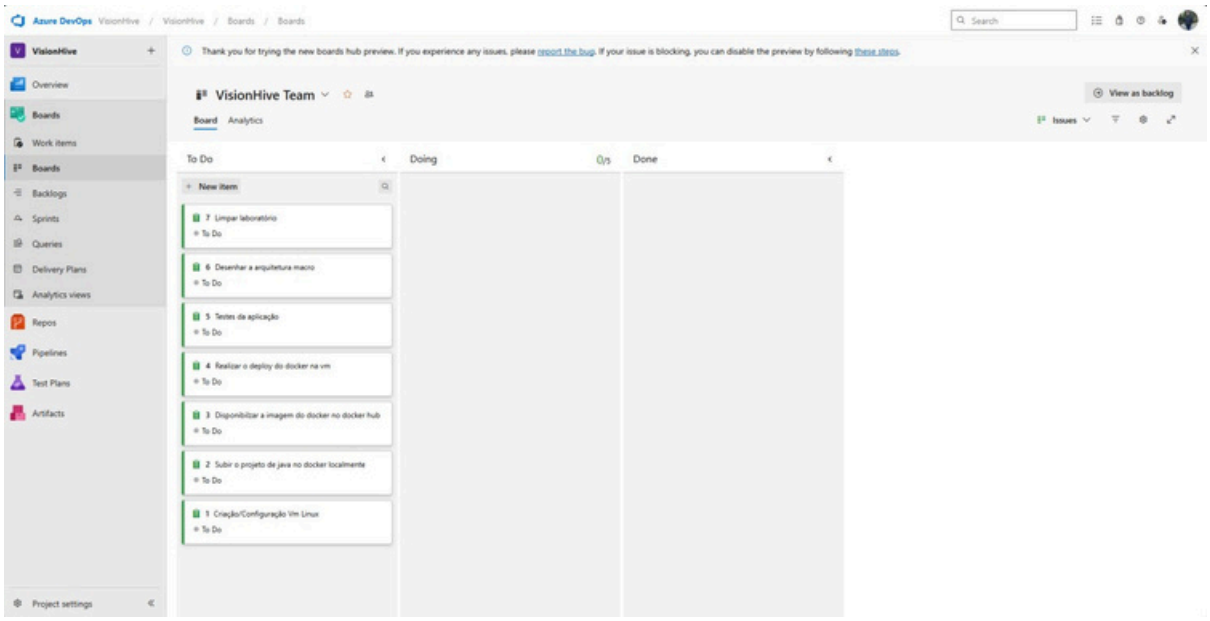
https://github.com/JoaoMichaeli/VisionHive-DevOps

Criação do Azure board para melhor organização

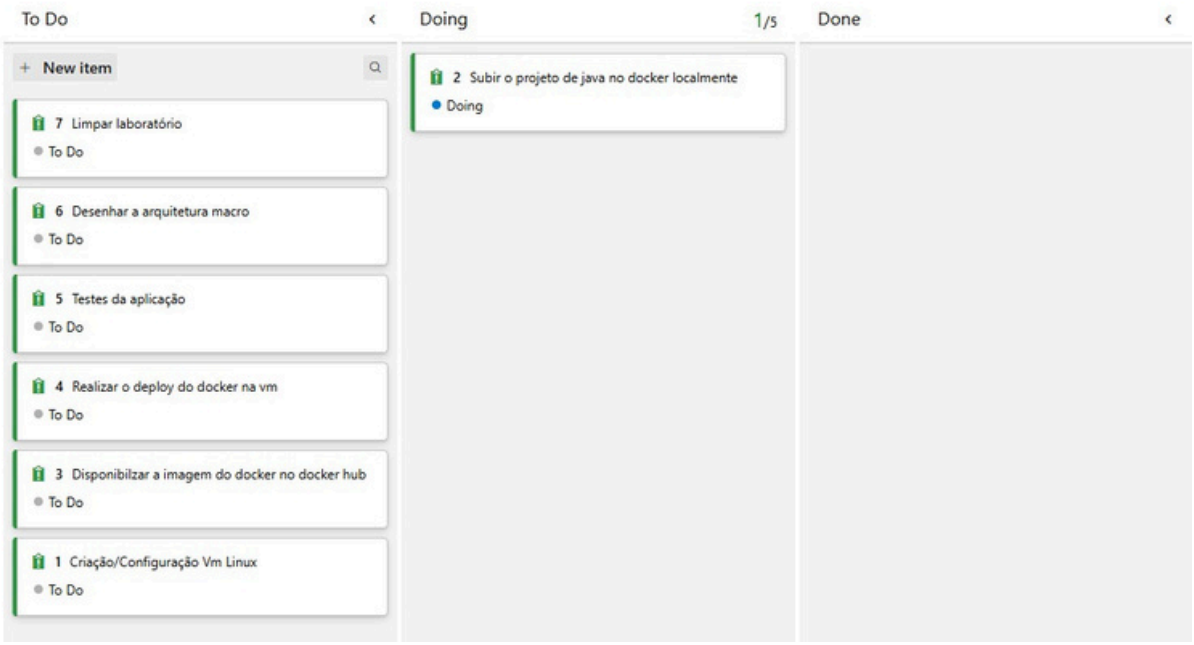




Preenchimento da lista To Do



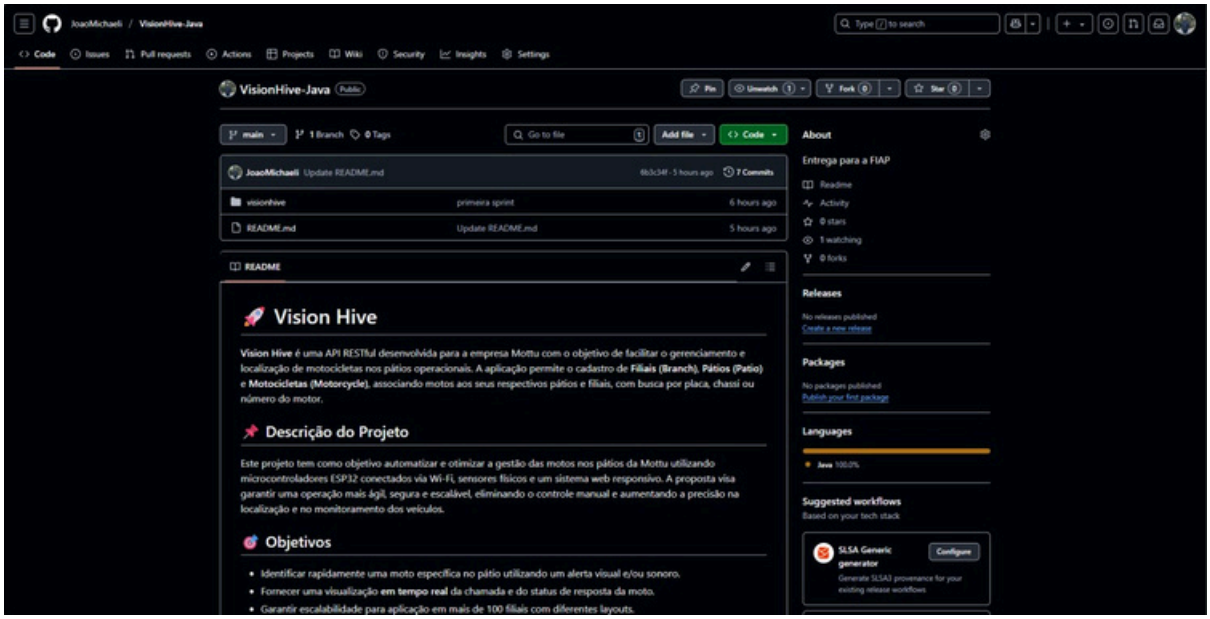
Primeiro vamos subir o java no docker local, para poseriormente subir no docker hub



Criar o Dockerfile

O projeto java está nesse repositório do github:

https://github.com/JoaoMichaeli/VisionHive-Java



Dockerfile do projeto

```dockerfile
FROM maven:3.8.5-openjdk-17 AS build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests


FROM openjdk:17-alpine
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Criar e testar a imagem Docker localmente

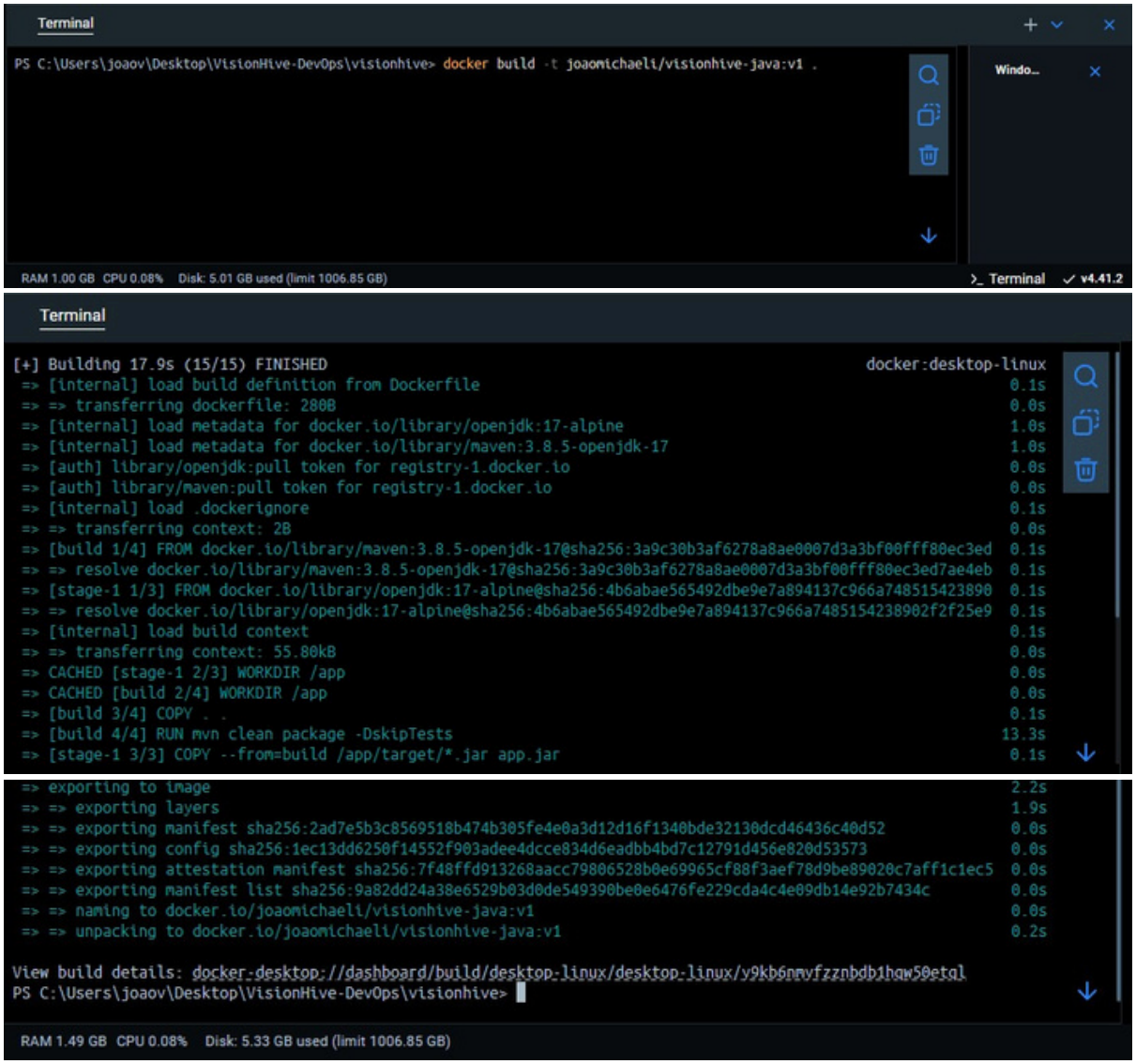
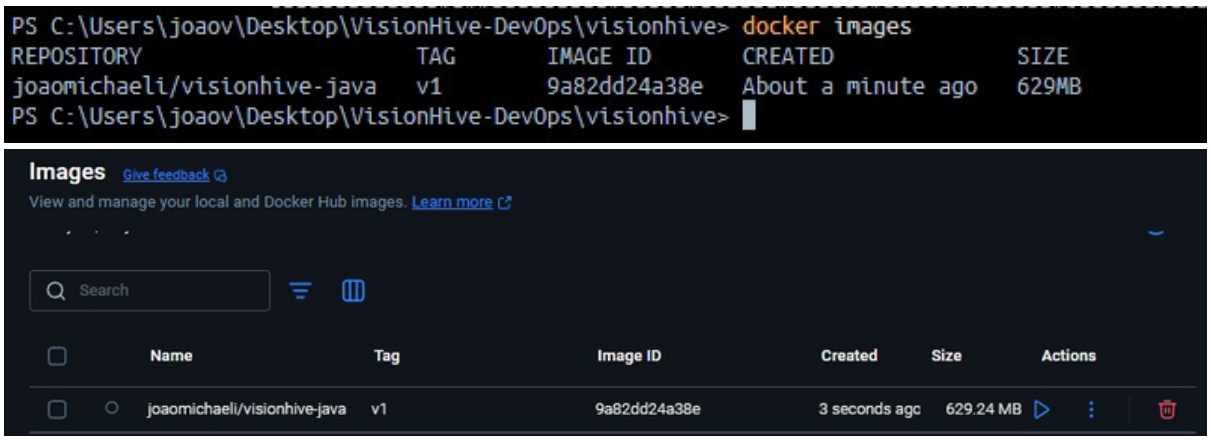
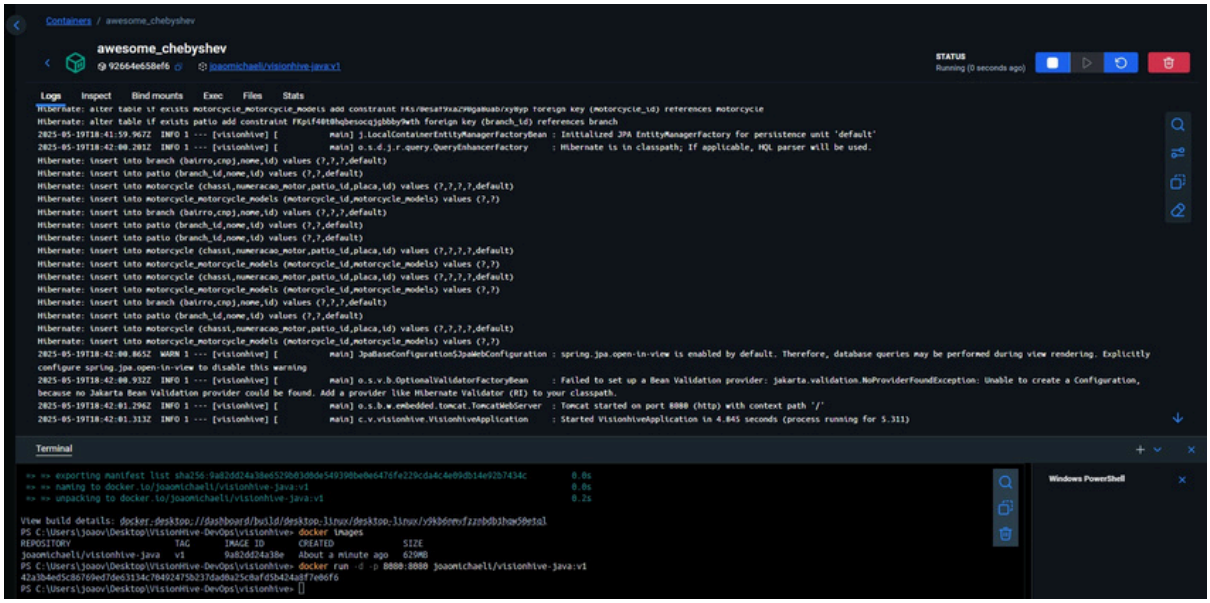


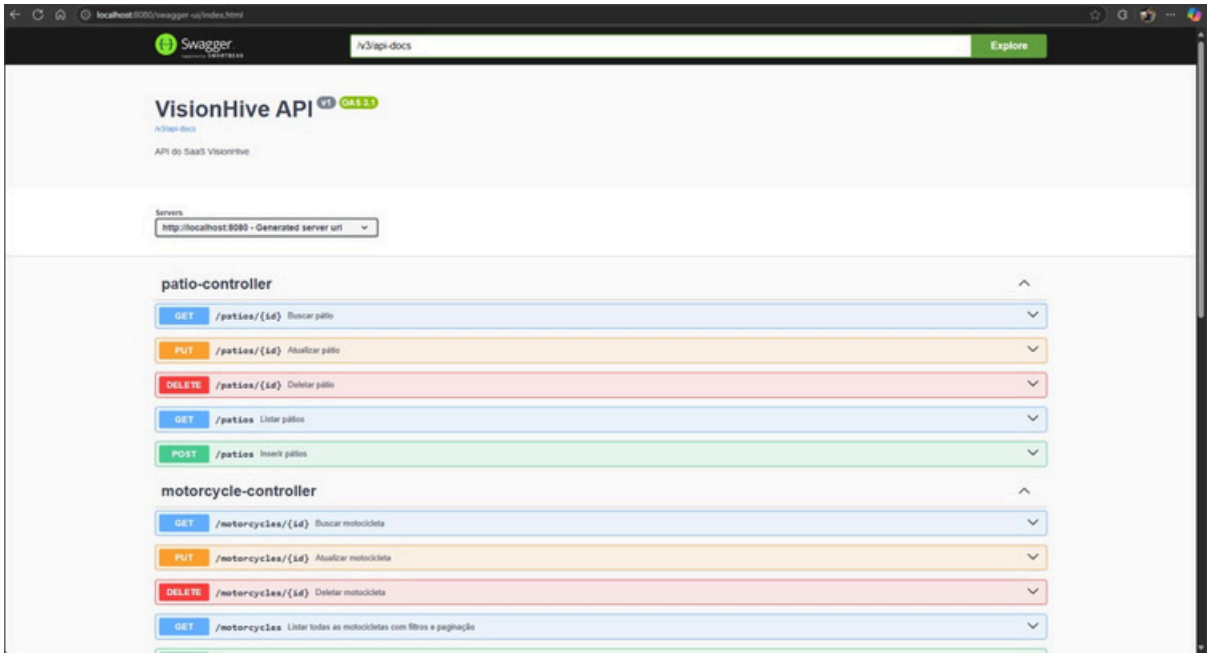
Imagem criada



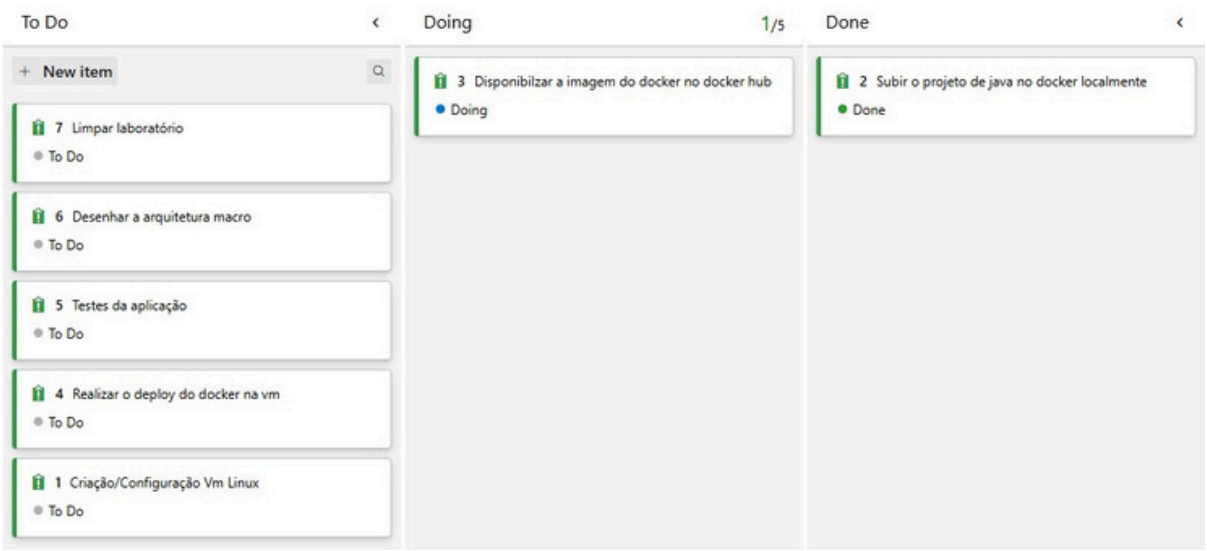
Como podemos ver, o projeto ja está funcionando localmente



Como essa é a primeira entrega da solução em java, ainda não temos integração web gráfica, então para testarmos o deploy será utilizado via swagger



Após o sucesso do teste, vou subir a imagem no docker hub para o envio posterior a vm.



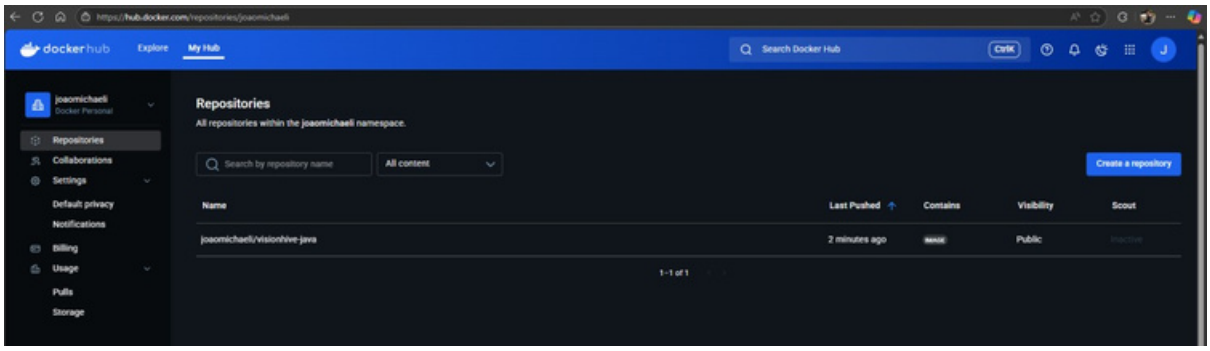
Agora vamos realizar a disponibilidade da imagem local para o hub

```
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker login
Authenticating with existing credentials... [Username: joaomichaeli]

i Info →To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker push joaomichaeli/visionhive-java:v1
The push refers to repository [docker.io/joaomichaeli/visionhive-java]
c3b66361bb95: Pushed
7992fe636f76: Pushed
d8d715783b80: Pushed
53c9466125e4: Pushed
5843afab3874: Pushed
d902d246dc1f: Pushed
v1: digest: sha256:9a82dd24a38e6529b03d0de549390be0e6476fe229cda4c4e09db14e92b7434c size: 856
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive>
```

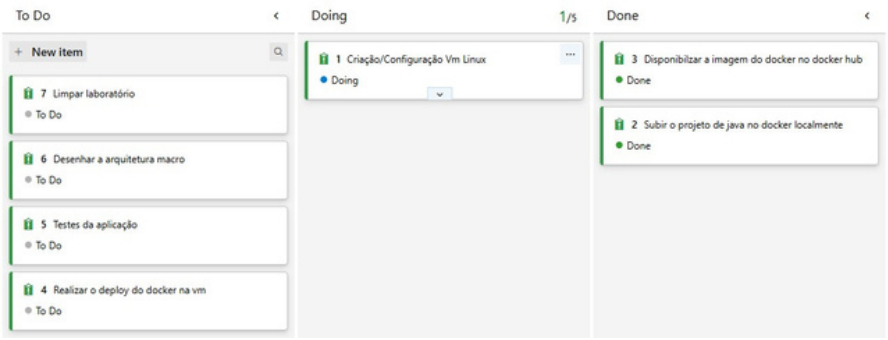
Imagem postada no hub

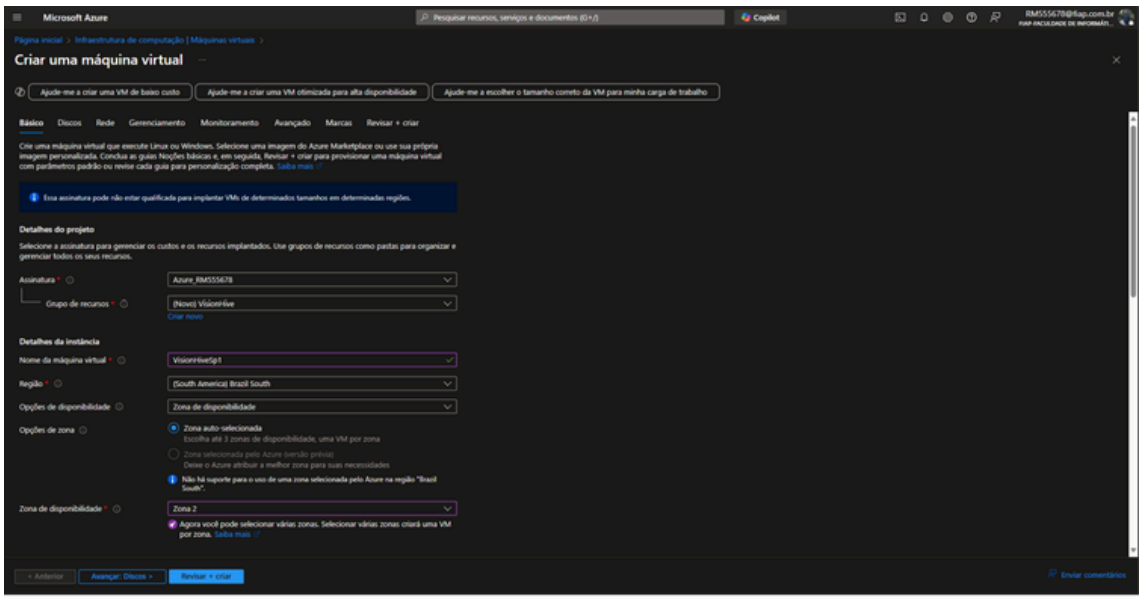


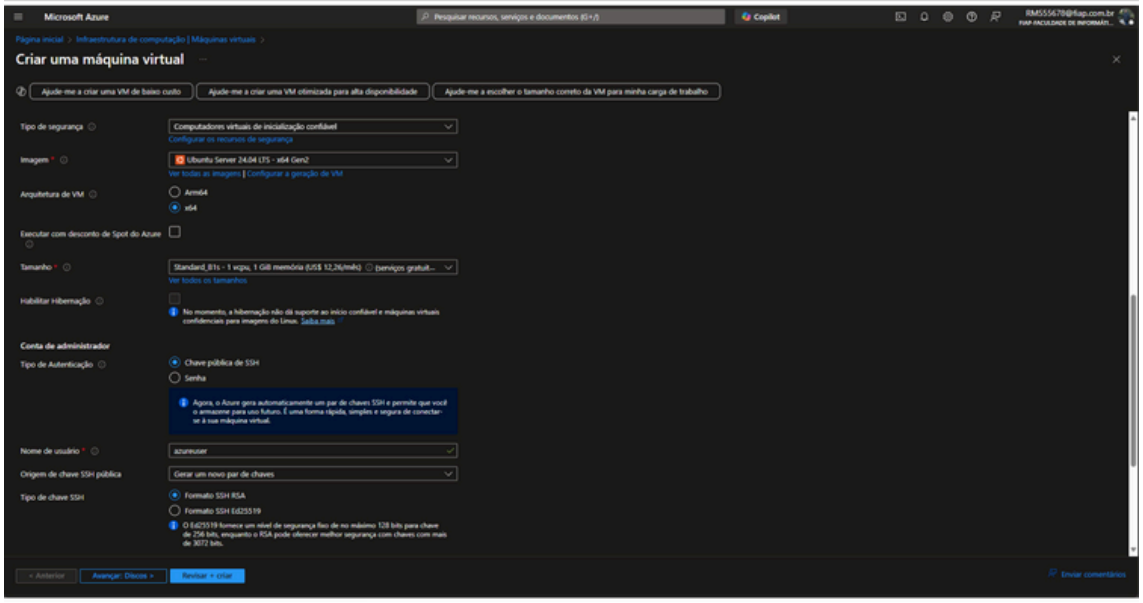
Com a imagem postada no docker hub, agora vamos criar e configurar a máquina virtual

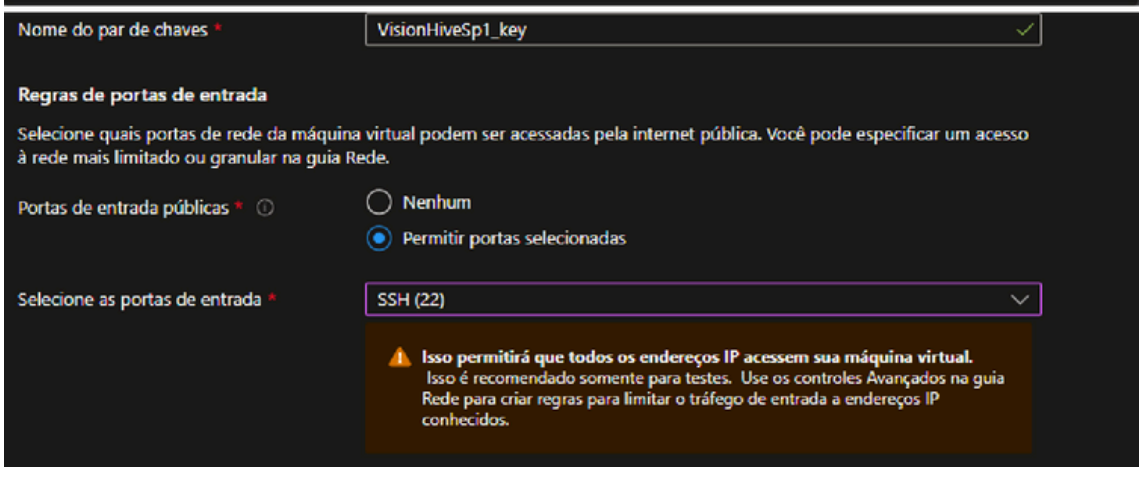
https://youtu.be/qW7178f7yGY

Criar a maquina virtual



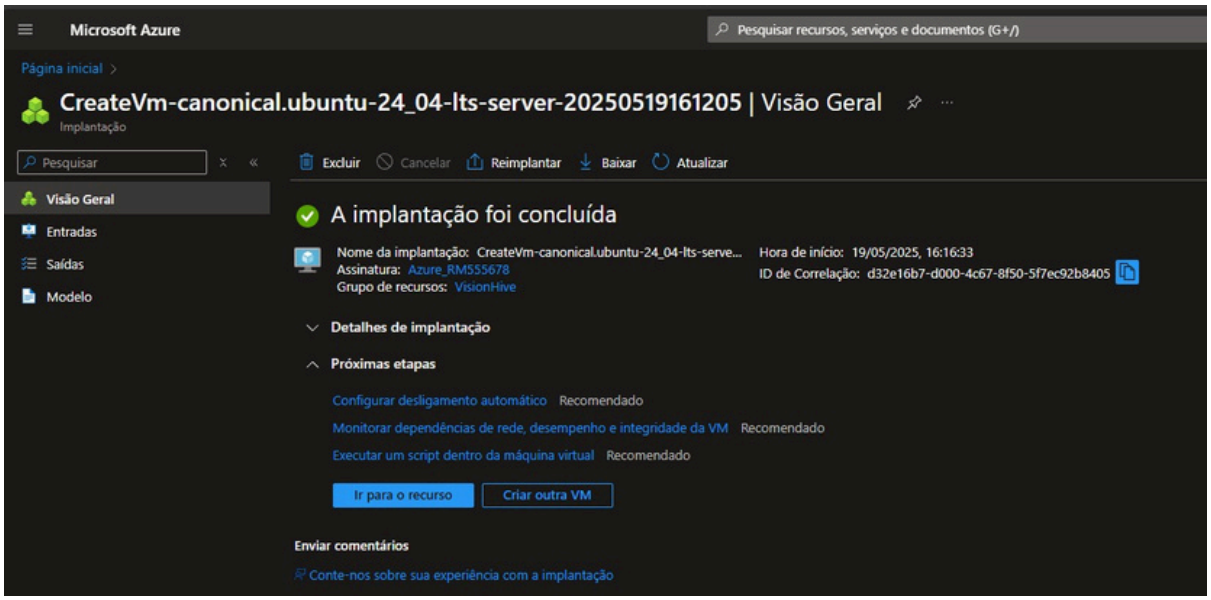




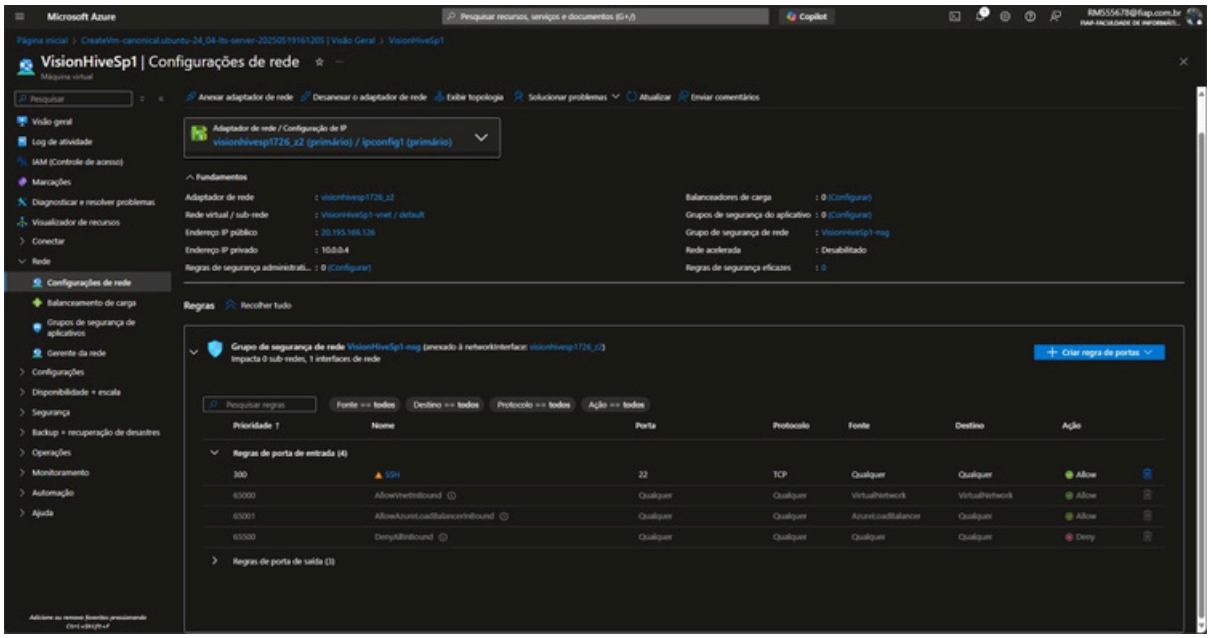


Precificação

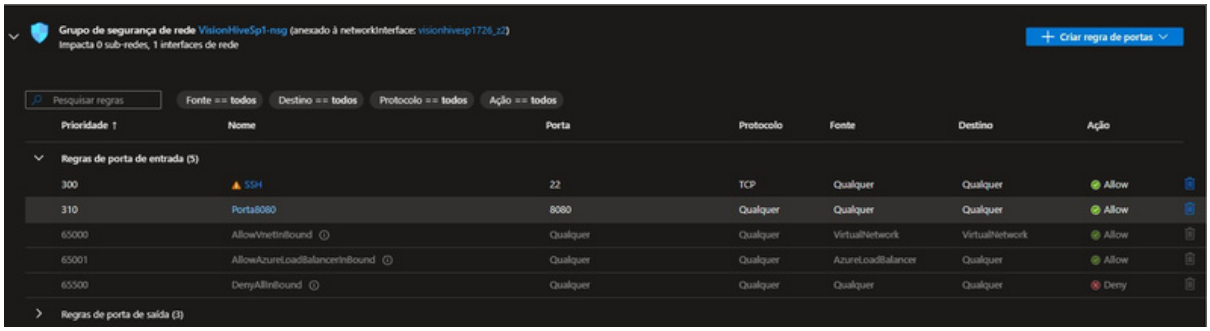
Implementação concluida



Criação das portas

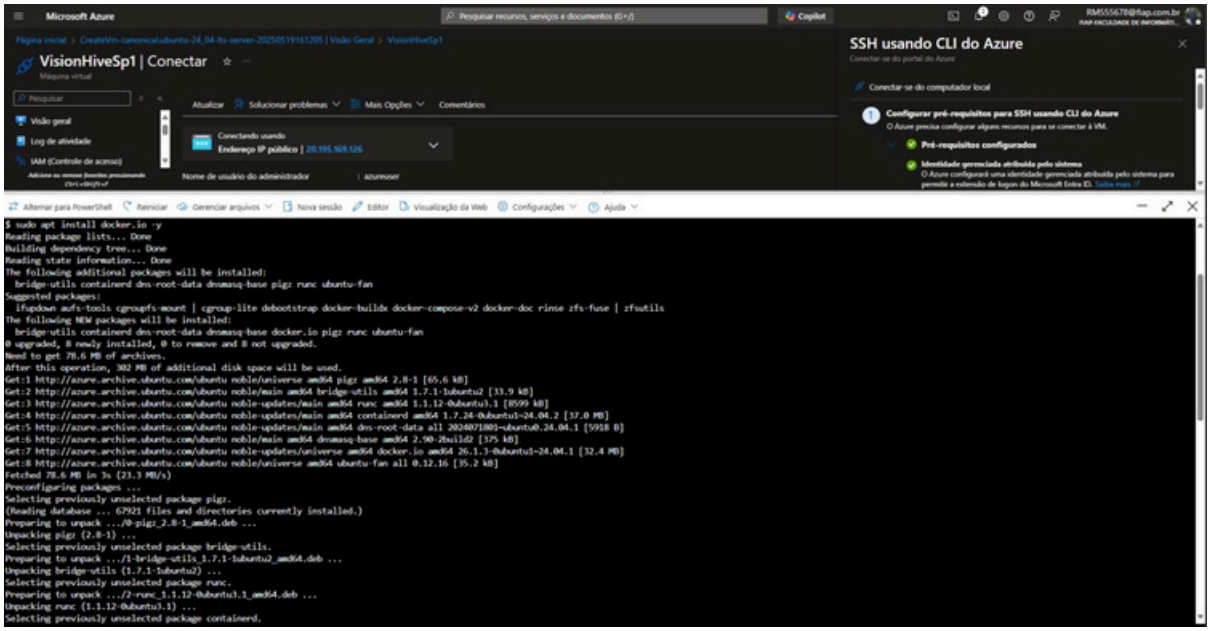


Como o java vai ser acessado pelo o swagger, vamos deixar somente a porta 8080 aberta



Iramos utilizar o ssh do azure mesmo

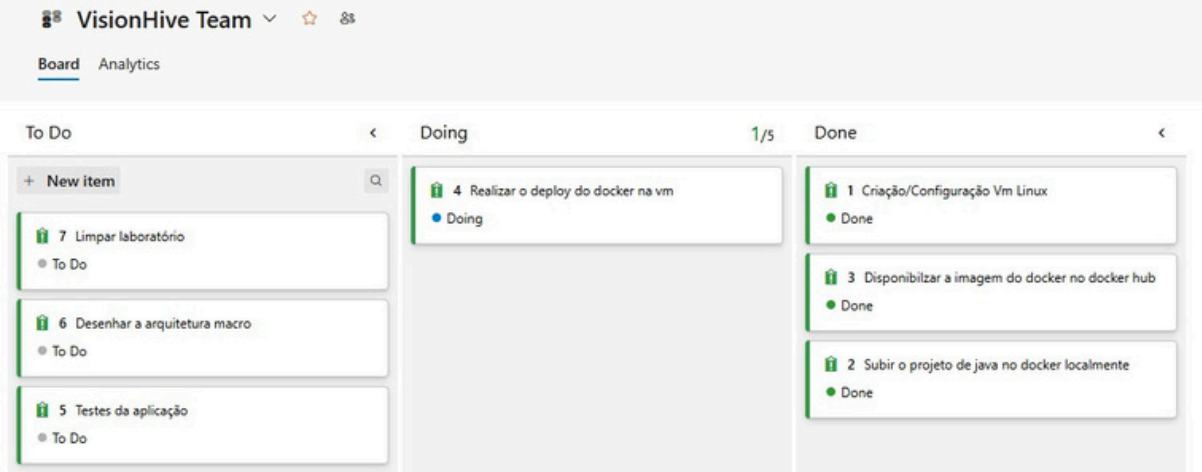
Instalando o docker



Docker instalado na vm com a última versão

```
$ sudo systemctl start docker
$ sudo systemctl enable docker
$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
```

Agora vamos fazer o deploy para a vm

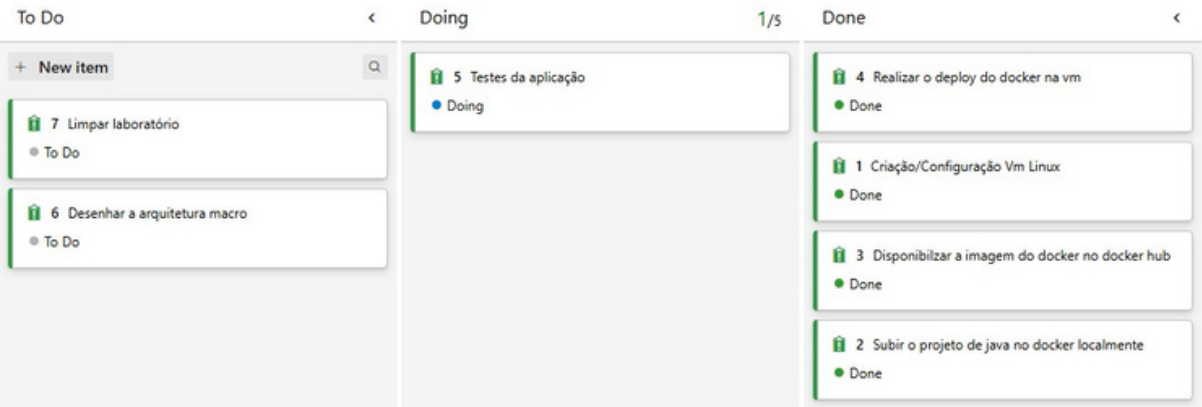


Fazendo o pull da imagem do docker

```
$ sudo docker pull joaomichaeli/visionhive-java:v1
v1: Pulling from joaomichaeli/visionhive-java
5843afab3874: Pull complete
53c9466125e4: Pull complete
d8d715783b80: Pull complete
d902d246dc1f: Pull complete
7992fe636f76: Pull complete
Digest: sha256:9a82dd24a38e6529b03d0de549390be0e6476fe229cda4c4e09db14e92b7434c
Status: Downloaded newer image for joaomichaeli/visionhive-java:v1
docker.io/joaomichaeli/visionhive-java:v1
```

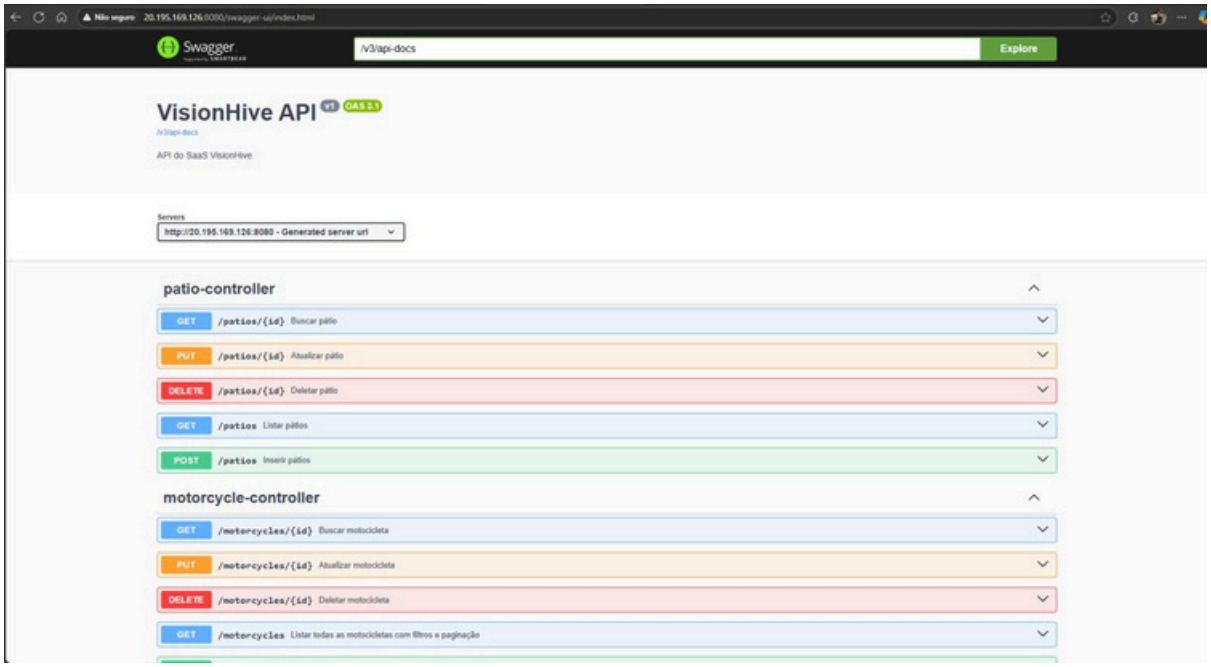
```
$ sudo docker run -d -p 8080:8080 joaomichaeli/visionhive-java:v1
ca62c090c390c341df6494a500c5e50d407f13af55772369cce7d951f2eb8d9a
```

Como havia demonstrado no começo, o java ainda não possui interface web, o acesso é somente do swagger

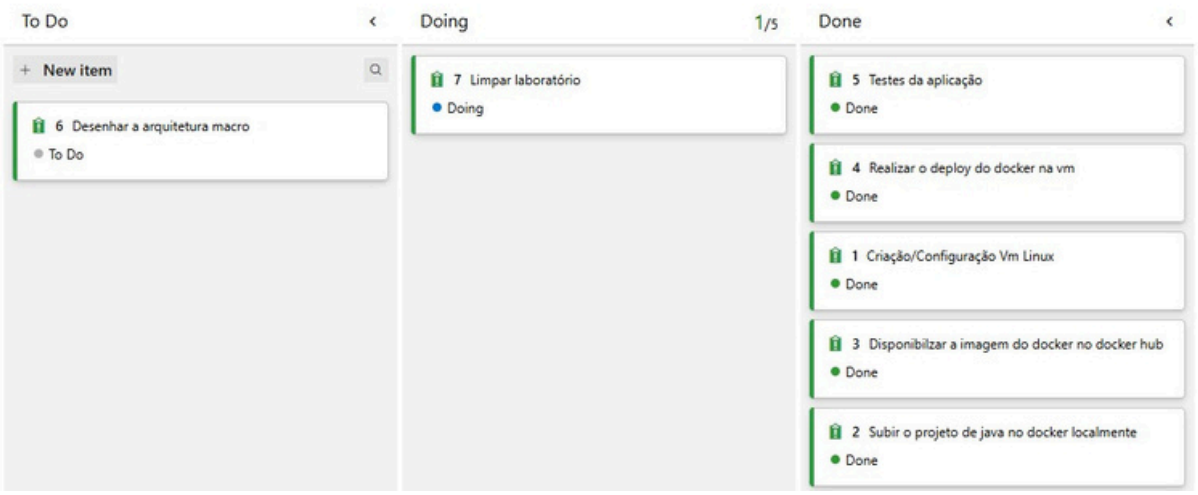


O swagger fica disponível no seguinte acesso:

http://<IP_VM>/ swagger-ui/index.html

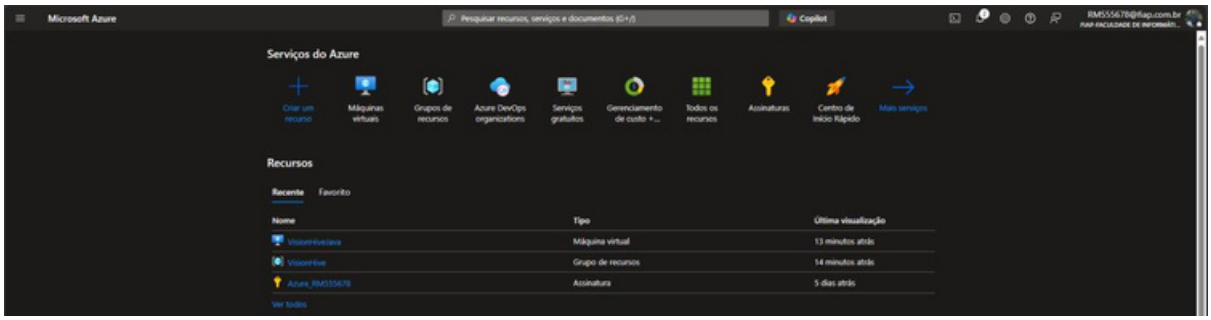


Agora é a limpeza do laboratório

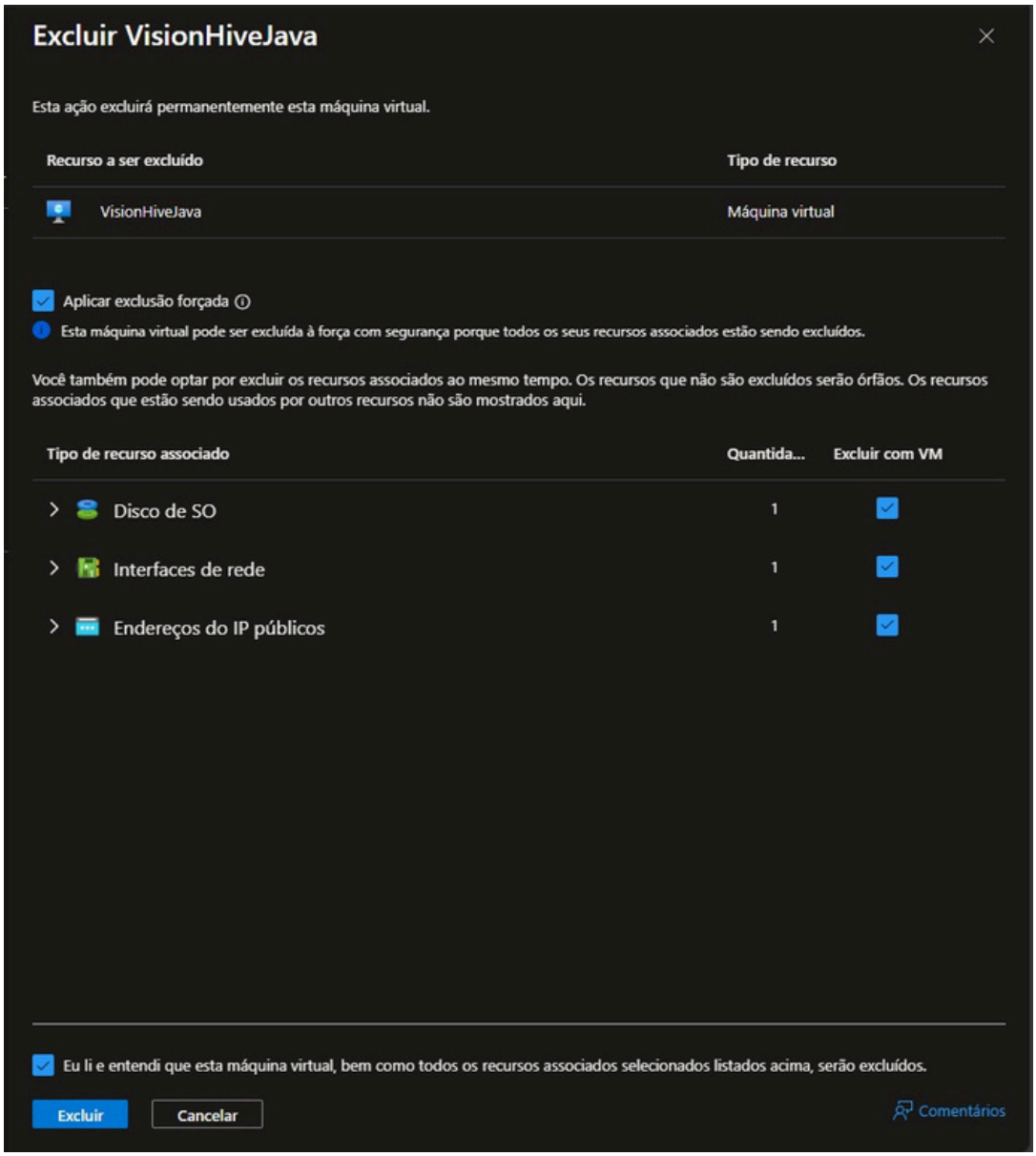


Primeiro vamos limpar o Azure

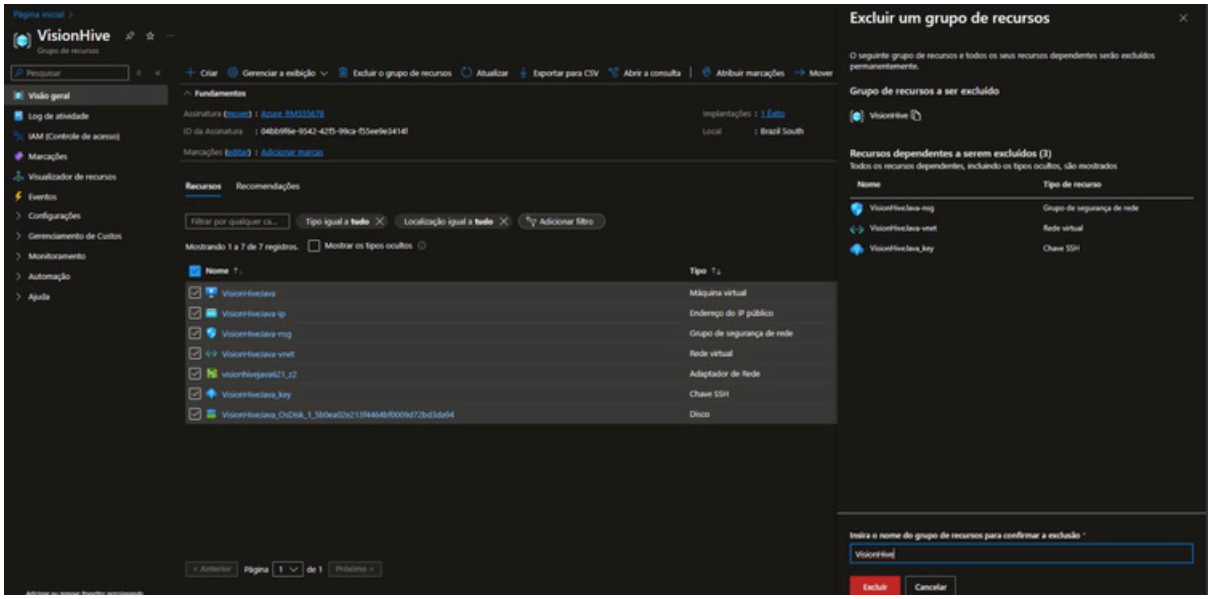
Serviços criados



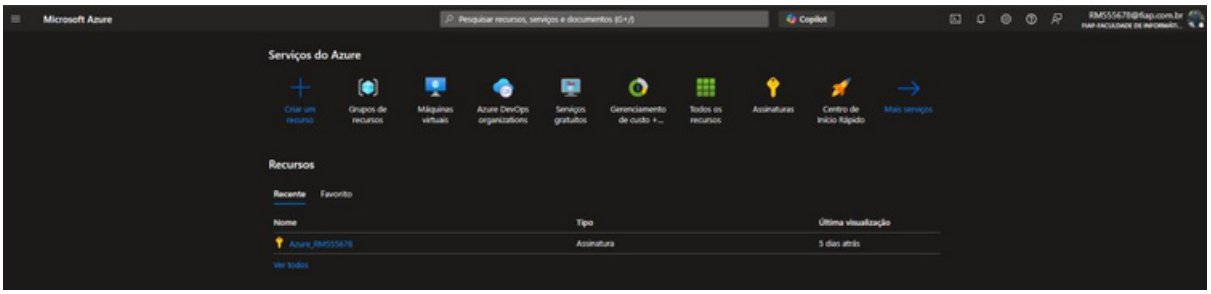
Excluindo a Vm



Excluindo grupo de recurso

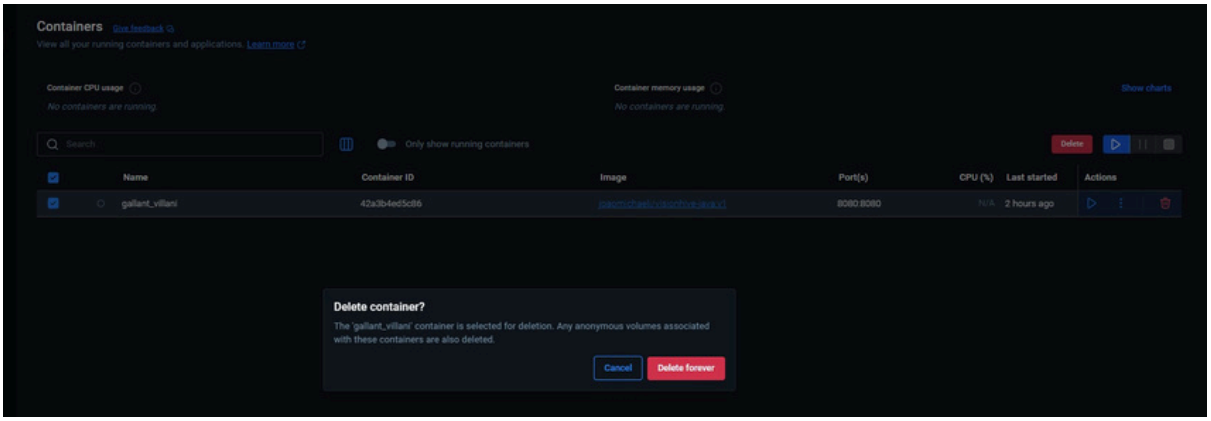


Serviços limpos

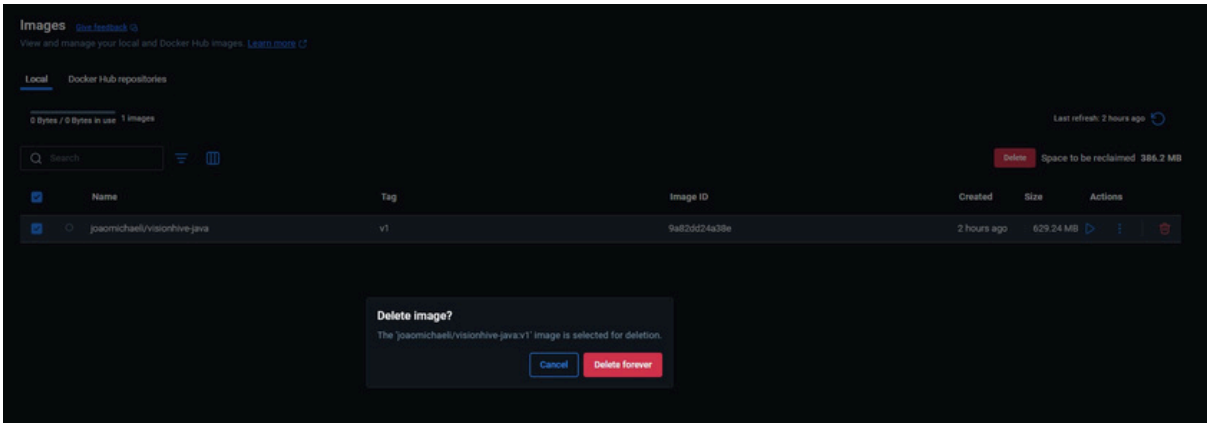


Agora vamos limpar o docker

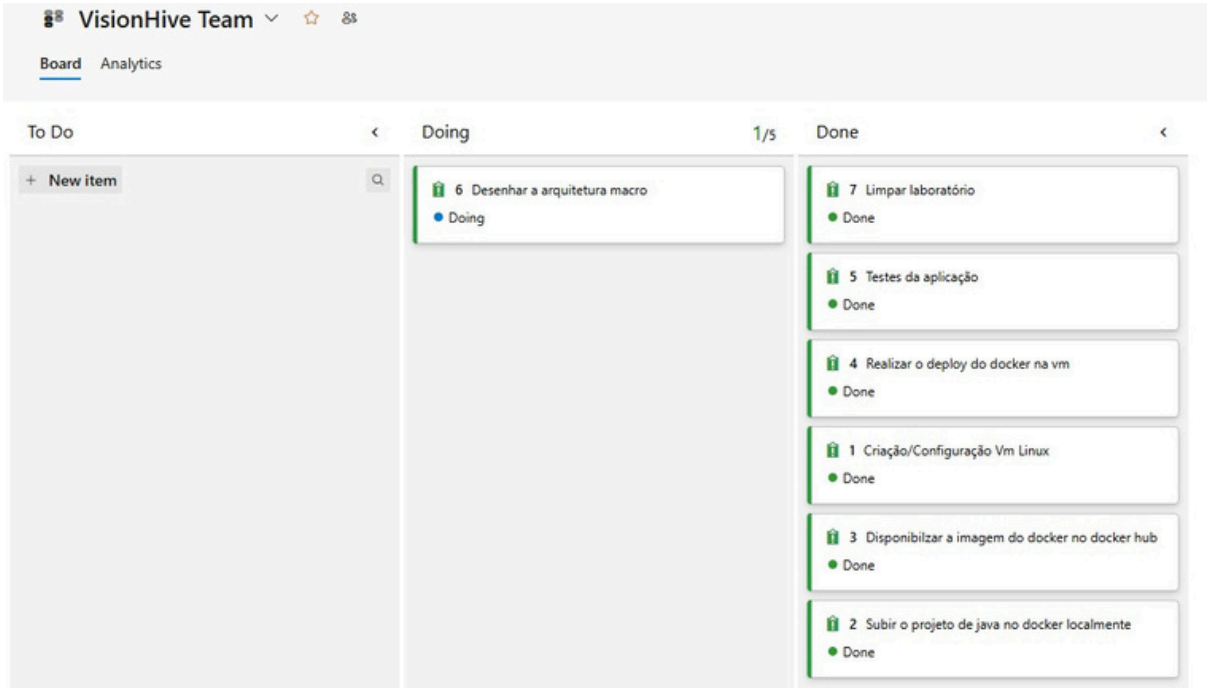
Excluindo o container



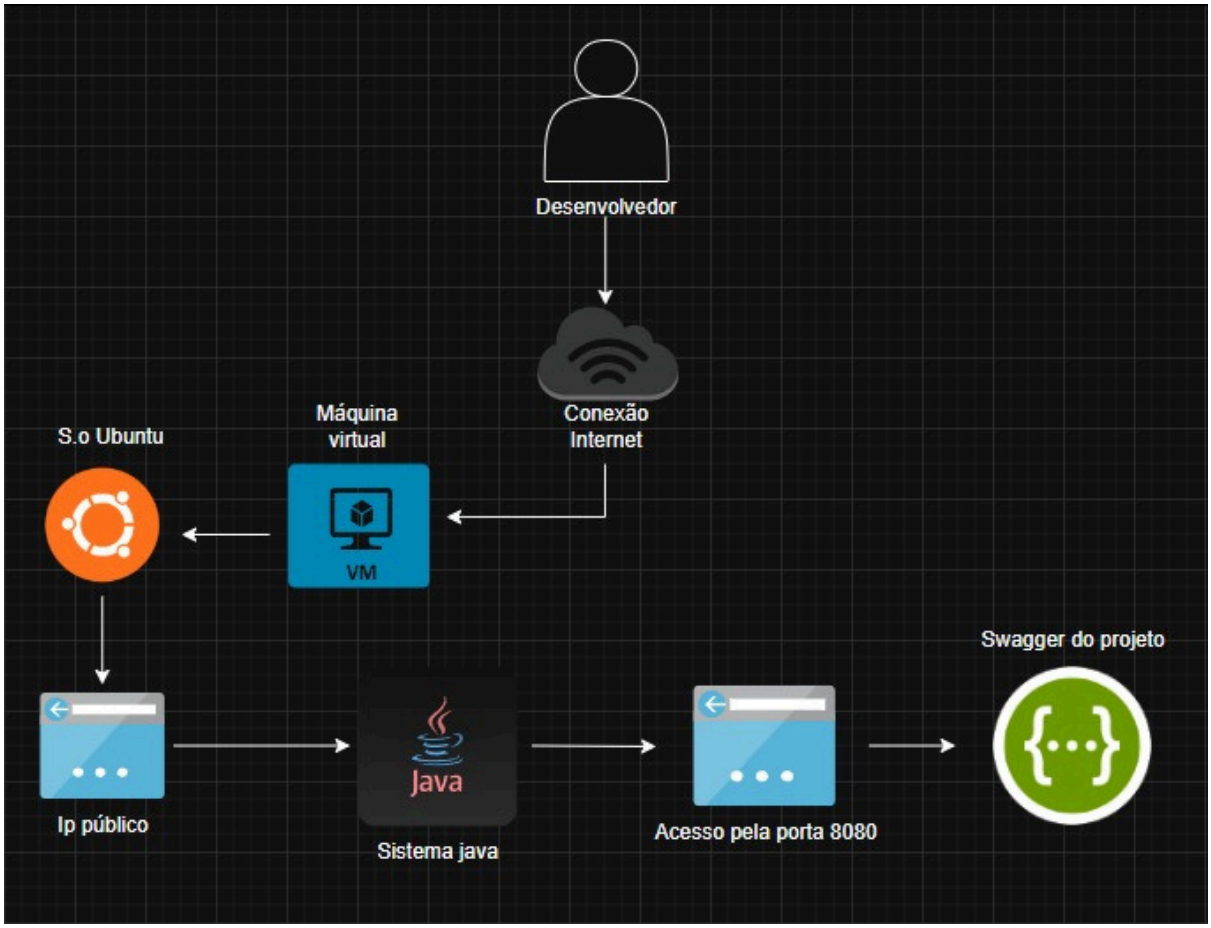
Excluindo a imagem



Agora vamos fazer o diagrama da solução



Arquitetura feita no Draw.io



Board finalizado