Relatório XPTO

## Vagrant Para Cima

Passos

1. Inserir o VagrantFile dentro da pasta do nosso projeto.
2. No gestor Hyper-v, configurar um switch virtual desta forma:

Uma imagem com texto, captura de ecrã, software, Página web

Os conteúdos gerados por IA poderão estar incorretos.

1. Inserir o seguinte conteúdo dentro do VagrantFile:

Vagrant.configure("2") do |config|

# Box

config.vm.box = "gusztavvargadr/sql-server"

# Configurações do provedor Hyper-V

config.vm.provider "hyperv" do |hv|

# Configuração de memória

hv.memory = 2048

# Configuração de CPU (opcional, ajuste conforme necessário)

hv.cpus = 2

end

# Redirecionamento de portas

config.vm.network "forwarded\_port", guest: 1433, host: 1234

# Configuração de rede privada (opcional, descomente se necessário)

# config.vm.network "private\_network", type: "dhcp"

config.vm.network "private\_network", type: "dhcp", bridge: "ExternalSwitch"

config.vm.synced\_folder ".", "/vagrant", disabled: true

end

1. Abrir a powershell como administrador, ir até à localização do nosso ficheiro VagrantFile e após isso, executar o seguinte commando: “**vagrant up --provider=hyperv**”. A máquina virtual será então configurada e iniciada.
2. Após a máquina virtual iniciar, para que as máquinas se consigam pingar, foi necessário abrir a powershell como administrador e executar o seguinte commando para alterar a configuração da firewall: “**New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4 -Direction Inbound -Action Allow**”.
3. Abrir o Microsoft SQL Management Studio e efetuar login com Windows Authentication. Após isso, iremos alterar a password do utilizador SA de forma a conseguirmos usar o mesmo na conexão do nosso projeto. Para isso seguimos o caminho Security > Login > SA (Clicar com o botão direito) > Properties. Alterar a password para “Teste123!”. Clicar no botão “OK” para aplicar a mudança.
4. No nosso projeto efetuamos a instalação das seguintes bibliotecas:
   1. Microsoft.EntityFrameworkCore.SqlServer;
   2. Microsoft.EntityFrameworkCore.Tools;
   3. Microsoft.EntityFrameworkCore.Design;
5. Configuramos o ficheiro appsettings.json com a ligação à base de dados da nossa máquina virtual;

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=192.168.1.164,1433;Database=Reservations;User Id=sa;Password=Teste123!;"

}

}

1. De seguida executamos os seguintes commandos dentro da pasta do nosso projeto:
   1. dotnet tool update --global dotnet-ef
   2. dotnet ef migrations add InitialCreate
2. Após isso, tivemos que ativar o protocolo TCP/IP no SQL Server Configuration Manager e reiniciar o serviço de base de dados para permitir que a máquina onde o nosso projeto consiga comunicar com a base de dados.
3. De seguida executamos este commando “**dotnet ef database update**”.
4. Após este commando, executamos se as chamadas que efetuamos pela API são refletidas na base de dados e sim, estão a ser refletidas.
5. De seguida criamos um docker file assim configurado:

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

WORKDIR /app

COPY . .

RUN dotnet restore

RUN dotnet publish -c Release -o out

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime

WORKDIR /app

COPY --from=build /app/out .

EXPOSE 8080

CMD ["dotnet", "RestauranteFinal.dll"]

1. Criamos um ficheiro .dockerignore com o seguinte conteúdo:

.vagrant/

.git/

.github/

\*.md

Dockerfile

.vs/

obj/

bin/

1. Instalamos e iniciamos o software Docker Desktop, e de seguida, na consola, na pasta do nosso projeto corremos o seguinte commando **“docker build -t restaurante-api .”.**
2. De seguida executamos o seguinte commando **“docker run -d -p 8080:8080 --name restaurante-container restaurante-api”**. “Para criar um container com a imagem que acabamos de criar”.