1. Escolhi uma arquitetura baseada em micro serviço afim de separar o front e para ter uma fácil manutenção e escalabilidade, no qual temos um proxy reverso (Nginx) fazendo com que as apis internas no Docker não se conversem entre si, somente através do Nginx assim um front que for acessar a api, uma boa prática do Heroku.
2. Api (.net core 2) (container 1)

Como um dos requisitos era subir em algum container (Docker) então optei em utilizar o .net core 2 no qual ele implentou 100% o .net framework 4.6.1, achei melhor utilizar apenas web.api separando totalmente do frontend.

1. Web (NodeJS) (container 2)

Decidi por separar totalmente o front do backend, como não utilzei alguma ferramenta como gulp ou webpack, acabei optando por utilizar o NodeJS para fazer todo o front.

4 – Nginx (Proxy Reverso), requisito para não deixar a porta exposta.

5 – Docker, estou utilizando o Docker compose, assim para melhorar a manipulação de todos os containers do projeto.

6- Estava colocando o Redis para fazer o cache do cardápio e dos ingredientes na view, mas faltou um pouco do tempo.