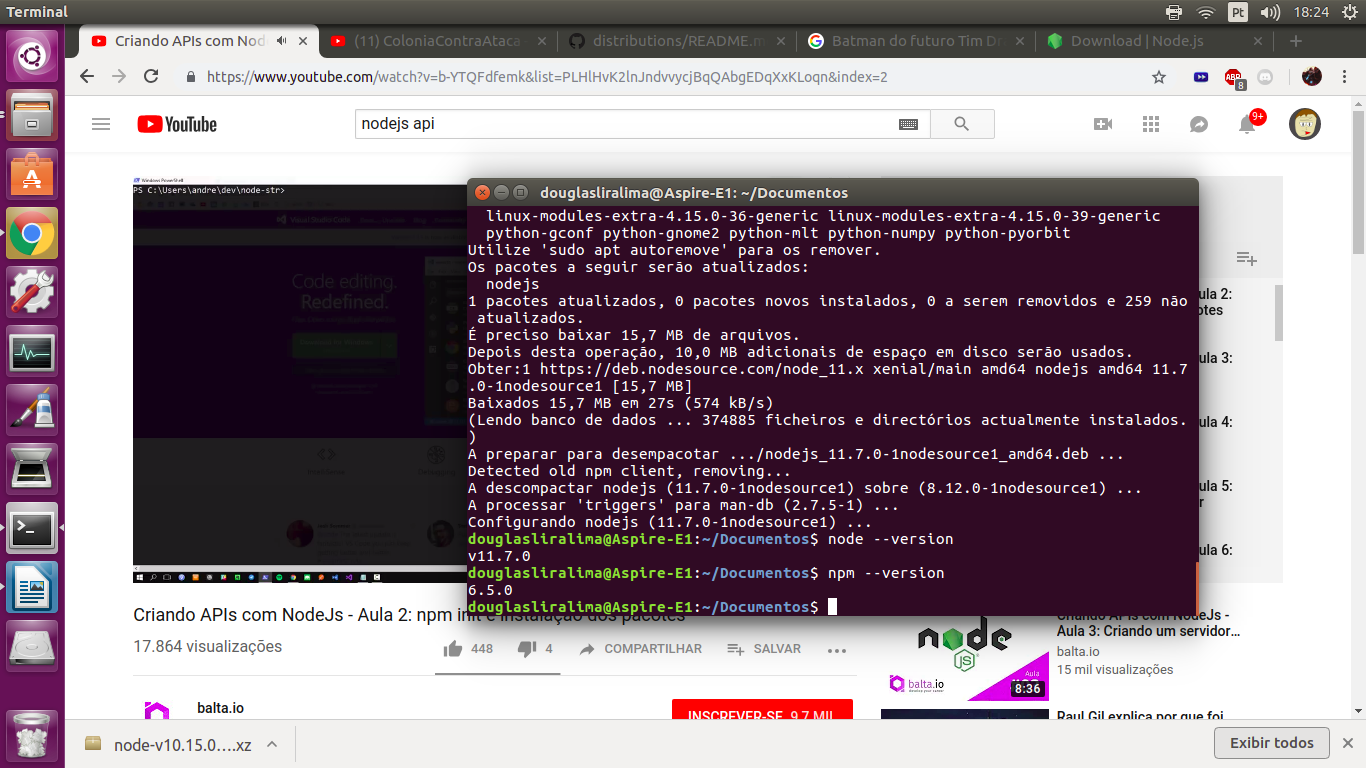
Primeiro você precisa intalar o nodejs na sua máquina, se for windows é só ir no site, se for ubuntu, instruções no site <https://github.com/nodesource/distributions/blob/master/README.md>

Podemos ver se ele está istalado com os comandos npm --version(node package manager)(gerenciador de pacotes do node o pip do node) e node --version(executor de javascript)



Para começar um projeto nodejs se cria como em django, só que com um comando diferente:

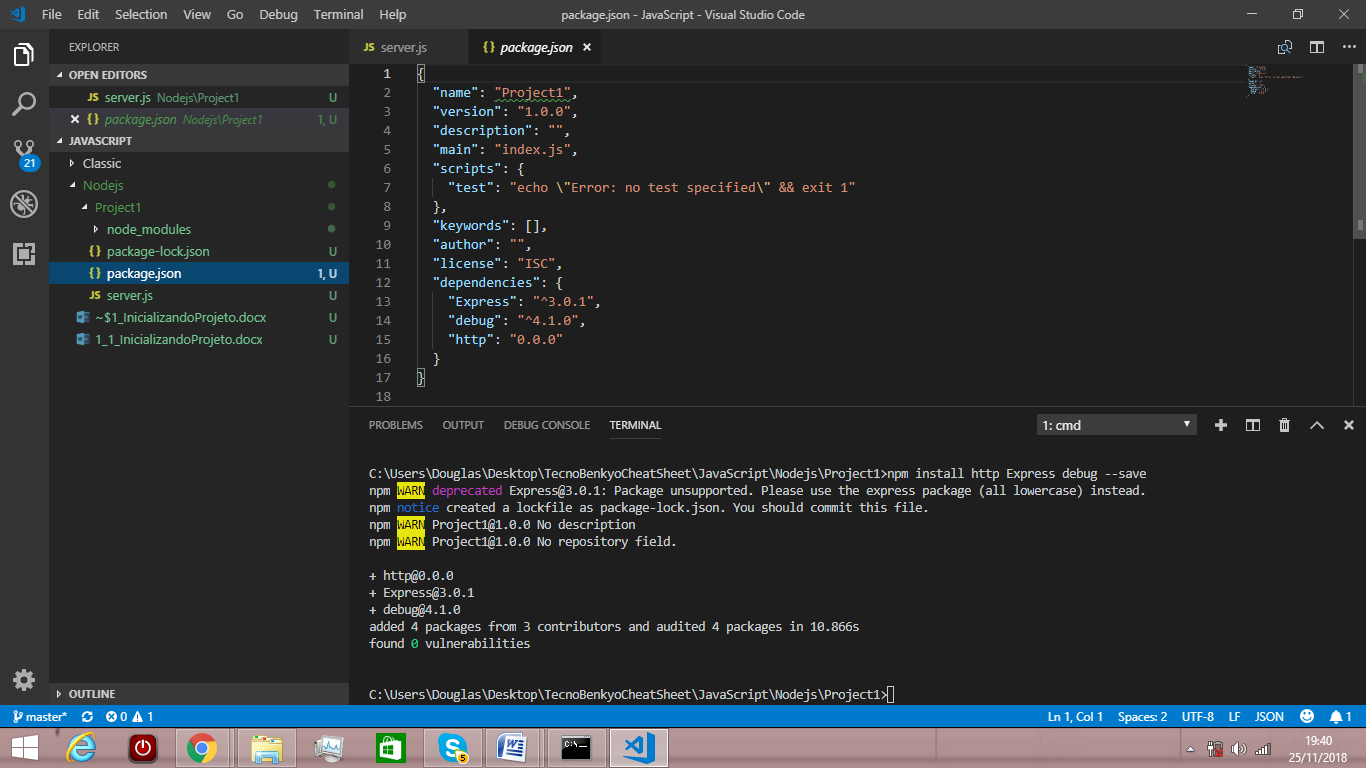
npm init ou npm init –y(caso queira um projeto padrão)

Com isso ele vai inicializar um json com umas configurações padrões do nodejs, existe um monte de informações nesse json q são colocados o padrão pelo –y ou então a gente vai construindo ele.

O nodejs é basicamente puro javascript, logo nos precisamos instalar as coisas com o npm(node package manager). Ele tem a capacidade de receber e mandar requests por si só, nós declaramos uma porta em que o nodejs deve ficar olhando para gerenciar de algum tal jeito toda vez que receber um request. Instalamos três pacotes agora, pois o node sozinho não consegue criar um servidor http pelo comando:

npm install http express debug --save

Usamos o --save para que fique tudo em uma paste q a gente possa dar .gitignore e nunca upar ela num servidor, pois assim essas dependências ficam gravados no package.json:

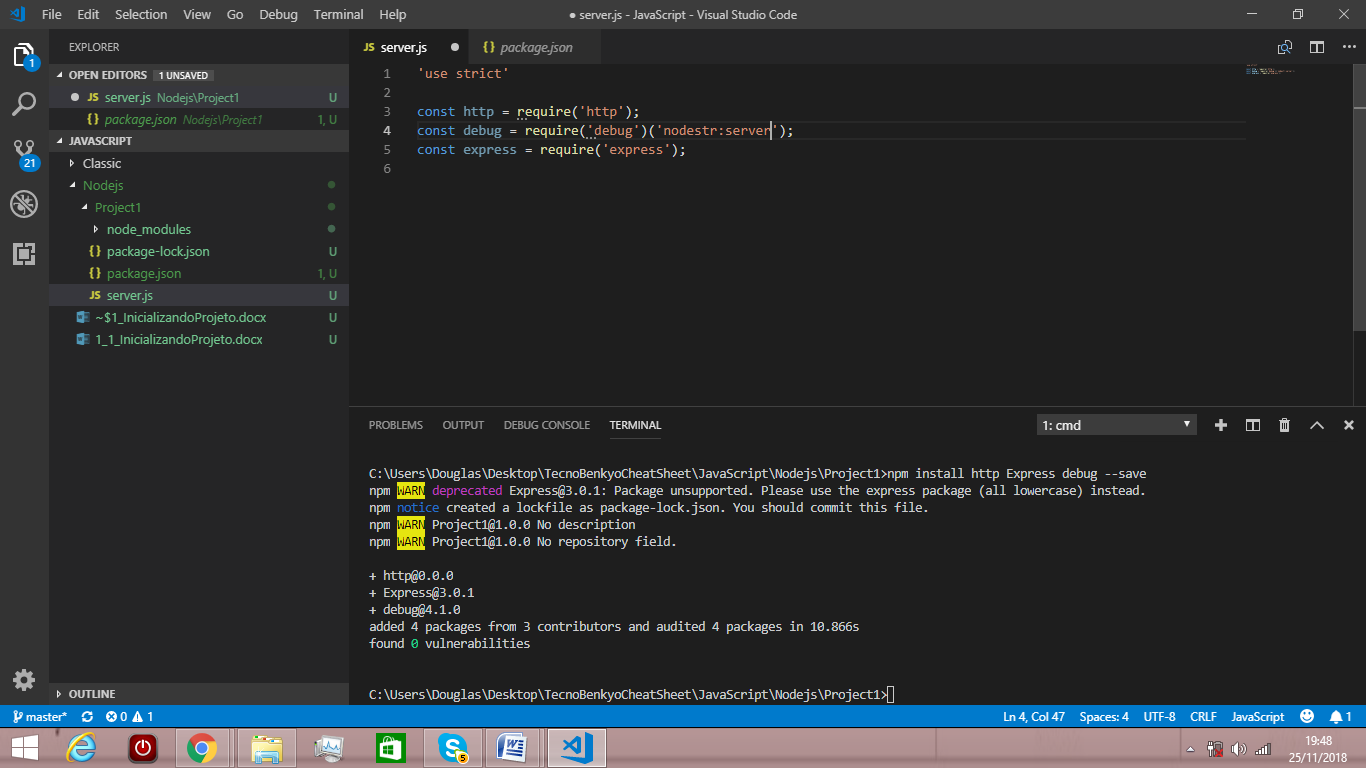


Qualquer pessoa ao pegar o projeto, caso queira reinstalar essas dependências ou pacotes podem as pegar usando o comando abaixo e as dependencies serão checados:

npm install

O debug é para debugar obviamente, o Express é para podermos usar MVC(Model View Controller) e o http é para fazermos de fato o servidor http que possa ser um listener de alguma porta.

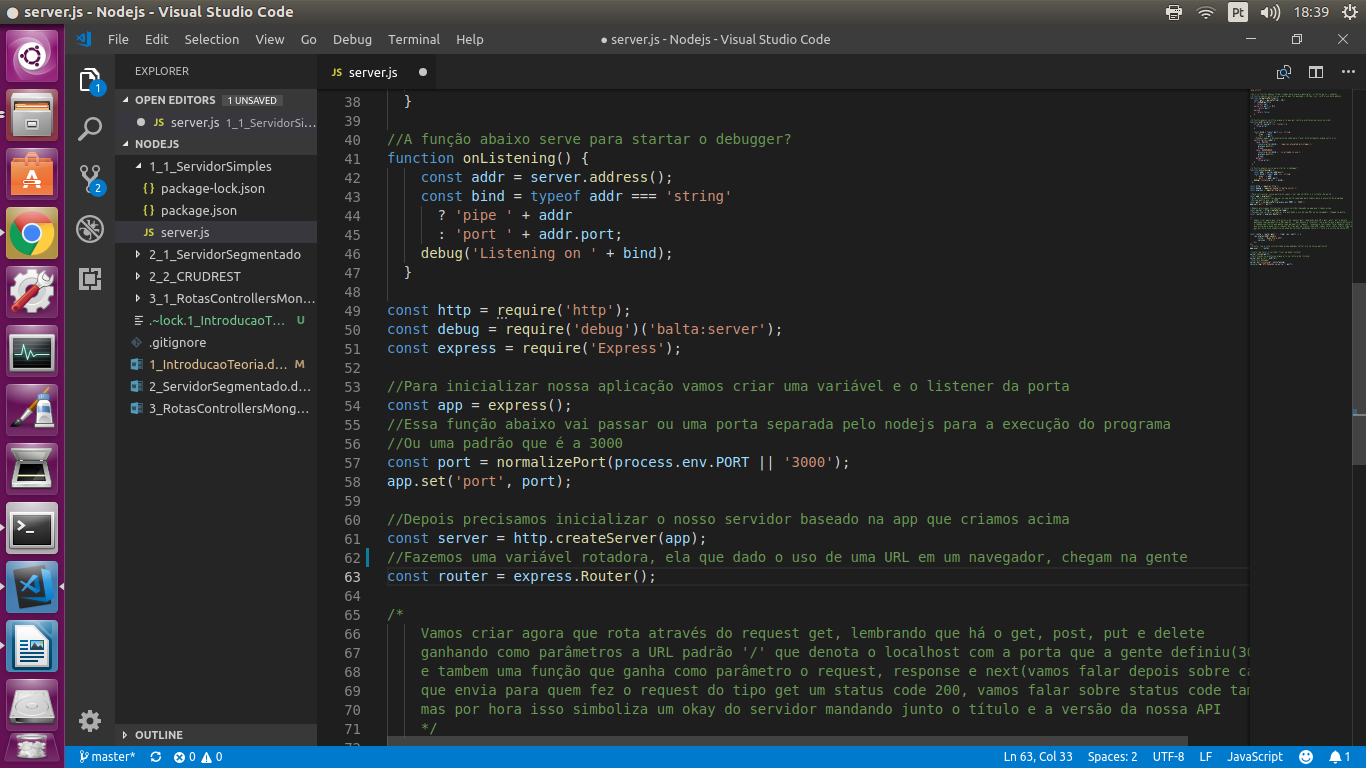
Para criar uma aplicação aqui basta criar um arquivo js chamado Server.js, é ele o arquivo javascript que fará os comandos. A primeira coisa que fazemos lá são a importação desses pacotes que vamos usar, a importação aqui é bem diferente:

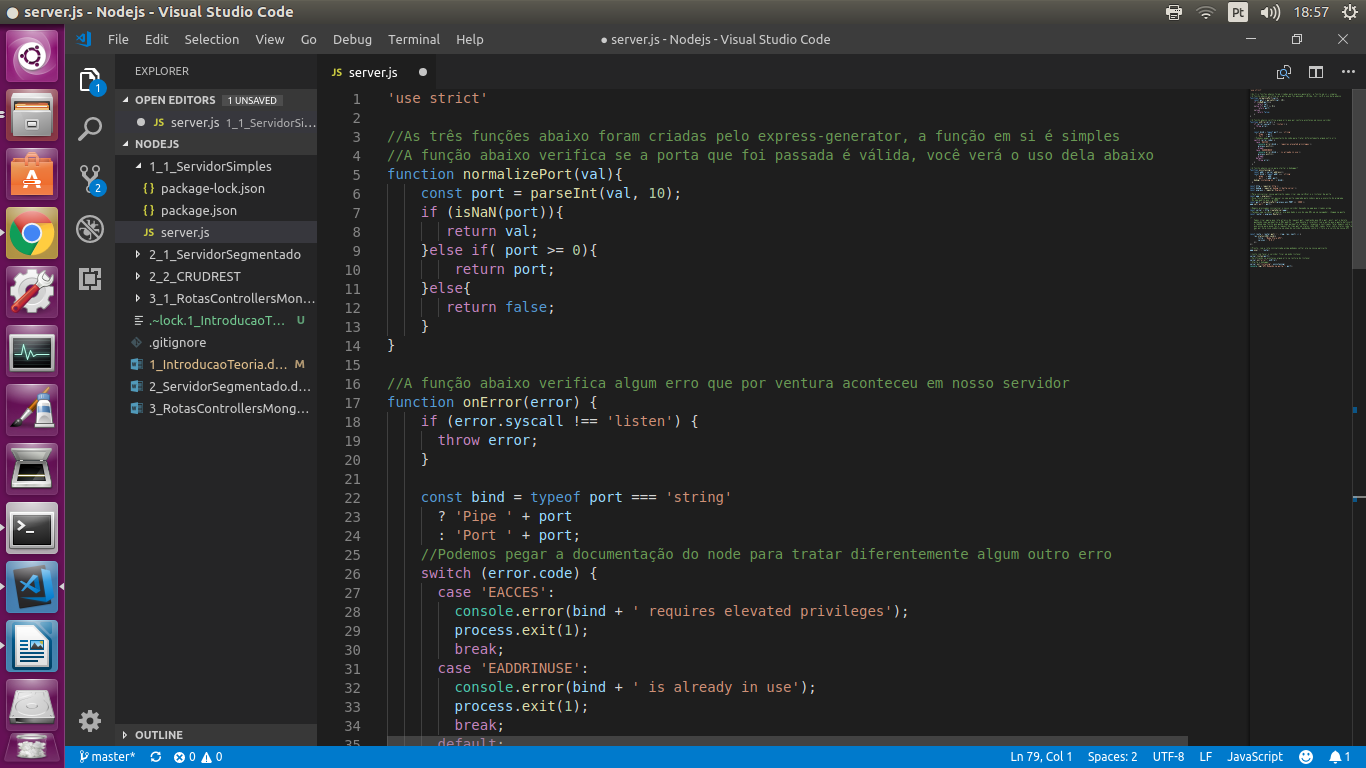


Tudo o que colocarmos sem um caminho ela vai procurar da NodeModules, a partir de agora, olhar os comentários do Server.js que está explicando o resto caso queira, mas vamos transcorrer aqui também

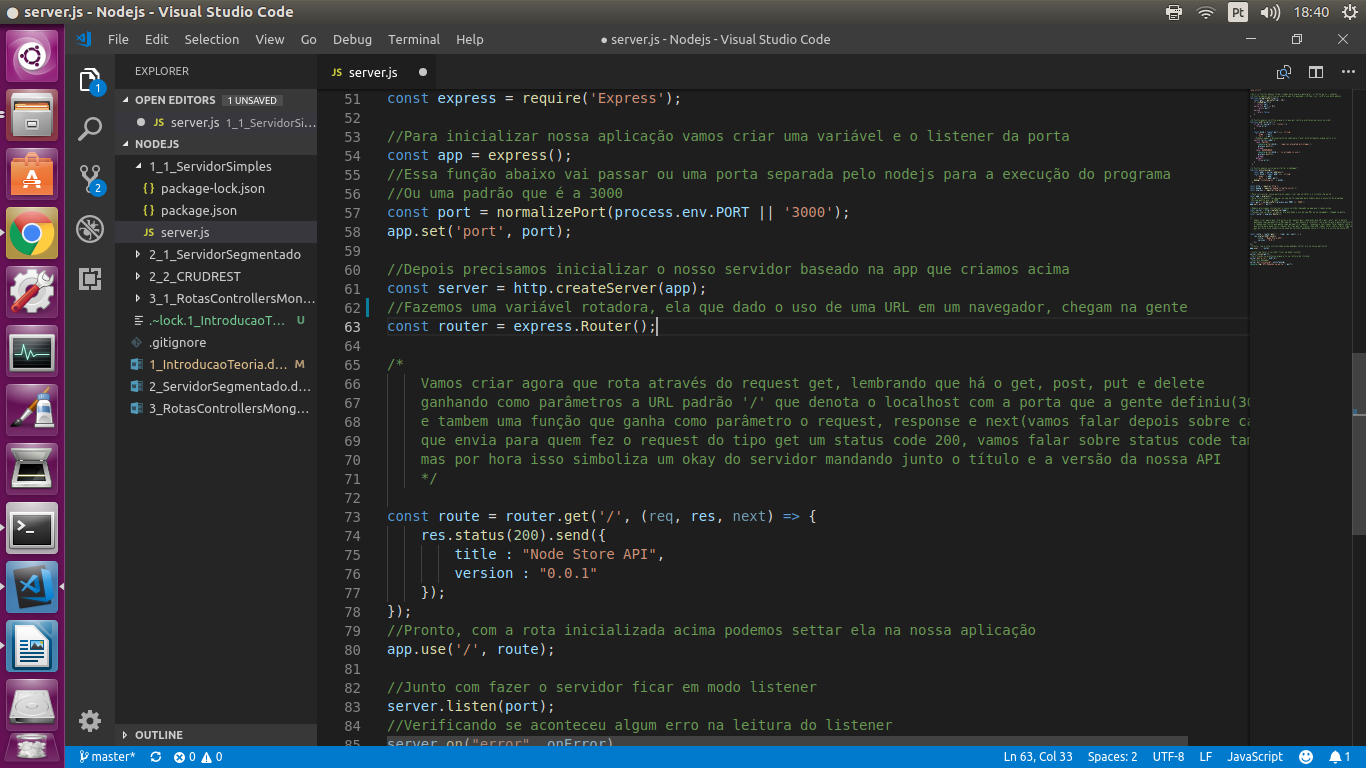
A função normalizePort é uma que o cara diz que pegou pronta de um gerador de código do express, não entendi direito, mas a função tá abaixo também

Toda aplicação web vai ter as variáveis de ambiente de processamento(process.env), nela ele nos daria por padrão uma porta para trabalharmos.



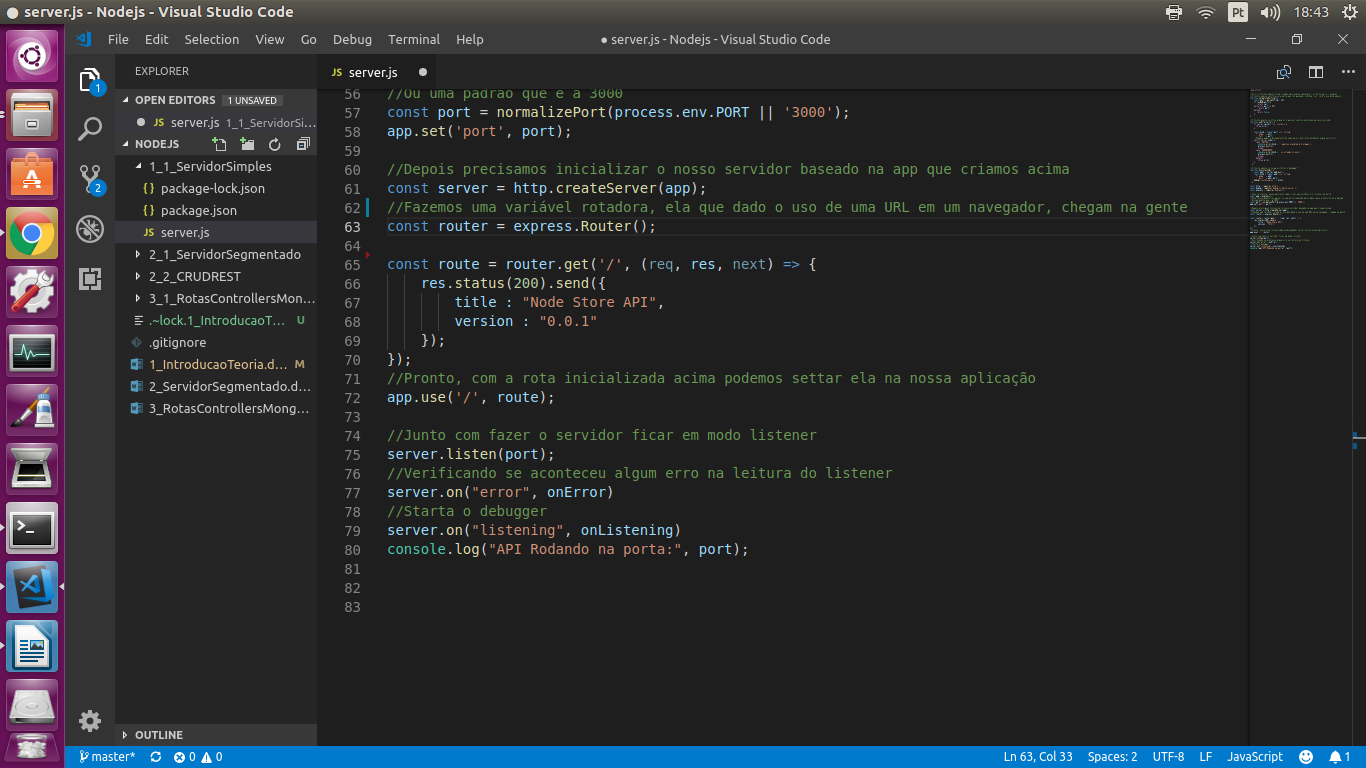


Com o listener criado, podemos passa-lo para o servidor:

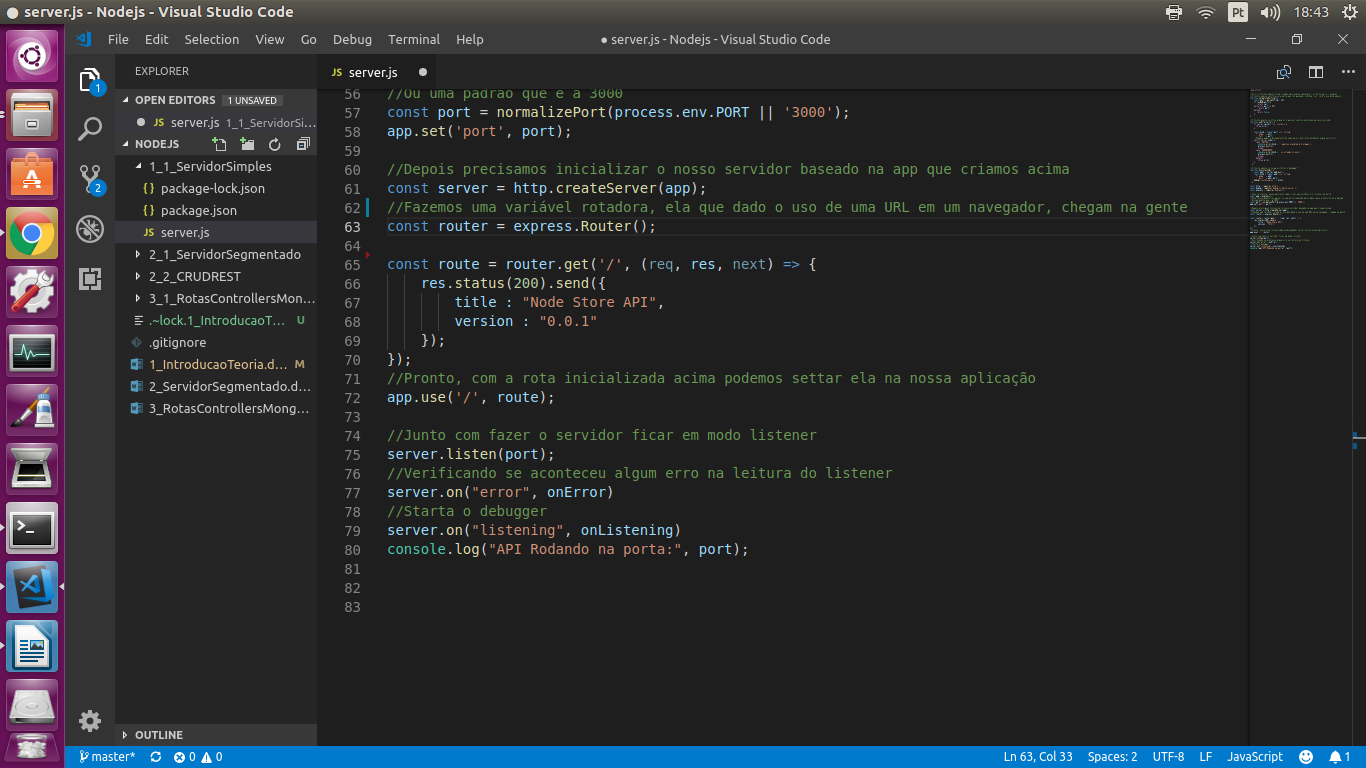


A última coisa que fazemos no listener é definir ações que ele vai tomar dependendo do request que ele receber (existe o get, post, put e delete), veja na linha 65 que definimos definimos que um tratamento para a rota ‘/’(padrão do localhost), para um request do tipo get.

Nessa rota ainda definimos uma função como parâmetro o request, response e next(vamos falar depois sobre cada), que envia para quem fez o request do tipo get um status code 200 na linha 66, vamos falar sobre status code também, mas por hora isso simboliza um okay do servidor, mandando junto um bundle com o título e a versão da nossa API

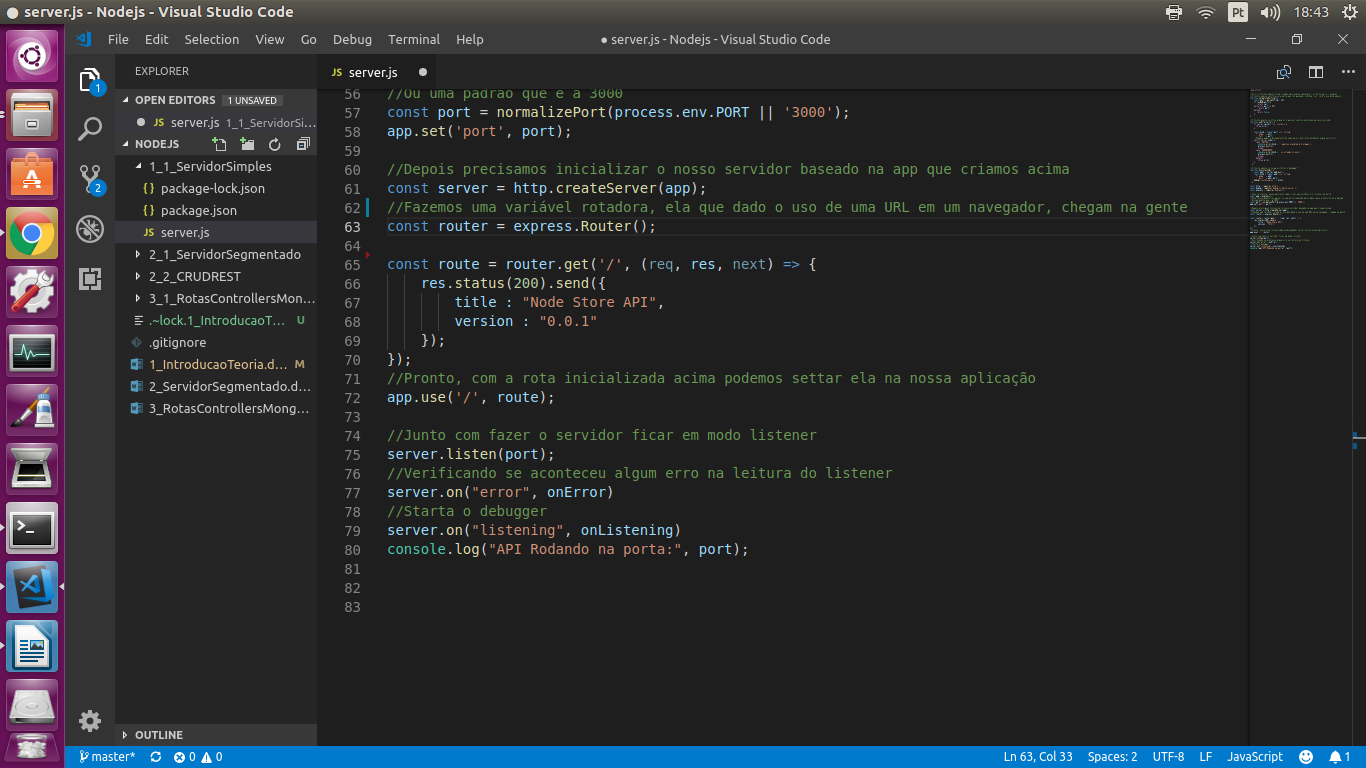


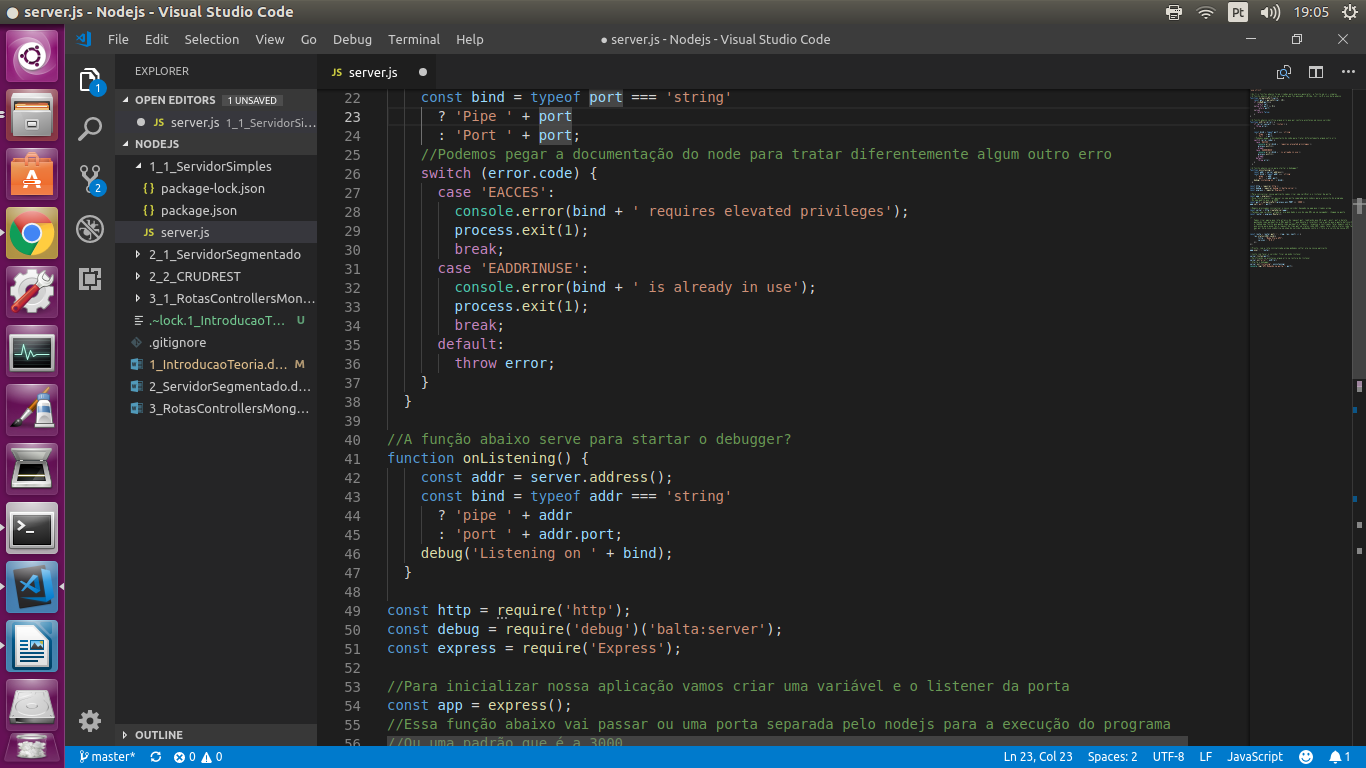
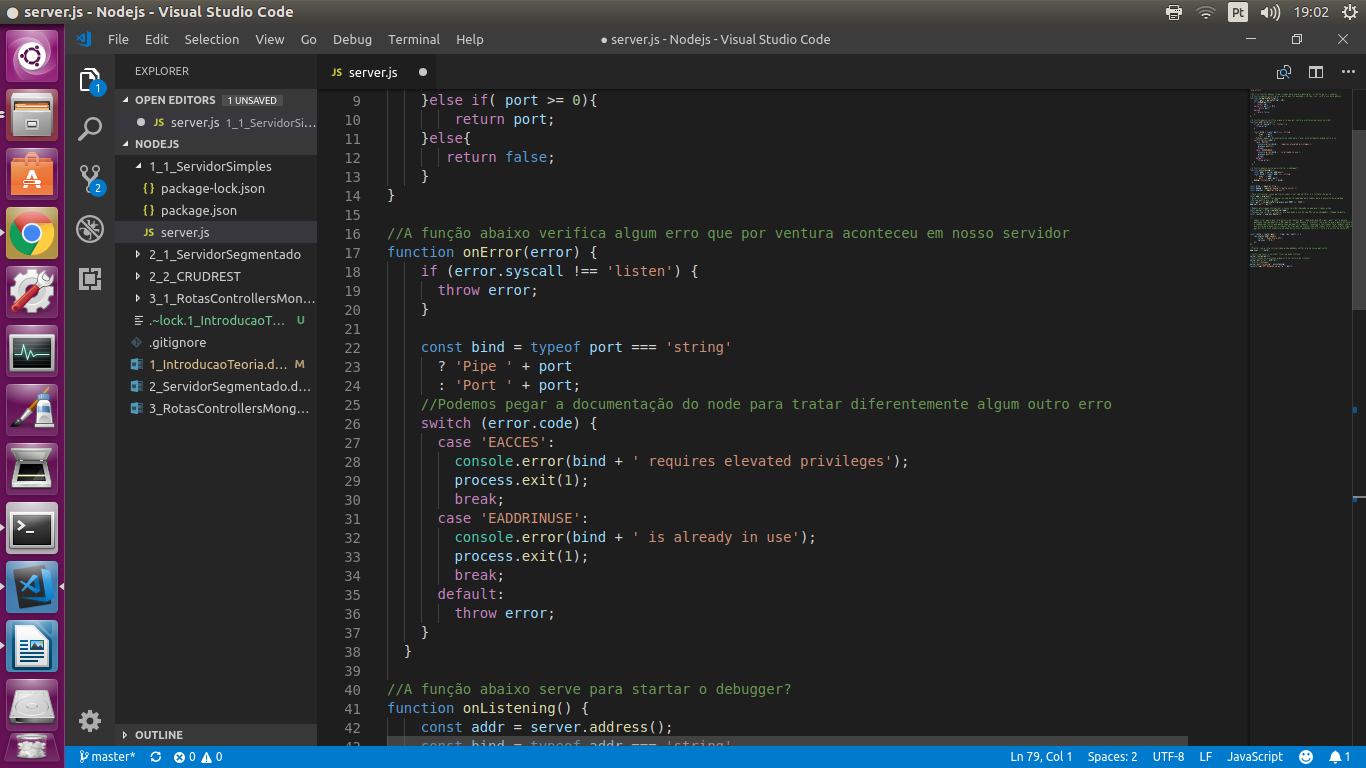
Agora é só deixar o servidor em modo listener



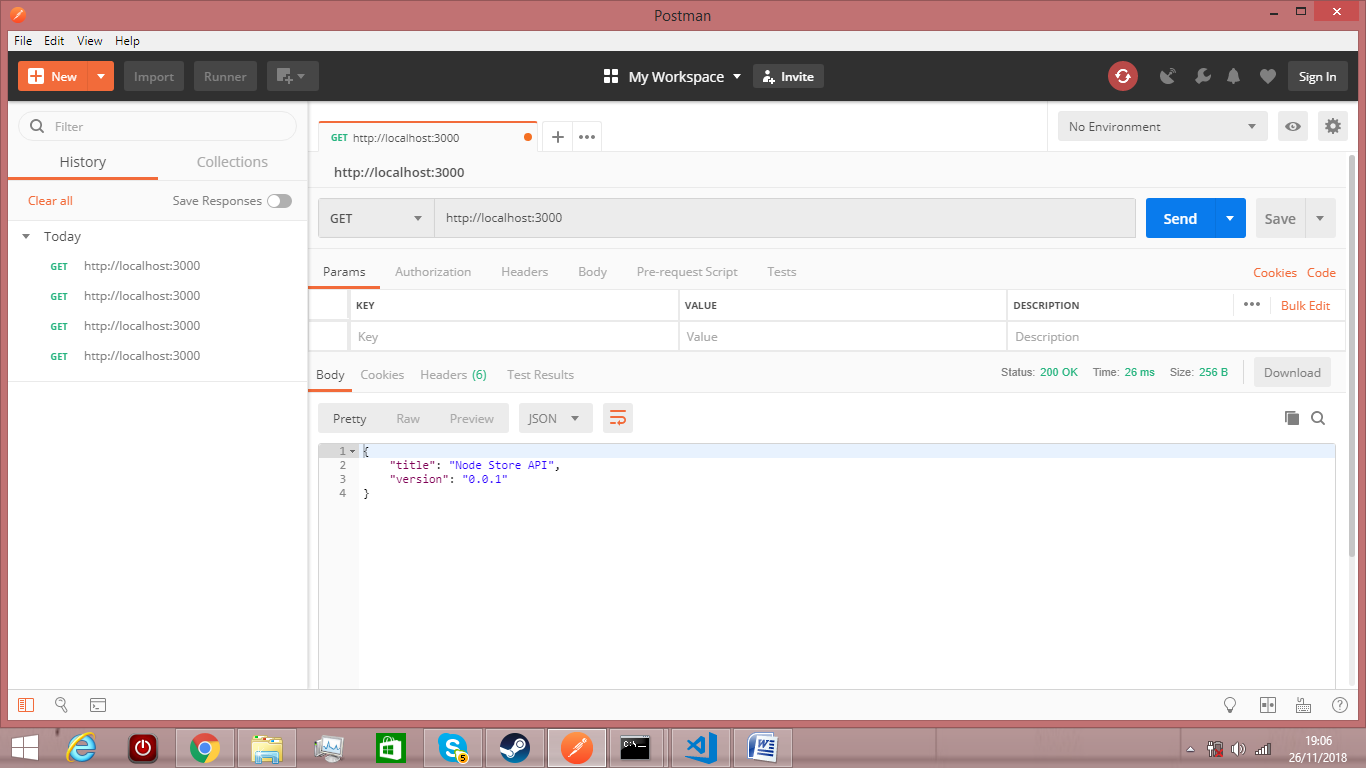
Lidando Com Erros

Vamos fazer uma checagem de erros com as duas funções abaixo, elas também foram dadas pelo express generator e apenas vão te dizer qual erro aconteceu no caso do onError e o evento listening em tese é para inicializar o debugger, mas o cara não disse como funciona ‘-’





Uma coisa útil para testar os nossos programas depois de prontos é fazendo o request de algum cliente, como não estamos dando nen



Basicamente será através dele que vamos dar os get, posts usando o código que definirmos em app que coordena o que se faz com as demandas do cliente