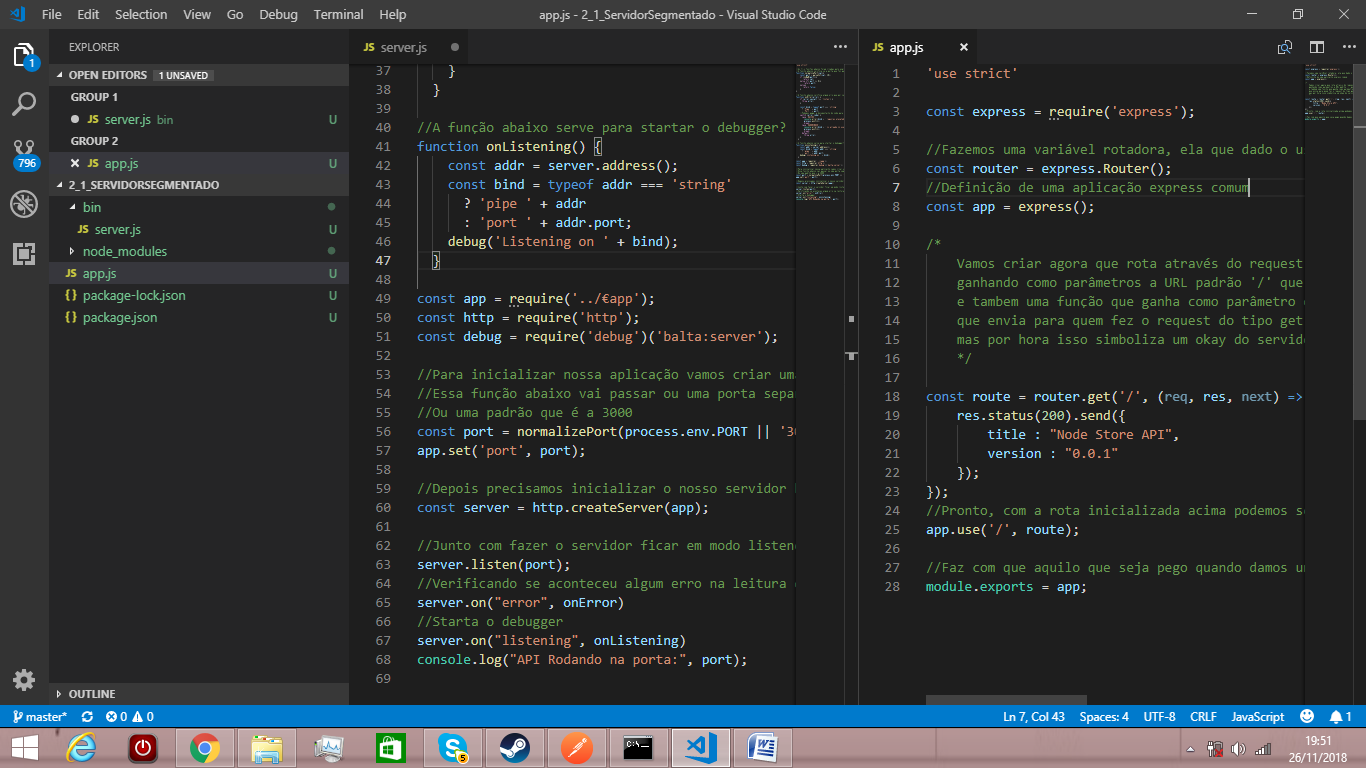
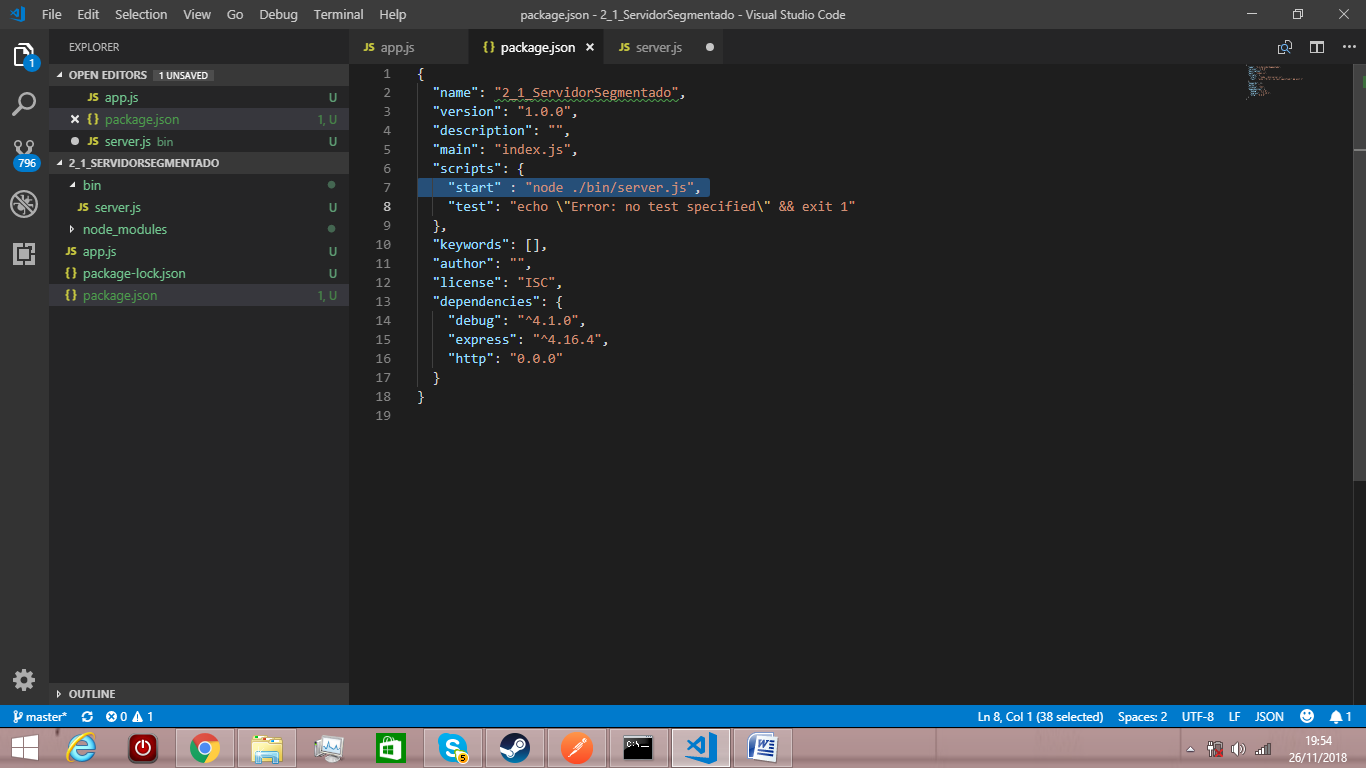
É interessante que vamos segmentando o nosso servidor em files separados para deixa-lo mais fácil de se ler e realizar a manutenção. A primeira coisa que fazemos é separar o app do Server em javascripts separados



Veja que as únicas coisas diferentes nesse código para o que tinhamos antes é que no app temos o código na linha 28 do app.js, e no servidor importamos nosso listener com o código na linha 49 de server.js.

Para deixar mais fácil a produção em nosso documento, podemos declarar scripts para serem usados em sequência após algum comando npm, por exemplo, vamos fazer um comando start através do script que começa a rodar o servidor



Podemos deixar o servidor atualizar sozinho após qualquer alteração nos arquivos dos diretórios para que a gente n precise manualmente ter que ficar atualizando o servidor através do nodemon, ele fica lá observando uma pasta e quando há uma alteração nos arquivos ele att:

npm install nodemon --save-dev

O –dev é para que qndo um outro desenvolvedor dê o npm install, ele n instale automaticamente o nodemon

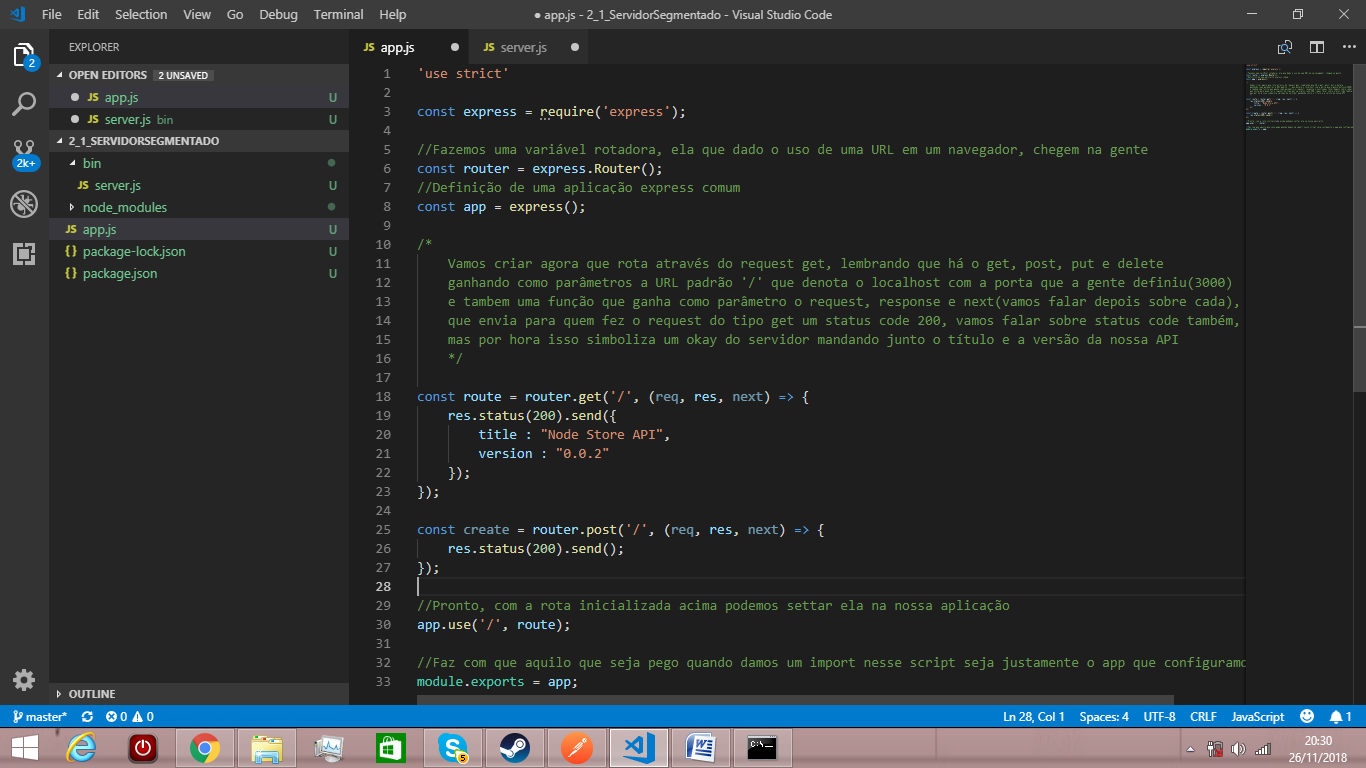
Então basta usar ele ao invés de node para rodar os seus aplicativos, pode se definir um script para uso rápido do nodemon como foi feito com o node

Vamos agora aprender como fazemos os métodos Create, Read, Update e Delete o CRUD da API, vamos aprender o que são cada uma delas quando estivermos definindo suas funções.

Primeiro é bom saber que toda vez que se cria uma rota temos as opções GET, POST, PUT, DELETE, PATH... mas os mais usados são os 4 primeiros, eles são usados para:

GET -> Obter informações, POST -> Enviar informações, PUT -> Atualizar, DELETE -> Excluir alguma informação

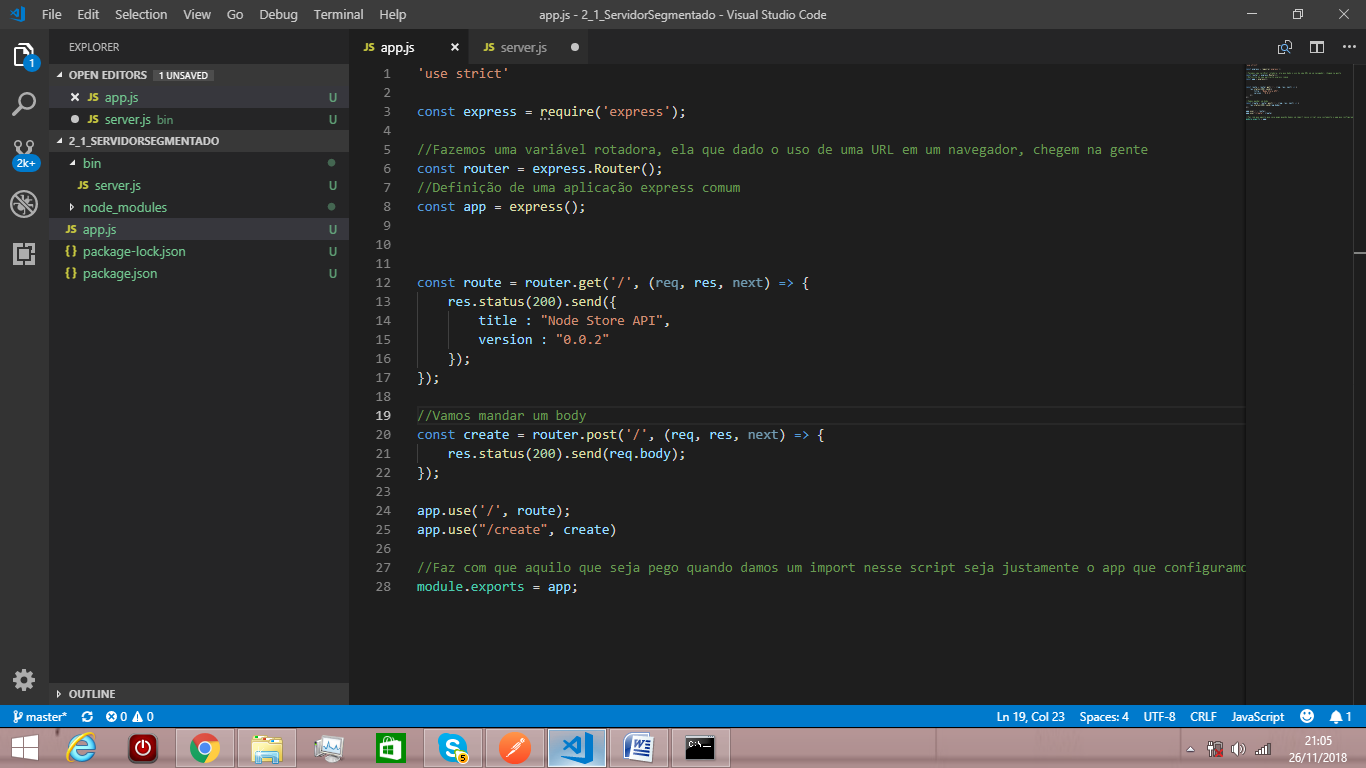
Uma mesma rota pode fazer ter um script para todas essas opções de uma vez, tanto que sempre que vamos fazer algum método CRUD, deixamos eles na mesma rota...



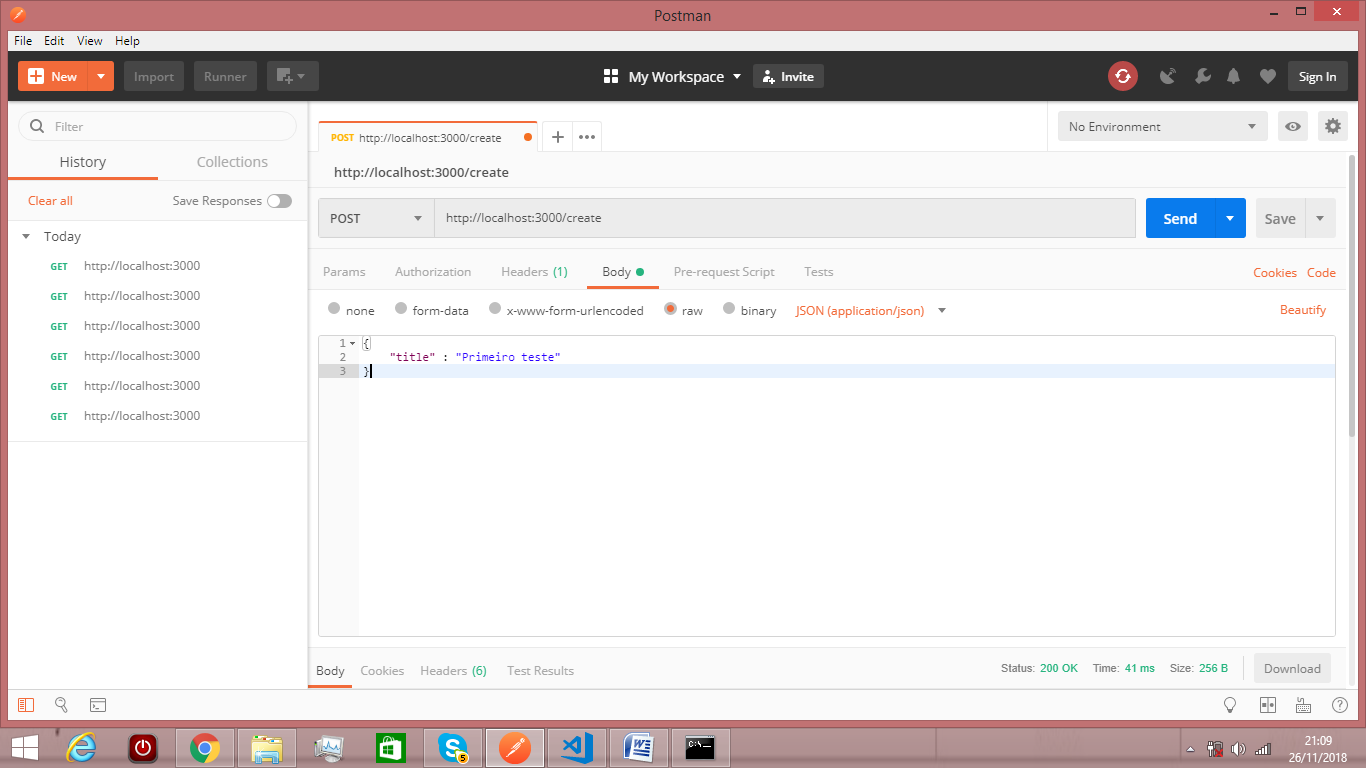
Toda vez que digitamos um endereço de URL ou recebemos um post fazemos um request(req), Se ao fazer um request ou dar um post no servidor, não enviarmos nada, o servidor vai ficar esperando até dar timeout, funciona meio que TCP

Obs: O status 200 -> Ok, 201 -> create , 400 -> BadRequest, 401 -> não autenticado, 403 -> AcessoNegado, 500 -> qualquer erro que tenha acontecido

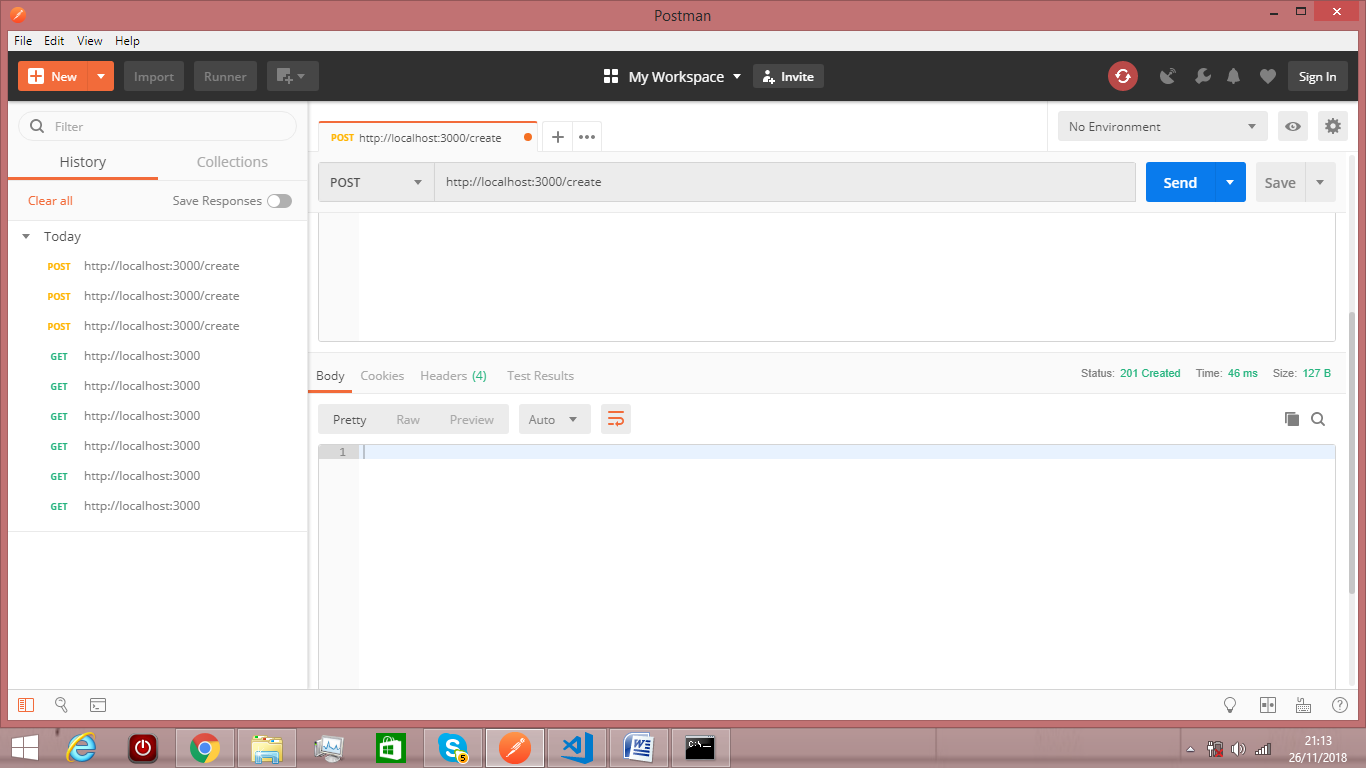
Vamos fazer pegar aquilo que o cliente nós mandar dectando o seu post e criando uma nova rota apenas para poder ser mais visível no postman:



Vamos simular que em seu post o cliente estará nos mandando um arquivo JSON(JavaScript Object Norm):



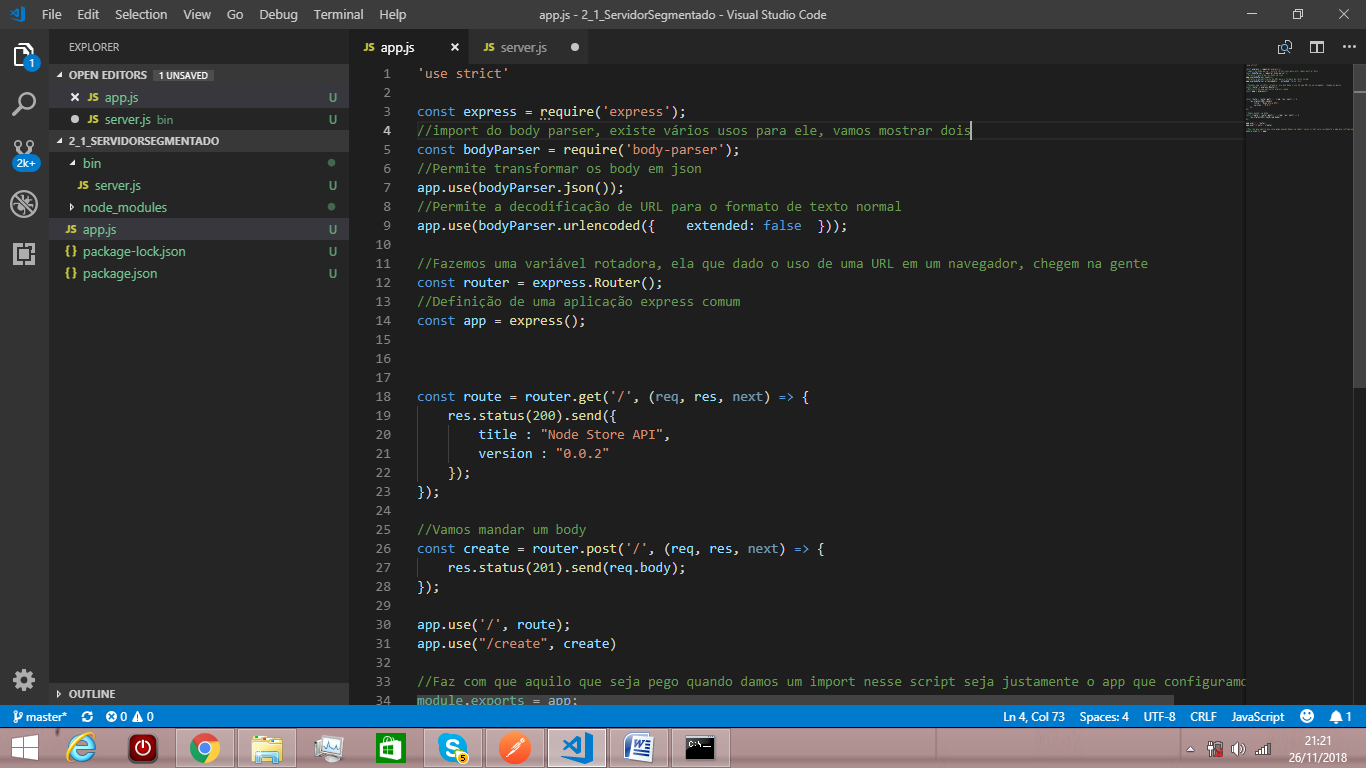
Veja que ao fazer esse send, o servidor nos retorna o código de created, mas ele não conseguiu transformar o req.body em nada, retornando o json que mandamos para ele.



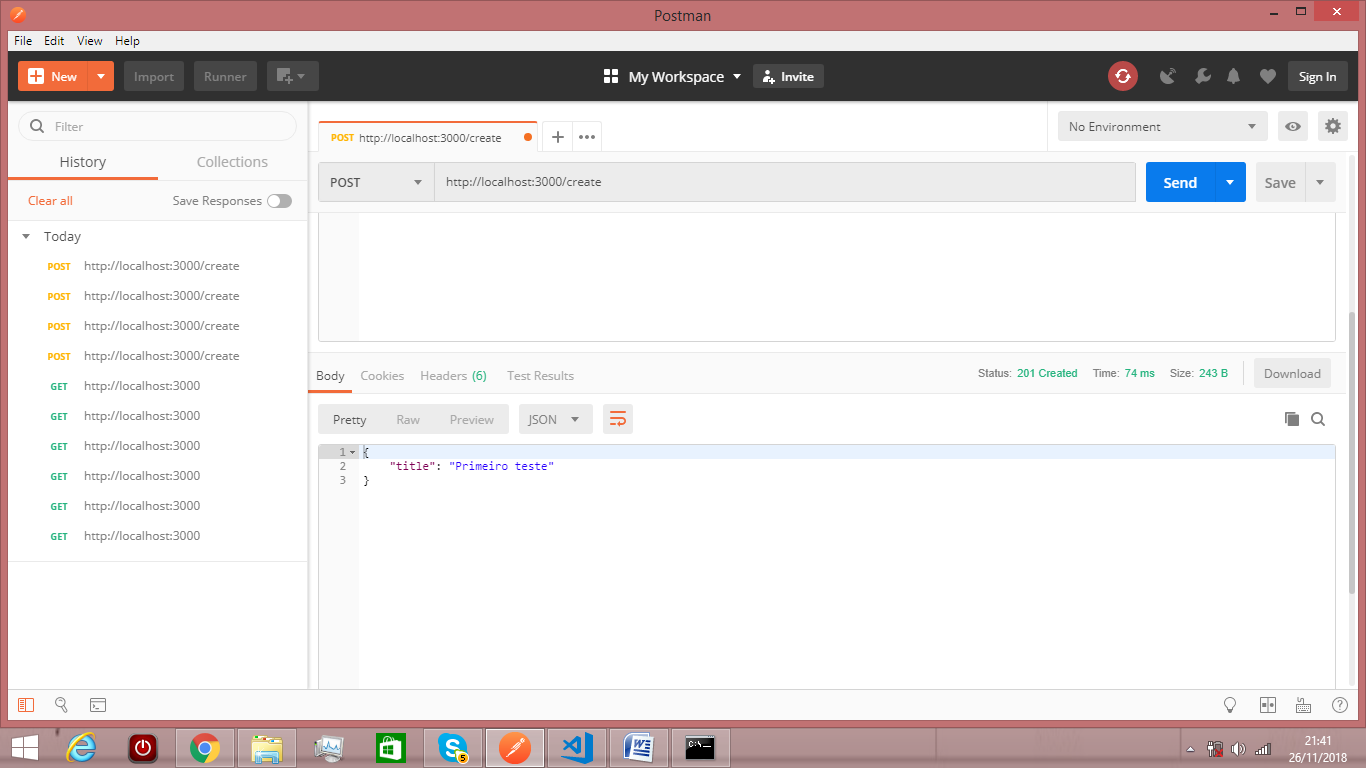
Para fazer isso precisamos de um outro pacote chamado bodyparser do node

npm install body-parser –save

Agora sim podemos usa-lo no programa



E com esse trecho de código, ao dar sendo com o mesmo objeto JSON, o servidor consegue o retornar



Uso do put e do delete segue no app do servidor segmentado, não nenhum outro detalhe lá a respeito disso

