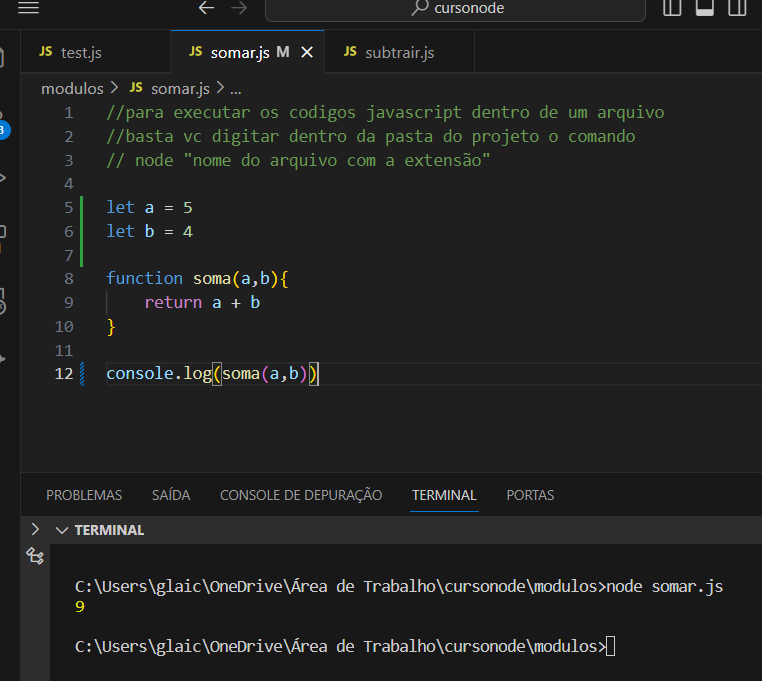
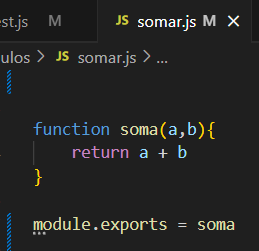
Primeiro passo com node

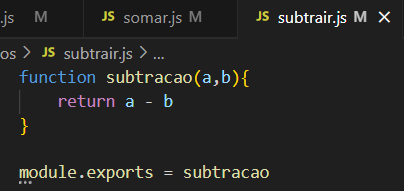


Para executar o código que esta em um arquivo Javascript, basta ir no terminal e digitar o comando node mais “nome do arquivo” e clicar enter. Se tiver algum erro no código vai aparecer no terminal após o enter.

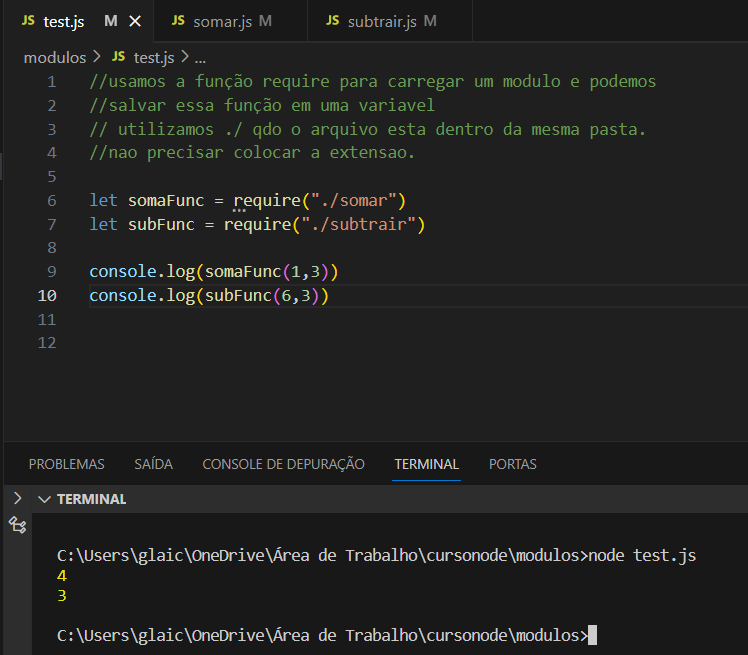
Trabalhando com módulos



Criamos o arquivo somar.js e estamos exportando a função soma



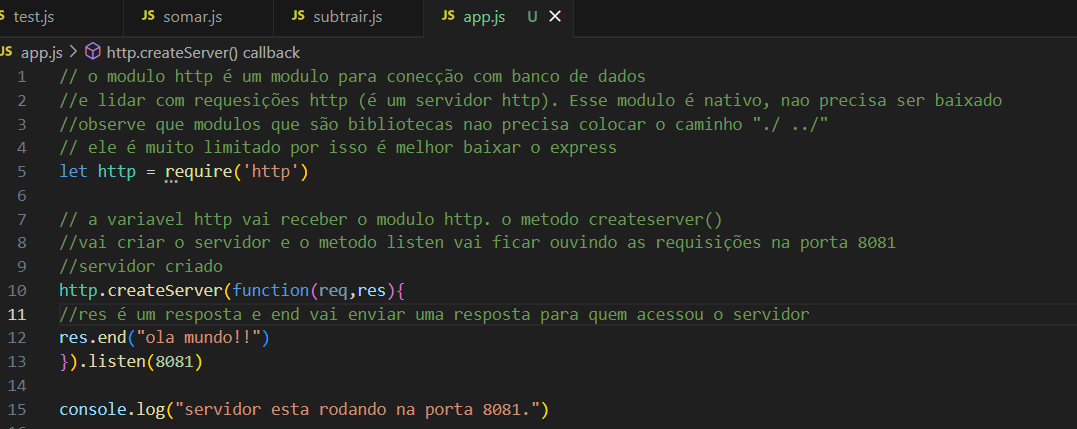
Criamos o arquivo subtrair.js e estamos exportando a função subtracao



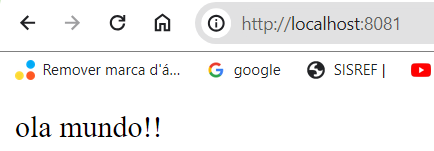
No arquivo test.js estamos importando a função soma e a função subtração por meio da função require e executamos as mesma neste arquivo.

Utilizando o modulo padrão do node para criar um servidor HTTP.

((porem vamos trabalhar com o Express para criar um servidor HTTP)



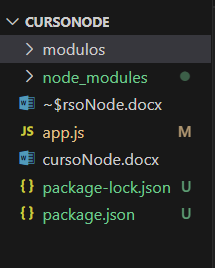
Levantando o servidor com o comando no terminal “node app.js” e indo no navegador, vamos ver



Criando um servidor utilizando o express

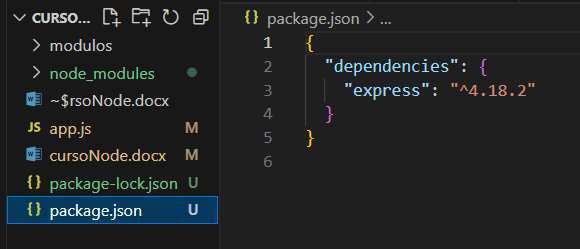
para instalar o express e qualquer outro modulo externo vc precisar entrar na pasta do seu projeto no prompet de comando e digitar o comando abaixo. Desta forma o express vai ficar salvo na pasta do projeto para ser utilizado. Npm é o gerenciador de pacotes do node.





Ao executar o comando foi criado os arquivos package.jon e package-lock.json e a pasta node\_modules.

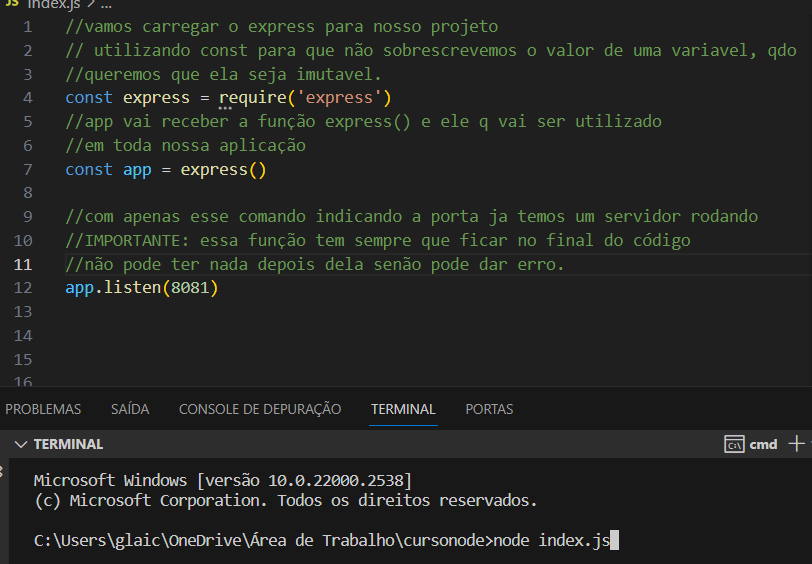
No arquivo package.jon vai ficar as dependências e scripts do nosso projeto

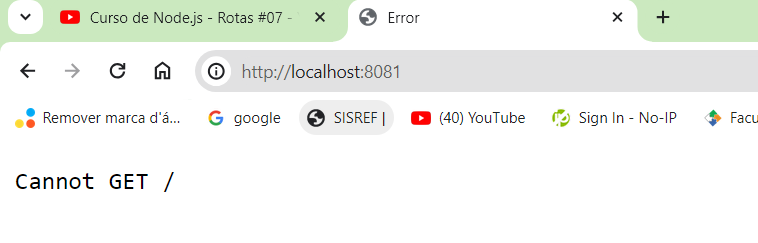


O arquivo package-lock.json vai conter as informações de todos os módulos instalados.

A pasta node\_modulos vai ficar arquivos e pastas das dependências instaladas. Essa pasta nós não enviamos para deploy pois é serar criada automaticamente no servidor em produção. vamos criar o arquivo gitginore e vamos colocar um código informando para a pasta node\_ modulo não ser incluída no versonamento.

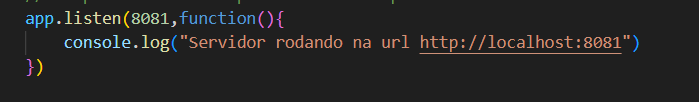






Aqui vemos que o servidor recebeu um get mas não soube processar pois ainda não foi criada a rota principal.

Adicionamos uma função de call-back para indicar que o servidor esta rodando na porta indicada. Indicando que o servidor esta ouvindo na porta 8081

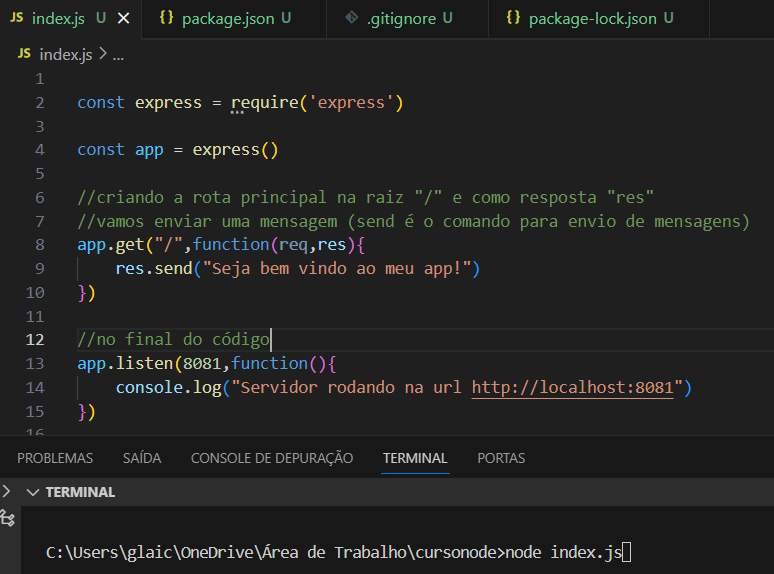


Vamos criar a primeira roda, a rota principal (raiz), que tem que ser um get, pois qdo entramos em um site o navegador envia um get para o site. As rotas tem que ficar acima de app.listen.

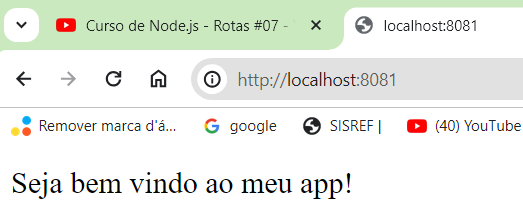
req representa a requisição enviada do lado do cliente, essa requisição pode conter parâmetros na url ou parâmetros e dados no seu corpo.

res representa a resposta que damos a essa requisição.

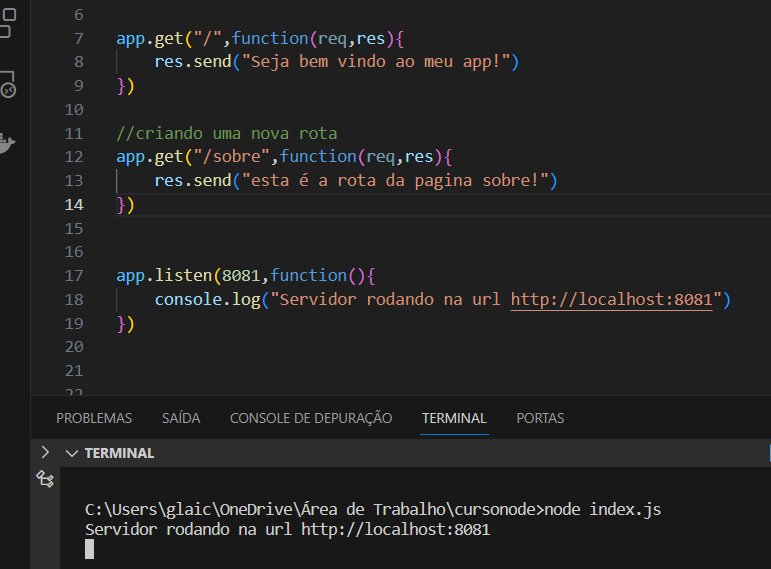
No terminal vamos levantar o servidor. Sempre que fazermos uma alteração no código vamos ter que derrubar (Ctrl + C) e levantar o servidor manualmente para que as mudanças sejam feitas.



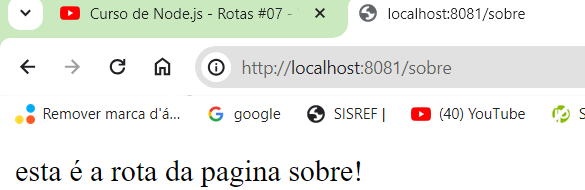
No navegador vamos ter uma resposta do servidor.



Criando um nova rota. Em uma rota podemos enviar (res) apenas uma mensagem (send), podemos enviar um arquivo, podemos abrir uma pagina html (sendFile).

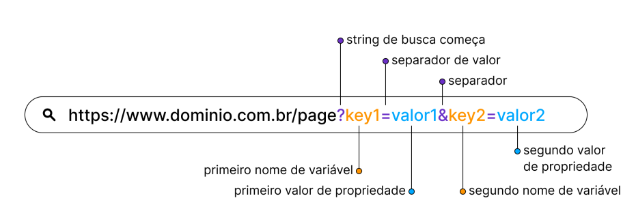


Digitando a rota no navegador

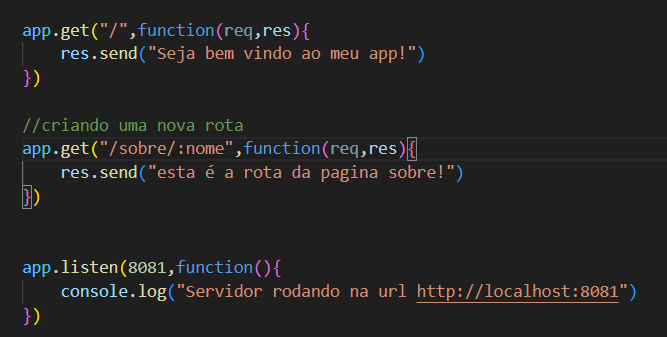


Criando rotas com parâmetros

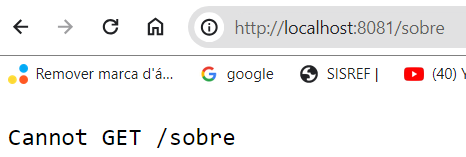
Parâmetros são dados que enviamos na URL. Utilizados pelo método get



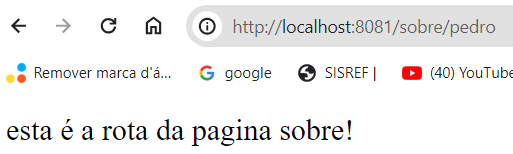
Veja na rota sobre colocamos um “/:” isso significa que vai ser um parâmetro depois dos dois pontos e em seguida o parâmetro “nome”.



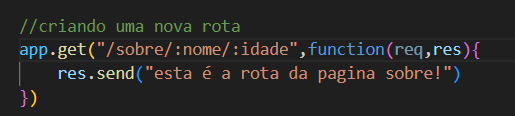
Veja que agora a roda sobre não foi encontrada, pois temos que indicar o parâmetro.



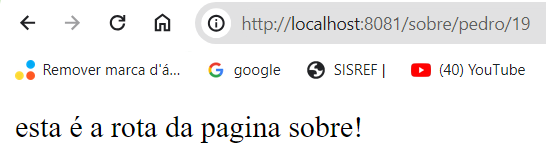
Incluindo o parâmetro na rota



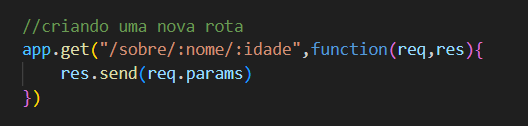
Podemos enviar mais parâmetros.



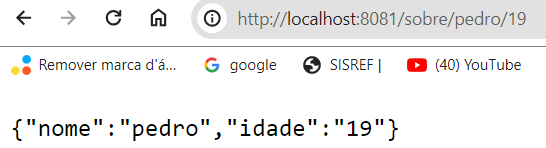
Devemos incluir todos os parâmetros na rota.



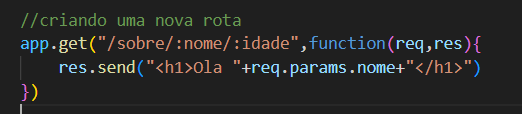
Podemos pegar os parâmetros da requisição através do req, com o comando req.params, e enviamos isso como resposta.

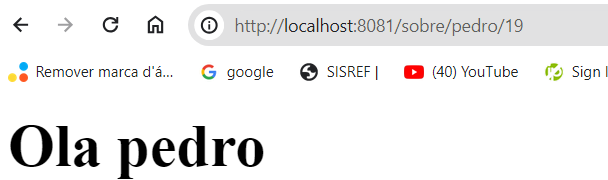


No navegar vai exibir os parâmetros em um json



Podemos pegar um parâmetro em especifico e trabalhar com ele. Inclusive podemos enviar tags html.





LEMBRETE: vc só pode fazer o res.send uma vez, o código abaixo esta errado.

