# Agilizando novos projetos com CLI

Introdução

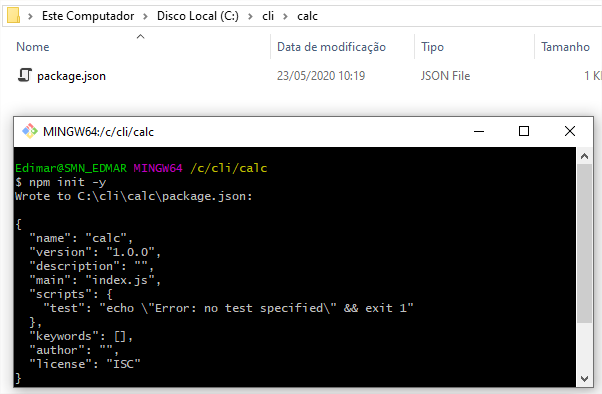
Neste post irei abordar como criar uma CLI (interface de linha de comando) em nodejs, o conteúdo será para iniciante no assunto, ela vai ser básica, porém servirá de base para entendermos alguns conceitos. No próximo post, irei criar uma outra CLI com uma utilidade mais aplicável, aguardem.

Breve explicação sobre CLI

CLI nada mais é do que uma aplicação (como outra qualquer) com a diferença em que ela é construída para rodar dentro de um terminal de linha de comando, ou seja, sem interface gráfica. Podemos pegar o exemplo da calculadora do Windows, que é uma aplicação feita para desktops, e nesse post vamos fazer uma para rodar somente através de linha de comando, a intenção é abrir o terminal e digitar “calculadora soma 1 3” e obter o resultado 4.

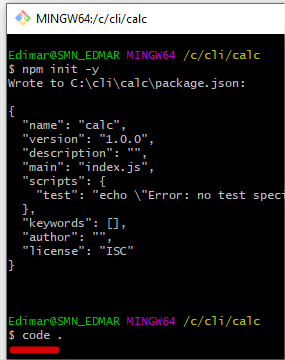
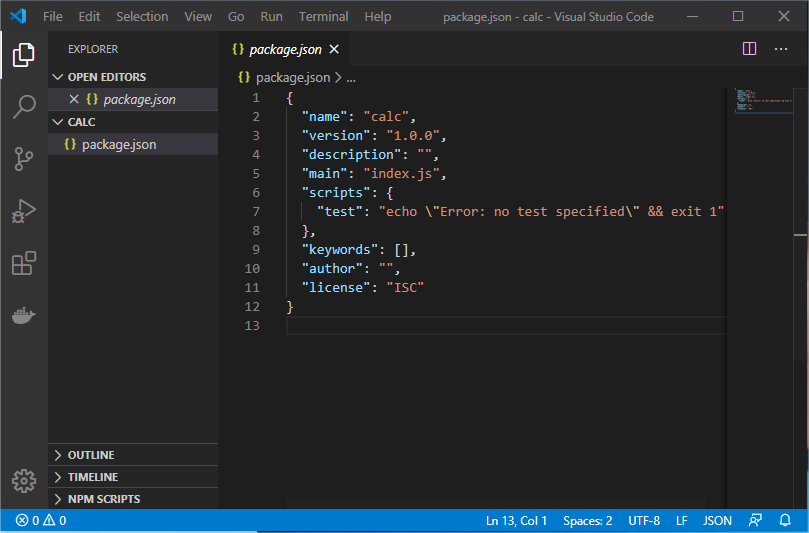
Iniciando a criação da CLI de calculadora

Primeiramente vamos criar a estrutura da nossa CLI, crie uma pasta (com nome e local que desejar) e rode o comando “npm init -y” para criar o arquivo package.json.



Observe que o comando “npm init -y” é uma CLI que cria pra gente o arquivo package.json automaticamente, agilizando nossa vida! Caso essa CLI não tivesse sido utilizada, teríamos que criar esse arquivo e todo seu conteúdo manualmente.

Abra a pasta criada com seu editor de código preferido, como vou utilizar o VS Code, é só executar no terminal o comando “code .” (observe que esse comando também é uma CLI, sua função é abrir o VS Code para a gente, e como passamos o parâmetro “.” (ponto), estamos instruindo a CLI para abrir o VS dentro do diretório atual que está sendo executado 😉)

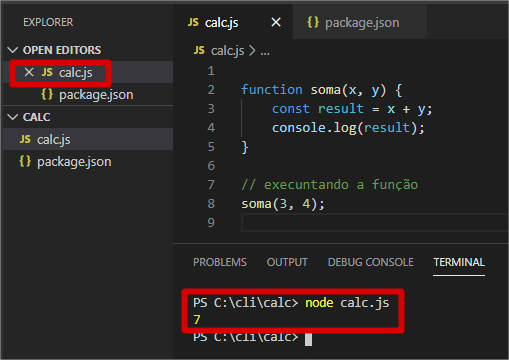
 

Reflexão

Até aqui, podemos observar a importância das CLI’s referente a agilidade na execução de tarefas, elas são poderosas aplicações e podem ser desenvolvidas para diversas utilidades, seja para criar arquivos de configuração “npm init -y”, para executar aplicações “code .”, executar nossa calculadora que estamos criando “calc soma 1 3”, enfim existe uma infinidade de aplicabilidade.

Continuando nossa CLI

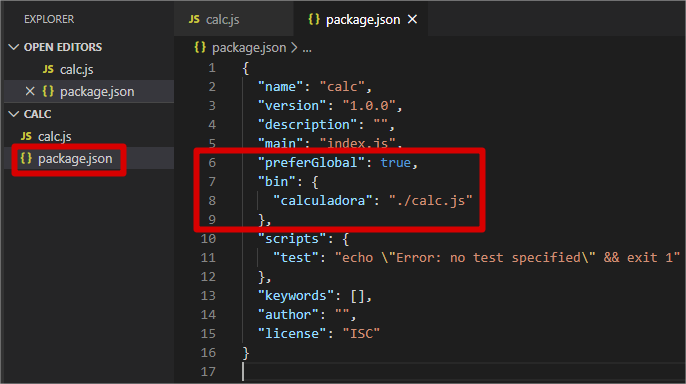
Vamos criar um arquivo “calc.js”, à princípio terá somente com a função “soma”, para testarmos podemos executar o arquivo com o comando “node calc.js”, e veremos o resultado 7, com isso a função principal e única da nossa calculadora está pronta.



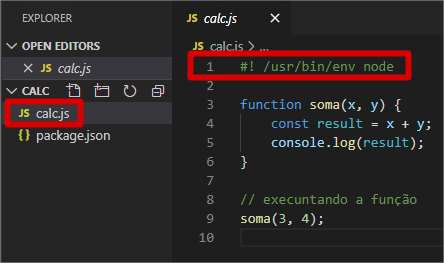
O próximo passo é alteramos a chamada e configurações da nossa aplicação, na imagem acima estamos executando o arquivo calc.js com o comando “node calc.js”, entretanto, como eu disse lá no inicio do post, a intenção é digitar o comando “calculadora soma 1 3”, onde “calculadora” será o nome da nossa CLI, “soma” será a action (ação) que será disparada e “1” e “3” será os parâmetros enviados para a ação.

Alterando o package.json

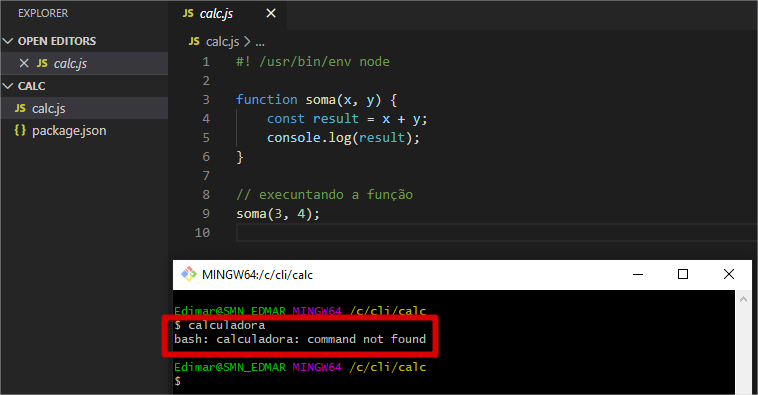
Vamos incluir os dois comandos abaixo no nosso arquivo package.json, o primeiro comando é para instruir a instalação da nossa aplicação de forma global, e o segundo é para direcionar o comando “calculadora” (que será executado no terminal) para nosso arquivo “calc.js”.



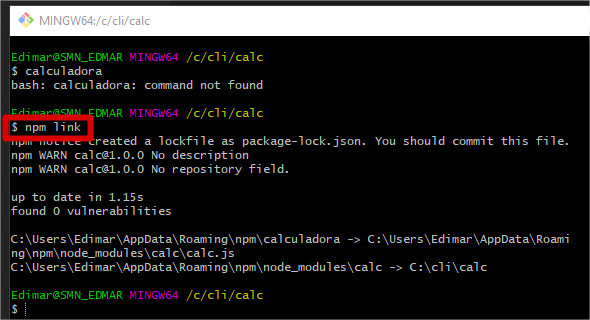
O próximo passo é alterar nosso arquivo “calc.js” para instrui-lo a ser executado com o node, para isso é só inserir na linha 1 o comando destacado abaixo.



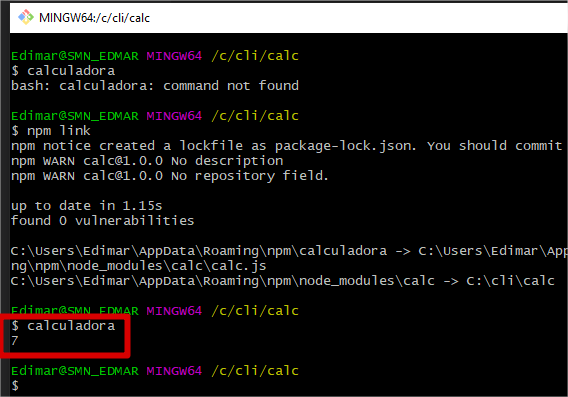
Nesse ponto, se já tentarmos rodar nossa CLI ainda não dará certo, pois ainda não a instalamos na nossa máquina, aparecerá uma mensagem parecida com a abaixo:



Para instalar nossa CLI, vamos utilizar o comando “npm link”, com isso nossa aplicação estará instalada globalmente na nossa máquina.

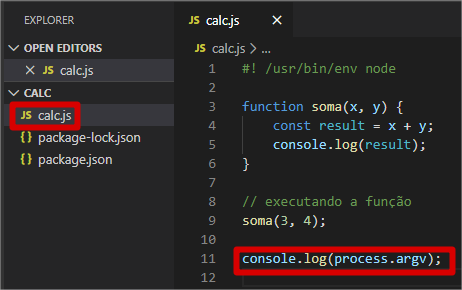


Agora sim, digitando o comando “calculadora” (definido no package.json) nossa CLI já será executada exibindo o resultado 7 (do cálculo da soma) 😊.

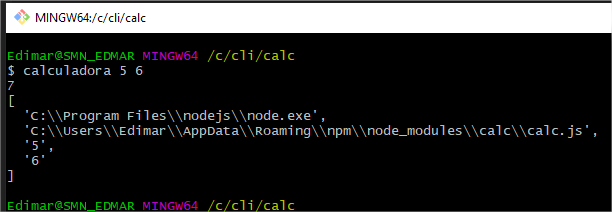


Passando os parâmetros

Se executarmos nosso comando “calculadora” passando os argumentos 5 6, exemplo: “calculadora 5 6” podemos recuperar esse argumentos dentro do nosso arquivo “calc.js” com o “process.argv”, vamos alterar nosso arquivo (incluindo a linha 11 da imagem abaixo) para exibir o conteúdo dessa variável que é uma matriz.

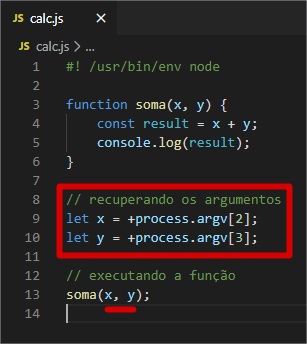


Após incluído a linha acima, execute novamente a CLI passando os dois argumentos “5” e “6” (conforme imagem abaixo) para vermos o conteúdo na matriz:



Observe que a matriz possui 4 itens, onde os dois primeiros se referem a forma de como o script foi executado e o caminho do arquivo que foi executado, e os outros dois são os argumentos que passamos via terminal com o comando “calculadora 5 6”. Pronto agora já aprendemos como capturar os argumentos enviados 😉

Agora que já sabemos como capturar os argumentos, vamos alterar nosso arquivo “calc.js” para capturar os argumentos e passa-los para função que realiza a soma, conforme imagem abaixo:



Observe que tive que utilizar o operador “+” (nas linhas 9 e 10) para converter os argumentos para números, pois como vimos, a matriz “argv” armazena somente string, se você não converter os argumentos, o resultado da soma abaixo seria “56” (pois a função soma iria fazer uma concatenação ao em vez de 11 que seria a soma) 😊

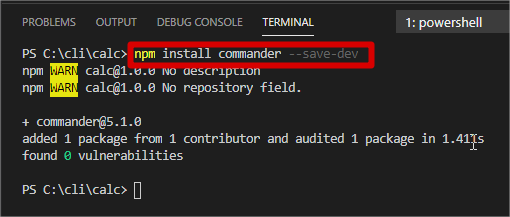


Com isso nossa CLI está quase pronta. Lembra que nossa intenção é digitar o comando “calculadora soma 1 3”, então, está faltando o “soma”, que é a nossa ação, não dá pra chamar apenas “calculadora 1 3”, pois o usuário que utilizar essa CLI não saberá que será realizado uma soma, e também se quiséssemos implementar outras ações como subtração, multiplicação, divisão, etc. seria necessário incluir no comando qual ação o usuário deseja, para isso vamos utilizar o Commander.

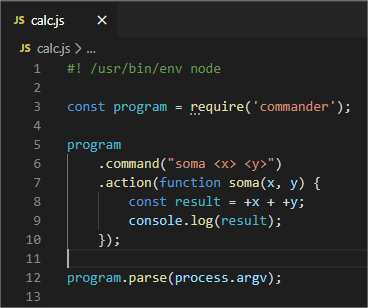
Implementando ações com o Commander

O comander basicamente é um pacote disponível para baixar no npm, com ele nos conseguiremos enviar nossa ação de soma e escuta-la dentro do nosso arquivo “calc.js”, então vamos lá.

Primeiramente vamos instalar o pacote do commander usando o npm



Agora vamos alterar nosso arquivo “calc.js” para ficar igual a imagem abaixo:



* Linha 3: estamos importando o ‘commander’ que acabamos de instalar
* Linha 6: estamos definhando um comando para escutar nossa ação, observe que na string passada como argumento estamos dizendo qual é o comando e os argumentos “x” e “y” que são obrigatórios por estarem envolvidos por “<>”, caso eles fosse opcionais deveriam estar envolvidos por “[]”
* Linha 7: estamos definindo uma ação a ser tomada quando o usuário executar o comando da linha 6, a ação é justamente nossa função “soma” que foi transferida para dentro da “action”.
* Linha 8: tivemos que alterar a função para converter as variáveis “x” e “y” (colocando o operador “+” antes de cada uma)
* Linha 12: essa linha é necessária para que o “commander” consiga ler os argumentos que o usuário que enviar.

Agora se você executar o comando “calculadora soma 1 3” terá o resultado “4” assim como esperávamos.



Com isso chegamos ao fim da criação da nossa CLI. 👏👏👏

Publicando no NPM

Para que outras pessoas possa utilizar nossa CLI vamos publicada no NPM, primeiramente temos que alterar o nome da nossa CLI (localizado na propriedade “name” do arquivo “package.json”) para um nome disponível no npm, para isso podemos procurar em https://www.npmjs.com/nome\_da\_sua\_cli, se ainda não existir nenhum pacote com esse nome, então podemos seguir com ele.

Com o nome definido, agora é só seguir os passos abaixo:

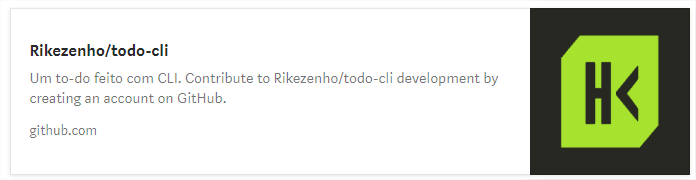
* Criar uma conta no NPM site atual https://www.npmjs.com/
* Executar no terminal npm login e fazer o login com o usuário e senha criado
* Executar no terminal (dentro da pasta da nossa CLI) o comando npm publish

Pronto nossa CLI agora está disponível para qualquer pessoa que queira utilizar, para testarmos podemos voltar ao site do npm e verificar que nosso pacote já estará visível juntamente com o comando para instala-lo.

Finalizando

Com isso finalizamos esse post, espero ter ajudado com algum conhecimento.

O conteúdo da nossa aplicação está no repositório abaixo



Qualquer dúvida ou sugestão, favor comentar abaixo.

Até a próxima, valeu galera!