Node CLI! Ame-as ou Deixe-as.

Olá!

Eu sou Beatriz Herculano

E-mail: biaher.oliveira@gmail.com

Agenda

- O que é uma CLI
- Como uma CLI se comporta
- Padrões de uma CLI
- UX em uma CLI
- Por que usar node?
- O passo a passo do projeto
- Tratamento de erros
- Publicar no NPM

1

O que é uma CLI?

Uma interface! Onde você dá comandos por meio de texto e o programa executa uma ação:

Command Line Interface.

Como uma CLI app se comporta?

- Command
- Options
- Arguments
- Inputs
- Outputs

Minha-cli comando --option “argumento aqui” mais-coisas

dotnet new console -o myApp

Padrões de CLIs

- Filter---------- input–> do-something –> output
- Cantrip--------- do-something
- Source --------- generate-data –> stdout
- Sink ---------- input–> process-data
- Compiler ------- file –> process-contents –> transformed-file

UX?

- --help | -h | help
- --version | -v | version
- Menus informativos no help.
- Dicas
- Informações visuais

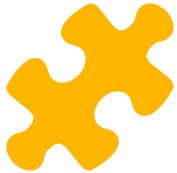
Libs e mais Libs

O melhor do nosso mundo

O que nos vamos usar?

minimist	chalk
Modulo para os argumentos da CLI	Colorir os logs no terminal
cli-progress	ora
Barras de progresso	Spinner

O que você pode usar?



- Fs
- Axios
- Path
- Regex
- Banco de dados

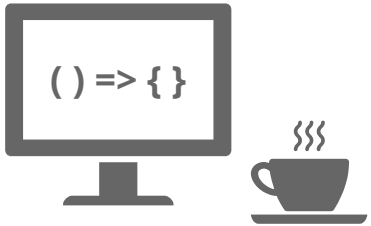
O que você pode fazer?

Consultar APIs

Trabalhar facilmente com arquivos JSON, xml

scaffolding

O projeto



Ou mais ou menos isso

Passo a passo

Criar uma bin/robo-tdc

```
#!/usr/bin/env node
require('..')()
```

Criar uma index.js no root do projeto

```
module.exports = () => {
      console.log(“Minha CLI”);
}
```

Passo a passo

Modifica o package.json

```
"bin": {
   "robo-tdc": "bin/robo-tdc"
 }
```

Criar o simlink com o npm

```
chmod +x bin/robo-tdc
npm link
```

Passo a passo

Captura os comandos

```
const args = minimist(process.argv.slice(2))
let cmd = args._[0] || 'help'
```

Para cada comando uma ação

```
switch (cmd) {
      case 'cmd':
            require('./cmds/MEU-CMD')(args)
            break
}
```

Captura os comandos

```
switch (cmd) {
        case 'version':
            require('./cmds/version.js')(args)
            break

        case 'help':
            require('./cmds/help')(args)
            break

        default:
            console.log(`"${cmd}" is not a valid command!`)
            break
    }
```

cmds/help.js

```
module.exports = (args) => {
      console.log(
      `
            robo-tdc [command] <options>

            version ..... mostra a versão do pacote
            help ..... mostra o menu de ajuda para um
      comando`)
}
```

cmds/version.js

```
const { version } = require('../package.json')

module.exports = (args) => {
   console.log(`v${version}`)
}
```

cmds/alarme.js

```
const chalk = require('chalk');
module.exports = (args) => {
    let hora = args.hora
    if (hora) {
        console.log(`Alarme criado para ${chalk.green(hora)}`)
    } else {
        console.log(`Alarme criado para esse mesmo horário para
amanhã`);
    }
}
```

cmds/cronome

```
const chalk = require('chalk');
const error = require('../utils/error')
module.exports = async (args) => {
    let tempo = args.tempo || args.t
    if (tempo) {
        if (+tempo) {
            tempo = +tempo
        } else {
            error("Tempo informado invalido, informe a quantidade de segundos para o cronômetro", true);
        }
    } else {
        console.log(`Tempo não informado, utilizando 10 segundos por padrão`);
        tempo = 10;
    }
    for (i = 1; i <= tempo; i++) {
        await new Promise(resolve => setTimeout(resolve, 1000))
        console.log(chalk.green(i));
    }
}
```

cmds/bolo.js

```
const ora = require('ora')
const cliProgress = require('cli-progress');
module.exports = async (args) => {
console.log("Estou preparando a massa")
   const spinner = ora().start()
   await new Promise(resolve => setTimeout(resolve, 3000))
   spinner.stop();
   console.log('O bolo está indo para o forno');
   const bar = new cliProgress.Bar({
       format: 'assando [{bar}] {percentage}%'
   });
   bar.start(100, 0);
   await new Promise(resolve => setTimeout(resolve, 3000))
   bar.update(20);
// … espera um tempinho e continua atualizando a progress bar ate ela
// acabar
let bolo = ` desenho de bolo em ASCII`
   console.log(bolo);
}
```

index.js

```
const args =
minimist(process.argv.slice(2))
    let cmd = args._[0] || 'help'

    if (args.version || args.v) {
        cmd = 'version'
    }

    if (args.help || args.h) {
        cmd = 'help'
    }
```

```
switch (cmd) {
        case 'alarme':
            require('./cmds/alarme')(args)
            break
        case 'cronometro':
            require('./cmds/cronometro')(args)
            Break
       case 'bolo':
            await require('./cmds/bolo')(args)
            break
        case 'version':
            require('./cmds/version.js')(args)
            break
        case 'help':
            require('./cmds/help')(args)
            break
        default:
            error(`"${cmd}" is not a valid command!`,
true)
            break
    }
```

cmds/help.js

```
module.exports = (args) => {
   const subCmd = args._[0] === 'help'
       ? args._[1]
       : args._[0]
   console.log(menus[subCmd] || menus.main)
}
```

```
const menus = {
   main: `
     //Descrição anterior
      alarme .............. cria um alarme para amanhã no
mesmo horario
     cronometro .......... inicia um cronometro
     bolo ................ prepara um bolo bem bonito`,
   alarme: `
   robo-tdc alarme <options>
   --hora ..... a hora para criar o alarme`,
   cronometro: `
   robo-tdc cronometro <options>
   --tempo ..... o tempo para o cronometro em segundos
       Se nenhum tempo foi passado é usado o tempo de 10
segundos`,
   bolo: `
   robo-tdc bolo
   faz a massa e assa um bolo bem bonito`
}
```

O que fazer quando eu tiver **ERROS** ?

E como sempre, a solução é sair correndo e dar uma mensagem de erro

Sucesso = 0

Deu ruim = 1

1	Catchall for general errors
2	Misuse of shell builtins (according to Bash documentation)
126	Command invoked cannot execute
127	"command not found"
128	Invalid argument to exit
128+n	Fatal error signal "n"
130	Script terminated by Control-C
255*	Exit status out of range

utils/error.js

```
const chalk = require('chalk')
module.exports = (message, exit) => {
    console.error(chalk.red(message))
    exit && process.exit(1)
}
```

Publicar no npm

- .npmignore || .gitignore?
- Package.json.
  - Descrição
  - Keywords
  - Autor
  - Licença
  - Engines
  - Repositorio

Publicar no npm

```
npm login
npm publish
```

Referencias

https://timber.io/blog/creating-a-real-world-cli-app-with-node/

https://cli-guide.readthedocs.io/en/latest/design/patterns.html

http://tldp.org/LDP/abs/html/exitcodes.html

Obrigada!

Perguntas?

@BiaherOliveira & biaher.oliveira@gmail.com &
BeatrizHerculano