



# GopherCon Brasil

*Go for everybody.*

<09/05/2024>

# Go CLIs além do óbvio

<Matheus Mina>

<Staff Software Engineer @ PicPay>

# CONTEÚDO

CLIs?	01
Princípios	02
Go for Go	03
Links úteis	04
Perguntas	05



# sobre mim



## SOBRE MIM

Pode me chamar de Mina!

Mineiro de Varginha, trabalhando com tecnologia há mais de 10 anos! No tempo livre cuido de cachorro e pratico alguma luta!

<SEÇÃO 01>

---

# Por quê estamos falando de CLIs?

Yesterday 10:07 PM

Bro I have a very important question  
for you

Is Command Line Interface (CLI)  
pronounced C - L - I or "Klee"

It's klee

<CLI?>

---

CLI == Command Line Interface

<IMPORTÂNCIA>

---

CLIs ajudam os devs em funções no dia a dia, abstraindo funcionalidades e complexidades.



## &lt;EXEMPLOS&gt;

Talvez você utilize...

1. git & github;
2. aws-cli & kubectl & kubectl;
3. Go

<IMPORTÂNCIA>

---

Trabalhando em equipes de plataforma, já otimizamos diversas coisas através de CLIs para os nossos devs!

<SEÇÃO 02>

---

# Quais os princípios para criar uma boa CLI?

<HUMAN-FIRST DESIGN>

---

Foque nos humanos e em como eles vão interagir com o seu programa.

## <COMPOSIÇÃO>

---

Seu programa vai ser utilizado de maneiras inesperadas, então ele deve ser simples para se integrar em outros sistemas.

## <CONSISTÊNCIA>

---

Seja consistente em sua CLI.

Use convenções existentes, como as da UNIX, POSIX e etc.

## <COMUNIQUE O SUFICIENTE>

---

Pouca comunicação pode fazer o usuário achar que o programa travou, muita comunicação pode poluir a verdadeira resposta.

<DESCOBERTA FACILITADA>

---

Deve ser fácil e simples entender e descobrir como seu programa funciona.



## <CONVERSAÇÃO COMO NORMA>

---

A interação com CLIs é uma forma de conversação. Entender a forma que você deseja manter essa interação te ajuda a criar uma CLI melhor.

<ROBUSTEZ>

---

O seu programa deve lidar com o inesperado da melhor forma.

<CAOS>

---

Todo mundo vai quebrar algum princípio. Devemos entender qual estamos quebrando e o porquê de quebrar.

<SEÇÃO 03>

---

# Go for Go: construindo sua CLI

## <FERRAMENTAS>

---

A comunidade e a linguagem nos oferece diversas de ferramentas para ajudar a construir nossa CLI.



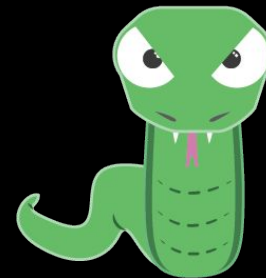
## GoReleaser

Gera binários em Go para  
diversas plataformas



## Cobra

Framework para CLIs  
modernos em Go



## Viper

Solução completa para  
configurações em Go

## <AJUDA & DOCUMENTAÇÃO>

---

Deve ser simples pro seu usuário saber como usar sua ferramenta ou descobrir como fazer algo novo.

## <AJUDA & DOCUMENTAÇÃO>

---

Os textos devem ser concisos,  
mas explicativos.

Se possível, dê exemplos!



01

Cobra nos fornece automaticamente um menu de ajuda, autocomplete e sugestão em casos de erro.

02

A biblioteca godoc cria documentação por código:  
<https://pkg.go.dev/golang.org/x/tools/cmd/godoc>

03

Quanto mais fontes de documentação, melhor pro usuário!  
Mas mais complexo é de se manter atualizado.

## &lt;COMANDOS&gt;

```
rootCmd := &cobra.Command{
    Use:     "mycmd",
    Short:   "I love CLIs",
    Long:    "Very Long desc about CLIs",
    Run:     func(cmd *cobra.Command,
args []string) {
        // do something...
    }}
}
```

Cria um  
comando  
principal para  
sua CLI

## <SUBCOMANDOS>

---

```
cmd := &cobra.Command{  
    Use: "dosomething",  
    Run: func(...) {}  
}
```

```
rootCmd.AddCommand(cmd)
```

Cria um  
subcomando.

<ARGS x FLAGS>

---

```
go test ./... -v
```

<ARGS x FLAGS>

---

```
go test file1.go file2.go
```

!=

```
go test file2.go file1.go
```

<ARGS x FLAGS>

---

```
go test . -v -f=o.json
```

==

```
go test . -f=o.json -v
```

## &lt;ARGS x FLAGS&gt;

```
f := ""

cmd := &cobra.Command{
    Use: "dosomething",
    Args: cobra.MinimumNArgs(1),
    Run: func(...) {}
}

cmd.Flags().StringVarP(&f, "full",
    "f", "short", "long desc")
```

Prefira flags ao invés de argumentos.

01

Se for fazer algo perigoso ou destrutivo, peça confirmação do usuário!

02

Faça o padrão ser a coisa certa para a maioria dos casos de uso.

03

Não leia segredos ou dados sensíveis através de flags ou argumentos.



<CONFIG x ENV VARS x FLAGS>

---

config file != env vars != flags

## &lt;CONFIG x ENV VARS x FLAGS&gt;

```
viper.SetDefault("ENV", "QA")
viperAutomaticEnv()

err := viper.ReadInConfig()
if err != nil {
    return fmt.Errorf("error reading
config file: %w", err)
}
```

Ordem de configuração:

1. Flags
2. Shell's env vars
3. Project Config
4. User config
5. System config

## <INPUT & OUTPUT>

---

Lembre-se da conversação como norma e do foco na interação humana!

01

Charm libs te ajudam a criar CLIs interativas.  
<https://charm.sh/>

02

Utilize barras de progresso em operações longas.  
<https://github.com/schollz/progressbar>

03

Use flags para especificar o formato da resposta para máquina, se necessário.

## &lt;ERROS&gt;

```
x, err := doSomething()

if err != nil {
    return fmt.Errorf("human
readable err: %w", err)
}
```

Usuários devem ser capazes de entender os erros.

## &lt;ERROS&gt;

```
func main() {
    defer recoverPanic()
    cmd.Execute()
}

func panicRecover() {
    err := recover()
    if err != nil {
        fmt.Println("error!")
    }
}
```

Não exploda  
com erros  
desconhecidos.

## &lt;ERROS&gt;

```
if viper.Get("ENABLE_SENTRY") {
    sentry.Init(sentry.ClientOptions{
        Dsn: "mydsn",
    })
}

err := cmd.Execute()
if err != nil {
    sentry.CaptureException(err)
}
```

Capture os erros  
com alguma  
ferramenta!

## <PERFORMANCE>

---

Performance é interessante, mas sua ferramenta pode ser utilizada em ambientes com recursos limitados!



## <PERFORMANCE>

---

Se for usar concorrência, alguns padrões podem te ajudar!

Fiquem de olho na palestra do Cássio Botaro!

## &lt;PROFILING&gt;

```
$ go test -bench=. -benchmem  
-cpuprofile c.out -memprofile  
m.out  
  
goos: darwin  
goarch: arm64  
pkg: github.com/mfbmina/foo  
BenchmarkFib10-8 6900780 168.8  
ns/op  
PASS  
ok github.com/mfbmina/foo 1.617s
```

É possível gerar  
um profiles  
através de  
testes de  
benchmark.

## <PROFILING>

---


Recomendo assistir a palestra do Alex Rios sobre PGO para entender como utilizar esses profiles!

<SEÇÃO 04>

---


# Links úteis

<https://cliq.dev>



Excelente guia para  
construir sua CLI,  
independente de  
linguagem.

Post by  
Adam Czapski



Post que estrutura  
bem alguns conceitos  
de uma boa CLI.

Thread by  
@thewizardlucas





Thread no Twitter/X  
bem detalhada sobre  
padrões de UX para  
CLIs.



# CONTATO

CONECTE-SE COMIGO!

[mfbmina.dev](https://mfbmina.dev) 

[linkedin.com/in/mfbmina/](https://linkedin.com/in/mfbmina/) 

[x.com/mfbmina](https://x.com/mfbmina) 



# Muito obrigado!

Perguntas?