DevFest

Sul de Minas

Go CLIs além do óbvio

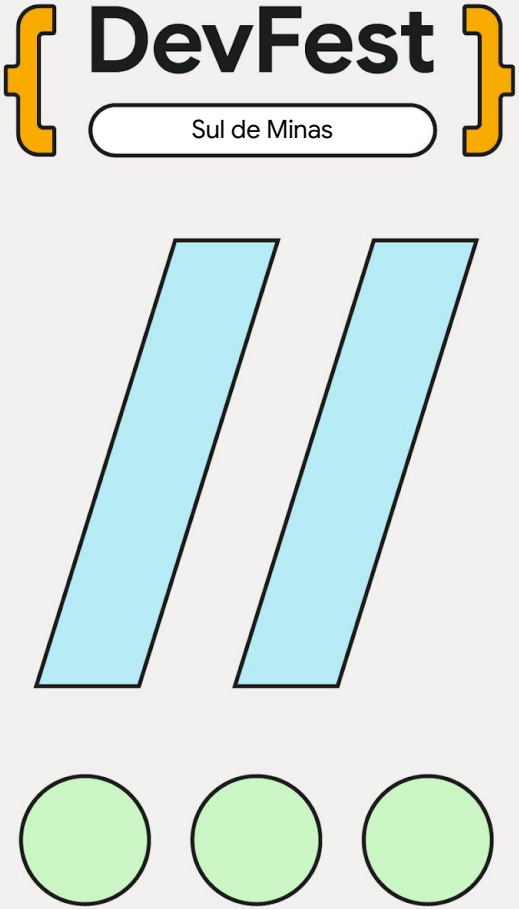
Matheus Mina
Staff Software Engineer @ PicPay

Google Developer Groups

Mineiro de Varginha, MG.

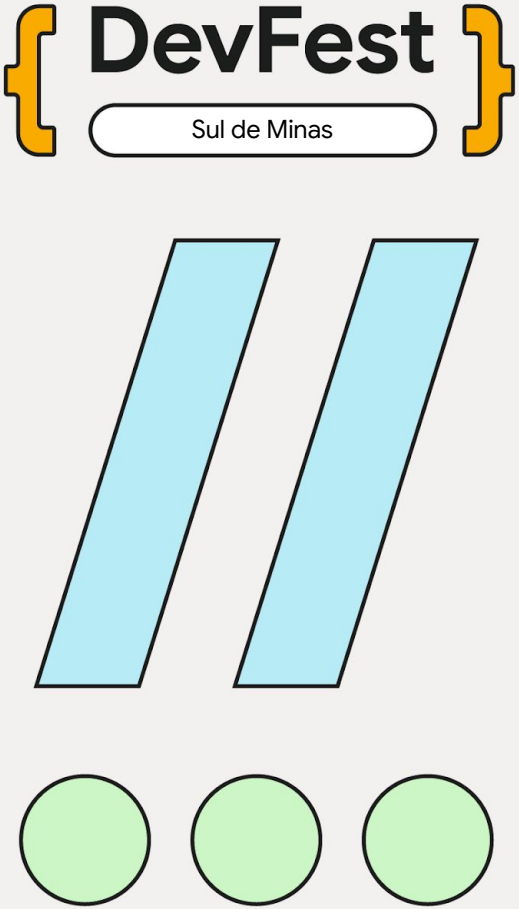
Staff Software Engineer e Tech Lead @ PicPay

10+ anos na área



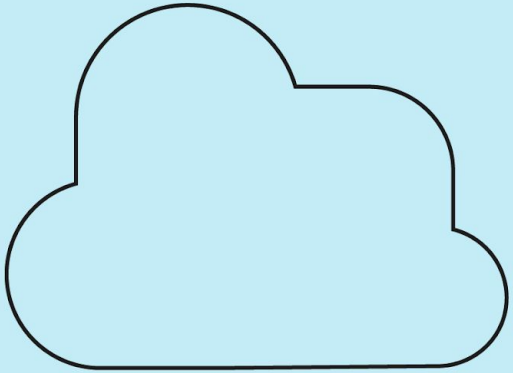
Agenda

- CLIs?
- Princípios
- Go for Go
- Referências
- Perguntas



DevFest

Sul de Minas



Por que estamos falando de CLIs?

Em pleno 2025, quase 2026...

Google Developer Groups

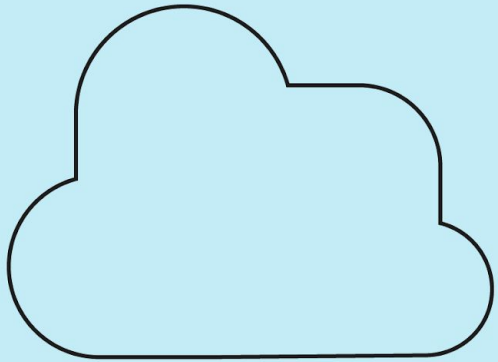
CLI == Command Line Interface

Talvez você **utilize** alguma dessas...

git, github, aws-cli, kubectl, kubectx, go, python, pip3, gemini-cli...

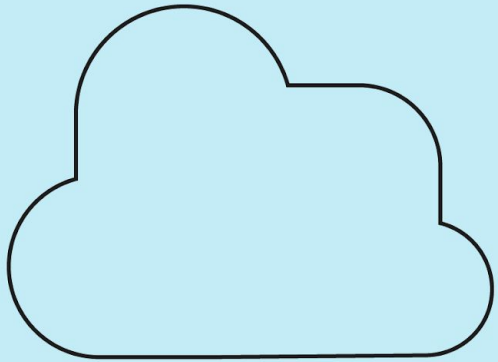
DevFest

Sul de Minas



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

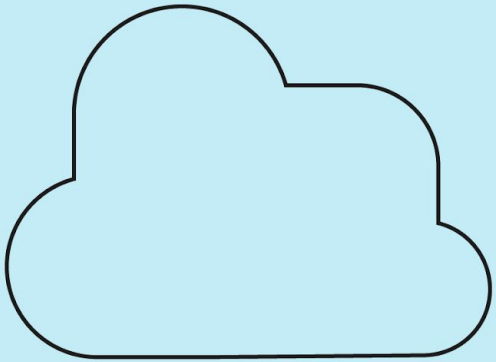
DevFest
Sul de Minas

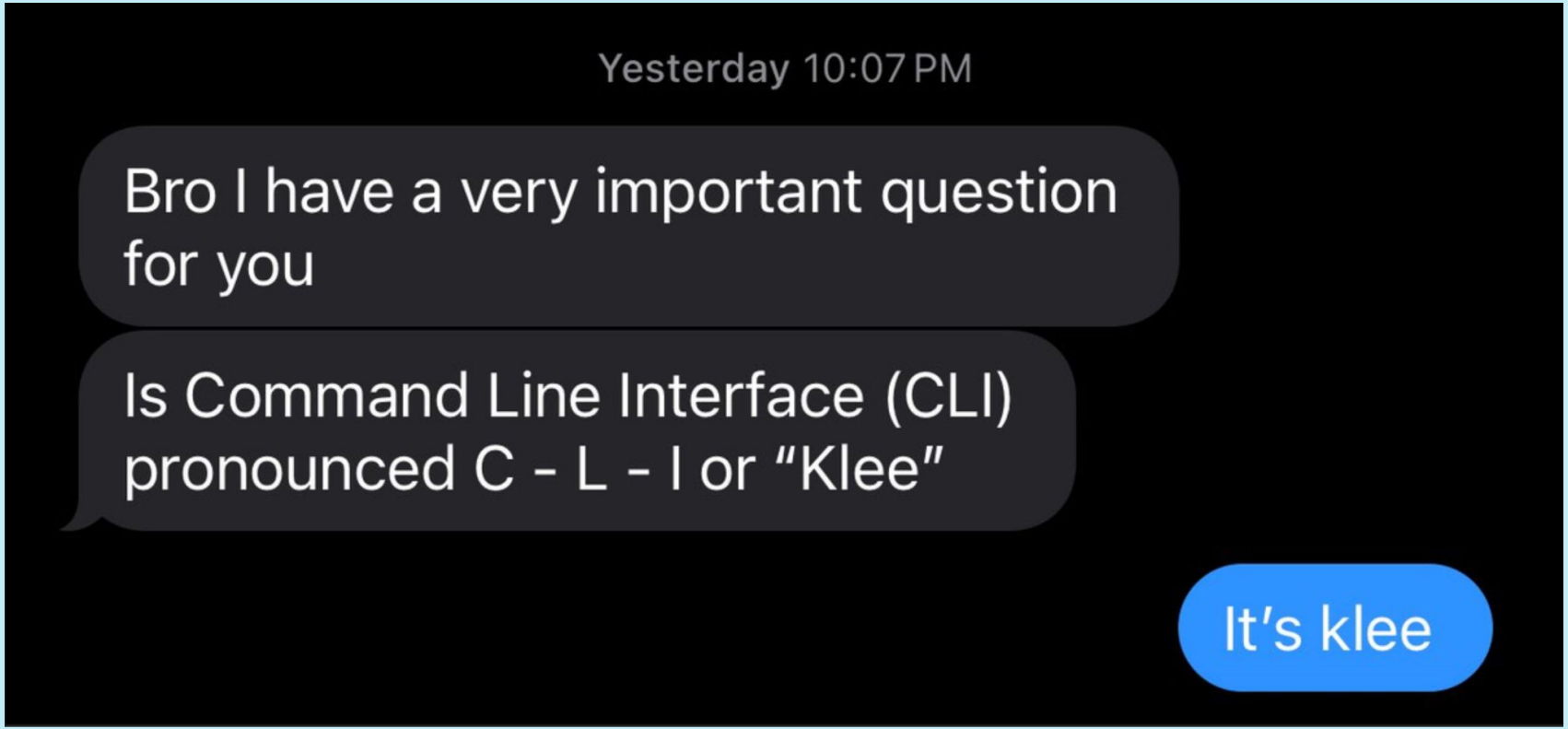


Equipes de plataforma **otimizam** coisas por meio de CLIs para os nossos devs!

DevFest

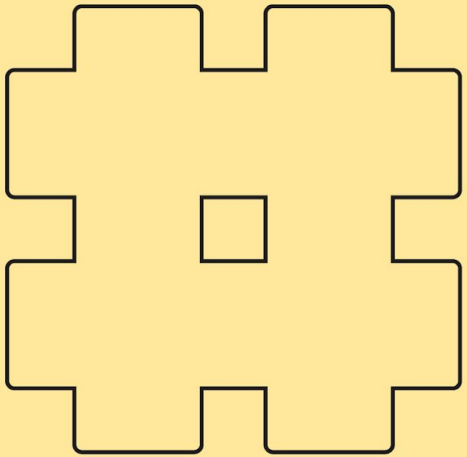
Sul de Minas




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

DevFest

Sul de Minas



Quais os princípios para criar uma boa CLI?

Já que os princípios de uma web app já são bem difundidos...

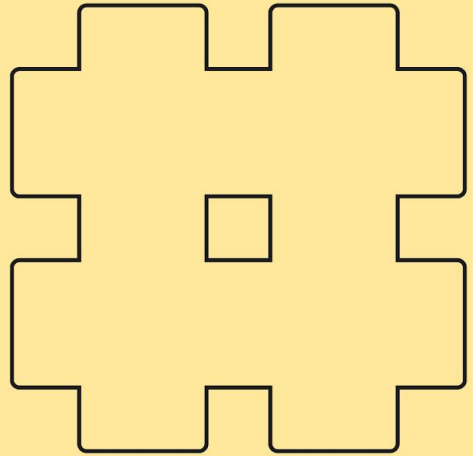
Google Developer Groups

DevFest

Sul de Minas

Princípio do Human-First

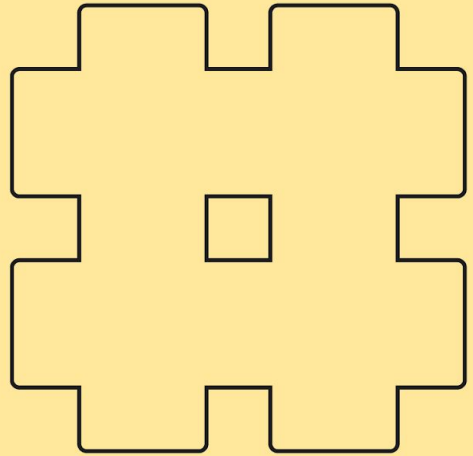
Foque nos humanos e em como eles vão interagir com o seu programa.



DevFest
Sul de Minas

Princípio da Descoberta

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

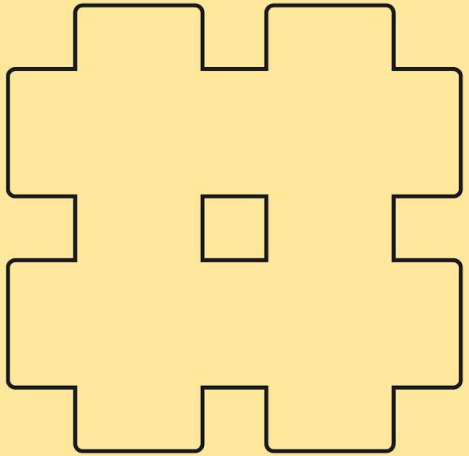


Princípio do Conversação

A interação com CLIs é uma forma de conversação.

Entender como você deseja manter essa interação te ajuda a criar uma CLI melhor.

DevFest
Sul de Minas

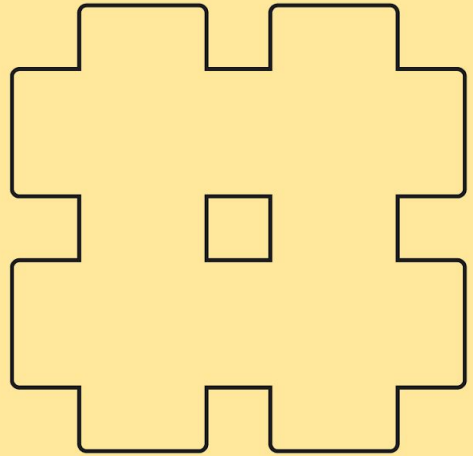


Princípio da Comunicação

Pouca comunicação faz você achar que o programa travou.

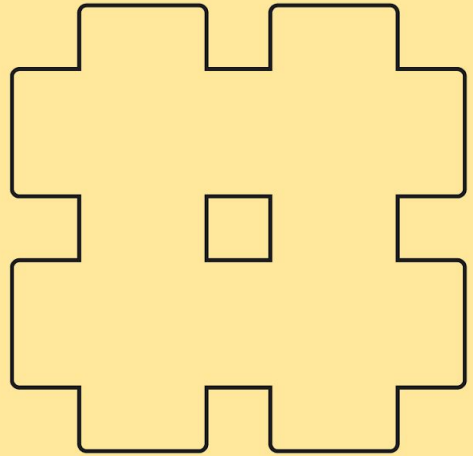
Muita comunicação pode poluir a verdadeira resposta.

DevFest
Sul de Minas



Princípio da Composição

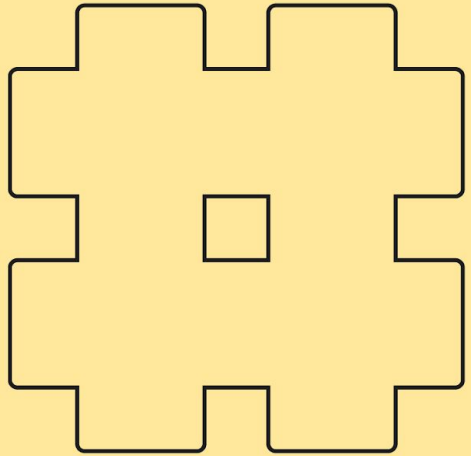
Seu programa vai ser utilizado de maneiras inesperadas, então ele deve ser simples para se integrar em outros sistemas.



DevFest
Sul de Minas

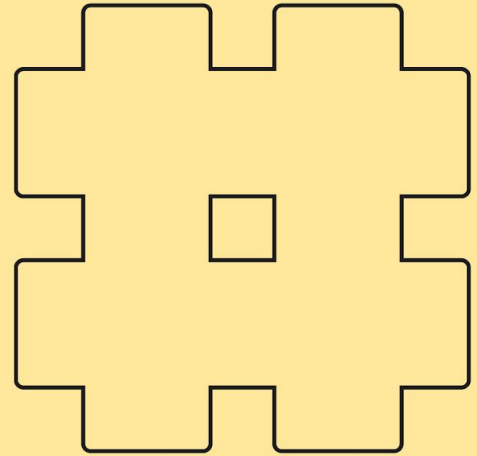
Princípio da Consistência

Use convenções existentes, como as da UNIX, POSIX, etc. Seja consistente em sua CLI.



Princípio da Robustez

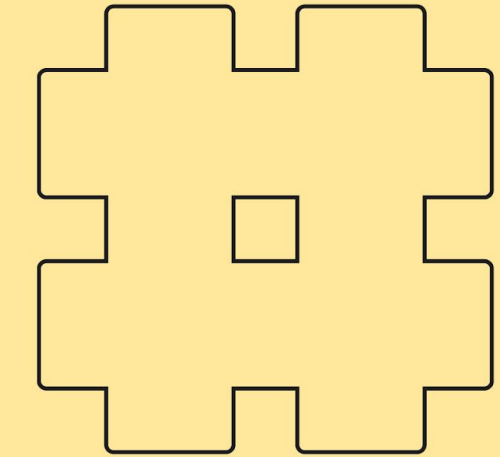
O seu programa deve lidar com o inesperado da melhor forma.



Princípio do Caos

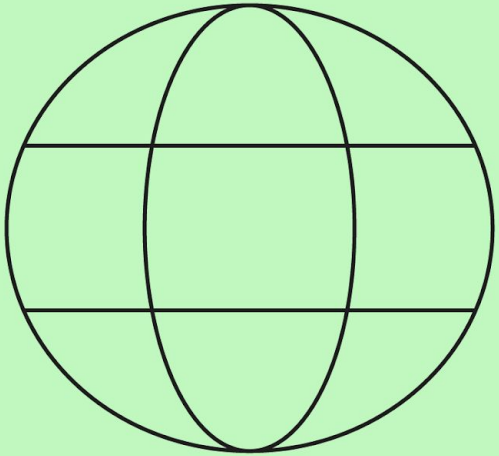
Todo mundo vai quebrar algum princípio!

Devemos entender qual estamos quebrando e o porquê de quebrar.



DevFest

Sul de Minas



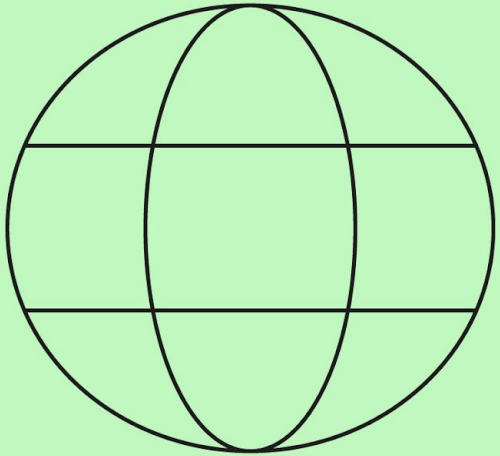
Go for Go!

Construindo sua CLI da melhor forma possível...

Google Developer Groups

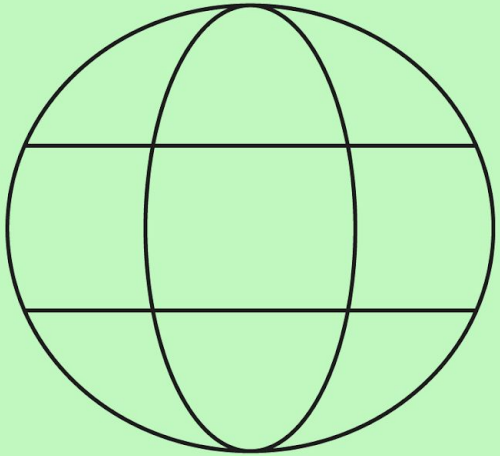
A **comunidade** e a **linguagem** nos oferecem diversas de ferramentas para construir nossa CLI.

DevFest
Sul de Minas



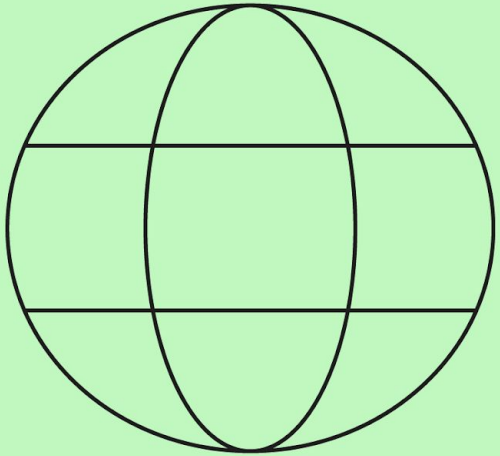
GoReleaser gera artefatos para diversas plataformas e distribuições.

DevFest
Sul de Minas



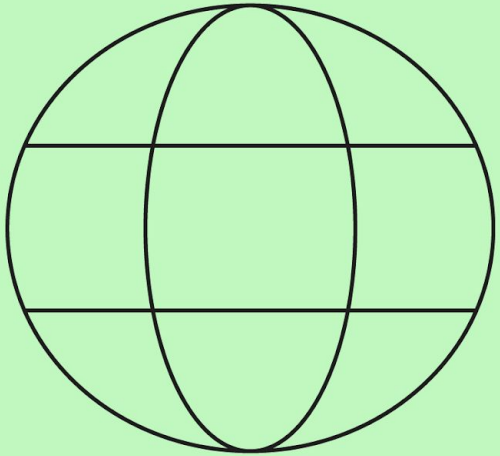
Viper é uma solução completa para configuração.

DevFest
Sul de Minas



Cobra é um framework para CLIs escritos em Go. Integrado com o **Viper**, entrega tudo o que é preciso.

DevFest
Sul de Minas

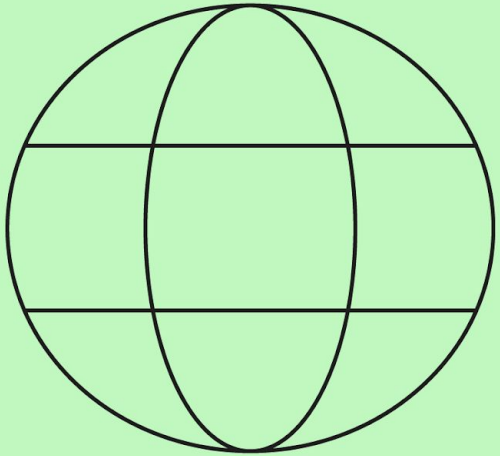


cobra fornece menus de ajuda!

godoc fornece doc de código.

Quanto mais **informação melhor**, mas mais **difícil** de manter.

DevFest
Sul de Minas

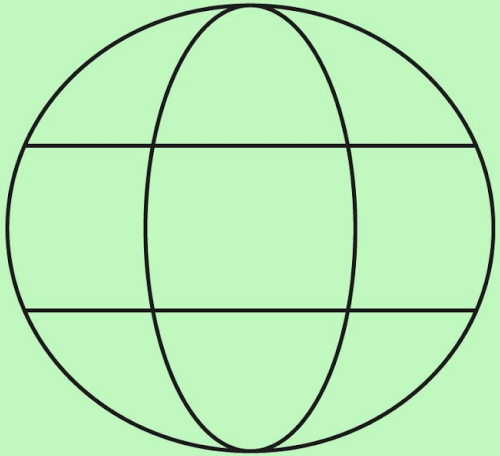


{ DevFest }

Sul de Minas

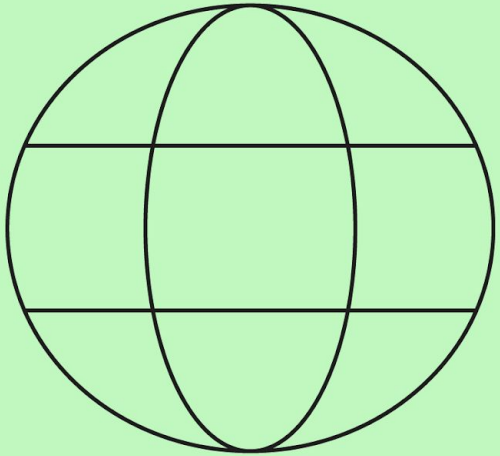
Deve ser **simples** para o usuário **utilizar** a ferramenta e **descobrir** algo novo.

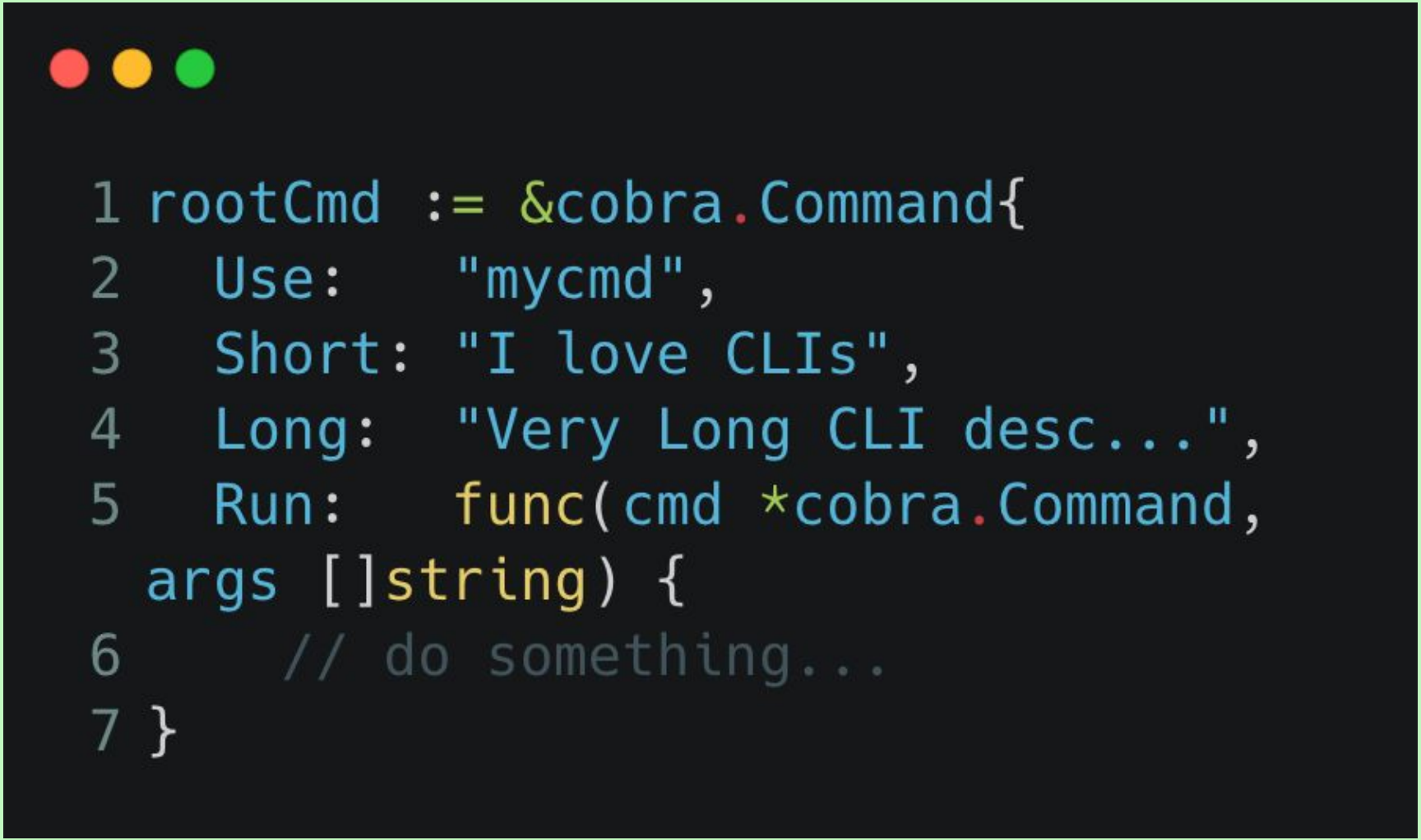
Os textos devem ser **concisos**, mas **explicativos**. Se possível, dê exemplos!



O comando **principal**
deve ser um substantivo

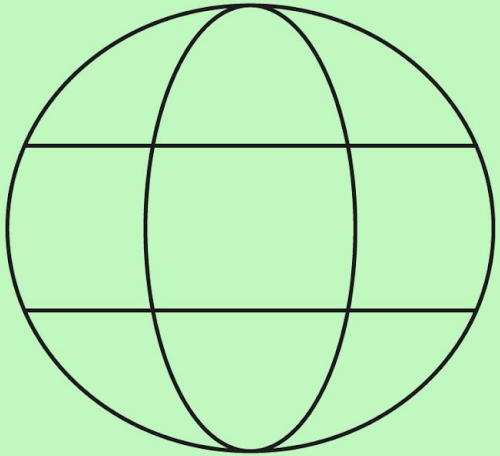
DevFest
Sul de Minas

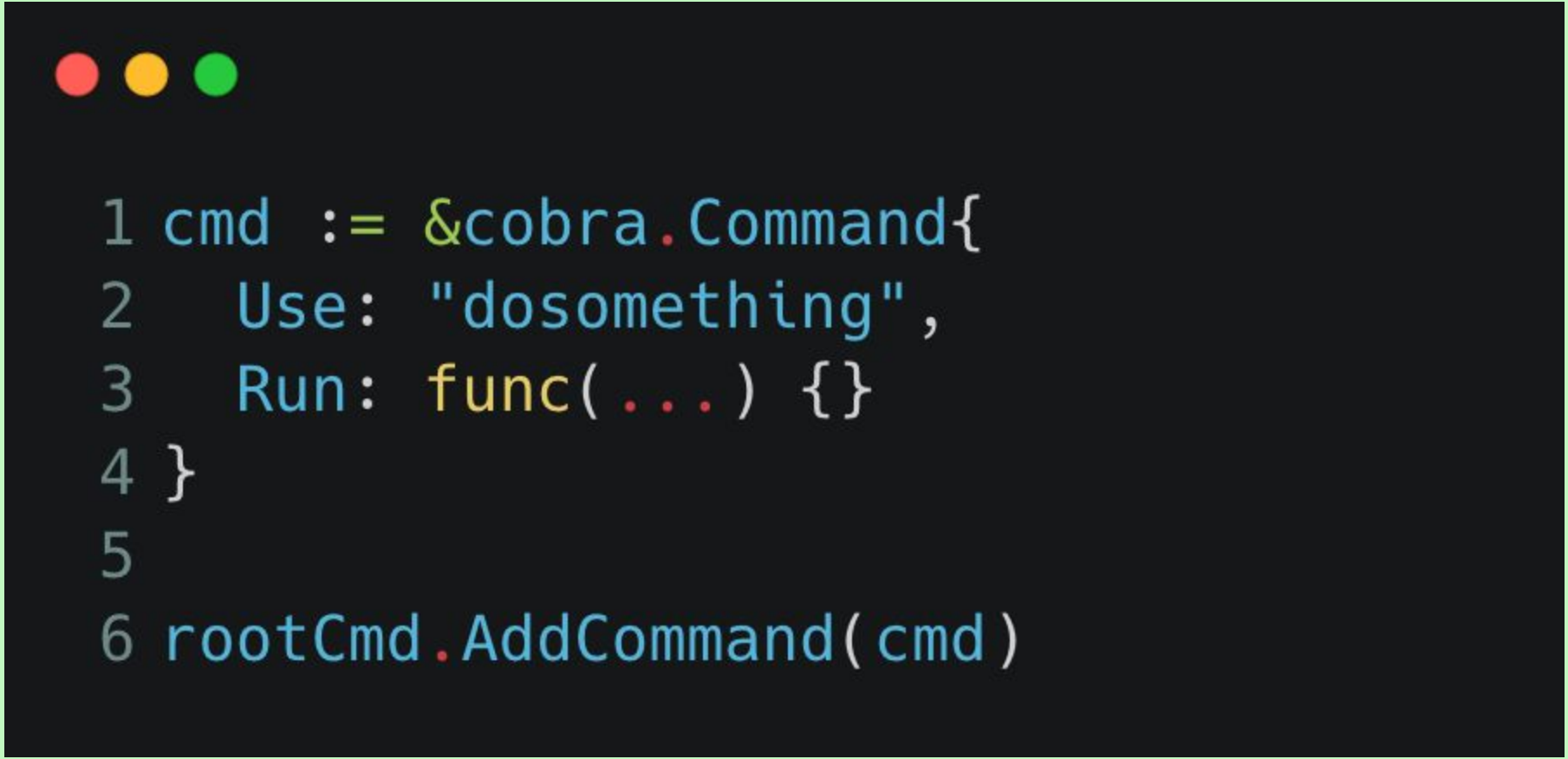



rootCmd := &cobra.Command{
Use: "mycmd",
Short: "I love CLIs",
Long: "Very Long CLI desc...",
Run: func(cmd *cobra.Command, args []string) {
// do something...
}

Subcomandos devem ser verbos

DevFest
Sul de Minas



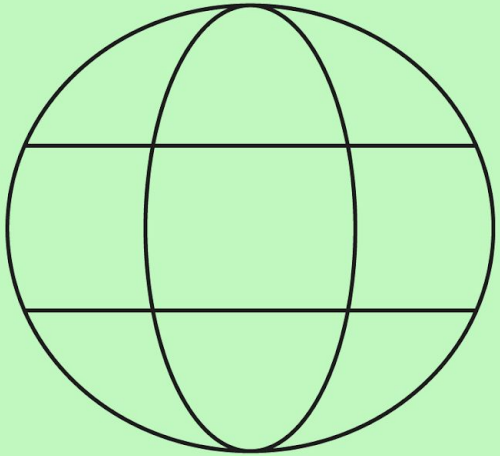

cmd := &cobra.Command{
Use: "dosomething",
Run: func(...) {}
}
rootCmd.AddCommand(cmd)

{ DevFest }

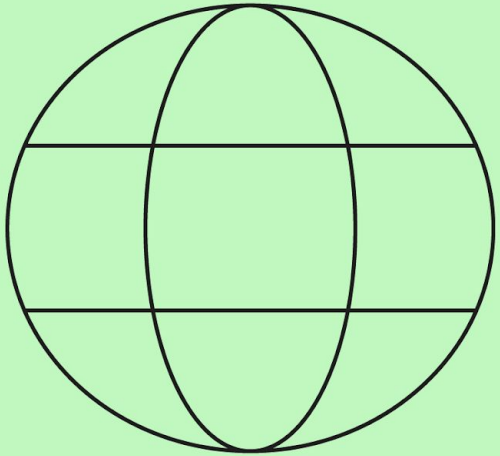
Sul de Minas

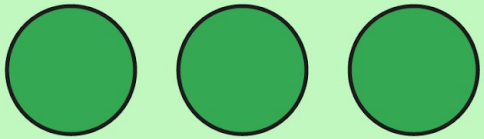
Como fornecer parâmetros para a CLI?

Por meio de **flags** e **argumentos**.



go test -v file1.go

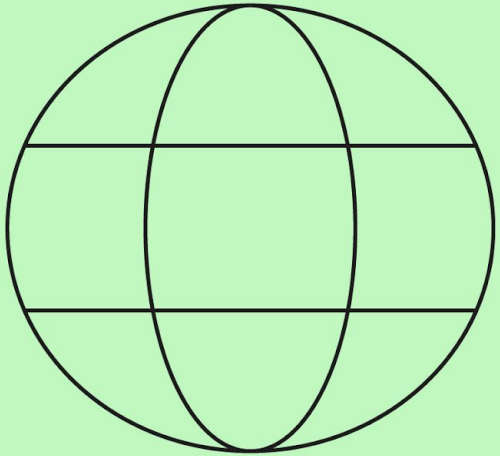


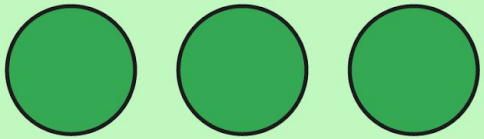


go run file1.go file2.go

!=

go run file2.go file1.go





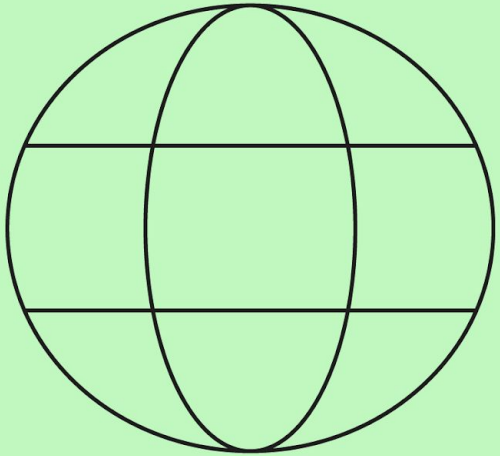
DevFest

Sul de Minas

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

==

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

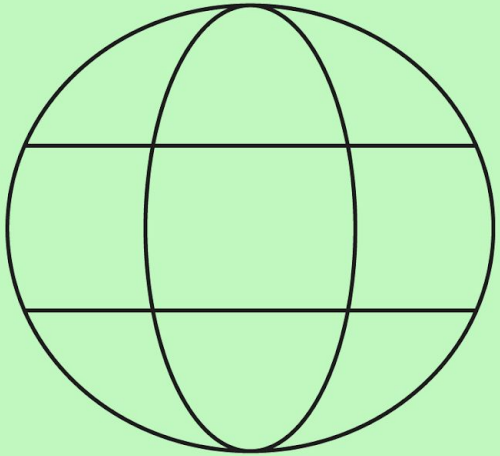



f := ""
cmd := &cobra.Command{
Use: "dosomething",
Args: cobra.MinimumNArgs(1),
Run: func(...) {}
}
cmd.Flags().StringVarP(&f, "full",
"f", "short", "long desc")

Prefira **flags** ao invés de **argumentos**.

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

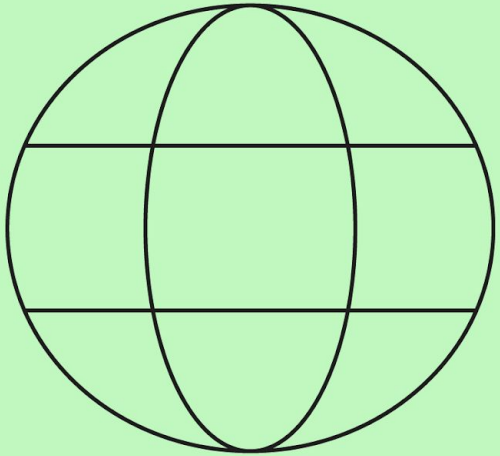
DevFest
Sul de Minas



DevFest
Sul de Minas

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

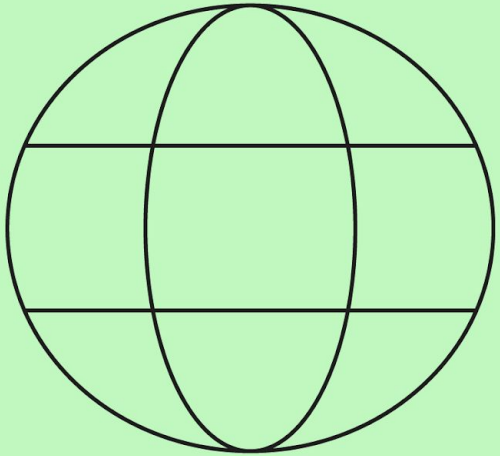
go test -v -f=o.json .



Não leia segredos e dados sensíveis por meio de **flags** ou **argumentos**.

Se for executar algo **destrutivo**, peça **confirmação** do usuário.

DevFest
Sul de Minas

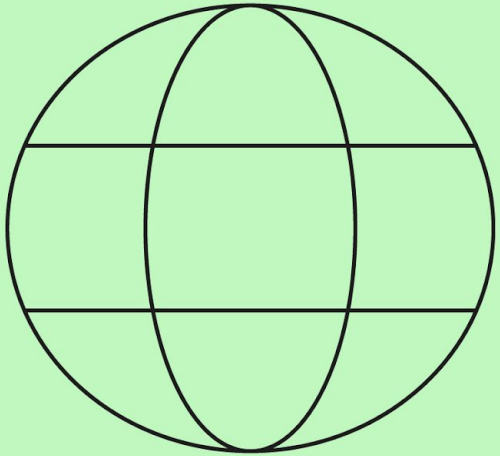


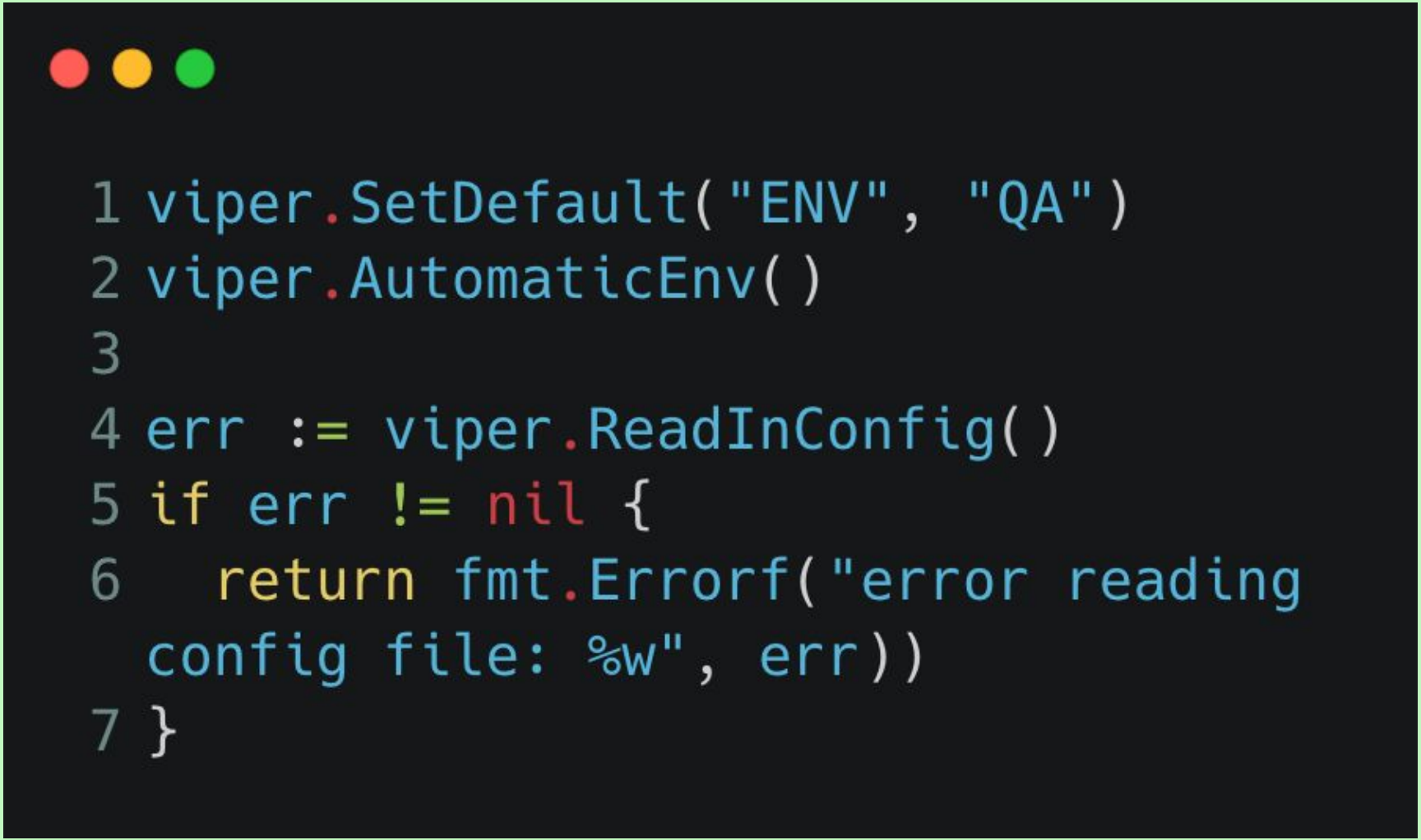
{ DevFest }

Sul de Minas

Como **configurar** sua CLI?

Por meio de **flags, env vars** e **config files**.

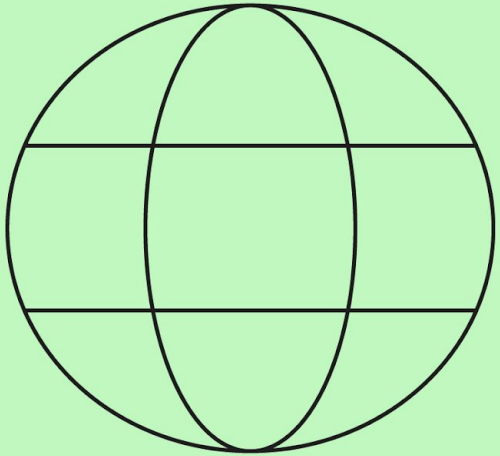



viper.SetDefault("ENV", "QA")
viper.AutomaticEnv()
err := viper.ReadInConfig()
if err != nil {
return fmt.Errorf("error reading config file: %w", err))
}

Ordem de configuração:

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

DevFest
Sul de Minas

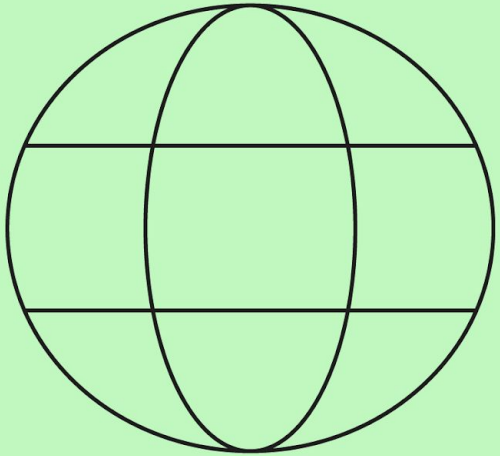


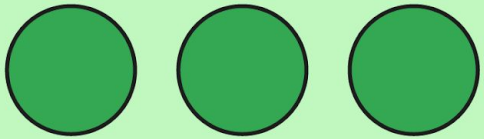
DevFest

Sul de Minas

Charm libs te ajudam a criar CLIs interativas.

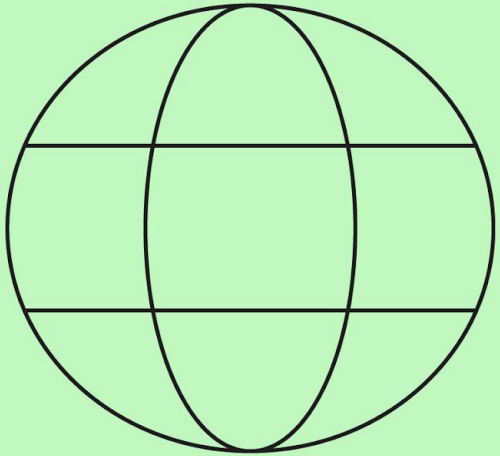
Utilize barras de progresso em operações longas.





Usuários devem
entender os erros.

DevFest
Sul de Minas

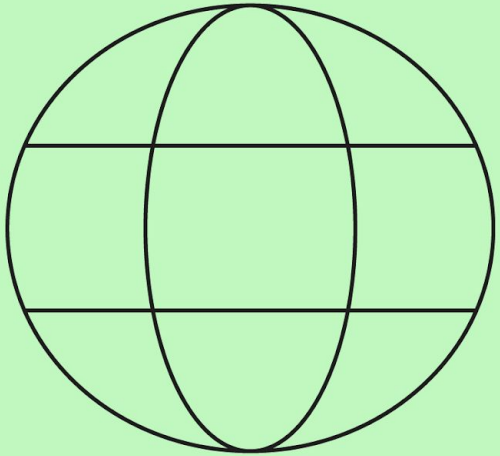


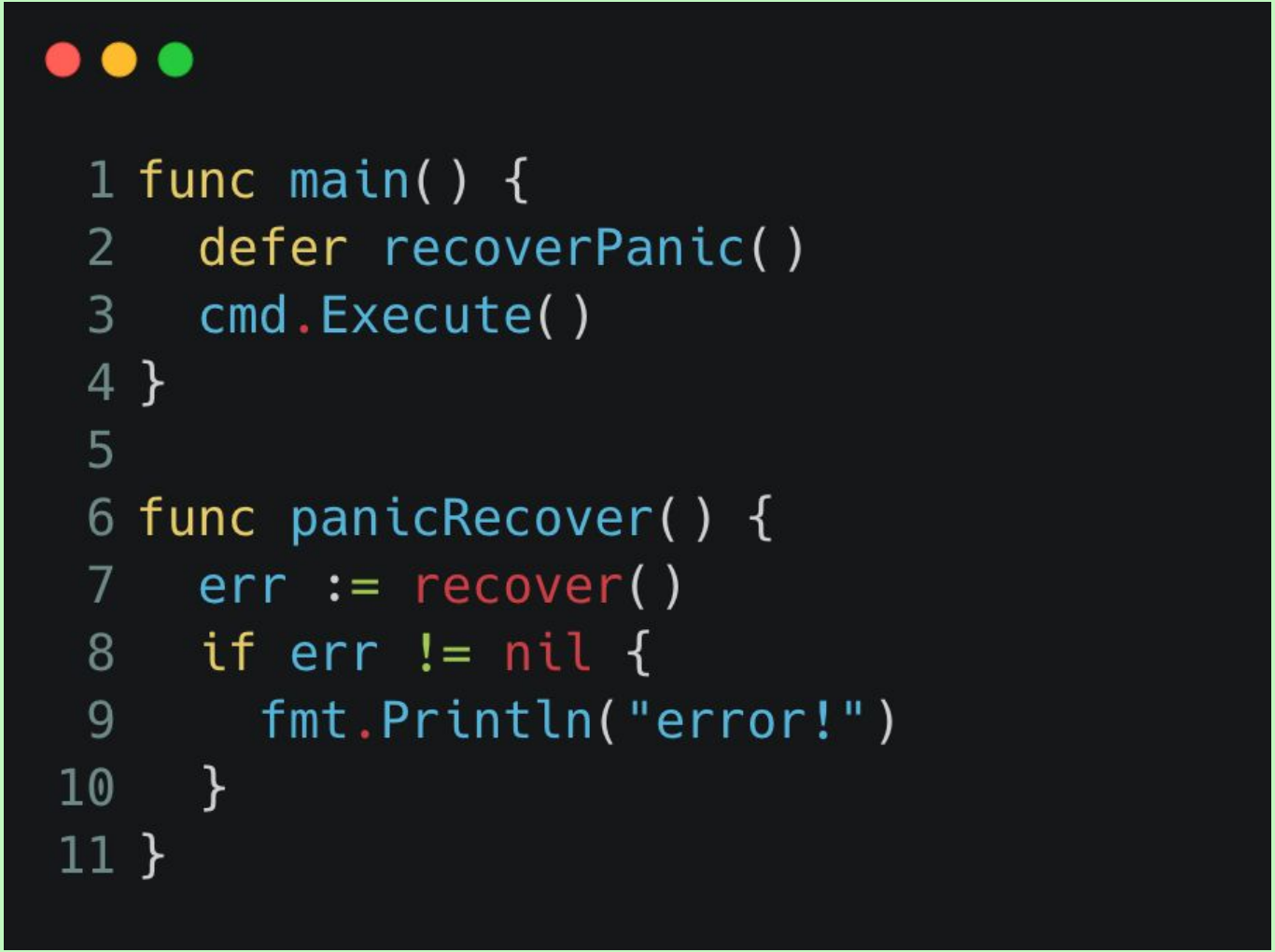

x, err := doSomething()
if err != nil {
return fmt.Errorf("human readable err: %w", err)
}

Não **exploda** com erros desconhecidos.

DevFest

Sul de Minas

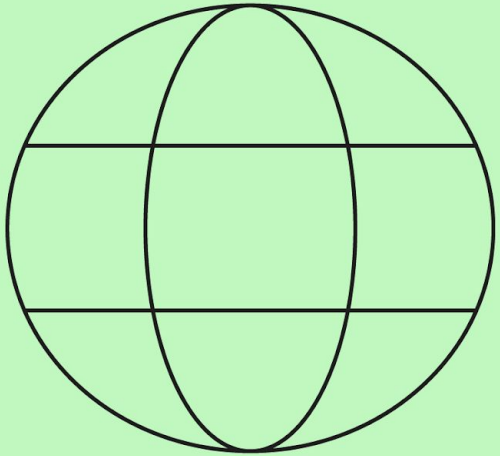



func main() {
defer recoverPanic()
cmd.Execute()
}
func panicRecover() {
err := recover()
if err != nil {
fmt.Println("error!")
}
}

Capture os **erros** com alguma ferramenta!

DevFest

Sul de Minas



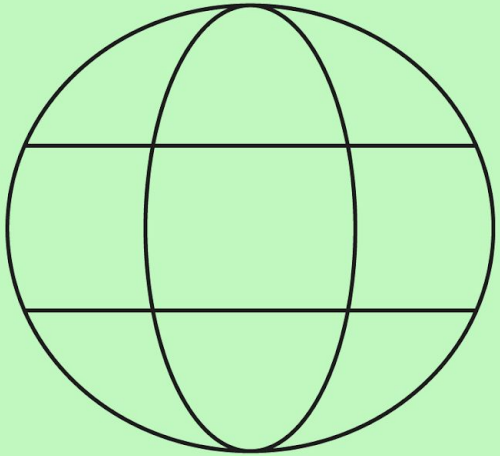

if viper.Get("ENABLE_SENTRY") {
sentry.Init(sentry.ClientOptions{
Dsn: "mydsn",
})
}
err := cmd.Execute()
if err != nil {
sentry.CaptureException(err)
}

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

Se for usar **concorrência**, alguns padrões podem te **ajudar**!

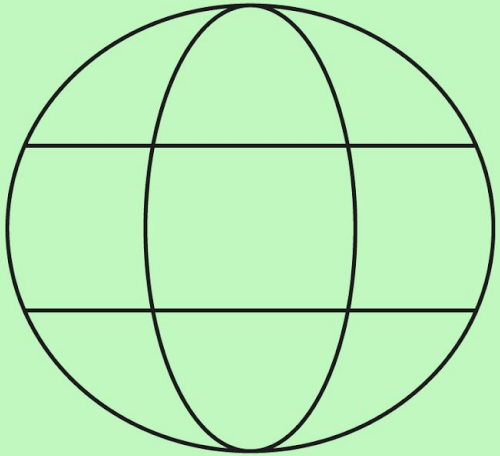
{ DevFest }

Sul de Minas



É possível gerar **profiles** através de **testes** de **benchmark**.

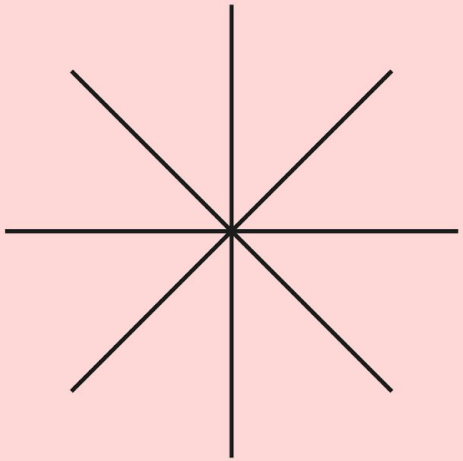
DevFest
Sul de Minas




$ 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

DevFest

Sul de Minas



Referências e links úteis

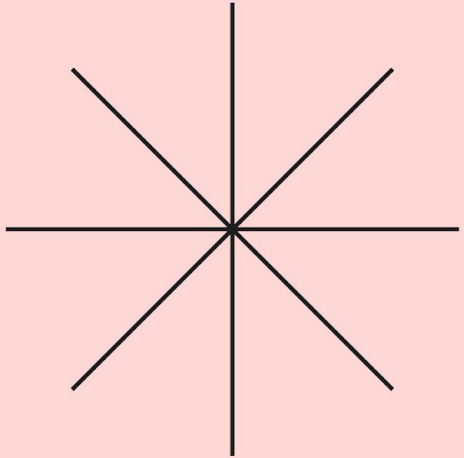
Para quem quiser aprofundar mais sobre os temas...

Google Developer Groups

clig.dev

Excelente guia para construir sua CLI, independente de linguagem.

DevFest
Sul de Minas

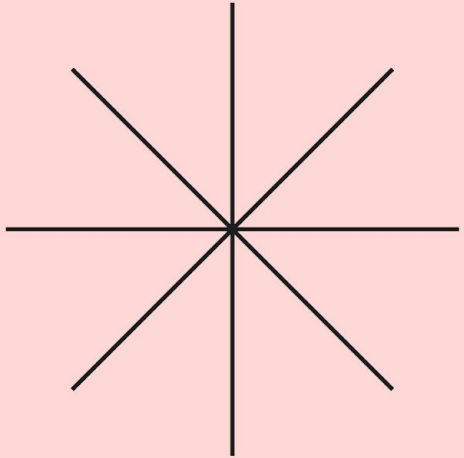


{ DevFest }

Sul de Minas

Post by
Adam Czapski

Post que estrutura bem alguns conceitos de uma boa CLI.

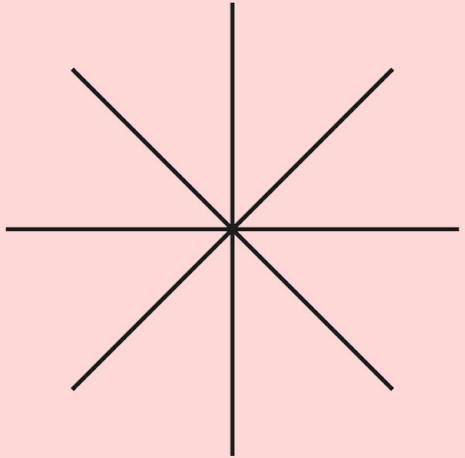


{ DevFest }

Sul de Minas

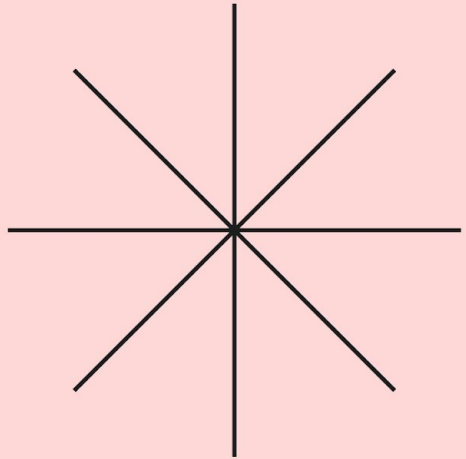
[Thread by @thewizardlucas]

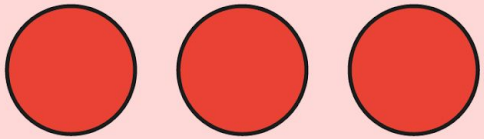
Thread no X bem detalhada sobre padrões de UX para CLIs.



mfbmina.dev

Meu blog pessoal, em que falo sobre engenharia de software, desenvolvimento e Go.





Alguma dúvida?

Obrigado!