01

Iniciando o monitoramento

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/624-golang/03/projetos/alura-golang-stage-fim-cap03.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Agora que já começamos a separar o nosso código em funções menores, já podemos dar início ao principal da aplicação, que é monitorar os sites, para ver se os mesmos estão online ou não. Então, vamos começar criando a função `iniciarMonitoramento`:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
}
```

E já vamos chamá-la caso o usuário digite o comando 1:

```
package main

import "fmt"
import "os"

func main() {

    exibeIntroducao()
    exibeMenu()
    comando := leComando()

    switch comando {
    case 1:
        iniciarMonitoramento()
    case 2:
        fmt.Println("Exibindo Logs...")
    case 0:
        fmt.Println("Saindo do programa")
        os.Exit(0)
    default:
        fmt.Println("Não conheço este comando")
        os.Exit(-1)
    }
}

// restante do código omitido
```

Se queremos que o programa fique testando se o site está online, ou caiu, ele precisa acessar o site. Se queremos acessar um site, precisamos realizar uma **requisição web**, utilizando a linguagem Go.

Para fazer requisições web, existe um pacote especialista nisso, dentro do `net`, pacote de internet do Go, há o `http`, pacote responsável pelas requisições web:

```
package main

import "fmt"
import "os"
import "net/http"

// restante do código omitido
```

Agora, vamos monitorar o site da Alura, então, na função `iniciarMonitoramento`, vamos definir uma variável com o nome do site:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
}
```

Com o site em mãos, vamos fazer uma requisição GET para o mesmo, utilizando a função `Get`, de `http`:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
    http.Get(site)
}
```

Quando acessamos o site da Alura através do browser, obtemos uma resposta, que é carregada no navegador. A mesma coisa acontece quando carregamos o site através do Go, essa resposta vem através de um retorno da função `Get`, que iremos guardar na variável `resp`:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
    resp := http.Get(site)
}
```

Ao salvar o arquivo, vemos que o Visual Studio Code aponta um erro, isso acontece porque a função `Get` retorna **mais de um valor**. Sim, existem funções no Go que retornam mais de um valor e a `Get` é uma delas, além da resposta, ela também retorna um possível erro que possa ter acontecido na requisição:

```
// restante do código omitido

func iniciarMonitoramento() {
```

```
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
    resp, err := http.Get(site)
}
```

Novidade isso, né? Vamos avaliar essa questão de funções que retornam mais de um valor no próximo vídeo.