Go: creare una CLI per il download di un file con indicatore di avanzamento

Per implementare un'applicazione CLI in Go che effettua il download di un file e mostra l'avanzamento del download, possiamo seguire dei passaggi precisi.

I passaggi sono i seguenti:

1. **Setup dei parametri**: Usiamo il package `flag` per il parsing dei parametri passati dalla riga di comando.
2. **Richiesta HTTP**: Usiamo il package `http` per inviare la richiesta e gestire la risposta.
3. **Visualizzazione dell'avanzamento del download**: Gestiamo lo stream di dati del download per calcolare la percentuale di avanzamento del download.

Possiamo scrivere il seguente codice:

```
package main

import (
    "flag"
    "fmt"
    "io"
    "net/http"
    "os"
    "strings"
```

```
)

func main() {
    // Ottiene l'URL come argomento
    flag.Parse()
    args := flag.Args()
    if len(args) < 1 {
        fmt.Println("Please provide a URL")
        return
    }
    url := args[0]

    // Inizializza il download
    err := downloadFile(url)
    if err != nil {
        fmt.Fprintf(os.Stderr, "Error downloading
file: %v\n", err)
        os.Exit(1)
    }
}

func downloadFile(url string) error {
    resp, err := http.Get(url)
    if err != nil {
        return err
    }
    defer resp.Body.Close()

    // Crea il file
    segments := strings.Split(url, "/")
    fileName := segments[len(segments)-1]
    file, err := os.Create(fileName)
    if err != nil {
        return err
```

```
    }
    defer file.Close()

    // Avanzamento
    totalBytes := resp.ContentLength
    var downloadedBytes int64 = 0

    // Buffer diviso in segmenti di 32 Kb
    buffer := make([]byte, 32*1024)
    for {
        n, err := resp.Body.Read(buffer)
        if n > 0 {
            _, writeErr := file.Write(buffer[:n])
            if writeErr != nil {
                return writeErr
            }
            downloadedBytes += int64(n)
            fmt.Printf("\rDownloading... %d%%
complete", 100*downloadedBytes/totalBytes)
        }
        if err == io.EOF {
            break
        }
        if err != nil {
            return err
        }
    }

    fmt.Println("\nDownload completed successfully")
    return nil
}
```

Il presupposto fondamentale per il funzionamento del codice è che il server remoto fornisca il corretto header HTTP `Content-Length` con la dimensione del file espressa in byte.