Packages for CLI development in Go

CLI Development Libraries

```
go get github.com/charmbracelet/bubbles/viewport

go get github.com/charmbracelet/
   bubbleteaithub.com/charmbracelet/bubbletea

go get github.com/charmbracelet/glamour

go get github.com/charmbracelet/lipgloss
```

Go CLI App

```
package main

import (
    "bufio"
    "fmt"
    "os"

    "github.com/charmbracelet/bubbles/table"
    tea "github.com/charmbracelet/bubbletea"
    "github.com/charmbracelet/lipgloss"
)

// +-------------------------+ //
// |                         |  //
// |          Style          |  //
// |                         |  //
// +-------------------------+ //

var baseStyle = lipgloss.NewStyle().
    BorderStyle(lipgloss.NormalBorder()).
    BorderForeground(lipgloss.Color("240"))

// +-------------------------+ //
// |                         |  //
// |       Interaction       |  //
// |                         |  //
// +-------------------------+ //

type model struct {
    table table.Model
}

func (m model) Init() tea.Cmd { return nil }

func (m model) Update(msg tea.Msg) (tea.Model, tea.Cmd) {

    var cmd tea.Cmd
    switch msg := msg.(type) {
```

```go
    case tea.KeyMsg:
        switch msg.String() {
        case "esc":
            if m.table.Focused() {

                m.table.Blur()
            } else {
                m.table.Focus()
            }
        case "q", "ctrl+c":
            return m, tea.Quit

        case "enter":
            return m, tea.Batch(

                tea.Printf("Let's go to %s", m.table.SelectedRow()[1]),
            )

        }

    }
    m.table, cmd = m.table.Update(msg)
    return m, cmd
}

func (m model) View() string {

    return baseStyle.Render(m.table.View()) + "\n"
}

func main() {

    // +--------------------------+ //
    // |                          |  //
    // |         Input            |  //
    // |                          |  //
    // +--------------------------+ //

    var message string = ""
    var key string = ""
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Print("Your Message: ")
    if scanner.Scan() {

        message = scanner.Text()

    } else {

        fmt.Println("Error reading your message", scanner.Err())
    }

    fmt.Printf("Your Key:")
    if scanner.Scan() {

        key = scanner.Text()

    } else {

        fmt.Println("Error reading your key", scanner.Err())
    }

    // +-------------------------+ //
    // |                         |  //
    // |         Output          |  //
    // |                         |  //
    // +-------------------------+ //

    columns := []table.Column{
        {Title: "Key", Width: 20},
        {Title: "Message", Width: 20},
        {Title: "Encrypted Message", Width: 20},
        {Title: "Decrypted Message", Width: 20},
    }

    rows := []table.Row{

        {key, message, "-", "-"},
    }

    t := table.New(
        table.WithColumns(columns),
        table.WithRows(rows),
        table.WithFocused(true),
        table.WithHeight(7),
    )

    s := table.DefaultStyles()
    s.Header = s.Header.
        BorderStyle(lipgloss.NormalBorder()).
        BorderForeground(lipgloss.Color("240")).
        BorderBottom(true).
        Bold(false)
    s.Selected = s.Selected.
        Foreground(lipgloss.Color("229")).
        Background(lipgloss.Color("57")).
        Bold(false)
    t.SetStyles(s)

    m := model{t}
    if _, err := tea.NewProgram(m).Run(); err != nil {
        fmt.Println("Error running program:", err)
        os.Exit(1)
    }
}
```