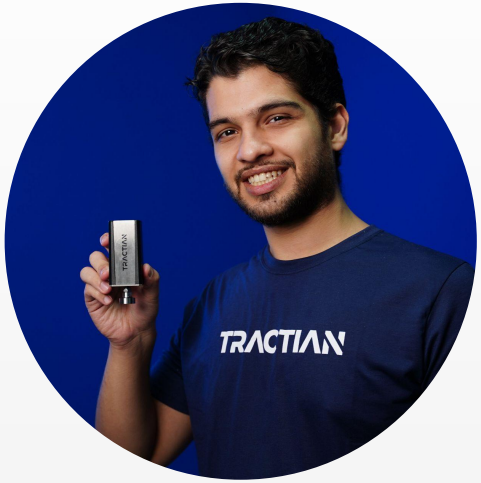
WORKSHOP

GOLANG SP



TRACTIAN

Senior Software Engineer

reneepc

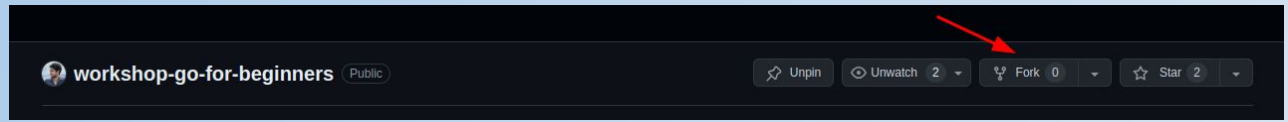
reneepc

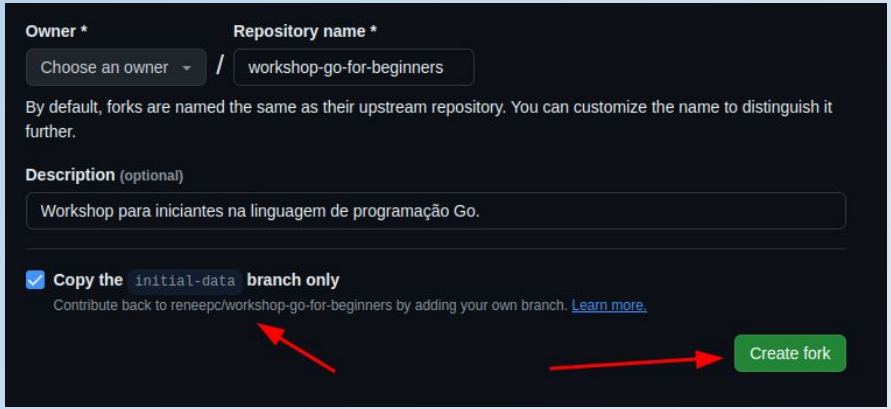
Especializado em transformar cafeína em código de qualidade duvidosa.

Faço `defer` das minhas obrigações até ser tarde demais.

WORKSHOP - SORTEADOR

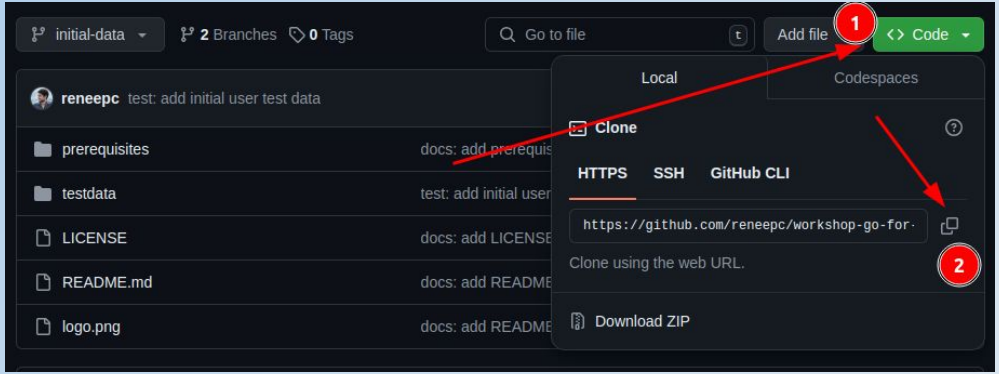
Fork do repositório



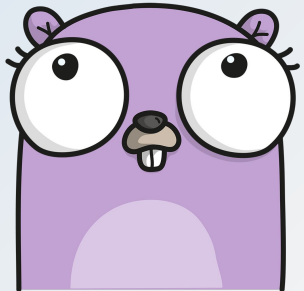


https://github.com/reneepc/workshop-go-for-beginners

Clonando o repositório



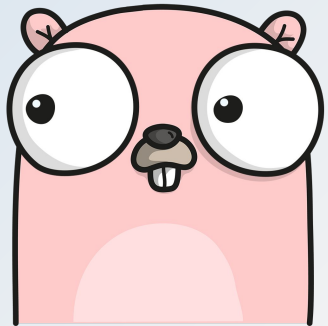

```
$> git clone <ctrl-shift-v/ctrl-v>
$> ...
$> cd workshop-go-for-beginners
$> code . # Caso esteja usando VSCode,
          # senão pode abrir no seu editor de preferência.
```



Main

```go
package main

import "fmt"

func main() {
  fmt.Println("Hello world")
}
```

```
$> go run main.go
```



Lendo um arquivo

```
import "os"


func ReadFile(name string) ([]byte, error)
```

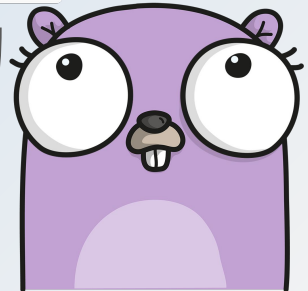
CSV

[]string

Header

```
1 Email,Name,Phone,CPF,Address
2 Henrique.Nogueira19@live.com,Marcilio Macedo,(67) 64854-6609,22628163101,457 Batista Marginal
3 Julia_Macedo77@live.com,Eliane Moreira,(54) 55001-1324,66875475855,24387 Angela Rua
4 Marily.Costa@live.com,Vanderson Saraiva Filho,+55 (50) 6505-7903,99652502510,9542 Nuno Marginal
5 Ronan83@gmail.com,Andrea Macedo,(20) 8199-8026,95649227470,413 Angelita Marginal
6 Josafa.Moraes@live.com,Marco Nogueira,+55 (12) 1968-3008,22787748759,81614 Saraiva Avenida
7 Lazara_Santos@yahoo.com,Barbara Oliveira Jr.,(35) 95500-1158,06444826560,719 Oliveira Travessa
8 Junis45@yahoo.com,Sr. Ettori Martins,(73) 6996-1299,27587444734,732 Altavir Avenida
9 Romao8@live.com,Célia Oliveira,(31) 3598-4933,40568314751,766 Rose Travessa
10 Clezio_Melo@gmail.com,Dulce Melo,(78) 10439-6601,57654718110,4496 Jamiro Viela
11 Lucimar_Moreira@hotmail.com,Sômulo Carvalho Filho,(20) 82887-9623,22472904347,6329 Teodoro Marginal
```

[][]string



Pacotes: os e encoding/csv

Discover Packages > Standard library > os

os package standard library

Version: go1.22.5 Latest | Published: Jul 2, 2024 | License: BSD-3-Clause | Imports: 17 | Imported by: 1,956,900

Jump to ... f

- Documentation
  - Overview
  - Index
  - Constants
  - Variables
  - Functions
  - Types
- Source Files
- Directories

Documentation

Rendered for linux/amd64

Overview

Package os provides a platform-independent interface to operating system functionality. The design is Unix-like, although the error handling is Go-like; failing calls return values of type error rather than error numbers. Often, more information is available within the error. For example, if a call that takes a file name fails, such as Open or Stat, the error will include the failing file name when printed and will be of type *PathError, which may be unpacked for more information.

The os interface is intended to be uniform across all operating systems. Features not generally available appear in the system-specific package syscall.

Here is a simple example, opening a file and reading some of it.

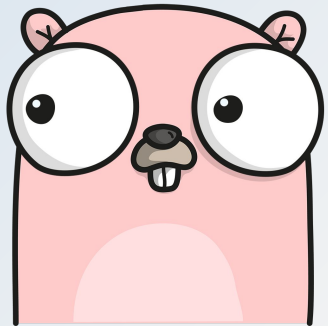
```
file, err := os.Open("file.go") // For read access.
if err != nil {
    log.Fatal(err)
}
```

If the open fails, the error string will be self-explanatory, like

```
open file.go: no such file or directory
```

The file's data can then be read into a slice of bytes. Read and Write take their byte counts from the length of the argument slice.

```
data := make([]byte, 100)
count, err := file.Read(data)
if err != nil {
    log.Fatal(err)
}
fmt.Printf("read %d bytes: %q\n", count, data[:count])
```



Extraindo em uma função

```
1 func ParseCsvFile(filePath string) ([][]string, error)
```

```
$> go run main.go
```

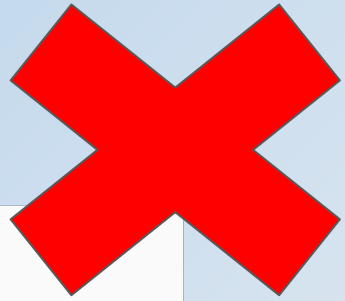
Extraindo para um novo arquivo (parser.go)

```
1 func ParseCsvFile(filePath string) ([][]string, error)
```

```
$> go run main.go
```

Extraindo para um novo arquivo (parser.go)

```
1 func ParseCsvFile(filePath string) ([][]string, error)
```



```
$> go run main.go
```

Extraindo para um novo arquivo (parser.go)

```
$> go run main.go parser.go
```

Extraindo para um novo arquivo (parser.go)

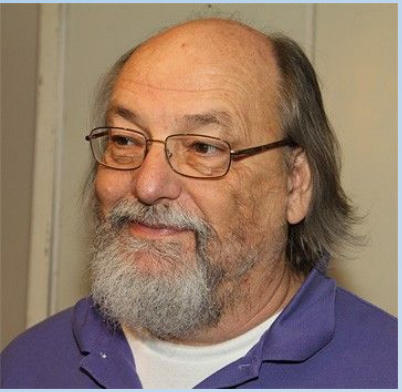
```
$> go build -o sorteador main.go parser.go
$> ./sorteador
```

Extraindo para um novo arquivo (parser.go)

```
$> gcc -o main main.c anotherSource.c
$> ./main
```

Extraindo para um novo arquivo (parser.go)

```
$> gcc -o main main.c anotherSource.c
$> ./main
```



Ken Thompson

Inventor do C e do Unix (Turing Award)

Rodando sem precisar especificar o arquivo

```
$> go run .
```

Extraindo para um novo arquivo (parser.go)

Extraindo para um novo arquivo (parser.go)

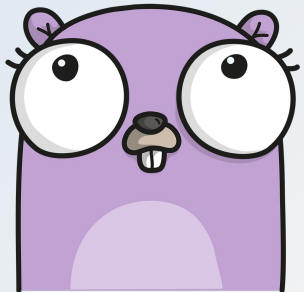
```
$> go run .
go: cannot find main module, but found .git/config in /home/reneepc/code/src/reneepc/workshop-go-for-beginners
        to create a module there, run:
        go mod init
```

Extraindo para um novo arquivo (parser.go)

```
$> go run .
go: go.mod file not found in current directory or any parent directory; see 'go help modules'
```

Módulos

- Múltiplos arquivos devem pertencer a um módulo
- O nome do módulo deve ser único



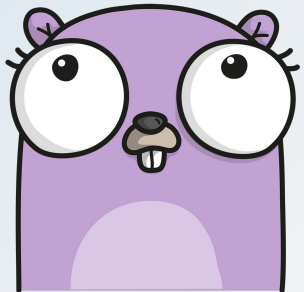
Módulos

Java

Domínio	Nome do pacote
workshop.microsoft.com.br	br.com.microsoft.workshop

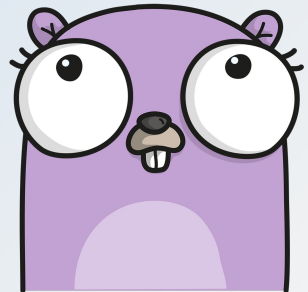
Javascript (NPM)

- @nome-da-organização/nome-do-pacote
- somente-nome-do-pacote (propenso a colisões)



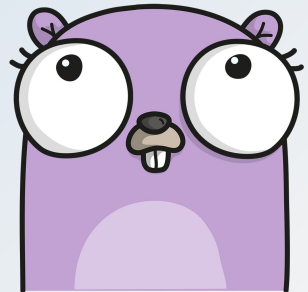
Módulos

```
$> go mod init github.com/<seu-username>/workshop-go-for-beginners
```



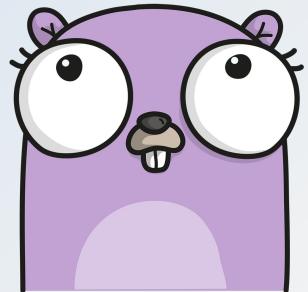
Módulos

```
go: creating new go.mod: module github.com/reneepc/worshop-go-for-beginners
go: to add module requirements and sums:
        go mod tidy
```



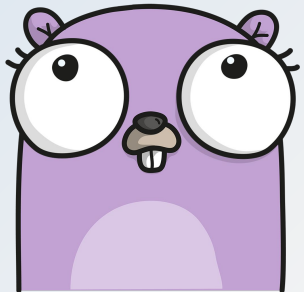
go.mod

```
module github.com/reneepc/workshop-go-for-beginners

go 1.22.0

require (
  gitlab.com/user/a-module v1.2.3
  go.pkg/another-module v1.2.3
)
```



Módulos

```
$> go run .
```



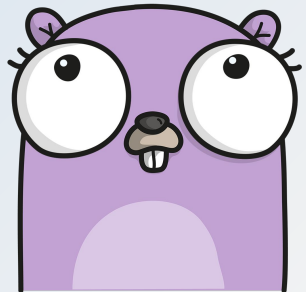
Sortear ganhadores

```go
import "math/rand/v2"

rand.IntN()
// ou
rand.Shuffle()
```

```go
import "math/rand/v2"

func SelectWinners(records [][]string, nWinners int) ([][]string, error) {
   ...
}
```

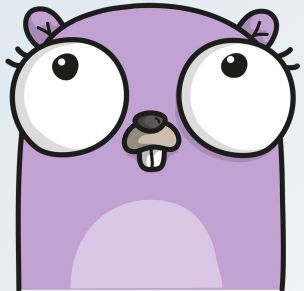
Adicionar flags de linha de comando

```
$> go run . -file testdata/users_003.csv -winners 10
```



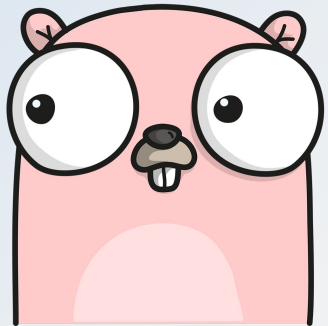
Adicionar flags de linha de comando

```
import "flag"

func main() {
    ... = flag.String()
}
```



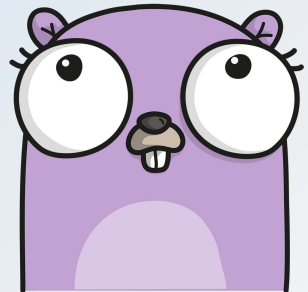
Removendo duplicados

- Recebam um [][]string e removam os duplicados.
- E-mail como critério



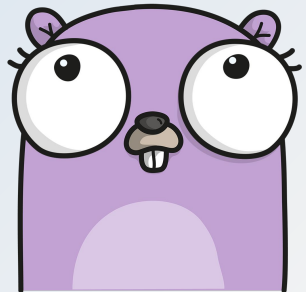
Salve os ganhadores em um arquivo

```
func SaveWinners(filePath string, winners [][]string) error
```



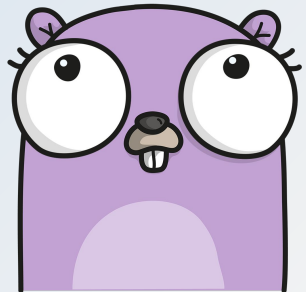
Precisamos avisar os ganhadores

- SMS
- WhatsApp
- Email



Precisamos avisar os ganhadores

- SMS
- WhatsApp
- Email



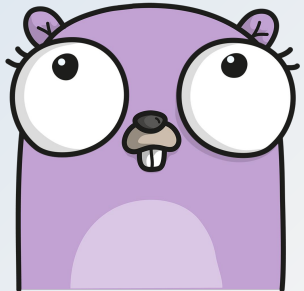
Precisamos avisar os ganhadores

- SMS
- WhatsApp
- Email

twilio

infobip

CLOUDTALK



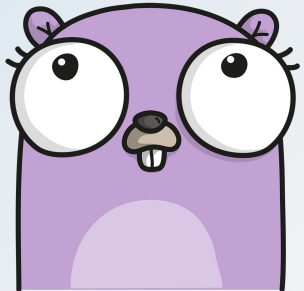
Precisamos avisar os ganhadores

- SMS
- WhatsApp
- **Email**

twilio

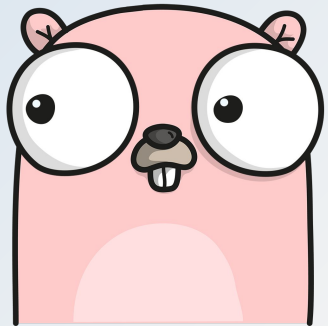
infobip

CLOUDTALK



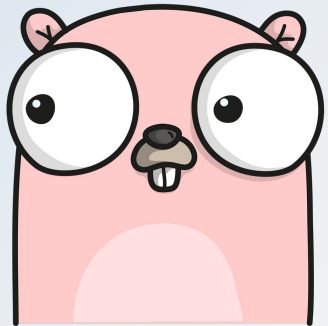
Instalando uma biblioteca externa

```
$> go get github.com/go-gomail/gomail
```



Mudança no go.mod

```
module github.com/reneepc/workshop-go-for-beginners

go 1.22.0

require github.com/go-gomail/gomail v0.0.0-20160411212932-81ebce5c23df

require (
    gopkg.in/alexcesaro/quotedprintable.v3 v3.0.0-20150716171945-2caba252f4dc // indirect
    gopkg.in/gomail.v2 v2.0.0-20160411212932-81ebce5c23df // indirect
)
```

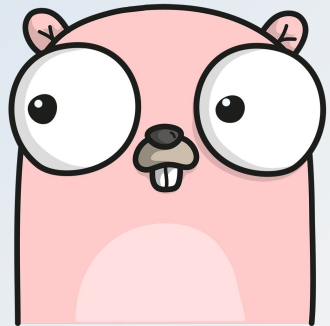


Mudança no go.mod

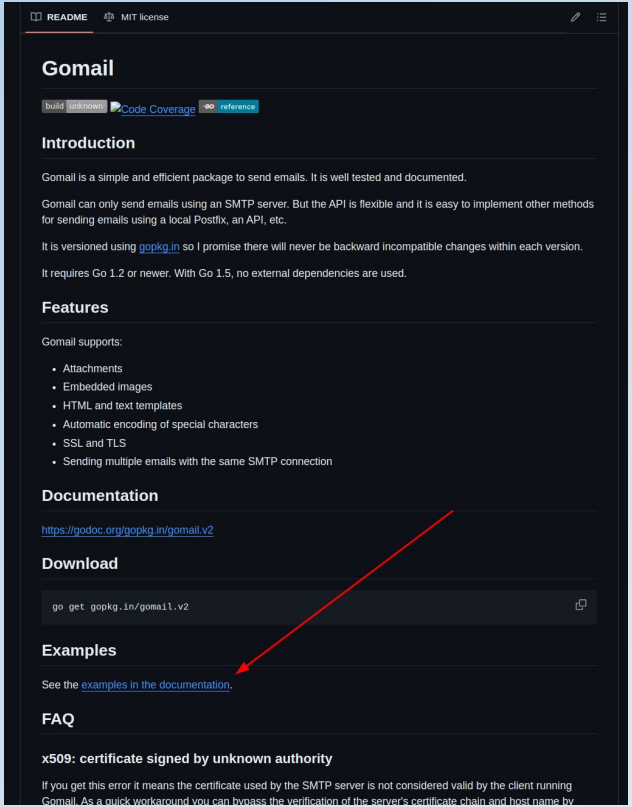
```
module github.com/reneepc/workshop-go-for-beginners

go 1.22.0

require github.com/go-gomail/gomail v0.0.0-20160411212932-81ebce5c23df

require (
    gopkg.in/alexcesaro/quotedprintable.v3 v3.0.0-20150716171945-2caba252f4dc // indirect
    gopkg.in/gomail.v2 v2.0.0-20160411212932-81ebce5c23df // indirect
)
```

Lib instalada

Dependências indiretas



Como enviar um e-mail



README MIT license

Gomail

build unknown Code Coverage reference

Introduction

Gomail is a simple and efficient package to send emails. It is well tested and documented.

Gomail can only send emails using an SMTP server. But the API is flexible and it is easy to implement other methods for sending emails using a local Postfix, an API, etc.

It is versioned using gopkg.in so I promise there will never be backward incompatible changes within each version.

It requires Go 1.2 or newer. With Go 1.5, no external dependencies are used.

Features

Gomail supports:

- Attachments
- Embedded images
- HTML and text templates
- Automatic encoding of special characters
- SSL and TLS
- Sending multiple emails with the same SMTP connection

Documentation

https://godoc.org/gopkg.in/gomail.v2

Download

```
go get gopkg.in/gomail.v2
```

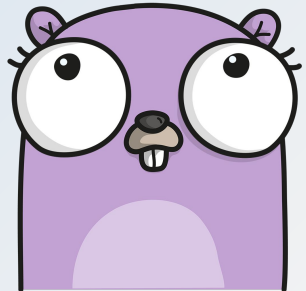
Examples

See the examples in the documentation.

FAQ

x509: certificate signed by unknown authority

If you get this error it means the certificate used by the SMTP server is not considered valid by the client running Gomail. As a quick workaround you can bypass the verification of the server's certificate chain and host name by



Como enviar um e-mail

README ¶

Gomail

build unknown Code Coverage go reference

Introduction

Gomail is a simple and efficient package to send emails. It is well tested and documented.

Gomail can only send emails using an SMTP server. But the API is flexible and it is easy to implement other methods for sending emails using a local Postfix, an API, etc.

It is versioned using gopkg.in so I promise there will never be backward incompatible changes within each version.

Expand ▾

Documentation

Overview

Package gomail provides a simple interface to compose emails and to mail them efficiently.

More info on Github: https://github.com/go-gomail/gomail

▼ Example

```
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err != nil {
    panic(err)
}
```

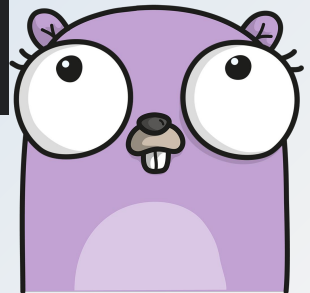
```
Output:
```

▸ Example (Daemon)

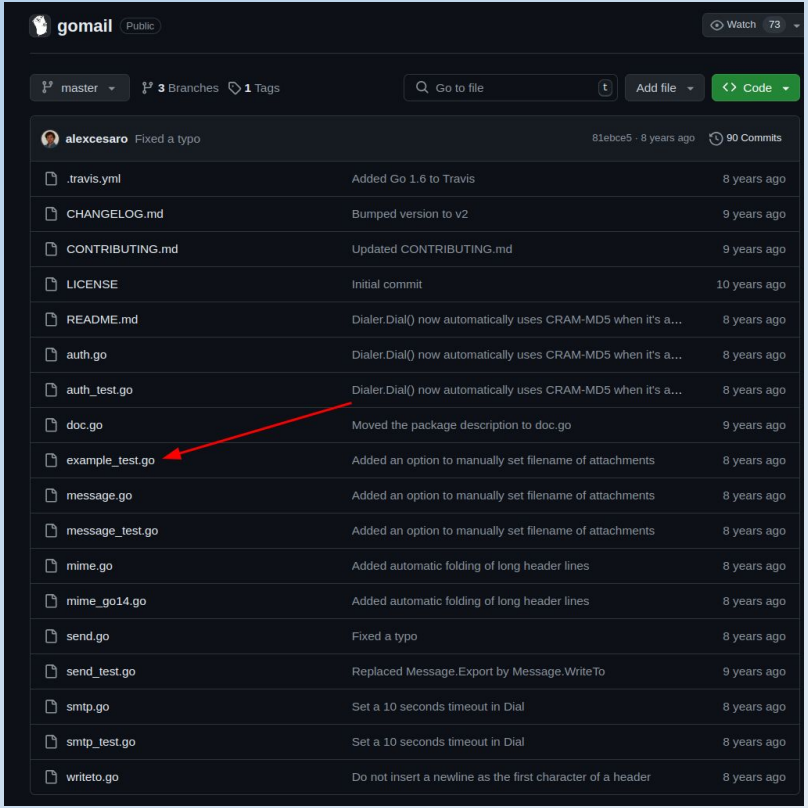
▸ Example (Newsletter) ¶

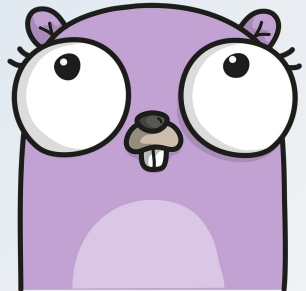
▸ Example (NoAuth)

▸ Example (NoSMTP)



Go doc


gomail
Public
Watch 73
master
3 Branches
1 Tags
Go to file
t
Add file
Code
alexcesaro Fixed a typo
81ebce5 · 8 years ago
90 Commits
.travis.yml Added Go 1.6 to Travis 8 years ago
CHANGELOG.md Bumped version to v2 9 years ago
CONTRIBUTING.md Updated CONTRIBUTING.md 9 years ago
LICENSE Initial commit 10 years ago
README.md Dialer.Dial() now automatically uses CRAM-MD5 when it's a... 8 years ago
auth.go Dialer.Dial() now automatically uses CRAM-MD5 when it's a... 8 years ago
auth_test.go Dialer.Dial() now automatically uses CRAM-MD5 when it's a... 8 years ago
doc.go Moved the package description to doc.go 9 years ago
example_test.go Added an option to manually set filename of attachments 8 years ago
message.go Added an option to manually set filename of attachments 8 years ago
message_test.go Added an option to manually set filename of attachments 8 years ago
mime.go Added automatic folding of long header lines 8 years ago
mime_go14.go Added automatic folding of long header lines 8 years ago
send.go Fixed a typo 8 years ago
send_test.go Replaced Message.Export by Message.WriteTo 9 years ago
smtp.go Set a 10 seconds timeout in Dial 8 years ago
smtp_test.go Set a 10 seconds timeout in Dial 8 years ago
writeto.go Do not insert a newline as the first character of a header 8 years ago



Go doc

func NewDialer

```
func NewDialer(host string, port int, username, password string) *Dialer
```

NewDialer returns a new SMTP Dialer. The given parameters are used to connect to the SMTP server.

func NewPlainDialer DEPRECATED Show

func (*Dialer) Dial

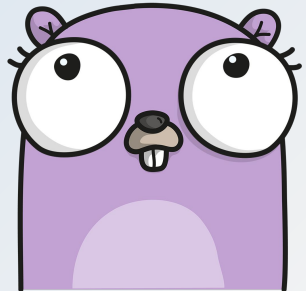
```
func (d *Dialer) Dial() (SendCloser, error)
```

Dial dials and authenticates to an SMTP server. The returned SendCloser should be closed when done using it.

func (*Dialer) DialAndSend

```
func (d *Dialer) DialAndSend(m ...*Message) error
```

DialAndSend opens a connection to the SMTP server, sends the given emails and closes the connection.



Go doc

func NewDialer

```
func NewDialer(host string, port int, username, password string) *Dialer
```

NewDialer returns a new SMTP Dialer. The given parameters are used to connect to the SMTP server.

func NewPlainDialer DEPRECATED Show

func (*Dialer) Dial

```
func (d *Dialer) Dial() (SendCloser, error)
```

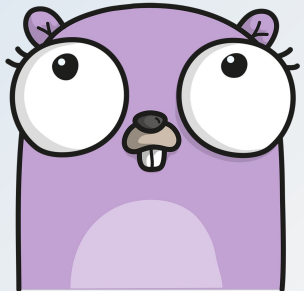
Dial dials and authenticates to an SMTP server. The returned SendCloser should be closed when done using it.

func (*Dialer) DialAndSend

```
func (d *Dialer) DialAndSend(m ...*Message) error
```

DialAndSend opens a connection to the SMTP server, sends the given emails and closes the connection.

```go
// NewDialer returns a new SMTP Dialer. The given parameters are used to connect
// to the SMTP server.
func NewDialer(host string, port int, username, password string) *Dialer {
    return &Dialer{
        Host:     host,
        Port:     port,
        Username: username,
        Password: password,
        SSL:      port == 465,
    }
}

// NewPlainDialer returns a new SMTP Dialer. The given parameters are used to
// connect to the SMTP server.
//
// Deprecated: Use NewDialer instead.
func NewPlainDialer(host string, port int, username, password string) *Dialer {
    return NewDialer(host, port, username, password)
}

// Dial dials and authenticates to an SMTP server. The returned SendCloser
// should be closed when done using it.
func (d *Dialer) Dial() (SendCloser, error) {
    conn, err := netDialTimeout("tcp", addr(d.Host, d.Port), 10*time.Second)
    if err != nil {
        return nil, err
    }
```



Go doc

func NewDialer

```
func NewDialer(host string, port int, username, password string) *Dialer
```

NewDialer returns a new SMTP Dialer. The given parameters are used to connect to the SMTP server.

func NewPlainDialer DEPRECATED Show

func (*Dialer) Dial

```
func (d *Dialer) Dial() (SendCloser, error)
```

Dial dials and authenticates to an SMTP server. The returned SendCloser should be closed when done using it.

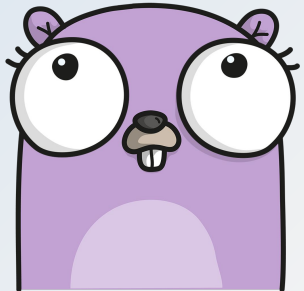
func (*Dialer) DialAndSend

```
func (d *Dialer) DialAndSend(m ...*Message) error
```

DialAndSend opens a connection to the SMTP server, sends the given emails and closes the connection.

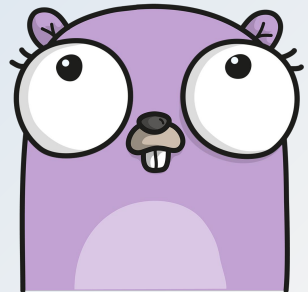
```go
// NewDialer returns a new SMTP Dialer. The given parameters are used to connect
// to the SMTP server.
func NewDialer(host string, port int, username, password string) *Dialer {
    return &Dialer{
        Host:     host,
        Port:     port,
        Username: username,
        Password: password,
        SSL:      port == 465,
    }
}

// NewPlainDialer returns a new SMTP Dialer. The given parameters are used to
// connect to the SMTP server.
//
// Deprecated: Use NewDialer instead.
func NewPlainDialer(host string, port int, username, password string) *Dialer {
    return NewDialer(host, port, username, password)
}

// Dial dials and authenticates to an SMTP server. The returned SendCloser
// should be closed when done using it.
func (d *Dialer) Dial() (SendCloser, error) {
    conn, err := netDialTimeout("tcp", addr(d.Host, d.Port), 10*time.Second)
    if err != nil {
        return nil, err
    }
```

https://pkg.go.dev/<nome-do-repositorio>

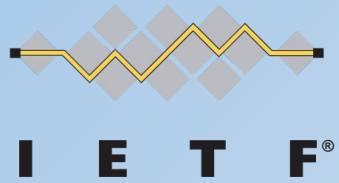


Como enviar um e-mail

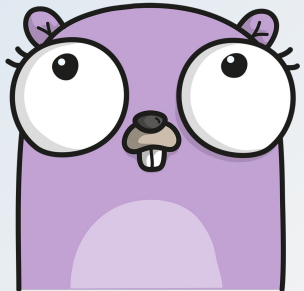
```
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err ≠ nil {
    panic(err)
}
```



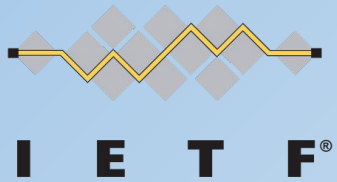
Quem determina como é um e-mail



Internet Engineering
Task Force

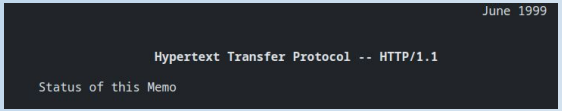


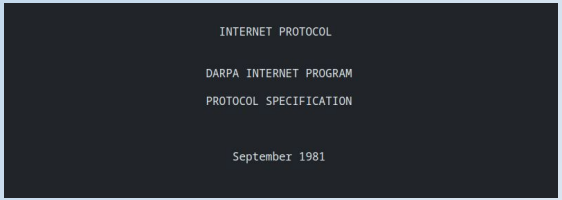
Quem determina como é um e-mail

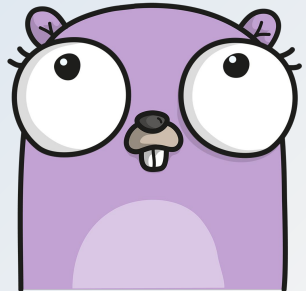


Internet Engineering Task Force

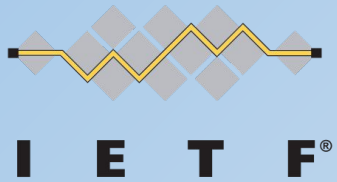
RFCs







Quem determina como é um e-mail



Internet Engineering Task Force

RFCs

Network Working Group — P. Resnick, Editor
Request for Comments: 2822 — QUALCOMM Incorporated
Obsoletes: 822 — April 2001
Category: Standards Track

Internet Message Format

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

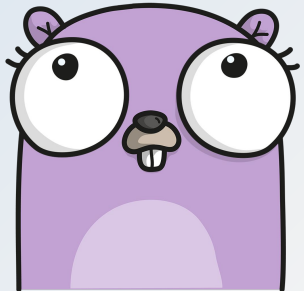
Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

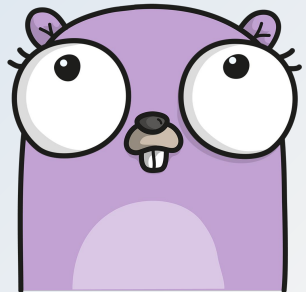
This standard specifies a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This standard supersedes the one specified in Request For Comments (RFC) 822, "Standard for the Format of ARPA Internet Text Messages", updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs.

Table of Contents



Como enviar um e-mail

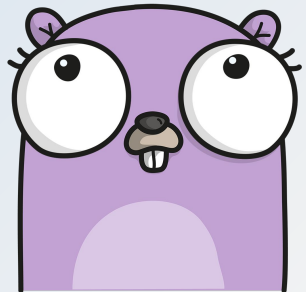
```
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err ≠ nil {
    panic(err)
}
```



Como enviar um e-mail

Criar uma mensagem

```go
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err ≠ nil {
    panic(err)
}
```

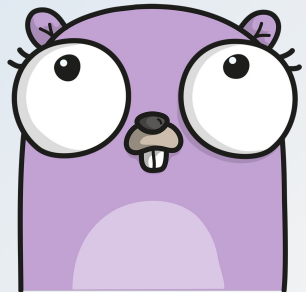


Como enviar um e-mail

Criar uma mensagem

Criar um dialer

```go
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err ≠ nil {
    panic(err)
}
```



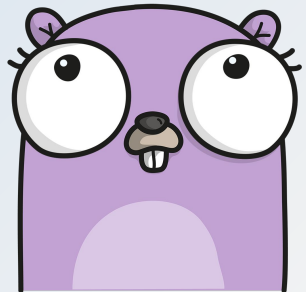
Como enviar um e-mail

Criar uma mensagem

Criar um dialer

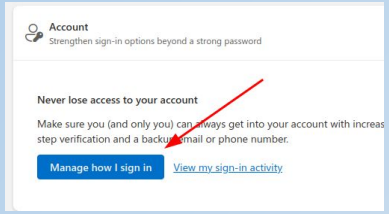
Enviar mensagem

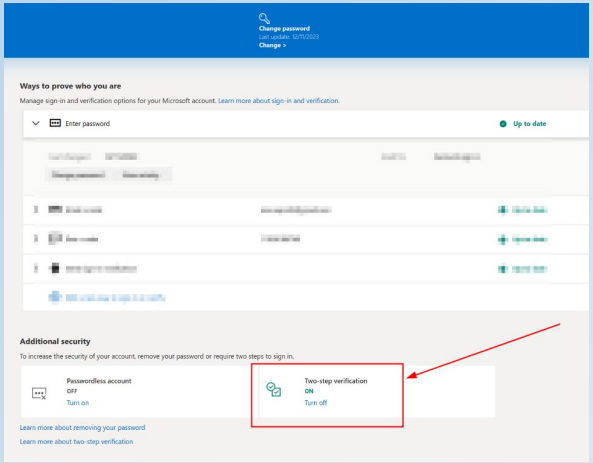
```go
m := gomail.NewMessage()
m.SetHeader("From", "alex@example.com")
m.SetHeader("To", "bob@example.com", "cora@example.com")
m.SetAddressHeader("Cc", "dan@example.com", "Dan")
m.SetHeader("Subject", "Hello!")
m.SetBody("text/html", "Hello <b>Bob</b> and <i>Cora</i>!")
m.Attach("/home/Alex/lolcat.jpg")

d := gomail.NewDialer("smtp.example.com", 587, "user", "123456")

// Send the email to Bob, Cora and Dan.
if err := d.DialAndSend(m); err ≠ nil {
    panic(err)
}
```



Criar senha de acesso para App

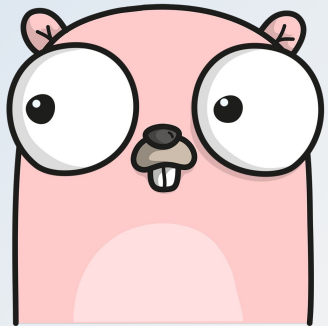
- Gmail: myaccount.google.com/apppasswords
- Outlook/Hotmail: account.microsoft.com/security





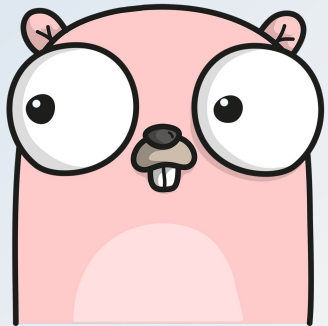
Criar o dialer

```
dialer := gomail.NewDialer("smtp.gmail.com", 587, <email>, <app-password>)
// or
dialer := gomail.NewDialer("smtp-mail.outlook.com", 587, <email>, <app-password>)
```



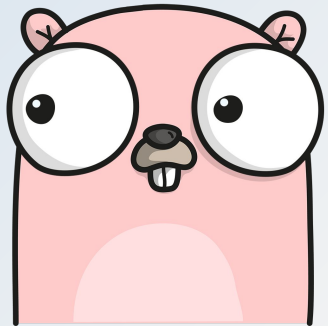
Criar a mensagem

```
message := gomail.NewMessage()
message.SetHeader("Subject", "You are a winner!")
message.SetHeader("From", <seu-email>)
message.SetHeader("To", <email-para-receber>)
message.SetBody("text/html", "Hello world!")
```



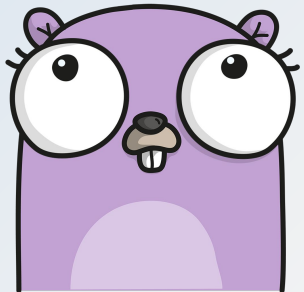
Enviar

```
dialer.DialAndSend(message)
```



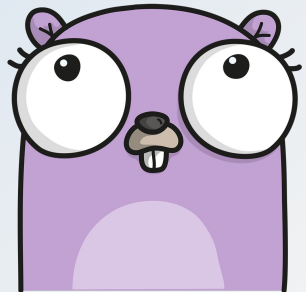
Como customizar uma mensagem de email

```
message := gomail.NewMessage()
message.SetHeader("Subject", "You are a winner!")
message.SetHeader("From", <seu-email>)
message.SetHeader("To", <email-para-receber>)
message.SetBody("text/html", "Hello world!")
```



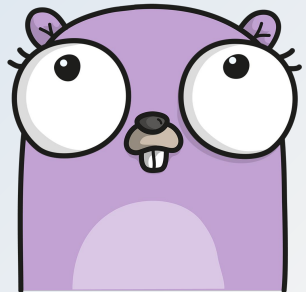
Como customizar uma mensagem de email

```
htmlString, err := os.ReadFile("email.html")
if err ≠ nil {
    log.Fatalf("Error reading email template: %v", err)
}

message.SetBody("text/html", string(htmlString))
```



Enviar para os ganhadores

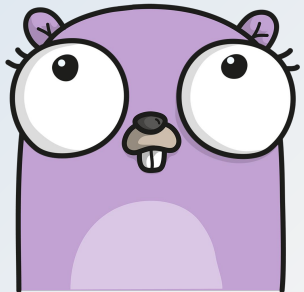
```
func extractEmails(winners [][]string) []string
```



Enviar para os ganhadores

```
func extractEmails(winners [][]string) []string
```

O email é o primeiro
elemento de cada linha



Obrigado!

GitHub

Repositório

LinkedIn

reneepc