ThoughtWorks®

construção e uso

# PRÁTICA DE CONJUNTOS

*Porquê e como implementar um tipo Set na linguagem Go.*

THE DEVELOPER'S CONFERENCE

**Luciano Ramalho**
**@ramalhoorg**

# OBJETIVOS

**1**

Demonstrar como e porque o uso de conjuntos pode simplificar a lógica.

**2**

Discutir alternativas para implementar coleções de tipo Set em Go.

ThoughtWorks®

# MOTIVAÇÃO

Quando senti saudade de conjuntos

# CASO DE USO #1

EXIBIR PRODUTO SE TODAS AS PALAVRAS DA CONSULTA APARE-CEREM NA DESCRIÇÃO

# SOLUÇÃO SEM CONJUNTOS #1

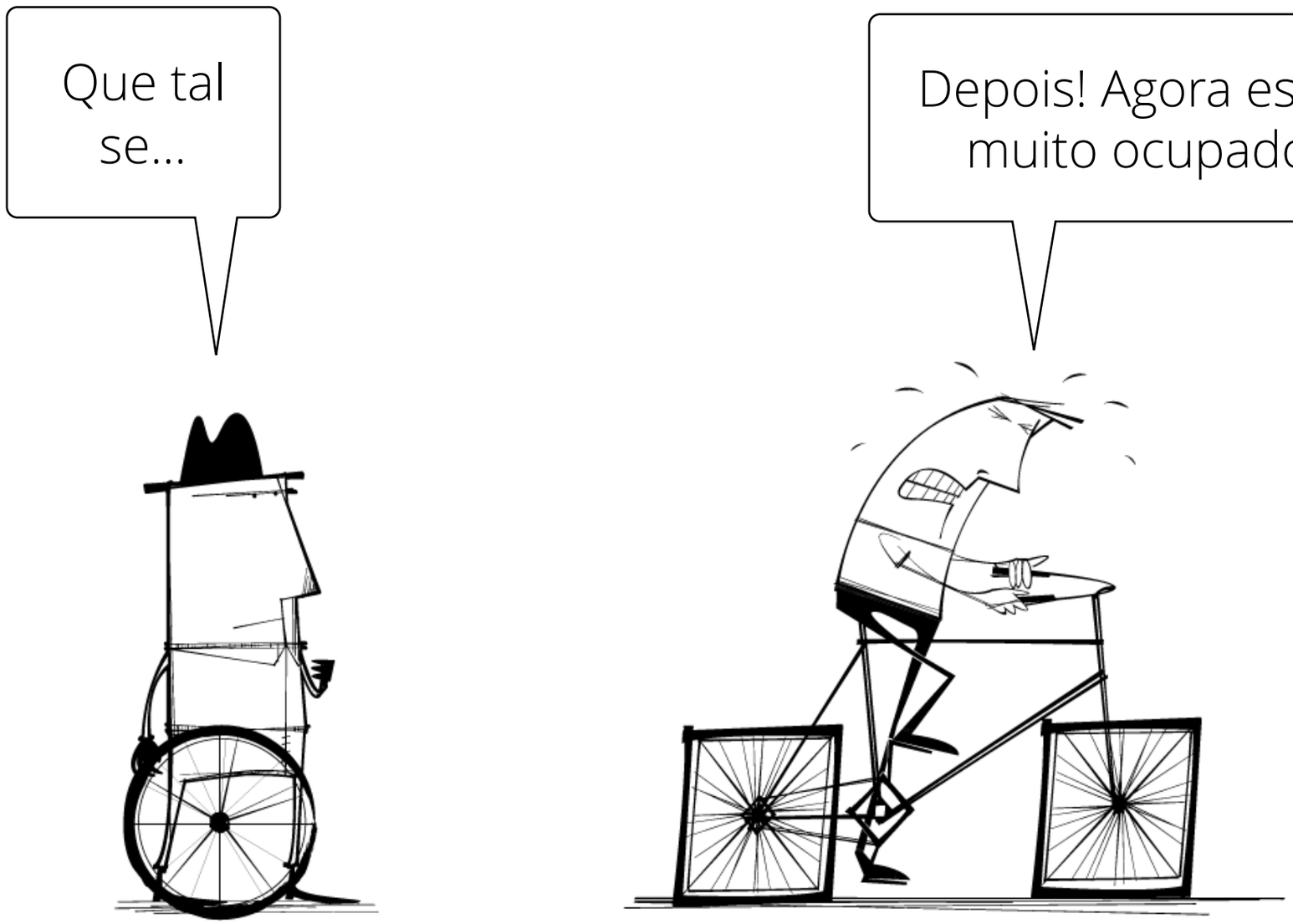
Já escrevi código assim...

```
func ContainsAll(slice, subslice []string) bool {
    for _, needle := range subslice {
        found := false
        for _, elem := range slice {
            if needle == elem {
                found = true
                break
            }
        }
        if !found {
            return false
        }
    }
    return true
}
```

# SOLUÇÃO SEM CONJUNTOS #2

Assim fica menos mal...

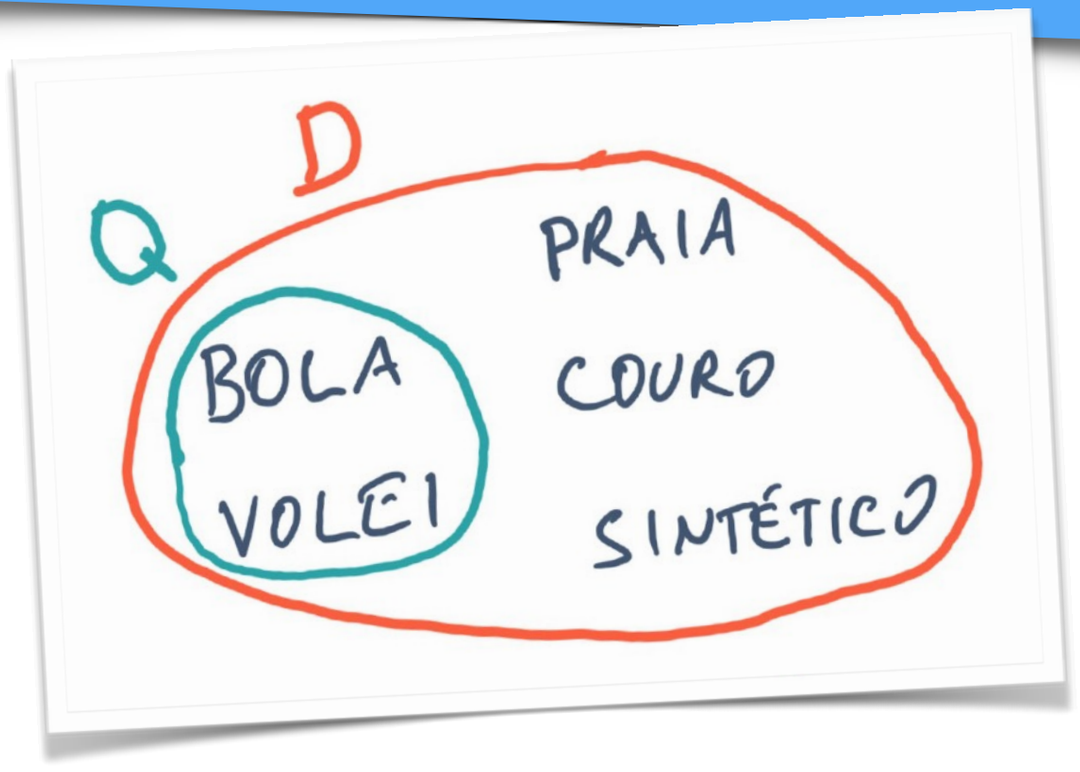
```go
func Contains(slice []string, needle string) bool {
    for _, elem := range slice {
        if needle == elem {
            return true
        }
    }
    return false
}

func ContainsAll(slice, subslice []string) bool {
    for _, needle := range subslice {
        if !Contains(slice, needle) {
            return false
        }
    }
    return true
}
```

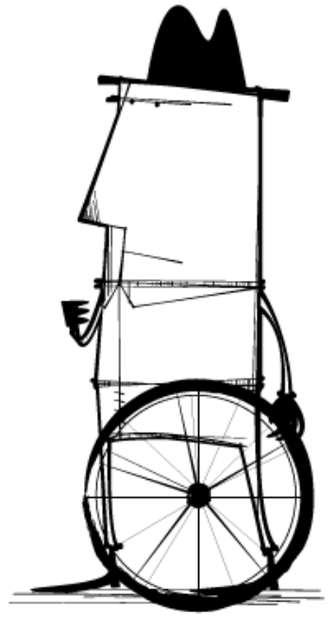

Que tal se...
Depois! Agora estou muito ocupado!

# CASO DE USO #1

*www.workcompass.com/*

EXIBIR PRODUTO SE TODAS AS PALAVRAS DA CONSULTA APARECEREM NA DESCRIÇÃO



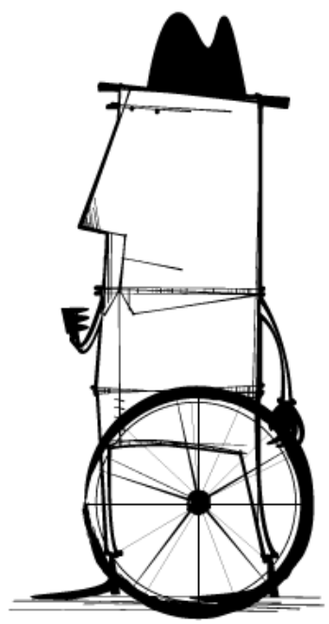



# CASO DE USO #1

*www.workcompass.com/*

EXIBIR PRODUTO SE TODAS AS PALAVRAS DA CONSULTA APARECEREM NA DESCRIÇÃO


$$Q \subset D$$

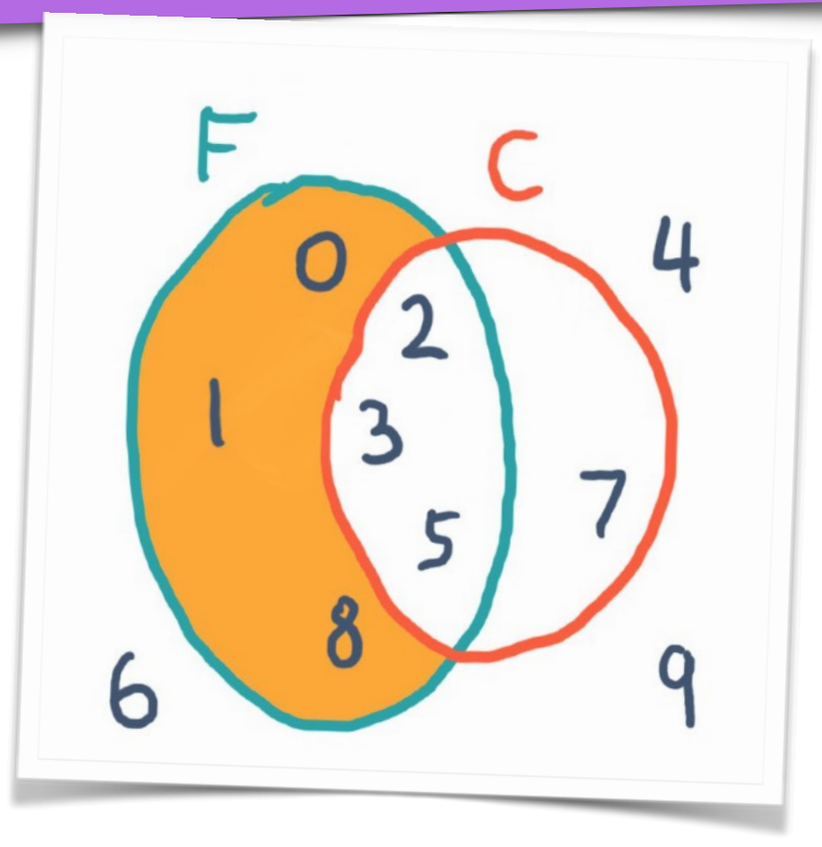


# CASO DE USO #2

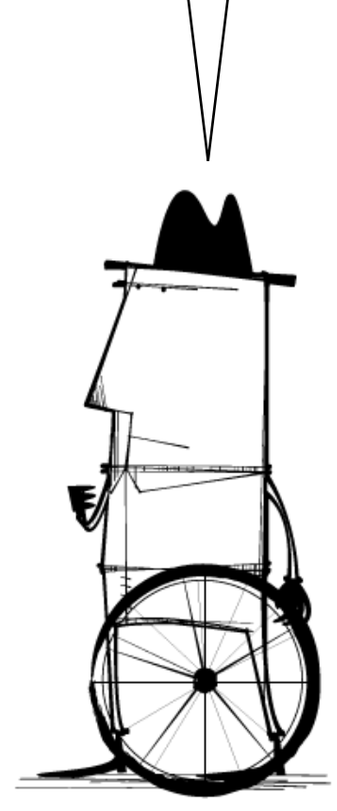
Marcar todos os produtos favoritados, exceto os que já estiverem no carrinho de compras.

# CASO DE USO #2

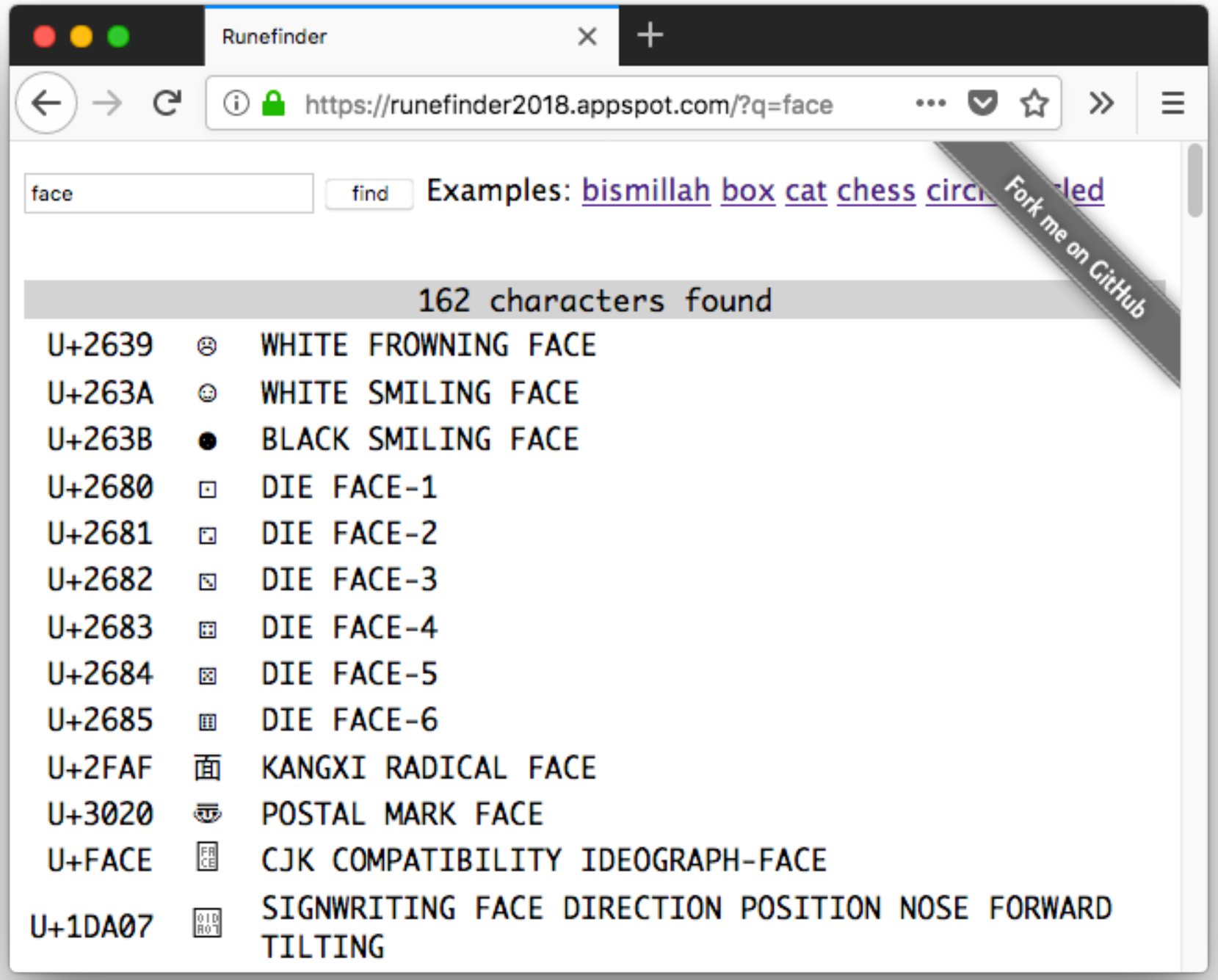
Marcar todos os produtos favoritados, exceto os que já estiverem no carrinho de compras.




$$F \setminus C$$

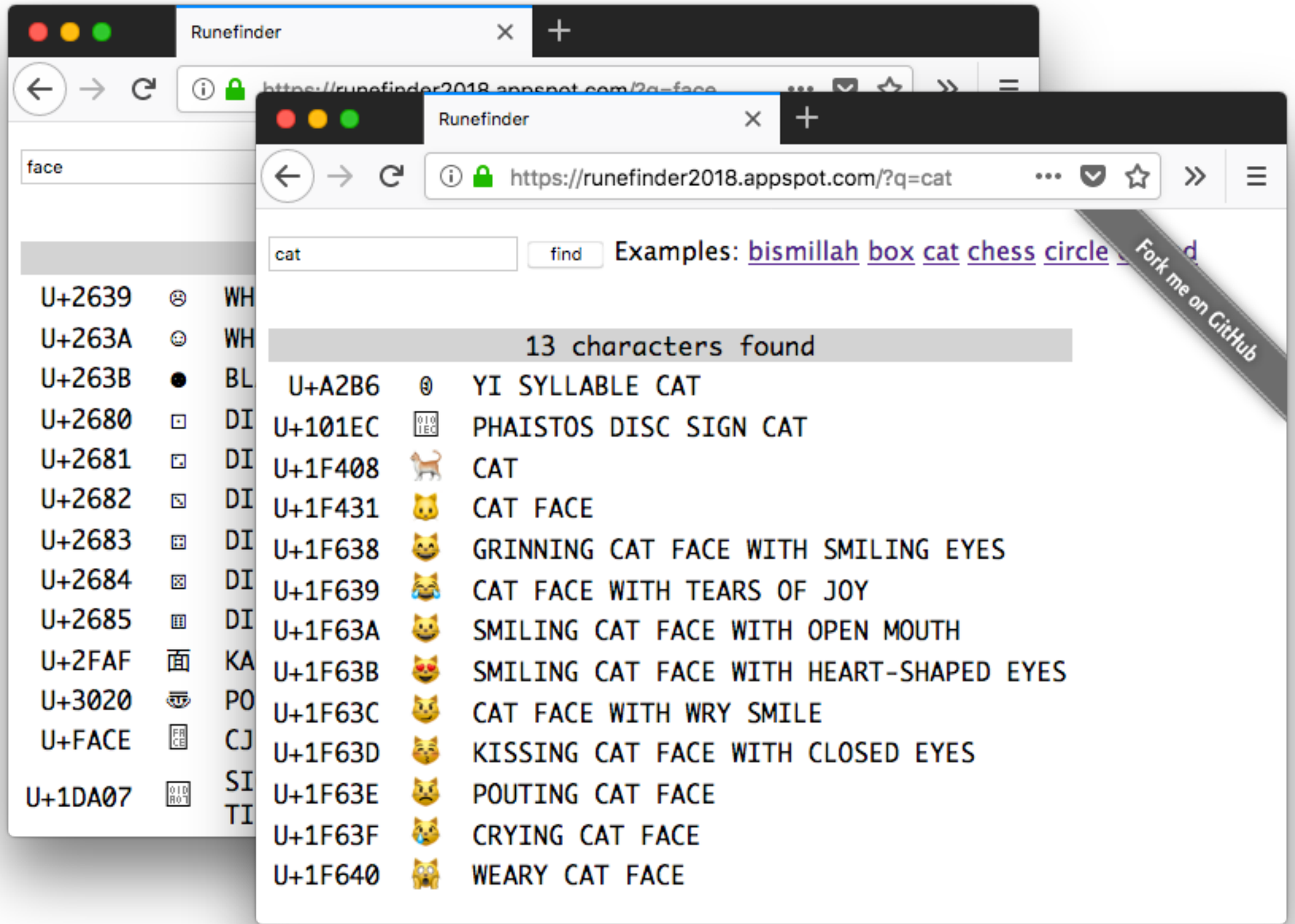


# CASO DE USO #3: RUNEFINDER



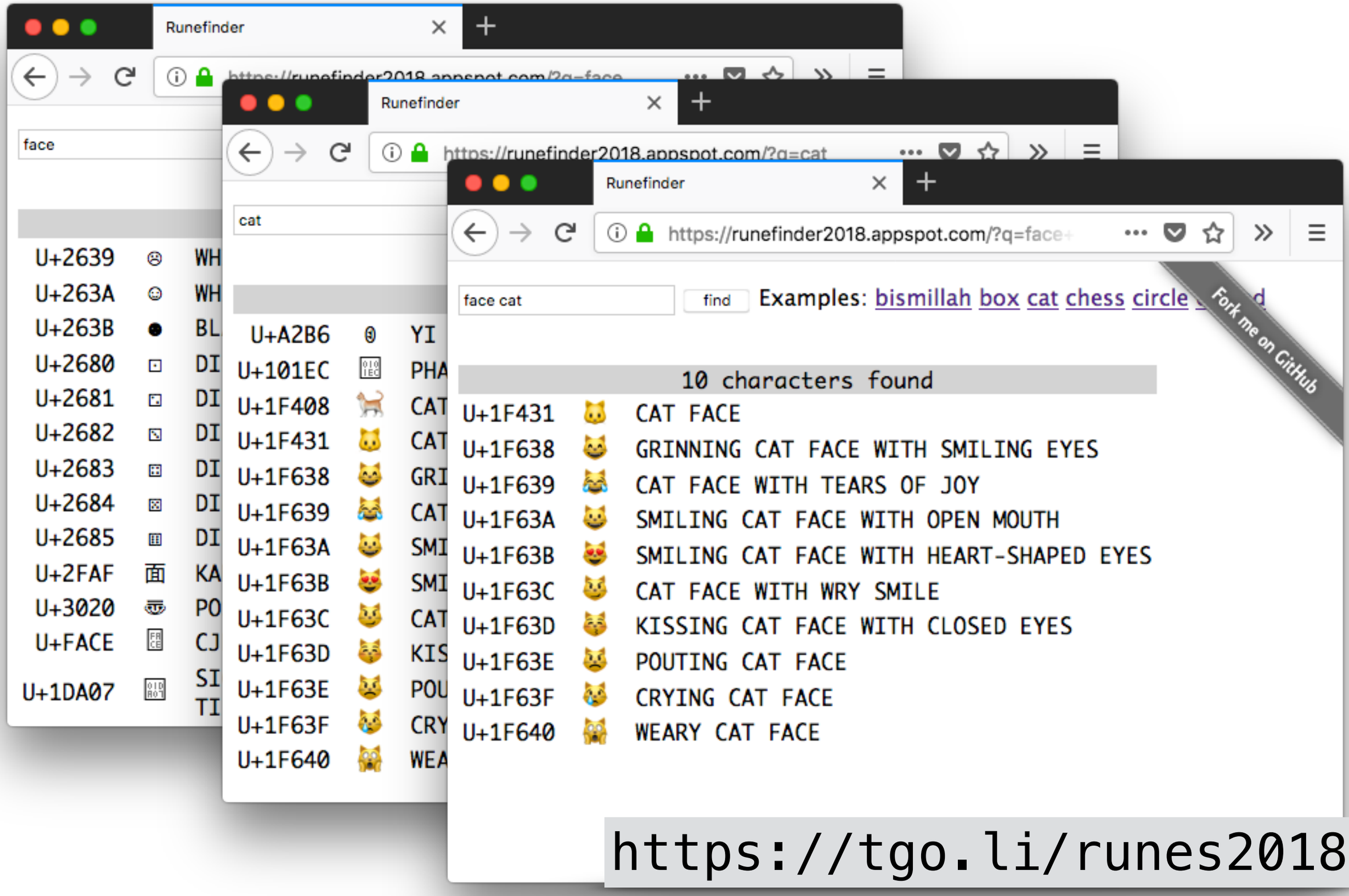

https://tgo.li/runes2018

# CASO DE USO #3: RUNEFINDER




https://tgo.li/runes2018

# CASO DE USO #3: RUNEFINDER



# CASO DE USO #3: ÍNDICE INVERTIDO

```
func Example_invertedIndex() {
    index := make(map[string]Set)
    index["SIGN"] = Make('#', '$', '%', '+', '<', '§', '©', '¬', '®', '°', '±', 'µ', '¶'
    index["REGISTERED"] = Make('®')
    index["CHESS"] = Make('♚', '♛', '♜', '♝', '♞', '♟', '♔', '♕', '♖', '♗', '♘', '♙')
    index["BLACK"] = Make('⚑', '■', '🖤', '★', '☎', '☻', '♚', '♛', '♜', '♝', '♞', '♟', '♠
    result := index["CHESS"].Intersection(index["BLACK"])
    fmt.Println(result)
    // Output:
    // Set{♚ ♛ ♜ ♝ ♞ ♟}
}
```

# CASO DE USO #3: ÍNDICE INVERTIDO

"REGISTERED SIGN" é o nome oficial de ® em UnicodeData.txt

```
func Example_invertedIndex() {
    index := make(map[string]Set)
    index["SIGN"] = Make('#', '$', '%', '+', '<', '§', '©', '¬', '®', '°', '±', 'µ', '¶'
    index["REGISTERED"] = Make('®')
    index["CHESS"] = Make('♚', '♛', '♜', '♝', '♞', '♟', '♔', '♕', '♖', '♗', '♘', '♙')
    index["BLACK"] = Make('⚑', '■', '🖤', '★', '☎', '☻', '♚', '♛', '♜', '♝', '♞', '♟', '♠
    result := index["CHESS"].Intersection(index["BLACK"])
    fmt.Println(result)
    // Output:
    // Set{♚ ♛ ♜ ♝ ♞ ♟}
}
```

# CASO DE USO #3: ÍNDICE INVERTIDO

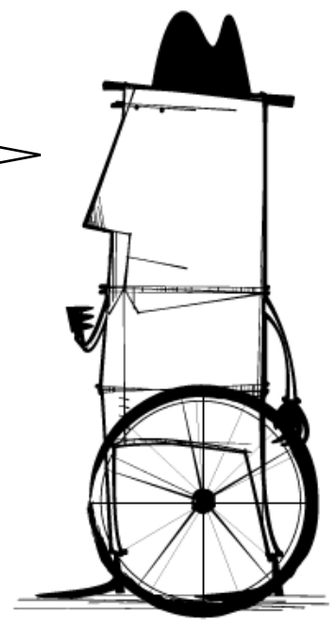
```
func Example_invertedIndex() {
    index := make(map[string]Set)
    index["SIGN"] = Make('#', '$', '%', '+', '<', '§', '©', '¬', '®', '°', '±', 'µ', '¶',
    index["REGISTERED"] = Make('®')
    index["CHESS"] = Make('♚', '♛', '♜', '♝', '♞', '♟', '♔', '♕', '♖', '♗', '♘', '♙')
    index["BLACK"] = Make('⚑', '■', '🖤', '★', '☎', '☻', '♚', '♛', '♜', '♝', '♞', '♟', '♠
    result := index["CHESS"].Intersection(index["BLACK"])
    fmt.Println(result)
    // Output:
    // Set{♚ ♛ ♜ ♝ ♞ ♝}
}
```

index é o índice invertido: as chaves são palavras, e os valores são conjuntos de runas cujo nome contém a palavra. Ex. a runa '®' aparece nas chaves "REGISTERED" e "SIGN"

# CASO DE USO #3: ÍNDICE INVERTIDO

```
func Example_invertedIndex() {
    index := make(map[string]Set)
    index["SIGN"] = Make('#', '$', '%', '+', '<', '§', '©', '¬', '®', '°', '±', 'µ', '¶'
    index["REGISTERED"] = Make('®')
    index["CHESS"] = Make('♚', '♛', '♜', '♝', '♞', '♟', '♔', '♕', '♖', '♗', '♘', '♙')
    index["BLACK"] = Make('⚑', '■', '🖤', '★', '☎', '☻', '♚', '♛', '♜', '♝', '♞', '♟', '♠
    result := index["CHESS"].Intersection(index["BLACK"])
    fmt.Println(result)
    // Output:
    // Set{♚ ♛ ♜ ♝ ♞ ♟}
}
```

$$A \cap B$$



ThoughtWorks®

# LÓGICA E CONJUNTOS

Uma relação íntima

Nobody has yet discovered a branch of mathematics that has successfully resisted formalization into set theory.

*Thomas Forster*
*Logic Induction and Sets, p. 167*

Ainda não descobriram um ramo da matemática que não possa ser formalizado pela teoria dos conjuntos.

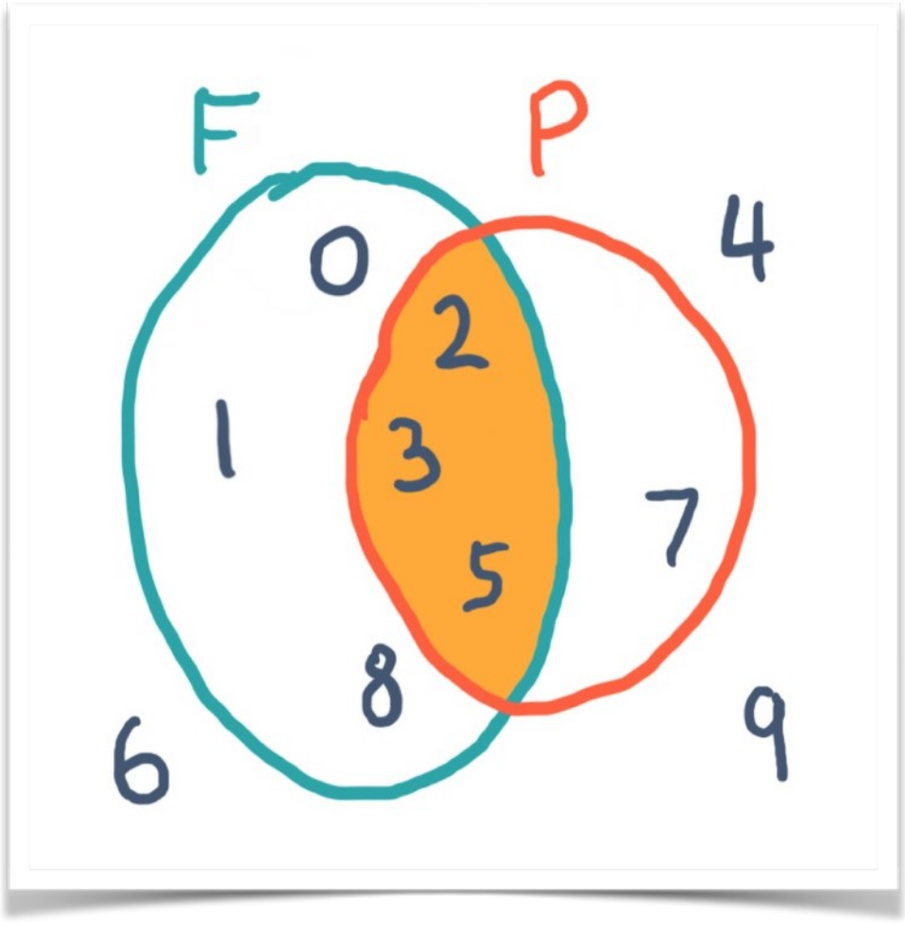
*Thomas Forster*
*(tradução livre de Logic Induction and Sets, p. 167)*

# CONJUNÇÃO LÓGICA: INTERSECÇÃO

*x pertence à intersecção de A e B.*

é o mesmo que:

*x pertence ao conjunto A e*
*x também pertence ao conjunto B.*




Em notação matemática:

$$x \in (A \cap B) \iff (x \in A) \land (x \in B)$$

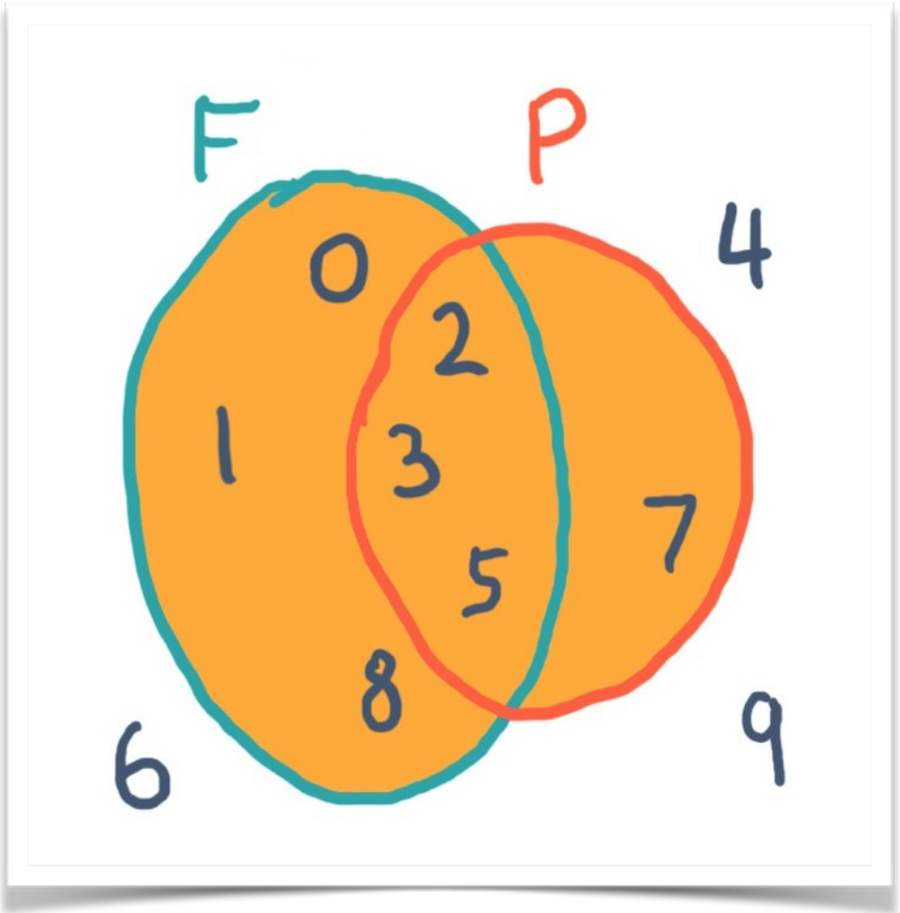
Em computação: **AND**

# DISJUNÇÃO LÓGICA: UNIÃO

*x pertence à união de A e B.*

é o mesmo que:

*x pertence ao conjunto A e/ou*
*x pertence ao conjunto B.*




Em notação matemática:

$$x \in (A \cup B) \iff (x \in A) \vee (x \in B)$$

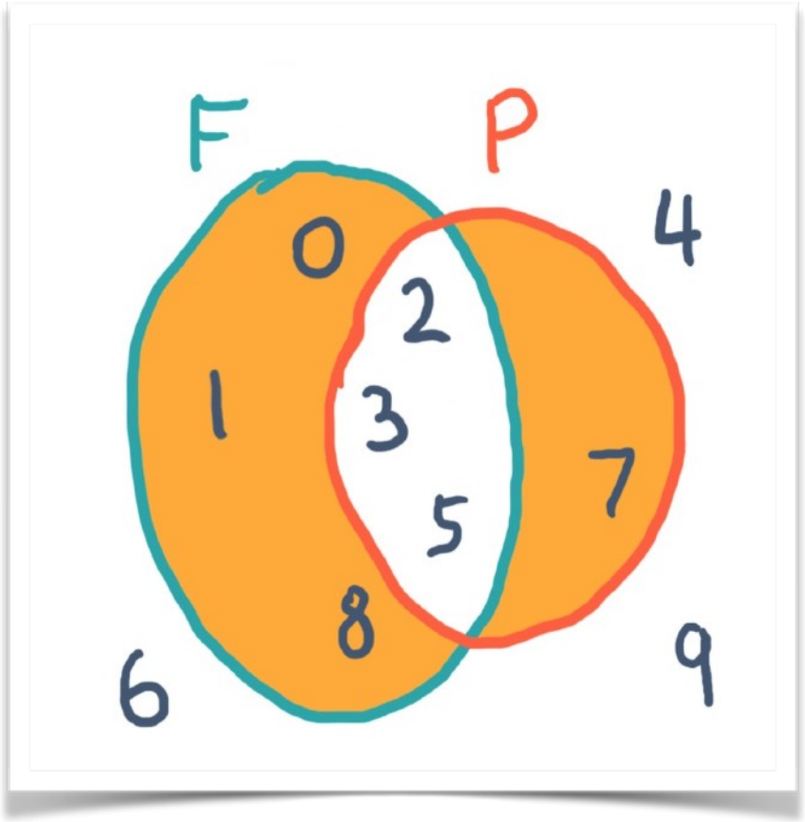
Em computação: **OR**

# DIFERENÇA SIMÉTRICA

*x pertence ao conjunto A ou*
*x pertence ao conjunto B*
*mas não pertence aos dois*

é o mesmo que:

*x pertence à união de A e B menos a intersecção de A e B.*




Em notação matemática:

$$x \in (A \ \Delta \ B) \iff (x \in A) \veebar (x \in B)$$

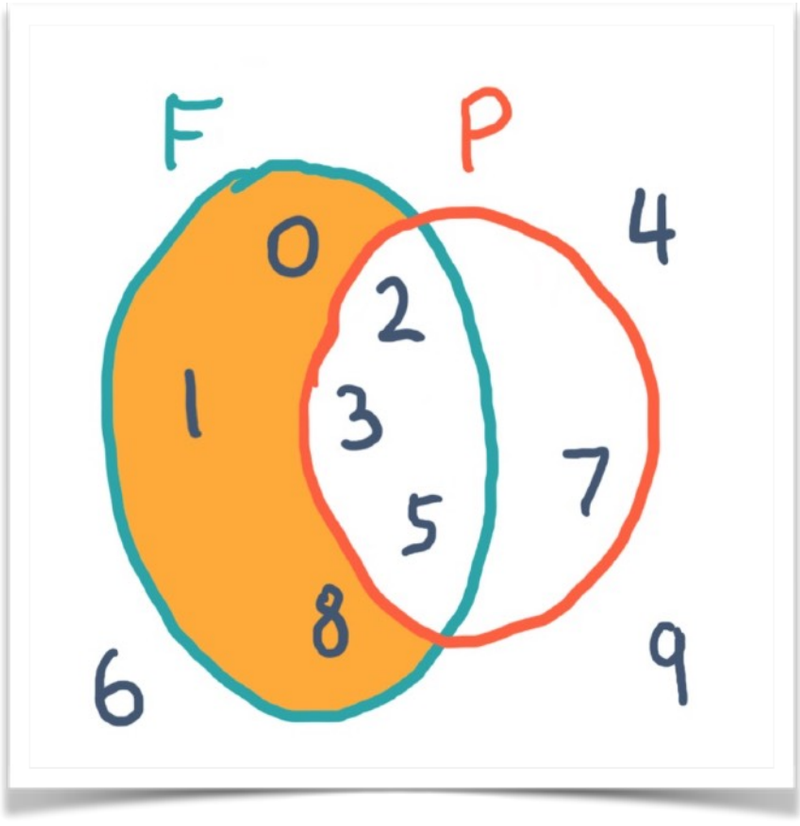
Em computação: **XOR**

# DIFERENÇA

*x pertence ao conjunto A mas não pertence ao conjunto B.*

é o mesmo que:

*elementos de A menos os de B*




Em notação matemática:

$$x \in (A \setminus B) \iff (x \in A) \wedge (x \notin B)$$

ThoughtWorks®

# CONJUNTOS EM VÁRIAS LINGUAGENS

# CONJUNTOS EM VÁRIAS BIBLIOTECAS-PADRÃO

Algumas linguagens/plataformas que implementam conjuntos em sua biblioteca-padrão.

| | |
|---|---|
| **Java** | Interface **Set** com < 10 métodos; 8 implementações |
| **Python** | **set** e **frozenset** com > 10 métodos e operadores |
| **.Net (C# etc.)** | Interface **ISet** com > 10 métodos; 2 implementações |
| **JavaScript (ES6)** | **Set** com < 10 métodos |
| **Ruby** | **Set** com > 10 métodos e operadores |

ThoughtWorks®

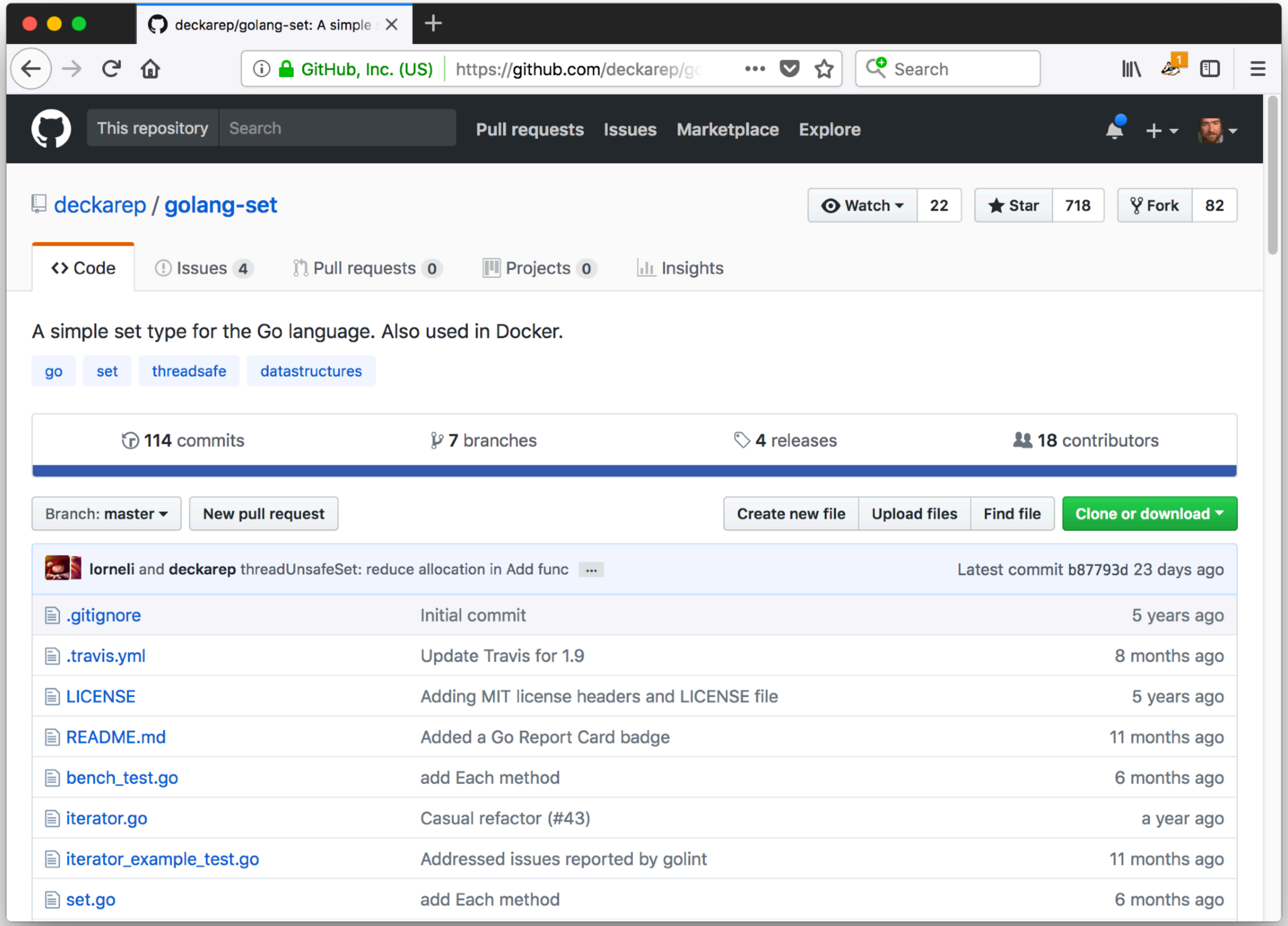
# CONJUNTOS "GENÉRICOS" EM GO

Elementos tipo **interface{}**

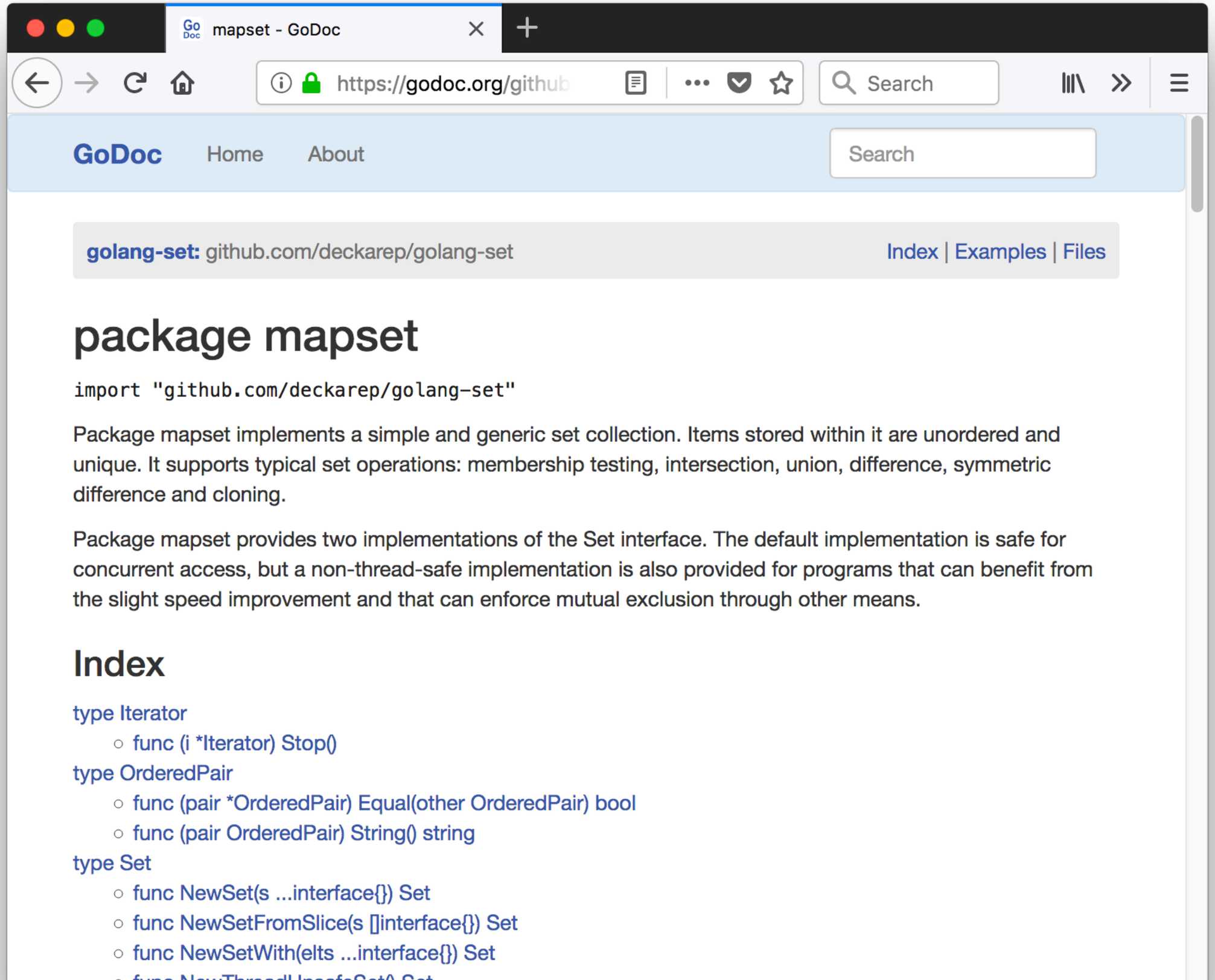
ThoughtWorks®

# GOLANG-SET

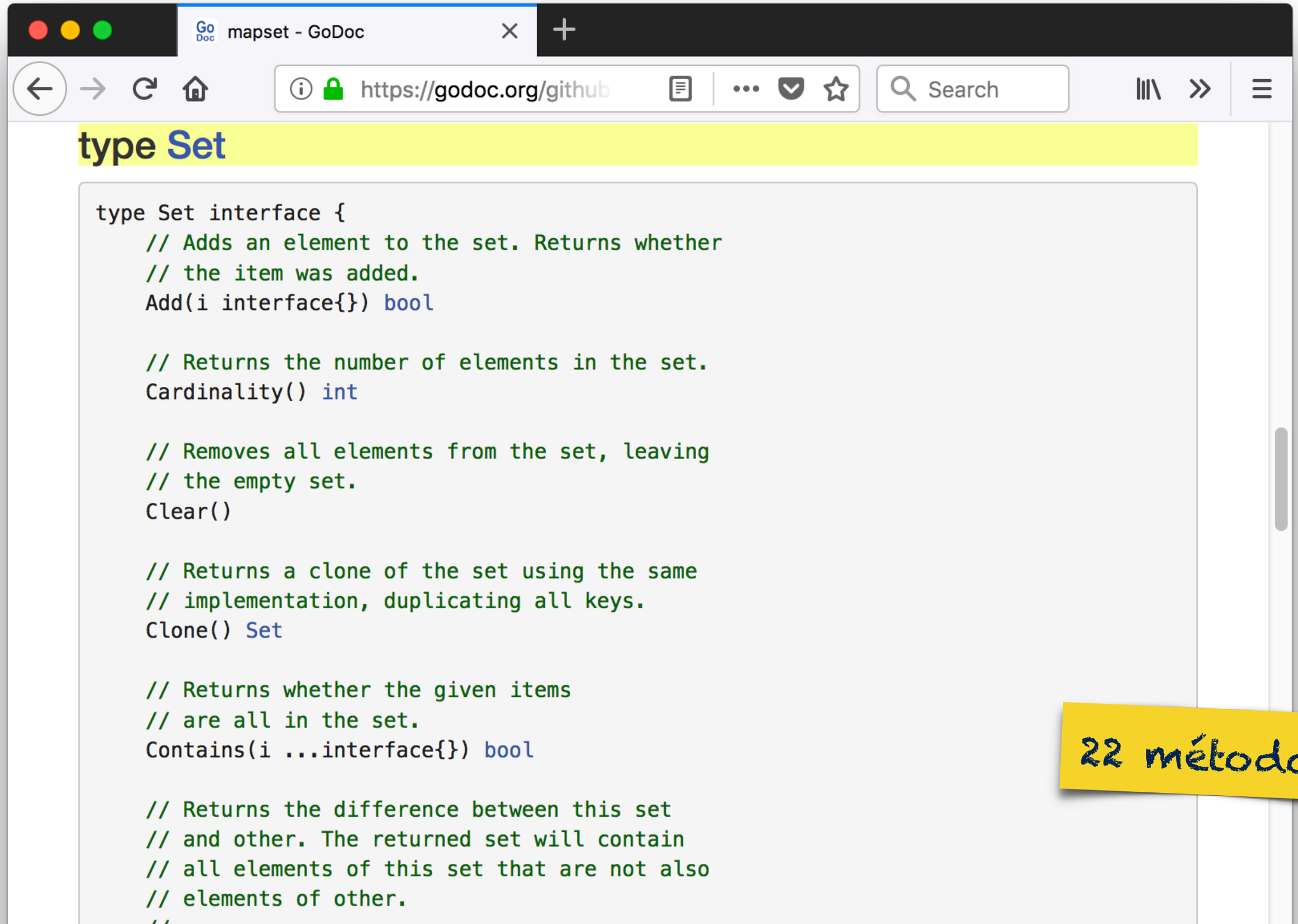
By Ralph Caraveo (@deckarep)


deckarep/golang-set: A simple
GitHub, Inc. (US)
https://github.com/deckarep/g
Search
This repository
Search
Pull requests
Issues
Marketplace
Explore
deckarep / golang-set
Watch 22
Star 718
Fork 82
Code
Issues 4
Pull requests 0
Projects 0
Insights
A simple set type for the Go language. Also used in Docker.
go
set
threadsafe
datastructures
114 commits
7 branches
4 releases
18 contributors
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
lorneli and deckarep threadUnsafeSet: reduce allocation in Add func
Latest commit b87793d 23 days ago
.gitignore Initial commit 5 years ago
.travis.yml Update Travis for 1.9 8 months ago
LICENSE Adding MIT license headers and LICENSE file 5 years ago
README.md Added a Go Report Card badge 11 months ago
bench_test.go add Each method 6 months ago
iterator.go Casual refactor (#43) a year ago
iterator_example_test.go Addressed issues reported by golint 11 months ago
set.go add Each method 6 months ago

# GOLANG-SET, VULGO MAPSET...



# GOLANG-SET: INTERFACE SET



# GOLANG-SET: IMPLEMENTAÇÕES

## threadUnsafeSet

```
type threadUnsafeSet map[interface{}]struct{}

func newThreadUnsafeSet() threadUnsafeSet {
    return make(threadUnsafeSet)
}
```

## threadSafeSet

```
type threadSafeSet struct {
    s threadUnsafeSet
    sync.RWMutex
}

func newThreadSafeSet() threadSafeSet {
    return threadSafeSet{s: newThreadUnsafeSet()}
}
```

# GOLANG-SET: CODE REVIEW

elementos tipo interface{}

```
func (set *threadUnsafeSet) Add(i interface{}) bool {
    _, found := (*set)[i]
    if found {
        return false //False if it existed already
    }

    (*set)[i] = struct{}{}
    return true
}

func (set *threadUnsafeSet) Contains(i ...interface{}) bool {
    for _, val := range i {
        if _, ok := (*set)[val]; !ok {
            return false
        }
    }
    return true
}
```

# GOLANG-SET: EXEMPLO

```go
func Example_invertedIndex() {
    index := make(map[string]mapset.Set)
    index["CHESS"] = mapset.NewSet('♚', '♛', '♜', '♝', '♞', '♟', '♔',
    index["BLACK"] = mapset.NewSet('⚑', '■', '🖤', '★', '☎', '☻', '♚',
    result := index["CHESS"].Intersect(index["BLACK"])
    // fmt.Println(result) outputs in random order:
    // Set{9818, 9820, 9821, 9823, 9819, 9822}
    // The next five lines make the result testable:
    list := []string{}
    for char := range result.Iter() {
        list = append(list, string(char.(rune)))
    }
    sort.Strings(list)
    fmt.Println(list)
    // Output:
    // [♚ ♛ ♜ ♝ ♞ ♟]
}
```

ThoughtWorks®

# GOLANG-SET: MORE CODE REVIEW

# GOLANG-SET: CODE REVIEW

```
func (set *threadUnsafeSet) Difference(other Set) Set {
    _ = other.(*threadUnsafeSet)

    difference := newThreadUnsafeSet()
    for elem := range *set {
        if !other.Contains(elem) {
            difference.Add(elem)
        }
    }
    return &difference
}
```

# GOLANG-SET: CODE REVIEW

```go
func (set *threadUnsafeSet) IsSubset(other Set) bool {
    _ = other.(*threadUnsafeSet)
    for elem := range *set {
        if !other.Contains(elem) {
            return false
        }
    }
    return true
}

func (set *threadUnsafeSet) IsProperSubset(other Set) bool {
    return set.IsSubset(other) && !set.Equal(other)
}
```

ThoughtWorks®

# GODS

Go Data Structures

elementos tipo interface{}

emirpasic/gods: GoDS (Go Data

GitHub, Inc. (US) https://github.com/emirpasic/g

Search

This repository Search Pull requests Issues Marketplace Explore

emirpasic / gods

Watch 188 Star 4,025 Fork 383

Code Issues 11 Pull requests 2 Projects 0 Wiki Insights

GoDS (Go Data Structures). Containers (Sets, Lists, Stacks, Maps, Trees), Sets (HashSet, TreeSet), Lists (ArrayList, SinglyLinkedList, DoublyLinkedList), Stacks (LinkedListStack, ArrayStack), Maps (HashMap, TreeMap, HashBidiMap, TreeBidiMap), Trees (RedBlackTree, AVLTree, BTree, BinaryHeap), Comparators, Iterators, Enumerables, Sort, JSON

go golang data-structure map tree set list stack iterator enumerable sort avl-tree red-black-tree b-tree binary-heap

275 commits 1 branch 13 releases 11 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

emirpasic committed on Dec 26, 2017 Merge pull request #69 from mahadevTW/indexOfMethod ... Latest commit b2394df on Dec 26, 2017

| | | |
|---|---|---|
| containers | Update enumerable.go | 7 months ago |
| examples | - serialization example | a year ago |
| lists | Add IndexOf method to DoublyLinkedList | 4 months ago |
| maps | - tree-map and tree-bidi-map (de)serialization | a year ago |
| sets | - JSON serialization for all sets | a year ago |

# GODS: PACKAGE SETS

2 implementações:
HashSest, TreeSet

sets - GoDoc

https://godoc.org/github.com/e

Search

**gods:** github.com/emirpasic/gods/sets

Index | Files | Directories

## package sets

```
import "github.com/emirpasic/gods/sets"
```

Package sets provides an abstract Set interface.

In computer science, a set is an abstract data type that can store certain values and no repeated values. It is a computer implementation of the mathematical concept of a finite set. Unlike most other collection types, rather than retrieving a specific element from a set, one typically tests a value for membership in a set.

Reference: https://en.wikipedia.org/wiki/Set_%28abstract_data_type%29

## Index ¶

type Set

### Package Files

sets.go

## type Set

```
type Set interface {
    Add(elements ...interface{})
    Remove(elements ...interface{})
    Contains(elements ...interface{}) bool

    containers.Container
}
```

interface:
7 métodos

containers - GoDoc

https://godoc.org/github

## type Container

```
type Container interface {
    Empty() bool
    Size() int
    Clear()
    Values() []interface{}
}
```

Container is base interface that all data structures implement.

ThoughtWorks®

# GEN

Type-driven code generation for Go

# GEN: GERADOR DE CÓDIGO COM TIPOS PARAMETRIZADOS

gen · type-driven code generati

clipperhouse.github.io/gen/

Search

home

- Quick start
- Annotations
- Usage
- FAQ

typewriters

- Listing
- Adding
- Implementing

slice

- Aggregate[T]
- All
- Any
- Average
- Average[T]
- Count
- Distinct
- DistinctBy
- First
- GroupBy[T]
- Max
- Max[T]
- MaxBy
- Min
- Min[T]
- MinBy

# gen (v4)

source · changelog

## Type-driven code generation for Go

**gen** is an attempt to bring some generics-like functionality to Go. It uses type annotations to add “of T” functionality to your packages.

gen generates code for your types, at development time, using the command line. It is not an import; the generated source becomes part of your package and takes no external dependencies.

## Quick start

Of course, start by installing Go, setting up paths, etc. Then:

```
go get github.com/clipperhouse/gen
```

Create a new Go project, and `cd` into it. Create a `main.go` file and define a type.

Now, mark it up with a `+gen` annotation in an adjacent comment like so:

```
// +gen slice:"Where,Count,GroupBy[string]"
type MyType struct {}
```

And at the command line, simply type:

```
gen
```

# GEN: MODO DE USAR

**1.** Marque o seu código com comentários especiais:

```go
// +gen set
type Codepoint rune


func (c Codepoint) String() string {
```

**2.** Execute o comando **gen**

**3.** Adicione ao seu projeto o arquivo .**go** gerado

`Exemplo em: https://tgo.li/2vtcMjw`

# GEN: MEU CÓDIGO FONTE (CODEPOINTS.GO)

```go
// Package cptset implements a set of Codepoints (a.k.a. runes)
package cptset

import (
    "bytes"
    "sort"
)

// +gen set
type Codepoint rune

func (c Codepoint) String() string {
    return string(c)
}

// ToStringSlice returns elements as a slice of strings
func (set CodepointSet) ToStringSlice() []string {
    var s []string
    for v := range set {
        s = append(s, string(v))
```

Exemplo em: https://tgo.li/2vtcMjw

# GEN: O ARQUIVO GERADO (CODEPOINT_SET.GO)

código derivado de golang-set

```go
// Generated by: main
// TypeWriter: set
// Directive: +gen on Codepoint

package cptset

// Set is a modification of https://github.com/deckarep/golang-set
// The MIT License (MIT)
// Copyright (c) 2013 Ralph Caraveo (deckarep@gmail.com)

// CodepointSet is the primary type that represents a set
type CodepointSet map[Codepoint]struct{}

// NewCodepointSet creates and returns a reference to an empty set.
func NewCodepointSet(a ...Codepoint) CodepointSet {
    s := make(CodepointSet)
    for _, i := range a {
        s.Add(i)
    }
```

Exemplo em: https://tgo.li/2vtcMjw

ThoughtWorks®

# CONJUNTOS ESPECIALIZADOS

ThoughtWorks®

# MAP CRU

# USO DE MAP COMO CONJUNTO

Chaves de um **map** formam um conjunto
com duas características essenciais:

- Garantia de unicidade:
  uma chave só pode ocorrer uma vez.
- Verificação de pertencimento em O(1):
  tempo praticamente constante independente do número de elementos.

```
if _, existe := conjunto[elem]; existe {
    // o elemento está no conjunto
}
```

Demais métodos? Fique à vontade para implementar!

# USO DE MAP COMO CONJUNTO

Curiosidade:
era assim que a gente se virava em Python até 2003 – usando chaves de um dict como um conjunto!

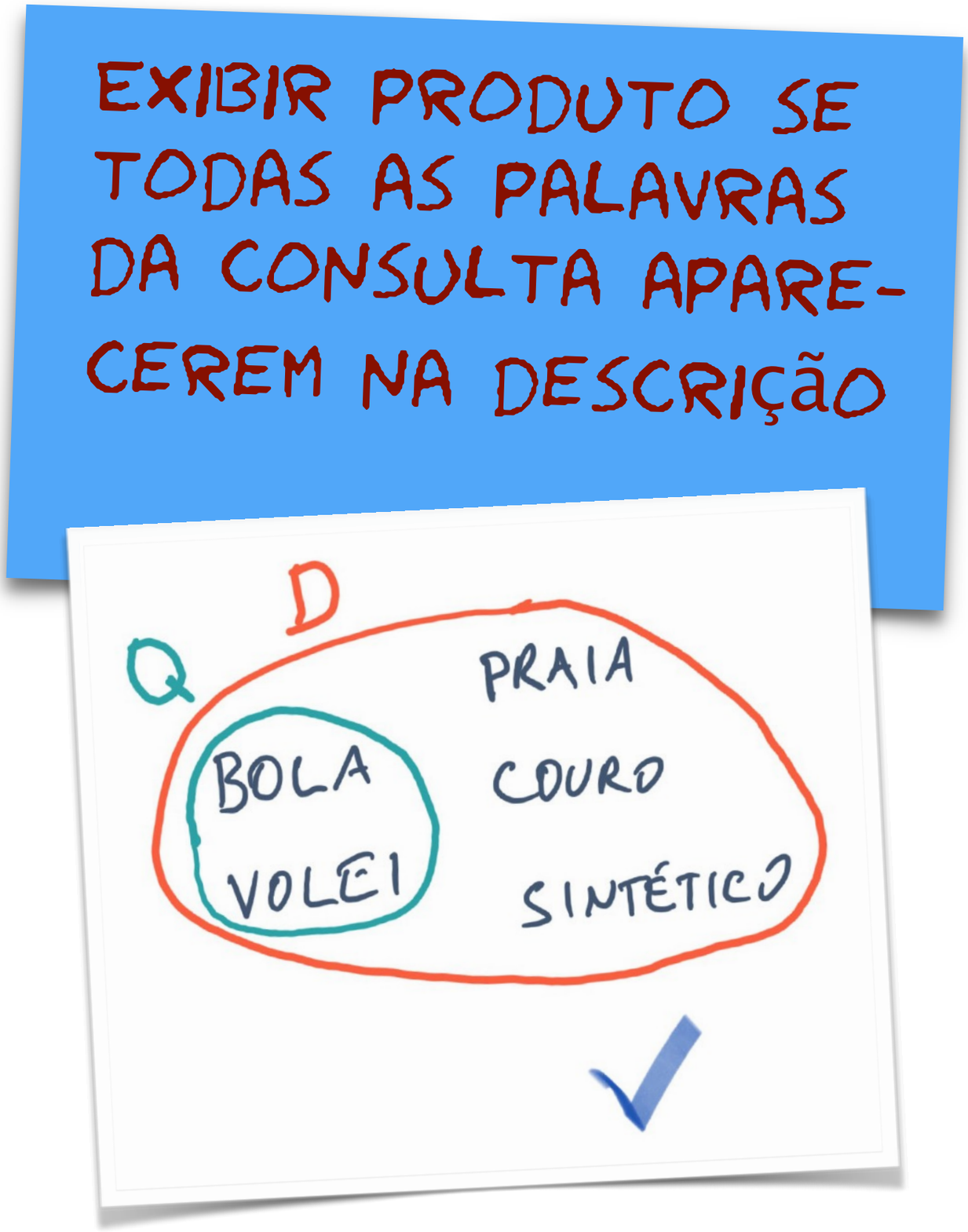
Chaves de um **map** formam um conjunto com duas características essenciais:

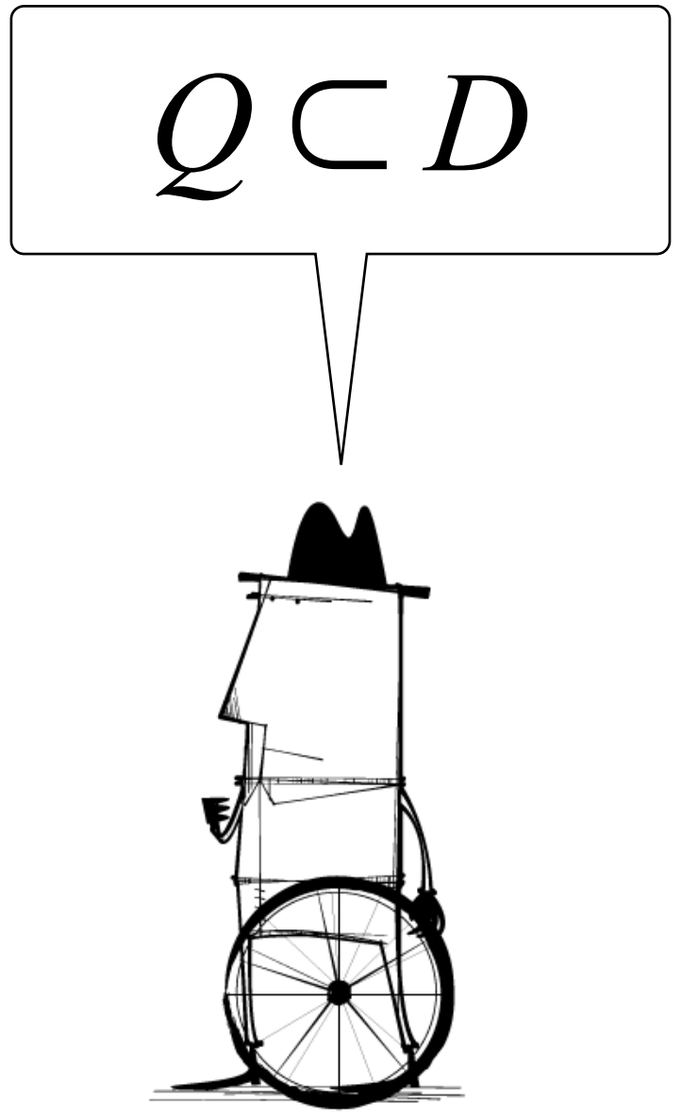
- Garantia de unicidade:
  uma chave só pode ocorrer uma vez.
- Verificação de pertencimento em O(1):
  tempo praticamente constante independente do número de elementos.

```
if _, existe := conjunto[elem]; existe {
    // o elemento está no conjunto
}
```

Demais métodos? Fique à vontade para implementar!

# RELEMBRANDO: CASO DE USO #1



# SOLUÇÃO COM MAP FAZENDO PAPEL DE CONJUNTO

Solução mais eficiente que as anteriores, para slices grandes:

```
func ContainsAll(slice, subslice []string) bool {
    set := make(map[string]struct{})
    for _, elem := range slice {
        set[elem] = struct{}{}
    }
    for _, needle := range subslice {
        if _, found := set[needle]; !found {
            return false
        }
    }
    return true
}
```

map de <tipo-elemento> para struct{}

# SOLUÇÃO COM MAP FAZENDO PAPEL DE CONJUNTO

Solução mais eficiente que as anteriores, para slices grandes:

```
func ContainsAll(slice, subslice []string) bool {
    set := make(map[string]struct{})
    for _, elem := range slice {
        set[elem] = struct{}{}
    }
    for _, needle := range subslice {
        if _, found := set[needle]; !found {
            return false
        }
    }
    return true
}
```

preencher o "conjunto"

# SOLUÇÃO COM MAP FAZENDO PAPEL DE CONJUNTO

Solução mais eficiente que as anteriores, para slices grandes:

```go
func ContainsAll(slice, subslice []string) bool {
    set := make(map[string]struct{})
    for _, elem := range slice {
        set[elem] = struct{}{}
    }
    for _, needle := range subslice {
        if _, found := set[needle]; !found {
            return false
        }
    }
    return true
}
```

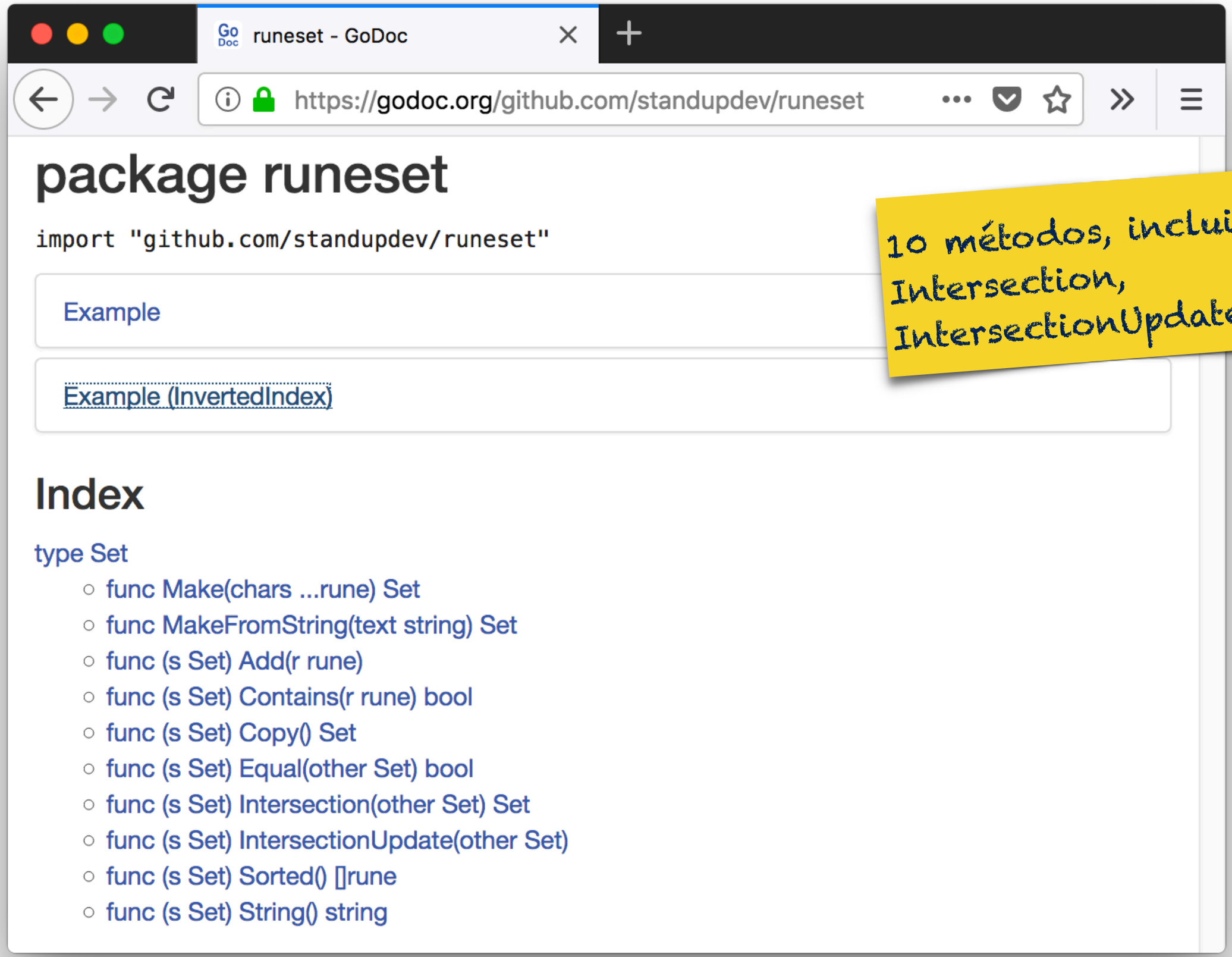
busca em O(1)

ThoughtWorks®

# MAP COM NOME

standupdev/runeset

# RUNESET: TIPO MAP COM NOME E MÉTODOS



# RUNESET: CARACERÍSTICAS

Não encapsula o map: pode acessado e modificado direto.

Sintaxe melhor para criar e testar elementos: `s.Add(e)`
`s.Contains(e)`

Exemplo: `https://github.com/standupdev/runeset`

# RUNESET: DEFINIÇÃO DO TIPO E MÉTODOS ESSENCIAIS

```go
// Set represents a set of runes
type Set map[rune]struct{}

// Add a rune to the set.
func (s Set) Add(r rune) {
    s[r] = struct{}{} // zero-byte struct
}

// Contains reports whether set contains given rune
func (s Set) Contains(r rune) bool {
    _, found := s[r]
    return found
}
```

elementos do tipo rune

https://github.com/standupdev/runeset

# RUNESET: CONSTRUTORES

Convenção:

Usar **`Make`** para criar objetos manipulados por referências.

cria um Set vazio

```go
// Make creates and returns a new Set
func Make(chars ...rune) Set {
    s := Set{}
    for _, c := range chars {
        s.Add(c)
    }
    return s
}

// MakeFromString creates and returns a new Set
func MakeFromString(text string) Set {
    return Make([]rune(text)...)
}
```

https://github.com/standupdev/runeset

# RUNESET: CONSTRUTORES

`MakeFromString`: set de runas a partir de string

```go
// Make creates and returns a new Set
func Make(chars ...rune) Set {
    s := Set{}
    for _, c := range chars {
        s.Add(c)
    }
    return s
}

// MakeFromString creates and returns a new Set
func MakeFromString(text string) Set {
    return Make([]rune(text)...)
}
```

custo zero para passar uma slice: runas...

# RUNESET: INTERSECÇÃO

```go
// Intersection returns a new set: the intersection of s AND other
func (s Set) Intersection(other Set) Set {
    result := Set{}
    if len(other) > 0 {
        for r := range s {
            if other.Contains(r) {
                result.Add(r)
            }
        }
    }
    return result
}

// IntersectionUpdate changes receiver in-place, keeping only
// elements that are in it AND in other.
func (s Set) IntersectionUpdate(other Set) {
    for r := range s {
        if !other.Contains(r) {
            delete(s, r)
        }
    }
}
```

https://github.com/standupdev/runeset

# RUNESET: USO EM ÍNDICE INVERTIDO (CASO DE USO #3)

```
func Example_invertedIndex() {
    index := make(map[string]Set)
    index["SIGN"] = Make('#', '$', '%', '+', '<', '§', '©', '¬', '®', '°', '±', 'µ', '¶'
    index["REGISTERED"] = Make('®')
    index["CHESS"] = Make('♚', '♛', '♜', '♝', '♞', '♟', '♔', '♕', '♖', '♗', '♘', '♙')
    index["BLACK"] = Make('⚑', '■', '🖤', '★', '☎', '☻', '♚', '♛', '♜', '♝', '♞', '♟', '♠
    result := index["CHESS"].Intersection(index["BLACK"])
    fmt.Println(result)
    // Output:
    // Set{♚ ♛ ♜ ♝ ♞ ♟}
}
```

# RUNESET: MÉTODOS ÚTEIS

---

Além de operações da teoria dos conjuntos...

`MakeFromString`: construtor que aceita string.

`Sorted`: devolve slice de runas em ordem ascendente

`String`: elementos ordenados para facilitar exemplos.

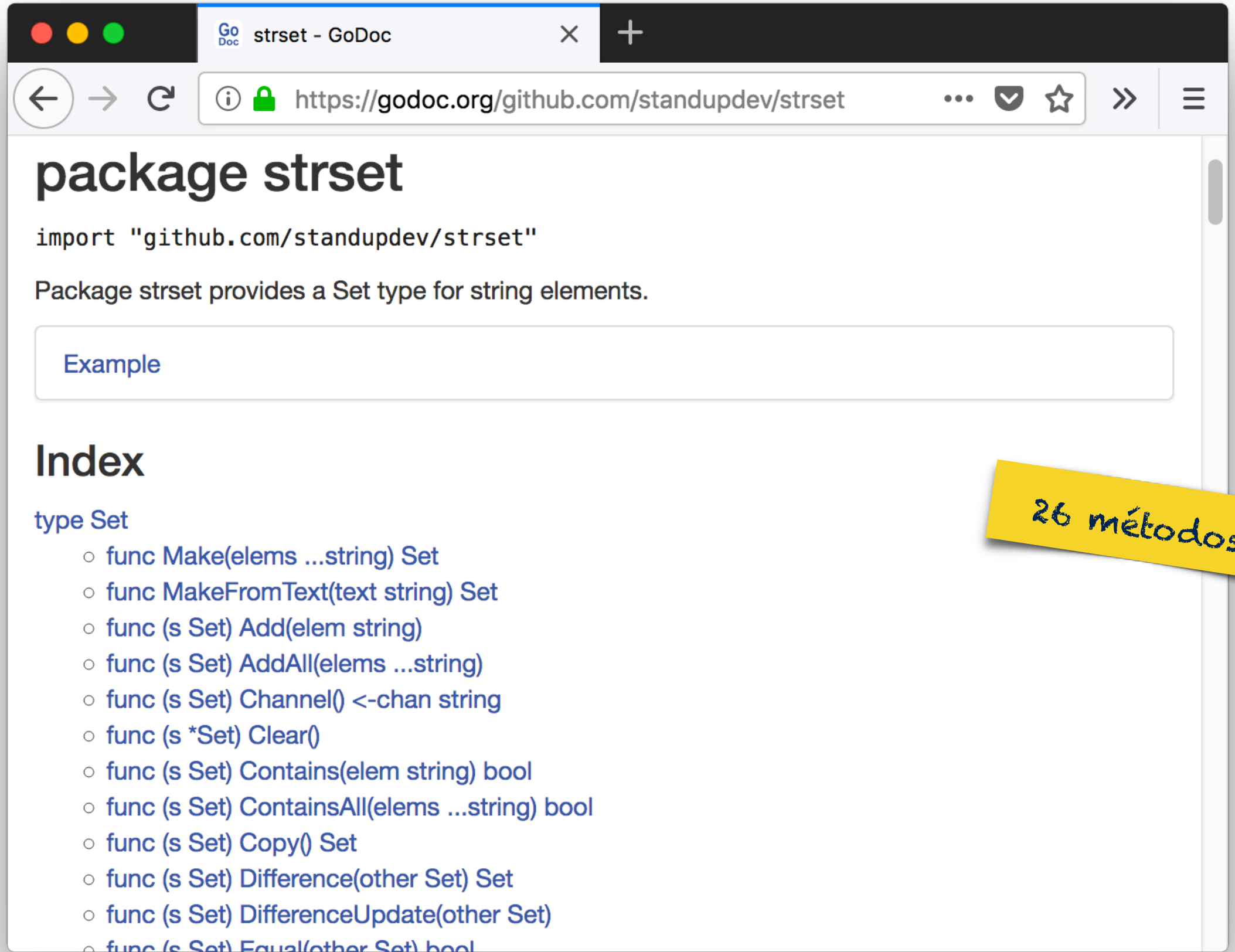
`Equal`: facilita muito os testes.

`Copy`: importante especialmente em coleções mutáveis.

ThoughtWorks®

# MAP ENCAPSULADO

# STRSET: TIPO MAP ENCAPSULADO EM STRUCT



# STRSET: CARACTERÍSTICAS

Protege o map: campo privado só pode acessado via métodos e funções do mesmo pacote.

Não pode ser construído com `Set{}`: necessário usar `Make()`.

Exemplo: `https://github.com/standupdev/strset`

# DEFINIÇÃO DO TIPO E CONSTRUTORES

```go
// Set of strings.
type Set struct {
    store map[string]struct{}
}

// Make creates and returns a new Set.
func Make(elems ...string) Set {
    s := Set{}
    s.store = make(map[string]struct{})
    for _, elem := range elems {
        s.store[elem] = struct{}{}
    }
    return s
}

// MakeFromText creates and returns a new Set from
// a string of elements separated by whitespace.
func MakeFromText(text string) Set {
    return Make(strings.Fields(text)...)
}
```

https://github.com/standupdev/strset

# ORGANIZAÇÃO DO PACOTE STRSET

| | |
|---|---|
| **core.go** | Make, Len, Contains, String, Equal, Copy, ToSlice, Channel |
| **operators.go** | Intersection, Union, Difference, SymmetricDifference |
| **relations.go** | SubsetOf, SupersetOf |
| **updaters.go** | Add, AddAll, Remove, RemoveAll, Pop, Clear, IntersectionUpdate, UnionUpdate, DifferenceUpdate, SymmetricDifferenceUpdate |

Nota: todos os métodos que modificam um `Set` depois de criado estão no arquivo `updaters.go`.

`https://github.com/standupdev/strset`

# COMPARAR APIS PARA ESCOLHER NOMES DE MÉTODOS

API-comparison.ods

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **Math symbol** | **Python operator** | **Python method** | **Java Set interface** | **gopl.io intset** | **emirpasic/ GoDS** | **deckarep/ golang-set** | **standupdev/ strset** |
| 2 | | | | | | | | |
| 3 | S ∪ Z | s \| z | s.union(it, …) | | | | s.Union(z) | s.Union(z) |
| 4 | | s =\| z | s.update(it, …) | | s.UnionWith(z) | | | s.UnionUpdate(z) |
| 5 | | | | s.addAll(c) | s.AddAll(e…) | s.Add(e…) | | s.AddAll(e…) |
| 6 | | | s.add(e) | s.add(e) | s.Add(e) | s.Add(e…) | s.Add(e) | s.Add(e) |
| 7 | | | | | | | | |
| 8 | S ∩ Z | s & z | s.intersection(it, …) | | | | s.Intersection(z) | s.Intersection(z) |
| 9 | | s &= z | s.intersection_update(it, …) | s.retainAll(c) | IntersectWith | | | s.IntersectionUpdate(z) |
| 10 | | | | | | | | |
| 11 | S \ Z | s - z | s.difference(it, …) | | | | s.Difference(z) | s.Difference(z) |
| 12 | | s -= z | s.difference_update(it, …) | s.removeAll(c) | s.DifferenceWith(z) | | | s.DifferenceUpdate(z) |
| 13 | | | | | | | | s.RemoveAll(e…) |
| 14 | S Δ Z | s ^ z | s.symmetric_difference(it) | | | | | s.SymmetricDifference(z) |
| 15 | | s ^= z | s.symmetric_difference_update | | s.SymmetricDifference(z) | | s.SymmetricDifference(z) | s.SymmetricDifferenceUpdate(z) |
| 16 | | | | | | | | |
| 17 | e ∈ S | e in s | s.__contains__(e) | s.contains(e) | s.Has(e) | | | s.Contains(e) |
| 18 | | | | s.containsAll(c) | | s.Contains(e… | s.Contains(e…) | s.ContainsAll(e…) |
| 19 | | | | | | | | |
| 20 | S = Z | s == z | s.__eq__(z) | s.equals(o) | | | s.Equal(z) | s.Equal(z) |
| 21 | S ⊆ Z | s <= z | s.issubset(it) | | | | s.IsSubset(z) | s.SubsetOf(z) |
| 22 | S ⊂ Z | s < z | s.__lt__(z) | | | | s.IsProperSubset(z) | |
| 23 | S ⊇ Z | s >= z | s.issuperset(it) | | | | | s.SupersetOf(z) |
| 24 | S ⊃ Z | s > z | s.__gt__(z) | | | | s.IsProperSuperset(z) | |
| 25 | | | | | | | | |
| 26 | | | s.isdisjoint(z) | | | | | |
| 27 | S × Z | | | | | | s.CartesianProduct(z) | |
| 28 | | | | | | | s.PowerSet() | |
| 29 | | | | | | | | |
| 30 | | len(s) | s.__len__() | s.size() | s.Len() | s.Size() | s.Cardinality() | s.Len() |
| 31 | | not s | s.__bool__() | | | s.Empty() | | |
| 32 | | | s.__iter__() | s.toArray() | s.Elems() | s.Values() | s.ToSlice() | s.ToSlice() |
| 33 | | | | | | | | |
| 34 | | | s.clear() | s.clear() | s.Clear() | s.Clear() | s.Clear() | s.Clear() |
| 35 | | | s.copy() | | s.Copy() | | s.Clone() | s.Copy() |

Sheet1

# RUNESCAN: EXEMPLO DE USO DE STRSET

```go
// ParseLine parses a line in the UnicodeData.txt file returning
// the rune, the name and a set of words build from the name.
func ParseLine(line string) (rune, string, strset.Set) {
    fields := strings.Split(line, ";")
    code, _ := strconv.ParseInt(fields[0], 16, 32)
    name := fields[1]
    wordStr := strings.Replace(fields[1], "-", " ", -1)
    words := strset.MakeFromText(wordStr)
    if fields[10] != "" {
        name += fmt.Sprintf(" (%s)", fields[10])
        wordStr = strings.Replace(fields[10], "-", " ", -1)
        words.AddAll(strings.Fields(wordStr)...)
    }
    return rune(code), name, words
}
```

```
// filter returns a list where each item is a [3]string with the
// U+XXXX codepoint, the character (as a string) and the name of the
// Unicode characters whose name cointains all words in the query.
func filter(text io.Reader, query string) [][3]string {
    result := [][3]string{}
    query = strings.Replace(query, "-", " ", -1)
    terms := strset.MakeFromText(strings.ToUpper(query))
    scanner := bufio.NewScanner(text)
    for scanner.Scan() {
        line := scanner.Text()
        if strings.TrimSpace(line) == "" {
            continue
        }
        char, name, nameWords := ParseLine(line)
        if terms.SubsetOf(nameWords) {
            result = append(result,
                [3]string{fmt.Sprintf("U+%04X", char),
                    string(char), name})
        }
    }
    return result
}
```

ThoughtWorks®

# CONCLUSÃO

# CONJUNTOS NA PRÁTICA

Operações de conjunto podem simplificar sua lógica

`golang-set` (@deckarep): a implementação *rica* mais popular

`gen` pode produzir um `golang-set` específico para seu tipo

Codar um tipo set é um ótimo exercício de programação

Se precisar de `Set` para elementos string, pode usar `strset`

```
https://github.com/standupdev/strset
```

Bom mesmo seria isso...

```
var s1 set[string]
s2 := make(set[uint64])
```

# MUITO GRATO!

*Luciano Ramalho*

*@ramalhoorg | @standupdev*

*luciano.ramalho@thoughtworks.com*

ThoughtWorks®