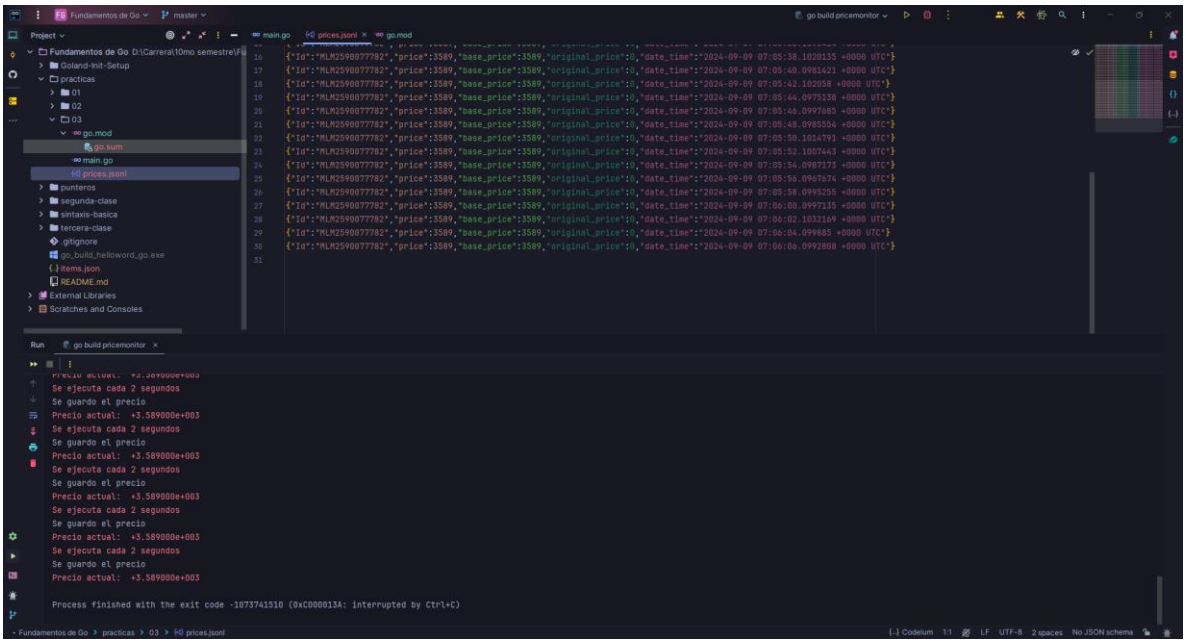
Fundamentos de Go
master
go build pricemonitor
Project
Fundamentos de Go D:\Carrera\10mo semestre\Fu
Goland-Init-Setup
practicas
01
02
03
go.mod
go.sum
main.go
prices.jsonl
punteros
segunda-clase
sintaxis-basica
tercera-clase
.gitignore
go_build_helloword_go.exe
items.json
README.md
External Libraries
Scratches and Consoles
main.go
prices.jsonl
go.mod
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:38.1020135 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:40.0981421 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:42.102058 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:44.0975138 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:46.0997685 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:48.0985534 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:50.1014791 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:52.1007443 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:54.0987173 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:56.0967674 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:05:58.0995255 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:06:00.0997135 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:06:02.103216Y +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:06:04.099885 +0000 UTC"}
{"Id":"MLM2590077782","price":3589,"base_price":3589,"original_price":0,"date_time":"2024-09-09 07:06:06.0992808 +0000 UTC"}
Run
go build pricemonitor
Se ejecuta cada 2 segundos
Se guardo el precio
Precio actual: +3.589000e+003
Se ejecuta cada 2 segundos
Se guardo el precio
Precio actual: +3.589000e+003
Se ejecuta cada 2 segundos
Se guardo el precio
Precio actual: +3.589000e+003
Se ejecuta cada 2 segundos
Se guardo el precio
Precio actual: +3.589000e+003
Se ejecuta cada 2 segundos
Se guardo el precio
Precio actual: +3.589000e+003
Process finished with the exit code -1073741510 (0xC000013A: interrupted by Ctrl+C)
Fundamentos de Go > practicas > 03 > prices.jsonl
Codeium
1:1
LF
UTF-8
2 spaces
No JSON schema

```go
package main

import (
    "encoding/json"
    "fmt"
    "github.com/robfig/cron/v3"
    "io"
    "net/http"
    "os"
    "time"
)

type PriceMonitor struct {  3 usages  new *
    Id            string
    Price         float64 `json:"price"`
    BasePrice     float64 `json:"base_price"`
    OriginalPrice float64 `json:"original_price"`
    DateTime      string  `json:"date_time"`
    httpClient    *http.Client
}

Codeium: Refactor | Explain | Docstring | ×
low complexity (33%)
func (p *PriceMonitor) Fetch() error {  2 usages  new *
    response, err := p.httpClient.Get("https://api.mercadolibre.com/items/" + p.Id)
    if err != nil {
        fmt.Println( a...: "error al obtener el precio del item")
        return err
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            fmt.Println( a...: "error al cerrar el body")
        }
    }(response.Body)
    body, err := io.ReadAll(response.Body)
    if err != nil {
        fmt.Println( a...: "error al leer el body")
        return err
    }

    err = json.Unmarshal(body, &p)
    p.DateTime = time.Now().UTC().String()
    if err != nil {
        return err
    }

    return nil
}

Codeium: Refactor | Explain | Docstring | ×
low complexity (13%)
func (p *PriceMonitor) Save() error {  new *
    jsonFile, erro := os.OpenFile( name: "prices.jsonl",
        os.O_APPEND|os.O_CREATE|os.O_WRONLY,  perm: 0644)

    if erro != nil {
        fmt.Println( a...: "error al abrir el json")
        return erro
    }
    defer jsonFile.Close()

    jsonData, err := json.Marshal(p)
    if err != nil {
        return err
    }

    jsonFile.Write(jsonData)
    jsonFile.WriteString( s: "\n")

    fmt.Println( a...: "Se guardo el precio")

    return nil
}

Codeium: Refactor | Explain | Docstring | ×
low complexity (46%)
func main() {  new *
    client := &http.Client{}
    cronJob := cron.New()

    _, err := cronJob.AddFunc( spec: "@every 2s", func() {
        println( args...: "Se ejecuta cada 2 segundos")
        priceMonitor := PriceMonitor{
            Id:         "MLM2590077782",
            httpClient: client,
        }

        err := priceMonitor.Fetch()
        if err != nil {
            return
        }

        err = priceMonitor.Save()
        if err != nil {
            return
        }

        println( args...: "Precio actual: ", priceMonitor.Price)

        err = priceMonitor.Fetch()
        if err != nil {
            fmt.Println( a...: "error al obtener el precio del item")
        }
    })
    if err != nil {
        return
    }
    cronJob.Start()

    select {}
}
```