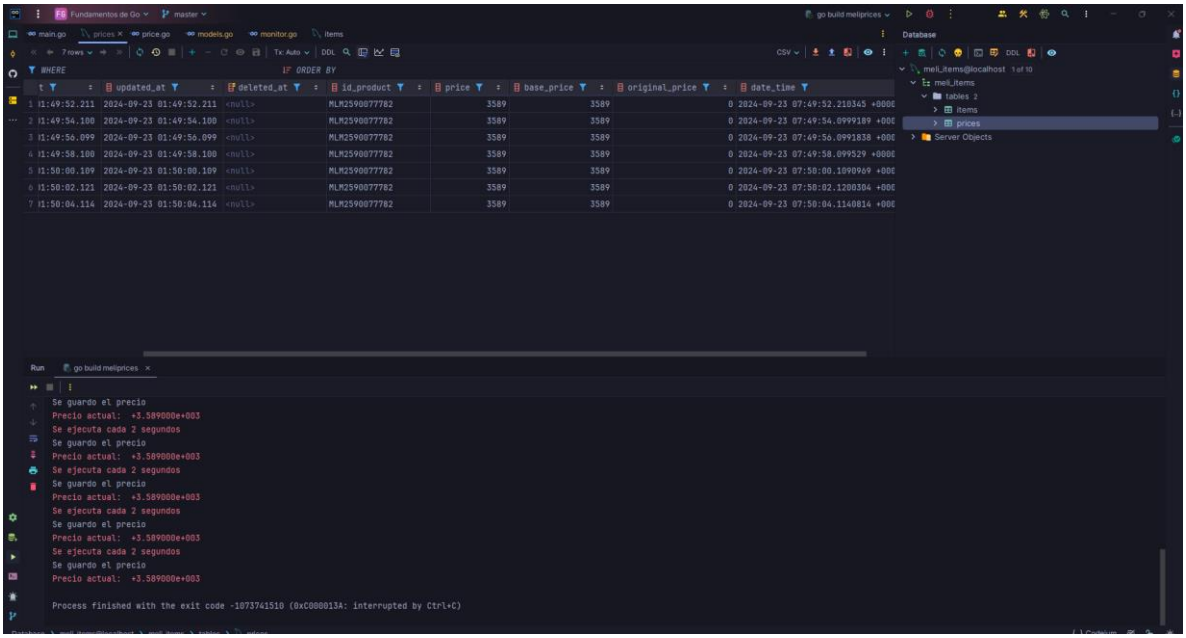
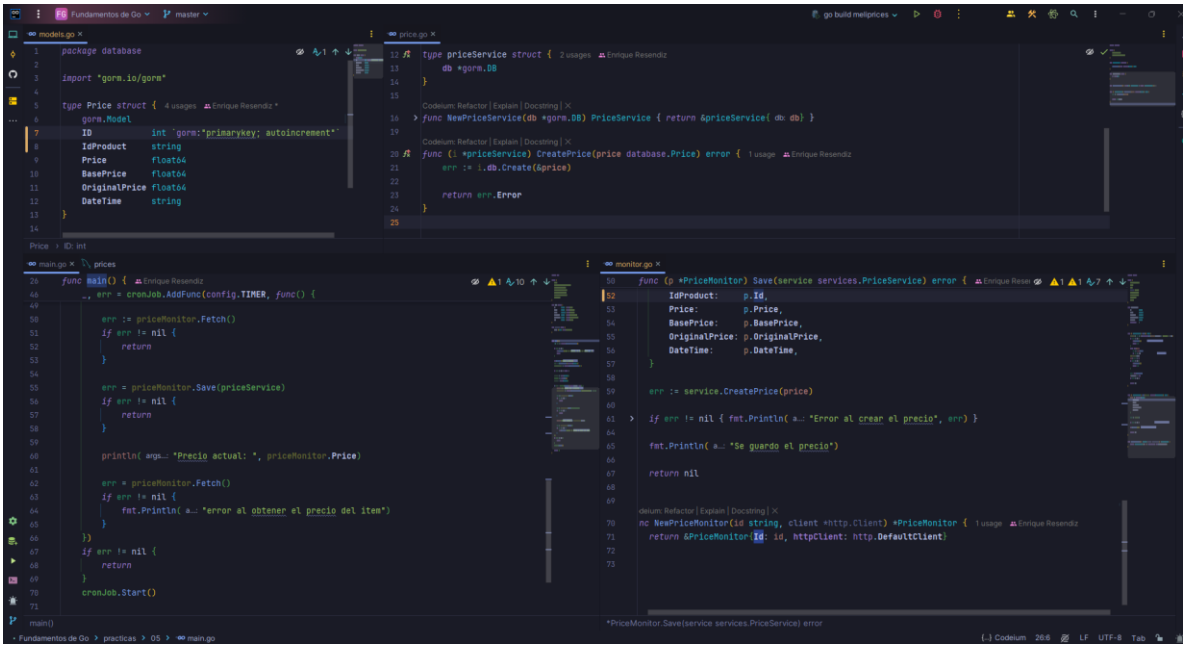

Fundamentos de Go
master
go build meliprices
main.go
prices
price.go
models.go
monitor.go
items
Database
WHERE
ORDER BY
updated_at
deleted_at
id_product
price
base_price
original_price
date_time
MLM2590077782
meli_items@localhost
meli_items
tables 2
items
prices
Server Objects
Run
go build meliprices
Se guardo el precio
Precio actual: +3.589000e+003
Se ejecuta cada 2 segundos
Process finished with the exit code -1073741510 (0xC000013A: interrupted by Ctrl+C)


Fundamentos de Go
master
go build meliprices
models.go
price.go
package database
import "gorm.io/gorm"
type Price struct
gorm.Model
ID int `gorm:"primarykey; autoincrement"`
IdProduct string
Price float64
BasePrice float64
OriginalPrice float64
DateTime string
type priceService struct
db *gorm.DB
func NewPriceService(db *gorm.DB) PriceService { return &priceService{ db: db} }
func (i *priceService) CreatePrice(price database.Price) error
err := i.db.Create(&price)
return err.Error
main.go
prices
monitor.go
func main() {
err := cronJob.AddFunc(config.TIMER, func() {
err := priceMonitor.Fetch()
if err != nil {
return
err = priceMonitor.Save(priceService)
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")
cronJob.Start()
func (p *PriceMonitor) Save(service services.PriceService) error
IdProduct: p.Id,
Price: p.Price,
BasePrice: p.BasePrice,
OriginalPrice: p.OriginalPrice,
DateTime: p.DateTime,
err := service.CreatePrice(price)
if err != nil { fmt.Println(a... "Error al crear el precio", err) }
fmt.Println(a... "Se guardo el precio")
return nil
func NewPriceMonitor(id string, client *http.Client) *PriceMonitor
return &PriceMonitor{Id: id, httpClient: http.DefaultClient}