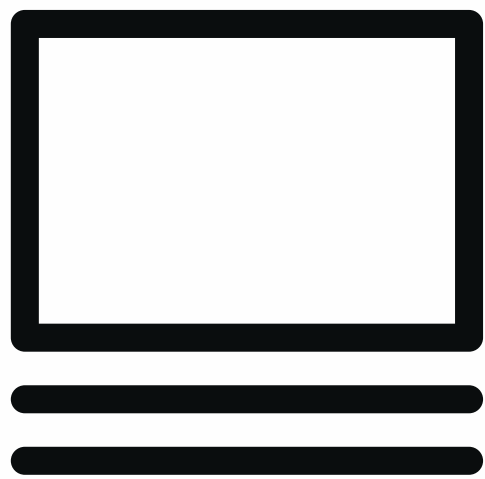
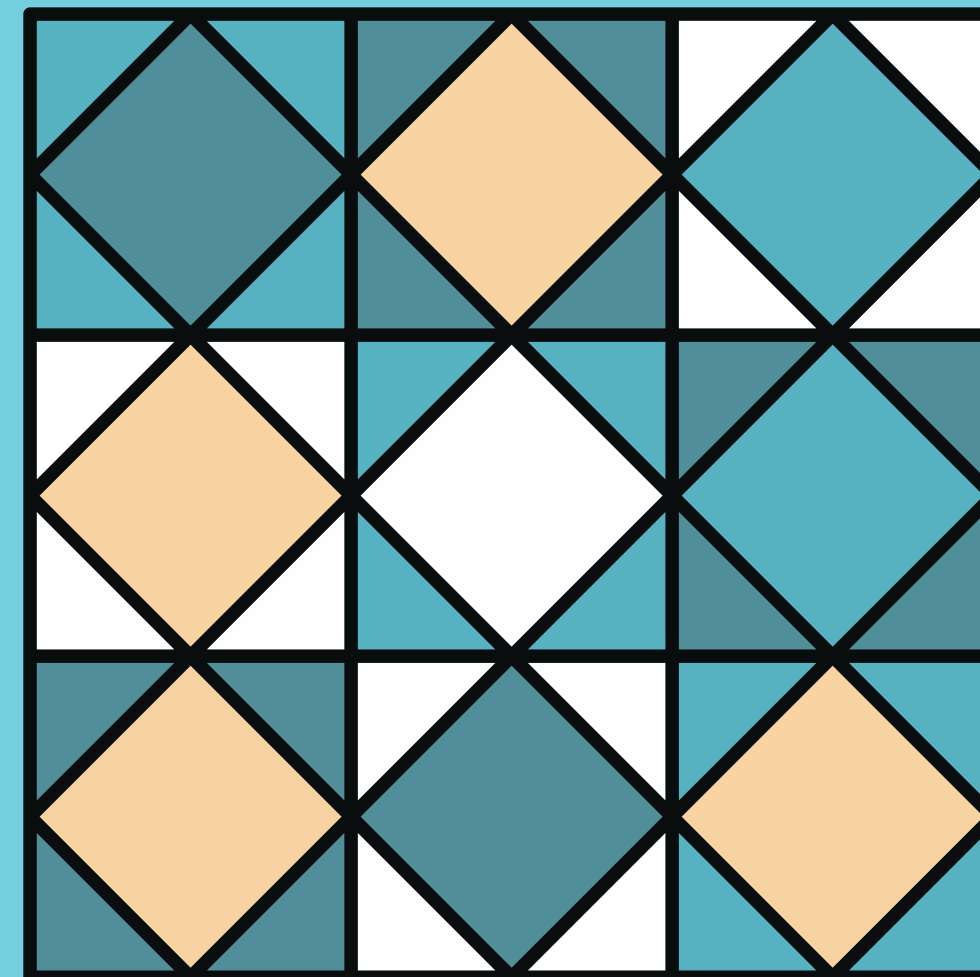


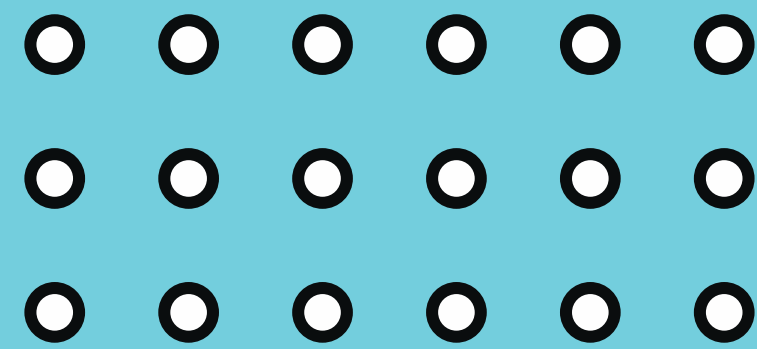


Sistemas Distribuidos 2024-1



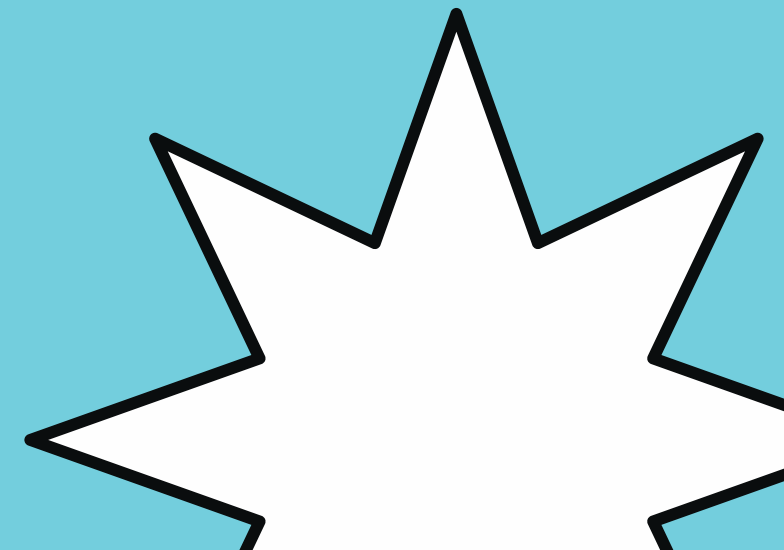
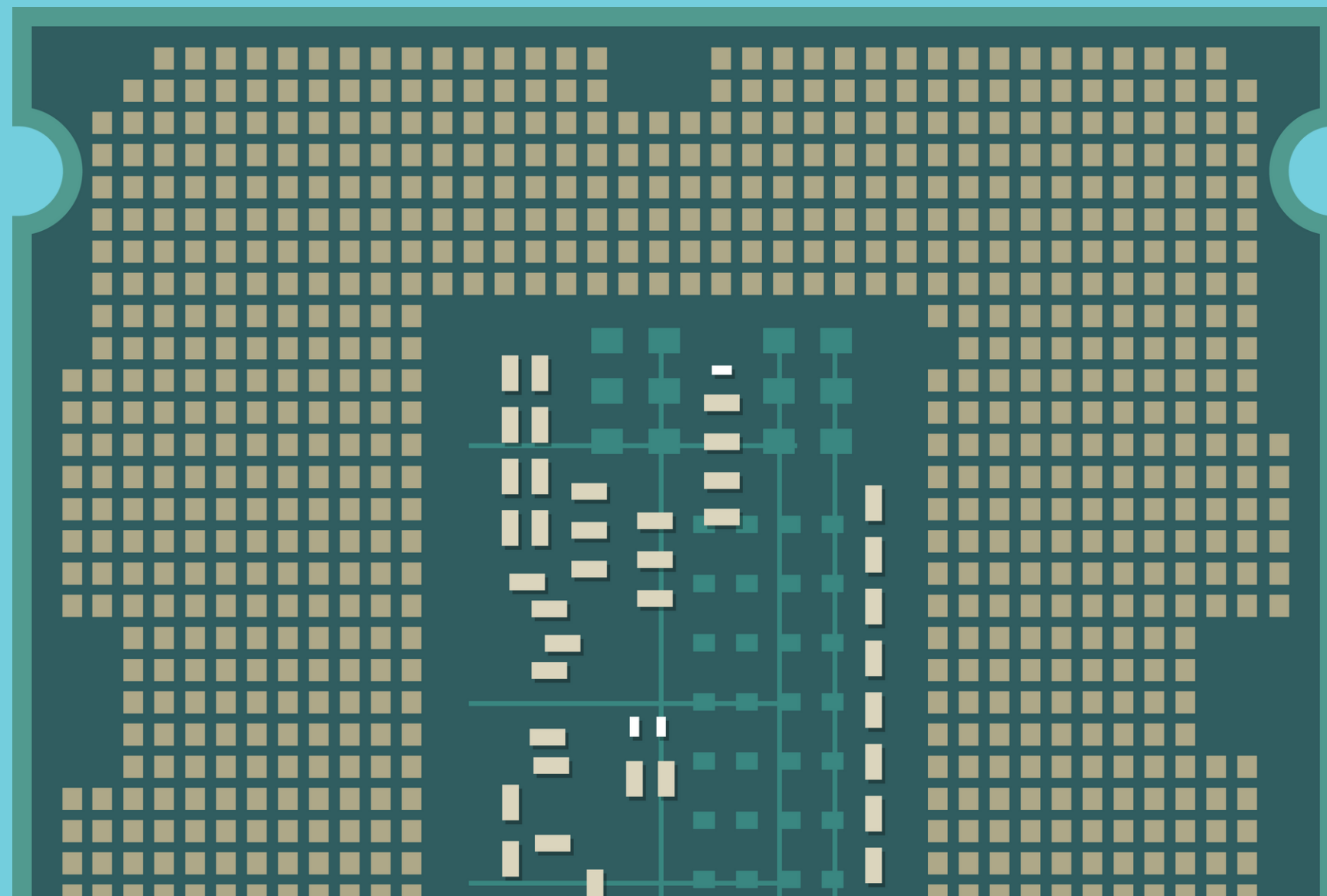
# INTRODUCCIÓN A GOLANG





## HISTORIA

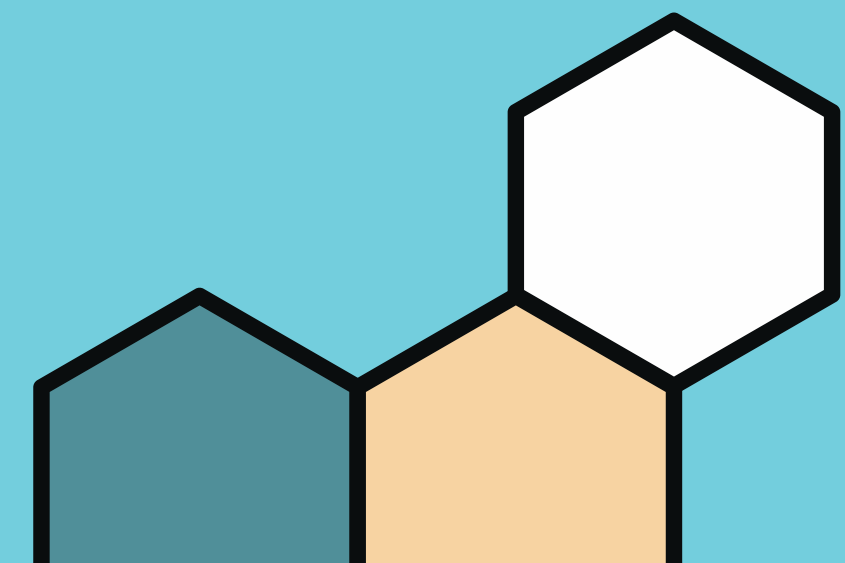
- Creado por Robert Griesemer, Rob Pike y Ken Thompson en Google en 2007
- Lanzamiento oficial en noviembre de 2009
- Diseñado para mejorar la productividad en la era de la programación multicore, redes y grandes bases de datos.
- Pensado en la escalabilidad

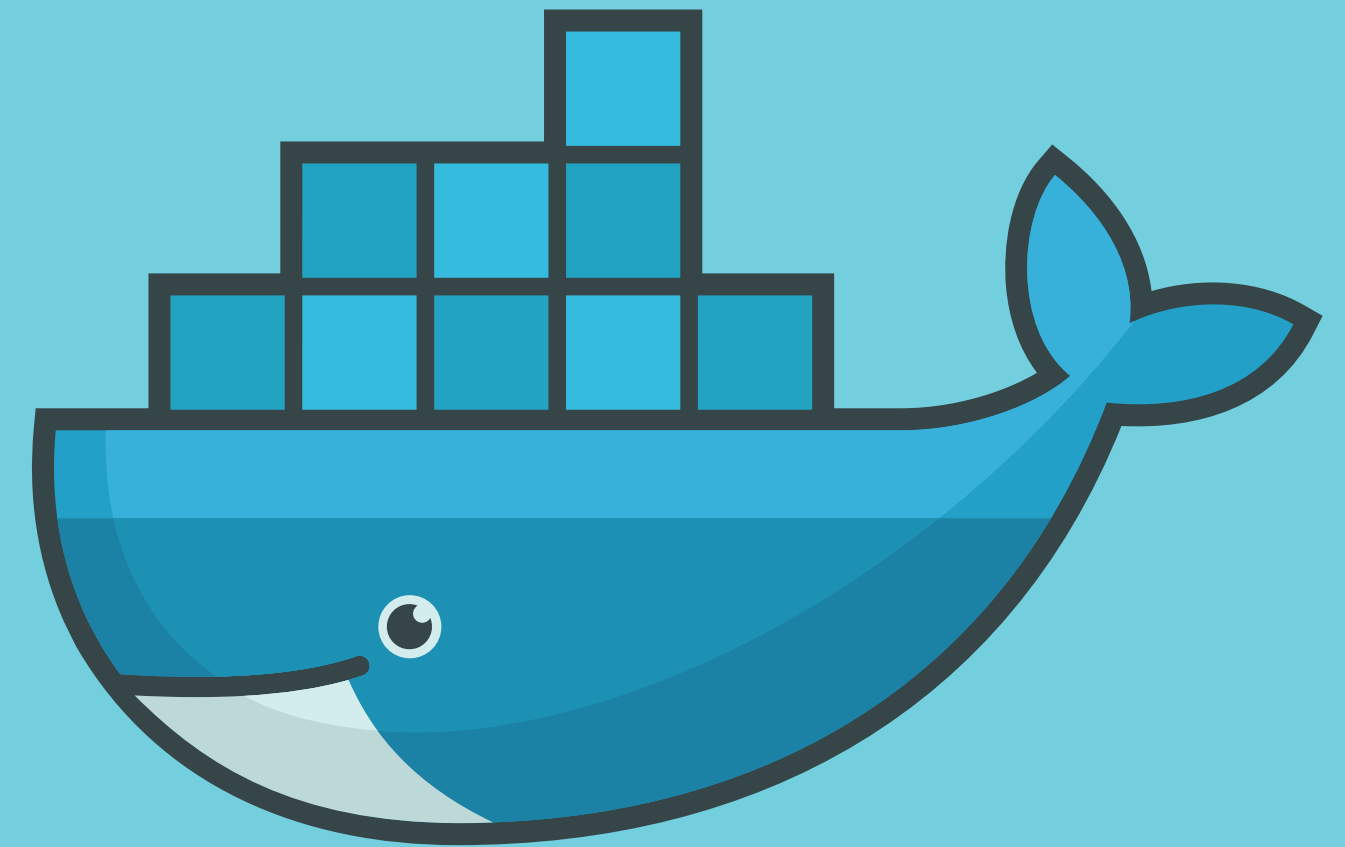






# USOS DE GO

- Desarrollo de servidores web y aplicaciones en red
  - desarrollo de software de sistemas y herramientas de comando
  - contenedores (Docker esta hecho en go) y orquestación (ej. Kubernetes)
  - Programación concurrente y en paralelo.
- 



# APLICACIONES MODERNAS DE GO



## SERVICIOS EN LA NUBE


Desarrollo de servicios que se basan en el uso de internet para la transferencia de datos como archivos

## MICROSERVICIOS

Su escalabilidad facilita el trabajo de mejora continua

## DESARROLLO DE APIS

Su funcionamiento sencillo y rápido lo hacen perfecto para generar apis



# FUNCIONAMIENTO DE GO

## 01

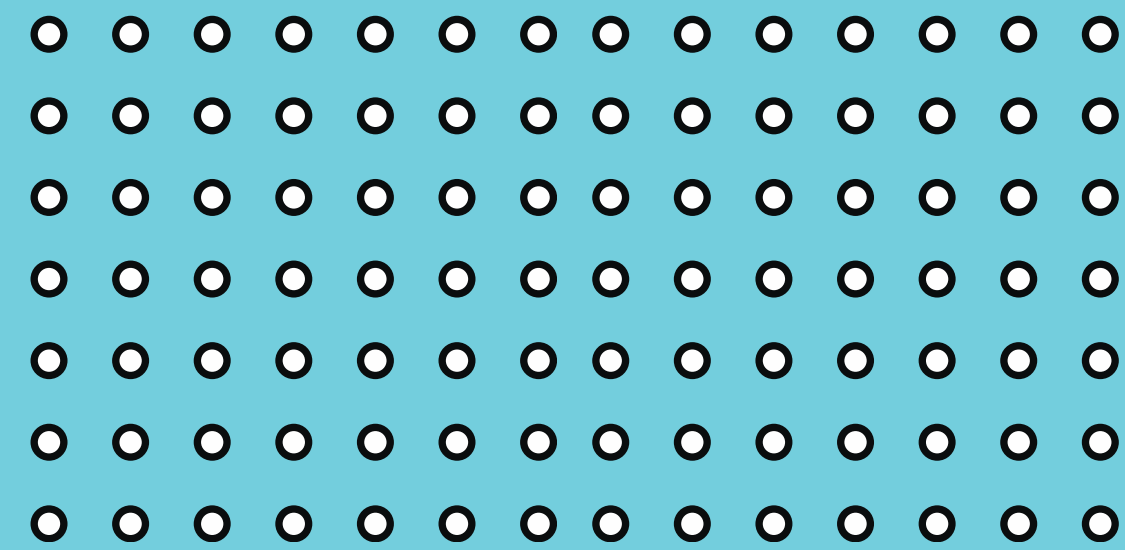
Similar a C pero con mejoras en el manejo de la memoria y la gestión de la concurrencia (con soporte nativo)

## 02

Recolector de basura eficiente y sintaxis limpia

# EJEMPLO SENCILLO

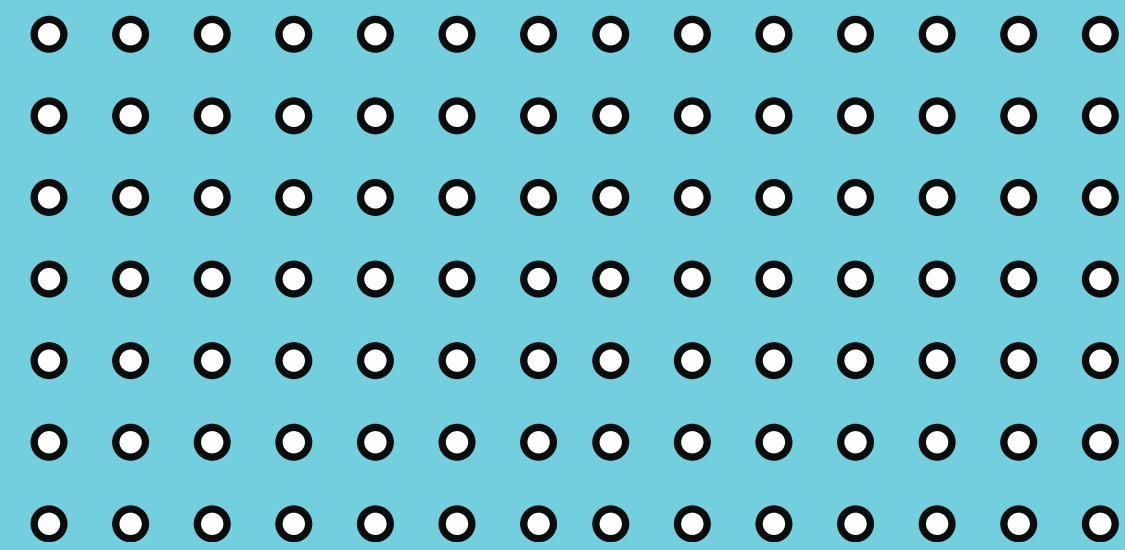
Lo básico, un hola mundo



```
1  package main
2  import "fmt"
3
4  func main() {
5      fmt.Println("Hola, mundo!")
6  }
```

# EJEMPLO 2

La concurrencia



```
1  package main
2  √ import (
3      "fmt"
4      "sync"
5  )
6
7  √ func say(text string, wg *sync.WaitGroup) {
8      defer wg.Done()
9      fmt.Println(text)
10 }
11
12 √ func main() {
13     var wg sync.WaitGroup
14     wg.Add(1)
15     go say("Hola", &wg)
16     wg.Wait()
17 }
```





# RECURSOS ADICIONALES

DOCUMENTACIÓN

TUTORIALES

REPOS



¿CONSULTAS?



## ACTIVIDAD: TAREAS EN PARALELO

Desarrolle un programa en Go que simule un procesador de tareas que ejecuta varias tareas en paralelo. Cada tarea será representada por una función que simula tiempo de ejecución mediante un sleep. El programa debe poder ejecutar un número determinado de tareas en paralelo, controlado por un parámetro, y finalizar cuando todas las tareas hayan completado su ejecución.



## DESAFÍO

Modifique el programa para que el número de tareas en paralelo sea dinámico, basado en un argumento de línea de comando. Además, gestiona los errores que puedan surgir si el número de tareas en paralelo es mayor que el número total de tareas.

