Go Programming Language

Mercedes Wyss (fb)
@itrjwyss (twitter, github)

¿Qué es Go?

Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

¿Qué es Go?

- Más conocido como Golang

- Compilado
- Concurrente
- Open-source
- Tipado estático
- garbage collected

Historia

Devs +502

Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Robert Griesemer, Rob Pike, Ken Thompson

- 2007 Robert Griesemer, Rob Pike y Ken Thompson, ingenieros en Google empezaron el desarrollo de este nuevo lenguaje de programación, basandose en C, C++ y Python.
- 2009 es lanzado durante el Google I/O
- 2013, marzo 28 que es considerado estable.

¿Quiénes lo están usando?

Dropbox

mercado
Libre.com®

monzo

Devs +502

Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

¿Para qué lo están usando?

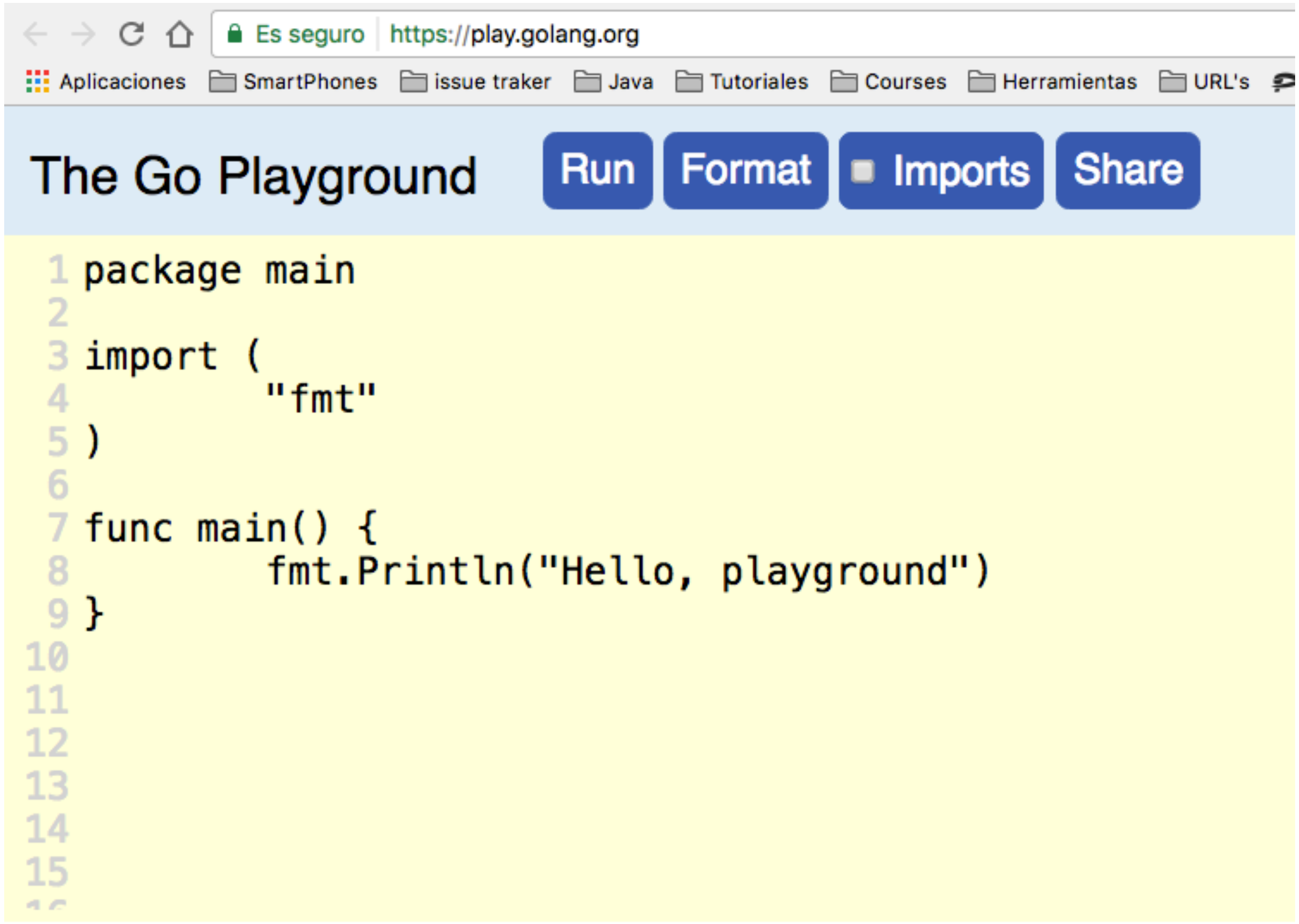
1. Data Science.
2. Backend.

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Algunas de sus Características

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

play.golang.org



```go
package main

import (
	"fmt"
)

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

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Valores Cero

```go
package main

import "fmt"

type Sampler interface {
    Sample()
}

func init() {
    fmt.Printf("I go first!\n")
}

func main() {

    var i int        //Nemeric
    var b bool       //Boolean
    var s string     //String
    var f Sampler    //Interface

    fmt.Printf("Num[%v], Bool[%v], Str[%v], Interface[%v]", i, b, s, f)
}
```

```
I go first!
Num[0], Bool[false], Str[], Interface[<nil>]
```

Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Funciones, Multiple Retorno

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Funciones, Multiple Retorno

```go
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Funciones, Multiple Retorno

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Procedimiento

Funciones, Multiple Retorno

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Funciones, Multiple Retorno

Parámetros

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Funciones, Multiple Retorno

Valores de retorno

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Funciones, Multiple Retorno

```go
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

Funciones, Multiple Retorno

```
package main

import "fmt"
import "errors"

func foo (s string) (string, error) {
  if s == "" {
    msg := fmt.Sprintf("Invalid input[%s]\n", s)
    return "", errors.New(msg)
  }

  return fmt.Sprintf("Hello %s\n", s), nil
}

func dothislater() {
  fmt.Printf("Ok, we're done.\n")
}

func main() {

  defer dothislater() //use of defer

  s, err := foo("Go") //multi-return example
  if err != nil {
    fmt.Printf("%s", err)
  } else {
    fmt.Printf("%s", s)
  }
}
```

```
Hello Go
Ok, we're done.
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Defer

```go
package main

import "fmt"

func main() {
    fmt.Println("counting")

    for i := 0; i < 10; i++ {
        defer fmt.Println(i)
    }

    fmt.Println("done")
}
```

Defer

```
counting
done
9
8
7
6
5
4
3
2
1
0
```

```go
package main

import "fmt"

func main() {
    fmt.Println("counting")

    for i := 0; i < 10; i++ {
        defer fmt.Println(i)
    }

    fmt.Println("done")
}
```

Defer

```
package main

import "fmt"

func test(){
    fmt.Println("probando")
    defer fmt.Println("x")
    defer fmt.Println("Y")
    fmt.Println("Fin")
}

func main() {
    test()

    fmt.Println("counting")

    for i := 0; i < 10; i++ {
        defer fmt.Println(i)
    }

    fmt.Println("done")
}
```

Defer

```
probando
Fin
Y
x
counting
done
9
8
7
6
5
4
3
2
1
0
```

```go
package main

import "fmt"

func test(){
    fmt.Println("probando")
    defer fmt.Println("x")
    defer fmt.Println("Y")
    fmt.Println("Fin")
}

func main() {
    test()

    fmt.Println("counting")

    for i := 0; i < 10; i++ {
        defer fmt.Println(i)
    }

    fmt.Println("done")
}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Structure, Method, Receiver

```
package main

import "fmt"

type Employee struct {
    Name    string
    Age     int
}

//Using 'pointer' Receiver
func (u *Employee) Rename(newName string) {
    u.Name = newName
}

func main() {
    e := Employee{"John Doe", 31}
    fmt.Printf("Employee %+v\n", e)
    e.Rename("Jane Doe")
    fmt.Printf("Employee %v\n", e)
}
```

Structure, Method, Receiver

```
package main

import "fmt"

type Employee struct {
    Name    string
    Age     int
}

//Using 'pointer' Receiver
func (u *Employee) Rename(newName string) {
    u.Name = newName
}

func main() {
    e := Employee{"John Doe", 31}
    fmt.Printf("Employee %+v\n", e)
    e.Rename("Jane Doe")
    fmt.Printf("Employee %v\n", e)
}
```

Structure, Method, Receiver

```
package main

import "fmt"

type Employee struct {
    Name    string
    Age     int
}

//Using 'pointer' Receiver
func (u *Employee) Rename(newName string) {
    u.Name = newName
}

func main() {
    e := Employee{"John Doe", 31}
    fmt.Printf("Employee %+v\n", e)
    e.Rename("Jane Doe")
    fmt.Printf("Employee %v\n", e)
}
```

Structure, Method, Receiver

```go
package main

import "fmt"

type Employee struct {
    Name    string
    Age     int
}

//Using 'pointer' Receiver
func (u *Employee) Rename(newName string) {
    u.Name = newName
}

func main() {
    e := Employee{"John Doe", 31}
    fmt.Printf("Employee %+v\n", e)
    e.Rename("Jane Doe")
    fmt.Printf("Employee %v\n", e)
}
```

Structure, Method, Receiver

```go
package main

import "fmt"

type Employee struct {
    Name    string
    Age     int
}

//Using 'pointer' Receiver
func (u *Employee) Rename(newName string) {
    u.Name = newName
}

func main() {
    e := Employee{"John Doe", 31}
    fmt.Printf("Employee %+v\n", e)
    e.Rename("Jane Doe")
    fmt.Printf("Employee %v\n", e)
}
```

```
Employee {Name:John Doe Age:31}
Employee {Jane Doe 31}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Interface

```go
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

Interface

```go
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Interface

```
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

Interface

```go
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

Devs +502

Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Interface

```go
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

Interface

```
package main

import "fmt"
import "math/rand"

type MyInt int

type Sampler interface {
    Sample() int
}

func (i MyInt) Sampler() int {
    return rand.Int()
}

func main() {
    var i MyInt
    fmt.Printf("MyInt: %d\n", i)
    fmt.Printf("Sampler: %d\n", i.Sampler())
}
```

MyInt: 0
Sampler: 134020434

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Gorutinas

```go
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {

    f("direct")

    go f("goroutine")

    go func(msg string) {
        fmt.Println(msg)
    }("going")

    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Gorutinas

```go
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {

    f("direct")

    go f("goroutine")

    go func(msg string) {
        fmt.Println(msg)
    }("going")

    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}
```

Gorutinas

```
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {

    f("direct")

    go f("goroutine")

    go func(msg string) {
        fmt.Println(msg)
    }("going")

    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}
```

```
direct : 0
direct : 1
direct : 2
goroutine : 0
going
goroutine : 1
goroutine : 2
<enter>
done
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

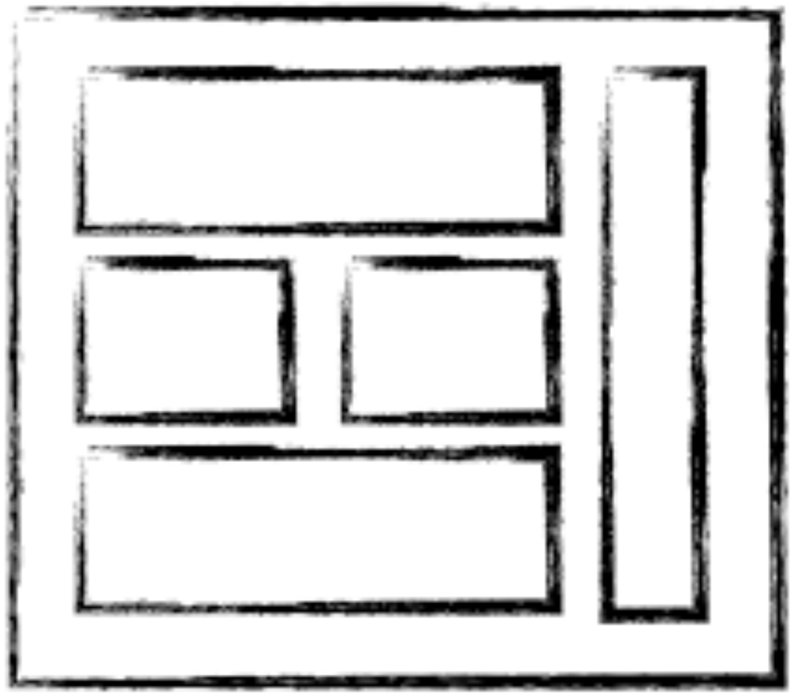
Server Side

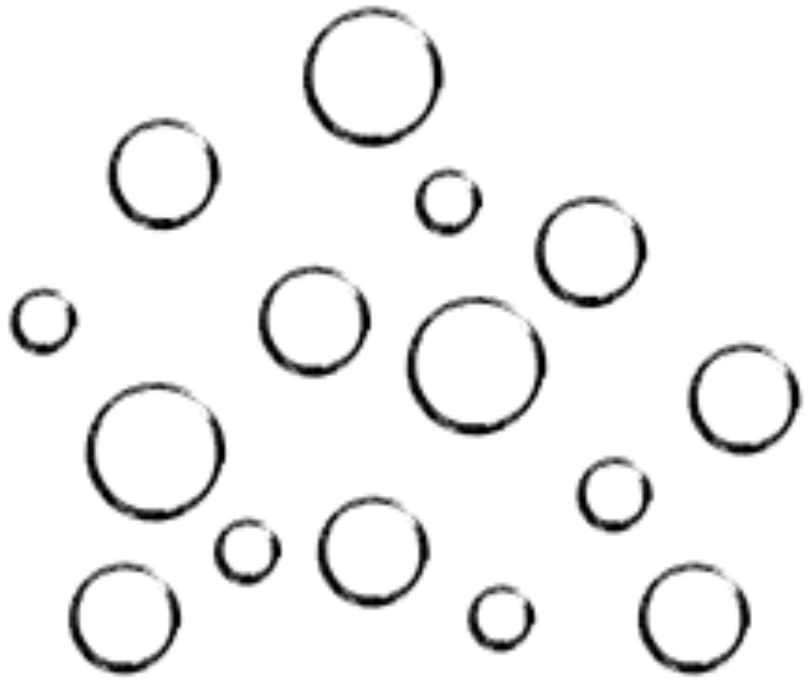
```go
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hi there, I love %s!", r.URL.Path[1:])
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

```
go run <path>/server.go
```

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

Server Side


MONOLITHIC/LAYERED


MICRO SERVICES

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala