Generating RSA PEM Key Pair using Go

A Step-by-Step Guide to Generate and Test RSA Key Pair in Go

Ravi Tiwari · Follow
Published in System Weakness
4 min read · Apr 3

Listen Share

RSA is a widely used public-key cryptography algorithm that is used for secure communication over the internet. In this article, we will learn how to generate an RSA PEM key pair from pure Go code and how to test it.

To generate an RSA PEM key pair, we need to follow these steps:

1. Generate a new RSA private key.
2. Encode the private key to the PEM format.
3. Extract the public key from the private key.
4. Encode the public key to the PEM format.

Let's dive into the code to generate an RSA PEM key pair from pure Go:

```
package main

import (
    "crypto/rand"
    "crypto/rsa"
    "crypto/x509"
    "encoding/pem"
    "fmt"
    "os"
```

```
)

func main() {
    // Generate a new RSA private key with 2048 bits
    privateKey, err := rsa.GenerateKey(rand.Reader, 2048)
    if err != nil {
        fmt.Println("Error generating RSA private key:", err)
        os.Exit(1)
    }

    // Encode the private key to the PEM format
    privateKeyPEM := &pem.Block{
        Type:  "RSA PRIVATE KEY",
        Bytes: x509.MarshalPKCS1PrivateKey(privateKey),
    }
    privateKeyFile, err := os.Create("private_key.pem")
    if err != nil {
        fmt.Println("Error creating private key file:", err)
        os.Exit(1)
    }
    pem.Encode(privateKeyFile, privateKeyPEM)
    privateKeyFile.Close()

    // Extract the public key from the private key
    publicKey := &privateKey.PublicKey

    // Encode the public key to the PEM format
    publicKeyPEM := &pem.Block{
        Type:  "RSA PUBLIC KEY",
        Bytes: x509.MarshalPKCS1PublicKey(publicKey),
    }
    publicKeyFile, err := os.Create("public_key.pem")
    if err != nil {
        fmt.Println("Error creating public key file:", err)
        os.Exit(1)
    }
    pem.Encode(publicKeyFile, publicKeyPEM)
    publicKeyFile.Close()

    fmt.Println("RSA key pair generated successfully!")
}
```

In the above code, we first generate a new RSA private key with 2048 bits using the `rsa.GenerateKey` function. We then encode the private key to the PEM format using the `pem.Encode` function and write it to a file named `private_key.pem`.

Next, we extract the public key from the private key and encode it to the PEM format using the `pem.Encode` function. We write the public key to a file named `public_key.pem`.

Finally, we print a success message to the console.

To run this program, save the code to a file named `generate_rsa_key_pair.go`, and then run the following command in your terminal:

```
go run generate_rsa_key_pair.go
```

This will generate a private key file named `private_key.pem` and a public key file named `public_key.pem` in your current directory.

To test the generated RSA key pair, we can use the following steps:

1. Load the private key from the `private_key.pem` file.
2. Sign a message using the private key.
3. Load the public key from the `public_key.pem` file.
4. Verify the signature using the public key.

Here is the code to test the generated RSA key pair:

```
package main

import (
    "crypto"
    "crypto/rand"
    "crypto/rsa"
    "crypto/sha256"
    "crypto/x509"
    "encoding/pem"
    "fmt"
    "io/ioutil"
    "os"
)
```

```
func main() {
    // Load the private key from the file
    privateKeyFile, err := ioutil.ReadFile("private_key.pem")
    if err != nil {
        fmt.Println("Error loading private key file:", err)
        os.Exit(1)
    }
    privateKeyBlock, _ := pem.Decode(privateKeyFile)
    privateKey, err := x509.ParsePKCS1PrivateKey(privateKeyBlock.Bytes)
    if err != nil {
        fmt.Println("Error parsing private key:", err)
        os.Exit(1)
    }

    // Sign a message using the private key
    message := []byte("Hello, world!")
    hash := sha256.Sum256(message)
    signature, err := rsa.SignPKCS1v15(rand.Reader, privateKey, crypto.SHA256, has
    if err != nil {
        fmt.Println("Error signing message:", err)
        os.Exit(1)
    }

    // Load the public key from the file
    publicKeyFile, err := ioutil.ReadFile("public_key.pem")
    if err != nil {
        fmt.Println("Error loading public key file:", err)
        os.Exit(1)
    }
    publicKeyBlock, _ := pem.Decode(publicKeyFile)
    publicKey, err := x509.ParsePKCS1PublicKey(publicKeyBlock.Bytes)
    if err != nil {
        fmt.Println("Error parsing public key:", err)
        os.Exit(1)
    }

    // Verify the signature using the public key
    hash = sha256.Sum256(message)
    err = rsa.VerifyPKCS1v15(publicKey, crypto.SHA256, hash[:], signature)
    if err != nil {
        fmt.Println("Error verifying signature:", err)
        os.Exit(1)
    }

    fmt.Println("RSA key pair tested successfully!")
}
```

In the above code, we first load the private key from the `private_key.pem` file and parse it using the `x509.ParsePKCS1PrivateKey` function. We then sign a message using the private key and the `rsa.SignPKCS1v15` function.

Next, we load the public key from the `public_key.pem` file and parse it using the `x509.ParsePKCS1PublicKey` function. We verify the signature using the public key and the `rsa.VerifyPKCS1v15` function.

Finally, we print a success message to the console.

To test this program, save the code to a file named `test_rsa_key_pair.go`, and then run the following command in your terminal:

```
go run test_rsa_key_pair.go
```

This will load the RSA key pair from the `private_key.pem` and `public_key.pem` files, sign

In conclusion, generating an RSA PEM key pair from pure Go code is a straightforward process that can be accomplished with just a few lines of code. With the generated key pair, you can use it to encrypt and decrypt data, sign and verify messages, and establish secure communication channels over the internet.

Code — https://github.com/rtiwariops/CodeHub/tree/main/rsa-pem-key-pair

Go Golang Keypair Rsa Pem

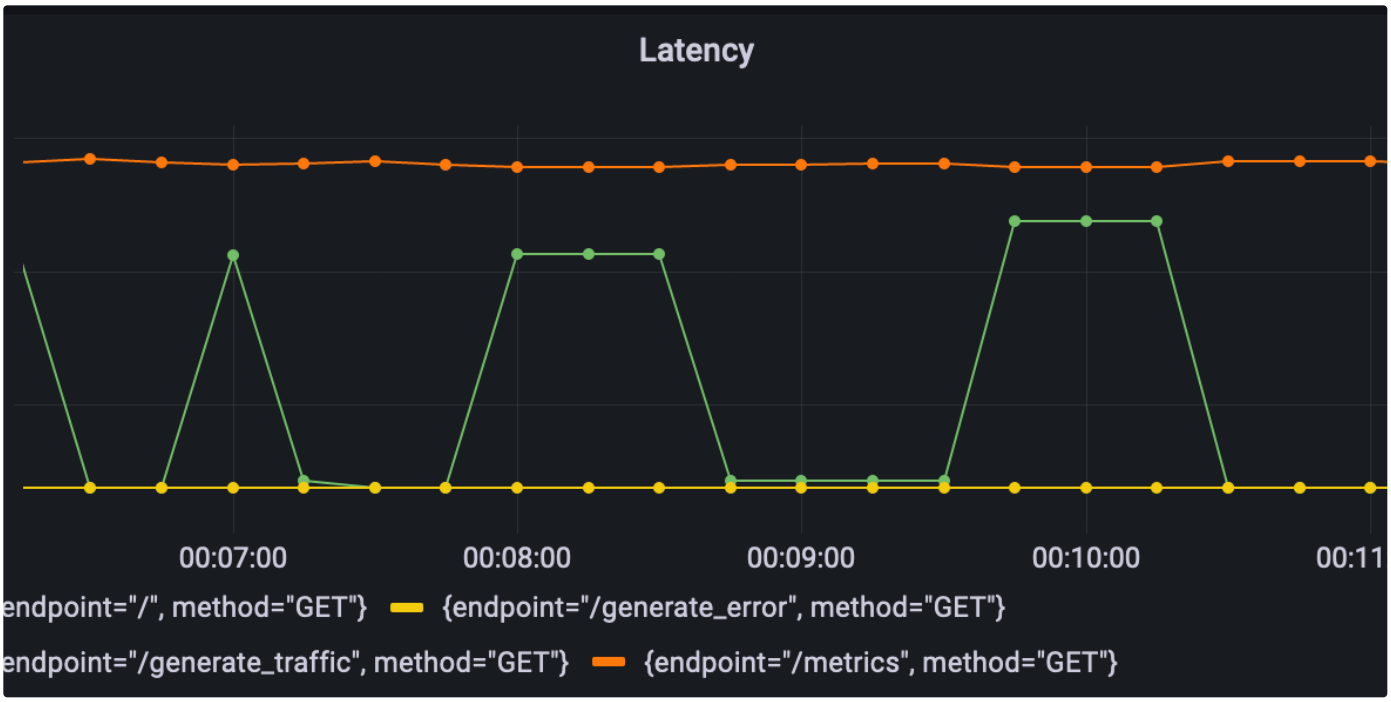
Follow

Written by Ravi Tiwari

47 Followers · Writer for System Weakness

Experienced hands-on CTO with 20+ years of cloud native microservices expertise, driving solutions for large-scale enterprises.

More from Ravi Tiwari and System Weakness



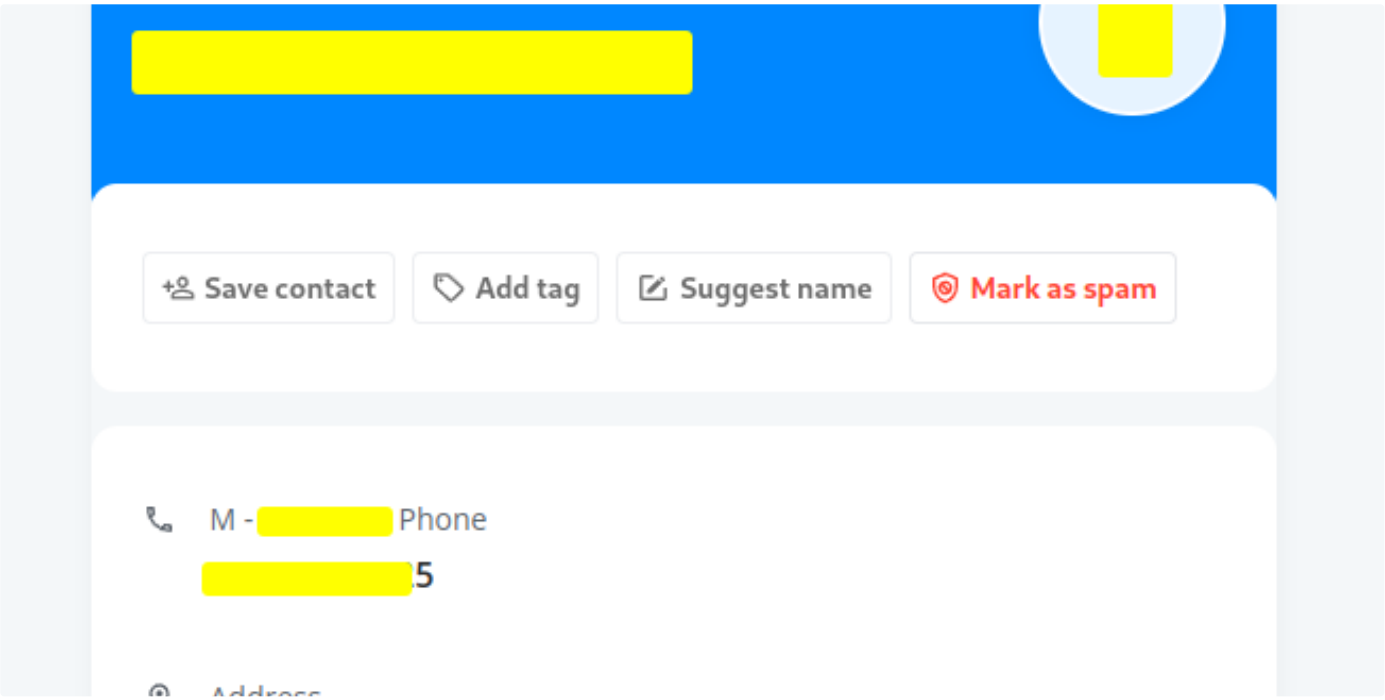
Ravi Tiwari

Four Golden Signals Of Monitoring: Site Reliability Engineering (SRE) Metrics

Golden Signal Monitoring using FastAPI on k8s

8 min read · Mar 2

96 1



Mr Jokar in System Weakness

Track Anyone with just a Phone Number | OSINT Investigation

You can be an OSINT Investigator, CTF Player or simply someone who is getting spam calls. Someone who is trying to verify the number you...

3 min read · May 14

285 2

Ravi Tiwari

Building Domain-Driven Microservices in the Pharmacy Industry: A Python Example

Designing and Implementing a Microservices Architecture with DDD Principles for Efficient Pharmacy Management

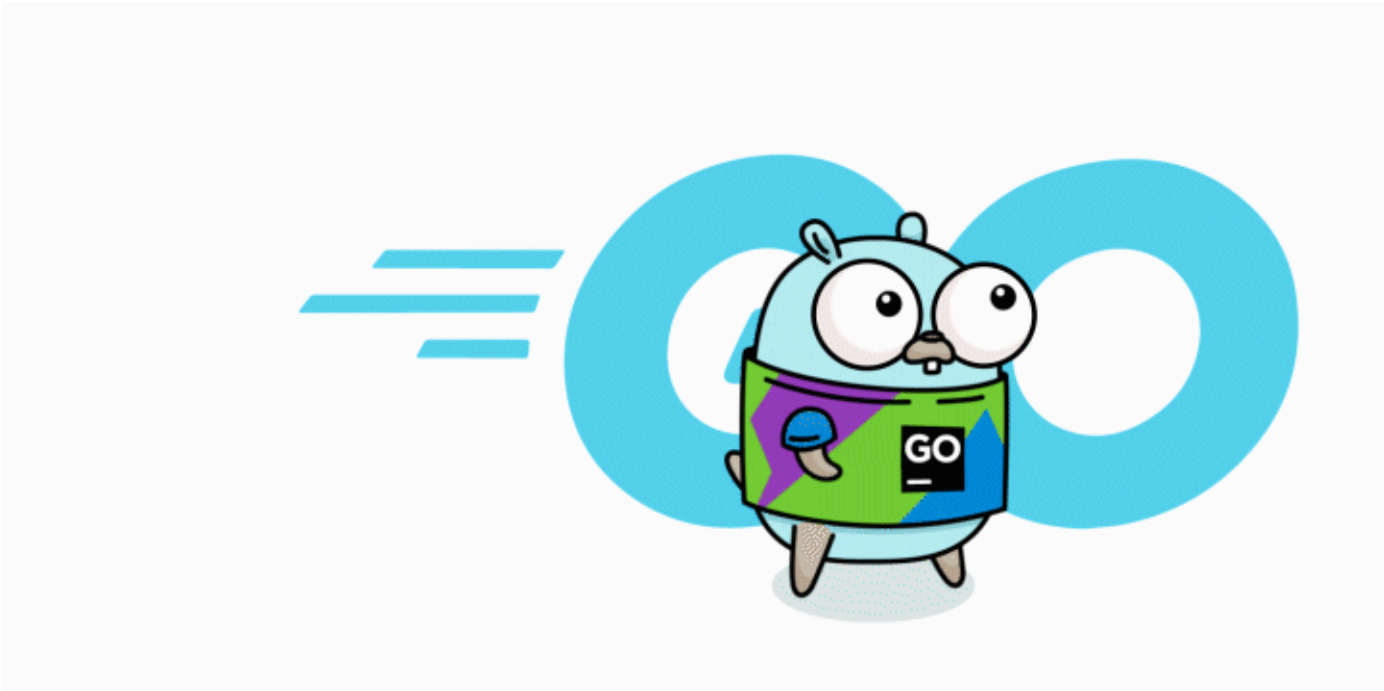
5 min read · Apr 17


1

See all from Ravi Tiwari

See all from System Weakness

Recommended from Medium



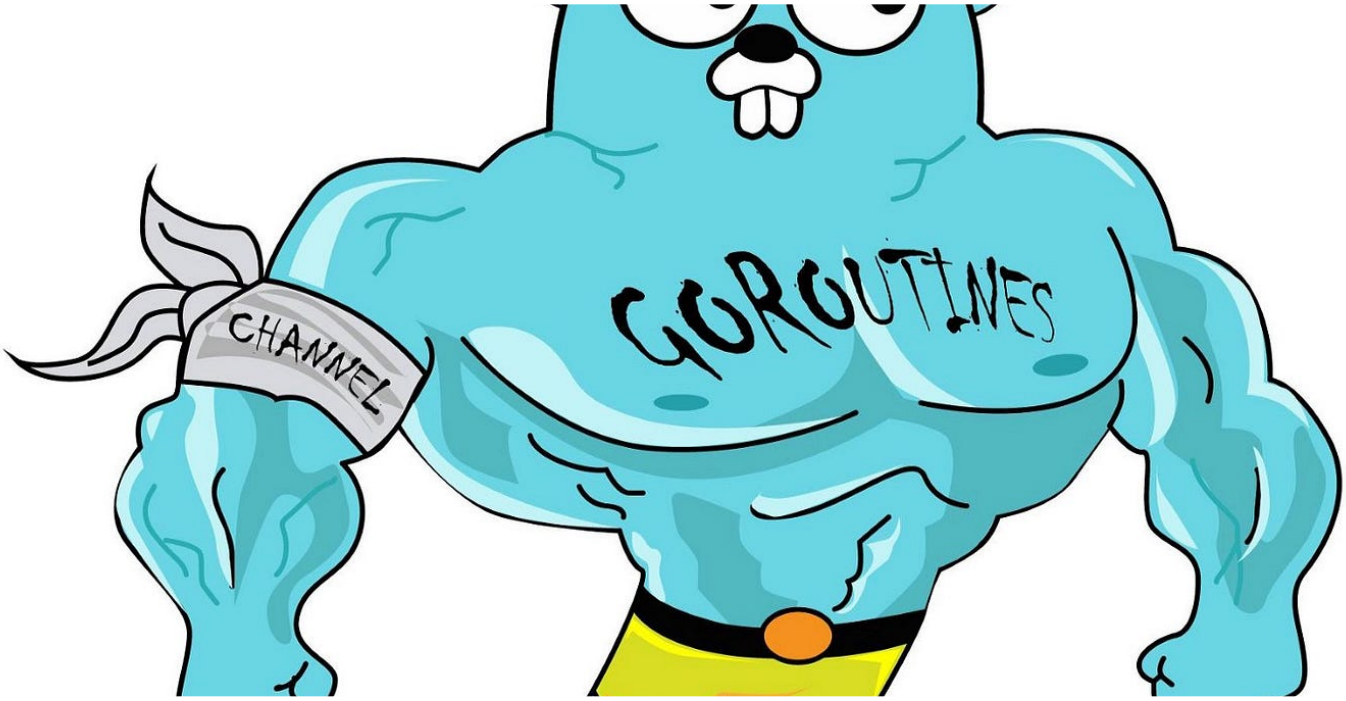
Abhinav

A Comprehensive Guide to Authentication and Authorization in Go with Code (Golang)

Authentication and authorization are fundamental aspects of web application security. They help ensure that only authorized users can...

· 6 min read · Aug 2

2



Ramseyjiang in Level Up Coding

Goroutines size and how many of them are in your laptop

In this article, I explain goroutines and how to calculate each goroutine size using codes.

· 4 min read · Feb 22

69

Lists

General Coding Knowledge

20 stories · 215 saves

Now in AI: Handpicked by Better Programming

262 stories · 87 saves

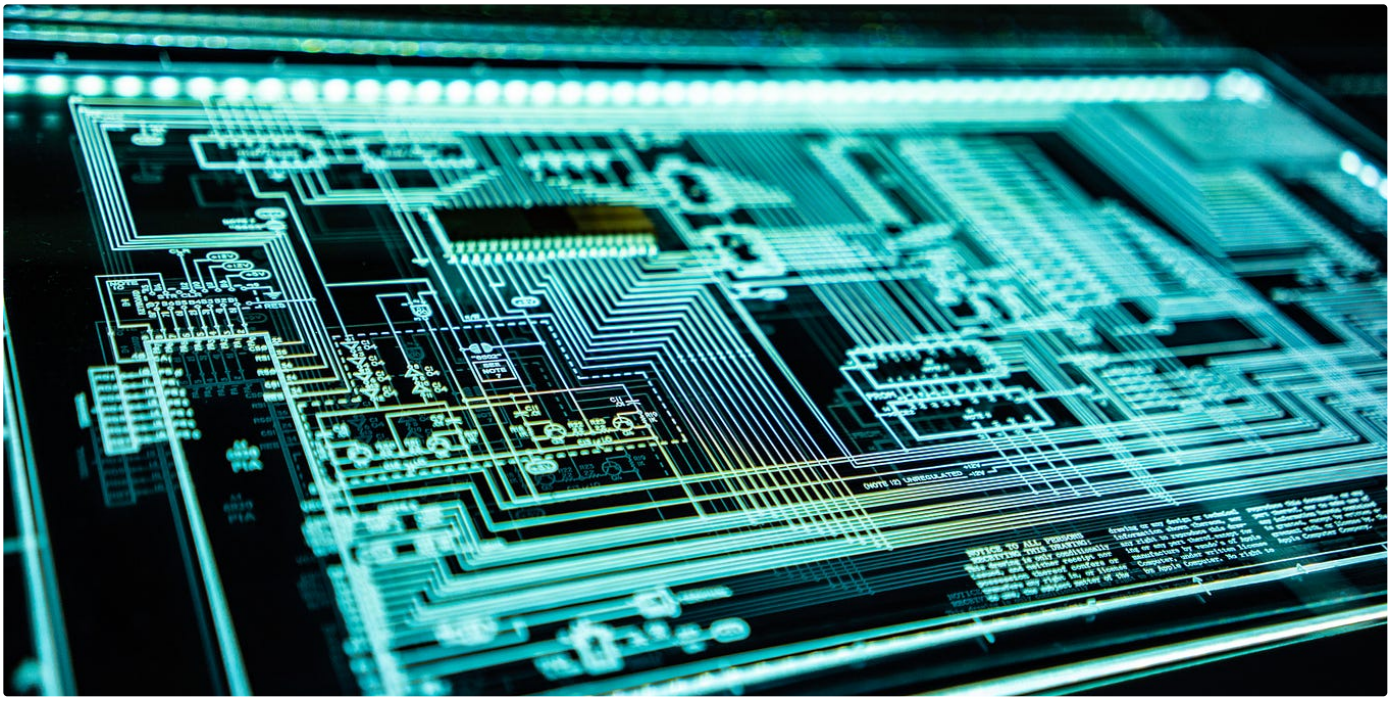
Ramseyjiang in Level Up Coding

What's the thread safety in Golang? Five thread-safe ways.

In this article, I will introduce the thread-safe in Golang and introduce several thread-safe ways in Golang.

· 4 min read · Mar 7

131



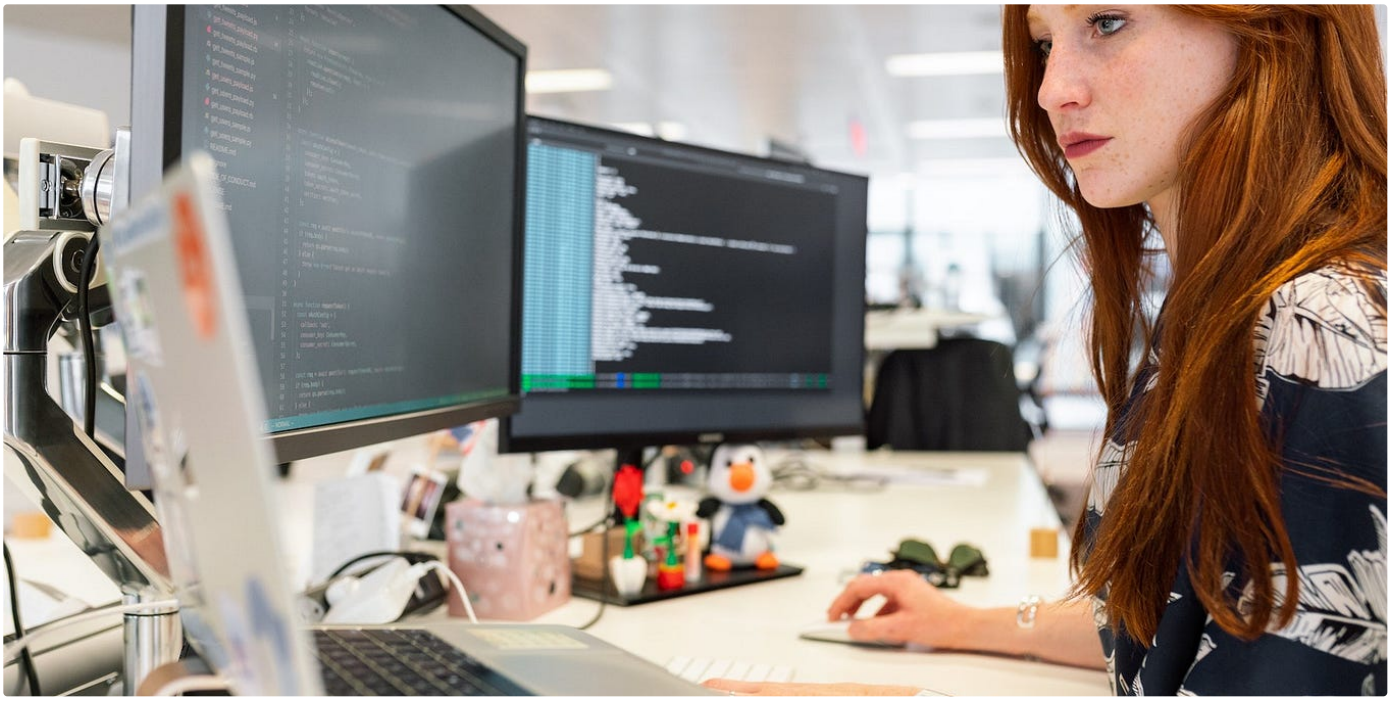
Midnight Firesale in Stackademic

Unveiling the Top 10 Gotchas in the Go (Golang) Programming Language

Novices and veterans alike will find they occassionaly make these common mistakes.

· 5 min read · 5 days ago

70 1



The Coding Diaries in The Coding Diaries

Why Experienced Programmers Fail Coding Interviews

A friend of mine recently joined a FAANG company as an engineering manager, and found themselves in the position of recruiting for...

· 5 min read · Nov 2, 2022

6.6K 133

See more recommendations