## Come creare un server SMTP di base con Go

Il protocollo Simple Mail Transfer Protocol (SMTP) è fondamentale per l'invio di email. Implementare un server SMTP personalizzato può essere utile per comprendere meglio come funziona il protocollo e per sviluppare applicazioni su misura. In questo articolo, esploreremo come creare un server SMTP di base utilizzando il linguaggio di programmazione Go.

Go fornisce una libreria standard ricca di funzionalità, ma per un server SMTP, useremo una libreria esterna chiamata go-smtp, che semplifica la gestione di un server SMTP.

```
go get github.com/emersion/go-smtp
```

Inizia importando i pacchetti standard di Go e la libreria go-smtp:

```
package main

import (
        "fmt"
        "log"
        "net"
        "github.com/emersion/go-smtp"
)
```

Il gestore (handler) è una struttura che implementa l'interfaccia `smtp.Backend`. Questo è il punto in cui si definisce come gestire i messaggi ricevuti.

```
type EmailHandler struct {}

func (e *EmailHandler) Login(state *smtp.ConnectionState, username, password string)
(smtp.Session, error) {
        return nil, smtp.ErrAuthUnsupported
}

func (e *EmailHandler) AnonymousLogin(state *smtp.ConnectionState) (smtp.Session, error) {
        return &Session{}, nil
}

// Implementazione della sessione SMTP
type Session struct {}

func (s *Session) Mail(from string, opts smtp.MailOptions) error {
        log.Printf("Email ricevuta da: %s", from)
        return nil
}

func (s *Session) Rcpt(to string) error {
        log.Printf("Destinatario: %s", to)
        return nil
}
```

```
func (s *Session) Data(r io.Reader) error {
        buf := new(bytes.Buffer)
        buf.ReadFrom(r)
        log.Printf("Contenuto email: %s", buf.String())
        return nil
}

func (s *Session) Reset() {}
func (s *Session) Logout() error { return nil }

func (e *EmailHandler) NewSession(state *smtp.ConnectionState) (smtp.Session, error) {
        return &Session{}, nil
}
```

Una volta definito il gestore, possiamo configurare e avviare il server SMTP.

```
func main() {
        handler := &EmailHandler{}

        // Configura il server
        server := smtp.NewServer(handler)
        server.Addr = ":2525"
        server.Domain = "localhost"
        server.AllowInsecureAuth = true

        // Avvia il server
        log.Printf("Avvio del server SMTP sulla porta %s", server.Addr)
        if err := server.ListenAndServe(); err != nil {
                log.Fatal(err)
        }
}
```

## Conclusione

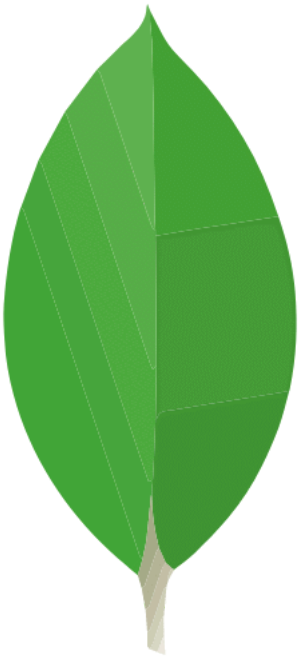
Abbiamo creato un server SMTP di base con Go, che può ricevere email e gestire mittenti e destinatari. Questo server è una base su cui è possibile costruire funzionalità più avanzate, come l’autenticazione, il supporto TLS o l’integrazione con un sistema di archiviazione. Approfondire il funzionamento del protocollo SMTP e la libreria go-smtp ti permetterà di realizzare soluzioni personalizzate per le tue esigenze.

## Applicazioni Correlate

- 

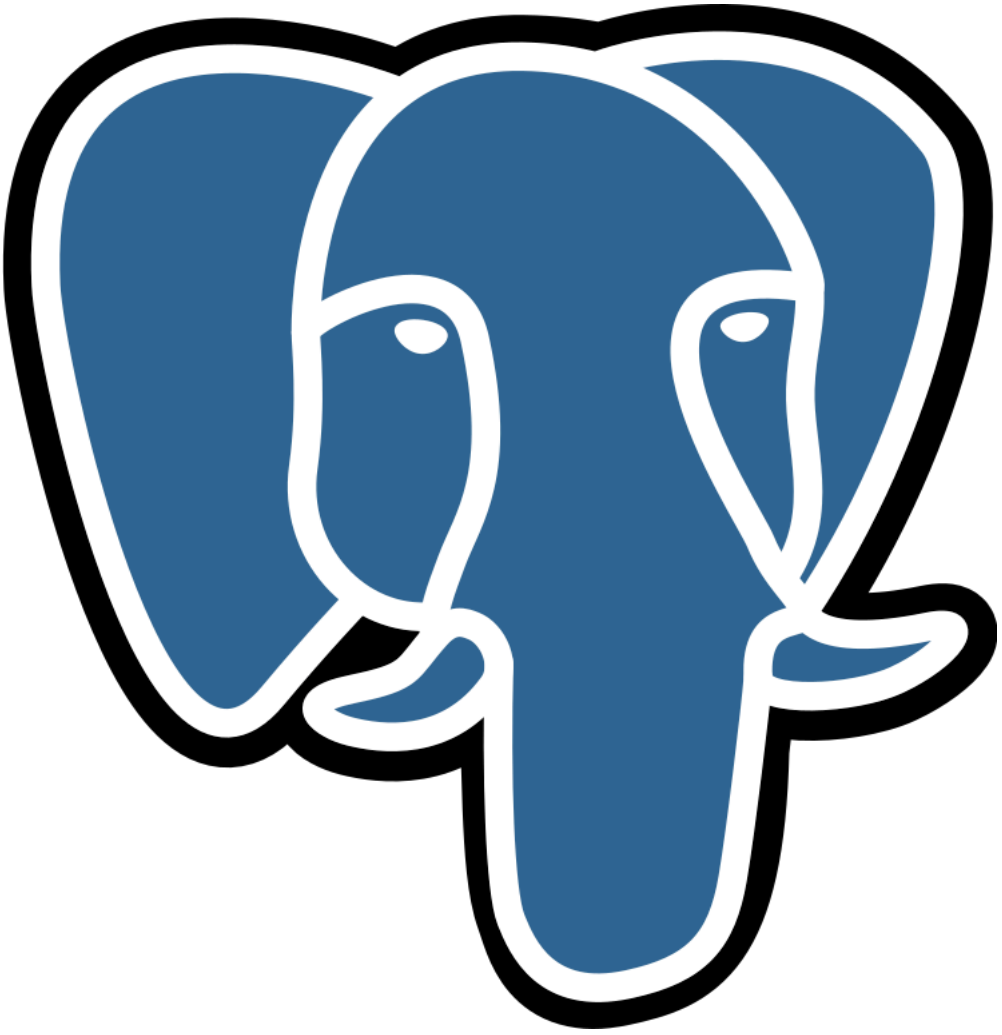
  **Go Placeholder Image**

  Applicazione in Go per la creazione di immagini segnaposto.
  DockerDocker ComposeGoJavaScript



### Go MongoDB App

Applicazione basata su MongoDB ed implementata in Go con il driver ufficiale.
DockerDocker ComposeGoMongoDB



## Go PostgreSQL App

Applicazione basata su PostgreSQL e sviluppata in Go.
DockerDocker ComposeGoPostgreSQL