package main  
  
import (  
 "github.com/gorilla/mux"  
 "net/http"  
 "os"  
 "os/signal"  
 "syscall"  
 "time"  
  
 "context"  
 "github.com/evgeniySeleznev/person-enrichment-service/pkg/logger"  
)  
  
type Server struct {  
 httpServer \*http.Server  
 logger logger.Logger  
 router \*mux.Router  
}  
  
func NewServer(port string, router \*mux.Router, logger logger.Logger) \*Server {  
 return &Server{  
 httpServer: &http.Server{  
 Addr: ":" + port,  
 Handler: router,  
 ReadTimeout: 10 \* time.*Second*,  
 WriteTimeout: 10 \* time.*Second*,  
 },  
 router: router,  
 logger: logger,  
 }  
}  
  
func (s \*Server) Start() {  
 go func() {  
 s.logger.Info("Starting server on " + s.httpServer.Addr)  
 if err := s.httpServer.ListenAndServe(); err != nil && err != http.ErrServerClosed {  
 s.logger.Fatal("Server error", err)  
 }  
 }()  
  
 quit := make(chan os.Signal, 1)  
 signal.Notify(quit, syscall.*SIGINT*, syscall.*SIGTERM*)  
 <-quit  
  
 s.logger.Info("Shutting down server...")  
 ctx, cancel := context.WithTimeout(context.Background(), 10\*time.*Second*)  
 defer cancel()  
  
 if err := s.httpServer.Shutdown(ctx); err != nil {  
 s.logger.Error("Server shutdown error", err)  
 }  
}