Go Minimal API Project Structure
Minimal HTTP API using only net/http, no external frameworks or routers.

#go #golang #api #http #rest #minimal #stdlib PNG PDF Copy Prompt

Project Directory

myapi/

Why This Structure?

Go 1.22+ has powerful routing built into net/http. This structure uses zero
external dependencies—just the standard library. Perfect for simple APIs,
learning Go, or when you want minimal attack surface.

Key Directories

main.go - Server setup, routing, and startup
internal/handler/ - HTTP handlers by domain
internal/model/ - Request/response structs
internal/config/ - Environment variables and settings

Go 1.22+ Routing

Getting Started

1. go mod init myapi
2. Create handlers in internal/handler/
3. Set up routes in main.go
4. go run .

When To Use This

Simple APIs with few endpoints
Zero external dependencies needed
Learning Go HTTP patterns
Microservices and Lambda functions
Internal tools and utilities

Trade-offs

No middleware chain - Implement middleware pattern yourself Manual validation - No request binding or validation
Basic routing - No route groups or complex patterns

Best Practices

Use Go 1.22+ for method-based routing
Keep handlers thin, logic in services
Return proper HTTP status codes
Add health check endpoint
Use context for cancellation

Naming Conventions

Packages - Singular nouns (handler, model, config) Handlers - Methods on Handler struct

Files - Lowercase, by domain (users.go, health.go)

main.go Entry point and…
go.mod
. gitignore
README.md
. env.example
Makefile Build and run c…
internal/ Private applica…
handler/ HTTP handlers
handler.go Handler depende…
health.go
users.go
model/
user.go

response.go JSON helpers
config/
config.go Env loading

// main.go
package main

import (
"log"
"net/http"
"myapi/internal/handler"
)

func main() {
h := handler.New()

mux := http.NewServeMux()
mux.HandleFunc("GET /health", h.Health) mux.HandleFunc("GET /users", h.ListUsers) mux.HandleFunc("GET /users/{id}", h.GetUser) mux.HandleFunc("POST /users", h.CreateUser)

log.Fatal(http.ListenAndServe(":8080", mux))
}