alterra
academy

Middleware

OUTLINE

alterra academy

- What is Middleware?
- Implementation Middleware
- Echo Middleware
- Type Echo Middleware
- Implementation
  - Log Middleware
  - Auth Middleware
  - JWT Middleware

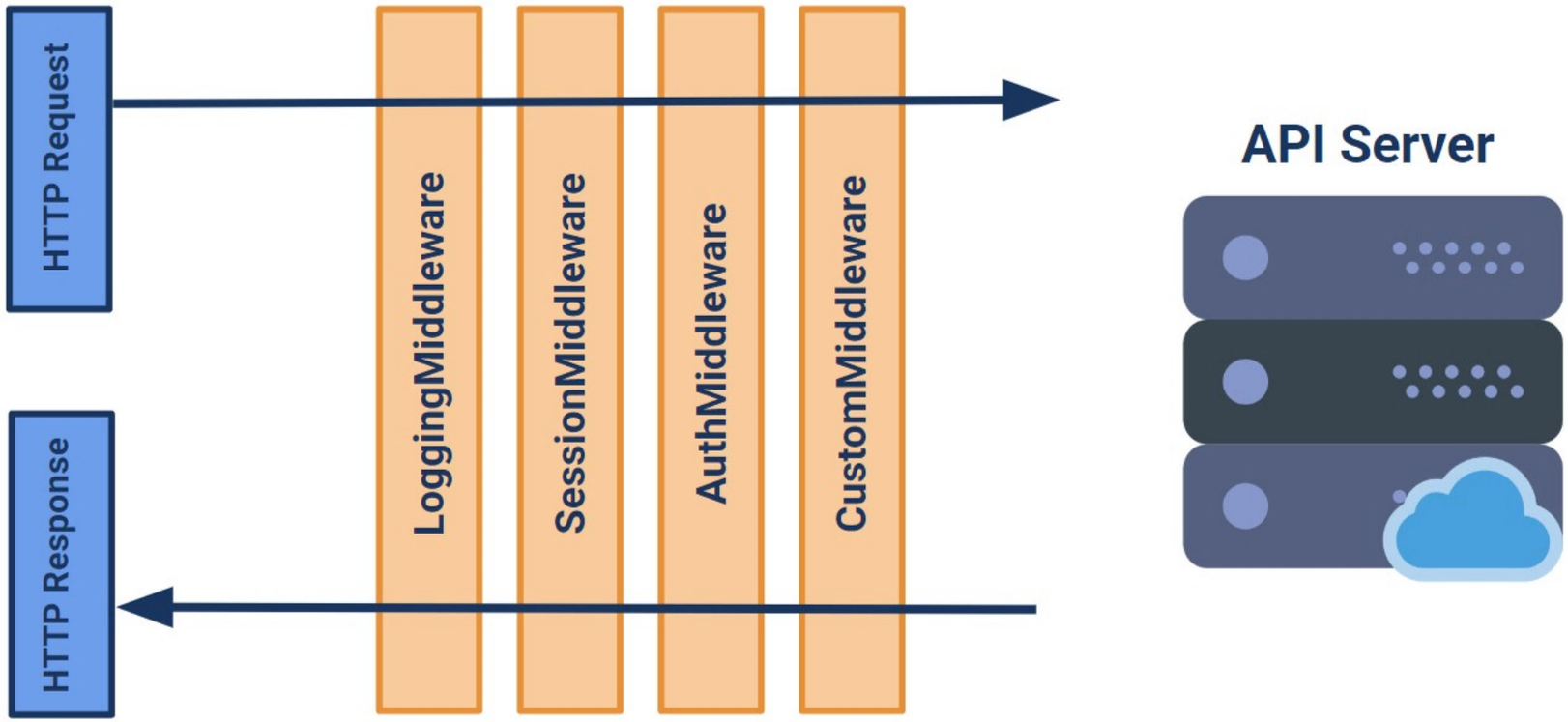
Middleware

alterra academy

Middleware is an entity that hooks into server's request/response processing.

Middleware, allow us to implement various functionalities around incoming HTTP requests to your server and outgoing responses.

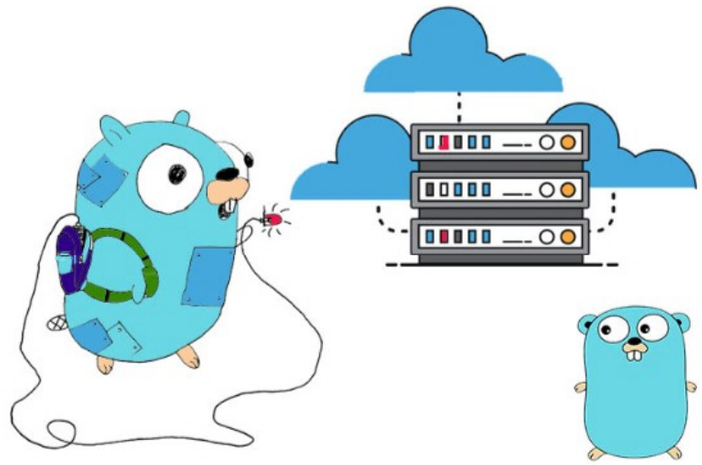
Implementation Middleware



alterra academy

Example Third Party Middleware

- Negroni
- Echo
- Interpose
- Alice
- Or you make own?



alterra academy

Middleware

- Basic Auth
- Body Dump
- Body Limit
- CORS
- CSRF
- Casbin Auth
- Gzip
- JWT
- Key Auth
- Logger
- Method Override
- Proxy
- Recover
- Redirect
- Request ID
- Rewrite
- Secure
- Session
- Static
- Trailing Slash

echo Middleware

SETUP MIDDLEWARE ECHO

alterra academy

Echo#Pre()

Executed before router processes the request

- HTTPSRedirect
- HTTPSWWWRedirect
- WWWRedirect
- AddTrailingSlash
- RemoveTrailingSlash
- MethodOverride
- Rewrite

Echo#Use()

Executed after router processes the request and has full access to echo.Context API

- BodyLimit
- Logger
- Gzip
- Recover
- BasicAuth
- JWTAuth
- Secure
- CORS
- Static

alterra academy

HTTPS REDIRECT

HTTPS redirect middleware redirects

http requests to https.

For example, http://labstack.com will be redirected to https://labstack.com.

middleware.go

```
e := echo.New()
e.Pre(middleware.HTTPSRedirect())
```

alterra academy

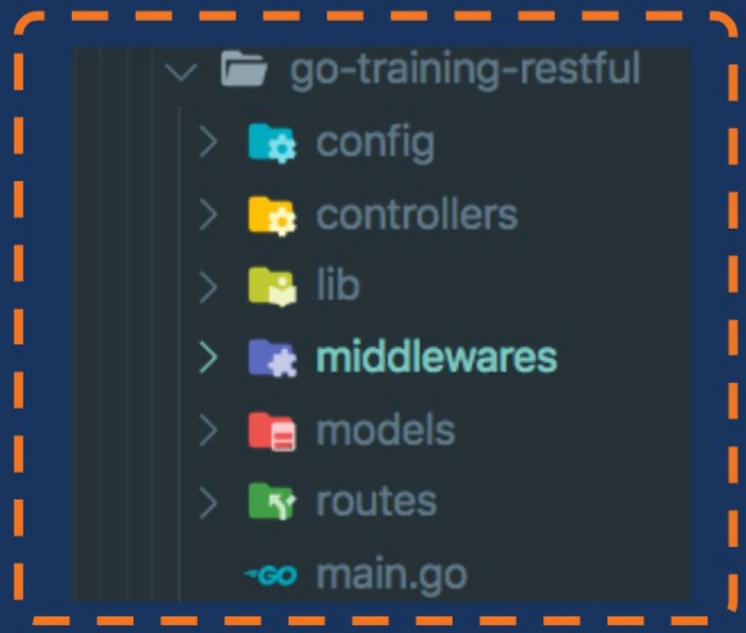
Log Middleware

alterra
academy

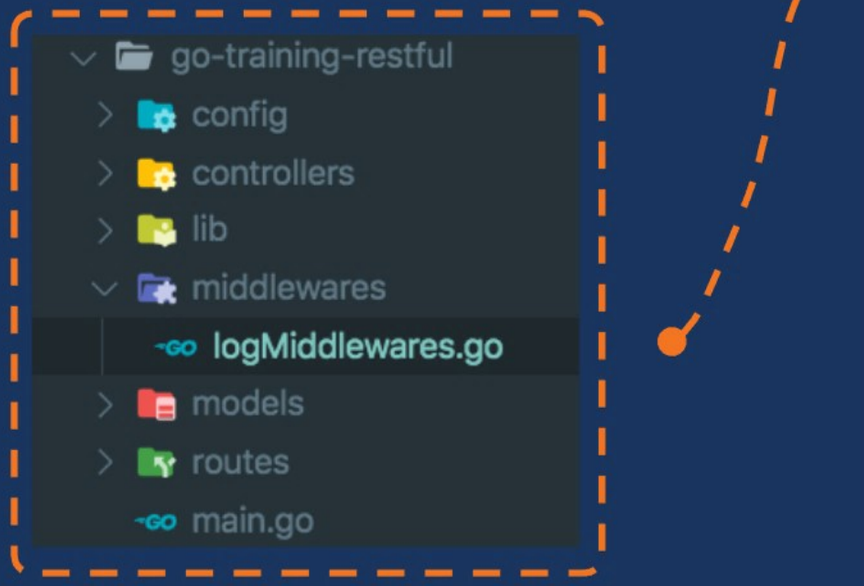
Logger Middleware

- **Logs** the information HTTP request.
- As a **footprint**.
- Datasource for **analytics**.

1 # CREATE FOLDER MIDDLEWARE



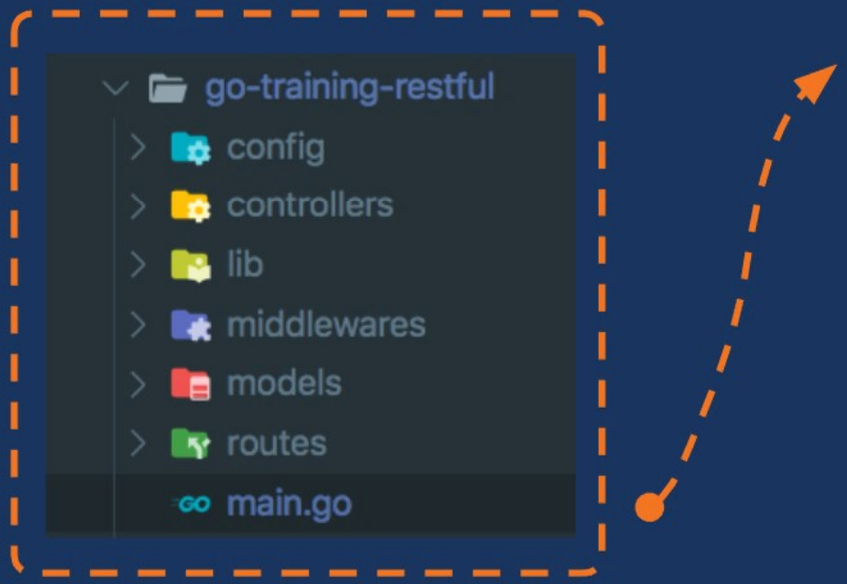
2 # CREATE
LOG MIDDLEWARE



Docs: https://echo.labstack.com/middleware/logger

```go
package middlewares

import (
  "github.com/labstack/echo"
  "github.com/labstack/echo/middleware"
)

func LogMiddlewares(e *echo.Echo) {

e.Use(middleware.LoggerWithConfig(middleware.LoggerConfig{
    Format: `[${time_rfc3339}] ${status} ${method}
${host}${path} ${latency_human}` + "\n",
  }))
}
```

3 # IMPLEMENT MIDDLEWARE



alterra academy

```go
package main

import(
  "github.com/iswanulumam/go-training-restful/routes"
  m "github.com/iswanulumam/go-training-restful/middleware
)

func main() {
  e := routes.New()
  // implement middleware logger
  m.LogMiddlewares(e)
  // start the server, and log if it fails
  e.Logger.Fatal(e.Start(":8000"))
}
```

4 # Result

alterra academy

```
PROBLEMS 8   OUTPUT   DEBUG CONSOLE   TERMINAL                    1: go

   ____    __
  / __/___/ /  ___
 / _// __/ _ \/ _ \
/___/\__/_//_/\___/ v4.1.6
High performance, minimalist Go web framework
https://echo.labstack.com
____________________________________O/_______
                                    O\
⇨ http server started on [::]:8000
[2019-07-25T09:47:33+07:00] 404 GET localhost:8000/user 64.75µs
[2019-07-25T09:47:39+07:00] 404 GET localhost:8000/hello 31.747µs
[2019-07-25T09:47:56+07:00] 200 GET localhost:8000/users 755.539µs
[2019-07-25T09:47:57+07:00] 200 GET localhost:8000/users 704.589µs
```

Result of implementation **logger middleware**

alterra academy

Auth Middleware

alterra
academy

WHY USING AUTHENTICATION

- User Identification
- Secure Data and Information

alterra academy

Authentication Middleware

Basic Authentication

JSON Web Token

alterra
academy

WHAT IS BASIC AUTHENTICATION?

Basic Auth is an http authentication technique request, this method requires username and password information to be inserted in the request header.

alterra academy

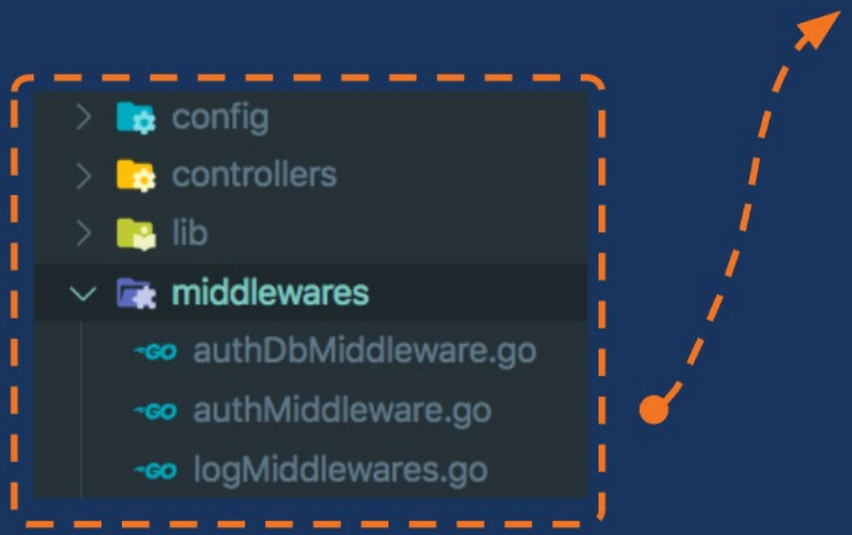
Information Encoded Format :

```
'Authorization: Basic ' +
base64encode('username:password')
```

Header Generate :

```
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

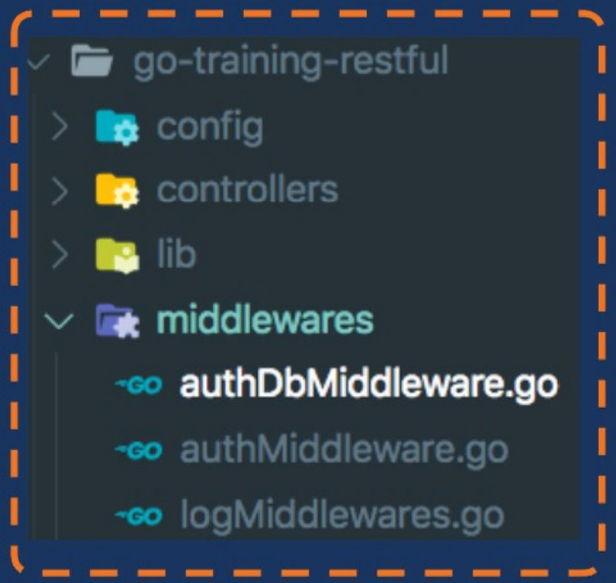
1 # CREATE
AUTH MIDDLEWARE



alterra academy

```go
func BasicAuth(username, password string, c echo.Context)
(bool, error) {
  if username == "admin" && password == "admin" {
    return true, nil
  }
  return false, nil
}
```

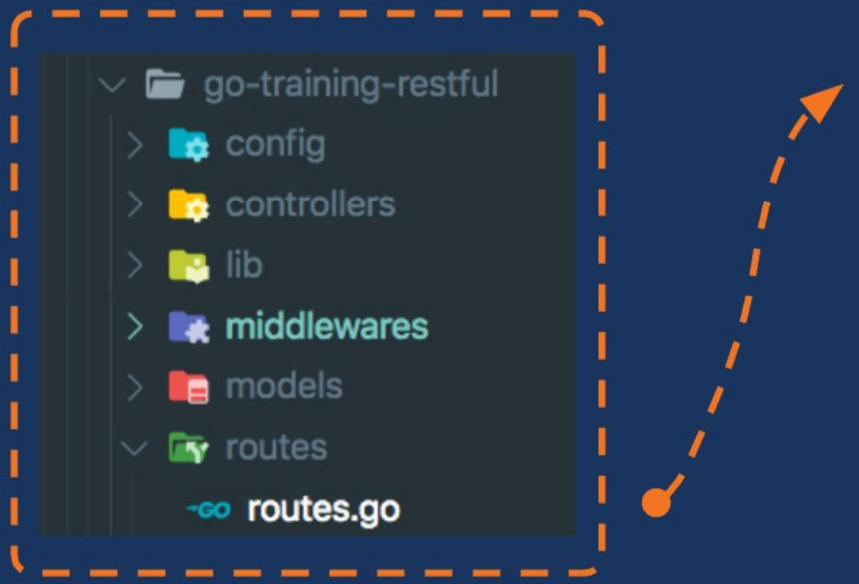
3 # CREATE
AUTH MIDDLEWARE (DB)



Docs: https://echo.labstack.com/middleware/logger

```go
package middlewares

import(
  "github.com/labstack/echo"
  "github.com/iswanulumam/go-training-restful/config"
  "github.com/iswanulumam/go-training-restful/models"
)

var db = config.DB

func BasicAuthDB(username, password string, c
echo.Context) (bool, error) {
  var db = config.DB
  var user models.User
  if err := db.Where("email = ? AND password = ?",
username, password).First(&user).Error; err != nil {
    return false, nil
  }
  return true, nil
}
```

2 # IMPLEMENT MIDDLEWARE



alterra academy

```go
package routes

import (
  c "github.com/iswanulumam/go-training-restful/controllers"
  m "github.com/iswanulumam/go-training-restful/middlewares"

  echoMid "github.com/labstack/echo/middleware"
  "github.com/labstack/echo"
)

func New() *echo.Echo {
  e := echo.New()

  e.GET("/users", c.GetUsersController)
  e.GET("/users/:id", c.GetUserController)
  e.POST("/users", c.CreateUserController)

  // implement middleware with group routing
  eAuth := e.Group("")
  eAuth.Use(echoMid.BasicAuth(m.BasicAuthDB))

  eAuth.DELETE("/users/:id", c.DeleteUserController)
  eAuth.PUT("/users/:id", c.UpdateUserController)

  return e
}
```

alterra academy

JWT Middleware

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

alterra academy

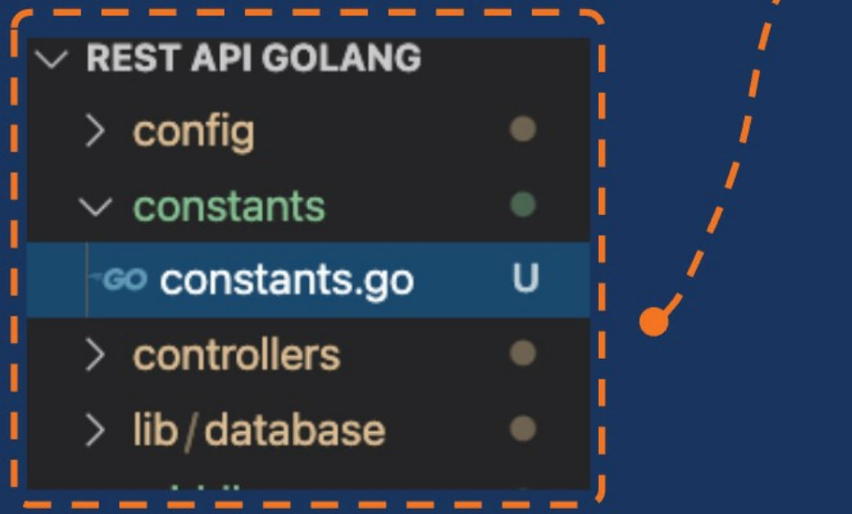
Information Encoded Format :

`'Authorization: Bearer + Token`

Header Generate :

```
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXRob3J
pemVkIjp0cnVlLCJleHAiOjE2MTQwNzU3NDMsInVzZXJJZCI
6MX0.3RGYBl7pwh7J86Cmhcp78AMi3LU-10_Wuu5Rt098vCk
```

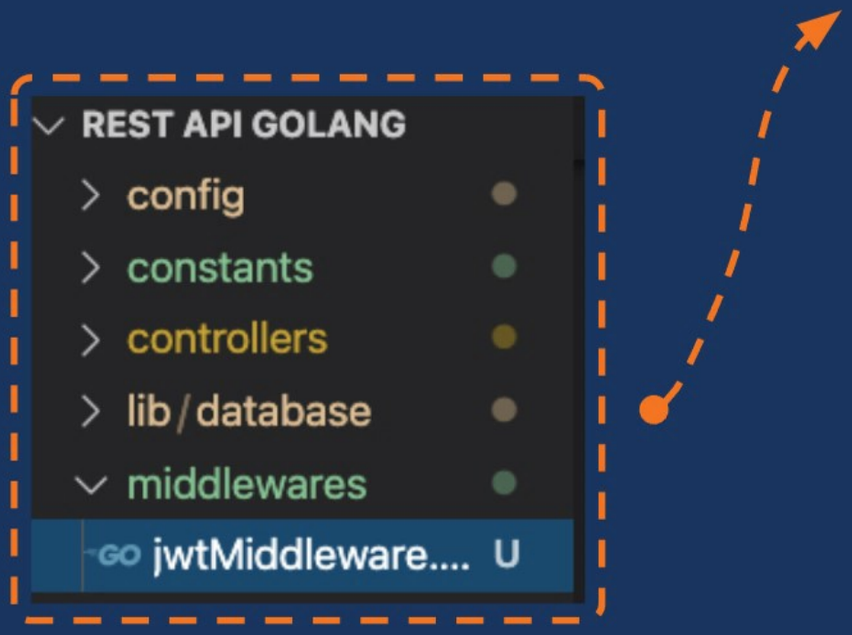
1 # CREATE CONSTANTS




```go
package constants

const SECRET_JWT = "legal"
```

alterra academy

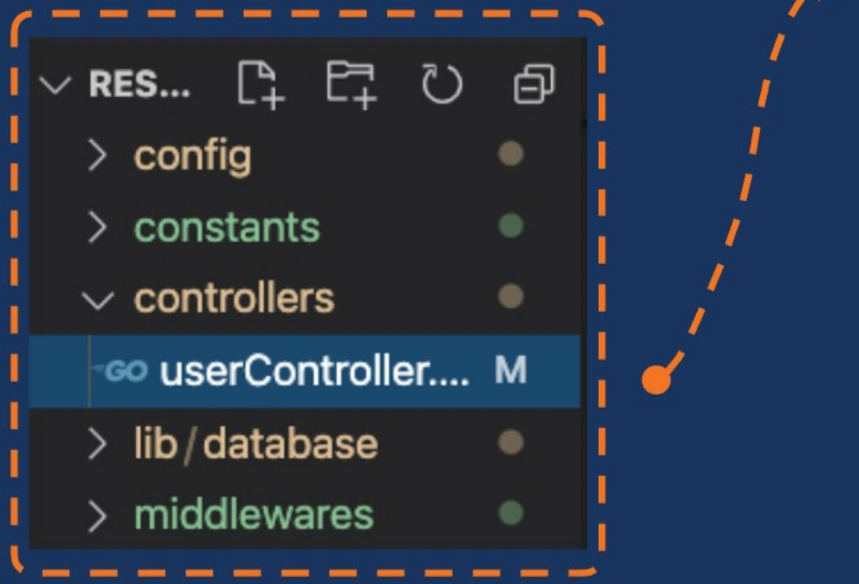
2 # CREATE
JWT MIDDLEWARE



alterra academy

```go
package middlewares

import (
    "project/constants"
    "time"

    jwt "github.com/dgrijalva/jwt-go"
    "github.com/labstack/echo"
)

func CreateToken(userId int) (string, error) {
    claims := jwt.MapClaims{}
    claims["authorized"] = true
    claims["userId"] = userId
    claims["exp"] = time.Now().Add(time.Hour * 1).Unix() //Token
expires after 1 hour
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    return token.SignedString([]byte(constants.SECRET_JWT))

}

func ExtractTokenUserId(e echo.Context) int {
    user := e.Get("user").(*jwt.Token)
    if user.Valid {
        claims := user.Claims.(jwt.MapClaims)
        userId := claims["userId"].(int)
        return userId
    }
    return 0
}
```

3 # IMPLEMENT
LOGIN CONTROLLER



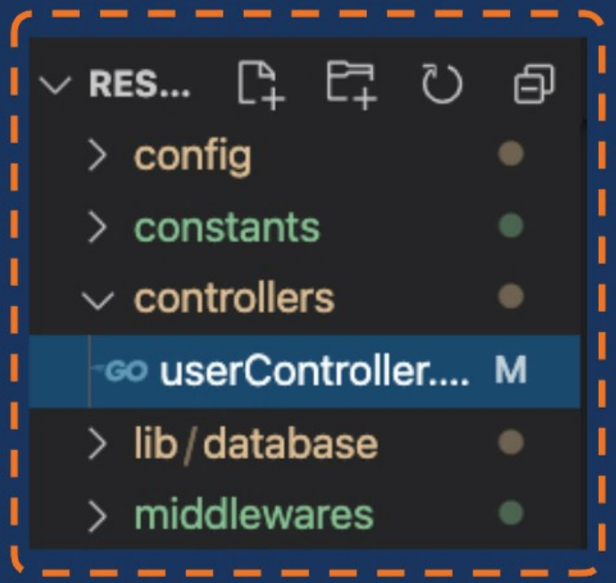
Docs:

alterra academy

```
...
func LoginUsersController(c echo.Context) error {
   user := models.Users{}
   c.Bind(&user)

   users, e := database.LoginUsers(&user)
   if e != nil {
       return echo.NewHTTPError(http.StatusBadRequest,
e.Error())
   }
   return c.JSON(http.StatusOK, map[string]interface{}{
       "status": "success login",
       "users":  users,
   })
}


...
```

4 # IMPLEMENT

DETAIL CONTROLLER #2

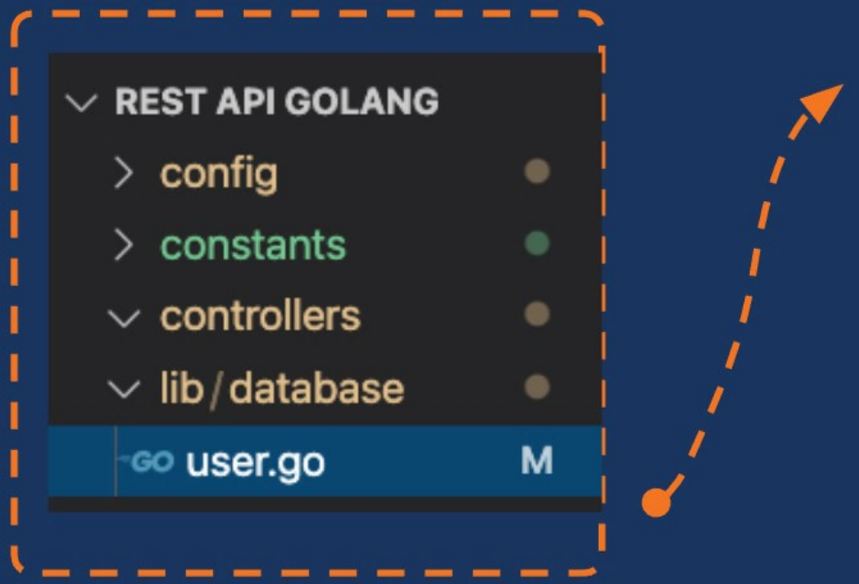


Docs:

alterra academy

```
...

func GetUserDetailControllers(c echo.Context) error {
   id, err := strconv.Atoi(c.Param("id"))

   if err != nil {
       return echo.NewHTTPError(http.StatusBadRequest,
err.Error())
   }

   users, err := database.GetDetailUsers((id))

   if err != nil {
       return echo.NewHTTPError(http.StatusBadRequest,
err.Error())
   }

   return c.JSON(http.StatusOK, map[string]interface{}{
       "status": "success",
       "users":  users,
   })
}


...
```

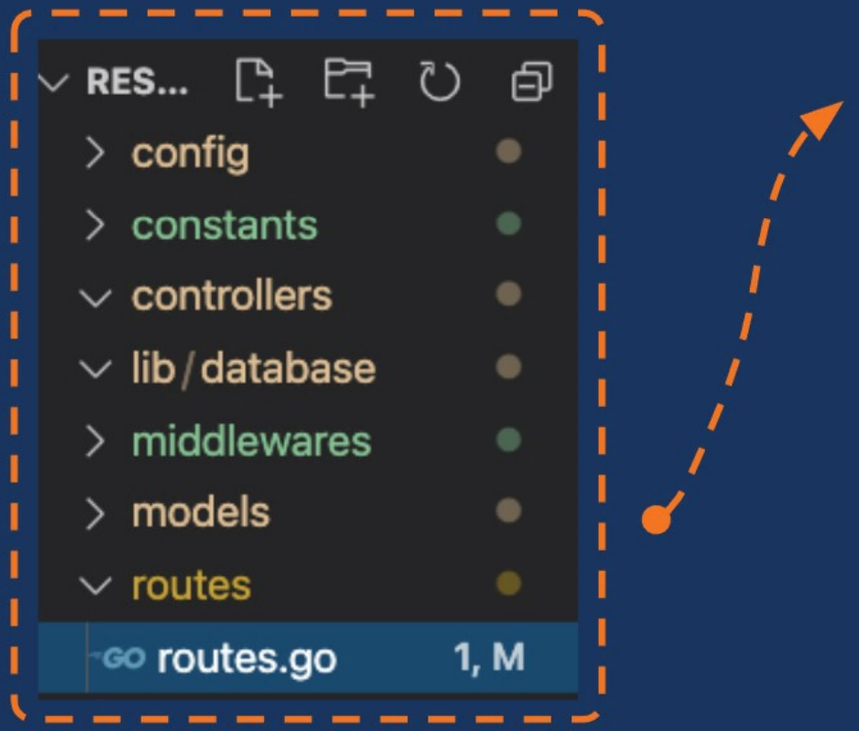
5 # IMPLEMENT
LIB DATABASE



alterra academy

```go
func GetDetailUsers(userId int) (interface{}, error) {
    var user models.Users

    if e := config.DB.Find(&user, userId).Error; e != nil {
        return nil, e
    }
    return user, nil
}

func LoginUsers(user *models.Users) (interface{}, error) {

    var err error
    if err = config.DB.Where("email = ? AND password = ?",
user.Email, user.Password).First(user).Error; err != nil {
        return nil, err
    }

    user.Token, err = middlewares.CreateToken(int(user.ID))
    if err != nil {
        return nil, err
    }

    if err := config.DB.Save(user).Error; err != nil {
        return nil, err
    }

    return user, nil
}
```

6 # IMPLEMENT ROUTE



alterra academy

```go
func New() *echo.Echo {
    e := echo.New()

    e.GET("/users", controllers.GetUserControllers)
    e.GET("/users/:id", controllers.GetUserDetailControllers)
    e.POST("/users", controllers.CreateUsersController)
    e.POST("/login", controllers.LoginUsersController)

    // JWT Group
    r := e.Group("/jwt")
    r.Use(middleware.JWT([]byte(constants.SECRET_JWT)))
    r.GET("/users/:id", controllers.GetUserDetailControllers)

    return e
}
```

.end

RESTful API