alterra
academy

ORM & Code Structure

alterra academy

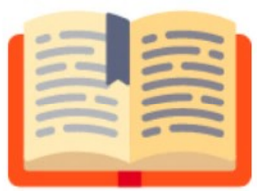
OUR RULES

Silent Mode

Ask Question

Go Toilet

TIME ALLOCATION



Explanation

Challenge

Review

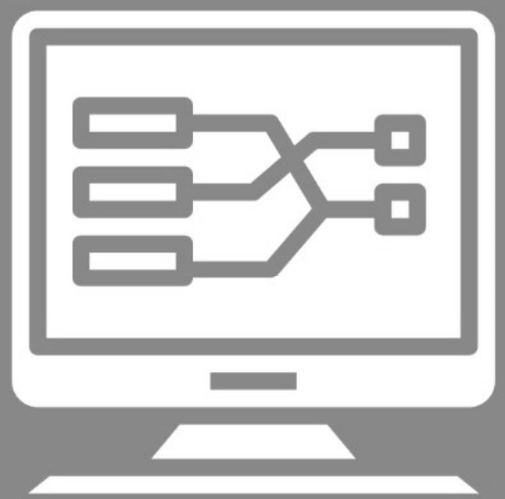
OUTLINE

alterra academy

- What is Object Relational Mapping (ORM) ?
- What is GORM?
- Database Connection & Migration
- CRUD RESTful
- What is MVC?
- Why Need Structure?
- Structuring Project

alterra academy

Let's get started!



alterra academy

ORM

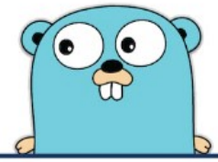
Object Relational Mapping

Object-relational mapping (ORM, O/RM, and O/R mapping tool) in computer science is a ***programming technique*** *for* ***converting data between incompatible type*** *systems using* ***object-oriented programming*** *languages*

(https://en.wikipedia.org/wiki/Object-relational_mapping)

alterra
academy

WHAT IS GORM?



The fantastic ORM library
for Golang

Object Relational Mapping

Table : person

ID	Name	Age	Sex
1	Budi	40	male
2	Yanto	41	male
3	Winda	26	female

Object representative

```
type Person struct {
  gorm.Model
  ID   uint64
  Name string
  Age  int32
  Sex  string
}

var person Person
person.ID = 1
person.Name = "Alex"
person.Age = 22
person.Sex = "Male"
```

alterra academy

ORM Advantages

Less repetitive query
Automatically fetch data into ready to use object
Simple way if you want to screening data before store it in database
Some have feature cache query

ORM Disadvantages

alterra academy

Add a layer in code and cost the overhead process
Load un-necessary relationship data
Complex raw query can be longy to write with ORM (>10 table joins)
Specific SQL function related to one vendor may not supported or no specific function (MySQL : FORCE INDEX)

DATABASE MIGRATION

alterra academy

A way to update the database version to comply the application version changes.

Changes can be upgrade to newest version or a rollback to the previous version.

Why DB Migration

alterra academy

Database update simplicity
Database rollback simplicity
Track changes on database structure
Database structure history writed on a code
Always compatible with application version changes

alterra academy

STEP

Install GORM

- GORM: ORM Library for Golang

https://gorm.io/docs/

- GORM-Migrate

https://gorm.io/docs/migration.html

alterra

INSTALL GORM & MySQL DRIVER

-terminal

```
$ go get -u gorm.io/gorm
$ go get -u gorm.io/driver/mysql
```

alterra academy

How to connect go application With Database ?

Connection for MySQL Using GORM

```
<username>:<password>@/<db_name>?charset=utf8&parseTime=True&loc=Local
```

Step 1 # Create InitDB() for Database Connection

alterra

-code editor

```
// database connection
func InitDB() {

  // declare struct config & variable connectionString

  var err error
  DB, err = gorm.Open("mysql", connectionString)
  if err != nil {
    panic(err)
  }
}
```

Function InitDB() for crate connection application to database

Step 2 # Create User Schema & InitialMigration()

alterra

-code editor

```
# src/main.go
...
var (
  DB *gorm.DB
)

type User struct {
  gorm.Model
  Name     string `json:"name" form:"name"`
  Email    string `json:"email" form:"email"`
  Password string `json:"password" form:"password"`
}

func InitialMigration() {
  DB.AutoMigrate(&User{})
}
...
```

Create User schema for database using Struct (Object)

function InitialMigration() for create database in mysql

Step 3 # Call InitDB() and InitialMigration()

alterra

-code editor

```
package main

// import library

func init() {
  InitDB()
  InitialMigration()
}

...
```

Calling function InitDB() and InitialMigration() for use it.

Step 4 # Create GetUsersController()

alterra

-code editor

```
...

// get all users
func GetUsersController(c echo.Context) error {
  var users []User

  if err := DB.Find(&users).Error; err != nil {
    return echo.NewHTTPError(http.StatusBadRequest, err.Error())
  }
  return c.JSON(http.StatusOK, map[string]interface{}{
    "message": "success get all users",
    "users":   users,
  })
}

...
```

Create GetUserController() for get data users from database (ORM) and showing data to the resposes.

Step 5 # Create CreateUserController()

alterra

-code editor

```
...

// create new user
func CreateUserController(c echo.Context) error {
  user := User{}
  c.Bind(&user)

  if err := DB.Save(&user).Error; err != nil {
    return echo.NewHTTPError(http.StatusBadRequest, err.Error())
  }
  return c.JSON(http.StatusOK, map[string]interface{}{
    "message": "success create new user",
    "user":    user,
  })
}

...
```

Create CreateUserController() for binding user data from client and save user to database

Step 6 # Routing

alterra

-code editor

```
...

func main() {
  // create a new echo instance
  e := echo.New()

  // Route / to handler function
  e.GET("/users", GetUsersController)
  e.POST("/users", CreateUserController)

  // start the server, and log if it fails
  e.Start(":8000")
}

...
```

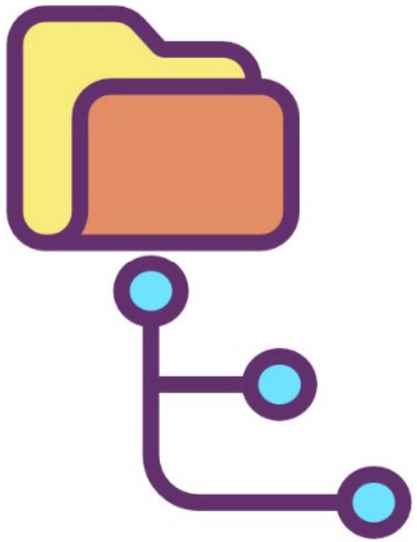
Define RESTful api routing by calling the Controller

CHALLENGE 1

alterra

Complete All REST API Routes:

```
e.GET("/users/:id", GetUserController)
e.DELETE("/users/:id", DeleteUserController)
e.PUT("/users/:id", UpdateUserController)
```



Code Structure

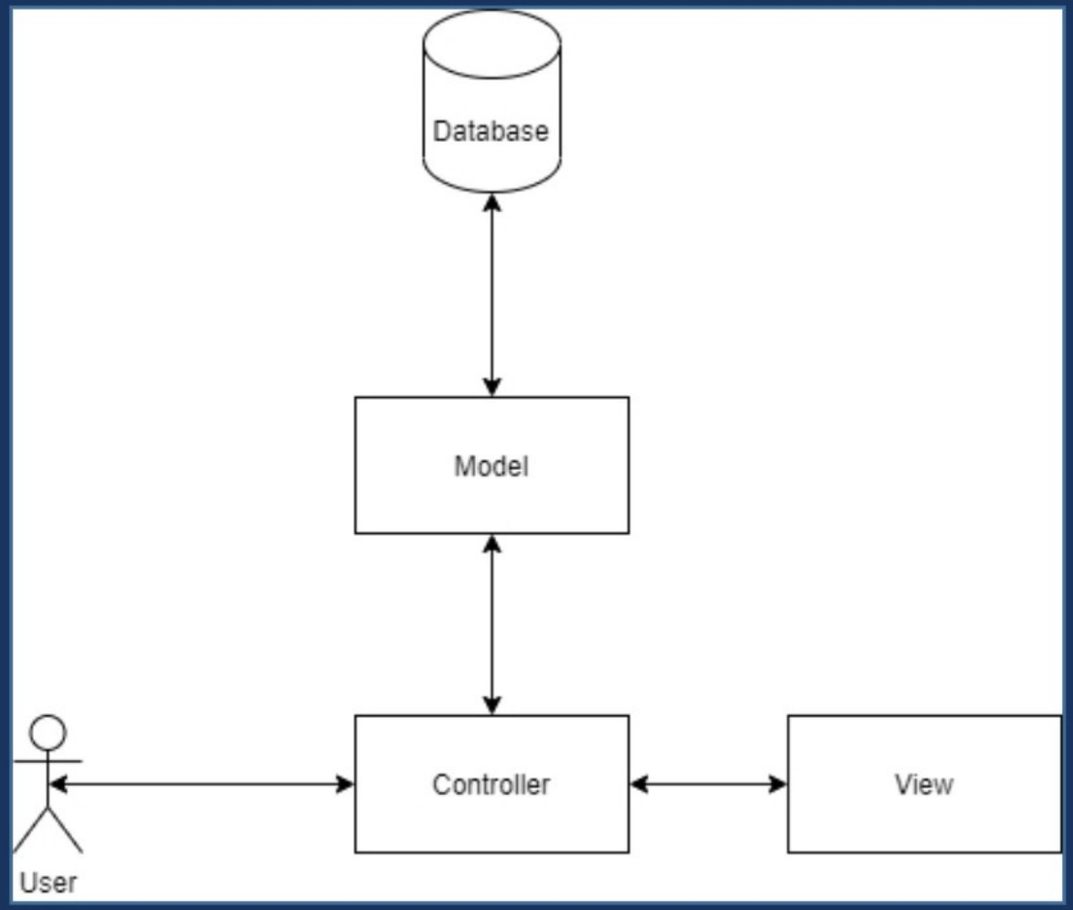
Structuring your project using Model-View-Controller (MVC)

alterra academy

MVC is short for **Model**, **View**, and **Controller**.
MVC is a popular way of organizing your code.

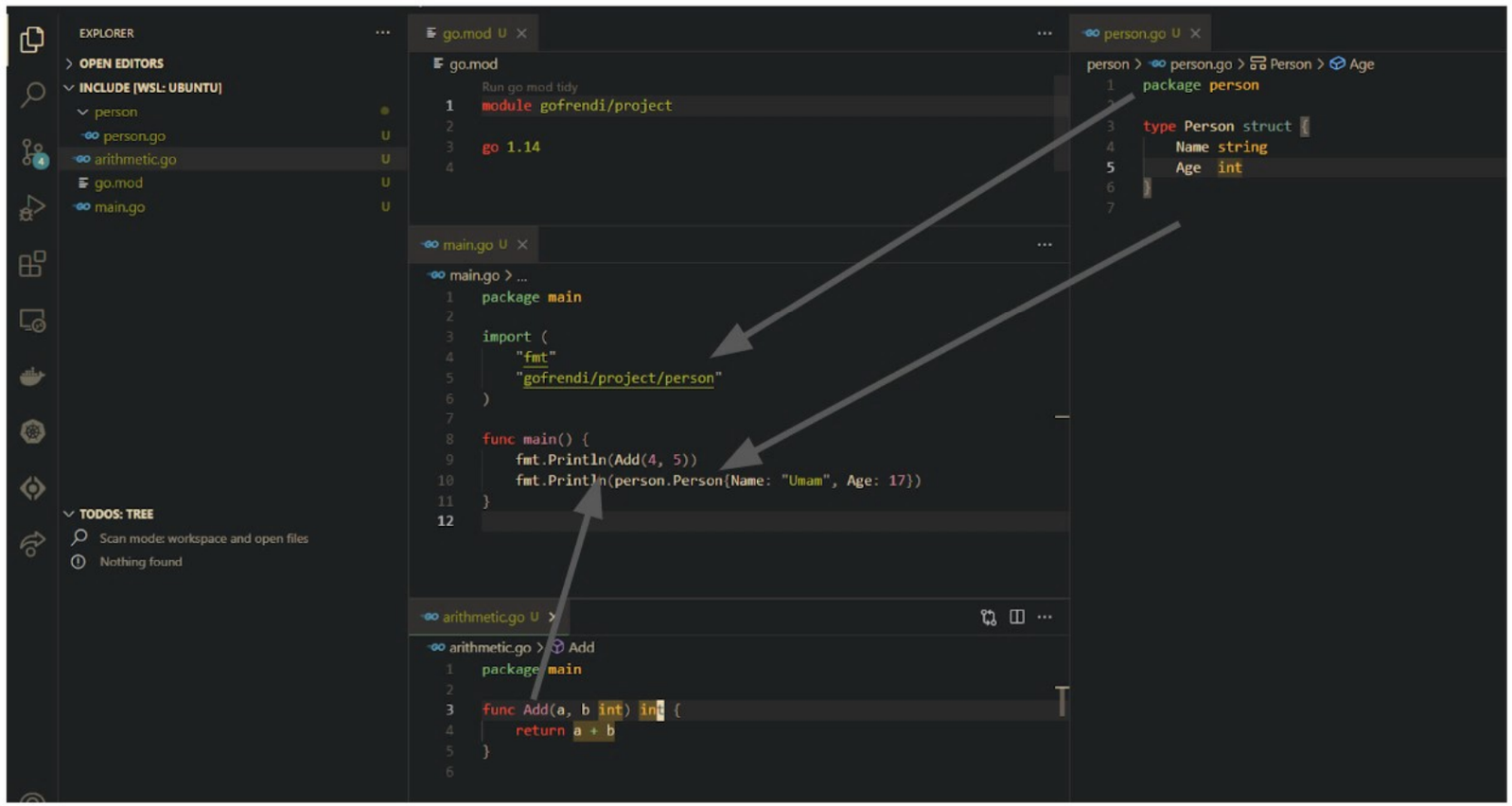
The big idea behind MVC is that **each section of your code has a purpose**, and those purposes are different.

alterra academy



Using Resources From Other Packages

alterra academy



alterra academy

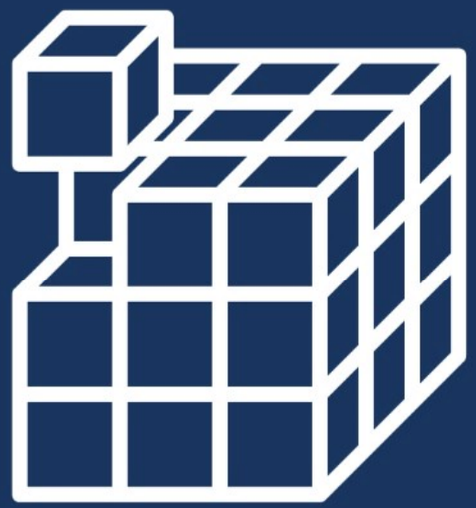
Why Need Structure ?

To achieve to **modular application**.

Implement **separation of concerns**.

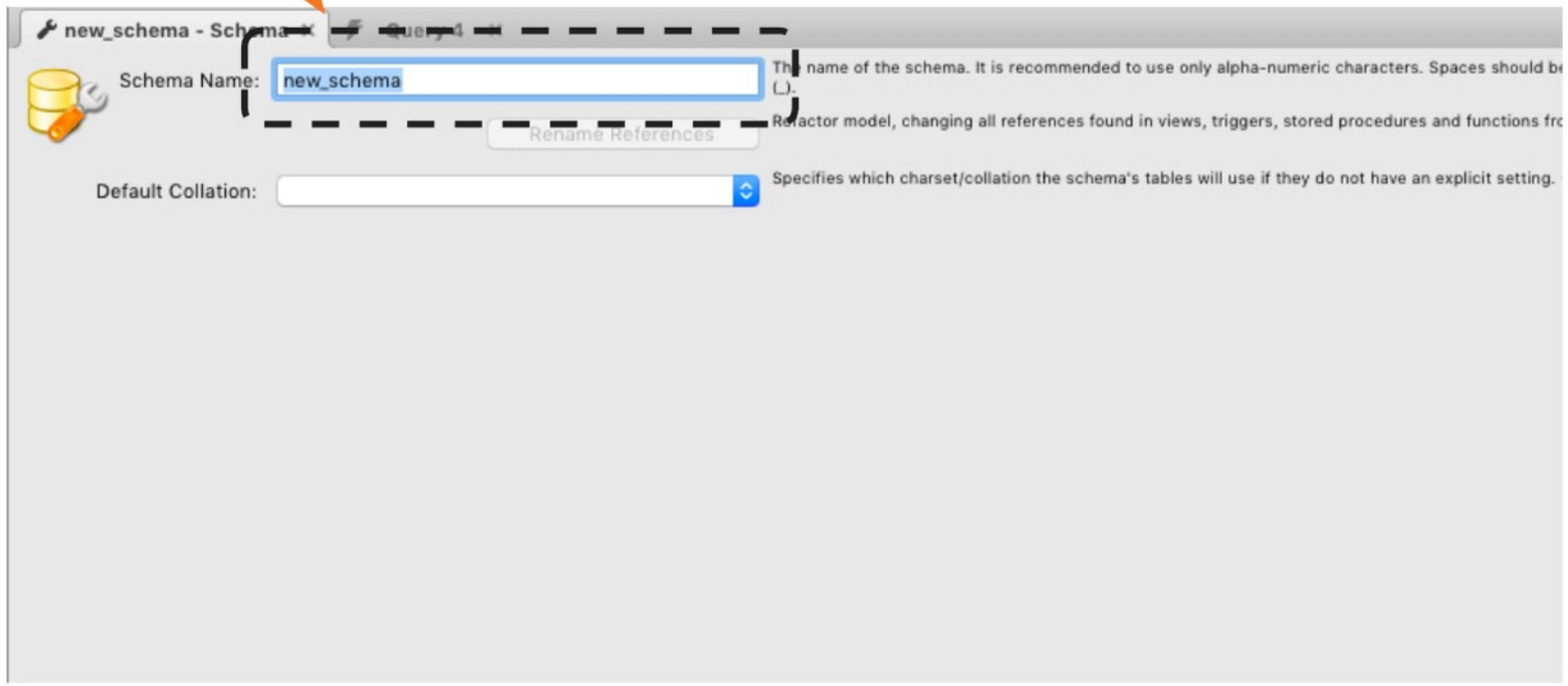
Less conflict on versioning.

alterra
academy



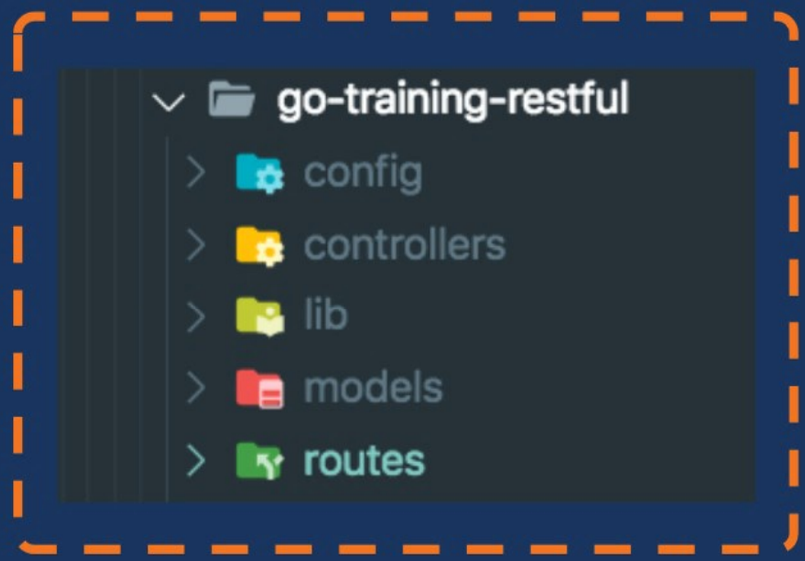
Code Structuring

1 # CREATE DATABASE IN MYSQL

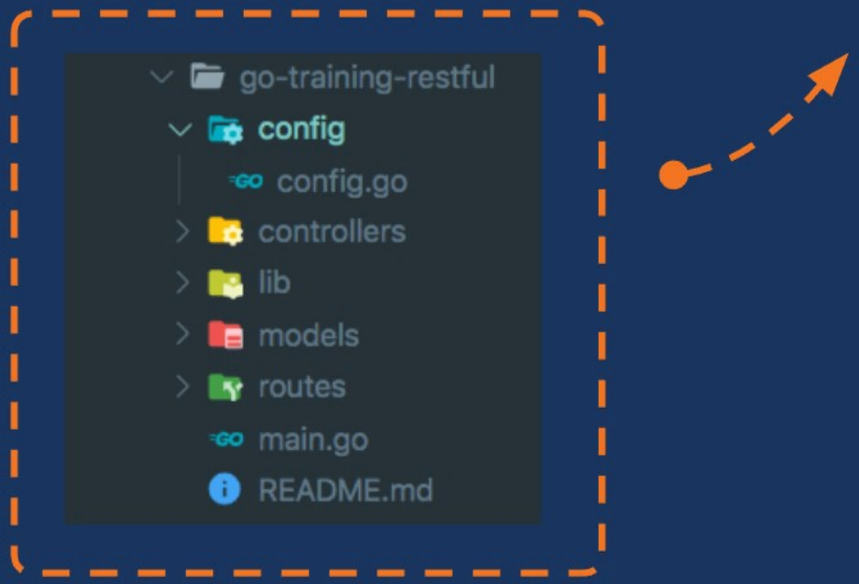


alterra academy

2 # CREATE FOLDER STRUCTURE



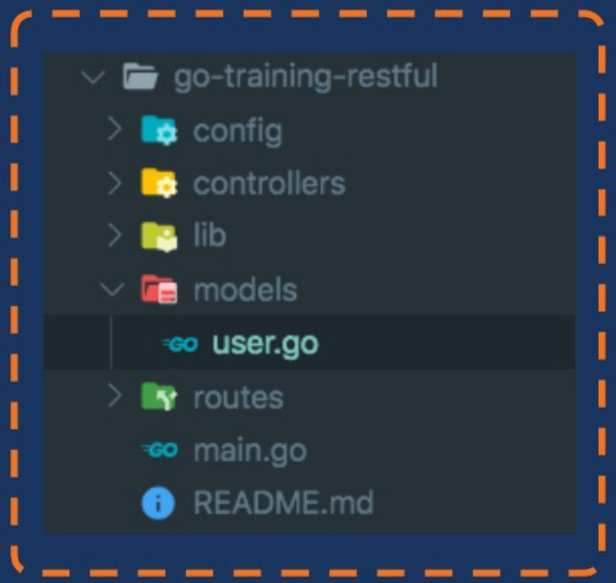
3 # CREATE CONFIGURATION



alterra academy

```go
import (
    "fmt"
    "project/models"
    "gorm.io/driver/mysql"
    "gorm.io/gorm"
)

var DB *gorm.DB

func InitDB() {

    config := map[string]string{
        "DB_Username": "root",
        "DB_Password": "123ABC4d.",
        "DB_Port":     "3306",
        "DB_Host":     "127.0.0.1",
        "DB_Name":     "training",
    }

    connectionString :=
fmt.Sprintf("%s:%s@tcp(%s:%s)/%s?charset=utf8&parseTime=True&loc=Loc
al",
        config["DB_Username"],
        config["DB_Password"],
        config["DB_Host"],
        config["DB_Port"],
        config["DB_Name"])

    var e error
    DB, e = gorm.Open(mysql.Open(connectionString), &gorm.Config{})
    if e != nil {
        panic(e)
    }
    InitMigrate()
}

func InitMigrate() {
    DB.AutoMigrate(&models.Users{})
}
```

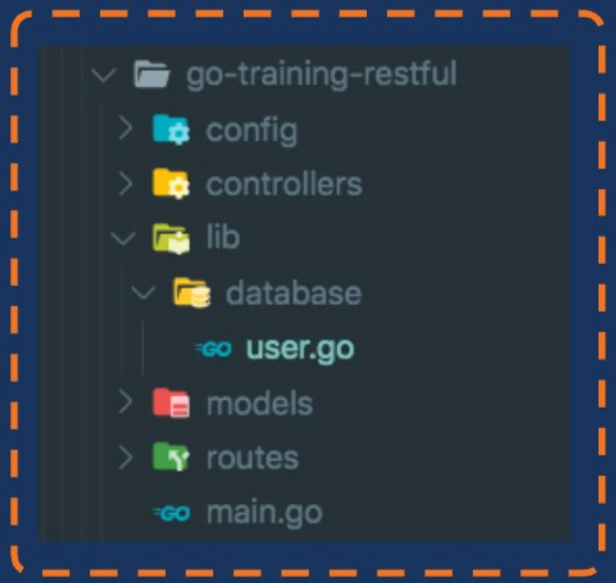
4 # CREATE MODEL



alterra academy

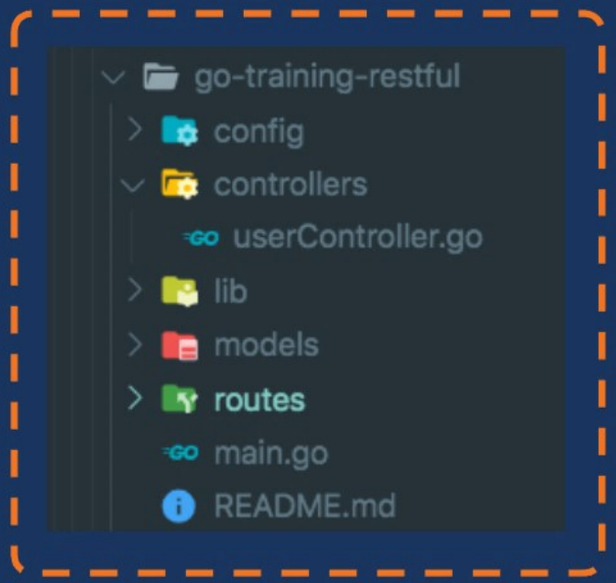
```go
package models

import (
  "github.com/jinzhu/gorm"
)

type User struct {
  gorm.Model
  Name     string `json:"name" form:"name"`
  Email    string `json:"email" form:"email"`
  Password string `json:"password" form:"password"`
}
```

5 # CREATE
LIB DATABASE




```go
package database

import (
    "project/config"
    "project/models"
)

func GetUsers() (interface{}, error) {
    var users []models.Users

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

6 # CREATE CONTROLLER

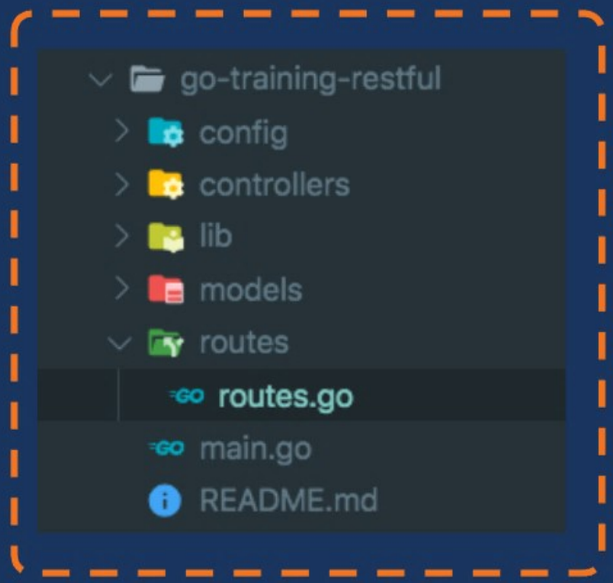


alterra academy

```go
package controllers

import (
    "net/http"
    "project/lib/database"
    "project/models"

    "github.com/labstack/echo"
)

func GetUserControllers(c echo.Context) error {
    users, e := database.GetUsers()

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

. . .

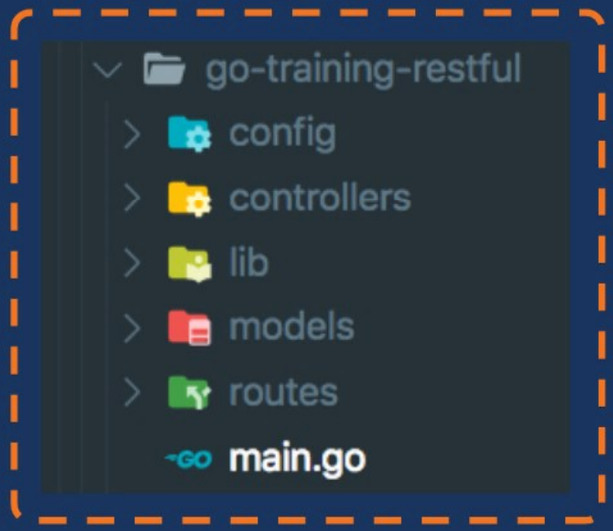
7 # CREATE ROUTER




```go
package routes

import (
    "project/controllers"

    "github.com/labstack/echo"
)

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

    e.GET("/users", controllers.GetUserControllers)

    return e
}
```

alterra academy

8 # CREATE MAIN




```go
package main

import (
    "project/config"
    "project/routes"
)

func main() {
    config.InitDB()
    e := routes.New()
    e.Logger.Fatal(e.Start(":8000"))
}
```

alterra academy

alterra academy

RUNNING echo

-terminal

```
$ go run main.go
App listening on port :8080
   ____    __
  / __/___/ /  ___
 / _// __/ _ \/ _ \
/___/\__/_//_/\___/ v4.1.4
High performance, minimalist Go web framework
https://echo.labstack.com
____________________________________O/_______
                                    O\
⇨ http server started on [::]:8000
```

alterra

Terimakasih