**TRƯỜNG ĐẠI HỌC TIỀN GIANG**

**Khoa Công Nghệ Thông Tin**

**🙞🙞🙞•🔿🔿•🙜🙜🙜**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**PHÁT TRIỂN PHẦN MỀM NGỒN MỞ**

**ĐỀ TÀI**

**TÌM HIỂU VỀ GOLANG & ỨNG DỤNG VÀO WEB BACK-END**

**Sinh viên thực hiện:**

**Nguyễn Đông Thức 020101060**

**Nguyễn Duy Tân 020101125**

**Lớp: ĐHCNTT20B**

**GV hướng dẫn: Đoàn Chí Trung**

**Tiền Giang, tháng 5 năm 2023**

**PHIẾU ĐÁNH GIÁ ĐỒ ÁN**

1. MSSV:

2. MSSV:

**Họ và tên người đánh giá:**

**Người đánh giá là:**  🞏 Người hướng dẫn 🞏 Người phản biện

Người đánh giá dựa vào việc đọc quyển báo cáo, xem demo sản phẩm, dự phiên chấm đồ án để nhận xét đồ án theo các gợi ý sau:

**1. Về quyển báo cáo**

**Về hình thức**: quyển báo cáo có hình thức trình bày và cấu trúc theo đúng quy định? Câu văn có mạch lạc, sáng sủa? Không có lỗi chính tả?

**Về phần tổng quan**: tác giả có trình bày một cách rõ ràng các nội dung cần thiết hay không? Nội dung cần thiết có thể bao gồm: mục tiêu, nhiệm vụ, phạm vi của đồ án; các chức năng, công dụng của phần mềm cần đạt được trong đồ án; bối cảnh thực tiễn, ứng dụng của đồ án; giải thuật, phương pháp, nhiệm vụ đặt ra trong đồ án; công nghệ được sử dụng trong đồ án...

**Về phân tích, thiết kế, cài đặt giải pháp và kết quả đề tài:** Các giải pháp phân tích, thiết kế, cài đặt để giải quyết nhiệm vụ của đồ án có hợp lý? Đề tài mới hoặc phương pháp thực hiện có tính sáng tạo? Chất lượng và ý nghĩa của kết quả đạt được như thế nào?

**2. Về sản phẩm/ chương trình demo:** các chức năng của chương trình có đáp ứng yêu cầu đặt ra? Sản phẩm có khả năng ứng dụng trong thực tiễn?

**3. Về báo cáo và trả lời chất vấn:** báo cáo có tốt không? (trình bày rõ ràng, đúng thời gian). Trả lời chất vấn có tốt không (hiểu đúng câu hỏi, trả lời đúng và tập trung vào vấn đề đặt ra, không lạc đề)?

|  |  |
| --- | --- |
|  | Người đánh giá  (Ký và ghi rõ họ tên) |

**LỜI CẢM ƠN**

**🙞🙞🕮🙜🙜**

Chúng em xin gửi lời cảm ơn chân thành đến thầy Đoàn Chí Trung là giáo viên hướng dẫn đồ án cho chúng em. Thầy đã luôn theo sát quá trình thực hiện đồ án, nhiệt tình hướng dẫn, chỉ bảo để chúng em hoàn thành đồ án này.

Trong quá trình làm đồ án, tuy chúng em đã cố gắng hết sức để tìm hiểu, trao dồi kiến thức để có thể hoàn thành tốt đồ án của mình nhưng chắc chắn không tranh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và góp ý của quý thầy cô.

Chúng em xin chân thành cảm ơn!

Tiền Giang, tháng 5 năm 2023

Nhóm sinh viên thực hiện

**CHƯƠNG 1: Tổng quan**

* 1. **Giới thiệu đề tài**

**1.1.1 Lý do chọn đề tài:**

Đề tài của em là tìm hiểu về golang và ứng dụng của nó trong phát triển ứng dụng web back-end. Em chọn đề tài này vì em muốn tìm hiểu về một ngôn ngữ lập trình mới và tiềm năng, có thể giải quyết được những vấn đề và thách thức trong lĩnh vực phần mềm hiện nay.

Golang là một ngôn ngữ lập trình được thiết kế bởi Google, ra mắt vào năm 2009. Golang là một ngôn ngữ mã nguồn mở, có thể được sử dụng và phát triển bởi bất kỳ ai. Golang có nhiều ưu điểm như hiệu năng cao, đa nền tảng, đa luồng và cú pháp đơn giản. Golang có thể chạy trên nhiều hệ điều hành khác nhau, từ Windows, Linux, Mac OS cho đến các thiết bị nhúng. Golang cũng hỗ trợ việc lập trình đồng thời, cho phép xử lý nhiều tác vụ cùng lúc mà không gặp phải các vấn đề như deadlock hay race condition. Golang cũng có cú pháp đơn giản và rõ ràng, dễ đọc và bảo trì. Golang không có các tính năng phức tạp như kế thừa hay đa hình, mà sử dụng các khái niệm như interface và struct để tổ chức mã nguồn.

Golang rất phù hợp cho việc phát triển ứng dụng web back-end, vì nó có thể xử lý nhanh và an toàn các yêu cầu từ người dùng, cơ sở dữ liệu và các dịch vụ khác. Golang cũng có nhiều thư viện và công cụ hỗ trợ cho việc tạo ra các chức năng web back-end như xác thực, API, mã hóa và nhiều hơn nữa. Golang là một ngôn ngữ lập trình tiên tiến và đáng tin cậy cho việc phát triển ứng dụng web back-end.

Em hy vọng đề tài này sẽ giúp em nâng cao kỹ năng lập trình và hiểu biết về golang và ứng dụng web back-end.

**1.1.2 Mục tiêu:**

* Tìm hiểu về golang và các tính năng của nó.
* Tìm hiểu về các thư viện và công cụ hỗ trợ cho việc phát triển web back-end bằng golang.
* Thực hành viết code bằng golang để xây dựng các ứng dụng web back-end.

**CHƯƠNG 2: Golang cơ bản**

**2.1 Các cú pháp phổ biến của Golang:**

**Khai báo gói**

Mỗi chương trình golang phải thuộc về một gói. Chúng ta sử dụng từ khóa **package** để khai báo gói.

Ví dụ:

package main

Trong ví dụ trên, chúng ta khai báo chương trình thuộc về gói **main**. Gói **main** là gói đặc biệt, được sử dụng cho các chương trình có thể thực thi. Các gói khác thường được sử dụng để cung cấp các thư viện hoặc các hàm cho các gói khác.

**Nhập gói**

Để sử dụng các hàm hoặc các kiểu dữ liệu được định nghĩa trong các gói khác, chúng ta phải nhập gói đó vào chương trình. Chúng ta sử dụng từ khóa **import** để nhập gói. Ví dụ:

import "fmt"

Trong ví dụ trên, chúng ta nhập gói **fmt** vào chương trình. Gói **fmt** là một gói trong thư viện chuẩn của golang, cung cấp các hàm để định dạng và xuất nhập dữ liệu.

Chúng ta có thể nhập nhiều gói cùng lúc bằng cách sử dụng cặp ngoặc đơn.

Ví dụ:

import (

"fmt"

"math"

)

Trong ví dụ trên, chúng ta nhập hai gói **fmt** và **math** vào chương trình. Gói **math** là một gói trong thư viện chuẩn của golang, cung cấp các hàm toán học.

**Khai báo hàm**

Để định nghĩa một hàm trong golang, chúng ta sử dụng từ khóa **func**. Cấu trúc của một hàm như sau:

func name(parameters) (results) {

// body

}

Trong đó:

* **name** là tên của hàm.
* **parameters** là danh sách các tham số được truyền vào hàm. Mỗi tham số có tên và kiểu dữ liệu. Nếu có nhiều tham số cùng kiểu, chúng ta có thể chỉ viết kiểu một lần ở cuối.
* **results** là danh sách các kết quả được trả về từ hàm. Mỗi kết quả có tên và kiểu dữ liệu. Nếu chỉ có một kết quả, chúng ta có thể bỏ qua tên và chỉ viết kiểu.
* **body** là phần thân của hàm, chứa các câu lệnh để thực hiện công việc của hàm.

Ví dụ:

func add(x int, y int) int {

return x + y

}

Trong ví dụ trên, chúng ta định nghĩa một hàm có tên là **add**, nhận hai tham số kiểu int là x và y, và trả về một kết quả kiểu int là tổng của x và y.

Chúng ta có thể rút ngắn cách viết như sau:

func add(x, y int) int {

return x + y

}

**Khai báo biến**

Để khai báo một biến trong golang, chúng ta sử dụng từ khóa **var**. Cấu trúc của một biến như sau:

var name type = expression

Trong đó:

* **name** là tên của biến.
* **type** là kiểu dữ liệu của biến.
* **expression** là giá trị ban đầu của biến.

Nếu giá trị ban đầu được chỉ định rõ ràng, chúng ta có thể bỏ qua kiểu dữ liệu. Ví dụ:

var x = 10 // x is an int

var y = "Hello" // y is a string

Chúng ta có thể khai báo nhiều biến cùng lúc bằng cách sử dụng cặp ngoặc đơn. Ví dụ:

var (

a = 1

b = 2

c = 3

)

Chúng ta cũng có thể sử dụng toán tử **:=** để khai báo và gán giá trị cho biến ngắn gọn hơn.

Ví dụ:

x := 10 // same as var x = 10

y := "Hello" // same as var y = "Hello"

Lưu ý: Toán tử **:=** chỉ có thể sử dụng trong một hàm.

**Khai báo hằng**

Để khai báo một hằng trong golang, chúng ta sử dụng từ khóa **const**. Cấu trúc của một hằng như sau:

const name type = expression

Trong đó:

* **name** là tên của hằng.
* **type** là kiểu dữ liệu của hằng.
* **expression** là giá trị không đổi của hằng.

Nếu giá trị không đổi được chỉ định rõ ràng, chúng ta có thể bỏ qua kiểu dữ liệu.

Ví dụ:

const pi = 3.14 // pi is a float64

const hello = "Hello" // hello is a string

Chúng ta có thể khai báo nhiều hằng cùng lúc bằng cách sử dụng cặp ngoặc đơn.

Ví dụ:

const (

pi = 3.14

hello = "Hello"

)

**Kiểu dữ liệu**

Golang có nhiều kiểu dữ liệu cơ bản như sau:

* Kiểu số nguyên: int8, int16, int32, int64, uint8, uint16, uint32, uint64.
* Kiểu số thực: float32, float64.
* Kiểu số phức: complex64, complex128.
* Kiểu chuỗi: string.
* Kiểu luận lý: bool.

Ngoài ra, golang còn có các kiểu dữ liệu phức tạp như sau:

* Kiểu mảng: array.
* Kiểu slice: slice.
* Kiểu map: map.
* Kiểu struct: struct.
* Kiểu interface: interface.

Chú ý: Golang không có kiểu con trỏ (pointer).

**Toán tử số học**

Toán tử số học được sử dụng để thực hiện các phép tính số học như cộng, trừ, nhân, chia và chia lấy dư. Đây là danh sách các toán tử số học trong golang:

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| + | a + b | Cộng a và b |
| - | a - b | Trừ a và b |
| \* | a \* b | Nhân a và b |
| / | a / b | Chia a cho b |
| % | a % b | Chia lấy dư a cho b |

Ví dụ:

package main

import "fmt"

func main() {

a := 10

b := 3

fmt.Println(a + b) // 13

fmt.Println(a - b) // 7

fmt.Println(a \* b) // 30

fmt.Println(a / b) // 3

fmt.Println(a % b) // 1

}

**Toán tử gán**

Toán tử gán được sử dụng để gán giá trị cho một biến. Đây là danh sách các toán tử gán trong golang:

| **Toán tử** | **Ví dụ** | **Tương đương** |
| --- | --- | --- |
| = | x = y | Gán y cho x |
| += | x += y | Gán x + y cho x |
| -= | x -= y | Gán x - y cho x |
| \*= | x \*= y | Gán x \* y cho x |
| /= | x /= y | Gán x / y cho x |
| %= | x %= y | Gán x % y cho x |

Ví dụ:

package main

import "fmt"

func main() {

x := 10

x += 5 // same as x = x + 5

fmt.Println(x) // 15

x -= 3 // same as x = x - 3

fmt.Println(x) // 12

x \*= 2 // same as x = x \* 2

fmt.Println(x) // 24

x /= 4 // same as x = x / 4

fmt.Println(x) // 6

x %= 3 // same as x = x % 3

fmt.Println(x) // 0

}

**Toán tử so sánh**

Toán tử so sánh được sử dụng để so sánh hai giá trị và trả về một giá trị kiểu bool (true hoặc false). Đây là danh sách các toán tử so sánh trong golang:

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| == | a == b | Kiểm tra a có bằng b không |
| != | a != b | Kiểm tra a có khác b không |
| < | a < b | Kiểm tra a có nhỏ hơn b không |
| <= | a <= b | Kiểm tra a có nhỏ hơn hoặc bằng b không |
| > | a > b | Kiểm tra a có lớn hơn b không |
| >= | a >= b | Kiểm tra a có lớn hơn hoặc bằng b không |

Ví dụ:

package main

import "fmt"

func main() {

a := 10

b := 3

fmt.Println(a == b) // false

fmt.Println(a != b) // true

fmt.Println(a < b) // false

fmt.Println(a <= b) // false

fmt.Println(a > b) // true

fmt.Println(a >= b) // true

}

## Toán tử logic

Toán tử logic được sử dụng để kết hợp hai giá trị kiểu bool và trả về một giá trị kiểu bool. Đây là danh sách các toán tử logic trong golang:

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| && | a && b | Kết quả là true nếu cả a và b đều là true |
| || | a || b | Kết quả là true nếu ít nhất một trong a hoặc b là true |
| ! | !a | Kết quả là true nếu a là false và ngược lại |

Ví dụ:

package main

import "fmt"

func main() {

a := true

b := false

fmt.Println(a && b) // false

fmt.Println(a || b) // true

fmt.Println(!a) // false

fmt.Println(!b) // true

}

**Toán tử bit**

Toán tử bit được sử dụng để thao tác trên các bit của một số nguyên. Đây là danh sách các toán tử bit trong golang:

| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| --- | --- | --- |
| & | a & b | Thực hiện phép AND bit trên a và b |
| | | a | b | Thực hiện phép OR bit trên a và b |
| ^ | a ^ b | Thực hiện phép XOR bit trên a và b |
| &^ | a &^ b | Thực hiện phép AND NOT bit trên a và b |
| << | a << n | Dịch trái a n bit |
| >> | a >> n | Dịch phải a n bit |

Ví dụ:

package main

import "fmt"

func main() {

a := 10 // 1010 in binary

b := 3 // 0011 in binary

fmt.Println(a & b) // 0010 in binary, 2 in decimal

fmt.Println(a | b) // 1011 in binary, 11 in decimal

fmt.Println(a ^ b) // 1001 in binary, 9 in decimal

fmt.Println(a &^ b) // 1000 in binary, 8 in decimal

fmt.Println(a << 2) // 101000 in binary, 40 in decimal

fmt.Println(a >> 2) // 0010 in binary, 2 in decimal

}

Đây là cách viết lại bài ở trên gắn ngọn hơn:

**CHƯƠNG 3: Golang ứng dụng làm web backend**

Golang là một ngôn ngữ lập trình tốt cho web backend. Web backend là phần xử lý dữ liệu và logic của web. Golang có web server sẵn có, nhiều web framework, kết nối dễ dàng với cơ sở dữ liệu, tạo ra web service hoặc API nhanh chóng và hiệu quả, và triển khai trên nhiều nền tảng.

Để làm một ví dụ về web backend bằng golang, chúng ta sẽ làm một web service để quản lý quán trà sửa.

## Bước 1: Thiết kế API

Đầu tiên, chúng ta cần thiết kế các endpoint cho API. Endpoint là đường dẫn để truy cập đến một chức năng của web service. API của chúng ta sẽ có các endpoint sau:

* /Rproduct GET: Lấy danh sách tất cả các sản phẩm, trả về dưới dạng JSON.
* /searchproduct POST: tìm kiếm sản phẩm từ dữ liệu yêu cầu gửi lên.
* /Cproduct POST: Thêm một sản phẩm mới từ dữ liệu yêu cầu gửi lên.
* /Uproduct POST: cập nhật thông tin sản phẩm từ dữ liệu gửi lên.
* /Dproduct POST: xóa sản phẩm theo id gửi lên.
* /signin POST: xác minh thông tin đăng nhập được gửi lên

## Bước 2: Tạo thư mục cho code

Tiếp theo, chúng ta cần tạo một thư mục cho code. Tạo một thư mục có tên là web-backend-gin. Tạo một module để quản lý các phụ thuộc. Ta có thể thực hiện các điều trên bằng cách chạy các lệnh sau trong terminal:

$ mkdir web-backend-gin

$ cd web-backend-gin

$ go mod init example.com/web-backend-gin

## Bước 3: Tạo dữ liệu

Trong bước này, chúng ta sẽ tạo ra một kiểu struct để biểu diễn một sản phẩm.

Mở trình soạn thảo và tạo file sanpham.go trong thư mục web-backend-gin. Nhập vào file main.go nội dung sau:

Trong đoạn code trên, chúng ta đã khai báo một kiểu struct Album với ba trường: ID, Title và Artist. Chúng ta cũng đã khởi tạo slice albums với ba phần tử là ba album nhạc khác nhau.

## Bước 4: Viết handler để trả về tất cả các album

Trong bước này, chúng ta sẽ viết một hàm để xử lý yêu cầu GET đến endpoint /albums. Hàm này sẽ trả về danh sách tất cả các album trong slice albums dưới dạng JSON.

Để viết hàm này, chúng ta sẽ sử dụng web framework Gin. Gin là một web framework nhanh và linh hoạt cho golang.

Thêm hai dòng sau vào đầu file main.go:

import (

"net/http"

"github.com/gin-gonic/gin"

)

Sao chép

Sau đó, viết hàm getAlbums như sau:

func getAlbums(c \*gin.Context) {

// Trả về mã HTTP 200 (OK) và danh sách các album dưới dạng JSON

c.JSON(http.StatusOK, albums)

}

Sao chép

Hàm getAlbums nhận vào một tham số kiểu \*gin.Context. Đây là kiểu được Gin sử dụng để lưu trữ thông tin về yêu cầu và phản hồi HTTP. Hàm này sử dụng phương thức JSON của \*gin.Context để gửi phản hồi HTTP với mã 200 (OK) và nội dung là slice albums được mã hóa thành JSON.

Để đăng ký hàm getAlbums làm handler cho yêu cầu GET đến endpoint /albums, chúng ta cần tạo một đối tượng kiểu \*gin.Engine. Đây là kiểu được Gin sử dụng để quản lý các route và middleware. Chúng ta có thể gọi phương thức GET của \*gin.Engine để đăng ký handler cho route /albums.

Thêm đoạn code sau vào cuối file main.go:

func main() {

// Tạo một đối tượng kiểu \*gin.Engine

router := gin.Default()

// Đăng ký handler getAlbums cho route /albums

router.GET("/albums", getAlbums)

// Khởi động web server ở cổng 8080

router.Run("localhost:8080")

}

Sao chép

Lưu file main.go và chạy lệnh sau để biên dịch và chạy chương trình:

$ go run .

Sao chép

Giờ bạn có thể kiểm tra API của bạn bằng công cụ curl. Mở terminal mới và chạy lệnh sau:

$ curl http://localhost:8080/albums

Sao chép

Bạn sẽ nhận được output như sau:

[

{

"ID": "1",

"Title": "Blue Train",

"Artist": "John Coltrane"

},

{

"ID": "2",

"Title": "Jeru",

"Artist": "Gerry Mulligan"

},

{

"ID": "3",

"Title": "Sarah Vaughan and Clifford Brown",

"Artist": "Sarah Vaughan"

}

]