

**KHOA CÔNG NGHỆ PHẦN MỀM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

∙∙∙∙∙∙🙞🙜∙∙∙∙∙∙





**ĐỒ ÁN MÔN HỌC:**

ĐỒ ÁN 1

**ĐỀ TÀI:** TÌM HIỂU VỀ GOLANG VÀ ỨNG DỤNG

***Giảng viên hướng dẫn:*** Nguyễn Tấn Toàn

***Lớp:***SE121.K21.PMCL

***Nhóm sinh viên thực hiện:***

1. Chung Nguyễn Trường Duy 17520388
2. Hà Thanh Huy 17520568

Nội dung

[**CHƯƠNG I:TÌM HIỂU VỀ GOLANG** 5](#_Toc45457609)

[1. Giới thiệu 5](#_Toc45457610)

[1.1. Giới thiệu chung 5](#_Toc45457611)

[1.2. Đặc điểm 5](#_Toc45457612)

[1.3. Lý do sử dụng 6](#_Toc45457613)

[2. Cách cài đặt 6](#_Toc45457614)

[2.1. Cài đặt trên Windows 6](#_Toc45457615)

[2.2. Cài đặt trên MacOSX 7](#_Toc45457616)

[2.3. Cài đặt trên Ubuntu 7](#_Toc45457617)

[3. Cấu trúc chương trình 8](#_Toc45457618)

[3.1. Go Package 8](#_Toc45457619)

[3.2. Import Package 8](#_Toc45457620)

[3.3. Go function 9](#_Toc45457621)

[4. Cú pháp 9](#_Toc45457622)

[4.1. khai báo biến 9](#_Toc45457623)

[4.2. Gán giá trị cho biến 10](#_Toc45457624)

[4.3. Khởi tạo biến 10](#_Toc45457625)

[4.4. Câu lệnh trong Go 11](#_Toc45457626)

[5. Kiểu dữ liệu 11](#_Toc45457627)

[5.1. Kiểu Dữ Liệu String trong Golang 11](#_Toc45457628)

[5.2. Kiểu Dữ Liệu Integer trong Golang 11](#_Toc45457629)

[5.3. Kiểu Dữ Liệu Unsigned Integer trong Golang 12](#_Toc45457630)

[5.4. Kiểu Dữ Liệu boolean trong Golang 12](#_Toc45457631)

[5.5. Kiểu Dữ Liệu Float trong Golang 12](#_Toc45457632)

[5.6. Kiểu Dữ Liệu byte trong Golang 12](#_Toc45457633)

[6. Cấu trúc rẽ nhánh 12](#_Toc45457634)

[6.1. Mệnh đề If 12](#_Toc45457635)

[6.2. Mệnh đề If rút gọn 13](#_Toc45457636)

[6.3. Mệnh đề if...else 13](#_Toc45457637)

[7. Switch 14](#_Toc45457638)

[7.1. Switch Statement 14](#_Toc45457639)

[7.2. Cú Pháp Câu Lệnh Switch 14](#_Toc45457640)

[7.3. Ví dụ về Câu Lệnh Switch 14](#_Toc45457641)

[8. Cấu trúc vòng lặp 16](#_Toc45457642)

[9. Function 18](#_Toc45457643)

[9.1. Định nghĩa function 18](#_Toc45457644)

[9.2. Định Nghĩa Function trong Go 18](#_Toc45457645)

[9.3. Tham số trong Function 18](#_Toc45457646)

[10. Golang nâng cao 19](#_Toc45457647)

[10.1. Array và Slice 19](#_Toc45457648)

[10.2. Defer 23](#_Toc45457649)

[10.3. Goroutine 24](#_Toc45457650)

[10.4. Chanel 25](#_Toc45457651)

[10.5. Mutex 27](#_Toc45457652)

[**CHƯƠNG II: ỨNG DỤNG** 29](#_Toc45457653)

[1. Hiện trạng 29](#_Toc45457654)

[1.1. Mục đích và nhu cầu thực tế của đề tài 29](#_Toc45457655)

[1.2. Hiện trạng tổ chức 29](#_Toc45457656)

[1.3. Hiện trạng nghiệp vụ 30](#_Toc45457657)

[1.4. Hiện trạng tin học 30](#_Toc45457658)

[1.5. Xác định yêu cầu hệ thống 31](#_Toc45457659)

[1.6. Sơ đồ nghiệp vụ 32](#_Toc45457660)

[2. Mô hình usecase 33](#_Toc45457661)

[2.1. Sơ đồ Usecase tổng quát 33](#_Toc45457662)

[2.2. Danh sách Actor 33](#_Toc45457663)

[2.4. Đặc tả Usecase 34](#_Toc45457664)

[3. Phân tích 47](#_Toc45457665)

[3.1. Sơ đồ lớp (mức phân tích) 47](#_Toc45457666)

[3.2. Danh sách các lớp đối tượng 48](#_Toc45457667)

[3.3. Mô tả chi tiết các đối tượng 48](#_Toc45457668)

[4. Thiết kế dữ liệu 55](#_Toc45457669)

[4.1. Sơ đồ logic 55](#_Toc45457670)

[4.2. Mô tả các kiểu dữ liệu trong sơ đồ logic 56](#_Toc45457671)

[5. Thiết kế kiến trúc. 61](#_Toc45457672)

[5.1. Mô hình kiến trúc hệ thống. 61](#_Toc45457673)

[5.2. Mô tả chi tiết thành phần trong hệ thống. 61](#_Toc45457674)

[6. Thiết kế giao diện 63](#_Toc45457675)

[6.1. Danh sách màn hình. 63](#_Toc45457676)

[6.2. Mô tả chi tiết các màn hình. 64](#_Toc45457677)

[7. Kết luận 73](#_Toc45457678)

[7.1. Kết quả đạt được 73](#_Toc45457685)

[7.2. Hạn chế và khó khăn 73](#_Toc45457686)

[7.3. Hướng phát triển 73](#_Toc45457687)

[8. Tài liệu tham khảo 74](#_Toc45457688)

**BẢNG PHÂN CÔNG**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ tên** | **Công việc** |
| 17520388 | Chung Nguyễn Trường Duy | FrontEnd  Code: Room,Customer, Service  Test :Service,Customer,Block, DefaultService |
| 17520568 | Hà Thanh Huy | Code:Bill, Contract, Auth, Redis, UploadFile, Block, User  Test: Contract, Room, Auth, Bill, UploadFile, User,Unit |

# **CHƯƠNG I:TÌM HIỂU VỀ GOLANG**

## Giới thiệu

### Giới thiệu chung

Ngôn ngữ Go hay còn gọi là Golang là một ngôn ngữ lập trình được phát triển tại Google vào năm 2007 bởi Robert Griesemer, Rob Pike và Ken Thompson.

Golang thuộc kiểu ngôn ngữ lập trình biên soạn compiled language, điều này có nghĩa tập tin chứa [mã nguồn](https://www.codehub.vn/Lap-Trinh-Co-Ban-Danh-Cho-Nguoi-Moi-Bat-Dau/Ma-Nguon-va-Ma-May) (source code) viết bằng Golang không thể chạy được trực tiếp bởi máy tính. Thay vào đó chúng ta cần biên soạn (compile) tập tin này sử dụng [phần mềm biên soạn mã nguồn](https://www.codehub.vn/Lap-Trinh-Co-Ban-Danh-Cho-Nguoi-Moi-Bat-Dau/Thuat-Ngu-Co-Ban) (hay compiler) để tạo ra một tập tin khác chứa mã máy. Sau đó máy tính mới có thể chạy tập tin đã được biên soạn này.

Golang compilder là phần mềm hoàn toàn miễn phí.

### Đặc điểm

Các đặc điểm nổi bật của ngôn ngữ Golang có thể liệt kê ra gồm có:

* Go là ngôn ngữ lập trình biên soạn (compiled language). Phần mềm viết bởi Go cần được biên soạn sử dụng Go compile trước khi có thể chạy được.
* Go có cú pháp tương tự như C.
* Go là một ngôn ngữ lập trình kiểu tĩnh (hay *static type*), các biến hoặc dữ liệu trong Golang sẽ không thay đổi trong suốt chương trình.
* Golang hỗ trợ xử lý đa luồng multi-threading.
* Golang hỗ trợ tính năng quản lý bộ nhớ mạnh mẽ mà C không có.
* Golang sở hữu một bộ thư viện phong phú mà lập trình viên có thể tận dụng.

Ngôn ngữ lập trình Go đã được ra mắt cộng đồng vào tháng 11 năm 2009 và được sử dụng trong một số hệ thống phần mềm của gã khổng lồ công nghệ Google. Ngoài ra Golang cũng là ngôn ngữ được sử dụng để viết phần mềm nổi tiếng Docker.

### Lý do sử dụng

Tại sao bạn lại phải dùng đến Golang trong khi hàng tá các ngôn ngữ lập trình phía server khác như Python, Ruby, NodeJS... cũng có thể làm được những điều tương tự?

Sau đây là một số ưu điểm tôi nhận ra sau một thời gian dùng Golang:

* Concurrency là một phần có sẵn của ngôn ngữ lập trình này, do đó viết một chương trình multithread sẽ giống như một miếng bánh gato. Nó được lưu trữ bởi goroutines và channels- những khái niệm này sẽ được nhắc tới ở các phần sau.
* Golang là một ngôn ngữ biên dịch, mã nguồn sẽ được biên dịch sang mã nhị phân (binary), đây là phần còn thiếu trong Javascript - NodeJs.
* Các đặc điểm của ngôn ngữ này vô cùng đơn giản, toàn bộ các đặc điểm có thể tóm gọn lại trong một vài trang giấy thậm chí bạn có thể viết riêng cho mình cả một chương trình biên dịch.
* Bộ biên dịch của Go hỗ trợ liên kết tĩnh (static linking), tất cả code có thể được kết nối tĩnh thông qua một đoạn mã nhị phân và có thể triển khai trên server đám mây dễ dàng mà không cần lo lắng đến các dependency.

## Cách cài đặt

### Cài đặt trên Windows

Trên Windows sau khi tải tập tin cài đặt Go dành cho Windows về máy bạn chỉ cần click đúp vào tập tin và tiến hành cài đặt Go theo hướng dẫn.

Theo mặc định máy tính sẽ cài đặt Go trong thư mục c:\Go, thư mục này còn được gọi là thư muc $GO\_ROOT. Ngoài ra, trình cài đặt cũng sẽ tự động thêm thư mục c:\Go\bin vào [biến môi trường](https://www.codehub.vn/Cai-dat-bien-moi-truong-tren-Windows) trên máy. Thư mục này còn được gọi là thư mục $GO\_PATH.

Sau khi kết thúc việc cài đặt bạn cần khởi động lại máy tính để các thay đổi có hiệu lực.

### Cài đặt trên MacOSX

Tương tự như Windows thì bạn cũng có thể cài đặt Golang sử dụng tập tin cài đặt với phần mở rộng.pkg dành cho các máy tính chạy hệ điều hành Mac OSX.

Theo mặc định máy tính sẽ cài đặt Go trong thư mục /usr/local/go. Thư mục này còn được gọi là $GO\_ROOT.

$ echo $GO\_ROOT

Trình cài đặt cũng sẽ tự động thêm thư mục /usr/local/go/bin vào biến môi trường trên máy. Thư mục này còn được gọi là $GO\_PATH.

$ echo $GO\_PATH

Bạn cần tắt các cửa sổ dòng lệnh đang mở và sau đó mở lại để các thay đổi trên có hiệu lực.

### Cài đặt trên Ubuntu

Thông thường trước khi cài đặt một phần mềm trên Ubuntu thì bước đầu tiên chúng ta cần làm đó là cập nhật các package index và đồng thời các gói package đang được cài đặt trên hệ thống. Mở chương trình [cửa sổ dòng lệnh termial](https://www.codehub.vn/Command-line-co-ban-cho-nguoi-moi-bat-dau) trên Ubuntu và chạy câu lệnh dưới đây:

$ sudo apt-get update

$ sudo apt-get -y upgrade

Sau khi kết thúc việc cập nhật chúng ta có thể bắt đầu tiến hành cài đặt Go:

$ sudo apt-get install golang-go

Câu lệnh trên sẽ cài đặt phiên bản Go đang được phân phối trên package repository của Ubuntu. Để kiểm tra lại phiên bản Go mới được cài đặt bạn chạy câu lệnh:

$ go version

Theo mặc định máy tính sẽ cài đặt Go trong thư mục /usr/local/go. Thư mục này còn được gọi là $GO\_ROOT.

$ echo $GO\_ROOT

Trình cài đặt cũng sẽ tự động thêm thư mục /usr/local/go/bin vào biến môi trường trên máy. Thư mục này còn được gọi là $GO\_PATH.

$ echo $GO\_PATH

Bạn cần tắt các cửa sổ dòng lệnh đang mở và sau đó mở lại để các thay đổi trên có hiệu lực.

## Cấu trúc chương trình

Cấu trúc của một chương trình viết bằng [ngôn ngữ Golang](https://www.codehub.vn/Golang-Co-Ban/Go-Gioi-Thieu) gồm có 3 thành phần chính sau:

* Tên package
* Các package khác được import
* Các hàm (function)

### Go Package

Một tập tin Golang được bắt đầu bằng việc khai báo tên package. Package trong go được dùng để nhóm một hoặc nhiều tập tin có liên quan với nhau. Tên package sử dụng chữ in thường và được định nghĩa ở đầu chương trình.

package "main"

Với tập tin dùng để chạy chương trình thì tên package được quy định cần phải đặt là main đồng thời chúng ta cũng cần định nghĩa một hàm với tên cũng là main. Trình biên soạn Go compiler sẽ sử dụng hàm main này để chạy chương trình:

package "main"

func main() {}

Tập tin chạy chương trình này thường import vào các package khác để sử dụng.

### Import Package

Package có thể được import để sử dụng trong một package khác. Từ khoá import được sử dụng để import package:

import "fmt"

Ở ví dụ phía trên, package fmt (*formated* input/output) cung cấp các hàm (method) hữu ích trong việc xử lý dữ liệu input nhập vào và output đầu ra.

Bạn có thể import nhiều package cùng lúc trong cùng một câu lệnh duy nhất:

import(  
"test\_package\_1"  
"test\_package\_2"  
)

### Go function

Hàm để nhóm một hoặc nhiều dòng code khác nhau để thực hiện một tác vụ nhất định và chúng ta có thể thực hiện việc thực thi các đoạn code trong hàm này lặp đi lặp lại bằng việc gọi hàm.

Function trong Golang được định nghĩa sử dụng từ khóa func theo sau bởi tên function và kiểu dữ liệu của giá trị trả về của function:

func myFunction() {  
 fmt.Println("gọi từ myFunction()!")  
}

Hàm main là một hàm đặc biệt và sẽ được tự động chạy bởi Go mà không cần chúng ta gọi hàm này. Hàm main được định nghĩa duy nhất một lần và chúng ta không sử dụng hàm main cho mục đích khác.

package main

import "fmt"

func myFunction() {  
 fmt.Println("gọi từ myFunction()!")  
}

func main() {  
 myFunction()  
 fmt.Println("Hello, World!")  
}

Với các hàm chỉ được sử dụng trong package chứa tập tin mà hàm đó được khai báo (hay private function) thì tên hàm sẽ được bắt đầu bằng chữ in thường như myFunction ở trên. Ngược lại với các hàm được sử dụng bởi package khác thì tên của chúng sẽ được bắt đầu bởi chữ in hoa như hàm Println trong package fmt.

## Cú pháp

* 1. **khai báo biến**

Biến được khai báo sử dụng từ khoá var và theo sau là kiểu dữ liệu:

var variable\_name data\_type

Ví dụ dưới đây khai báo biến a với kiểu dữ liệu số nguyên (integer) và biến b với kiểu dữ liệu là chuỗi (string):

var a int

var b string

Các kiểu dữ liệu cung cấp sẵn trong Go và được sử dụng phổ biến hay sử dụng có thể kể đến bao gôm số nguyên int (integer), chuỗi string, số thập phân float...

* 1. **Gán giá trị cho biến**

Sau khi khởi tạo biến thì chúng ta có thể gán giá trị cho biến để sử dụng.

var a int  
a = 5

Golang là ngôn ngữ kiểu tính (static type) và khác với ngôn ngữ kiểu động (dynamic type) thì kiểu dữ liệu của giá trị của biến không được phép thay đổi sau khi khai báo:

var a int  
a = 5 // ok  
a = "5"// lỗi

Các biến được khởi tạo nhưng chưa được gán giá trị thì Golang sẽ tự động gán giá trị zero-value cho chúng. Zero-value là gì thì tuỳ thuộc vào kiểu dữ liệu của biến đó, ví dụ với kiểu dữ liệu số nguyên int thì giá trị này là 0 với chuỗi string thì giá trị này là chuỗi trống...

var a int  
fmt.Println(a)

* 1. **Khởi tạo biến**

Việc khởi tạo biến và gán giá trị cho biến có thể được thực hiện trong cùng một câu lệnh duy nhất thay vì phải tách ra làm hai câu lệnh như trên:

var a = 5

Bạn cũng có thể gom việc khai báo nhiều biến biến hoặc khai báo và khởi tạo biến trong cùng một câu lệnh:

var a, b, c int = 1, 2 3;  
var d, e, f string;

* 1. **Câu lệnh trong Go**

Một câu lệnh hay statement trong golang được sử dụng để thực hiện một tác vụ nào đó như khai báo biến, thực hiện tính tổng hai biến hoặc hai số…

Ví dụ một câu lệnh của Golang như sau:

var a, b, c = 3 int

Câu lệnh trong Golang được viết trong một dòng và chúng ta không cần sử dụng dấu ; để kết thúc câu lệnh.

Các câu lệnh trong chương trình sẽ hình thành luồng chạy cũng như kiểm soát quá trình chạy của chương trình:

package main  
import "fmt"

func main() {  
 var a int;  
 a = 3; if a >3 {  
 fmt.Println("a lớn hơn 3");  
 } else {  
 fmt.Println("a nhỏ hơn 3")  
 }  
}

## Kiểu dữ liệu

### ****Kiểu Dữ Liệu String trong Golang****

Các chuỗi trong Golang bao gồm các ký tự được đặt trong dấu ngoặc kép "":

var a string  
a = "Abcdef"

Hoặc:

**var** a **string** = "Abcdef"

### ****Kiểu Dữ Liệu Integer trong Golang****

Kiểu số nguyên bao gồm các số 0 số nguyên âm và nguyên dương:

var a, b int  
a = 100  
b = -100

### ****Kiểu Dữ Liệu Unsigned Integer trong Golang****

Unsigned Integer bao gồm các số nguyên không âm (hay chính là số tự nhiên):

var a uint  
a = 100 // OK  
a = -100 // Lỗi

### ****Kiểu Dữ Liệu boolean trong Golang****

Kiểu boolean chỉ gồm 2 giá trị là True và False:

var a, b bool  
a = True  
b = 0 // Tương tự b = False

### ****Kiểu Dữ Liệu Float trong Golang****

Kiểu Float dùng để biểu diễn các số thực, chúng ta sử dụng float64 với hệ điều hành 64 bit và float32 với hệ điều hành 32 bit:

var a float64  
a = 1,234

### ****Kiểu Dữ Liệu byte trong Golang****

Kiểu dữ liệu byte tương đương với các số nguyên cần tới tối đa 8 bit bộ nhớ để lưu trữ.

var a byte  
a = 2

Kiểu này còn tương đương với int8

## Cấu trúc rẽ nhánh

* 1. **Mệnh đề If**

Câu lệnh if hay if statement trong Golang có cú pháp như sau:

if expression {

// code...

}

Code đặt bên trong câu lệnh if sẽ được chạy nếu và chỉ nếu biểu thức expression trả về giá tị là True.

Ví dụ:

var x int = 2 // hoặc gọn hơn có thể viết: x := 2

if x < 0 {

fmt.Println("x nhỏ hơn không)

}

Tương tự như vòng lặp for thì câu lệnh if hay if statement trong Golang không đòi hỏi chúng ta phải sử dụng cặp dấu ngoặc () xung quanh biển thức điều kiện tuy nhiên đòi hỏi bạn phải sử dụng cặp dấu {}.

### Mệnh đề If rút gọn

Short statement là câu lệnh được chạy ngay cả trước khi biểu thức điều kiện expression được chạy:

max := 10

if value := 3; v < max {

return value

} else {

return max;

}

Chúng ta thường sử dụng short statement trong câu lệnh If để thực hiện khai báo, khởi tạo giá trị cho một hoặc thâm chí là nhiều biến:

if a, b := 3, 5; a - b < 0 {

fmt.Println("a < b")

}

Tương tự như trong [vòng lặp For](https://www.codehub.vn/Golang-Co-Ban/Go-For-Loop) thì các biến được khởi tạo trong short statement của câu lệnh if sẽ chỉ có thể được sử dụng bên trong phạm vi của câu lệnh này (bao gồm cả phần code đặt bên trong if và else mà chúng ta sẽ tìm hiểu ngay dưới đây).

### Mệnh đề if...else

Câu lệnh if...else có cấu trúc như sau:

if expression\_1 {

// code...

} else expression\_2 {

// code...

}

Ví dụ:

if a, b := 3, 5; a - b < 0 {

fmt.Printf("%a lớn hơn %b", a, b)

} else {

fmt.Printf("%a nhỏ hơn %b", a, b)

}

Phần code bên trong if và else hoàn toàn có thể truy cặp giá trị của biến a và b.

## Switch

* 1. **Switch Statement**

Câu lệnh switch hay switch statement là một loại cấu trúc điều khiển giúp lập trình viên có thể dễ dàng kiểm tra biểu thức điều kiện với nhiều giá trị cho trước thay vì phải dùng else và elseif nhiều lần.

### Cú Pháp Câu Lệnh Switch

Cú pháp của câu lệnh switch như sau:

switch expression {

case value\_1:

// code

case value\_1:

// code

default:

// code

}

### Ví dụ về Câu Lệnh Switch

Ví dụ dưới đây sử dụng câu lệnh switch để so sánh giá trị của biến i với các giá trị khác nhau trong từng case (trường hợp) và nếu với mỗi case sẽ in ra các thông điệp khác nhau tương ứng với giá trị của biến i:

var i int = 2

switch i {

case 1:

fmt.Println("one")

case 2:

fmt.Println("two")

case 3:

fmt.Println("three")

}

Khi giá trị của biến i đúng với một case nào đó (trường hợp trên là case 2) thì Golang sẽ dừng việc chạy các case tiếp theo sau.

**Lưu ý:** Khác với các ngôn ngữ khá thì trong Golang chúng ta không cần thiết phải đặt câu lệnh break ở cuối mỗi case. Việc break câu lệnh switch sẽ được tự động xử lý bởi Golang.

Một ví dụ khác như trong đoạn code Go dưới đây cũng sử dụng câu lệnh switch để hiển thị ra tên hệ điều hành đang chạy chương trình bằng cách kiểm tra giá trị của biến os với các giá trị khác nhau như "linux", "darwin" (Mac OS X) trong từng case:

package main

import (

"fmt"

"runtime"

)

func main() {

fmt.Print("Go runs on ")

switch os := runtime.GOOS; os {

case "darwin":

fmt.Println("OS X.")

case "linux":

fmt.Println("Linux.")

default:

// freebsd, openbsd,

// plan9, windows...

fmt.Printf("%s.", os)

}

}

Trường hợp giá trị của biến os không rơi vào case nào trong 2 case được so sánh thì case default được sử dụng.

**Lưu ý:** Việc sử dụng case default là tùy ý và không bắt buộc. Chúng ta cũng có thể không sử dụng case nào trong câu lệnh switch.

## Cấu trúc vòng lặp

Trong Golang chỉ có một loại vòng lặp duy nhất là vòng lặp for.

for i := 0; i < 10; i++ {  
 sum += i  
}

Cấu trúc vòng lặp for gồm 3 phần chính:

* Câu lệnh khởi tạo: sẽ được thực thi ngay trước khi bắt đầu vòng lặp đầu tiên và chạy duy nhất một lần này.
* Biểu thức điều kiện: được thực thi trước mỗi vòng lặp (và chạy sau câu lệnh khởi tạo đối với vòng lặp đầu tiên).
* Câu lệnh sau (post statement): được thực thi sau mỗi vòng lặp kết thúc.

Ngoài ra thì chúng ta còn có các câu lệnh bên trong vòng lặp for: Nếu biểu thức điều kiện ở trên trả về giá trịn True thì các câu lệnh bên trong vòng lặp for sẽ được thực thi. Ngược lại, vòng lặp sẽ kết thúc và chương trình sẽ được tiếp tục bằng việc chạy các câu lệnh phía sau (nếu có).

Chúng ta cùng tham khảo một ví dụ sau đây về cách sử dụng vòng lặp for trong Go:

package main  
import "fmt"

func main() {  
 var a int  
 a = 1  
 for i := 0; i < 10 ; i++ {  
 a = a+i  
 fmt.Println(a)  
 }  
}

Với vòng lặp for trên chúng ta bắt đầu bằng câu lệnh khởi tạo gán giá trị cho biến i đếm giá trị ban đầu:

i := 0

Tiếp theo là biểu thức điều kiện so sánh giá trị của biến i với 10:

i < 10

**Lưu ý 1**: Biến được khai báo trong câu lệnh khởi tạo sẽ chỉ có thể được sử dụng trong phạm vi của vòng lặp.

Điều này có nghĩa với vòng lặp for ở trên chúng ta chỉ có thể sử dụng giá trị của biến i trong phạm vi của vòng lặp.

**Lưu ý 2**: Khác với các ngôn ngữ lập trình khác như [PHP](https://www.codehub.vn/tutorials/PHP-Co-Ban) hay [JavaScript](https://www.codehub.vn/tutorials/JavaScript-Co-Ban), trong Golang chúng ta không sử dụng cặp dấu ngoặc ( ) để bao xung quanh 3 thànhh phần của câu lệnh for.

Tương tự với PHP và JavaScript thì trong vòng lặp for của Golang thì các câu lệnh được thực thi trong mỗi vòng lặp bắt buộc phải được bao bọc quyanh cặp dấu ngoặc { }.

Ngoài ra **câu lệnh khởi tạo** và/hoặc **câu lệnh sau**có thể không cần thiết và có thể được bỏ trống trong vòng lặp for của Go:

sum := 1

for ; sum < 1000; {  
 sum += sum  
}

Trong đoạn code phía trên đây, chúng ta sử dụng giá trị của biến sum trong biểu thức điều kiện và biến này đã được khởi tạo ở ngoài vòng lặp for do đó chúng ta sẽ để trống biểu thức khởi tạo.  
Ngoài ra, ở vòng lặp trên chúng ta cũng bỏ qua câu lệnh sau (post statement).

Thêm một điều nữa rằng vì Golang không cung cấp cho chúng ta vòng lặp while tuy nhiên for có thể được sử dụng giống như while:

sum := 1  
for sum < 1000 {  
 sum += sum  
}

## Function

### Định nghĩa function

Một function (hay hàm) bao gồm một hay nhiều câu lệnh khác nhau để thực thiện một tác vụ nào đó và có thể được sử dụng lặp lại nhiều lần.

Function cần được định nghĩa trước khi sử dụng:

func hello\_function string {

return "Xin chào!"

}

Để sử dụng hàm chúng ta thực hiện việc gọi hàm, hàm được gọi đơn giản bằng cách thêm dấu () sau tên hàm và các giá trị cho tham số nếu có (chúng ta sẽ tìm hiểu về tham số của hàm ở các phần tiếp theo):

hello\_function()

### Định Nghĩa Function trong Go

Function trong Golang được định nghĩa sử dụng từ khóa func theo sau bởi tên function và kiểu dữ liệu của giá trị trả về của function:

func hello\_function string {

return "Xin chào!"

}

Ví dụ trên định nghĩa một hàm hello\_function với kiểu dữ liệu của giá trị trả về là string.

Trường hợp hàm không trả về giá trị thì chúng ta có thể bỏ qua khai báo kiểu giá trị trả về của hàm. Ví dụ:

func hello\_function {

fmt.Println("Xin chào!")

}

### Tham số trong Function

Parameter cho phép chúng ta đưa vào hàm các biến sẽ được sử dụng trong hàm. Khi khai báo parameter chúng ta cũng cần khai báo kiểu cho chúng:

func hello\_function (string name) {

fmt.Println("Xin chào! ", name)

}

hello\_function("Golang")

Ở trên chúng ta định nghĩa hàm hello\_function với một parameter tên là name và với kiểu dữ liệu là string. Và do đó khi gọi hàm này thì bạn sẽ cần truyền vào giá trị cho parameter này một string:

hello\_function("Golang")

Một hàm có thể không có hoặc có một hoặc nhiều parameter:

func sum(value1 int, value2 int) int {

return value1 + value2

}

Trong trường hợp các parameter có cùng kiểu dữ liệu bạn có thể gom lại như sau:

func sum(value1, value2 int) int {

return value1 + value2

}

Cú pháp như trên tương tự với cách khai báo nhiều biến với cùng một kiểu dữ liệu.

**Lưu ý:** Hàm main là một hàm đặc biệt và sẽ được tự động chạy bởi Go mà không cần chúng ta gọi hàm này. Hàm main được định nghĩa duy nhất một lần và chúng ta không sử dụng hàm main cho mục đích khác.

## Golang nâng cao

### Array và Slice

#### **Arrays**

[Array](https://tour.golang.org/moretypes/6) (mảng) trong Go tương tự các ngôn ngữ khác, tuy nhiên nó có **kích thước cố định** (fixed size) và các phần tử bên trong phải **cùng loại dữ liệu**. Ví dụ:

*// Khởi tạo một mảng gồm 2 string*

var a [2]string

*// Gán giá trị cho các phần tử trong mảng*

a[0] = "Hello"

a[1] = "World"

*// In kết quả ra console*

fmt.Println(a[0], a[1])

fmt.Println(a)

*// Khởi tạo một mảng gồm 6 số int và gán luôn giá trị cho nó*

primes := [6]int{2, 3, 5, 7, 11, 13}

fmt.Println(primes)

*// Khởi tạo mảng nhưng không ghi rõ kích thước (thay bằng dấu ba chấm),*

*// trình biên dịch sẽ tự hiểu dựa vào số phần tử đã khai báo*

numbers := [...]int{12, 78, 50}

fmt.Println(numbers)

Không giống đa số các ngôn ngữ khác, Array trong Go không phải là dạng tham chiếu (reference types) mà là dạng tham trị (value types). Khi gán giá trị nó cho một biến mới thì nó sẽ tạo ra một bản copy của Array cũ, và mọi thay đổi ở Array mới không ảnh hưởng gì đến Array cũ:

a := [...]int{1, 2, 3, 4, 5}

b := a *// b là một array mới có giá trị giống a*

b[0] = 9 *// Thay đổi giá trị một phần tử của b*

fmt.Println("a is ", a) *// In ra 1 2 3 4 5*

fmt.Println("b is ", b) *// In ra 9 2 3 4 5*

#### **Slices**

Slice là một tham chiếu đến Array, nó mô tả một phần (hoặc toàn bộ) Array. Nó có kích thước động nên thường được sử dụng nhiều hơn Array.

Slice có thể tạo ra từ một Array bằng cách cung cấp 2 chỉ số (low và high) xác định vị trí phần tử trong Array. Ví dụ:

// Khởi tạo Array primes

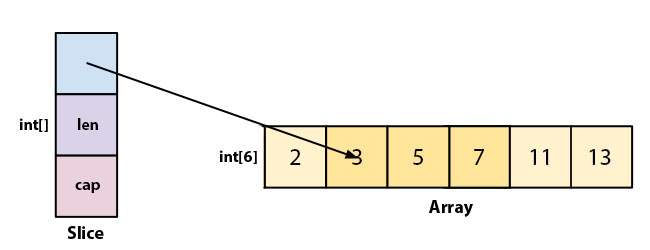
primes := [6]int{2, 3, 5, 7, 11, 13}

// Khởi tạo Slice s bằng cách cắt từ phần tử ở vị trí 1 (low) đến phần tử ở vị trí 3 (high - 1) của Array primes

var s []int = primes[1:4]

// In ra giá trị của Slice s

fmt.Println(s) // Giá trị của s là [3, 5, 7]

[[](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/5jozo13v25_go-slice-array.jpg)](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/5jozo13v25_go-slice-array.jpg)

Một Slice sẽ có 2 thuộc tính là **length** (len) và **capacity** (cap). Length là số phần tử chứa trong Slice, còn capacity là số phần tử chứa trong Array mà Slice tham chiếu đến (bắt đầu tính từ phần tử đầu tiên của Slice). Để lấy ra length của Slice ta dùng hàm **len()**, còn để lấy ra capacity thì ta dùng hàm **cap()**. Ví dụ:

s := []int{2, 3, 5, 7, 11, 13}

s = s[0:0] // s = [], len(s) = 0, cap(s) = 6

s = s[0:4] // s = [2, 3, 5, 7], len(s) = 4, cap(s) = 6

s = s[2:4] // s = [5, 7], len(s) = 2, cap(s) = 4, cap được tính từ vị trí số 2 trở đi

s = s[0:4] // s = [5, 7, 11, 13], len(s) = 4, cap(s) = 4

Khi tạo Slice ta có thể bỏ qua các chỉ số low và high, khi đó Go sẽ tự sử dụng giá trị mặc định: **0** cho low và **length của Slice** cho high. Ví dụ:

s := []int{2, 3, 5, 7, 11, 13}

s = s[:0] // s = [0:0]

s = s[:4] // s = [0:4]

s = s[2:] // s = [2:len(s)] => s = [2:4]

s = s[:4] // s = [0:4]

Ngoài việc tạo Slice như trên, chúng ta có thể tạo theo các cách sau:

* Khai báo như một mảng nhưng không chỉ ra kích thước mảng:

q := []int{2, 3, 5, 7, 11, 13}

* Sử dụng hàm make() với công thức: func make([]T, len, cap) []T

a := make([]int, 5) // len(a)=5

b := make([]int, 0, 5) // len(b)=0, cap(b)=5

Slice có zero value là **nil** (length = 0 và capacity = 0), **nil** tương đương với giá trị **null** trong các ngôn ngữ lập trình khác.

Do Slice chỉ là tham chiếu đến Array, do đó thay đổi giá trị của Slice sẽ làm thay đổi giá trị của Array mà nó tham chiếu đến. Nếu có nhiều Slice cùng tham chiếu đến một Array thì khi thay đổi giá trị một Slice có thể làm thay đổi giá trị các Slice khác. Ví dụ:

numbers := [4]int{1, 2, 3, 4}

a := numbers[0:2] // a = [1, 2]

b := numbers[1:3] // b = [2, 3]

b[0] = 5 // Thay đổi giá trị phần tử đầu tiên của Slice b

fmt.Println(a, b) // a = [1, 5], b = [5, 3]

fmt.Println(numbers) // numbers = [1, 5, 3, 4]

#### **Append**

Để bổ sung thêm phần tử cho slice, ta dùng hàm **append()** với công thức sau:  
func append(s []T, vs ...T) []T

Hàm này sẽ trả về một slice có chứa toàn bộ các phần tử của slice ban đầu và các phần tử mới thêm vào. Trong trường hợp slice ban đầu có sức chứa nhỏ (Array mà nó tham chiếu đến có size nhỏ), một Array mới có kích thước lớn hơn sẽ được tạo ra và slice mới sẽ tham chiếu đến Array đó.

var s []int

*// Append có thể hoạt động với nil slice.*

s = append(s, 0) *// s = [0]*

*// Append thêm một phần tử vào slice.*

s = append(s, 1) *// s = [0, 1]*

*// Append thêm nhiều phần tử vào slice.*

s = append(s, 2, 3, 4) *// s = [0, 1, 2, 3, 4]*

#### **Range**

**Range** là một hình thức của vòng lặp **for** dùng để duyệt qua một slice hoặc map (sẽ nhắc đến ở phần sau). Mỗi một vòng lặp sẽ trả về 2 giá trị: Giá trị đầu tiên là chỉ số (vị trí) của phần tử, và giá trị thứ hai là bản sao của phần tử đó (cùng giá trị). Ví dụ:

var pow = []int{1, 2, 4, 8, 16, 32, 64, 128}

for i, v := range pow {

fmt.Printf("i = %d, v = %d \n", i, v)

}

Trong trường hợp khi lặp chỉ sử dụng 1 trong 2 giá trị trả về thì ta sẽ bỏ qua giá trị còn lại bằng cách thay tên biến bằng ký tự **gạch dưới** (vì nếu không thì khi biên dịch sẽ báo lỗi biến được định nghĩa mà không sử dụng). Ví dụ:

var pow = []int{1, 2, 4, 8, 16, 32, 64, 128}

for \_, v := range pow {

fmt.Printf("v = %d \n", v)

}

### Defer

Trong Golang có một lệnh khá hay đó là defer, khi một câu lệnh được đặt sau từ khoá defer, thì nó sẽ được **thực thi khi hàm chứa nó kết thúc**. Cho dù bạn viết nó ở bất kì đâu.

**Ví dụ :**

func **Hello**() {

defer fmt.Println("World")

fmt.Println("Hello")

}

func **main**() {

fmt.Println("Say")

Hello()

}

Output của đoạn code trên sẽ là:

**Say**

Hello

World

### Goroutine

Một trong những điểm nổi bật của Golang là goroutine, nó có gì nổi bật để khiến cho cocurrency trong Go trở nên dễ dàng như vậy

Cấu trúc:

* Một goroutine được khởi tạo với chỉ 2KB trong stack size
* Goroutines chứa trong ít stack size nhưng có thể tự grow khi cần, thông thường sẽ có hàng trăm nghìn goroutines bên trong chương trình của bạn. Điều này cho phép lập trình viên use **concurrency structure** bên trong chương trình mà không cần quan tâm có bị overhead hay không.
* Goroutines chính là **function executing concurrently** cùng lúc với các goroutines khác trên cùng địa chỉ space
* Cấu trúc code:

struct P

{

Lock;

G \*gfree; // freelist, moved from sched

G \*ghead; // runnable, moved from sched

G \*gtail;

MCache \*mcache; // moved from M

FixAlloc \*stackalloc; // moved from M

uint64 ncgocall;

GCStats gcstats;

// etc

...

};

P \*allp; // [GOMAXPROCS]

type schedt struct {

glock mutex

gfree \*g

Ngfree int32

}

### Chanel

#### **Channels là gì**

Channels có thể được coi là các đường ống sử dụng mà Goroutines giao tiếp. Tương tự như cách nước chảy từ đầu này sang đầu kia trong đường ống, dữ liệu có thể được gửi từ một đầu và nhận từ đầu kia bằng channels.

#### **Khai báo channel**

Mỗi channel có một loại liên kết với nó. Loại này là loại dữ liệu mà channel được phép vận chuyển. Không có loại khác được phép vận chuyển bằng cách sử dụng channel.

chan T là một channel loại T

Giá trị 0 của một channel là không. Channels nil không được sử dụng và do đó channel phải được xác định bằng cách sử dụng make tương tự như map và slice.

Ví dụ:

package main

import "fmt"

func main() {

var a chan int

if a == nil {

fmt.Println("channel a is nil, going to define it")

a = make(chan int)

fmt.Printf("Type of a is %T", a)

}

}

Channel a khai báo chưa có giá trị. Do đó, các câu lệnh bên trong điều kiện if được thực thi và channel được xác định. a trong chương trình trên là channel int. Chương trình này sẽ xuất ra,

channel a is nil, going to define it

Type of a is chan int

Như thường lệ, khai báo cũng là một cách hợp lệ và ngắn gọn để xác định channel.

a := make(chan int)

Dòng mã trên cũng xác định channel int a.

#### **Gửi và nhận từ channel**

Cú pháp gửi và nhận dữ liệu từ channel được đưa ra dưới đây,

data := <-a//đọc từ kênh a

a<-data //gửi từ kênh a

Hướng của mũi tên đối với channel chỉ định liệu dữ liệu được gửi hay nhận.

Trong dòng đầu tiên, mũi tên chỉ ra từ a và do đó chúng ta đang đọc từ channel a và lưu trữ giá trị vào biến data.

Trong dòng thứ hai, mũi tên chỉ về phía a và do đó chúng ta đang ghi vào channel a.

#### **Gửi và nhận được chặn theo mặc định**

Gửi và nhận đến một channel đang bị chặn theo mặc định. Điều đó có nghĩa là gì? Khi dữ liệu được gửi đến một channel, điều khiển sẽ bị chặn trong câu lệnh gửi cho đến khi một số Goroutine khác đọc từ channel đó. Tương tự khi dữ liệu được đọc từ một channel, việc đọc bị chặn cho đến khi một số Goroutine ghi dữ liệu vào channel đó.

Thuộc tính này của channels là thứ giúp Goroutines giao tiếp hiệu quả mà không cần sử dụng khóa rõ ràng hoặc các biến có điều kiện khá phổ biến trong các ngôn ngữ lập trình khác.

Ví dụ:

package main

import "fmt"

func main() {

var a chan int

if a == nil {

fmt.Println("channel a is nil, going to define it")

a = make(chan int)

fmt.Printf("Type of a is %T", a)

}

}

Kết quả:

channel a is nil, going to define it

Type of a is chan int

### Mutex

Mutex là hình thức ngắn gọn để loại trừ lẫn nhau. Mutex được sử dụng khi bạn không muốn một tài nguyên bị truy cập bởi nhiều [Routine](https://vngeeks.com/go-goroutine/) con trong cùng một thời điểm. Mutex có 2 phương thức Lock() và Unlock(). Mutex được đặt trong package sync. Nên chúng ta phải import package sync. Các câu lệnh phải được thực hiên riêng rẽ lẫn nhau có thể được đặt trong mutex.Lock() hoặc mutex.Unlock(), do đó tránh được điều kiện Race.

mutex.Lock()

x = x + 1

mutex.Unlock()

Trong đoạn code bên trên x = x + 1 sẽ chỉ được chạy bởi một Goroutine tại bất cứ thời điểm nào do đó ngăn chặn được điều kiện Race.

Nếu một Goroutine nắm giữ lock (khoá) rồi và nếu một Goroutine mới đang cố gắng lấy được lock, thì Goroutine mới này sẽ đợi cho tợi khi Mutex được mở (unlock).

Hãy cùng xem ví dụ sau đây đếm số lần vòng lặp chạy. Trong chương trình này chúng ta mong đợi routine chạy vòng lặp 10 lần và count lưu trữ tổng số. Chúng ta gọi routine 3 lần nên tổng của nó nên là 30. count là biến global.

Ví dụ:

package main

import "fmt"

import "time"

import "sync"

import "strconv"

import "math/rand"

// Khai báo một instance mutex

var mu sync.Mutex

//Khai báo biến count được truy cập bởi tất cả các routine

var count = 0

// Sao chép count vào temp, thực hiện một vài xử lý (tăng dần) và lưu lại vào count

// tạm dừng một khoảng ngẫu nhiên được thêm vào giữa lúc đọc và ghi count

func process(n int) {

//Vòng lặp tăng count 10 lần

for i := 0; i < 10; i++ {

time.Sleep(time.Duration(rand.Int31n(2)) \* time.Second)

//Bắt đầu khoá ở đây

mu.Lock()

temp := count

temp++

time.Sleep(time.Duration(rand.Int31n(2)) \* time.Second)

count = temp

// Mở khoá

mu.Unlock()

}

fmt.Println("Count after i="+strconv.Itoa(n)+" Count:", strconv.Itoa(count))

}

func main() {

//lặp gọi process() 3 lần

for i := 1; i < 4; i++ {

go process(i)

}

//Tạm dừng để đợi cho tất cả routine hoàn thành

time.Sleep(25 \* time.Second)

fmt.Println("Final Count:", count)

}

Kết quả:

Count after i=3 Count: 21

Count after i=2 Count: 28

Count after i=1 Count: 30

Final Count: 30

# **CHƯƠNG II: ỨNG DỤNG**

## Hiện trạng

* 1. **Mục đích và nhu cầu thực tế của đề tài**

Hiện nay, công nghệ đang phát triền với tốc độ vượt bậc và trở thành một phần thiết yếu trong cuộc sống. Nhờ vào công nghệ, chúng ta có thể tiết kiệm được nhiều thời gian và công sức cho việc quản lý.

Dựa vào hiện trạng cho thuê và nhu cầu ngày càng tăng, việc áp dụng công nghệ và phần mềm vào việc quản lý nhà trọ cũng được chú ý tới. Đó là lí do mà nhóm chọn đề tài này.

* 1. **Hiện trạng tổ chức**
* **Hiện trạng:**

Nhà trọ: Nhà trọ Thùy Nhơn

Địa chỉ: 01 Lê Thước,Thảo Điền,Q2,TP.HCM

* *Chủ nhà trọ:* Là người cho thuê nhà trọ đã được hơn 20 năm
* **Một số quy định:**
* Đối với người chủ nhà trọ phải tình trạng sức khỏe tốt
* Có hiểu biết về sử dụng thiết bị công nghệ(Điện thoại thông minh,laptop,…)
  1. **Hiện trạng nghiệp vụ** 
     1. **Cho thuê:**

*Người thực hiện: Chủ nhà trọ & Người thuê phòng*

* Người Thuê phải có giấy tờ tùy thân(CMND,Căn cước,….)
* Phòng có sẵn phải đạt yêu cầu của người muốn thuê phòng đặt ra
* Kiểm tra chất lượng phòng,tình trạng điện, nước, mạng,…
* Người thuê cho biết các yêu cầu thêm ngày trả tiền, chu kỳ,…
  + 1. **Rời phòng:**

*Người thực hiện: Người thuê phòng & Chủ nhà trọ*

* Khi người thuê có nguyện vọng chấm dứt hợp đồng và chuyển đi
* Chủ nhà trọ kiểm tra tình trạng phòng hiện tại và yêu cầu đền bù nếu có
* Người thuê phòng thực hiện dọn đồ trước ngày đã báo trước
  + 1. **Kiểm tra các dịch vụ và xuất hóa đơn**

*Người thực hiện: Chủ nhà thuốc*

* Kiểm tra đồng hồ điện, nước,… và thống kê số tiền
* Kiểm tra các dịch vụ mà các phòng đang sử dụng(giữ xe,wifi,…).
* Lập hóa đơn với các mục chi phí rõ ràng
* Gửi cho người thuê phòng
  + 1. **Thanh toán hóa đơn**

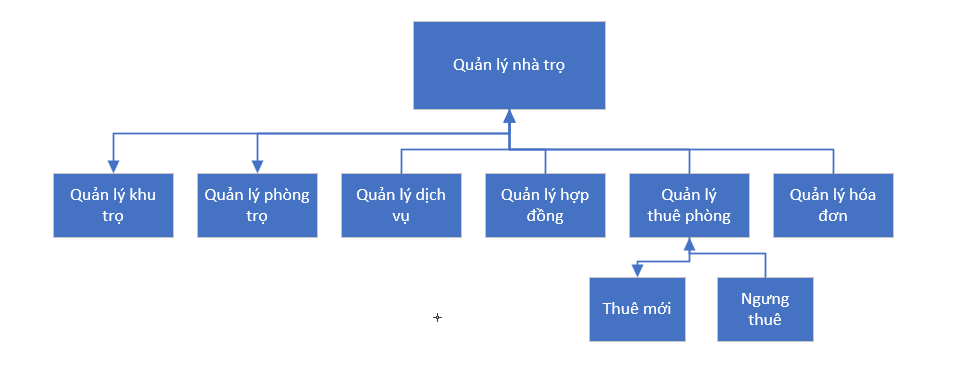
*Người thực hiện: Người thuê phòng*

* Kiểm tra các mục trên hóa đơn và báo lại nếu có sai sót
* Thanh toán cho chủ nhà trọ trước ngày chủ nhà đã đưa ra
  + 1. **Thống Kê và thanh toán**

*Người thực hiện: Chủ nhà trọ*

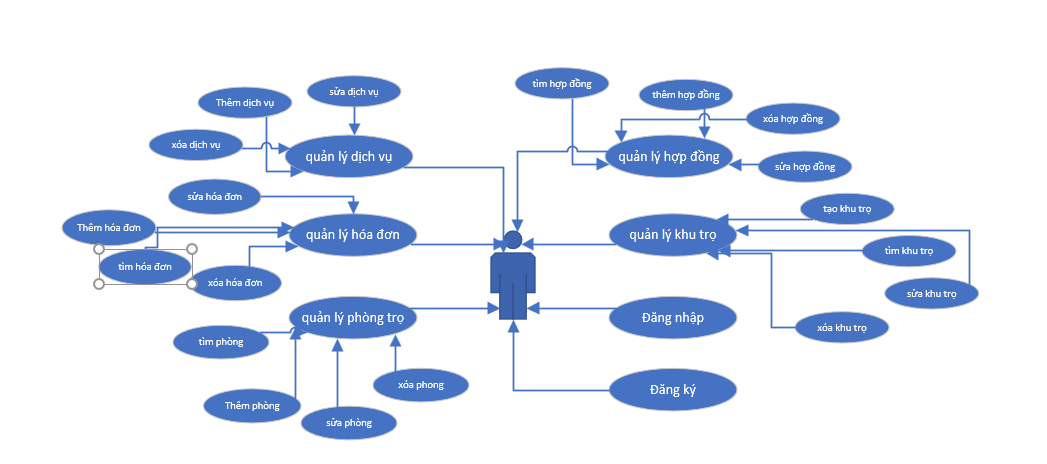
* Kiểm tra,bảo trì tình trạng các phòng trọ
* Thanh toán chi phí với nhà cung cấp dịch vụ(Điện, nước, internet,...)
  1. **Hiện trạng tin học**

* **Về phần cứng và phần mềm:**
* Nhà trọ có 1 máy tính.
* Cấu hình máy tầm trung.
* Có hỗ trợ truy cập Internet.
* Có mạng dây, kết nối ổn định.
* Được dùng thường xuyên để thực hiện thao tác cho thuê,thanh toán,...
* Chưa có phần mềm quản lý nhà trọ.
* **Về con người:**
* Trình độ tin học cơ bản.
* Sử dụng tốt các ứng dụng internet.
  1. **Xác định yêu cầu hệ thống**
     1. **Yêu cầu chức năng:**
* *Quản lý Cho thuê mới:* quản lý cho thuê phòng mới,quản lý các thông tin về thông tin người dùng
* *Quản lý phòng trọ:* chủ nhà trọ có thể thêm/xóa/sửa phòng trọ. Quản lý các thông tin phòng trọ về diện tích,dịch vụ,….
* *Quản lý khu trọ:* thêm/xóa/sửa khu trọ. Quản lý các thông tin tên khu trọ, địa chỉ,…
* *Quản lý hóa đơn:* quản lý hóa đơn thanh toán. Quản lý thông tin hóa đơn, chi tiết hóa đơn.
* *Quản lý dịch vụ:* quản lý dịch vụ đi kèm. Quản lý thông tin dịch vụ, chi tiết dịch vụ.
* *Quản lý hợp đồng:* quản lý tình trạng hợp đồng cho thuê,gia hạn,cắt hợp đồng,…
  + 1. **Yêu cầu phi chức năng:**
* Giao diện đẹp, đơn giản, dễ sử dụng.
* Hệ thống ổn định, ít chiếm tài nguyên của hệ thống.
* Các chức năng quản lý và tìm kiếm thao tác dễ dàng, thực hiện nhanh chóng.
  1. **Sơ đồ nghiệp vụ**



## Mô hình usecase

* 1. **Sơ đồ Usecase tổng quát**



* 1. **Danh sách Actor**

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa/Ghi chú |
| 1 | Chủ nhà trọ | Người có vị trí cao nhất, chịu trách nhiệm quản lý điều hành nhà trọ |

* 1. **Danh sách Usecase**

|  |  |  |
| --- | --- | --- |
| STT | Tên UseCase | Ý nghĩa/ Ghi chú |
| 1 | Đăng nhập |  |
| 2 | Đăng ký |  |
| 3 | Đăng xuất |  |
| 4 | Quản lý khu trọ | Thêm/Sửa thông tin khu trọ |
| 5 | Quản lý hóa đơn | Quản lý các hóa đơn thuê phòng,dịch vụ |
| 6 | Quản lý phòng trọ | Thêm/Xóa/Sửa thông tin phòng trọ |
| 7 | Quản lý hợp đồng | Thêm thông tin hợp đồng khách hàng |
| 8 | Quản lý dịch vụ | Kiểm tra/Sửa thông tin dịch vụ đi kèm |

* 1. **Đặc tả Usecase**
     1. **Usecase Đăng nhập**

|  |  |
| --- | --- |
| Tên chức năng | Đăng nhập |
| Tóm tắt | Chức năng đăng nhập vào hệ thống. |
| Dòng sự kiện chính | 1. Hệ thống hiển thị form đăng nhập.  2. Người dung nhập tên và tài khoản (cả 2 trường này đều bắt buộc nhập) và nhấn “Đăng nhập”.  3. Hệ thống kiểm tra thông tin đăng nhập  (Dòng sự kiện khác: Thông tin đăng nhập sai).  4. Hệ thống hiển thị form chính. |
| Dòng sự kiện khác | 1. Thông tin đăng nhập sai:  Hệ thống hiển thị thông báo tài khoản đăng nhập không hợp lệ. |

* + 1. **Usecase Đăng xuất**

|  |  |
| --- | --- |
| Tên chức năng | Đăng xuất |
| Tóm tắt | Chức năng đăng xuất ra hệ thống. |
| Dòng sự kiện chính | 1. Người dùng nhấn vào Đăng xuất  2. Hệ thống đăng xuất và trở về màn hình login |
| Dòng sự kiện khác |  |

* + 1. **Usecase Đăng ký**

|  |  |
| --- | --- |
| Tên chức năng | Đăng ký |
| Tóm tắt | Đăng ký tài khoản người dùng mới |
| Dòng sự kiện chính | 1. Người dùng nhấn tạo tài khoản mới  2. Người dùng nhập các thông tin cần thiết và xác nhận |
| Dòng sự kiện khác |  |

* + 1. **Usecase Quản lý Khu trọ**
       1. *Sửa khu trọ*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin khu trọ |
| Tóm tắt | Cập nhật thông tin của khu trọ  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một khu trọ nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm khu trọ*

|  |  |
| --- | --- |
| Tên chức năng | Thêm khu trọ |
| Tóm tắt | Thêm khu trọ mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một khu trọ nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa khu trọ*

|  |  |
| --- | --- |
| Tên chức năng | Xóa khu trọ |
| Tóm tắt | Xóa khu trọ  Hệ thống yêu cầu nhập mã khu trọ |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một khu trọ nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + 1. **Usecase Quản lý phòng trọ**
       1. *Sửa phòng trọ*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin phòng trọ |
| Tóm tắt | Cập nhật thông tin của phòng trọ  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục phòng trọ, chọn một phòng trọ nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm phòng trọ*

|  |  |
| --- | --- |
| Tên chức năng | Thêm phòng trọ |
| Tóm tắt | Thêm phòng trọ mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục phòng trọ, chọn một phòng trọ nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa khu trọ*

|  |  |
| --- | --- |
| Tên chức năng | Xóa phòng trọ |
| Tóm tắt | Xóa phòng trọ  Hệ thống yêu cầu nhập mã phòng trọ |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một phòng trọ nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + 1. **Usecase Quản lý** **hóa đơn**
       1. *Sửa hóa đơn*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin hóa đơn |
| Tóm tắt | Cập nhật thông tin của hóa đơn Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục hóa đơn, chọn một hóa đơn nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm hóa đơn*

|  |  |
| --- | --- |
| Tên chức năng | Thêm hóa đơn |
| Tóm tắt | Thêm hóa đơn mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục hóa đơn, chọn một hóa đơn nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa hóa đơn*

|  |  |
| --- | --- |
| Tên chức năng | Xóa hóa đơn |
| Tóm tắt | Xóa hóa đơn  Hệ thống yêu cầu nhập mã hóa đơn |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một hóa đơn nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + 1. **Usecase Quản lý hợp đồng**
       1. *Sửa hợp đồng*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin hợp đồng |
| Tóm tắt | Cập nhật thông tin của hợp đồng Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục hợp đồng, chọn một hợp đồng nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm hợp đồng*

|  |  |
| --- | --- |
| Tên chức năng | Thêm hợp đồng |
| Tóm tắt | Thêm hợp đồng mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục hợp đồng, chọn một hợp đồng nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa hợp đồng*

|  |  |
| --- | --- |
| Tên chức năng | Xóa hợp đồng |
| Tóm tắt | Xóa hợp đồng  Hệ thống yêu cầu nhập mã hợp đồng |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một hợp đồng nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + 1. **Usecase Quản lý dịch vụ**
       1. *Sửa dịch vụ*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin dịch vụ |
| Tóm tắt | Cập nhật thông tin của dịch vụ Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục dịch vụ, chọn một dịch vụ nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm dịch vụ*

|  |  |
| --- | --- |
| Tên chức năng | Thêm dịch vụ |
| Tóm tắt | Thêm dịch vụ mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục dịch vụ, chọn một dịch vụ nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa dịch vụ*

|  |  |
| --- | --- |
| Tên chức năng | Xóa dịch vụ |
| Tóm tắt | Xóa dịch vụ  Hệ thống yêu cầu nhập mã dịch vụ |
| Dòng sự kiện chính | 1. Vào mục khu trọ, chọn một dịch vụ nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + 1. **Usecase Quản lý khách hàng**
       1. *Sửa khách hàng*

|  |  |
| --- | --- |
| Tên chức năng | Cập nhật thông tin khách hàng |
| Tóm tắt | Cập nhật thông tin của khách hàng Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục khách hàng, chọn một khách hàng nhấn vào nút sửa.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Cập nhật”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Thêm khách hàng*

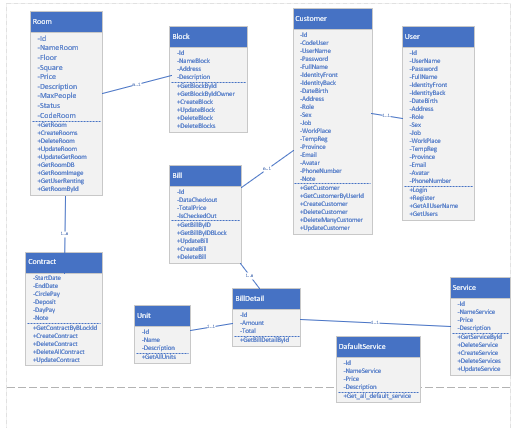
|  |  |
| --- | --- |
| Tên chức năng | Thêm khách hàng |
| Tóm tắt | Thêm khách hàng mới  Hệ thống chỉ cho phép cập nhật hầu hết các thông tin. |
| Dòng sự kiện chính | 1. Vào mục khách hàng, chọn một khách hàng nhấn vào nút Thêm.  2. Người dùng nhập các thông tin cần thiết (trong đó có một số thông tin bắt buộc) và nhấn “Thêm”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

* + - 1. *Xóa khách hàng*

|  |  |
| --- | --- |
| Tên chức năng | Xóa khách hàng |
| Tóm tắt | Xóa khách hàng  Hệ thống yêu cầu nhập mã dịch vụ |
| Dòng sự kiện chính | 1. Vào mục khách hàng, chọn một dịch vụ nhấn vào nút Xóa.  2. Người dùng nhấn “Xác nhận”.  3. Hệ thống kiểm tra thông tin, nếu các thông tin hợp lệ sẽ tiến hành bước tiếp theo.  (Dòng sự kiện khác: Thông tin không hợp lệ).  4. Hệ thống lưu dữ liệu và thông báo thành công.  (Dòng sự kiện khác: Không thể cập nhật xuống cơ sở dữ liệu) |
| Dòng sự kiện khác | 1. Thông tin không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập lại thông tin.  2. Không thể cập nhật cơ sở dữ liệu: Lỗi trong lúc cập nhật => Yêu cầu người dùng nhập lại thông tin, nếu vẫn bị lỗi nên liên hệ với nhóm phát triển. |

## Phân tích

* 1. **Sơ đồ lớp (mức phân tích)**



* 1. **Danh sách các lớp đối tượng**

|  |  |  |
| --- | --- | --- |
| STT | Tên lớp đối tượng | Ý nghĩa |
| 1 | Bill | Hóa đơn |
| 2 | BillDetail | Chi tiết hóa đơn |
| 3 | Contract | Hợp đồng |
| 4 | Unit | Đơn vi tính |
| 5 | DefaultService | Dịch vụ mặc định |
| 6 | Service | Dịch vụ |
| 7 | Block | Khu trọ |
| 8 | Room | Phòng trọ |
| 9 | User | Người dùng |
| 10 | Customer | Khác hàng |

* 1. **Mô tả chi tiết các đối tượng**
     1. **Bill**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã hóa đơn |
| 2 | DataCheckout | Time | public | Ngày thanh toán |
| 3 | TotalPrice | Float64 | public | Tổng tiền |
| 4 | IsCheckedOut | Bool | public | Đã thanh toán chưa |
| 5 | IdRoom | Integer | Public | Mã phòng |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetBillByID | public | Tìm hóa đơn bằng Id |
| 2 | GetBillByIDBLock | public | Tìm hóa đơn bằng Id khu trọ |
| 3 | UpdateBill | public | Cập nhật hóa đơn |
| 4 | CreateBill | public | Tạo hóa đơn |
| 5 | DeleteBill | public | Xóa hóa đơn |

* + 1. **BillDetail**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã chi tiết |
| 2 | Amount | Integer | public | Số lượng |
| 3 | Total | Float64 | public | Tổng tiền |
| 4 | IdBill | Integer | public | Mã hóa đơn |
| 5 | IdService | Integer | public | Mã dịch vụ |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetBillDetailById | public | Tìm chi tiết hóa đơn bằng mã hóa đơn |

* + 1. **Block**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | ID khu trọ |
| 2 | NameBlock | String | public | Tên khu trọ |
| 3 | Address | String | Public | Địa chỉ khu trọ |
| 4 | Description | String | Public | Mô tả khu trọ |
| 5 | IdOwner | Integer | Public | Chủ khu trọ |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetBlockById | public | Tìm khu trọ bằng id |
| 2 | GetBlockByIdOwner | public | Tìm khu trọ bằng id chủ khu trọ |
| 3 | CreateBlock | public | Tạo khu trọ |
| 4 | UpdateBlock | public | Cập nhật khu trọ |
| 5 | DeleteBlock | public | Xóa khu trọ |
| 6 | DeleteBlocks | public | Xóa nhiều khu trọ |

* + 1. **Contract**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | Integer | public | Mã hợp đồng |
| 2 | EndDate | Time | public | Ngày hết hạn |
| 3 | CirclePay | Integer | public | Chu kỳ trả tiền |
| 4 | Deposit | Integer | public | Đặt cọc |
| 5 | DayPay | Integer | public | Ngày trả |
| 6 | Note | String | public | Ghi chú |
| 7 | Startdate | Time | public | Ngày bắt đầu |
| 8 | IdRoom | Integer | public | Mã phòng |
| 9 | IdSlave | Integer | public | Mã người thuê |
| 10 | IdOwner | Integer | public | Mã chủ nhà |
| 11 | IdBlock | Integer | public | Mã khu trọ |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetContractByBLockId | public | Tìm hợp đồng bằng mã khu trọ |
| 2 | CreateContract | public | Tạo hợp đồng |
| 3 | DeleteContract | public | Xóa hợp đồng |
| 4 | DeleteAllContract | public | Xóa nhiều hợp đồng |
| 5 | UpdateContract | public | Cập nhật hợp đồng |

* + 1. **Customer**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã khách hàng |
| 2 | CodeUser | String | public | Code khách hàng |
| 3 | UserName | String | public | Tên tài khoản |
| 4 | Password | String | public | Mật khẩu |
| 5 | FullName | String | public | Tên khách hàng |
| 6 | IdentityFront | String | public | Mặt trước CMND |
| 7 | IdentityBack | String | public | Mặt sau CMND |
| 8 | DateBirth | Time | public | Ngày sinh |
| 9 | Address | String | public | Địa chỉ |
| 10 | Role | Integer | public | Quyền |
| 11 | Sex | String | public | Giới tính |
| 12 | Job | String | public | Công việc hiện tại |
| 13 | WorkPlace | String | public | Nơi làm việc |
| 14 | TempReg | Integer | public | Đăng ký tạm trú |
| 15 | Province | String | public | Tỉnh |
| 16 | Email | String | public | email |
| 17 | Avatar | String | public | ảnh đại diện |
| 18 | PhoneNumber | String | public | Số điện thoại |
| 19 | Note | String | public | Ghi chú |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetCustomer | public | Tìm khách hàng |
| 2 | GetCustomerByUserId | public | Tìm khách hàng bằng mã người dùng |
| 3 | CreateCustomer | public | Tạo khách hàng |
| 4 | DeleteCustomer | public | Xóa khách hàng |
| 5 | DeleteManyCustomer | public | Xóa nhiều khách hàng |
| 6 | UpdateCustomer | public | Cập nhật khách hàng |

* + 1. **Unit**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã đơn vị tính |
| 2 | Name | String | public | Tên đơn vị tính |
| 3 | Description | String | Public | Mô tả |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetAllUnits | public | Tìm tất cả đơn vị tính |

* + 1. **DefaultService**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | -Id | Integer | public | Mã dịch vụ |
| 2 | -NameService | String | public | Tên dịch vụ |
| 3 | -Price | Float64 | public | Giá |
| 4 | -Description | String | public | Mô tả |
| 5 | IdUnit | Integer | public | Mã đơn vị tính |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Get\_all\_default\_service | public | Tìm tất cả dịch vụ mặc định |

* + 1. **Room**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã phòng trọ |
| 2 | NameRoom | String | public | Tên phòng trọ |
| 3 | Floor | Integer | public | Số tầng |
| 4 | Square | Integer | public | Diện tích |
| 5 | Price | Float64 | public | Giá |
| 6 | Description | String | public | Mô tả |
| 7 | MaxPeople | Integer | public | Số người tối đa |
| 8 | Status | Integer | public | Trạng thái |
| 9 | CodeRoom | String | public | Mã phòng |
| 10 | IdBlock | Integer | public | Mã khu trọ |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetRoom | public | Tìm phòng trọ |
| 2 | CreateRooms | public | Tạo phòng trọ |
| 3 | DeleteRoom | public | Xóa phòng trọ |
| 4 | UpdateRoom | public | Sửa phòng trọ |
| 5 | UpdateGetRoom | public | Tìm phòng trọ |
| 6 | GetRoomDB | public | Tìm phòng theo trạng thái,khu trọ |
| 7 | GetRoomImage | public | Tìm ảnh phòng trọ |
| 8 | GetUserRenting | public | Tìm khách hàng đang thuê |
| 9 | GetRoomById | public | Tìm phòng trọ bằng mã phòng trọ |

* + 1. **Service**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã dịch vụ |
| 2 | NameService | String | public | Tên dịch vụ |
| 3 | Price | Float64 | public | Đơn giá |
| 4 | Description | String | public | Mô tả |
| 5 | IdUnit | Integer | public | Mã đơn vị tính |
| 6 | IdBlock | Integer | public | Mã khu trọ |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | GetServiceById | public | Tìm dịch vụ bằng mã dịch vụ |
| 2 | DeleteService | public | Xóa dịch vụ |
| 3 | CreateService | public | Tạo dịch vụ mới |
| 4 | DeleteServices | public | Xóa nhiều dịch vụ |
| 5 | UpdateService | public | Cập nhật dịch vụ |

* + 1. **User**
* Danh sách thuộc tính

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | Integer | public | Mã khách hàng |
| 3 | UserName | String | public | Tên tài khoản |
| 4 | Password | String | public | Mật khẩu |
| 5 | FullName | String | public | Tên khách hàng |
| 6 | IdentityFront | String | public | Mặt trước CMND |
| 7 | IdentityBack | String | public | Mặt sau CMND |
| 8 | DateBirth | Time | public | Ngày sinh |
| 9 | Address | String | public | Địa chỉ |
| 10 | Role | Integer | public | Quyền |
| 11 | Sex | String | public | Giới tính |
| 12 | Job | String | public | Công việc hiện tại |
| 13 | WorkPlace | String | public | Nơi làm việc |
| 14 | TempReg | Integer | public | Đăng ký tạm trú |
| 15 | Province | String | public | Tỉnh |
| 16 | Email | String | public | email |
| 17 | Avatar | String | public | ảnh đại diện |
| 18 | PhoneNumber | String | public | Số điện thoại |

* Danh sách phương thức

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên phương thức** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Login | public | Đăng nhập |
| 2 | Register | public | Đăng ký người dùng mới |
| 3 | GetAllUserName | public | Tìm tất cả tên người dùng |
| 4 | GetUsers | public | Tìm người dùng theo id |

## Thiết kế dữ liệu

### Sơ đồ logic

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên bảng dữ liệu** | **Diễn giải** |
| 1 | Bill | Danh sách các hóa đơn |
| 2 | BillDetail | Thông tin chi tiết hóa đơn |
| 3 | Contract | Danh sách các hợp đồng thuê phòng |
| 4 | Unit | Danh sách các đơn vị tính dịch vụ |
| 5 | DefaultService | Danh sách các dịch vụ mặc định |
| 6 | Service | Danh sách các dịch vụ |
| 7 | Block | Danh sách các khu trọ |
| 8 | Room | Danh sách các phòng trọ |
| 9 | User | Danh sách các người dùng |
| 10 | Customer | Danh sách các người thuê phòng |
| 11 | RoomImage | Danh sách ảnh phòng trọ |
| 12 | UserRoom | Danh sách người đang thuê theo phòng |

### Mô tả các kiểu dữ liệu trong sơ đồ logic

* + 1. **BillDetail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | Id | int | Not null | Khóa chính, tự động tăng |
| 2 | idBill | int |  | Mã hóa đơn |
| 3 | idService | int |  | Mã dịch vụ |
| 4 | Amount | int |  | Số lượng |
| 5 | totalPrice | decimal | 65 | Tổng tiền |

* + 1. **Bill**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | idRoom | int |  | Mã phòng |
| 3 | dateCheckOut | date |  | Ngày thah toán |
| 4 | totalPrice | decimal | 65 | Tổng giá |
| 5 | isCheckedOut | int |  | Đã thanh toán? |

* + 1. **DefaultService**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | nameService | varchar | 255 | Tên dịch vụ |
| 3 | price | decimal | 10 | Giá |
| 4 | description | varchar | 255 | Mô tả |
| 5 | idUnit | int |  | Mã đơn vị tính |

* + 1. **Service**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | nameService | varchar | 255 | Tên dịch vụ |
| 3 | price | decimal | 65 | Giá |
| 4 | idUnit | int |  | Mã đơn vị tính |
| 5 | description | varchar | 255 | Mô tả |
| 6 | idBlock | int |  | Mã khu trọ |

* + 1. **Block**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | nameBlock | varchar | 255 | Tên khu trọ |
| 3 | address | varchar | 255 | Địa chỉ |
| 4 | description | varchar | 255 | Mô tả |
| 5 | idOwner | int |  | Mã chủ nhà |

* + 1. **Contract**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | idRoom | int |  | Mã phòng |
| 3 | idOwner | int |  | Mã chủ nhà |
| 4 | idSlave | int |  | Mã người đại diện thuê nhà |
| 5 | startDate | date |  | Ngày bắt đầu |
| 6 | endDate | date |  | Ngày kết thúc |
| 7 | circlePay | int |  | Chu kỳ thanh toán |
| 8 | deposit | decimal | 65 | Tiền đặt cọc |
| 9 | dayPay | int |  | Ngày thanh toán |
| 10 | note | text |  | Ghi chú |
| 11 | idBlock | int |  | Mã khu trọ |

* + 1. **Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | codeUser | varchar | 50 | Code khách hàng |
| 3 | userName | varchar | 50 | Tên tài khoản |
| 4 | pass | varchar | 255 | Mật khẩu |
| 5 | fullName | varchar | 50 | Tên khách hàng |
| 6 | identifyFront | varchar | 300 | Mặt trước CMND |
| 7 | identifyBack | varchar | 300 | Mặt sau CMND |
| 8 | dateBirth | date |  | Ngày sinh |
| 9 | address | varchar | 255 | Địa chỉ |
| 10 | role | int |  | Quyền |
| 11 | sex | varchar | 10 | Giới tính |
| 12 | job | varchar | 100 | Công việc hiện tại |
| 13 | workPlace | varchar | 255 | Nơi làm việc |
| 14 | tempReg | int |  | Đăng ký tạm trú |
| 15 | province | varchar | 255 | Tỉnh |
| 16 | email | varchar | 255 | email |
| 17 | avatar | varchar | 300 | ảnh đại diện |
| 18 | phoneNumber | varchar | 20 | Số điện thoại |
| 19 | idOwner | int |  | Mã chủ nhà |
| 20 | note | text |  | Ghi chú |

* + 1. **RoomImage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | name | varchar | 255 | Tên hình |
| 3 | status | varchar | 50 | Trạng thái |
| 4 | url | varchar | 255 | Đường dẫn |
| 5 | codeRoom | varchar | 255 | Mã phòng |

* + 1. **Room**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | nameRoom | varchar | 50 | Tên phòng |
| 3 | floor | int |  | Số tầng |
| 4 | square | int | 250 | Diện tích |
| 5 | price | decimal | 65 | Giá |
| 6 | description | varchar | 255 | Mô tả |
| 7 | idBlock | int |  | Mã khu trọ |
| 8 | maxPeople | int |  | Số người tối đa |
| 9 | status | tinyint |  | Trạng thái |
| 10 | codeRoom | varchar | 50 | Mã phòng |

* + 1. **User**

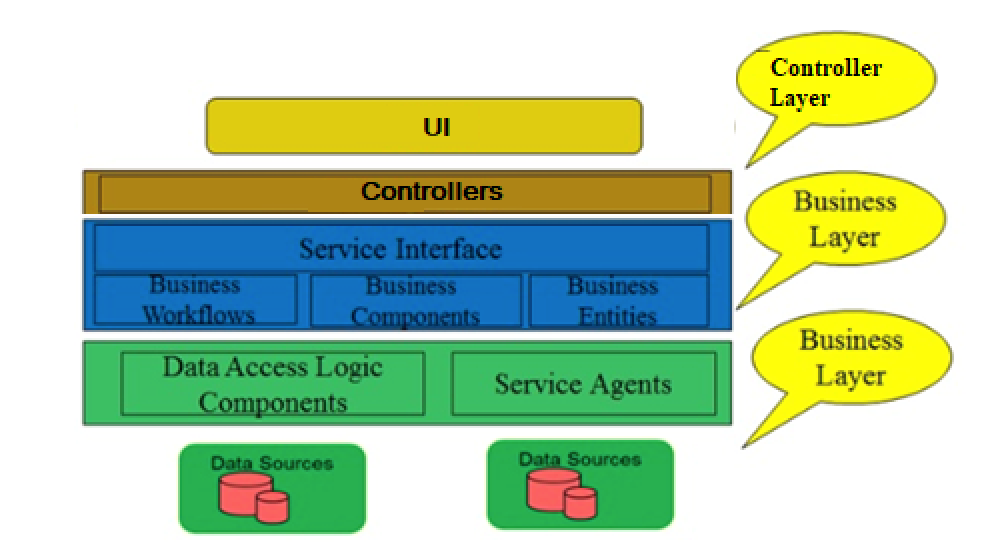
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | id | int | Not null | Khóa chính, tự động tăng |
| 2 | idOwner | int |  | Mã chủ nhà |
| 3 | userName | varchar | 50 | Tên tài khoản |
| 4 | pass | varchar | 255 | Mật khẩu |
| 5 | fullName | varchar | 50 | Tên khách hàng |
| 6 | identifyFront | varchar | 300 | Mặt trước CMND |
| 7 | identifyBack | varchar | 300 | Mặt sau CMND |
| 8 | dateBirth | date |  | Ngày sinh |
| 9 | address | varchar | 255 | Địa chỉ |
| 10 | role | int |  | Quyền |
| 11 | sex | varchar | 10 | Giới tính |
| 12 | job | varchar | 100 | Công việc hiện tại |
| 13 | workPlace | varchar | 255 | Nơi làm việc |
| 14 | tempReg | int |  | Đăng ký tạm trú |
| 15 | province | varchar | 255 | Tỉnh |
| 16 | email | varchar | 255 | email |
| 17 | avatar | varchar | 300 | ảnh đại diện |
| 18 | phoneNumber | varchar | 20 | Số điện thoại |

* + 1. **UserRoom**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu** | **Ràng buộc** | **Ý nghĩa/ghi chú** |
| 1 | idUser | int | Not null | Khóa chính, tự động tăng |
| 2 | idRoom | int | Not null | Khóa chính, tự động tăng |

## Thiết kế kiến trúc.

* 1. **Mô hình kiến trúc hệ thống.**



**Mô hình** gồm 4 phần chính:

* *Controller (C):*

Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view. Từ đó, C đưa ra dữ liệu phù hợp với người dùng. Bên cạnh đó, Controller còn có chức năng kết nối với model.

* ***Model (M):***

Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là một cầu nối giữa 2 thành phần bên dưới là View và Controller. Model thể hiện dưới hình thức là một cơ sở dữ liệu hoặc có khi chỉ đơn giản là một [file XML](https://monamedia.co/file-xml-la-gi/) bình thường. Model thể hiện rõ các thao tác với cơ sở dữ liệu như cho phép xem, truy xuất, xử lý dữ liệu,…

* *Business Logic Layer (BLL):* Lớp này phân ra thực hiện 2 nhiệm vụ:

+ Đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu được truyền xuống từ GUI trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị cơ sở dữ liệu.

+ Kiểm tra ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về GUI layer.

* *Data Access Layer (DAL):* Lớp này có chức năng giao tiếp với hệ quản trị cơ sở dữ liệu như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa …)
  1. **Mô tả chi tiết thành phần trong hệ thống.**
     1. **Business Logic Layer (BLL)**

Lớp này gồm 4 phần:

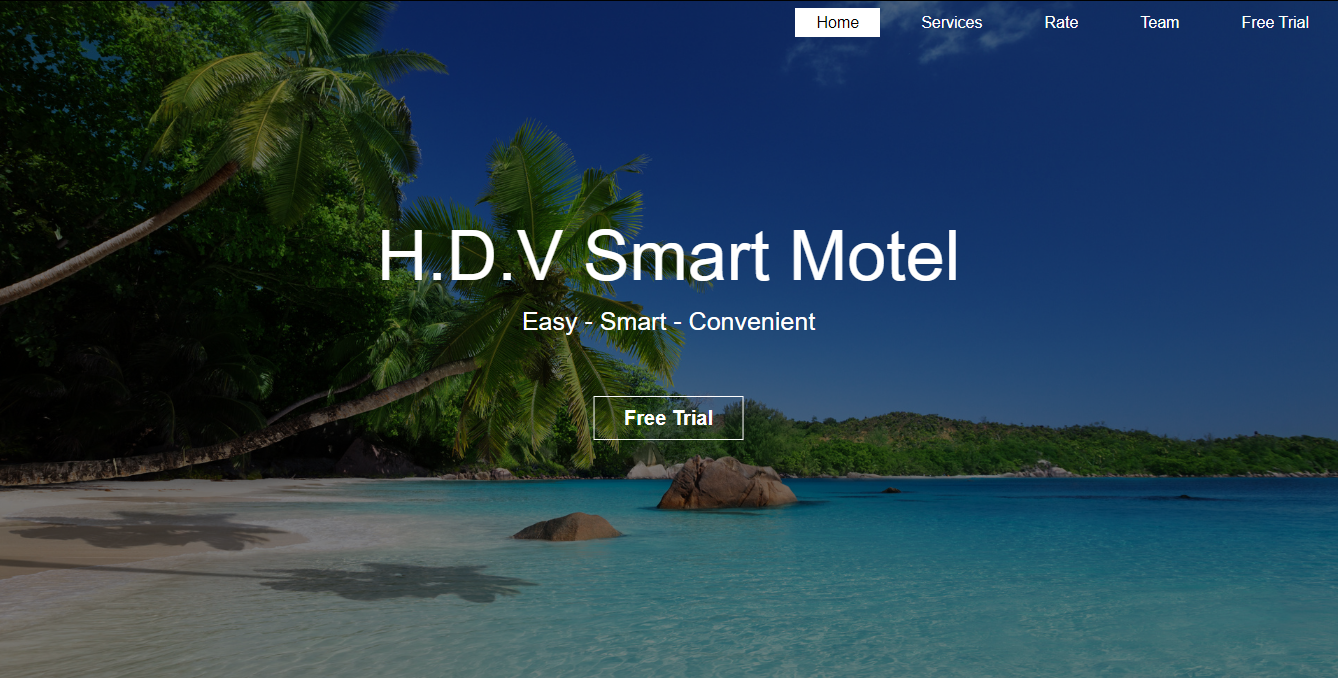
* *Service Interface:* Là thành phần giao diện lập trình mà lớp này cung cấp cho lớp Controller sử dụng.
* *Business Workflows:* Chịu trách nhiệm xác định và điều phối các quy trình nghiệp vụ gồm nhiều bước và kéo dài. Những quy trình này phải được sắp xếp thực hiện theo một thứ tự chính xác.
* *Business Components:* Chịu trách nhiệm kiểm tra các quy tắc nghiệp vụ, ràng buộc logic và thực hiện các công việc. Các thành phần nàycũng thực hiện các dịch vụ mà Service Interface cung cấp Service Workflows sẽ sử dụng nó.
* *Business Entities:* thường được sử dụng như Data Tranfer Object (DTO). Có thể dùng truyền dữ liệu giữa các lớp Controller và DAL.
  + 1. **Data Access Layer (DAL)**
* *Data Access Logic Components:* Chịu trách nhiệm lưu trữ và truy xuất dữ liệu từ các nguồn Data Sources.
* *Services Agents:* Giúp gọi và tương tác với các dịch vụ từ bên ngoài 1 cách dễ dàng và đơn giản.

## Thiết kế giao diện

* 1. **Danh sách màn hình.**

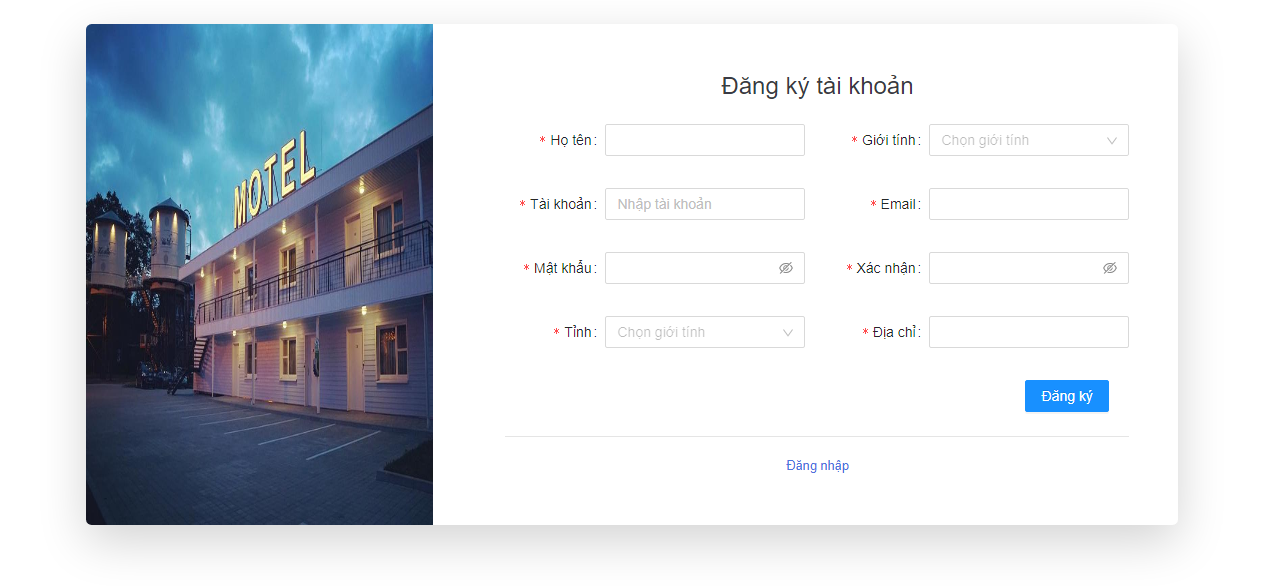
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên màn hình** | **Ý nghĩa/Ghi chú** |
| 1 | Index | Bắt đầu |
| 2 | Register | Đăng ký tài khoản mới |
| 3 | Login | Đăng nhập |
| 4 | DashBoard | Quản lý chung |
| 5 | Block | Quản lý khu trọ |
| 6 | Room | Quản lý phòng trọ |
| 7 | Service | Quản lý dịch vụ |
| 8 | Customer | Quản lý khách hàng |
| 9 | Contract | Quản lý hợp đồng |
| 10 | Bill | Quản lý hóa đơn |

* 1. **Mô tả chi tiết các màn hình.**
     1. **Index.**

****

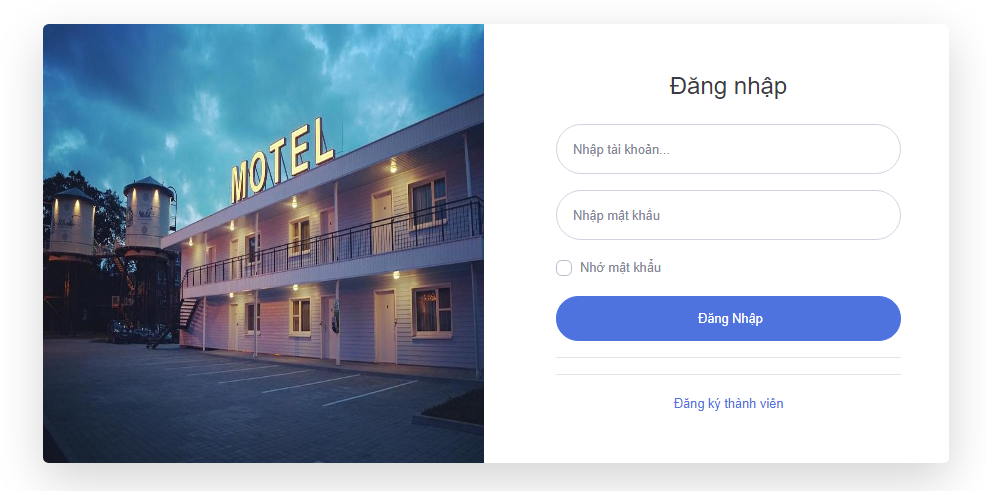
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Nhấn FreeTrial | Chuyển đến màn hình đăng nhập |

* + 1. **Register.**

****

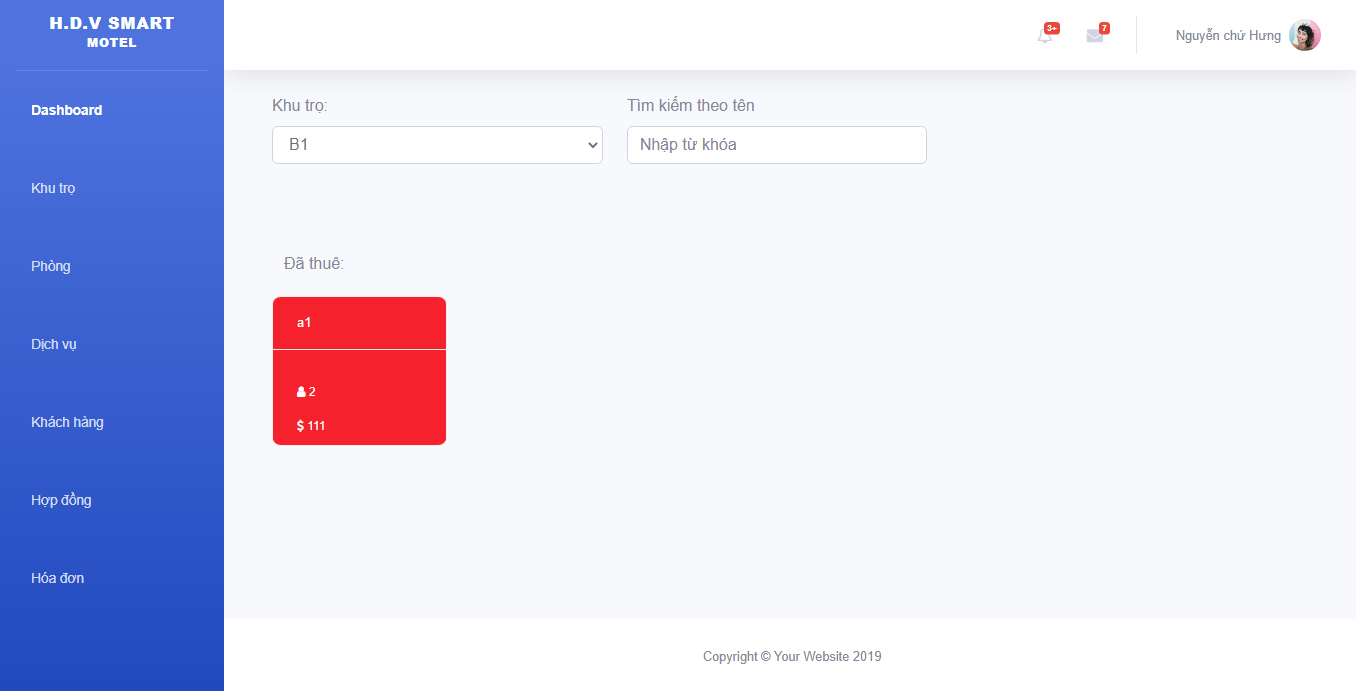
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Nhập Họ tên |  |
| 2 | Chọn giới tính |  |
| 3 | Nhập tài khoản |  |
| 4 | Nhập Email |  |
| 5 | Nhập mật khẩu |  |
| 6 | Nhập xác nhận |  |
| 7 | Nhập tỉnh |  |
| 8 | Nhập địa chỉ |  |
| 4 | Bấm nút Đăng ký | Thực hiện đăng ký |
| 5 | Bấm nút đăng nhập | Chuyển qua màn hình đăng nhập |

* + 1. **Login.**

****

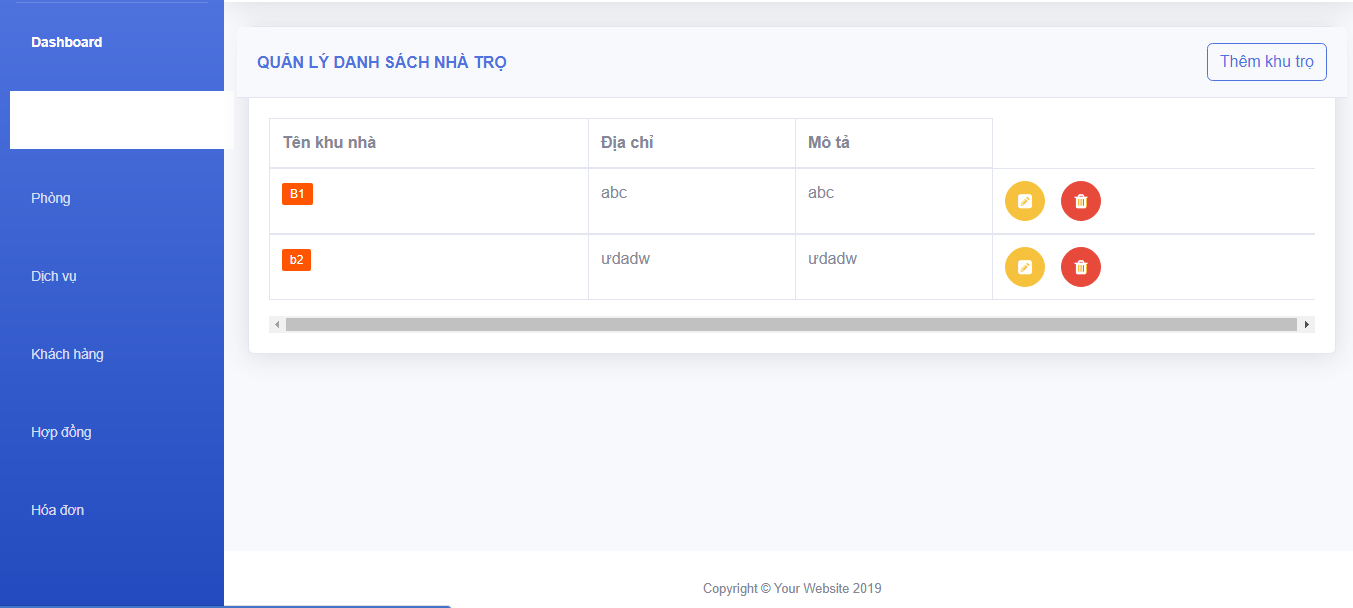
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Nhập tài khoản |  |
| 2 | Nhập mật khẩu |  |
| 3 | Chọn nhớ mật khẩu(hoặc không) |  |
| 4 | Bấm đăng nhập | Thực hiện đăng nhập |
| 5 | Bấm Đăng ký thành viên | Chuyển qua màn hình đăng ký |

**DashBoard.**

****

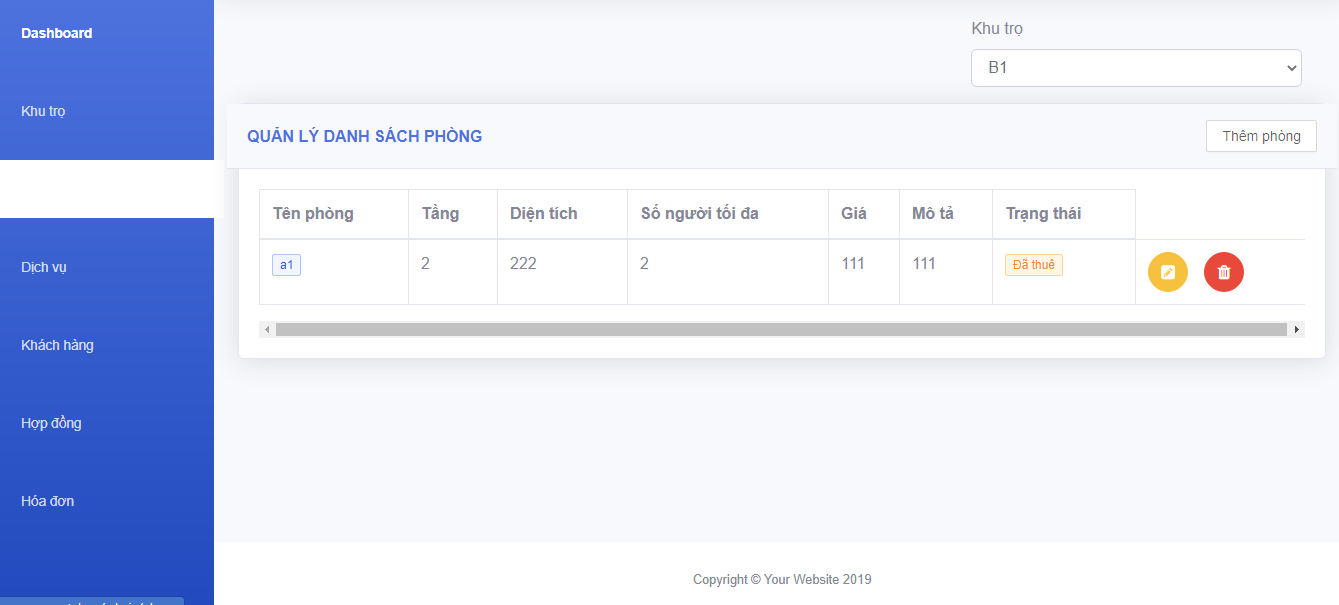
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Chọn khu trọ | Tìm phòng theo khu trọ |
| 2 | Nhập tìm kiếm theo tên | Tìm phòng theo tên |
| 3 | Nhấp vào phòng trọ | Hiện thông tin phòng trọ |

* + 1. **Block.**

****

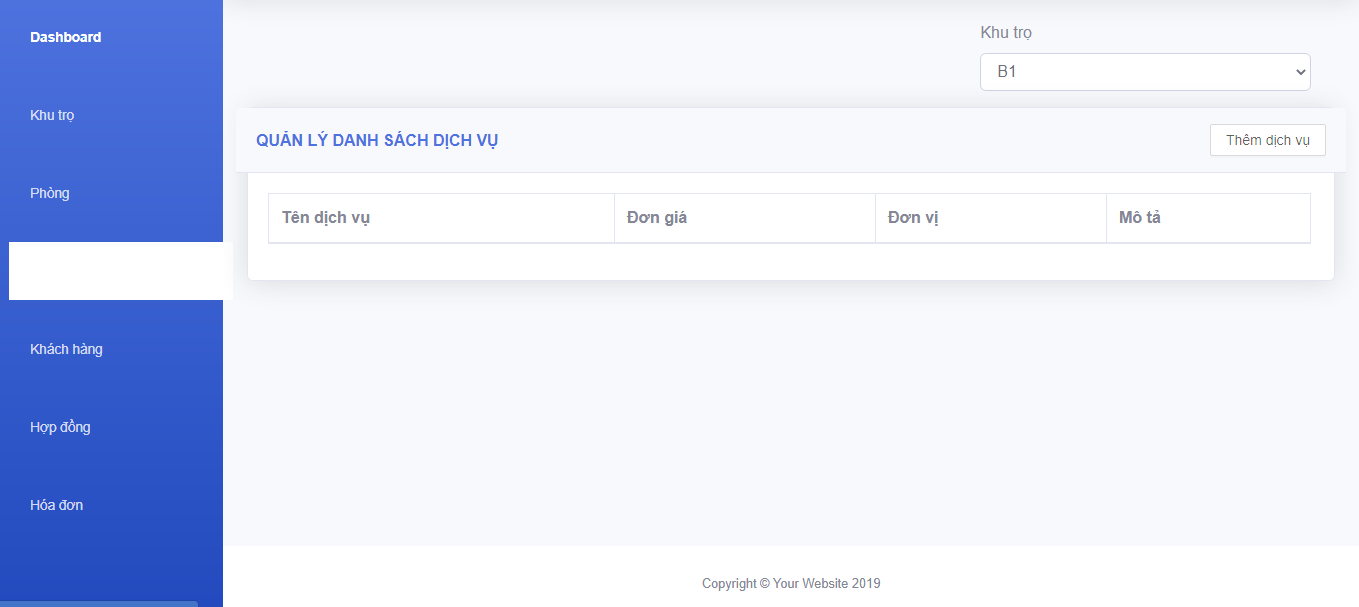
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm khu trọ”, nhập thông tin khu trọ mới. | Thêm khu trọ mới vào danh sách. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa khu trọ đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin khu trọ đã chọn. |

* + 1. **Room.**

****

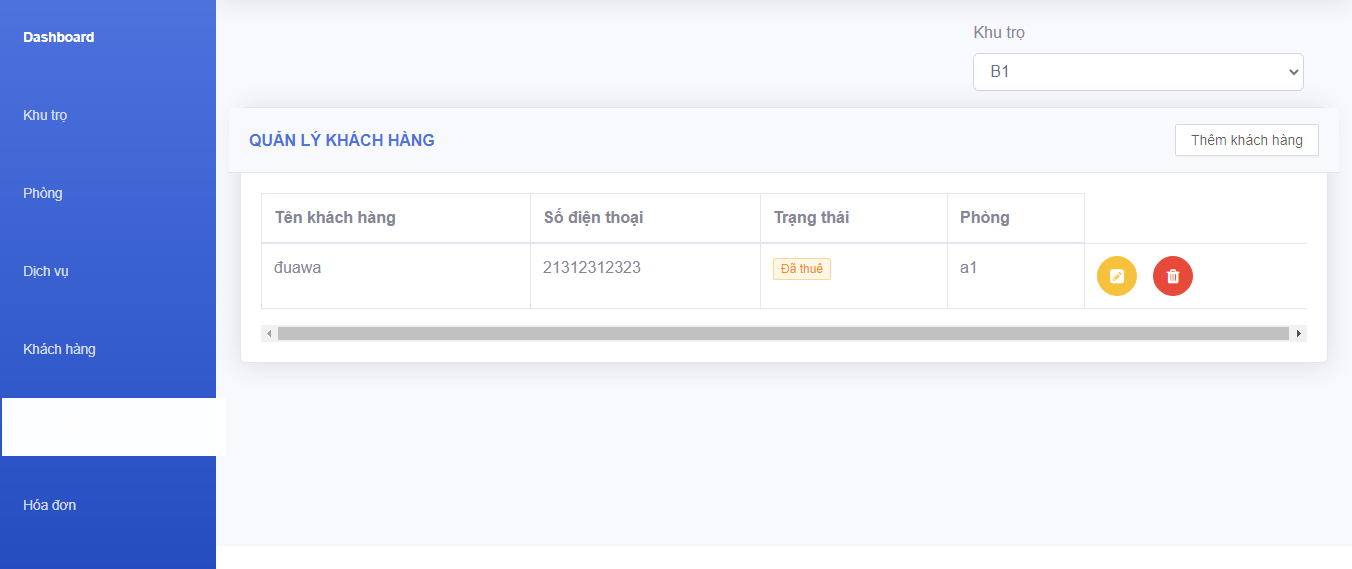
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm phòng”, nhập thông tin phòng trọ mới. | Thêm phòng trọ mới vào danh sách. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa phòng đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin phòng trọ đã chọn. |

* + 1. **Service.**

****

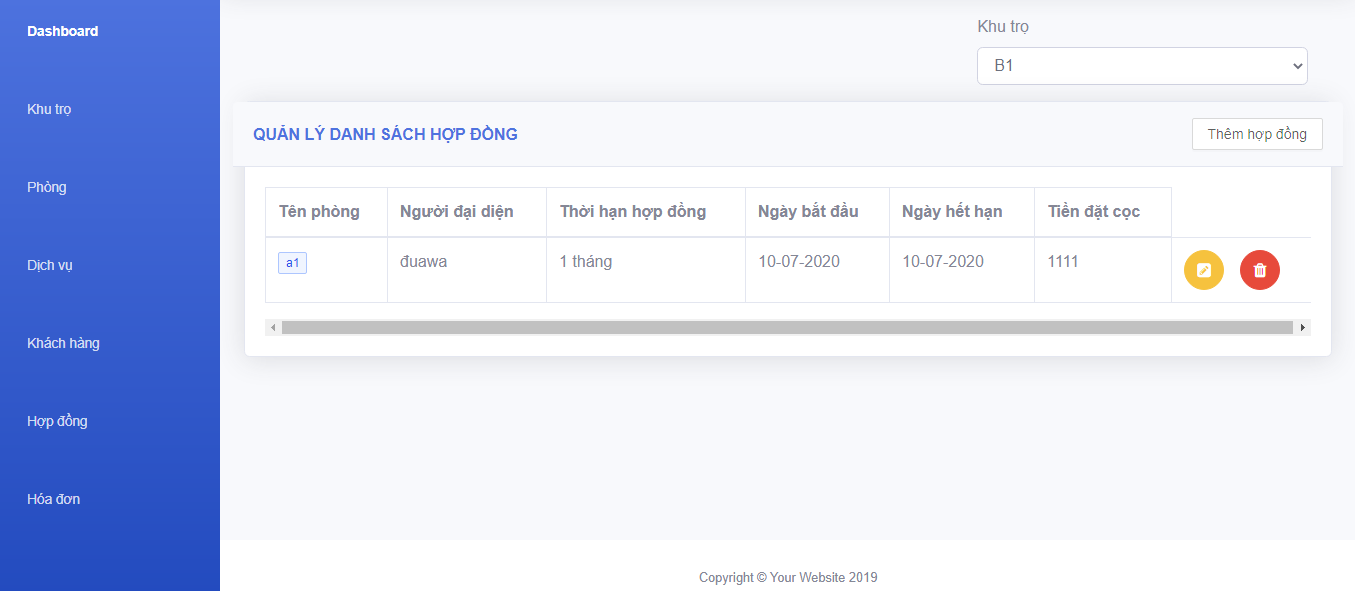
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm dịch vụ”, nhập thông tin dịch vụ mới. | Thêm dịch vụ mới. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa dịch vụ đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin dịch vụ đã chọn. |
| 4 | Chọn khu trọ | Hiện thị hợp đồng theo khu trọ đã chọn |

* + 1. **Customer.**

****

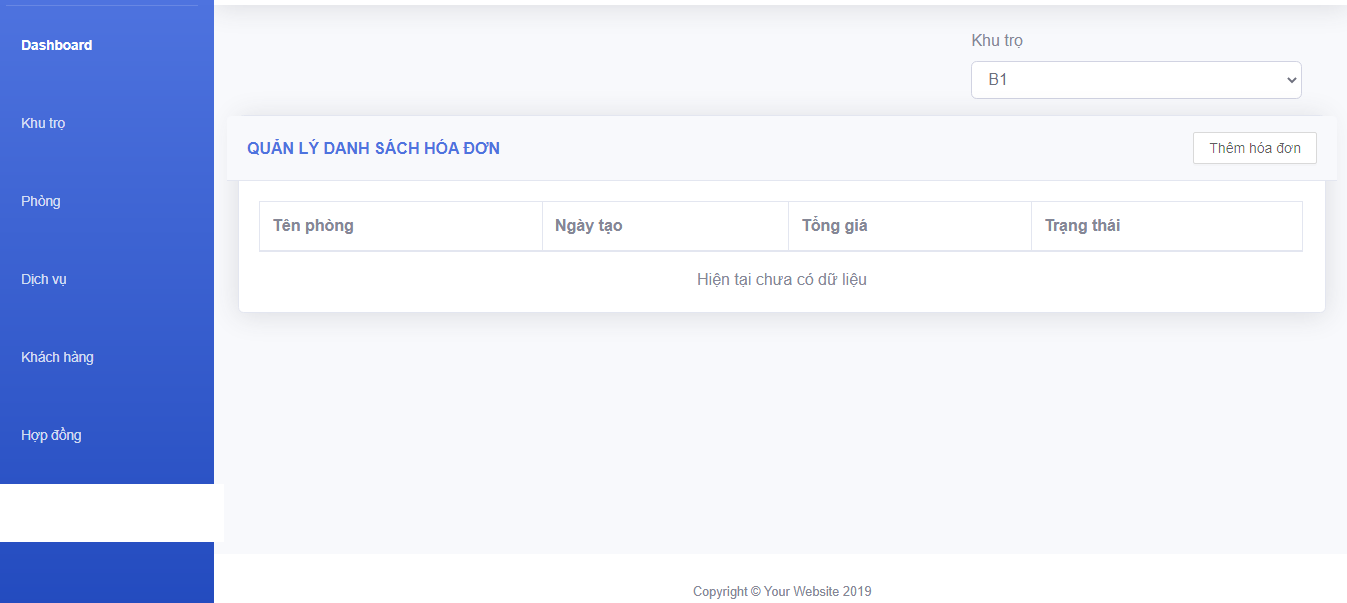
|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm khách hàng”, nhập thông tin khách hàng mới. | Thêm khách hàng mới. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa khách hàng đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin khách hàng đã chọn. |
| 4 | Chọn khu trọ | Hiện thị khách hàng theo khu trọ đã chọn |

* + 1. **Contract.**

****

|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm hợp đồng”, nhập thông tin hợp đồng mới. | Thêm hợp đồng mới. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa hợp đồng đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin hợp đồng đã chọn. |
| 4 | Chọn khu trọ | Hiện thị hợp đồng theo khu trọ đã chọn |

* + 1. **Bill.**

****

|  |  |  |
| --- | --- | --- |
| **STT** | **Cách sử dụng** | **Xử lý** |
| 1 | Bấm nút “Thêm Hóa đơn”, nhập thông tin hợp đồng mới. | Thêm Hóa đơn mới. |
| 2 | Bấm nút “Xóa”(biểu tượng trashbin màu đỏ) và xác nhận | Xóa Hóa đơn đã chọn. |
| 3 | Bấm nút “Sửa”(biểu tượng mà vàng) | Sửa thông tin Hóa đơn đã chọn. |
| 4 | Chọn khu trọ | Hiện thị Hóa đơn theo khu trọ đã chọn |

## Kết luận



### Kết quả đạt được

* Tìm hiểu được cơ bản về ngôn ngữ lập trình Golang
* Hiểu được cơ cấu tổ chức, quy trình và nghiệp vụ của đề tài quản lý nhà trọ.
* Xây dựng thành công phần mềm bằng Golang thỏa mãn các yêu cầu đề ra ban đầu
* Cải thiện khả năng làm việc nhóm, mọi người trong nhóm đều cố gắng hoàn thành tốt phần việc của mình.

### Hạn chế và khó khăn

* Phân công công việc chưa được hợp lý, đôi khi gặp mâu thuẫn trong công việc dẫn đến việc xử lí vấn đề chậm chạp, công việc trì trệ.
* Các phần công việc không đồng bộ nhau nên cần rà soát và chỉnh sửa lại cho mọi cá nhân.
  1. **Hướng phát triển**

Do hạn chế về kinh nghiệm, kiến thức và thời gian nên nhóm chỉ có thể hoàn thành hoàn toàn đề tài ở mức quy mô vừa và nhỏ. Các yêu cầu để ra tuy được hoàn thành nhưng lại chưa xử lý một cách tốt nhất. Đề tài quản lý nhà trọ khá phổ biến và có tính áp dụng cao. Vì thế nếu có thể, nhóm sẽ tập trung vào việc tìm hiểu sâu hơn và phát triển phần mềm ở quy mô lớn hơn.

## Tài liệu tham khảo

* <https://monamedia.co/mvc-la-gi-ung-dung-cua-mo-hinh-mvc-trong-lap-trinh/>
* <https://www.codehub.vn/Golang-Co-Ban/Go-Function>
* <https://topdev.vn/blog/golang-channel-la-gi/>
* <https://techmaster.vn/posts/35098/series-golang-co-ban-phan-21-channel>
* <https://www.codehub.vn/Golang-Co-Ban>
* <https://medium.com/@dinhhuy_67517/golang-l%C3%A0-g%C3%AC-golang-c%C6%A1-b%E1%BA%A3n-c304c4ffcbb3>
* <https://phocode.com/golang/go-lap-trinh-go/>

**HẾT**

------------------------------------------------------------------------------------------------